Progetto Dirigibile A.R.I.A

2005/2006

A cura di:

RUCATTI DANIELE
BOTTESI OMAR

L’idea di realizzare un dirigibile è partita l’anno scorso, durante un week-end trascorso con gli amici.
Già era chiara l’idea che dovevamo trovare qualcosa di “fuori dal comune” da portare all’esame di maturità, ma tra le tante idee non avevamo ancora scelto quella definitiva.
Quel giorno, dal finestrino dell’autobus vedemmo un modellino di dirigibile volare nel cielo e a pochi metri di distanza il suo pilota, con il telecomando in mano.
Quasi per scherzo saltò fuori la domanda: “Perché non costruire un dirigibile?”.
Da quel giorno cominciarono le ipotesi su come tale dirigibile avrebbe dovuto volare e come sarebbe stato pilotato.
Il nostro dirigibile non doveva essere però solo un pallone con dei motori, doveva avere qualcosa in più: poteva avere dei sensori per interagire con l’ambiente, poteva avere una telecamera direzionabile per effettuare riprese dall’alto, poteva essere comandato a distanza, poteva essere una completa stazione meteo oppure un semplice mezzo pubblicitario.
Tante idee fantastiche, ma nessun punto di partenza per realizzarle.
Comincia la ricerca. Settimane spese su internet per cercare materiale su dirigibili, elettronica, robotica e mille altre cose.
Lentamente tutti i tasselli hanno cominciarono a comporsi e riuscimmo a delineare un percorso da seguire per portare a termine il nostro obiettivo: costruire un robot del cielo.
Un progetto così ambizioso richiede tempo. Ci sono stati molti imprevisti, parte dei quali collettivamente attribuiti alla sfortuna.
Tuttavia siamo arrivati a portare a termine gran parte del lavoro, da cui abbiamo ricavato conoscenze nel campo della robotica, dell’automazione, dell’elettronica, dell’informatica e anche della chimica e della fisica.
Con questa tesina si vogliono elencare le caratteristiche principali del progetto e vedere in dettaglio i componenti che lo compongono, analizzando le scelte effettuate, i problemi riscontrati e le soluzioni adottate per risolverli.
Ci sarebbe da scrivere molto, ma non ne abbiamo il tempo. Tutte le informazioni omesse da questo scritto possono essere reperite nei documenti citati nella bibliografia oppure su internet.
Ringraziamo tutti quelli che in qualche modo hanno contribuito, nel loro piccolo, a sostenerci ed aiutarci durante questa epica impresa.

INDICE CONTENUTI

- **SCHEDINE DERIVAZIONI**
- **MINIBASE**
- **ESPANSIONI**

- **SOFTWARE**
  - **PROGRAMMI USATI**
  - **PICBASIC**
  - **PROGRAMMI PIC**
    - ⇒ **STRUTTURA A MODULI**
      - **MODULO CONTROLLO MOTORI CON PWM**
      - **MODULO CONTROLLO JOYPAD PLAYSTATION**
      - **MODULO COMUNICAZIONE I2C**
      - **MODULO SERVI R/C**
      - **MODULO SENSORI ULTRASUONI**
      - **MODULO SONAR**
      - **MODULO COMUNICAZIONE RADIO**
    - ⇒ **PROGRAMMI PRINCIPALI**
      - **PROGRAMMA CORE**
      - **PROGRAMMA FUFFY**
      - **PROGRAMMA PICMASTER**
      - **PROGRAMMA PIC PSX (JOYPAD)**
      - **PROGRAMMA PIC GESTORE COMUNICAZIONE DISPOSITIVI**
      - **PROGRAMMA MODULO CICLOPÈ**
  - **PROGRAMMI PC**
    - ⇒ **ARIA GUI – INTERFACCIA GRAFICA**
      - **FORM COMMAND**
      - **FORM AIRSHIP DATA**
      - **FORM EARTH BASE DATA**
      - **MODULO COMMSET (DIVISIONE PACCHETTI)**
      - **MODULO CONVERSIONE BINARIA (BINARY)**
      - **MODULO BUSSOLA**

- **FASE 3 – MESSA IN SERVIZIO**
  - **PREPARAZIONE AL PRIMO TEST SUL CAMPO**
  - **IL VOLO**

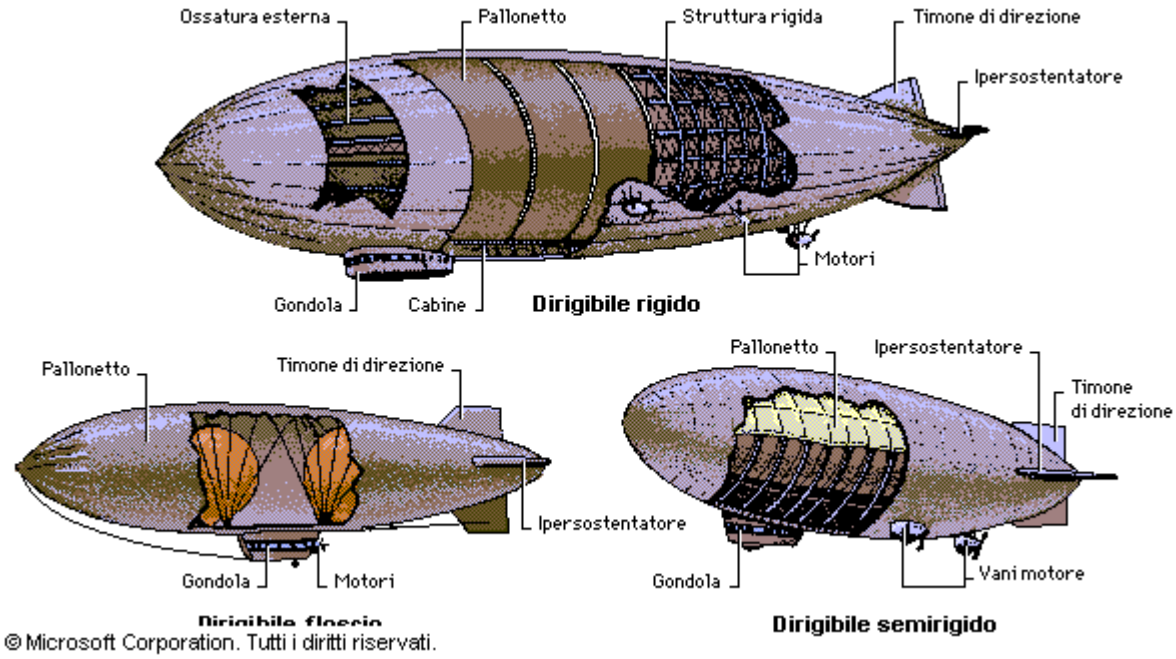
- **CONCLUSIONI**

FASE 1

RICERCHE PRELIMINARI

Le prime ricerche fatte riguardano il pallone. In internet abbiamo trovato informazioni sulle modalità costruttive dei dirigibili.
In sostanza esistono tre tipi di dirigibile:

- Quello Rigido ha una struttura interna rigida che sostiene delle celle contenenti il gas. Tali celle vengono sgonfiate per far scendere il velivolo e gonfiate per farlo salire.
- Quello Semirigido ha una struttura di sostegno per il pallone, il quale è riempito di gas per intero.
- Quello Floscio non ha strutture, e la forma del pallone viene conferite dalla pressione che il gas esercita dall'interno.



L'idea all'inizio era di costruire un dirigibile Floscio, ma analizzandone bene le caratteristiche abbiamo deciso di costruire un modello ibrido situato a metà strada tra il tipo floscio e quello Semirigido.
Il passo successivo è la scelta del gas. Nell'era dei dirigibili veniva usato l'idrogeno, gas facilmente infiammabile ed altamente esplosivo, a causa dell'elevato costo e della scarsa reperibilità dell'elio.
Date le testimonianze del disastro dell'Hindemburg, abbiamo deciso di evitare il contatto con il sopraccitato gas e passare a qualcosa di innocuo.
L'elio, gas nobile, inodore, insapore e non infiammabile.
Usato dai clown per fare la voce di paperino, usato per il contenimento di esplosioni, usato per i palloncini era il gas giusto per noi.

Abbiamo effettuato alcuni calcoli per trovare la spinta ascensionale dell'elio, dell'idrogeno e dell'aria calda valutandone le differenze, così da poter dimensionare a grandi linee il pallone.
Per quanto riguarda l'elio, sappiamo che, a 15°C e a 1 atm, ha una densità di 0.1638 g/l. facendo riferimento al principio di Archimede (ovvero un corpo immerso in un fluido riceve una spinta verso l'alto pari il peso del volume di fluido spostato), abbiamo trovato la capacità ascensionale che risulta essere:

$$1.1861 - 0.1638 = 1.0223\ \mathrm{g}/l$$

Dove la densità dell'aria è pari a 1.1861 g/l
In poche parole un litro di gas è in grado di sollevare circa un grammo.

ELIO

caratteristiche chimico – fisiche

massa molare:	**4.0026 g**
temperatura di ebollizione: (a 101,330 kPa)	**- 268.94 °C**
temperatura critica:	**- 267.95 °C**
pressione critica:	**227.5 kPa**
massa volumica del gas: (a 15 °C e 98,067 kPa)	**0.1638 g/l**
massa volumica relativa: (aria = 1)	**0.138**
capacità ascensionale: (a 20 °C e 100 kPa)	**1.02 g/l**
1 litro di liquido:	**763.1 litri di gas**

Per quanto riguarda l'aria, sappiamo che la densità dell'aria è 1.1861 g/l a 20°C e a 1atm e che il volume dei gas è regolato dalle leggi di Gay Lussac e Boyle

$$V(\theta) = V_0 \ (1 + \alpha\theta)$$

dove α è la costante che descrive come varia il volume dei gas a seconda della temperatura e vale 1/273 a 0°C. Tale formula è valida soltanto per i gas ideali, ma noi abbiamo eseguito i calcoli senza preoccuparci della composizione dell'aria.
Sappiamo che il volume è

$$V = m/d$$

Dove m è la massa e d la densità
di conseguenza sostituendola nella formula sopra troviamo che densità e temperatura sono inversamente proporzionali e ricaviamo per esempio che:
L'aria calda a 90°C è in grado di sollevare (1.1861 – 0.903) ≈ 0.3 g/l
Per quanto riguarda l'idrogeno, è il gas maggiormente presente in natura ed più leggero in assoluto, infatti la sua densità è circa 0,0899 g/l (molecola biatomica).
Per tanto ha una capacità ascensionale in aria a 20°C a 1atm di 1,136 g/l.
Il vantaggio ascensionale rispetto all'elio è di solo 0,1137 g/l ma con il vantaggio che pesa molto meno, e di questo se ne deve tenere conto.

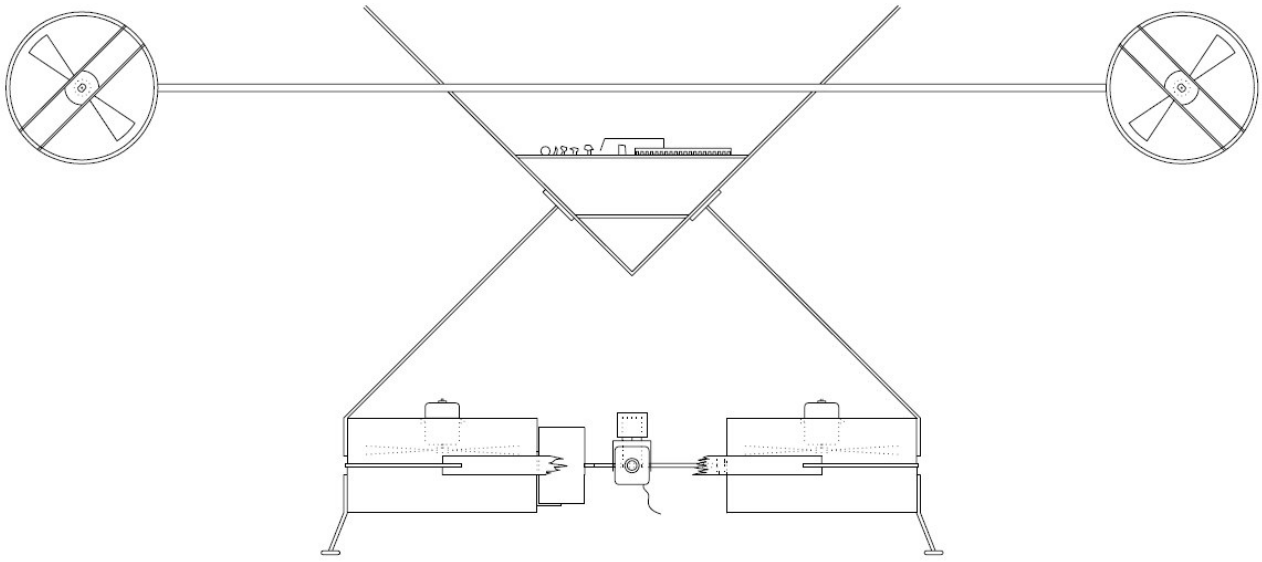
Ora è arrivato il momento di scegliere come fare il pallone.
Prima di tutto la forma e le dimensioni. Abbiamo stimato un peso complessivo dei componenti di 1,5 Kg, quindi il pallone deve poter sollevare tale peso oltre al peso del pallone stesso. Tenendo un margine di zavorra dovevamo costruire un recipiente dal volume di almeno 2,5m3.
Per la forma non potevamo evitare il classico "siluro". Quindi il pallone doveva avere una forma ovoidale allungata.
Il problema era il materiale. Per la costruzione di palloni e palloncini vengono usati diversi materiali:

- Lattice e vinile vengono usati soprattutto per grandi palloni pubblicitari.
- Teflon metallizzato e Mylar per palloncini da festa.

Il lattice ed il vinile pesano molto, quindi richiedono un pallone di dimensioni gigantesche.
Il Mylar è l'ideale. Leggero ed abbastanza resistente viene usato non solo per i palloncini, ma anche nel campo della fotografia, come dielettrico nei condensatori, per le tute spaziali ed altre applicazioni.
Prodotto dalla Dupont, è il materiale che riesce contenere meglio l'elio con un rapporto prezzo/prestazioni calibrato.
Il mylar non è facilmente reperibile sul mercato, ne fanno uso soprattutto grandi aziende e abbiamo avuto non poche difficoltà a reperirlo.
Dopo numerose prove con diversi materiali alternativi, come ad esempio il supermonokote (termoretraibile) e le coperte termiche di emergenza (poliestere alluminato), il mylar è stata la scelta definitiva.
Il materiale si salda a caldo oppure si incolla con appositi collanti.
Calcolando il peso complessivo del mylar e sommando zavorra e componenti il pallone doveva essere di circa 3m3.
Non è facile calcolare il volume di solidi come il nostro, quindi abbiamo realizzato un programma in Visual Basic che suddivide l'ovoide in tre solidi basilari: un cilindro centrale, due tronchi di cono e due semisfere alle estremità. Con questo programma abbiamo potuto analizzare le diverse vie costruttive del pallone velocemente.
La soluzione migliore è costruire il pallone a spicchi longitudinali, chiudendolo alle due estremità (con una cupola).
Per il gonfiaggio ed il rabbocco dell'elio è prevista una valvola.
Una struttura di sostegno per sensori e componenti, realizzata in balsa, deve essere fissata al pallone. Dei ganci incollati al mylar possono bastare per attaccare la struttura al pallone con un pezzo di spago.

Le ricerche successive riguardano l'elettronica e la struttura base della gondola del dirigibile.
La gondola contiene tutte le schede elettroniche del dirigibile. Il cuore è composto da alcuni microcontrollori PIC interconnessi tramite un bus.
Assieme ai PIC ci sono anche una memoria ram ed una memoria eeprom.
Il compito della scheda è di controllare quattro motori e leggere i sensori.
Le memorie sono usate per immagazzinare i dati volatili e non.
La forma della gondola è triangolare, perché dal punto di vista aerodinamico è meglio. Il sostegno centrale si sceglie di farlo in balsa perché è un materiale molto leggero. Sotto di esso vengono attaccati due motori per il movimento ascensionale del velivolo.

Altri due motori vengono messi uno per lato (orizzontalmente, al livello del sostegno centrale) per i movimenti direzionali.
Tutti i motori sono intubati per una migliore resa, ottenuta canalizzando l'aria. I tubi devono essere realizzati con un materiale leggero ma resistente (plastica o altro).
Il dirigibile sarà equipaggiato di sensori ad ultrasuoni per il rilevamento di ostacoli su media distanza e di infrarossi per rilevare il soffitto (sensori superiori).



L'idea principale è quella di realizzare un velivolo che può sia autoguidarsi nello spazio che essere radiocontrollato.
Una bussola elettronica è quindi d'obbligo per la prima funzione.
Per la seconda funzione si possono usare moduli radio già pronti all'uso per comunicare con la base terrestre (per esempio i moduli aurel).
Tali moduli lavorano sulla frequenza di 433Mhz, frequenza gratuita e libera per ogni utilizzo.
Il circuito del dirigibile deve poter misurare anche il livello delle batterie e del segnale radio (RSSI) per poter avvertire il pilota in caso di problemi o malfunzionamenti.
Si prevedono inoltre degli slot di espansione per future schedine ausiliarie e sensori extra, come per esempio sensori meteo, sensori di gas, sensori chimici…
Una di queste espansioni è già occupata da un modulo che sarà dedicato alla manovra di una minitelecamera wireless a colori.

La telecamera si può muovere sui due assi e trasmette i dati a corto raggio sulla banda UHF.
Nella base terrestre vi sarà un circuito composto da più microcontrollori PIC dedicato alla trasmissione dei comandi al dirigibile ed alla ricezione della telemetria, cioè i dati dei sensori e gli stati di funzionamento del velivolo.
Per comandare i movimenti dei motori, della telecamera e di tutte le funzioni principali si usa il joypad della PlayStation.
Questo GamePad è molto maneggevole, presenta due leve analogiche utili per il pilotaggio dei motori e numerosi tasti funzione. Per non parlare della vibrazione integrata che può essere un mezzo di segnalazione oltre che un ottimo "effetto speciale".
La base può essere inoltre interfacciata al pc, o ad un qualsiasi dispositivo compatibile RS232 grazie allo standard seriale.
Il dirigibile può quindi essere pilotato anche dal computer e sul monitor si potranno vedere i valori dei sensori (previo opportuno programma di interfaccia).
Per l'alimentazione si devono usare celle al litio ricaricabili per il dirigibile (ha bisogno di energia), mentre per la base terrestre bastano le normali batterie da supermercato.

Alcuni circuiti sono stati testati utilizzando i microcontrollori PICAXE, ma noi usiamo i PIC, programmabili con un linguaggio differente.
I PIC infatti sono programmabili in assembly (basso livello), mentre gli altri in Basic (alto livello).
Successivamente è stata trovata una soluzione:
Attraverso un compilatore si può convertire un programma scritto in un linguaggio di alto livello in uno di basso livello.
Per i PIC esistono compilatori in C, Java, Basic… Molti compilatori sono gratuiti, ma ancora in via di sviluppo. Per realizzare un programma complesso come il nostro ci serviva un compilatore completo e potente, con un linguaggio semplice e maneggevole come quello dei picaxe.
La soluzione è stata trovata in "PicBasic" un compilatore in Basic, tra i migliori disponibili e con un set di istruzioni adatte ad ogni più disparata applicazione.
I programmi vengono quindi realizzati in basic, compilati ottenendo il codice in assembly che viene assemblato con l'MPASM della Microchip,

ottenendo così il file binario con il codice operativo da caricare direttamente nella memoria programma del PIC.
Per programmare il pic si deve trasferire il programma dal pc attraverso un'apposita interfaccia hardware (il programmatore), che può essere seriale o parallela.
Noi usiamo un programmatore universale di tipo JDM autocostruito, compatibile con la maggioranza dei PIC.
Per caricare i dati si deve usare anche un software compatibile con il JDM: si può usare ICProg oppure WinPic800 (Entrambi freeware). Quest'ultimo supporta molti più dispositivi del primo, ma non funziona con alcuni computer di ultima generazione.
Il programmatore jdm da noi usato prende l'alimentazione diretta dalla seriale del pc, quindi con un laptop si incontrano errori durante la programmazione.
Per quanto invece riguarda i programmi residenti sul computer che si occupano di comunicare con la base, verranno realizzati in Visual Basic perché esso è un linguaggio di programmazione adatto alla creazione di interfacce grafiche in ambiente windows. In poche parole ci semplifica la vita, ma nulla vieta la creazione di programmi compatibili con altri linguaggi come il C. L'importante è la comunicazione attraverso lo standard RS232, seguendo le regole di comunicazione imposte dal pic della base.

FASE 2

PROGETTAZIONE

STRUTTURA

MECCANICA E STRUTTURA VELIVOLO

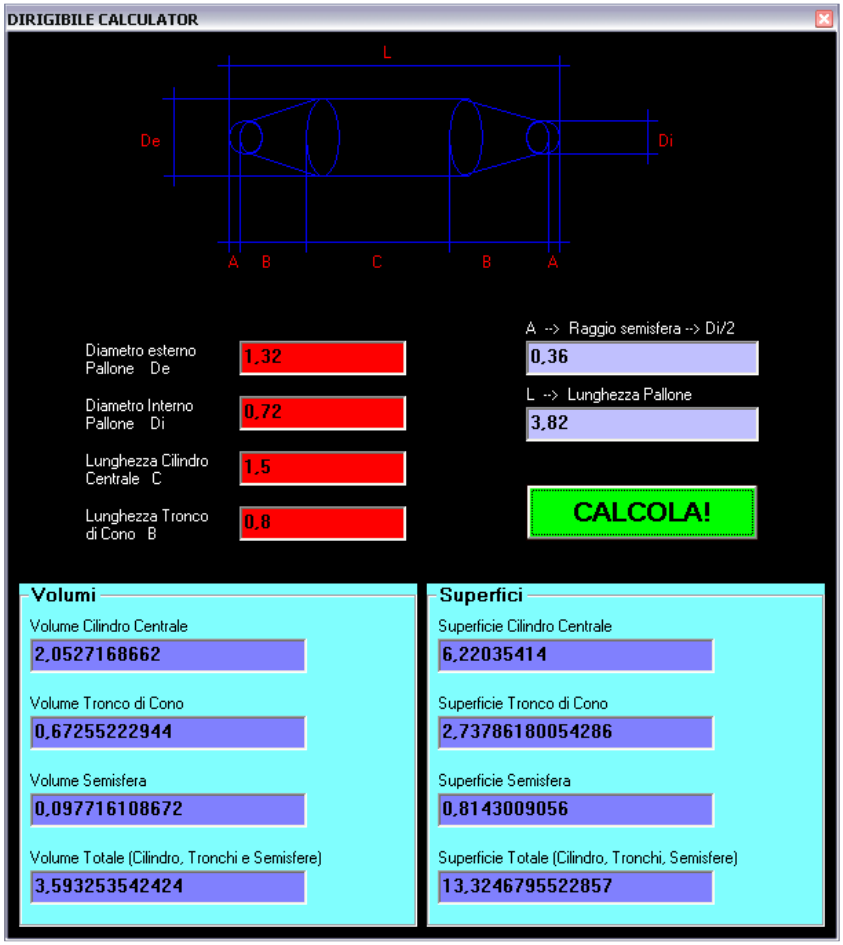
DIMENSIONAMENTO PALLONE:

Abbiamo dapprima stimato il peso che il pallone deve sollevare: 1,5kg, pallone escluso. Includendo anche il peso del pallone e del gas in esso contenuto (anche il gas pesa!) abbiamo stimato un peso massimo sollevabile di 2,5Kg.
Il passo successivo è stato verificare sperimentalmente il peso dei componenti. I dati ricavati sono simili a quelli ipotizzati all'inizio.

DESCRIZIONE	PESO
Supporti Motore	20 x 4 = 40g
Motori	35 x 4 = 140g
Batterie (2 celle) Con Slitta	150g
Vetronite	50g
Componenti Elettronici	100g
Videocamera	50g
Servomotori	30 x 4 = 120g
Sensori ed altri componenti (cavi, connettori…)	50g
Struttura Gondola	350g
Struttura Ciclopè (con motori)	200g
Cupola	100g
Struttura di Sostegno	250g
Coda (tre triangoli complessivi)	300g
Pallone (circa)	1200g
Totale Peso Che il gas Deve Poter Sollevare	2800g
Peso Sollevabile dal Pallone (Teorico) - Peso Pallone Escluso	3600g
Per raggiungere l'equilibrio del dirigibile è stata aggiunta una zavorra	1050g
Il pallone costruito è risultato più grande. Sollevamento Massimo Stimato	4000g circa

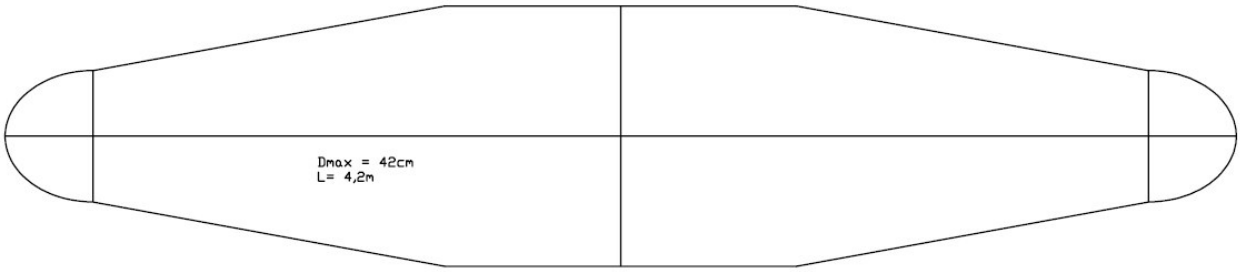
Considerando che l'elio non è mai puro e che la tenuta del materiale non è ottima, abbiamo deciso di sovradimensionare il pallone.
Con 4mc di gas si possono sollevare circa 4Kg, quindi un po' meno del doppio del peso totale da sollevare necessariamente.
Questo permette al pallone di alzarsi senza problemi, grazie all'ampio margine ed inoltre garantisce la possibilità di aggiustare l'equilibrio del pallone nell'atmosfera con accuratezza, usando una zavorra variabile (man mano che il pallone si sgonfia viene tolto peso).
Il materiale usato per la costruzione del pallone è il mylar da 25 micron.

Il materiale ci è stato fornito in metri lineari, quindi strisce di materiale: 16 metri con larghezza 1,5 metri.
È stato costruito un programma in Visual Basic per il calcolo veloce del volume del pallone. Il corpo è stato diviso in solidi elementari: Un cilindro al centro, due tronchi di cono ai lati con termiazioni a semisfera.
Il programma calcola il volume totale e la superficie totale dell'ovoide.
Con tale programma è stato possibile dimensionare la forma del pallone mantenendo il volume costante.



Ottenuto il pallone ideale al nostro scopo, abbiamo calcolato il numero e le dimensioni degli spicchi necessari per costruire un pallone corrispondente ai calcoli fatti.

Per la costruzione dell'ovoide abbiamo tagliato 6 spicchi di lunghezza 4,2m e larghezza massima 72cm.
Abbiamo costruito un modellino in scala del pallone utilizzando della normale carta. Così abbiamo verificato che gli spicchi progettati erano idonei e che il solido ottenuto rispondeva alle caratteristiche richieste.



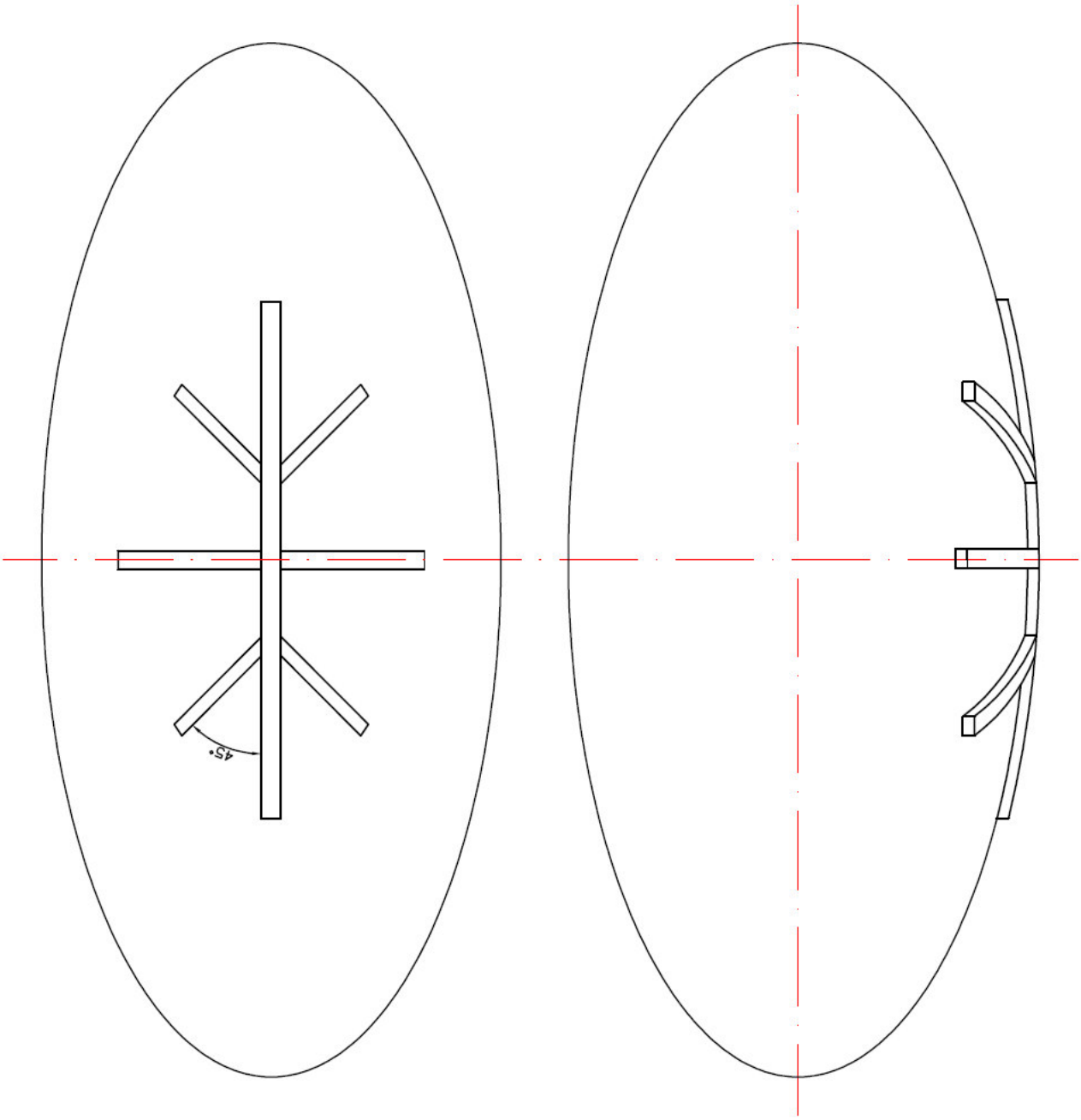
Gli spicchi sono stati saldati a caldo tramite un apposito ferro e le giunzioni sono state incollate con della colla per PVC per evitare il rischio di possibili perdite di gas.
Il pallone ottenuto è di lunghezza 3,8m e diametro 1,32m con un volume di circa 4mc, come da ipotesi fatte.

CODA (ALETTE):

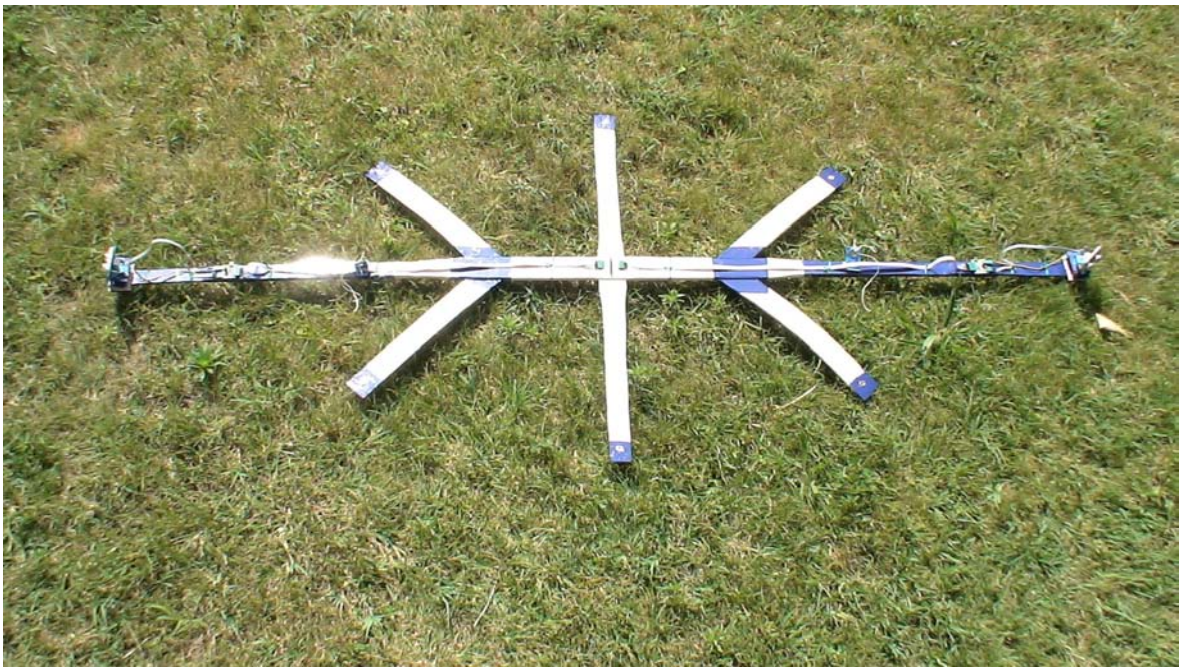
Ogni dirigibile ha sempre delle alette poste sulla coda, servono per conferire aerodinamicità e garantire il corretto pilotaggio. Infatti esse possono essere regolate dalla gondola permettendo di impostare le direzione desiderata.
Sulla coda del pallone abbiamo pensato di fissare tre alette direzionali, soprattutto per una questione di estetica in quanto sarebbe stato molto difficile realizzare l'apparato di controllo.
Sono state realizzate incollando tra di loro degli spezzoni di balsa: mettendo gli spezzoni su di un piano, abbiamo incollato con della colla a caldo le estremità ai lati più lunghi. Abbiamo poi tolto la collante in eccesso e rinforzato le giunte con della colla vinilica. Abbiamo ritagliato le sagome delle pinne che sono state successivamente verniciate di color rosso. L'aletta centrale presenta alla base un rettangolo di balsa che permette di ancorarla al pallone tramite dei gancetti in PVC, inoltre abbiamo effettuato dei fori rinforzati per il fissaggio tramite corde.

STRUTTURA DI SOSTEGNO:

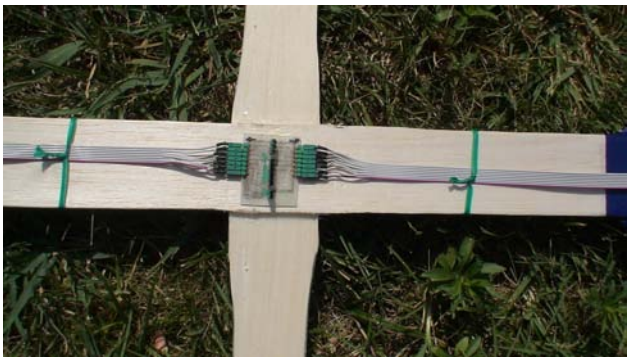
Il materiale usato per la realizzazione del pallone presenta già una certa rigidità, per cui non sarebbe necessario costruire una struttura di supporto, ma essendo, il nostro pallone, un modello ibrido tra uno floscio e uno semirigido, abbiamo comunque realizzato uno scheletro in balsa per il sostegno della gondola e dei sensori di rilevamento ostacoli.



La struttura è lunga circa 1.65m e larga 0.8m, ogni pezzo presenta una larghezza che varia da 2cm a 5 cm. La parte principale è a forma di croce realizzata con tre strati di balsa ultraleggera, inoltre vi sono quattro terminazioni sempre in balsa, poste a circa un quarto della lunghezza totale dalle estremità, e con un angolatura di 45° rispetto alla parte longitudinale. Tali estremità hanno lunghezza di 33cm e larghezza di circa 5cm.
In alcuni punti, la balsa è stata ricoperta con del materiale termoretraibile, incollabile a caldo per mezzo di un ferro da stiro, per conferire resistenza meccanica.



La struttura è stata fissata al pallone attraverso dei ganci per tutta la parte centrale, mentre tutte le parti terminali sono state forate e collegate al pallone attraverso un cordino. I fori sono stati opportunamente rinforzati con delle rondelle in ottone.
Al centro della struttura è collocata la schedina delle derivazioni da cui partono tutti i cavi flat che portano l'alimentazione e i dati alle derivazioni degli altri sensori.

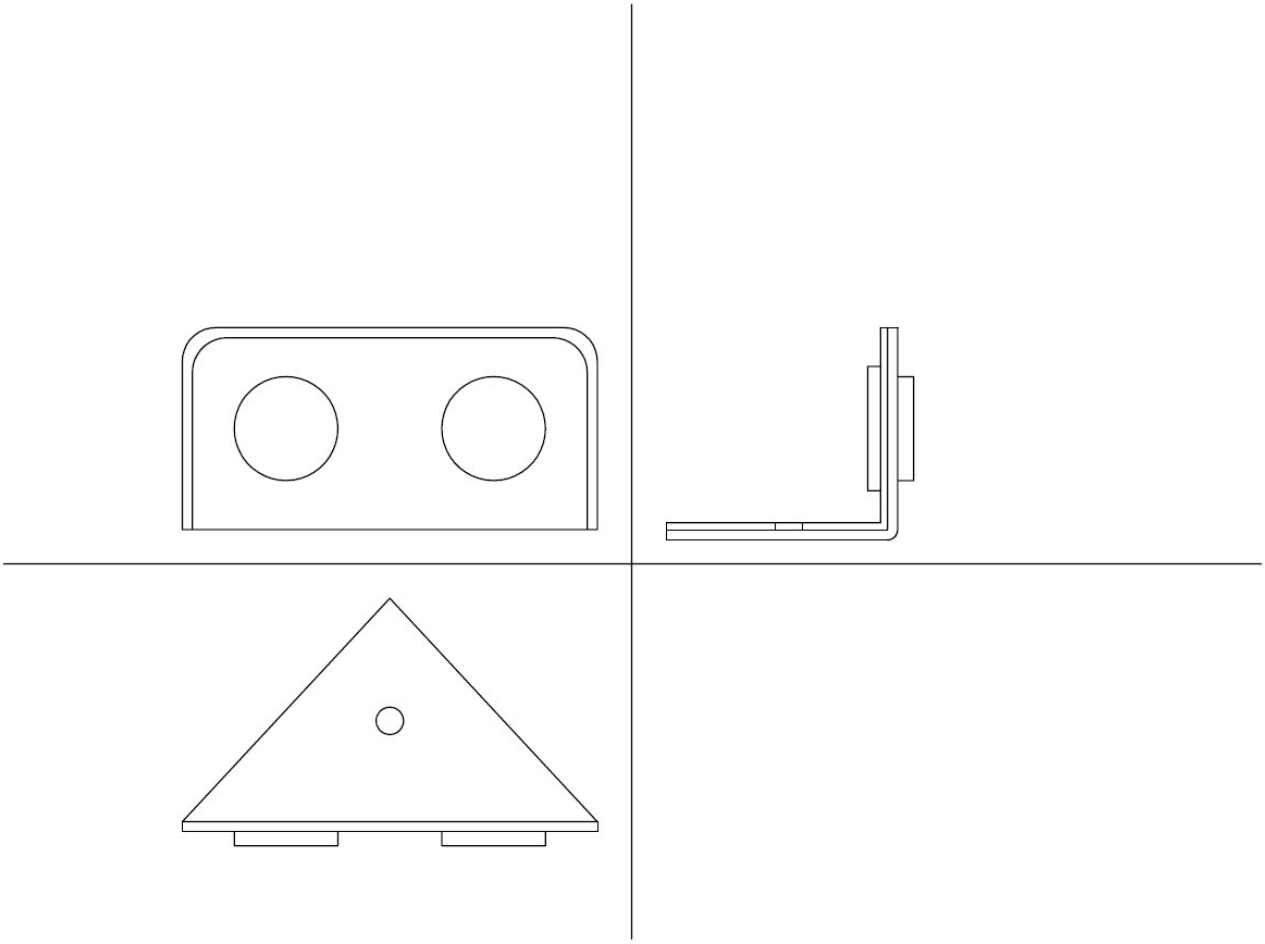


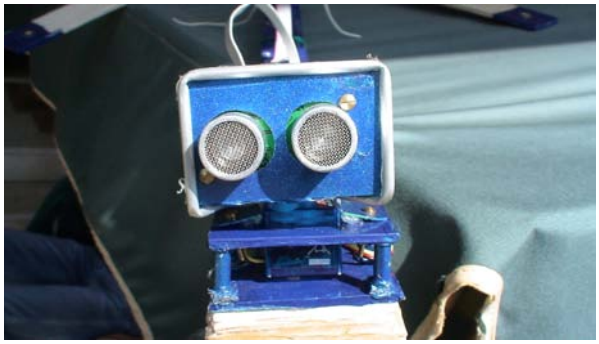
Verso la parti più esterne vi sono le altre due schedine di derivazione che postano l'alimentazione ai sensori infrarossi e agli ultrasuoni.

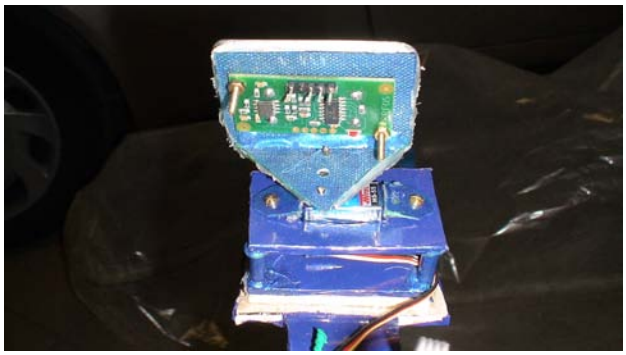
Alle due estremità principali vi sono alloggiati i due sensori ad ultrasuoni con i servomotori (fuffy) inclinati di circa 30° rispetto al piano in modo tale che puntino verso terra. Tali parti sono incollate alle struttura con della colla a caldo.

STRUTTURA MODULO FUFFY:

Lo scopo principale è quello di fissare in modo sicuro il modulo fuffy.
Vi sono due parti principali:
la prima riguarda l'ancoraggio del servomotore ed è costituita da due rettangoli di balsa 3x6cm, di cui uno forato al centro per l'inserimento del servomotore. I due pezzi di balsa sono ricoperti con del materiale termoretraibile e incollati tra di loro attraverso 4 astine in legno di circa 2cm. Il servomotore viene fissato alla struttura tramite dei bulloni da 2mm in ottone.
La seconda parte interessa il fissaggio del modulo ultrasuoni ed è realizzato tramite un supporto rettangolare 5x4cm in vetronite alleggerita con due fori per il posizionamento delle capsule del sensore. Alla base di esso è stato incollato un triangolo, sempre in vetronite, che forma cosi un angolo di 90° che permette il fissaggio, attraverso delle mini viti, della struttura al servomotore. Sono stati aggiunti dei supporti in alluminio per conferire resistenza meccanica.

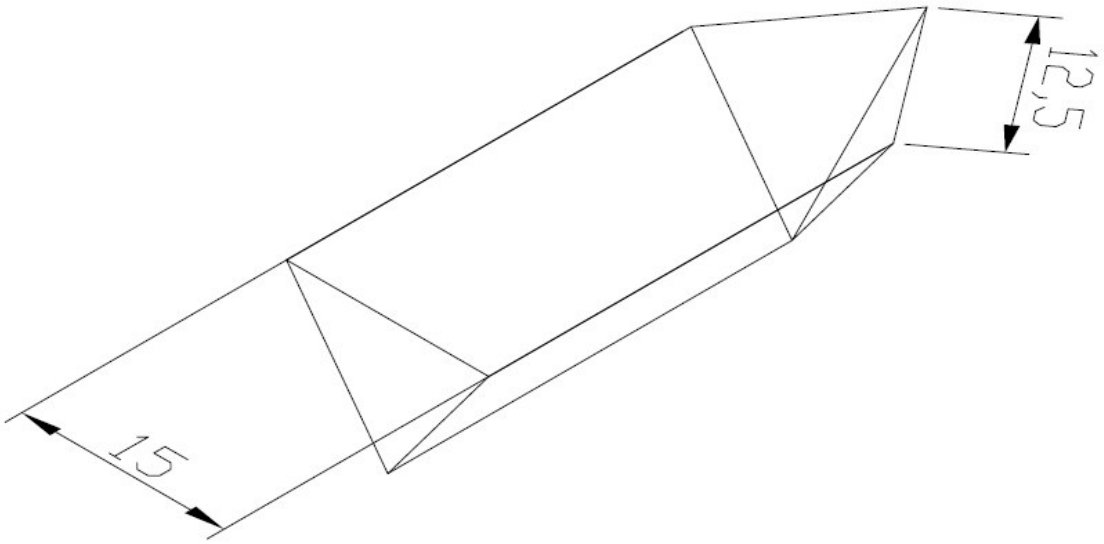




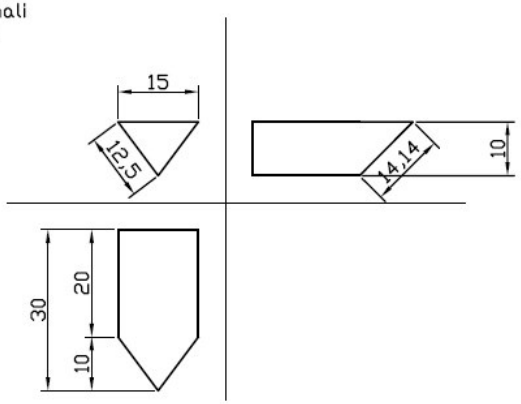


CABINA:

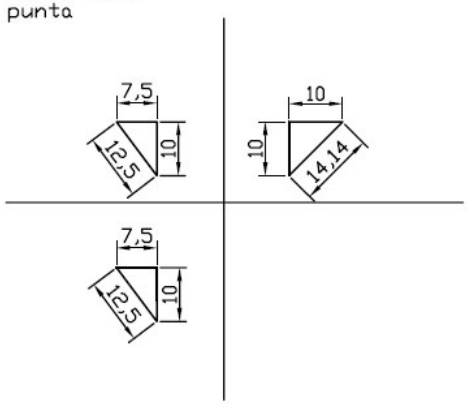
La cabina rappresenta la parte di struttura in cui sono contenuti i circuiti principali.
Ha una forma triangolare e termina nella parte posteriore con uno sportello per l'ispezione e l'inserimento delle batterie, mentre la parte anteriore è a punta per garantire una maggiore aerodinamicità.
Tale struttura è realizzata completamente in balsa, incollata con colla per legno e colla a caldo, e ha le seguenti dimensioni:
lunghezza tot: 40cm
larghezza: 15cm
profondità: 10cm



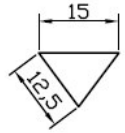
proiezioni
ortogonali
gondola



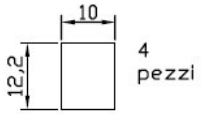
Quote della
punta



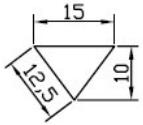
Quote
parte
posteriore



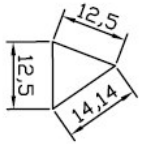
Quota pezzi da tagliare



4
pezzi



1 Pezzo
dietro



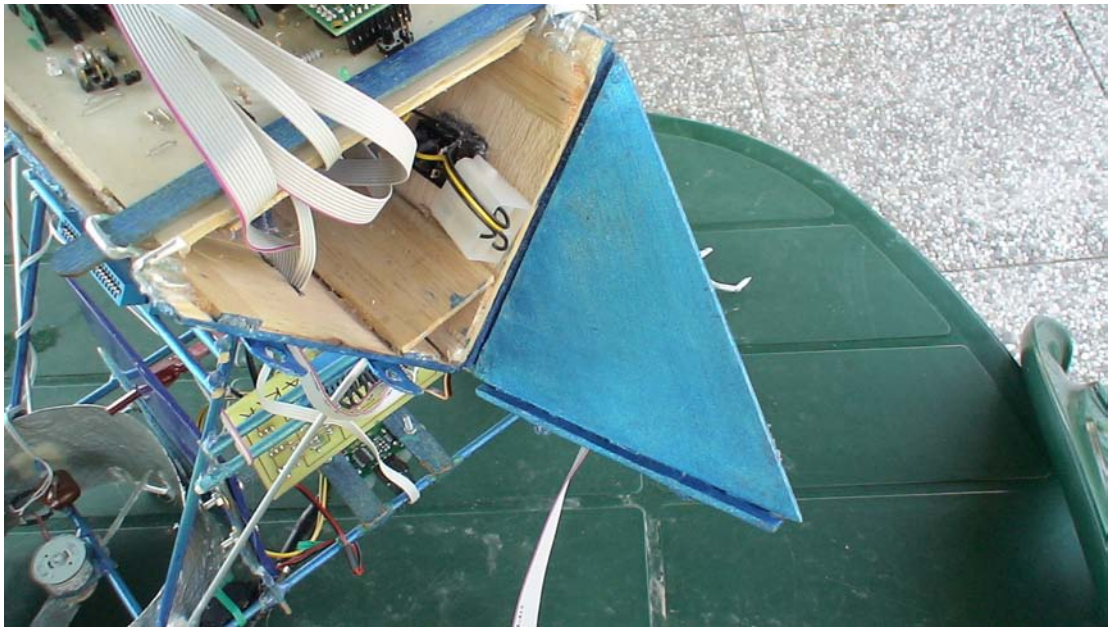
2 Pezzo
punta

Può essere suddivisa in quattro parti principale:

1. l'alloggiamento dei circuiti principali nella parte più alta, con un canale appena sotto la schedina in cui passano tutti i conduttori necessari;
2. l'alloggiamento delle batterie, posto ad un livello intermedio, è stato studiato in modo tale che le batterie siano fisse e che non possano entrare in contatto con i circuiti;
3. moduli radio posti nella punta lontani dalle interferenze
4. l'ultima parte posta nel settore più basso è uno scomparto dedicato ai pesi di calibrazione (zavorra), separata dal resto da uno strato di balsa.

Per la realizzazione della parte centrale della cabina, sono stati tagliati sei pezzi di balsa larghi 10cm e lunghi 12.5cm; ogni lato è composto da tre di questi tasselli incollati tra di loro da un altro pezzo di balsa lungo 30cm e largo 3cm, in modo tale da cerare un unico pezzo lungo 30cm con altezza di 12.5cm.

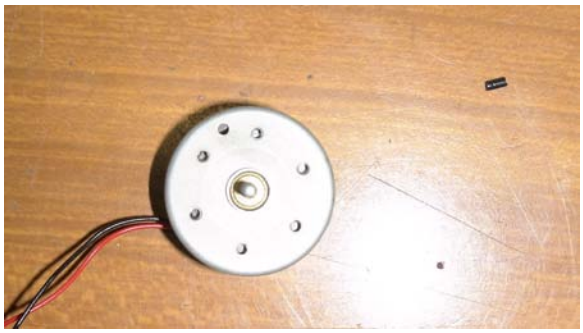
La parte posteriore è stata realizzata con un unico pezzo (vedi schema per le dimensioni). Inoltre abbiamo fatto in modo che tale parte fosse utilizzata come sportello: su di un lato abbiamo incollato un pezzo di tessuto (cotone) per realizzare la cerniera e sull'altro lato abbiamo incollato delle sporgenze di legno: sulla cabina sono fissate delle strisce di plastica flessibili dotate di un foro che va ad incastrarsi nelle sporgenze create sullo sportello, dando origine così alla nostra "serratura".

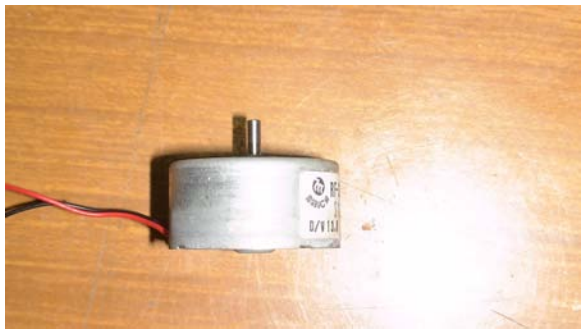


La punta è costituita da quattro triangoli dalle dimensioni indicate nello schema; attraverso due rettangoli di balsa di piccole dimensioni, usati come sostegno, vengono incollati due delle quattro parti tagliate in modo da ottenere un triangolo rettangolo isoscele di base 14.4cm e lati di 10cm. Infine tutti i pezzi ottenuti (i due rettangoli, lo sportello e i due triangoli della punta) vengono assemblati con la colla a caldo, molto più rapida della colla per legno, realizzando cosi la cabina. Nella parte bassa viene messa una striscia di balsa per conferire resistenza alla struttura e per creare lo spazio necessario per i pesi di calibrazione.
La cabina è stata infine dipinta di color blu metallizzato spalmato sulla balsa attraverso un pennello, abbiamo dovuto dare due mani di vernice perché il materiale usato è molto assorbente.

MOTORI, SUPPORTI MOTORI ED ELICHE:

Il dirigibile si muove per mezzo di motori in DC. Non sono stati fatti calcoli per la scelta del motore idoneo, ma si è deciso dall'inizio di usare materiale di recupero: I motorini del carrello dei lettori cd dei computer erano piccoli, leggeri ed abbastanza potenti per i nostri scopi.

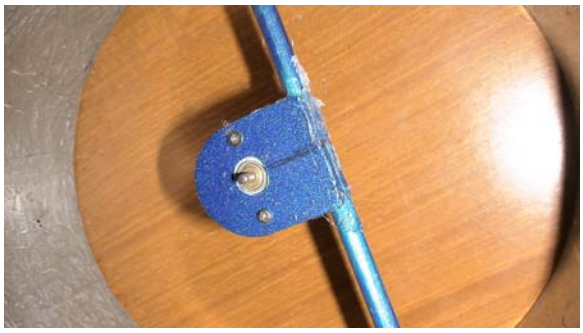




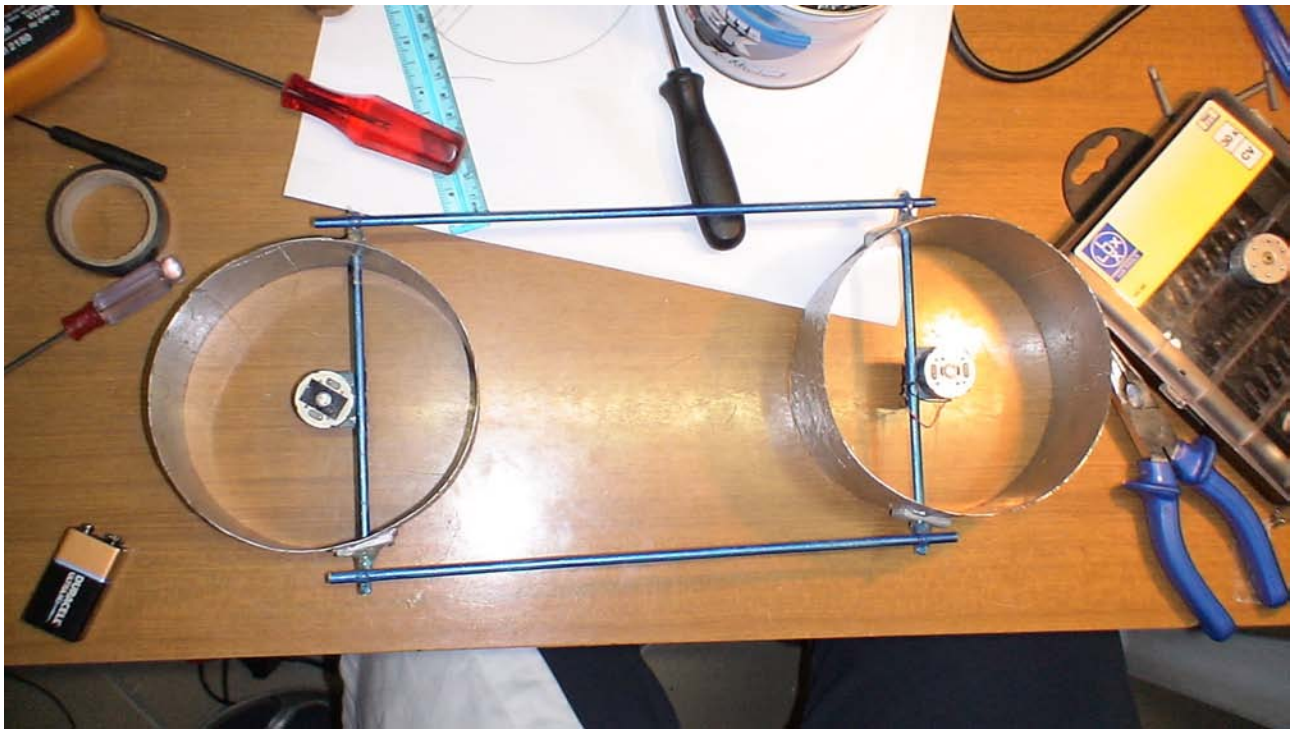
Abbiamo recuperato quattro motori, due per gli spostamenti direzionali (messi ai lati della cabina) e due per gli spostamenti ascensionali (sotto la cabina).
Per poter fissare i motori alla cabina però dovevamo costruire una struttura che li sostenesse. Abbiamo fatto ricerche in internet e scoperto che intubando i motori le prestazioni sarebbero state migliori con le eliche. Inoltre il tubo garantiva anche un buon sostegno del motore ed una buona protezione delle eliche.
Il tubo doveva essere leggero, quindi abbiamo provato diversi materiali che potevano essere adatti allo scopo.
Alla fine la scelta è caduta sulla vetroresina: Abbiamo utilizzato un tubo di plastica come stampo e lo abbiamo avvolto con la vetroresina. Una volta asciutta, la vetroresina è leggera e flessibile ma molto resistente.

Abbiamo tolto dal tubo il sostegno in vetroresina, ormai rappreso e lo abbiamo rifinito e verniciato.
I quattro sostegni ottenuti hanno diametro 13cm e pesano 21g cadauno.

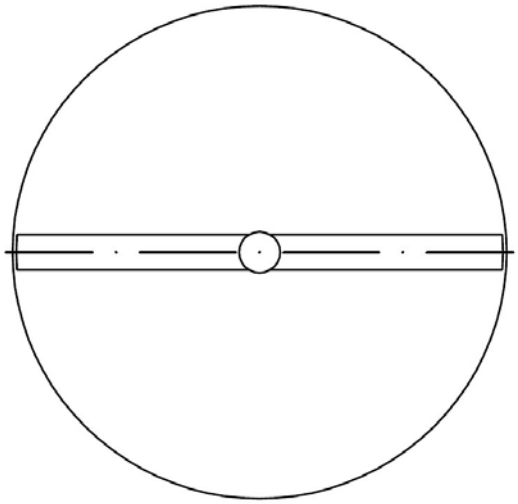
I sostegni sono stati forati alle due estremità per poter inserire un bastoncino di legno su cui fissare il motore. Al bastoncino è stata fissata una mascherina in vetronite (recuperata dalle basette ramate) alla quale viene fissato il motore attraverso delle vitine.
Gli stessi bastoni usati per il fissaggio dei motori sono stati usati come "punti di aggancio" per il resto della struttura e della cabina.

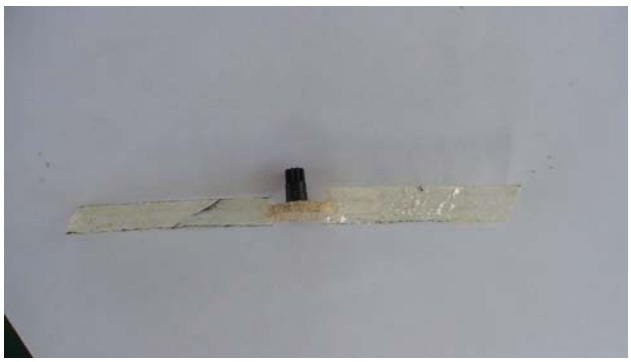


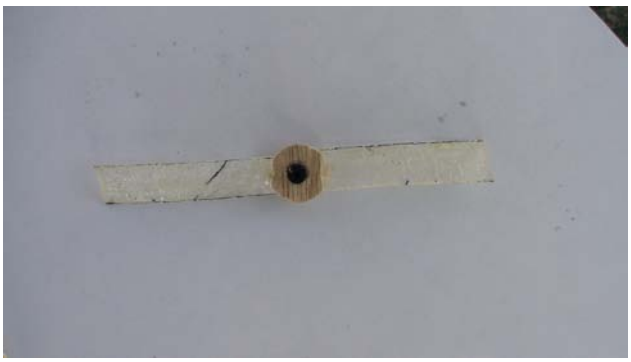
I motori ascensionali sono stati fissati sullo stesso livello con una struttura realizzata in legno. I due motori sono stati distanziati circa 20cm per poter usare lo spazio intermedio come alloggiamento per il modulo della telecamera.
Alla struttura sono inoltre state fissate delle astine in legno (inclinate di 60°) per l'aggancio con la cabina.

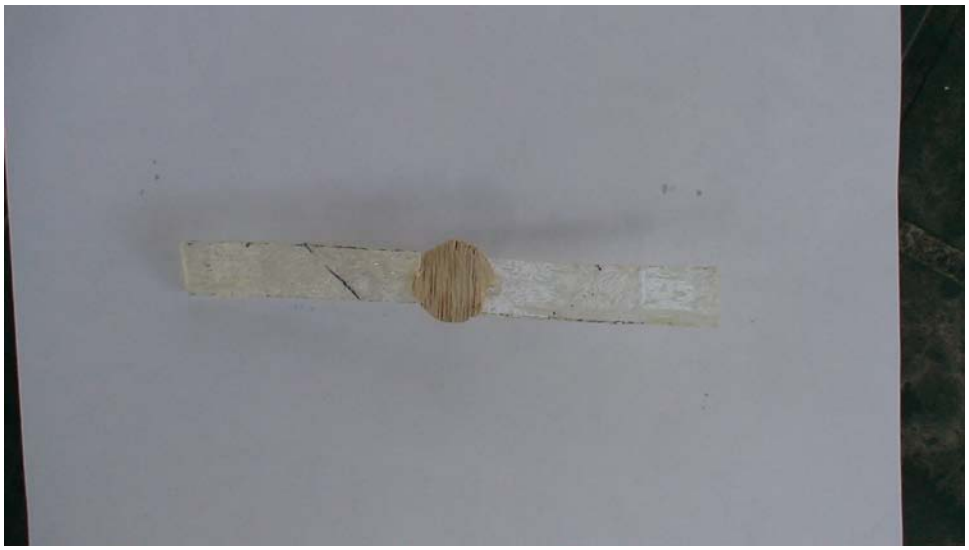


Gli spostamenti del pallone devono avvenire in ogni direzione, quindi le eliche devono essere “invertibili” cioè devono spingere in entrambe le direzioni.
Per la realizzazione delle eliche abbiamo usato la vetroresina.
Abbiamo fatto un foglio di vetroresina da cui abbiamo ritagliato delle striscioline di lunghezza 5,5cm e larghezza 1,5cm. L’elica è composta da un tondino centrale di balsa sul quale vengono incollate due striscioline di vetroresina con inclinazione 45°. In tal modo le eliche spingono in entrambe le direzioni. Per il fissaggio dell’elica al motore è stato incollato un pignone di plastica adatto nel centro del tondino di balsa.







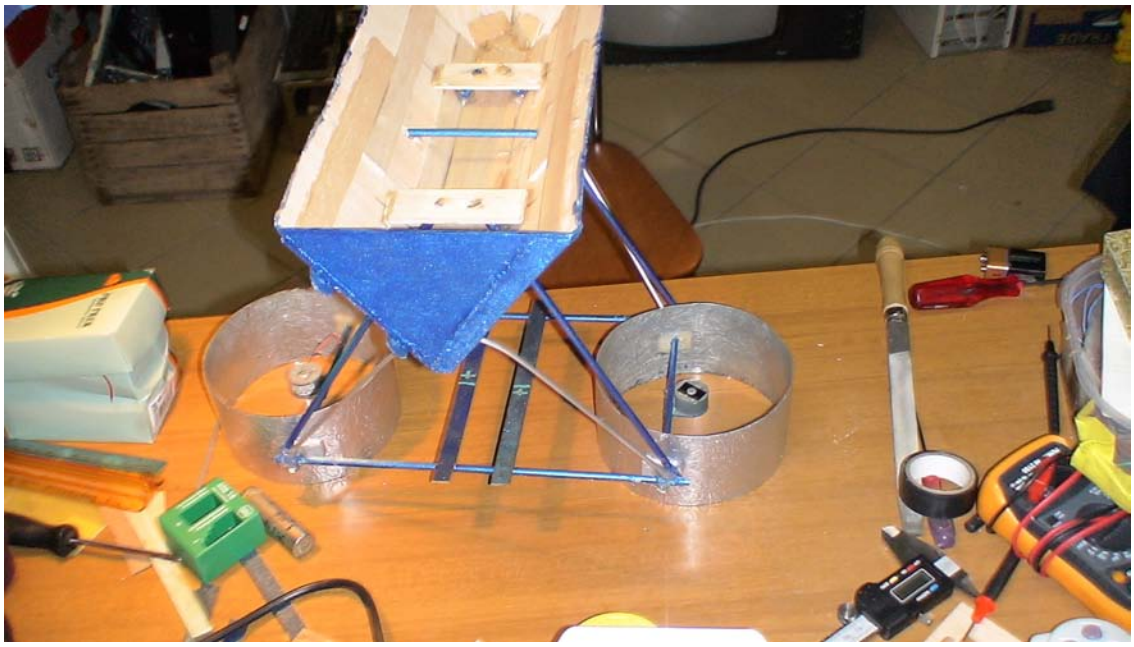


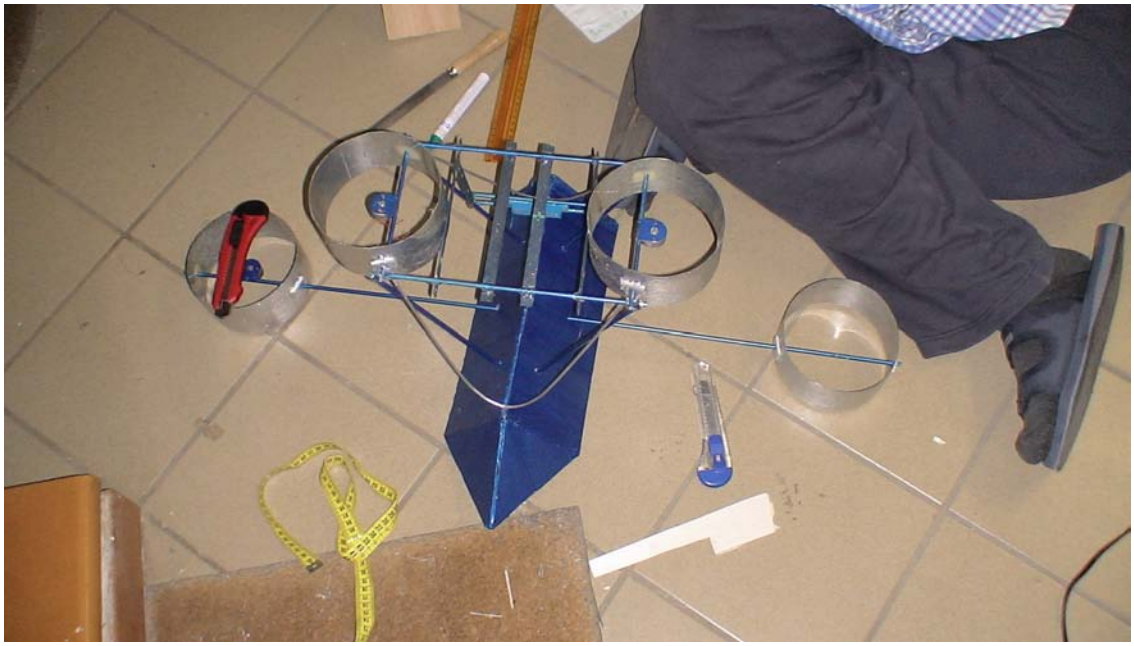
ASSEMBLAGGIO GONDOLA:

Per ultimare la gondola del dirigibile è necessario assemblare il tutto.
Nella cabina vengono fatti due fori, uno per lato, alla stessa altezza.
Nei fori viene fatta passare l'asta fissata ad uno dei supporti motore e dall'altra viene fissato successivamente l'altro supporto.
Il passante, fissato infine con della colla a caldo, sostiene così i due motori laterali (comprensivi di tubo).
Altri fori sono stati fatti in prossimità della punta della cabina.
Tali fori, eseguiti su entrambi i lati, stessa altezza, servono invece per il fissaggio della struttura dei motori ascensionali. Anche in questo caso si fa uso abbondante di colla a caldo.

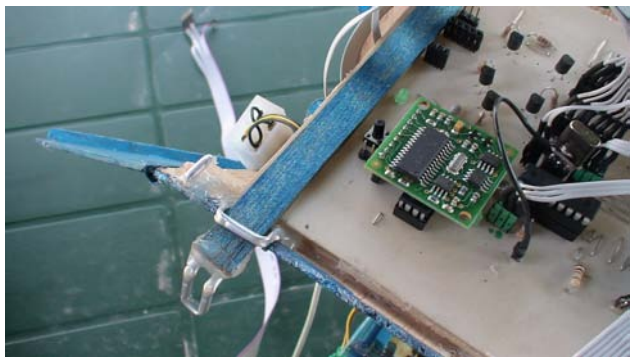
Le aste usate per il fissaggio entrano nella cabina fino a toccarsi e vengono fissate ulteriormente con un rettangolo di balsa. Lo spazio disponibile sotto il livello del rettangolo di sostegno viene usato come vano batterie.
È stata costruita una apposita slitta che viene inserita da dietro e va ad incastrarsi nelle profondità della cabina. Tale slitta sorregge le due celle al Litio usate per l'alimentazione dei circuiti.

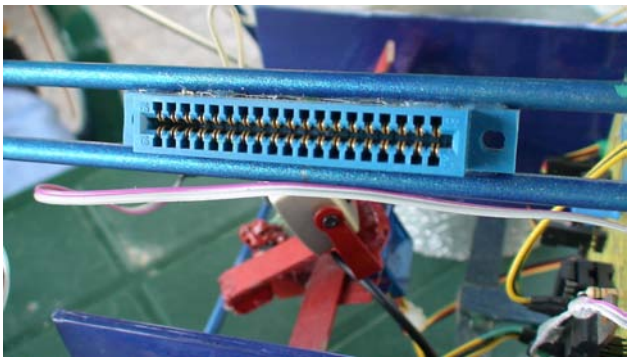
Nella cabina sono stati effettuati altri fori per il passaggio dei cavi dati e di alimentazione.





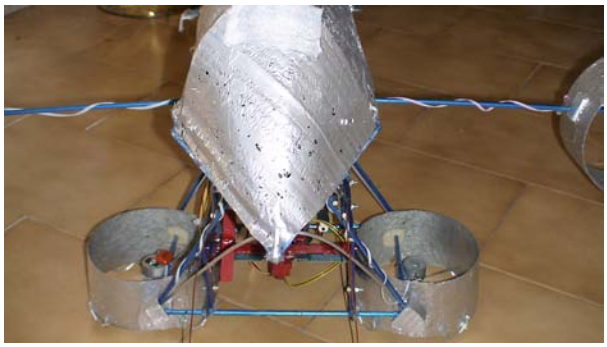
La gondola ottenuta deve essere adesso rifinita.
La struttura dei motori ascensionali è stata ampliata: è stato aggiunto il sostegno per il sensore di altitudine ad ultrasuoni, ed ulteriori rinforzi.
Inoltre sono stati fissati gli slot di espansione e sono stati effettuati i fori per il fissaggio del modulo della telecamera.
Ai supporti dei motori ascensionali sono stati fissati dei “piedini” in alluminio per evitare che il supporto si danneggi durante l’atterraggio.
Inoltre sono stati aggiunti dei ganci in legno per il fissaggio della scheda elettronica (che può essere rimossa) ed altri in alluminio per il fissaggio della cupola.

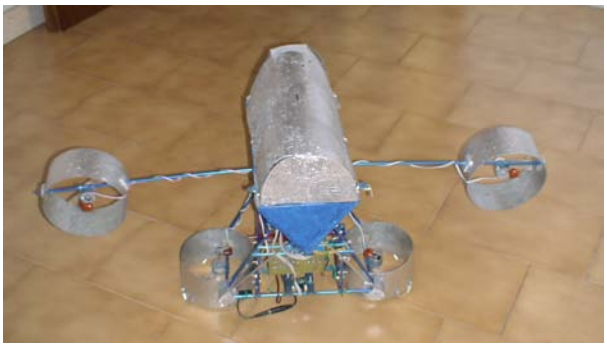


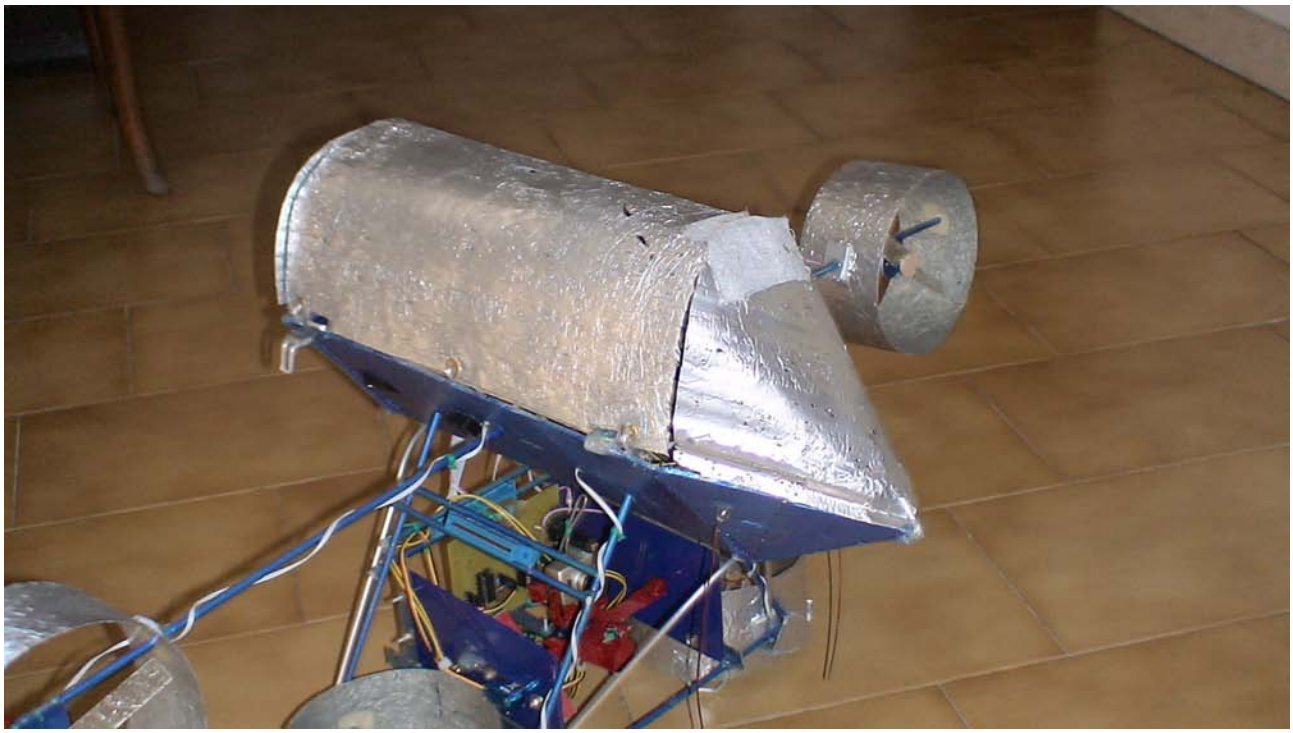


CUPOLA:

Riguarda la parte che copre la cabina. È costituita da un rettangolo in vetroresina alleggerita di dimensioni 30x27cm. Presenta un foro centrale per il passaggio dei cavi flat diretti alle schedine di derivazione e sei fori, tre per lato, per il fissaggio della cupola alla cabina. Tali fori sono stati rinforzati con delle rondelle in ottone fissate con della colla a caldo. Per coprire la punta viene utilizzata sempre della vetroresina a cui viene conferita la giusta curvatura attraverso dei supporti in alluminio, per determinare la corretta forma sono state effettuate delle prove con del semplice cartoncino fino a raggiungere il risultato prestabilito. Infine per chiudere la parte posteriore, abbiamo ritagliato da un foglio di vetroresina un semicerchio, di raggio 7.5cm, che presenta nella parte rettilinea una sporgenza che va ad incastrarsi tra lo sportello della cabina e la circuiteria. I tre pezzi ottenuti sono stati collegati tra di loro con della stoffa (cotone) incollata con della colla a caldo in modo da poter avere una certa mobilità. La struttura è stata dipinta con del colore spry grigio.



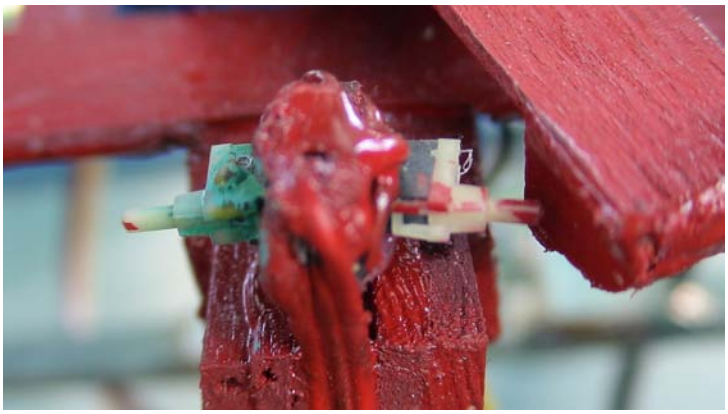




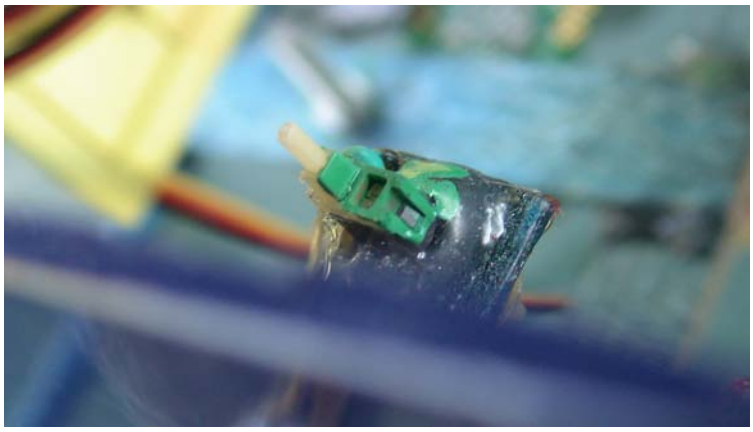
MODULO CICLOPÈ (TELECAMERA):

Il modulo Ciclopè è stato realizzato per dare la possibilità al dirigibile di effettuare riprese dall'alto. In sostanza è costituito da una struttura di sostegno per la telecamera ed i relativi motori di posizionamento.
La parte principale è l'asta di posizionamento sull'asse y: è costruita in legno e sostiene il motore per il posizionamento sull'asse x. I motori sono entrambi servomotori modificati. È stata tolta l'elettronica di controllo ed il potenziometro per poterli utilizzare come motoriduttori (piccoli e leggeri). Il motore è stato fissato per mezzo di viti all'asta centrale, sulla quale è stata realizzata una mini-struttura di sostegno.
Su tale mini-struttura sono stati fissati anche i finecorsa per l'asse x.

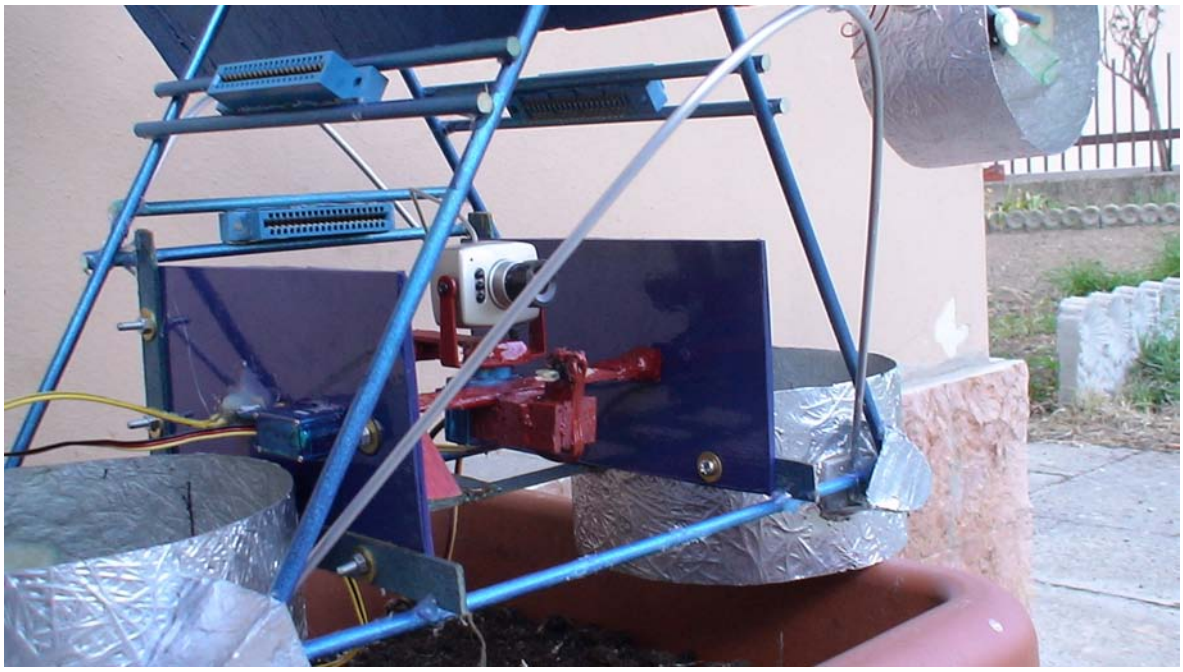
Al motore per il posizionamento sull'asse orizzontale è stata fissata la telecamera wireless (portata 30m uhf) attraverso la struttura metallica fornita assieme al dispositivo. È stato poi aggiunto un braccio di legno che va a premere i finecorsa quando la telecamera raggiunge l'angolazione massima (destra e sinistra).

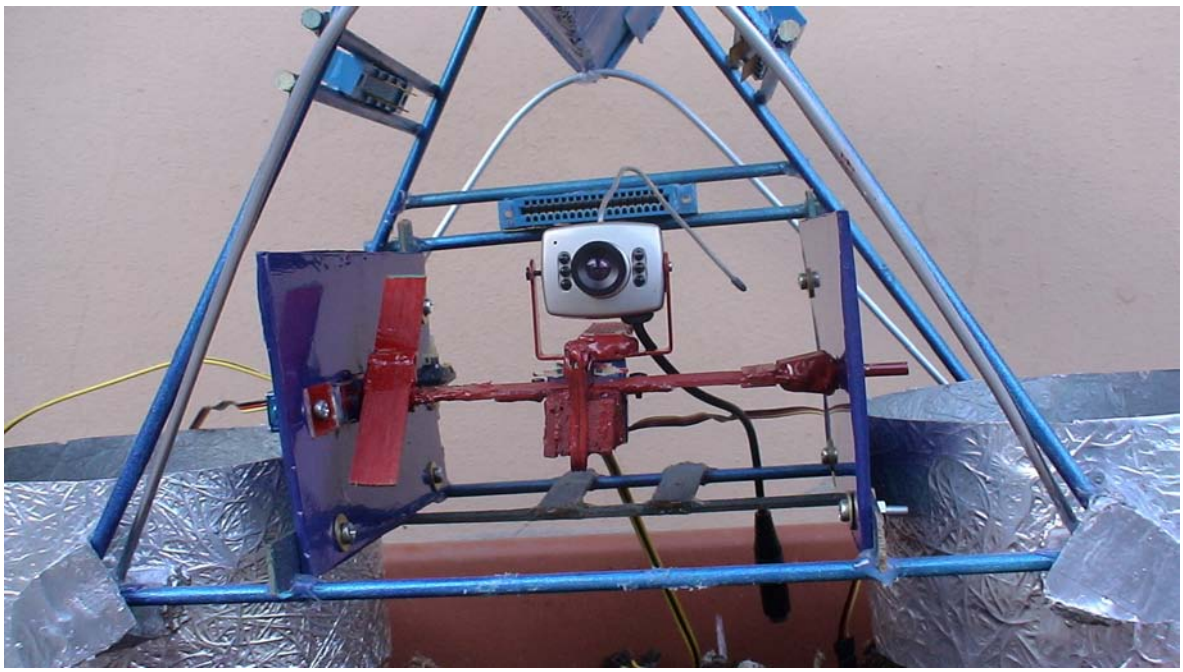


L'asta per il posizionamento sull'asse verticale è fissata ad una estremità al motore. A pochi millimetri dal motore è stato fissato un braccio di legno per la pressione dei finecorsa dell'asse y.

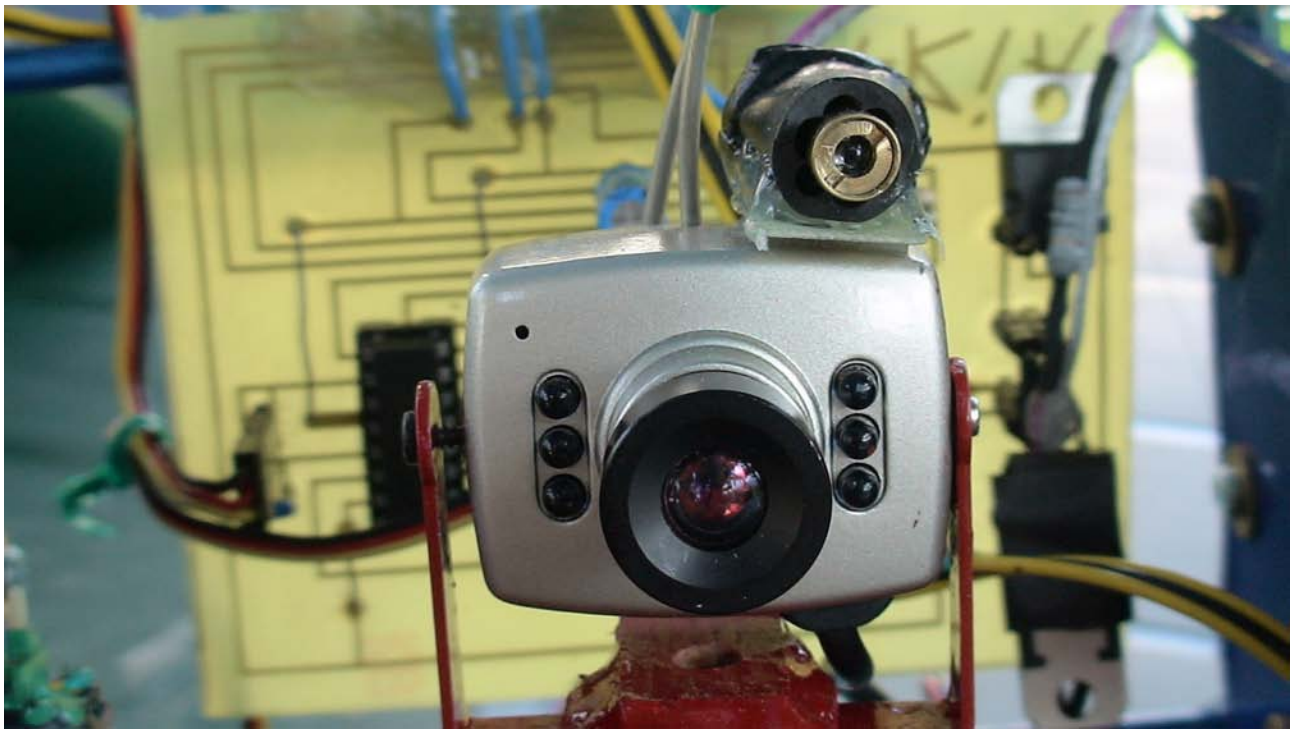


Il motore si incastra in un pannello laterale di balsa rivestita con il termoretraibile. Il pannello, che include i finecorsa del motore sopraccitato (su e giu), è fissato alla struttura dei motori ascensionali tramite delle viti. Tutti i fori sono stati rinforzati con rondelle in ottone.
Sull'altro lato, un altro pannello sostiene l'estremità dell'asta centrale in modo che possa ruotare.
I pannelli sostengono tutto quanto e possono essere rimossi, grazie alle viti, rendendo il modulo ciclopè estraibile.

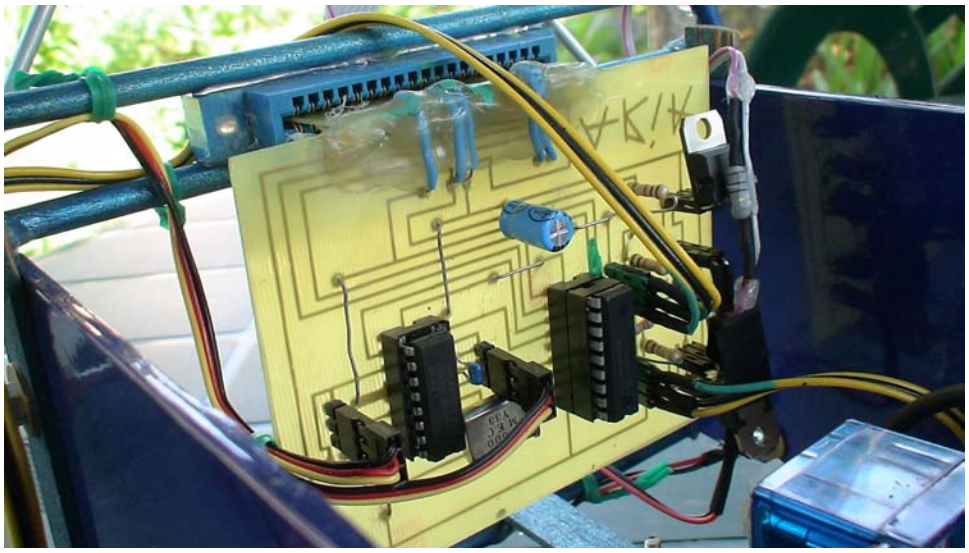


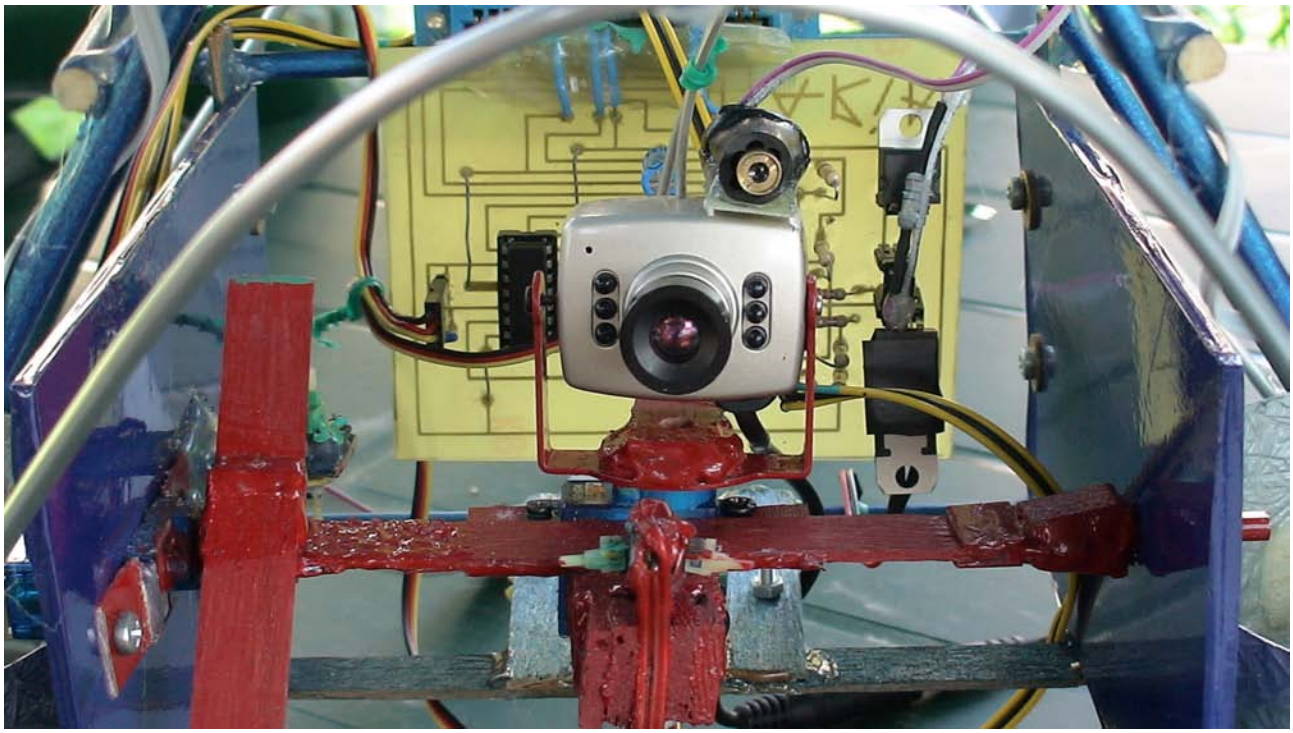


Sulla telecamera è stato fissato infine un puntatore laser tramite del nastro biadesivo. Il puntatore può essere utile per riprese con scarsa luminosità o come mezzo di segnalazione.



La schedina di controllo del modulo è sparata da quella principale, con la quale comunica per mezzo di un link seriale. Tale scheda viene inserita nello slot di espansione: Presenta un connettore realizzato su una basetta ramata incollato a 90° rispetto alla scheda di controllo.
Il connettore viene inserito ad incastro nella presa dell'espansione. I contatti della presa aderiscono alle piste ramate del connettore e questo garantisce il collegamento elettrico.
Il principio è lo stesso degli slot di espansione dei PC.

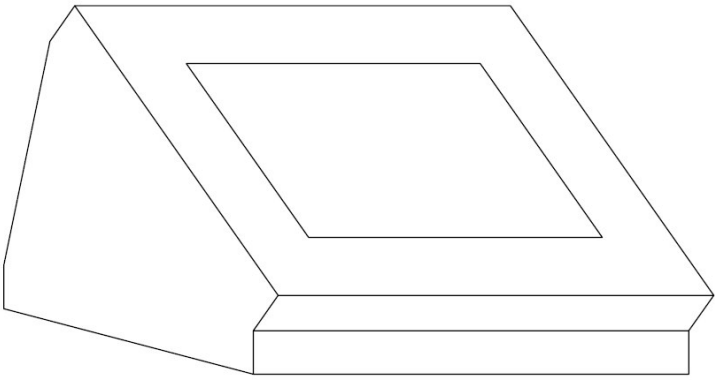


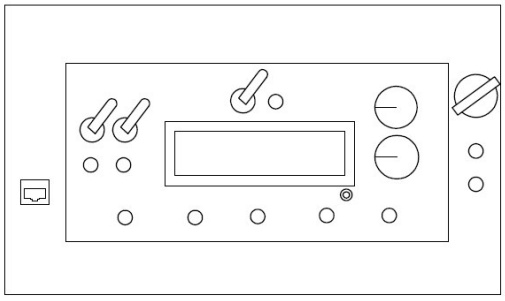


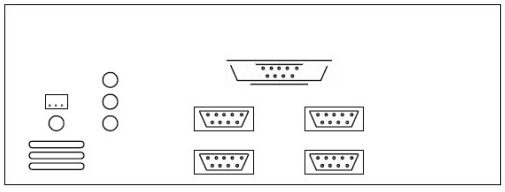
STRUTTURA BASE

COSTRUZIONE BASE:

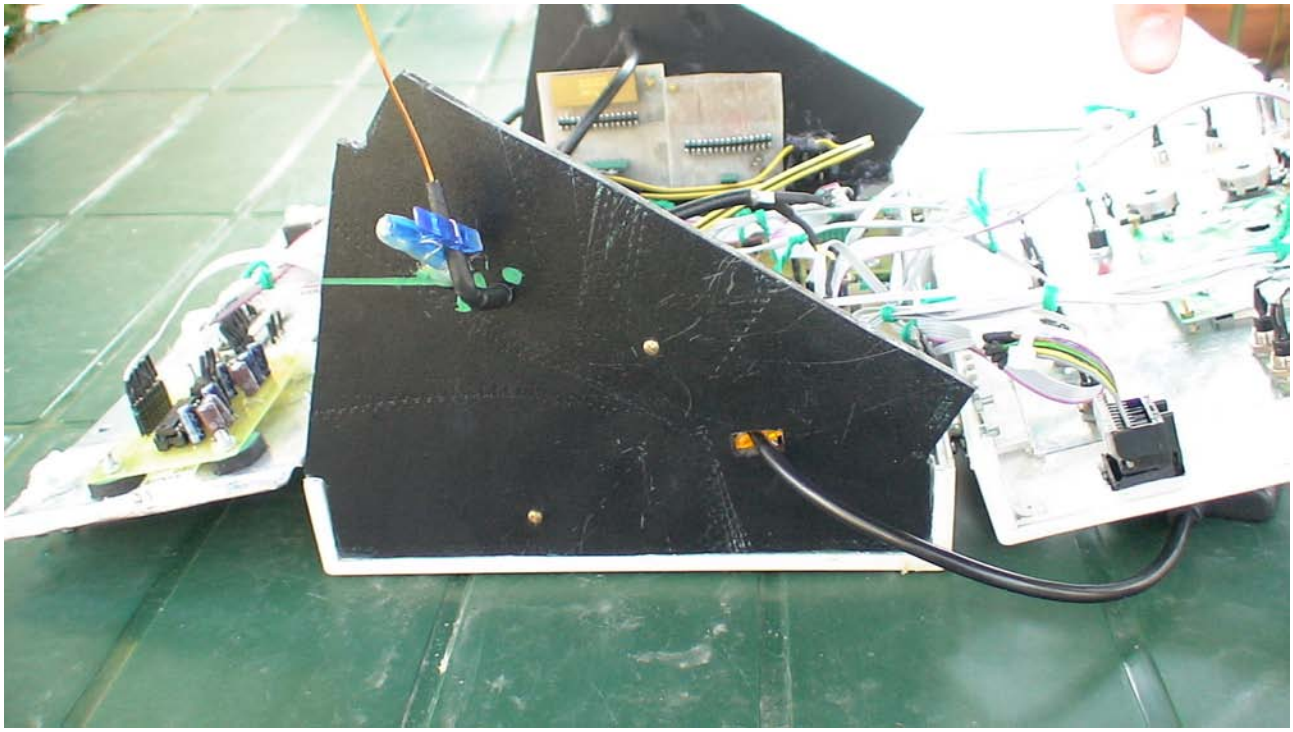
La base terrestre è stata ricavata da un vecchio UPS (gruppo di continuità) ormai inutilizzabile.
Abbiamo tolto dal suo interno i circuiti e le batterie e abbiamo iniziato a realizzare la nostra base.



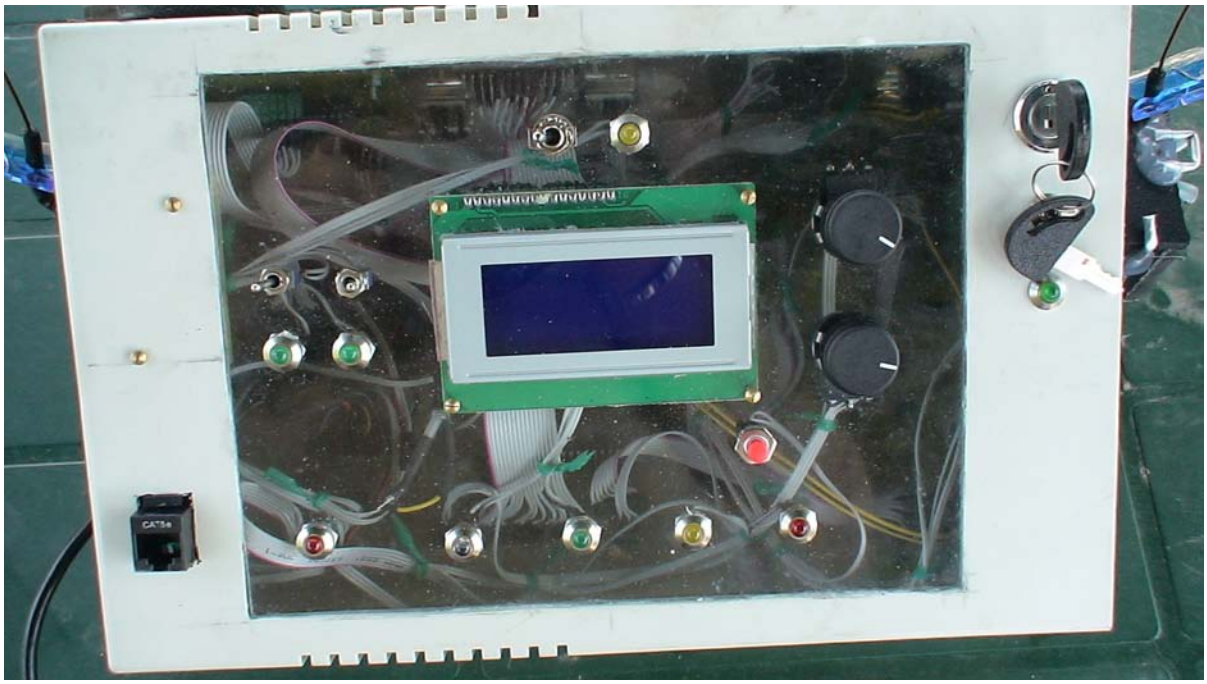




L’involucro dell’ups era composto da due parti pressoché simmetriche, creando cosi un parallelepipedo. Noi avevamo la necessità di avere un piano inclinato per cui abbiamo incollato ai lati dell’involucro due triangoli in pvc espanso (facile da modellare) opportunamente lavorarti. Su tali triangoli sono stati effettuati dei fori per la fuoriuscita delle antenne e per l’installazione dell’interruttore generale.



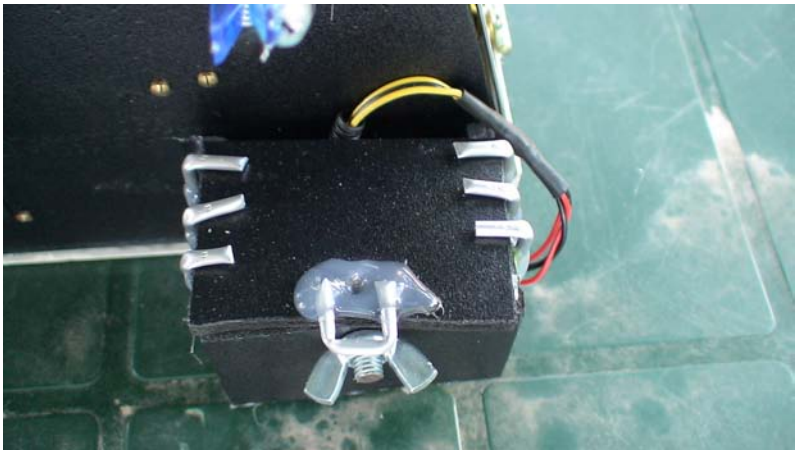
Le due parti dell’ups sono state collegate tra di loro attraverso una cerniera in metallo per consentirne l’apertura completa. Sulla parte inclinata è stato intagliato un rettangolo al posto del quale ve ne è stato messo uno in lexan, fissato con della colla (mastro d’ascia), per garantire la possibilità di vedere la circuiteria all’interno.
Inoltre sul rettangolo in lexan sono stati installati tutti i dispositivi di comunicazione, tra cui uno schermo lcd.

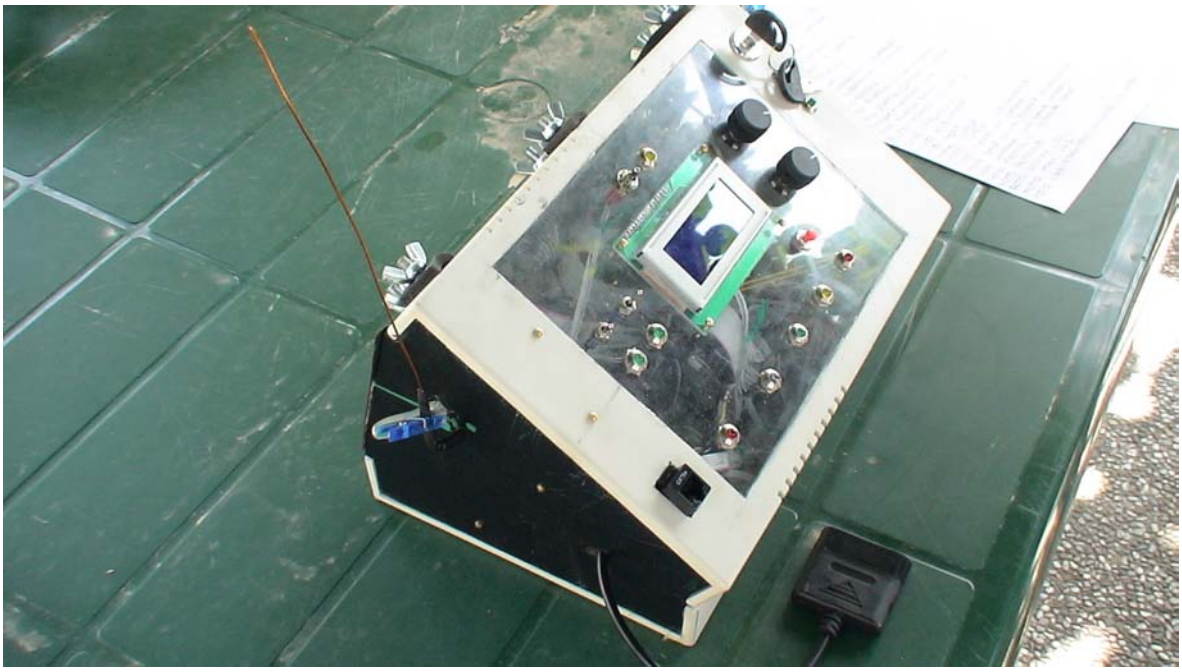


La parte posteriore è stata chiusa da un rettangolo di lamiera, ricavato da un vecchio alimentatore per computer. Tale rettangolo è stato fissato con delle cerniere alla base della scatola in modo tale che possa essere aperta sia davanti che dietro. Inoltre sulla lamiera sono stati fissate quattro porte seriali (2 per la comunicazione con i pc, 2 per la programmazione dei pic), 4 jack (2 per le operazioni di debug, 1 per l'alimentazione generale e 1 per l'alimentazione dei circuiti di debug), infine un connettore polarizzato per l'alimentazione esterna.

La chiusura completa della base avviane tenendo a contatto le due estremità mobili attraverso dei rettangoli di pvc espanso fissati con delle viti corredate di bulloni a farfalla.

Su di un lato della base è presente una scatolina, costruita sempre in pvc espanso. In tale contenitore vengono riposte le batterie necessarie per l'alimentazione della base. La scatolina ha un coperchio superiore a slitta con apposito gancetto per l'estrazione.



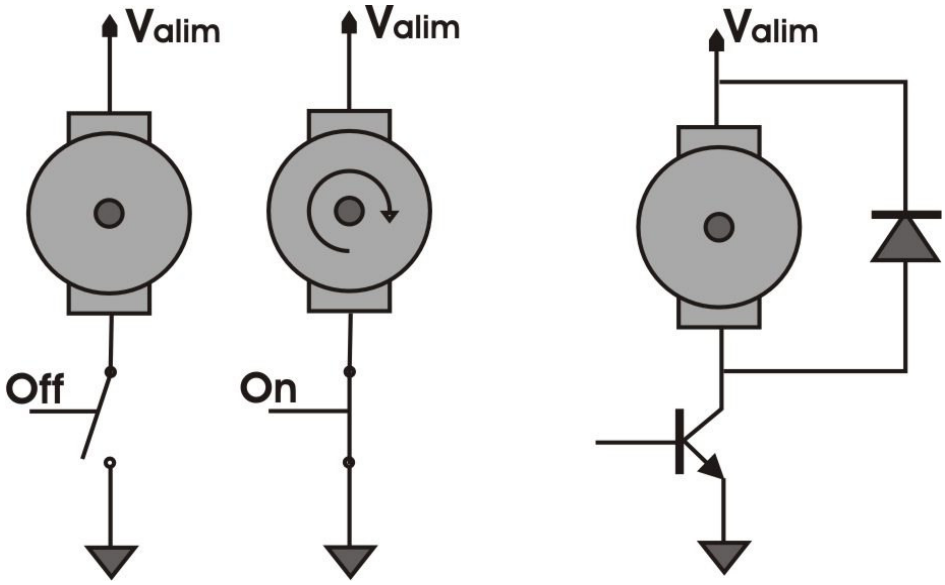


ELETTRONICA

CONTROLLO MOTORI

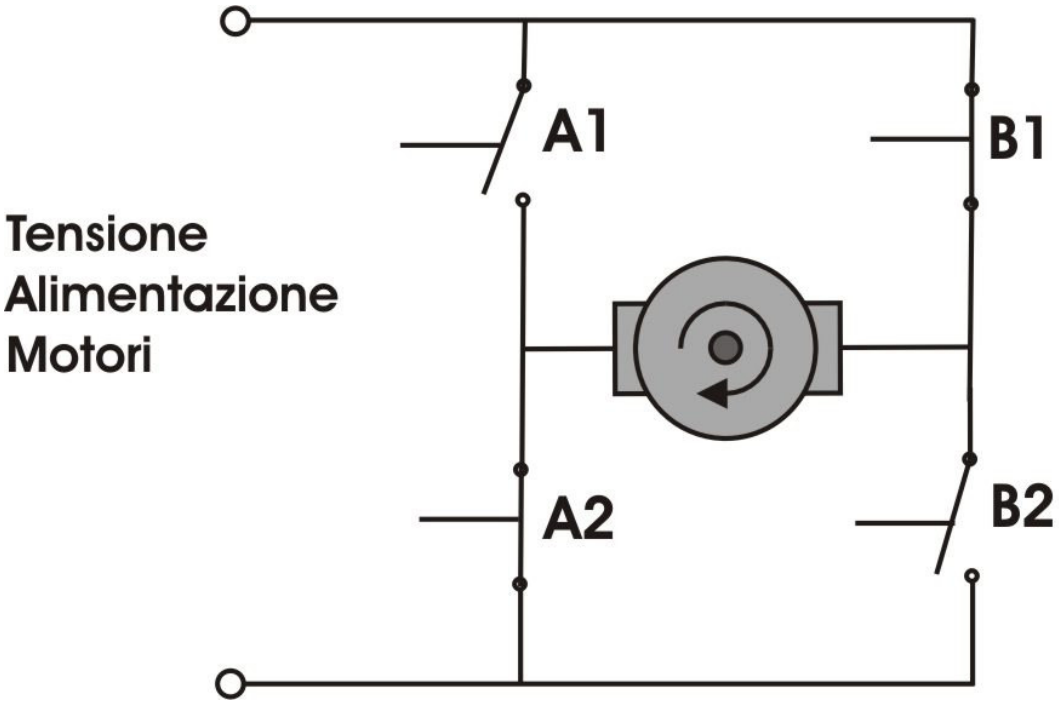
Vi sono vari tipi di regolazione e di controllo dei motori in corrente continua. Riportiamo alcuni dei modi di controllo:

Controllo ON-OFF di motori DC:
Il pilotaggio più semplice è quello ON-OFF che permette di comandare il motore solo alla massima velocità di rotazione in un verso (interruttore ON) oppure fermarlo (interruttore OFF): questo controllo può essere implementato con un interruttore (un mos o un transistor) e con un diodo di ricircolo necessario per evitare danni al resto del circuito (il motore è un carico con una componente induttiva).



pilotaggio ON-OFF: quando l'interruttore è aperto il motore è fermo, quando è chiuso gira alla massima velocità. A destra un esempio di implementazione

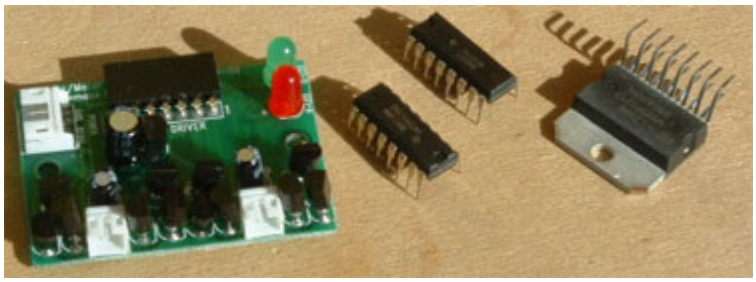
Il ponte ad H:
Il tipo di controllo appena presentato non permette né la regolazione della velocità del motore né la possibilità di far girare il motore in entrambi i versi di rotazione.
Per far girare il motore nel verso opposto è necessario infatti invertire il segno della corrente che passa all'interno del motore stesso, per far ciò si usa un circuito chiamato ponte H costituito da 4 interruttori comandati e da 4 diodi di ricircolo.



schema semplificato di un ponte H

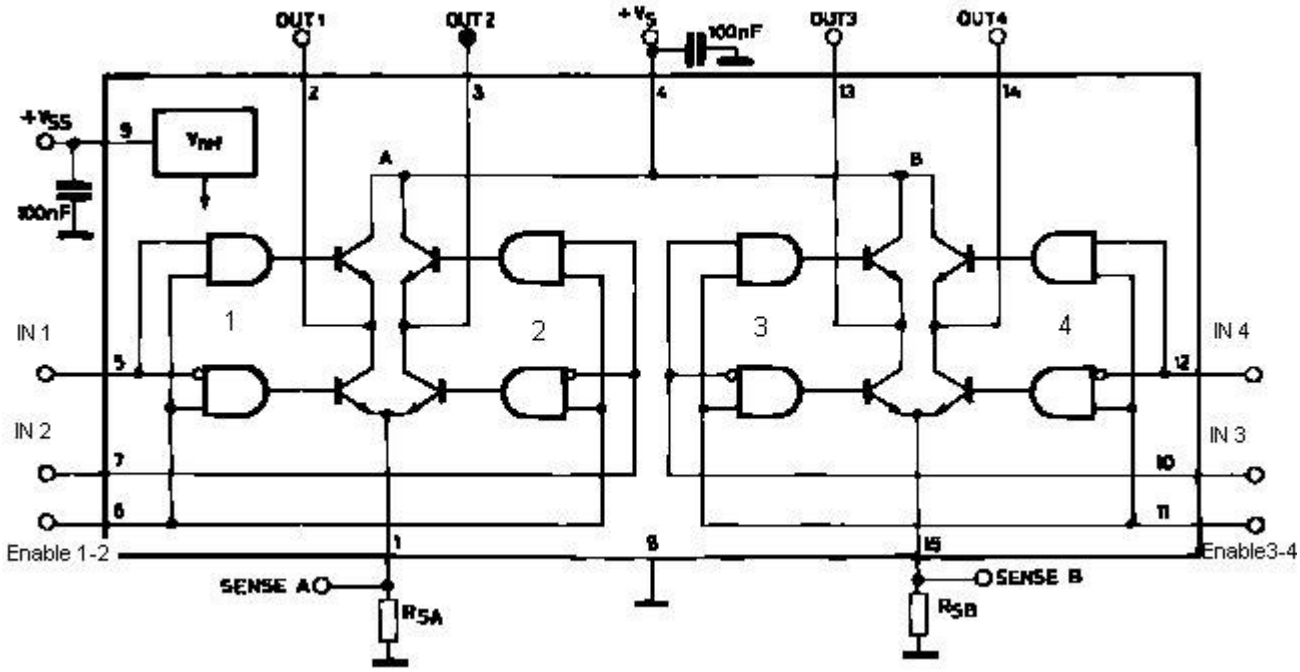
Attivando i transistor A1 e B2 la corrente scorre nel motore in un verso mentre attivando i transistor B1 e A2 la corrente scorre nel verso opposto. E' da evitare la configurazione in cui sono accesi entrambi i transistor A o entrambi i transistor B infatti la corrente di corto circuito su un lato del ponte potrebbe creare seri danni al ponte stesso o al circuito di alimentazione.

Ponti ad H discreti e ponti ad H integrati:
Esistono due tipi di ponti H: i ponti H discreti, costituiti da componenti sparsi come transistor e diodi e i ponti H integrati, in questo caso tutto il circuito è racchiuso in un package plastico. I ponti H integrati sono molto versatili e, oltre a garantire una bassa occupazione di area nel circuito (in alcuni casi, contengono anche i diodi di ricircolo), hanno buone prestazioni, possono essere montati in parallelo per ottenere alte correnti e riescono a lavorare in un intervallo di tensioni di alimentazione molto ampio (da 6V a 48V circa a seconda del modello).



Nell'ordine (da sx a dx): un doppio ponte H discreto (scheda controllo motori Cybot), un SN754410, un L293NE, un L298

La figura 4 rappresenta lo schema di funzionamento di un ponte integrato:

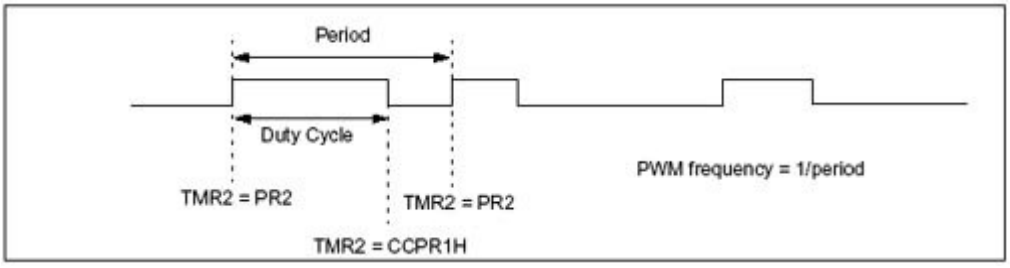


Schema di funzionamento interno di un ponte H integrato

Tale integrato è costituito da quattro mezzi ponti H (numerati in figura come 1-2-3-4) ognuno dei quali è costituito da due transistor e da una logica che li comanda in modo da accenderne solo uno alla volta: quando il transistor superiore di un mezzo ponte è in conduzione quello inferiore sarà necessariamente spento e viceversa. E' inoltre presente un comando di ENABLE che permette di inibire il funzionamento di una coppia di mezzi ponti.
Si può inoltre notare la presenza di due pin dedicati alla connessione di una resistenza di Sense per monitorare la corrente che scorre nel motore (utile per un controllare l'assorbimento di corrente ed evitare rischiosi stalli del motore) ma tale caratteristica non è presente in tutti i ponti H integrati.

Ricapitolando, per ognuno dei due ponti presenti nell'integrato abbiamo a disposizione due ingressi di controllo per permettere il passaggio di corrente in un verso e un ingresso di ENABLE per accendere e spegnere il ponte: ci sono più modi di pilotare il ponte H e per comprenderli è necessario aver capito bene a cosa servono questi tre ingressi.
Con l'introduzione del Ponte H abbiamo risolto il problema della rotazione del motore in entrambi i versi, ora è necessario risolvere l'altro problema, quello di poter regolare la velocità di rotazione, per far ciò è necessario introdurre il concetto di PWM.

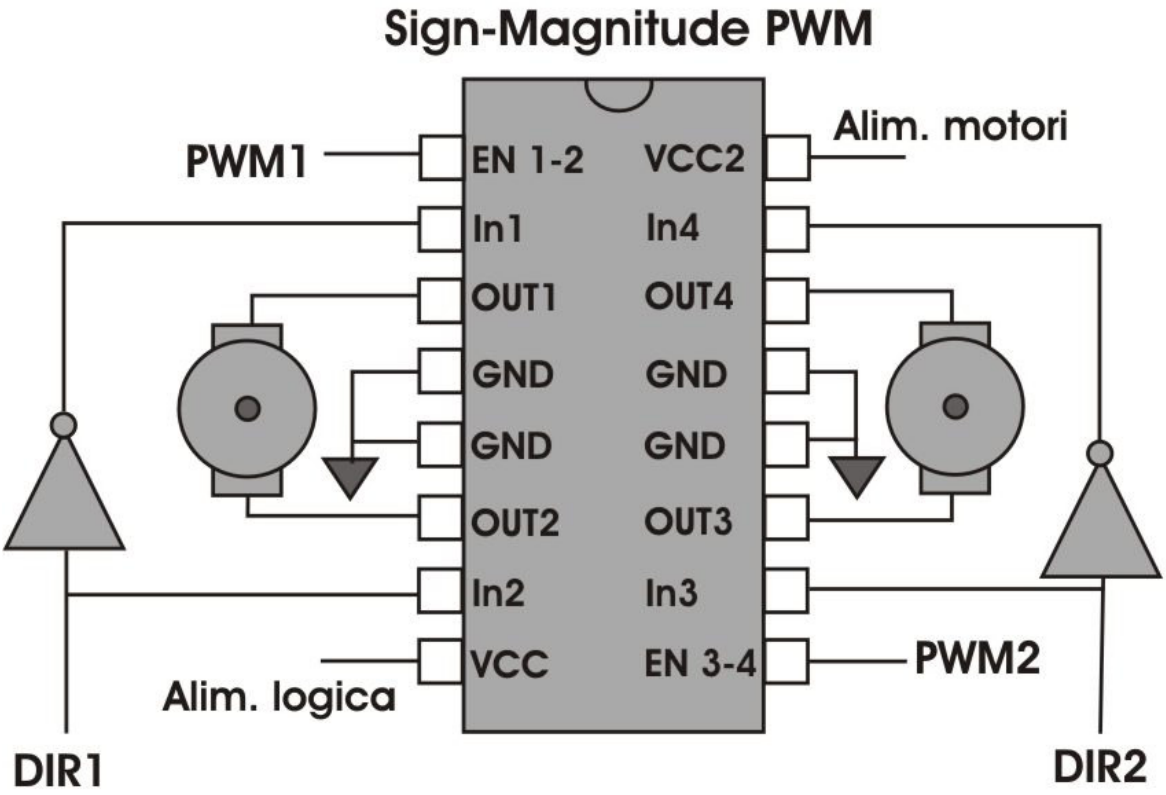
PWM:
Un segnale PWM (Pulse Width Modulation ovvero modulazione a variazione della larghezza d'impulso) è un' onda quadra di duty-cycle variabile che permette di controllare l'assorbimento (la potenza assorbita) di un carico elettrico (nel nostro caso il motore DC), variando (modulando) il duty-cycle.



Definizione di duty cycle

Un segnale PWM è caratterizzato dalla frequenza (fissa) e dal duty-cycle (variabile). Il duty-cycle è il rapporto tra il tempo in cui l'onda assume valore alto e il periodo T (l'inverso della frequenza: T=1/f). Ne segue che un duty-cycle del 50% corrisponde ad un'onda quadra che assume valore alto per il 50% del tempo, un duty-cycle dell'80% corrisponde ad un'onda quadra che assume valore alto per l'80% del tempo e basso per il restante 20%, un duty-cycle del 100% corrisponde ad un segnale sempre alto e un duty-cycle dello 0% ad un segnale sempre basso.
Ora è necessario capire come applicare il segnale PWM al ponte H per controllare il motore. Esistono in sostanza due modalità: il PWM sign-magnitude e il PWM locked anti-phase.

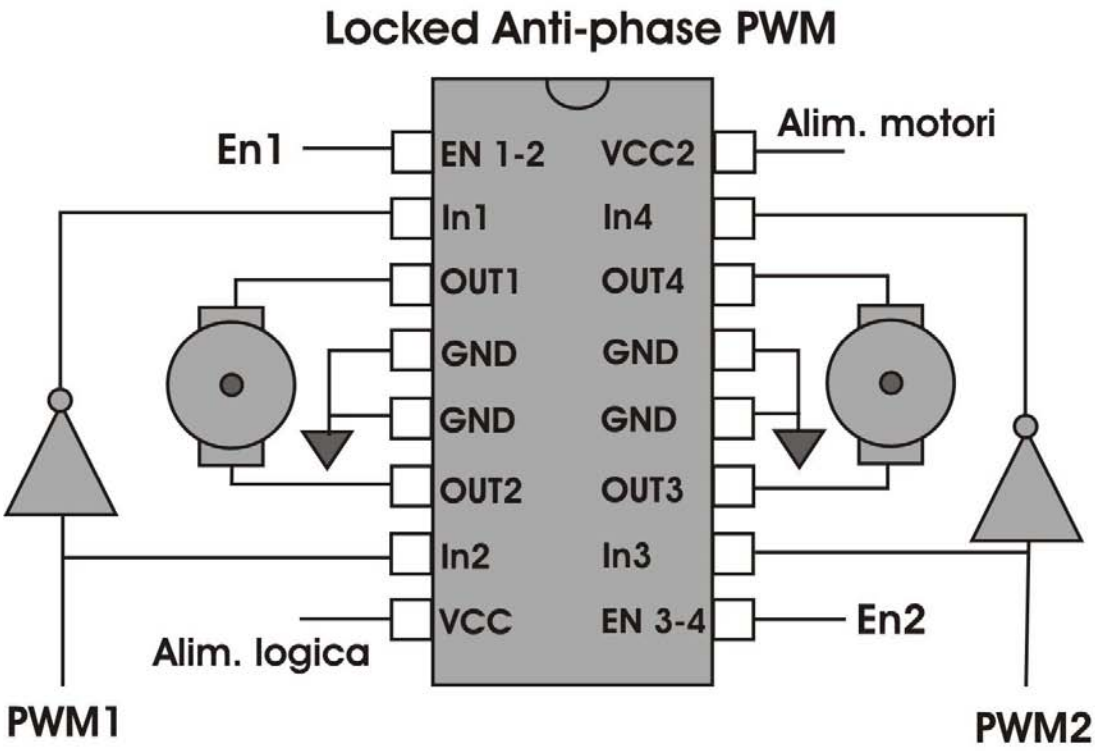
Sign-Magnitude PWM:
Il pilotaggio SM (Sign-Magnitude) consiste nell'inviare il segnale PWM all'ingresso di enable del ponte e di comandare la direzione di rotazione del motore tramite i due ingressi di controllo. Tali due ingressi devono essere comandati da segnali invertiti; utilizzando un inverter si riduce il numero di pin del microcontrollore necessari per il controllo.



Sign-Magnitude PWM

Per il controllo SM sono necessari quindi due segnali: il primo è un'onda quadra di duty-cycle variabile tra 0 e 100% che stabilisce la velocità di rotazione, il secondo è un segnale costante che determina il verso di rotazione (segnale basso rotazione in un verso, segnale alto rotazione nell'altro verso).

Locked-Anti-Phase PWM:
Il controllo LAP (locked-anti-phase) si basa sulla stessa configurazione circuitale del controllo SM tuttavia i segnali di comando sono applicati in modo diverso.



Locked anti-phase PWM

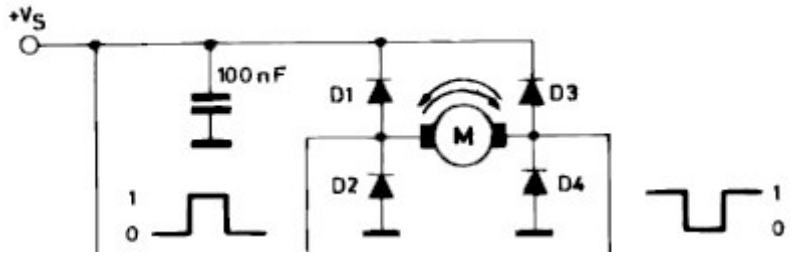
In questo caso il segnale PWM viene messo in ingresso all'invertitore in modo da avere ai due lati opposti del ponte due segnali invertiti tra loro; agendo sull'enable è possibile spegnere il rispettivo ponte.
Per il controllo LAP può bastare anche solo un segnale di comando (l'enable può essere fissato alto se non necessario) infatti l'onda quadra stabilisce sia la velocità che il verso di rotazione:
- Duty cycle a 0% : rotazione alla massima velocità in un verso
- Duty cycle al 50%: motore fermo
- Duty cycle al 100%: rotazione alla massima velocità nell'altro verso

Per il controllo dei motori in DC dell'aeronave e dei motori del modulo Ciclopè, abbiamo adottato il chip integrato SN754410 con le seguenti caratteristiche:

- Due ponti ad H integrati;
- Tensione di alimentazione dei motori: 4.5-36 V;
- Tensione della logica di controllo: 5 V;
- Massima corrente per ponte: 1 A;
- Uscite ad alta impedenza (tristate)

Nel nostro caso, il modo migliore per sfruttare al massimo la versatilità di questo integrato è di pilotare i motori in modalità SIGN-MAGNITUDE PWM**. A**bbiamo creato 3 onde PWM: due, generate dal PIC18F542, che comandano i due motori delle direzioni, mentre la terza, creata dal PIC16F88, che pilota i due motori ascensionali. Utilizziamo due microcontrollori, in quanto il PIC18F542 non ha sufficienti uscite PWM hardware. Ovviamente in tali PIC sono stati riservati altri pin per le direzioni.

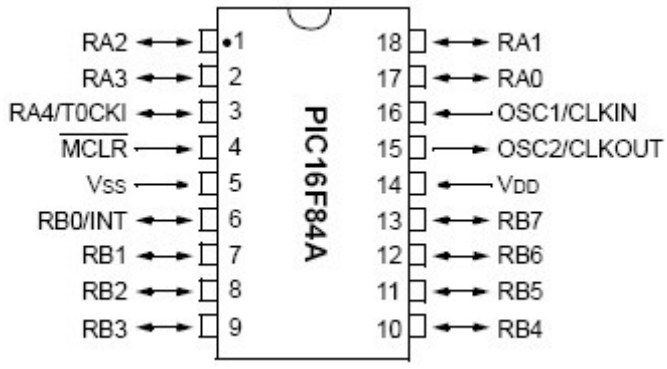
Inoltre sono state costruite due schedine da porre sopra i controllo motori sulle quali sono stati montati dei condensatori di filtraggio e dei diodi di scarica nella configurazione sotto riportata:



MICROCONTROLLORE PIC

Il PIC è un circuito integrato contenente un microprocessore, memorie ed altri dispositivi per l'esecuzione di operazioni complesse..
In poco spazio si ha un dispositivo programmabile in grado di eseguire piccole automazioni.

Ma la sua potenza non si limita a questo.
La memoria non volatile dei pic è suddivisa in due parti: la parte "codice" dove vi è caricato il programma realizzato dall'utente e la parte "dati" che è messa a disposizione dell'utente per immagazzinare dati a lungo termine (non vengono cancellati in assenza di alimentazione).
Il pic è dotato di una memoria Ram interna per l'esecuzione del programma e per il salvataggio delle variabili temporanee. Tale memoria è suddivisa in tre parti: La prima contiene i registri interni del PIC, contenenti tutte le impostazioni del pic (stati delle porte di i/o…), la seconda è la memoria di stack, che contiene gli indirizzi delle posizioni del programma salvate durante l'esecuzione di un salto e la terza che è messa a disposizione dell'utente per il salvataggio delle variabili di programma volatili.



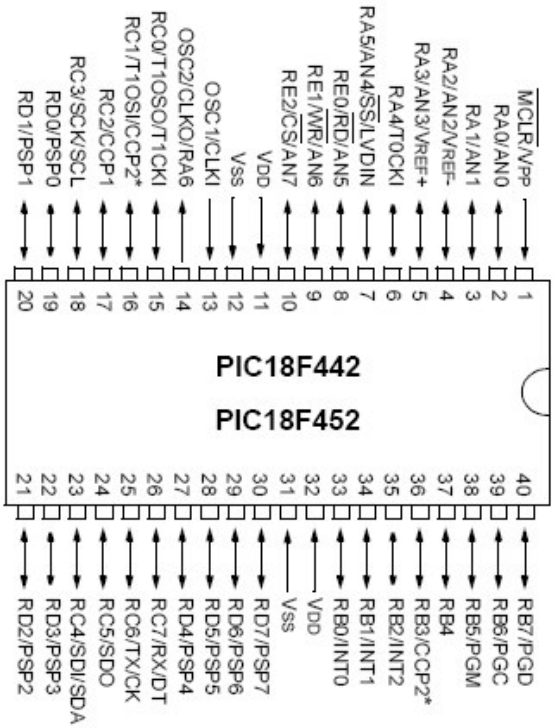
Il microcontrollore viene programmato attraverso un circuito che viene chiamato “programmatore”. Attraverso tale circuito è possibile interfacciare il computer con il pic per caricarvi dentro il programma. Tale operazione è eseguita per mezzo di un apposito software.
Una volta programmato il pic non è pronto all’uso, ma bisogna effettuare i collegamenti elettrici base per il suo funzionamento. Per prima cosa va collegata l’alimentazione (5V), poi il pin denominato MCLR (Negato) va collegato a Vcc. Tale pin è il Reset del PIC. A livello basso il pic viene resettato, a livello alto funziona normalmente.
Ultima importantissima cosa è il generatore di clock. Il dispositivi interni del PIC necessitano un segnale di sincronizzazione per funzionare correttamente. Il pic per esempio esegue un’istruzione ogni 4 cicli di clock. Il clock è un’onda quadra con duty-cycle 50% e frequenza fissa. Maggiore è la frequenza, maggiore è la velocità di lavoro del pic. Nel Datasheet sono indicate le frequenze massime di lavoro del pic e le frequenze di lavoro ottimali. Per generare il segnale di clock esistono diversi modi, ma quello più semplice è utilizzare un quarzo (oppure un oscillatore ceramico).
Tale componente viene connesso direttamente al pic attraverso i pin denominati OSC1 e OSC2. Esistono quarzi tarati a diverse frequenze, basta solo comperare quello adatto al nostro utilizzo. Una frequenza di 4Mhz va bene per tutti. Inoltre è necessario collegare dei condensatori ceramici (dell’ordine dei pF) tra ognuno dei pin del quarzo e GND. I valori di tali condensatori sono indicati nel datasheet del pic. L’utilizzo dei suddetti semplicemente ha lo scopo di impedire al quarzo di mettersi ad oscillare a frequenze diverse da quella nominale al momento dell’accensione. In caso questo si verifichi il programma del pic viene eseguito in modo anomalo.
Ora si può passare all’azione. Il pic all’interno hanno molti dispositivi avanzati per effettuare operazioni particolari: Vi sono pic con convertitori A/D, generatori di PWM, USART Hardware, convertitori D/A, interfacce USB….
Ogni modello è differente dall’altro, alcuni sono di semplice utilizzo, poco potenti, senza dispositivi avanzati come il 16F84A, mentre altri integrano moduli hardware come quelli indicati sopra.
Nel nostro progetto facciamo uso abbondante dei pic. Sono loro infatti che gestiscono l’intero sistema di funzionamento del dirigibile.

Ovviamente non potevamo accontentarci di pic economici perché molti compiti che dovevamo svolgere richiedevano moduli hardware particolari contenuti solo nei microcontrollori più avanzati.
Tutti i pic sono dotati di numerose porte di Input/Output, connesse ai rispettivi registri nella memoria ram. Tali porte possono essere usate per comunicare digitalmente con i dispositivi del circuito. Alcuni pin del microcontrollore possono essere usati anche per operazione specifiche, settando opportunamente i registri interni. Le operazioni specifiche sono gestite da moduli interni indipendenti, quelli di cui si parlava poco fa.
Nel nostro progetto abbiamo la necessità di molte porte digitali, per il collegamento con i vari sensori che comunicano per la stra-grande maggioranza serialmente o con impulsi pwm.
Inoltre ci sono dei motori da controllare, con eventuale possibilità di regolarli in velocità, quindi sono necessari pic con modulo pwm interno. Vedere in seguito l'utilizzo del pwm per il controllo dei motori.
Microcontrollori dotati di questa tecnologia integrano di solito anche moduli di comunicazione seriali hardware, detti UART (o USART), molto utili per la comunicazione avanzata tra dispositivi e per operazioni di debug.
Sebbene noi non utilizziamo per intero tutte le possibilità che il pic ci offre, abbiamo studiato i circuiti in modo tale che l'operatore possa utilizzare sia i componenti hardware indipendenti del dispositivo sia una normale routine software che esegue la medesima operazione sul pin impostato come i/o digitale.
In base all'utilizzo che ne abbiamo fatto ecco elencati i pic da noi utilizzati:

Gondola:

- **PIC18F452** - È un pic di ultima generazione, appartiene alla nuova famiglia Microchip (la ditta che li produce) 18.
  A differenza della serie precedente questi sono più performanti, possono arrivare a lavorare fino a 40Mhz, hanno una quantità di memoria (ram e flash) che può essere considerata "infinita" rispetto ai colleghi "serie 16". Il prezzo non cambia molto e si ha un microcontrollore che integra un gran numero di moduli hardware dedicati per operazioni particolari.
  Abbiamo scelto questo pic per svariati motivi: l'elevato numero di porte (è un 40pin), la presenza di un modulo pwm a 2 canali, la

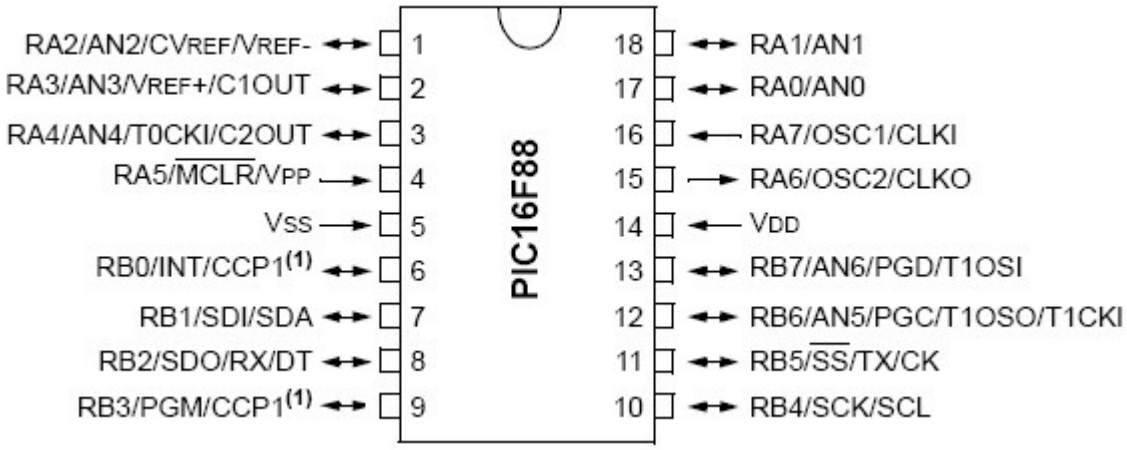
presenza di ingressi analogici (ADC), la presenza di una USART, e molto importante, la quantità di memoria disponibile per il programma e per le variabili.
Questa piccola belva infatti ha ben 16K di memoria flash (istruzioni a 16 bit), memoria dati separata di ben 256 bytes (EEPROM) e una ram da 768 Bytes (enorme!)



Le altre caratteristiche salienti sono indicate sul datasheet.

- **PIC16F88** - È anche questo un pic molto potente, se si tiene conto che il tutto è racchiuso in un package da 18pin. Appartiene alla serie 16 ma integra un buon numero di funzionalità extra: ha 1 uscita pwm, un modulo USART, svariati i/o digitali e può funzionare a velocità sostenute (20Mhz). Presenta anche la possibilità di funzionare attraverso l'oscillatore interno di cui è dotato, guadagnando così 2 canali digitali in più ed evitando il circuito oscillatore esterno. Il pic ha una memoria flash da 7168 Bytes, una EEPROM dati da 256 bytes e una Ram da 368 bytes.
La nostra scelta è caduta su questo pic perché esso è dotato di ampio spazio per il programma, importante perché è lui che deve leggere tutti i sensori autonomi (i sonar). Inoltre ci serviva un altro canale

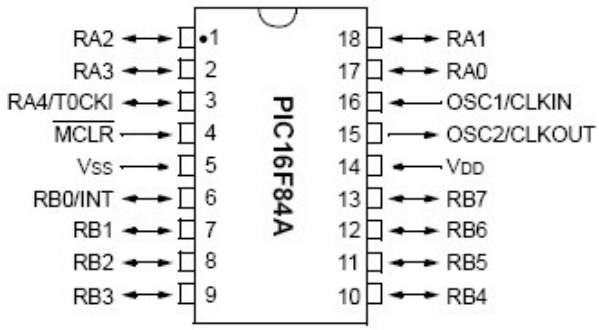
pwm per dare la possibilità all'utente di regolare anche la velocità dei motori ascensionali (quelli direzionali sono assegnati al 18F452).



Si consiglia di consultare il datasheet per l'elenco completo delle caratteristiche.

Modulo Telecamera:

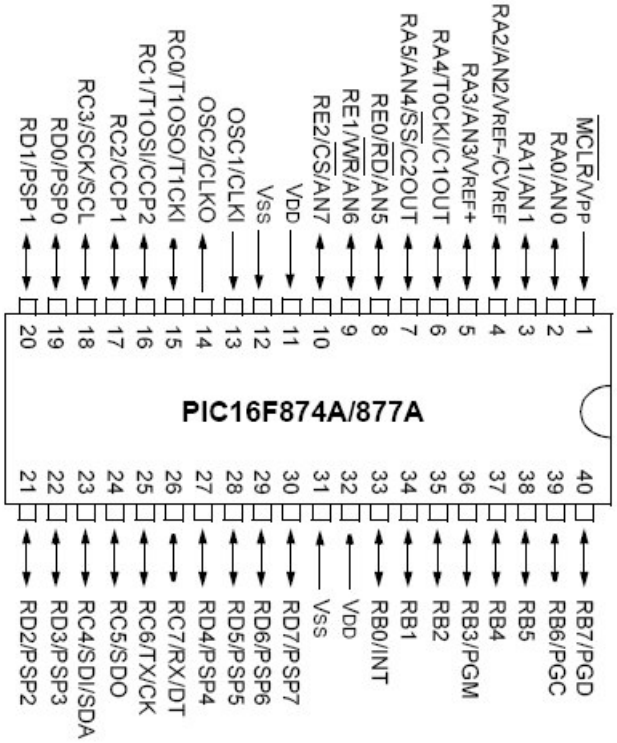
- **PIC16F84A** - Non penso che abbia bisogno di molte presentazioni. È il pic più conosciuto a livello mondiale. Successore del famosissimo 16C84 è uno dei pic più usati nelle scuole e per piccole applicazioni perché molto economico. Non presenta nessun dispositivo hardware interno per usi particolari, ma ha solamente porte digitali (13 canali in un package di 18 pin). Il pic ha inoltre poca memoria: 1024 words per il programma (istruzioni a 14 bit), 64 bytes di memoria dati (eeprom) e 68 bytes di memoria ram. Abbiamo usato questo pic perché era sufficiente per controllare l'hardware del modulo ciclopè. Infatti abbiamo bisogno solo di uscite ed ingressi digitali. Inoltre il programma è relativamente contenuto, quindi un pic come questo è l'ideale. Ricordatevi però questo "rottame" può arrivare alla velocità di 20Mhz nella revisione A!



Base Terrestre:

- **PIC16F88** - Ne abbiamo usati due. Uno per la gestione delle interfacce di comando e controllo e uno per l'interfacciamento e la lettura del joypad della playstation. La scelta è caduta di nuovo su questo modello proprio per l'elevata memoria associata ad un pic di piccole dimensioni (18 pin). Questo pic è semplicemente fantastico! Anche se non proprio economico…

- **PIC16F877A** - È uno dei modelli di testa della serie 16. è un pic da 40 pin, quindi integra un buon numero di canali digitali. Può lavorare alla velocità di 20Mhz ed integra diversi moduli dedicati: 2 canali pwm, modulo USART, modulo ADC, Modulo Comparatore…
  La scelta cade su questo pic per 3 motivi principali: L'elevato numero di pin di i/o, la presenza di convertitori A/D per la lettura delle tensioni ed infine l'elevata quantità di memoria.
  Questo pic infatti ha ben 14,3K di memoria programma (con istruzioni di 14 bit), 256 bytes di memoria dati e 368 bytes di memoria ram.

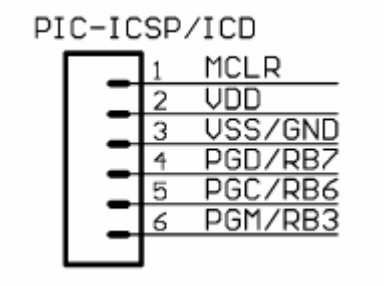


Consultare il datasheet per tutte le caratteristiche.

I moduli ultrasuoni acquistati e la bussola elettronica, come gran parte dei sensori con interfacce digitali in commercio, presentano un microcontrollore on-board in versione smd per il controllo del modulo e la comunicazione dei valori misurati. Di solito si tratta di pic della serie 12 o 16. Sono proprio dappertutto!

ICSP E PROGRAMMAZIONE DEL PIC

L'ICSP, acronimo di In Circuit Serial Programming, permette la programmazione in circuit: il pic rimane inserito nel circuito e può essere programmato, attraverso un cavetto che arriva dal programmatore ICSP, senza doverlo staccare continuamente (rovinandolo).
Attraverso un deviatore è possibile commutare lo stato del Pic tra Esecuzione e Programmazione.

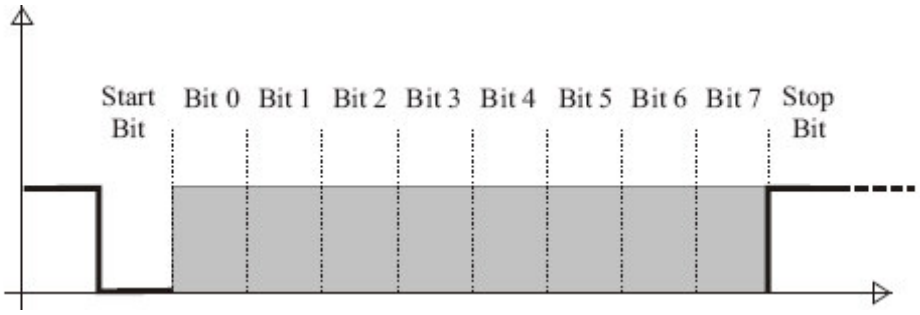


Il connettore ISCP presenta sei linee:

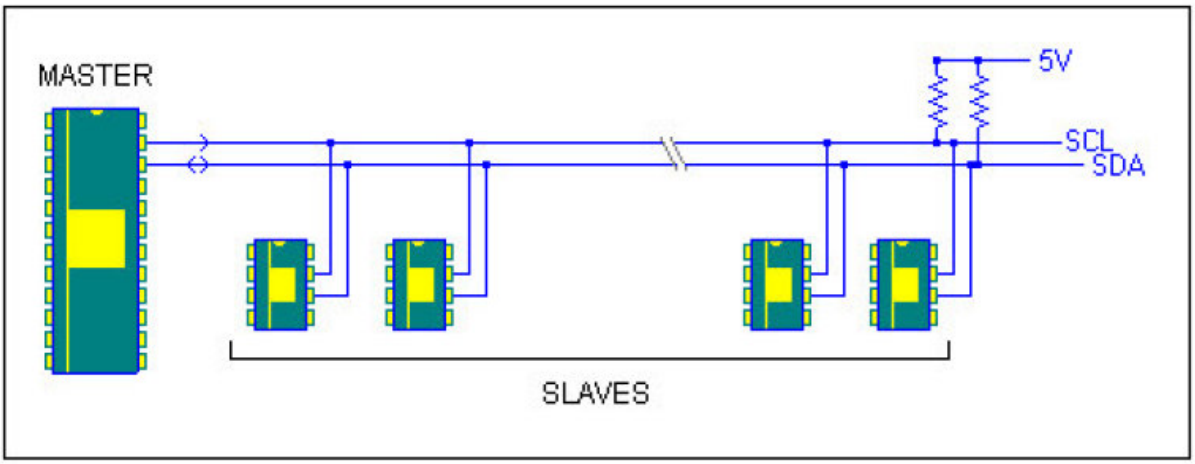
1. MCLR collegato ovviamente con quello del pic, porta l'ingresso alla tensione di programmazione di 13V;
2. VDD, l'alimentazione del pica 5V
3. VSS/GND la massa di riferimento per le tensioni;
4. PGD è la linea dati, ovvero dove transitano le informazioni relative al programma;
5. PGC è la linea clock, dato che l'ICSP è un tipo di programmazione seriale sincrona, scandisce la temporizzazione con cui vengono inviati i dati;
6. PGM viene usato per la programmazione in modalità "low voltage", che non richiede la tensione di programmazione di 13V. Per la programmazione classica questa linea va posta a GND.

COMUNICAZIONE INTERNA

Sia nella base terrestre che sulla gondola sono presenti più di un pic. Il problema adesso è farli comunicare tra loro.
Nella base terrestre il pic che si occupa del joypad deve comunicare i dati letti al pic che gestisce i dispositivi di comando. Quest'ultimo dopo aver elaborato i dati deve comunicarli al pic centrale (picmaster) che si occupa del link radio e della telemetria.
Sul velivolo esiste lo stesso problema. Il pic che legge i sensori deve comunicare all'altro i valori letti. L'altro pic che si occupa del controllo dell'aeronave deve comunicare al pic-sensori i parametri per il movimento dei motori ascensionali, gestiti da quest'ultimo.
Inoltre nel velivolo vi sono degli slot di espansione e ad uno di essi è connesso il modulo ciclopè. Il pic centrale deve comunicare anche con questo.
Per risolvere il problema dell'intercomunicazione abbiamo scelto due protocolli diversi, uno per la comunicazione tra i pic "di sistema" ed uno per la gestione del modulo della telecamera. Quest'ultimo è un dispositivo che accetta solo comandi, quindi non è necessario un collegamento bidirezionale. Proprio per questo abbiamo deciso di comunicare i comandi utilizzando semplicemente il protocollo seriale standard, con l'aggiunta di un canale digitale che indica al pic principale (livello alto) quando il modulo è pronto a ricevere i dati. La comunicazione avviene a livello software con velocità 2400baud, 8 bit, 1 bit di stop, nessuna parità.
La comunicazione seriale è un tipo di trasmissione dati che avviene utilizzando un solo canale. I singoli bit che compongono il "telegramma" dati vengono inviati uno per uno (serialmente) con una certa temporizzazione. Il dispositivo ricevente deve essere tarato sulla stessa base dei tempi per poter "decifrare" correttamente ciò che gli viene inviato. Il telegramma è di solito formato da 8 bit. Per indicare l'inizio della sequenza di bit è stata introdotta una condizione di start (chiamata bit di start) e per indicarne la fine vengono usati dei bit di stop, di solito sempre 1. Più telegrammi formano un "pacchetto" dati. La velocità di trasmissione (alias la base dei tempi) è espressa in baud (bit/s) ed indica in sostanza quanti bit si trasmettono al secondo (nel nostro caso 2400). Ogni bit quindi ha durata 1/2400 secondi.

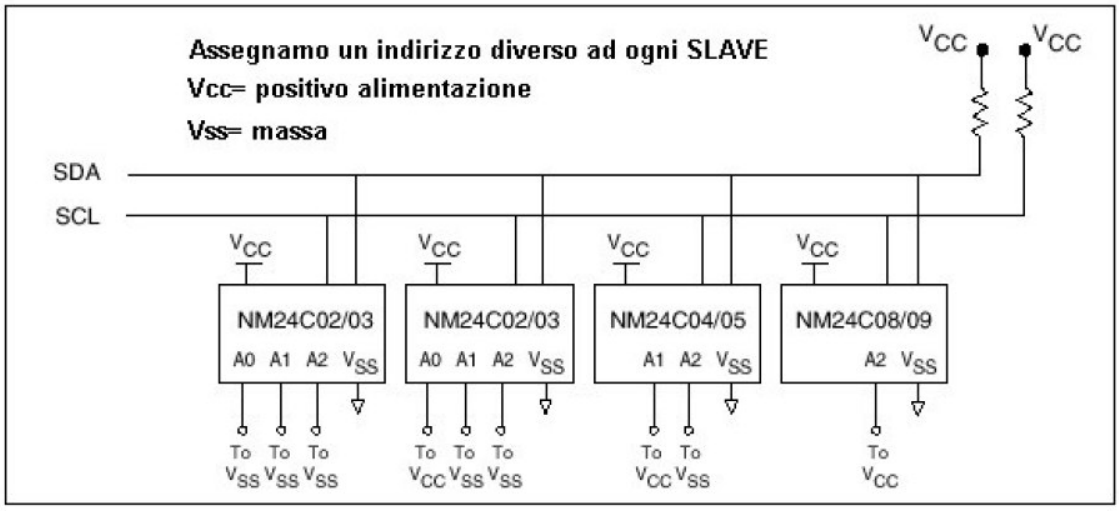


Per l'intercomunicazione dei pic invece la questione è ben più complessa: infatti avevamo la necessità di risparmiare pin di i/o, dovevamo mantenere una buona velocità e comunicare una grande quantità di dati.
La comunicazione è bidirezionale e questo crea già un problema. Poi non potevamo gestire le interruzioni perché il sistema diventava troppo complesso e lento, soprattutto per la base terrestre che ha 3 pic intercomunicanti. Proprio in questo ultimo caso diventerebbe complesso far comunicare ogni microcontrollore con gli altri 2: il primo deve essere connesso al secondo ed al terzo, il secondo al primo ed al terzo ed il terzo al primo ed al secondo.
L'unico modo era utilizzare un bus bidirezionale semplice: L'I2C è l'ideale.

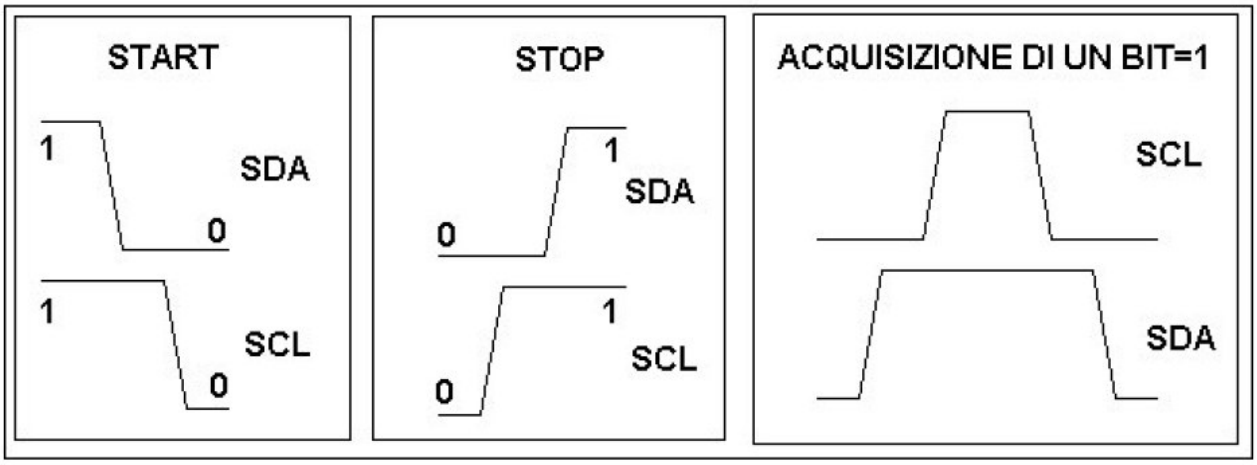


Il bus I2C (Inter Integrated Circuits) è nato presso la Philips, questo protocollo di comunicazione veniva usato nei televisori per far comunicare i vari dispositivi e le memorie interne. Adesso è uno dei sistemi più usati nella comunicazione tra sensori, memorie e microcontrollori.
Dai tempi delle origini il protocollo si è evoluto e oggi è possibile gestire un elevatissimo numero di dispositivi sulla stessa linea. Esistono integrati dalle funzioni più disparate che comunicano per mezzo di questo sistema:

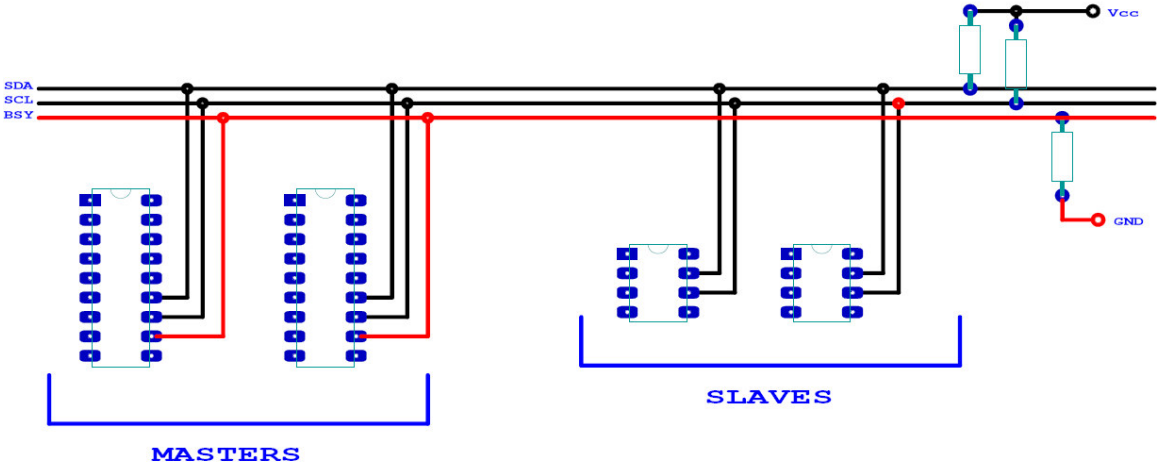
espanitori di uscite/ingressi, convertitori A/D e D/A, timer, calendari e anche circuiti integrati per espandere il bus. Ora il bus può anche essere ramificato in sotto-linee.
Il protocollo si basa su un sistema di trasmissione sincrono: Vi è un canale dati (SDA) ed un canale clock (SCL) per la temporizzazione.
Per funzionare il circuito richiede due resistenze di pullup da 4,7KOhm collegate a Vcc, una sull'SDA e l'altra sull'SCL. L'I2C un sistema di tipo Master-Slave, quindi è centralizzato. Il pic Master da la temporizzazione agli slave e decide se deve essere eseguita un'operazione di lettura o scrittura del dispositivo indirizzato (Slave). Ovviamente sul bus c'è un solo master e tanti slave. Ogni slave è contrassegnato da un indirizzo diverso, di solito impostabile tramite collegamenti elettrici sul dispositivo slave.



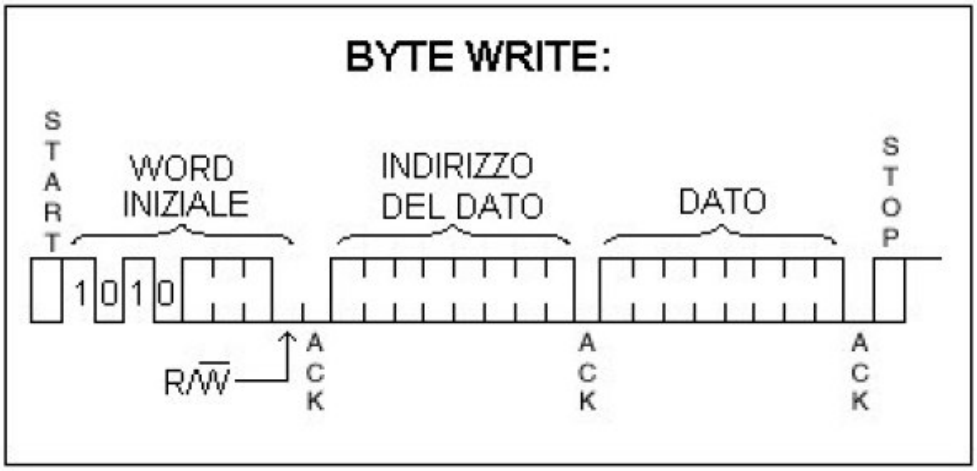
Quando il pic Master accede al bus scrive dapprima un telegramma di comando contenente l'operazione da svolgere (read/write), il codice identificativo del tipo di dispositivo slave (diverso a seconda del dispositivo slave) e il suo indirizzo. Solo lo slave che ha lo stesso indirizzo accetta i dati inviati (a gruppi di byte) successivamente al telegramma di comando.
La velocità di comunicazione può essere compresa tra i 100Khz e 400Khz (i valori indicano la frequenza del clock).
Ogni telegramma è composto da una condizione di start ed una di stop. In mezzo vi sono i dati. Tutte le operazioni sono svolte basandosi sul segnale di temporizzazione. Il dispositivo slave risponde con un ACK ogni telegramma (8 bit).

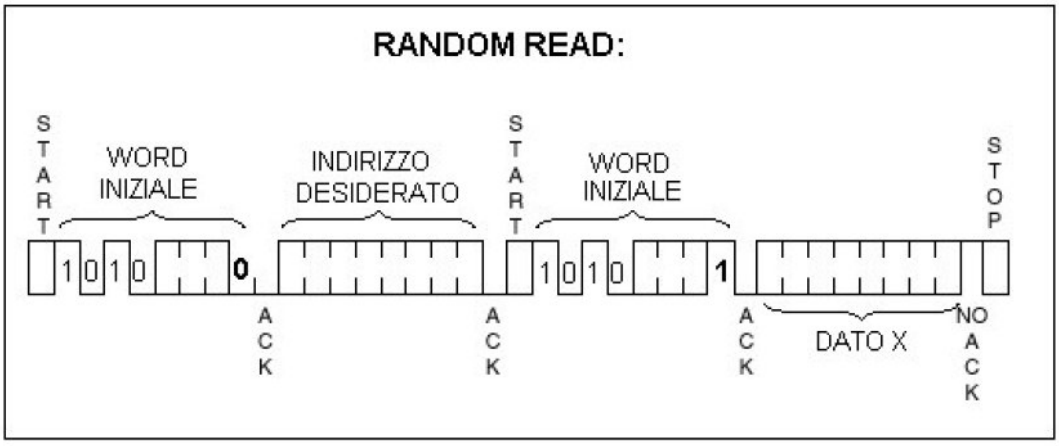


Per maggiori dettagli vedere il tutorial.
Attraverso il bus quindi si può comunicare bidirezionalmente, però il pic è sempre un dispositivo master. Quindi come vare a far comunicare due dispositivi master? Il problema si risolve in due passi: Viene inserita una memoria ram I2C (che è slave). Tale memoria viene letta e scritta dai pic Master attraverso il bus. Basterà leggere o scrivere (a seconda della soggetto dell'operazione) le stesse locazioni della memoria. Quindi il pic 1 legge sempre gli ultimi dati resi disponibili dall'altro pic. Questo incrementa notevolmente la velocità di comunicazione e ottimizza la gestione dei dati. Se ci sono 3 pic connessi alla stessa memoria ecco che ognuno "sa" i dati degli altri. Proprio come si voleva. La memoria Ram è quindi usata come SWAP.
Il secondo passo è la gestione dell'accesso al bus. Se vi sono più Master e due di questi tentano di accedere al Bus nello stesso istante l'unica cosa che ne ricaviamo è un grande cortocircuito sui pic.
Per evitare questo e gestire la priorità e l'accesso al bus tra i vari master è stato aggiunto un terzo canale chiamato BUSY. Quando un pic accede al bus alza il BUSY, indicando agli altri che il bus è occupato. Gli altri dispositivi Master attenderanno che il bus si liberi prima di tentare accedervi. Il controllo deve essere fatto con opportune temporizzazioni in modo che non vi sia in alcun modo la possibilità di collisioni tra i telegrammi.



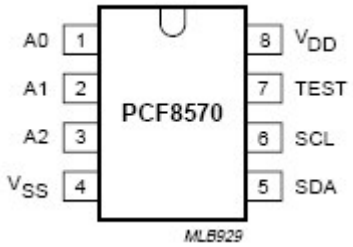
In questo modo abbiamo risolto i problemi ed inoltre con lo stesso bus possiamo leggere sensori ed altri dispositivi. La bussola elettronica si basa sul protocollo I2C per esempio. Nel circuito abbiamo inserito anche una memoria EEPROM per il salvataggio di eventuali dati non volatili e abbiamo portato i tre canali SDA, SCL e BUSY a tutte le espansioni.
Le locazioni delle memorie e quelle contenenti i valori della bussola devono essere indirizzate: il primo o i primi bytes dati inviati sul bus corrispondono all'indirizzo della locazione puntata mentre i successivi sono i dati letti o scritti dalla/nella locazione a seconda dell'operazione in corso (lettura/scrittura).



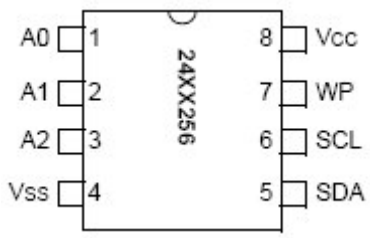


Le memorie da noi usate sono:

- **RAM PCF8570** - È prodotta dalla Philips e contiene 256 locazioni da 8 bit, per un totale di 256K. È contenuta in un package da 8 pin. L'indirizzo del dispositivo è impostabile attraverso 3 pin.
  I consumi sono bassissimi e può essere alimentata a tensioni relativamente ridotte. Il tempo di scrittura del dato nella locazione è dell'ordine dei microsecondi (us).



- **EEPROM 24LC256** - È prodotta dalla Microchip ed ha una configurazione 32K x 8 (locazioni 8 bit) per un totale di 256Kbyte di spazio. L'indirizzo del dispositivo è impostabile attraverso 3 pin (package 8 pin), può lavorare su bus fino a 400Khz, i consumi sono ridotti e presenta un'elevata immunità ai disturbi elettromagnetici.
  Il tempo di scrittura è dell'ordine delle decine di millisecondi (ms)

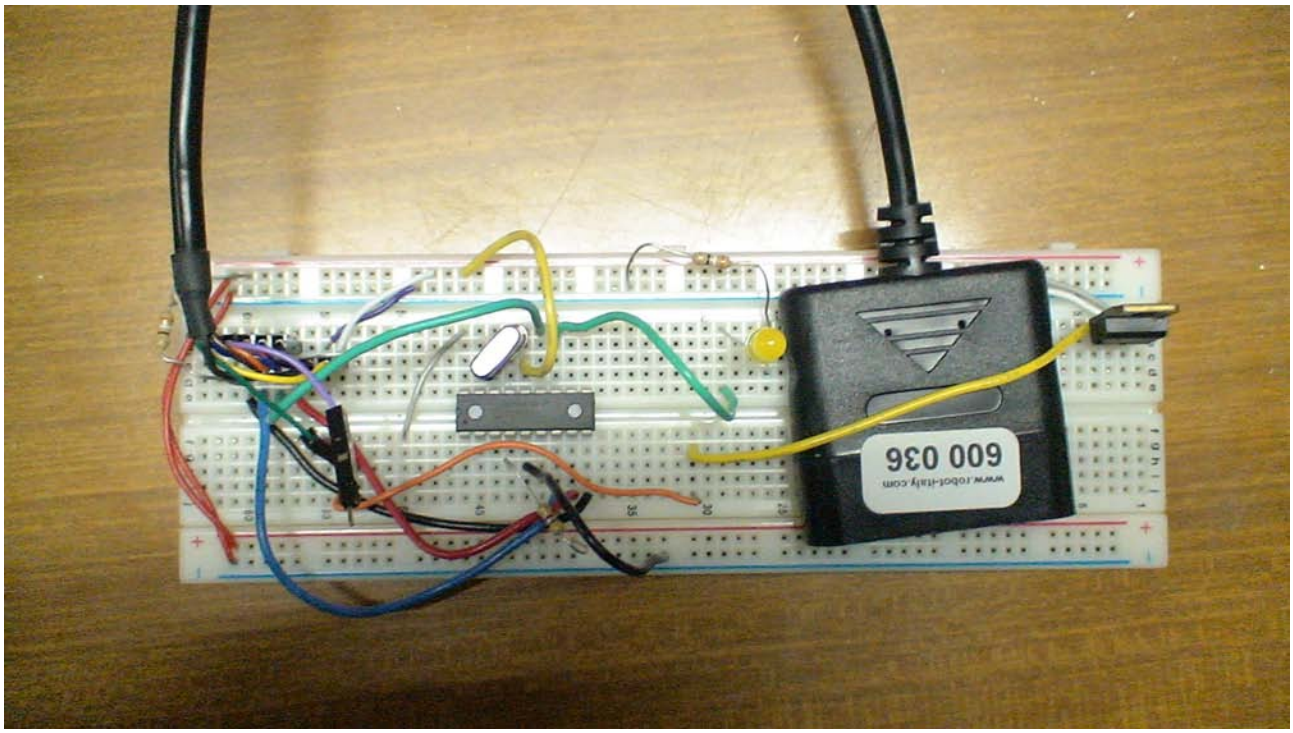


Vedere il datasheet per le caratteristiche dettagliate.

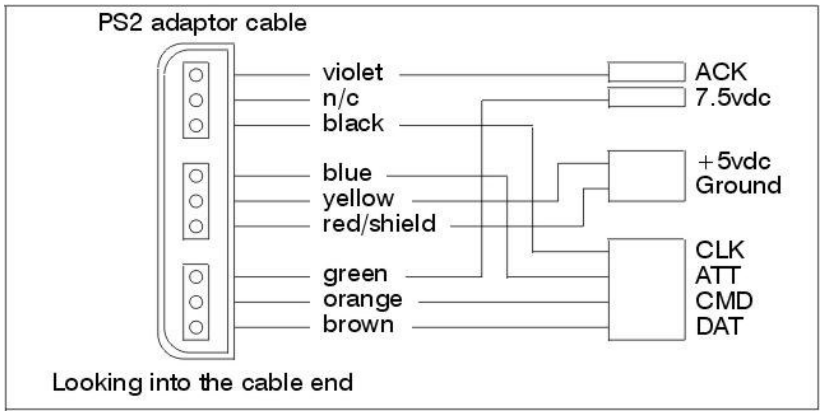
INTERFACCIA PSX

Per pilotare il dirigibile, oltre che con il computer, abbiamo pensato di usare un gamepad. Così sarebbe stato divertente come giocare con i videogiochi!

Abbiamo scoperto nel joypad della playstation tutte le qualità richieste e molto di più: Il gamepad presenta una lunga serie di pulsanti per impartire comandi speciali con il dirigibile, dei pulsanti direzionali per direzionare la telecamera e due leve analogiche proprio ottime per il pilotaggio dei motori direzionali. Inoltre il joypad può vibrare, questo è un ottimo mezzo di segnalazione oltre che un bellissimo “effetto speciale”.
Già l’anno scorso siamo riusciti a capire il funzionamento del joypad. Abbiamo avuto difficoltà all’inizio nel capire tutti i dettagli sulla comunicazione del dispositivo. Grazie a diversi tutorial scaricati dalla rete siamo riusciti a leggere manualmente il dispositivo, successivamente lo abbiamo interfacciato ad un pic e grazie ad un programma scritto appositamente lo possiamo leggere (i valori dei pulsanti) e scrivere (settaggi e vibrazione).



Il joypad viene alimentato a 5V ed i motori per la vibrazione interni funzionano ad 8V. Per cui l'alimentazione si compone in +5V, +8V, GND. La comunicazione in sostanza avviene in modo sincrono ed è gestita da quattro segnali: il segnale ATT è pilotato dal pic ed è l'enable del joypad. In poche parole è il segnale per accendere e spegnere la comunicazione con il dispositivo. Il segnale CLK è il clock della seriale sincrona (temporizzazione) per la comunicazione con il dispositivo. Il segnale CMD è il flusso dati serialmente inviato al joypad, in poche parole i comandi di configurazione del dispositivo. Il segnale DAT invece è il flusso dati serialmente ricevuto dal joypad, in pratica i dati dei pulsanti, delle leve analogiche e delle configurazioni attive del dispositivo.ù
Un ulteriore segnale, l'ACK può essere usato per verificare che i dati trasmessi arrivino correttamente a destinazione.



Perché il circuito funzioni è necessaria una resistenza di pullup da 1KOhm sul canale DAT ed una dello stesso valore sul canale ACK (se utilizzato). Il joypad comunica tutti i valori con logica binaria inversa, quindi è necessario negare tutti i bytes ricevuti. I valori delle leve analogiche sono comunicati solo se si è nella modalità analogica del joypad e vanno da 255 a 0 (dall'alto al basso e da destra a sinistra), mentre tutti i pulsanti sono attivi alti. Per la regolazione della vibrazione esistono un byte di comando per la potenza del motore vibrante grosso ed un bit per abilitazione del motore vibrante piccolo.
Consultare i tutorials per maggiori dettagli.

```
Standard Digital Pad

    BYTE    CMND    DATA

     01     0x01    idle
     02     0x42    0x41
     03     idle    0x5A    Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7
     04     idle    data    SLCT           STRT UP   RGHT DOWN LEFT
     05     idle    data    L2   R2    L1   R1   /\   O    X    []


Analogue Controller in Red Mode

    BYTE    CMND    DATA

     01     0x01    idle
     02     0x42    0x73
     03     idle    0x5A    Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7
     04     idle    data    SLCT JOYR JOYL STRT UP   RGHT DOWN LEFT
     05     idle    data    L2   R2   L1   R1   /\   O    X    []
     06     idle    data    Right Joy 0x00 = Left  0xFF = Right
     07     idle    data    Right Joy 0x00 = Up    0xFF = Down
     08     idle    data    Left Joy  0x00 = Left  0xFF = Right
     09     idle    data    Left Joy  0x00 = Up    0xFF = Down

Tutti i pulsanti attivi bassi e le leve analogiche invertite durante la fase di
lettura. L'inversione di logica viene fatta nel software.
```

Noi abbiamo configurato i tasti del joypad nel seguente modo:
Tasti Croce, Triangolo, Cerchio e Quadrato sono tasti funzione per attivare funzioni particolari del velivolo.
I tasti Start e Select vengono usati per la conferma dei comandi funzione.
I tasti R1 e R2 vengono usati rispettivamente per il comando della microcamera e del laser del modulo ciclopè.

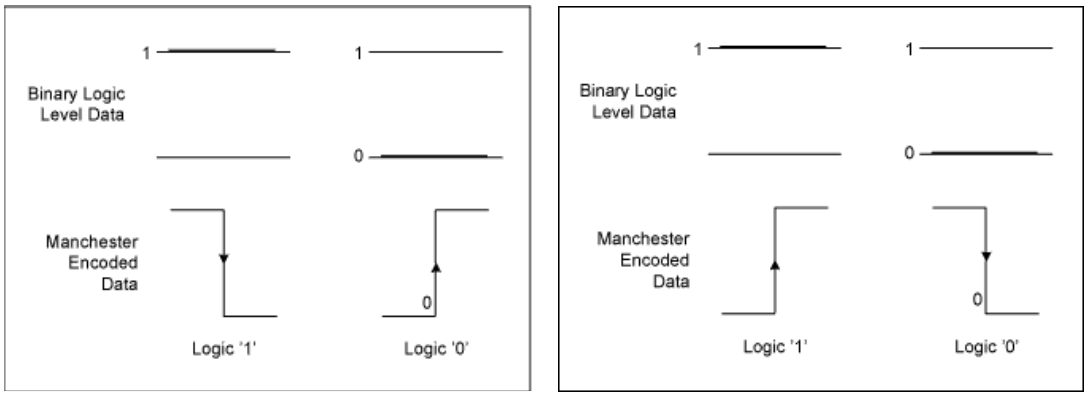
Le freccette direzionali sono usate per regolare la posizione della microcamera.
I tasti L1 e L2 sono usati per la regolazione ed il comando dei motori ascensionali.
Le leve analogiche sull'asse y sono usate per il pilotaggio dei motori direzionali sinistro e destro.
La modalità analogica è sempre attiva e non è possibile cambiarla.
Anche la vibrazione è sempre attiva e regolabile attraverso gli appositi valori di controllo.
Il pic che si occupa della gestione del joypad è inserito nel bus di comunicazione I2C ed è un modulo a se stante. Il microcontrollore legge il Joypad, genera i bytes di comando in base ai tasti e scrive tali valori in ram, legge la ram e imposta la vibrazione.
La gestione avanzata dei dati in ingresso e in uscita dell'interfaccia PSX è compito degli altri pic presenti nel circuito, in particolar modo del pic gestore dei dispositivi di comando e comunicazione.

COMUNICAZIONE RADIO

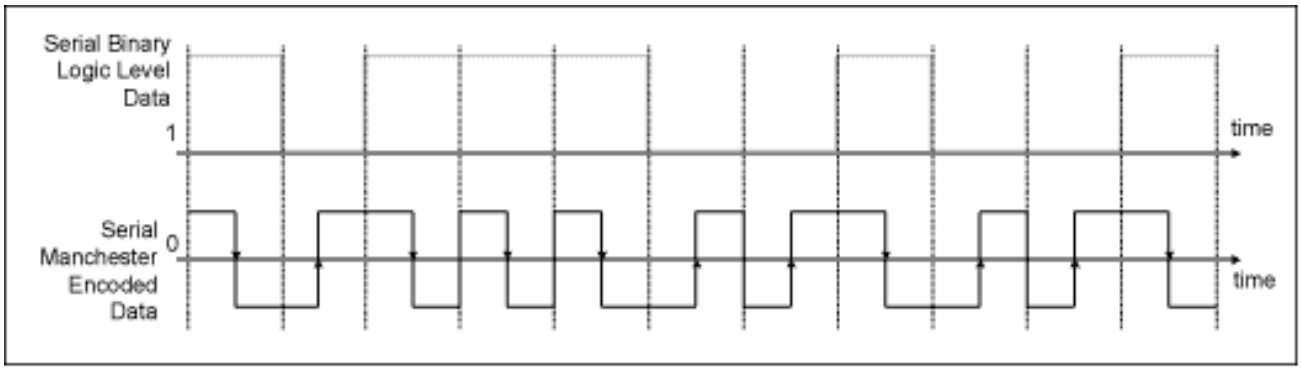
Il nostro dirigibile deve poter comunicare con la base terrestre.
La base invia i comandi: motori, moduli, avviamento sottoprogrammi automatici…
Il dirigibile riceve i comandi, legge tutti i sensori ed effettua un'elaborazione prima di attuare i comandi.
Sia i valori dei sensori che gli esiti delle operazioni (flags di stato) vengono rispediti alla base terrestre.
I dati ricevuti a terra, contenenti la telemetria vengono interpretati e visualizzati tramite i dispositivi di segnalazione: led, display LCD e computer.
La comunicazione deve essere quindi bidirezionale, perché i dati sono trasmessi in entrambe le direzioni.
Caratteristica principale del nostro progetto è il controllo remoto. Il dirigibile è un velivolo che, spostandosi nell'aria, non deve avere connessioni dati via cavo. Proprio per questo è necessaria una comunicazione radio.
Abbiamo previsto due sistemi di comunicazione radio:

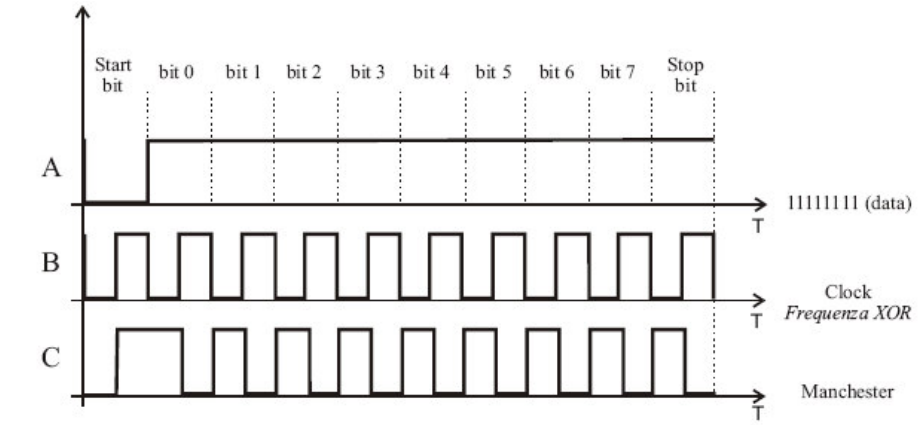
1. Controllo per mezzo di moduli ibridi (deve essere realizzata una codifica)
2. Controllo per mezzo di moduli RS-232 Embedded (Easy Radio) che permettono una comunicazione rapida e sicura.

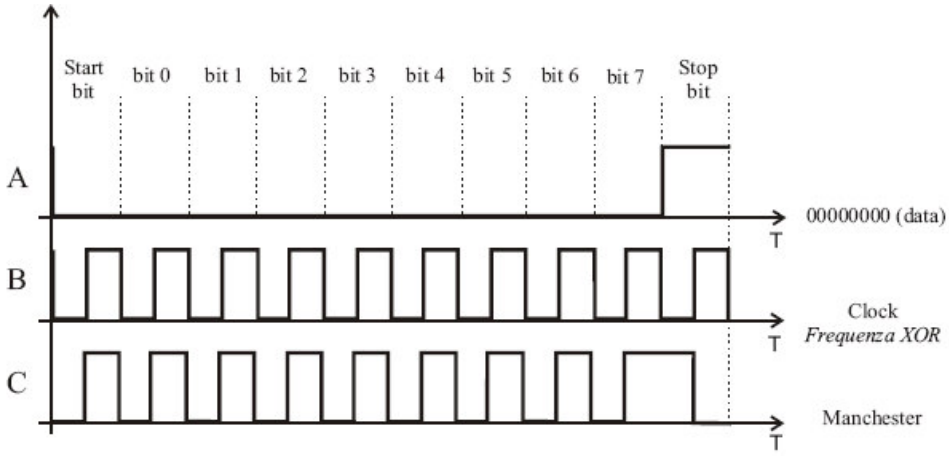
La comunicazione dati in generale si può suddividere in vari livelli:
al livello base c'è la codifica, che in sostanza è il modo in cui vengono rappresentati i dati che dobbiamo inviare o ricevere. Parlando di codice è quindi sottointeso che servirà un dispositivo (ed un algoritmo) che permetterà di codificare e decodificare i nostri dati.
Per la comunicazione radio il sistema di codifica più usato è il "Manchester". La codifica Manchester definisce i valori logici "1" e "0" come una transizione piuttosto che un livello digitale statico.
I decodificatore quindi presterà attenzione ai fronti di salita e di discesa per ricostruire i dati trasmessi.



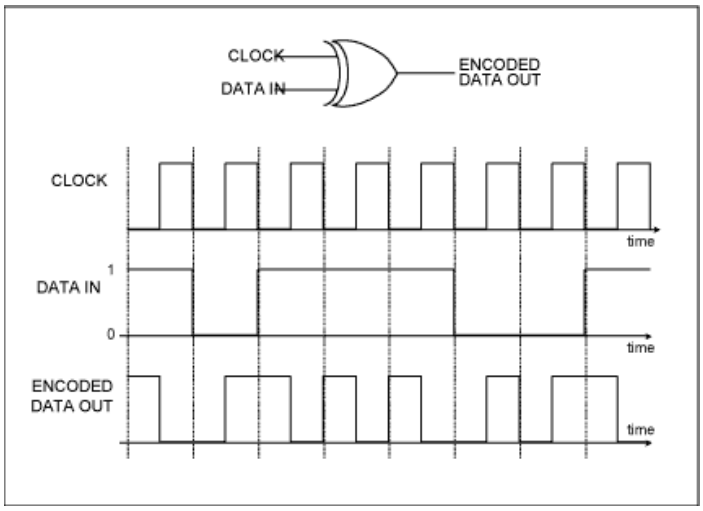
Una sequenza di bit codificati in Manchester richiede due livelli per ogni transizione perchè dalla definizione l'informazione è codificata come una transizione da un livello basso ad uno alto o viceversa. A differenza del protocollo seriale standard, la stessa sequenza di dati codificati include oltre al dato effettivo anche l'informazione necessaria a ricostruire il clock usato per l'operazione di codifica. Se si facesse uso del protocollo seriale RS-232 standard, in caso di un treno di bit tutti alti o tutti bassi potrebbero verificarsi dei problemi in fase di ricezione:
In caso venga persa la sincronizzazione (clock ricezione "non allineato" con il clock di trasmissione) infatti non è possibile ricostruire il dato trasmesso. Questo problema è accentuato dal fatto che non vi sono transizioni nel caso vi siano tutti i bit a "1" o tutti i bit a "0".
Con la Manchester si risolvono entrambi i problemi, dato che è impossibile avere una condizione di assenza di transizioni ed il clock è "allegato" al treno di dati.



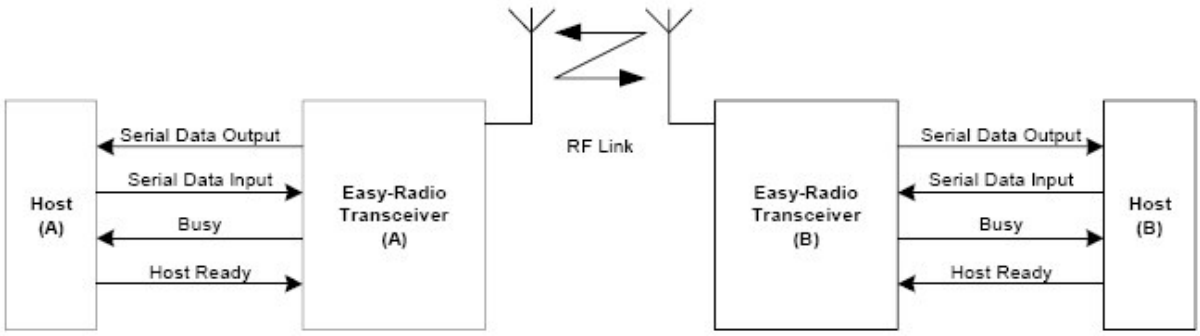




È possibile convertire un segnale seriale RS-232 in Manchester e viceversa utilizzando componenti logici base come ad esempio le porte logiche.

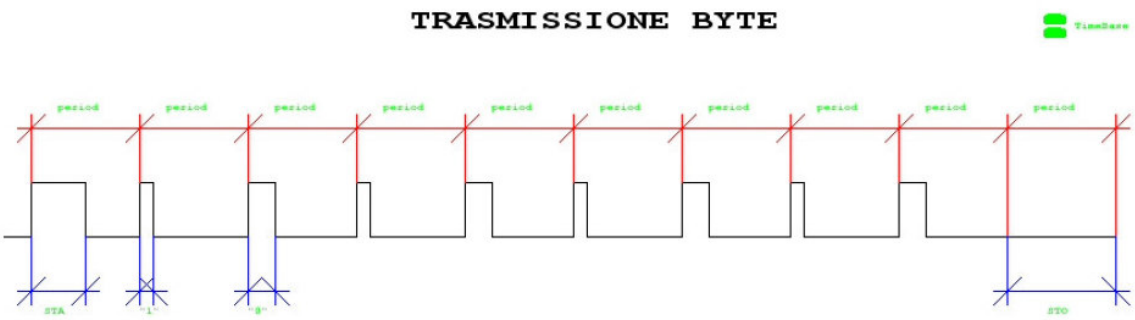


Il primo livello della comunicazione è il protocollo di trasmissione, che nel nostro caso deve essere per forza seriale. Il protocollo stabilisce il modo in cui i dati vengono inviati e si occupa di gestire l'inizio e la fine dei vari telegrammi (sequenze di bit) tramite apposite condizioni di start e di stop. Picbasic non gestisce un protocollo basato sulla codifica Manchester, inoltre sarebbe un'operazione troppo lunga creare un algoritmo per gestire tutti gli errori che si potrebbero verificare in una trasmissione radio.
La nostra scelta è caduta sui moduli Transceiver (Bidirezionali) Easy-Radio prodotti dalla LPRS (Low Power Radio Solutions).
Tali moduli comunicano con il microcontrollore attraverso il protocollo RS-232, gestito in maniera ottima dal compilatore. I dati inviati al modulo vengono messi in un buffer di trasmissione ed inviati utilizzando la codifica Manchester. È il modulo che si occupa della comunicazione, della codifica/decodifica dei dati e della gestione errori grazie ad un processore interno con algoritmo dedicato proprietario.



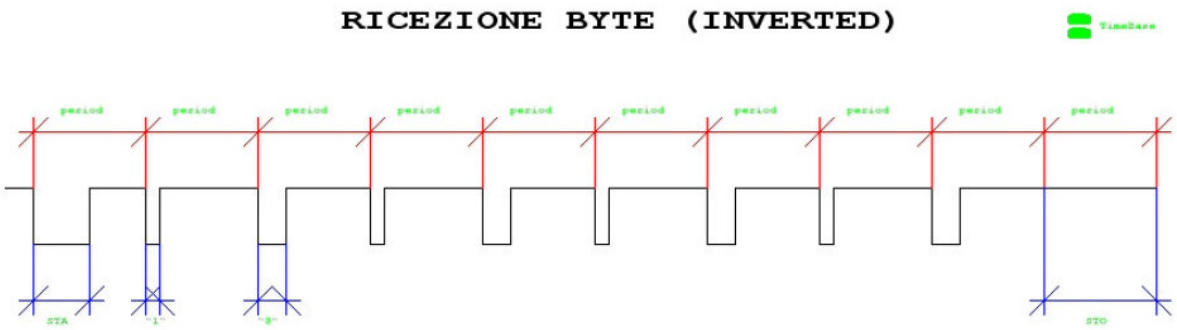
In tal modo la comunicazione radio tra la base terrestre ed il velivolo si semplifica notevolmente: basta solo occuparsi della struttura del pacchetto e della suddivisione dei vari telegrammi (livello 2) perché al resto pensa tutto il modulo easy-radio, con il quale è difficile perdere dati per strada. Il buffer di cui è dotato memorizza anche i dati in ricezione, che verranno inviati al microcontrollore quando richiesti.
Nonostante la scelta effettuata, è rilevante il fatto che abbiamo comunque prima tentato di sviluppare un nostro protocollo di comunicazione che a prima vista sembra simile come struttura ad uno con basato sulla codifica Manchester, ma il codice da noi realizzato è completamente diverso da quest'ultimo.
Le prove sono state fatte con dei moduli Ibridi prodotti dalla Aurel, società italiana Leader nel campo del controllo remoto. Per l'esattezza sono stati

un modulo ricevitore con alto guadagno ed un trasmettitore di tipo "booster" per lunghe portate. Due moduli accoppiati e gestiti attraverso una circuiteria logica a transistor permettono di stabilire un link bidirezionale. Per l'alimentazione del modulo boosterizzato si fa uso di un convertitore DC-DC (che porta la tensione di 5V ai 12V-14V necessari al funzionamento del modulo).
Abbiamo scritto un programma in picbasic che si occupa della trasmissione e della ricezione dei dati.
Nel nostro sistema di codifica il valore logico dei dati è assegnato al duty-cycle di un'onda quadra a periodo fisso.
A seconda della lunghezza (durata) dell'impulso alto (duty-cycle) vengono discriminati 4 tipi di bit: Il bit logico "1", il bit logico "0", il bit di start ed infine il bit di stop (ordine crescente).
Il risultato è un treno di impulsi dalla distanza variabile sulla linea temporale.



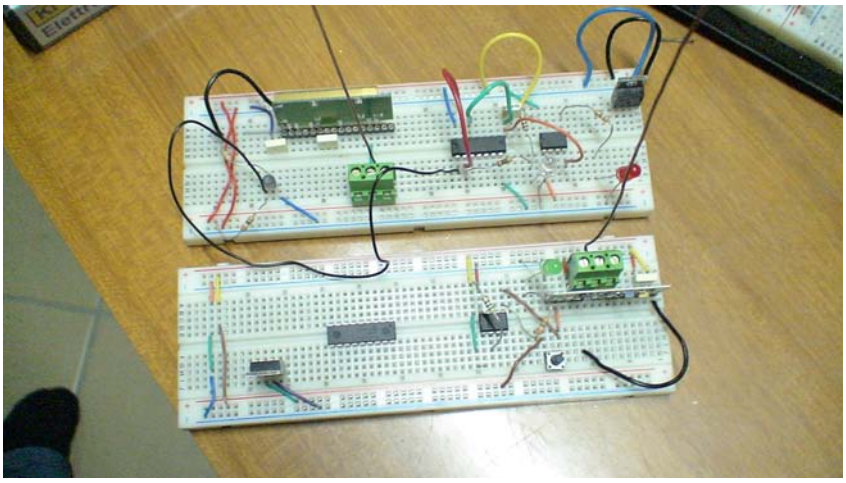
Per la fase di ricezione abbiamo interposto tra il ricevitore ed il pic un inverter (realizzato sempre con un transistor).L'inverter amplifica il segnale in uscita dal modulo, ma inverte la logica dei dati trasmessi.
Gli impulsi da considerare sono quindi quelli bassi, ricordando che la condizione di pausa tra un impulso e l'altro (tempo morto) è ora un livello alto (in trasmissione era basso).
L'inversione del dato non è comunque un effetto indesiderato, ma una garanzia studiata apposta per ridurre possibili errori dovuti ad interferenze magnetiche: Il pic è un dispositivo molto sensibile alle interferenze di questo che potrebbero creare variazioni sul valore logico in ingresso al microcontrollore.
Se il livello di pausa è mantenuto basso è più facile che impulsi non desiderati vengano interpretati dal programma come un errore in ricezione, ma se manteniamo il livello alto, le interferenze possono incidere solo al

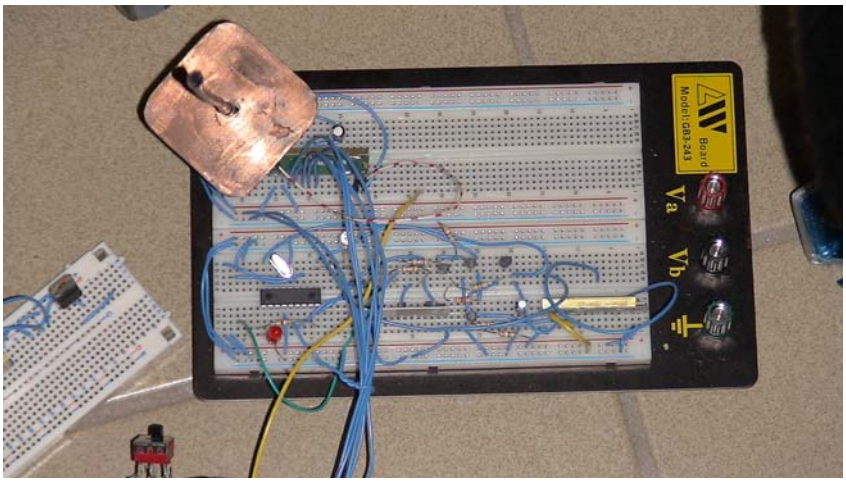
momento dell'impulso effettivo che è basso. Nella sequenza ricevuta il tempo morto è in quantità maggiore rispetto alla somma di tutti gli impulsi, quindi di fatto si riduce la percentuale di errori dovuti ai disturbi.



La decodifica dei dati avviene semplicemente misurando la durata degli impulsi in ingresso ed a seconda di quella vengono discriminati i 4 tipi di bit descritti in precedenza e ricostruito il telegramma.
Mentre viene usato il comandi PicBasic "Pulsout" per generare l'impulso della lunghezza desiderata, attraverso il comando "pulsin" viene calcolata la lunghezza degli impulsi in ingresso, ricevuti dal modulo.
Questo protocollo non si basa sul clock per decodificare i dati, ad esempio se anche il periodo fosse variabile al programma non farebbe alcuna differenza. Il tempo morto è stato scelto lungo solo per dare tempo al pic in ricezione di elaborare gli impulsi ricevuti.
Utilizzando il nostro protocollo è facile perdere dati o ricevere dati non corretti perché l'unico check errori che abbiamo implementato è quello di parità. Il programma segnala anche eventuali errori di lettura degli impulsi: in caso di ricezione di impulsi "troppo lunghi" o "troppo corti" il telegramma viene ignorato.
Durante i test effettuati il nostro protocollo si è dimostrato abbastanza efficace, ma su corte distanze. Quando la distanza aumenta e diminuisce il segnale portante le interferenze e la distorsione del segnale RF cominciano a creare errori di interpretazione dei livelli. Considerando che i moduli sono AM abbiamo raggiunto un bel traguardo: negli FM infatti si risente molto meno dei disturbi esterni, perché i dati vengono modulati in frequenza e non in ampiezza.
Non avendo la possibilità di effettuare prove con moduli FM e di dedicare ulteriore tempo allo sviluppo del nostro sistema di comunicazione abbiamo

preferito usare i semplici moduli easy-radio per avere la certezza di non perdere informazioni preziose durante lo scambio di dati con il dirigibile. Nonostante questo abbiamo realizzato l'hardware, sia della base che del velivolo, compatibile con entrambe le configurazioni: Moduli Ibridi (con transistors) e Modulo Easy-Radio.
L'unica cosa da fare è impostare sulla schedina la configurazione desiderata attraverso appositi Jumpers e provvedere poi all'interpretazione dei dati all'interno del Pic (è li che va messo l'algoritmo da noi creato).
Lo studio dei sistemi di comunicazione radio e la realizzazione del protocollo hanno richiesto più di 5 mesi di lavoro. I circuiti sono stati sviluppati e testati solo nella fase sperimentale del progetto, quindi il velivolo non è stato equipaggiato con i moduli ibridi, data anche la non completezza dell'algoritmo.

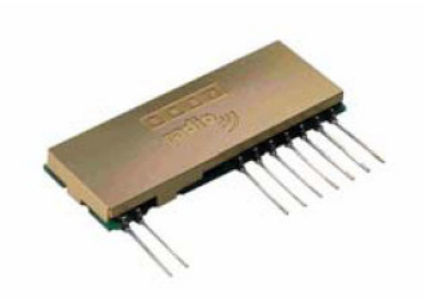


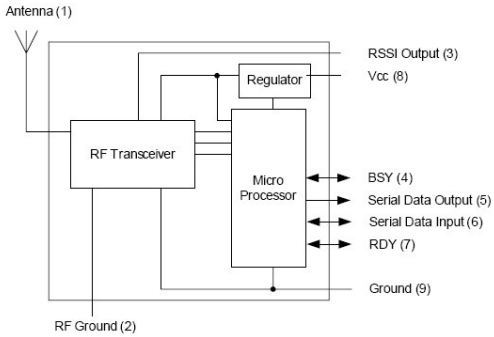


Tuttavia con questo scritto si vogliono elencare anche i problemi e le soluzioni che abbiamo adottato durante la sperimentazione dei suddetti

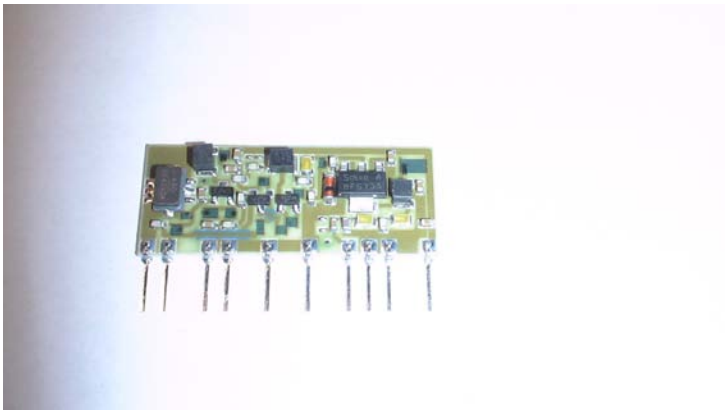
moduli, considerando che il passaggio all'Easy-Radio è avvenuto solo successivamente a questo capitolo.
I moduli radio usati sono questi:

- **ER400TRS (revisione -02)** prodotto dalla LPRS, lavora sulla frequenza 433Mhz in FM, potenza 10mW (portata 250m linea d'aria), bassi consumi, bidirezionale, elevata immunità ai disturbi, alta velocità di trasmissione con possibilità di criptare i dati, firmware proprietario aggiornabile e configurabile da PC (attraverso comandi AT via seriale), controllo interno degli errori sui dati ricevuti, uscita RSSI (livello segnale)… Queste sono solo alcune delle sue caratteristiche, vedere il datasheet per sapere tutto su questo gioiellino.





- **TX433BOOST** prodotto dalla Aurel, lavora sulla frequenza di 433Mhz in AM, potenza massima 400mW (fino a oltre 1Km di portata in linea d'aria), oscillatore SAW, doppia alimentazione (logica + stadio booster). Consultare il datasheet per la descrizione dettagliata delle caratteristiche.



- **RX-AM4SF** prodotto dalla Aurel, lavora sulla frequenza di 433Mhz in AM, oscillatore SAW, elevata immunità ai disturbi, elevato

guadagno, bassi consumi, uscita RSSI, alimentazione pre-amplificatore interno separata. Consultare il datasheet per la descrizione completa delle caratteristiche

ANTENNE

Le antenne sono un dispositivo tipicamente metallico, per la trasmissione e la ricezione di onde radio. Sono un elemento fondamentale di un sistema di trasmissione/ricezione in quanto hanno il compito di trasformare il segnale elettrico in ingresso in un segnale elettromagnetico trasmissibile nell'etere e dualmente di trasformare un segnale elettromagnetico proveniente dall'etere in uno elettrico. Le antenne possono comportasi come dei veri e propri amplificatori di segnale.

La forma, le dimensioni, la grandezza e il materiale con cui è costruita un'antenna ne determinano le sue caratteristiche di radiazione (direttivita, guadagno..) e cioè come avviene la trasformazione del segnale. Tutte le antenne sono progettate per operare su particolari range di frequenza.

Vi sono alcuni importanti parametri che caratterizzano le antenne:

1. *Valore di impedenza*

Ovvero un numero complesso che rappresenta l'equivalente in corrente alternata della resistenza in corrente continua. Quindi vi deve essere un adattamento di impedenza tra antenna e cavo (o tra qualsiasi altro componente) per permettere di raggiungere il massimo trasferimento di potenza all'antenna. Per rendere ciò possibile si deve fare in modo che l'impedenza dell'antenna sia uguale all'impedenza del cavo.
Qualora ciò non avvenisse si verifica una discontinuità chiamata interfaccia, cioè parte della onda non riesce ad arrivare all'antenna, viene riflessa e torna verso il trasmettitore. Ciò accade anche per un'onda in ingresso.

Vi sono valori di impedenza standard:

- 50 ohm: antenne WiFi, reti locali con cavo coi assale, strumenti di misura;
- 75 ohm: antenne televisore o radiofoniche.

2. *Frequenza di lavoro.*

Sappiamo che l'impedenza varia a seconda della frequenza, quindi un'antenna progettata per lavorare in una certa banda avrà una certa impedenza che cambia al variare della frequenza, cosi si verifica il fenomeno di riflessione.

Se un’antenna presenta un’impedenza diversa da quella voluta esistono vari accorgimenti per poterne variare il valore: adattatori di impedenza, cavi calibrati, trasformatori, ecc…

3. Guadagno di un’antenna.

Indica di quante volte è maggiore l'intensità di radiazione dell’antenna considerata, rispetto all’antenna isotropa (antenna ideale di riferimento), in un determinato punto. Il guadagno va inteso come una vera e propria amplificazione come potrebbe essere quella di un amplificatore attivo. E’ importante notare che tale guadagno entra in gioco in trasmissione ma anche in ricezione. Molte spesso, per aumentare le performance del sistema, si tende ad aumentare la potenza erogata dal trasmettitore (migliorando le cose solo in trasmissione), mentre sarebbe preferibile utilizzare antenne con guadagno più elevato migliorando così il sistema sia in trasmissione sia in ricezione, anche a costo di diminuire la potenza erogata dal trasmettitore per rimanere comunque nei limiti di legge.
Formalmente il guadagno è definito come:

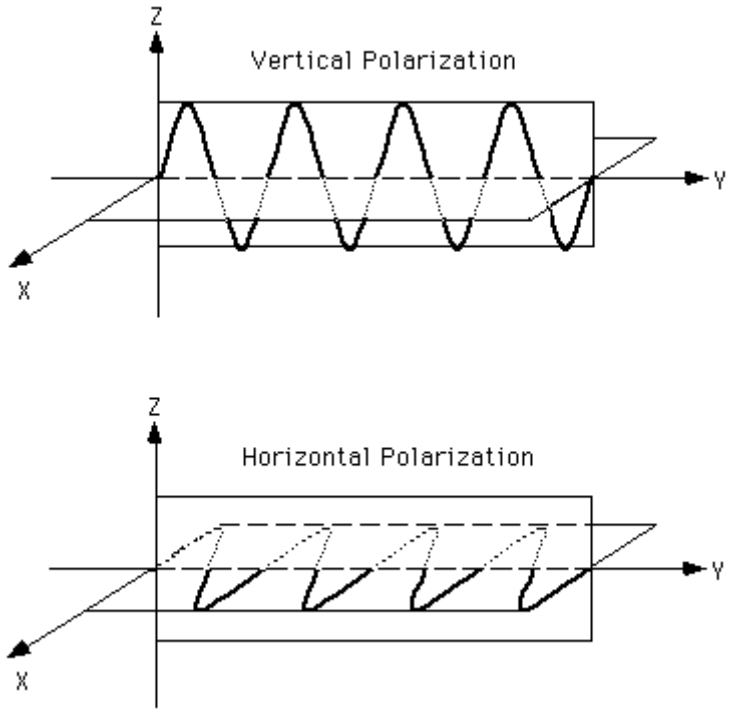
$$G = Ptx/Prif$$

Per comodità il guadagno di una antenna è tipicamente indicato in decibel rispetto all'antenna isotropa (dBi).

$$G\ [dB] = 10\log (Ptx/Prif)$$

4. Polarizzazione.

Un’altra caratteristica di una antenna è la polarizzazione del segnale, cioè come l’onda trasmessa oscilla nello spazio al variare del tempo.
Le polarizzazioni utilizzate nella pratica sono:
- Verticale: minore riflessione su superfici piane (terreno, acqua)
- Orizzontale: minore riflessione su superfici verticali (edifici, montagne, ecc…)
- Circolare: efficace in luoghi con molti corpi riflettenti



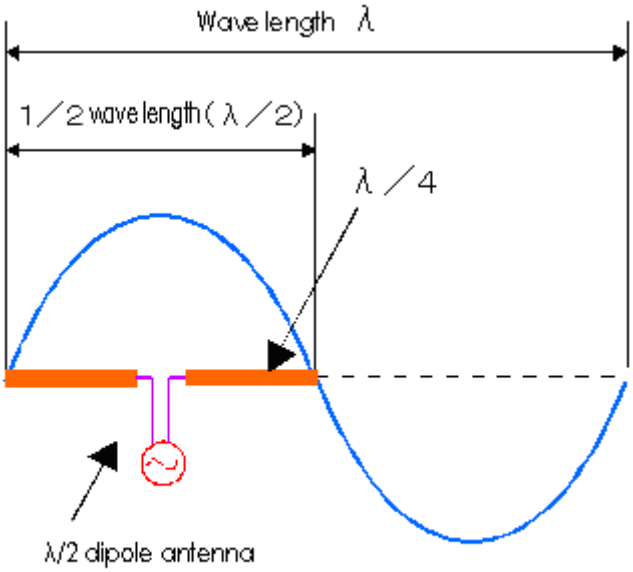
Un' antenna con polarizzazione verticale riceverà efficientemente solamente un' onda polarizzata verticalmente e viceversa.

5. Lunghezza antenna

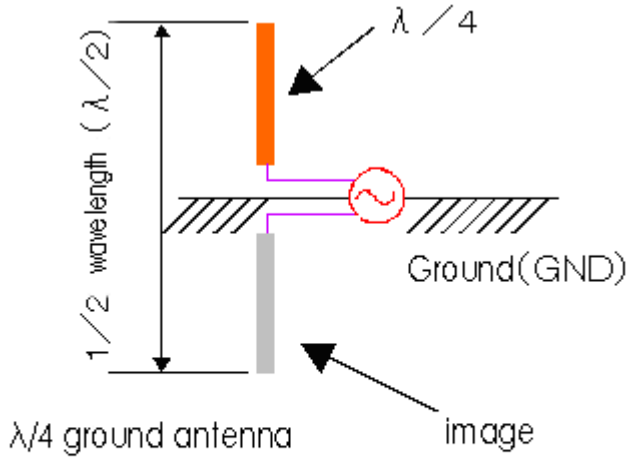
Poiché il trasmettitore deve emettere le onde radio usando una potenza limitata e la ricevente deve intercettare efficientemente le onde radio emesse, con la giusta lunghezza d'antenna le onde radio trasmesse realizzano una condizione di risonanza alla massima emissione di energia e questo vale anche per la parte ricevente. Inoltre per migliorare ulteriormente le prestazioni l'antenna dovrebbe essere mantenuta il più diritta possibile e non dovrebbe essere piegata a cerchio.
Oggi le apparecchiature tendono ad essere sempre più compatte e per questo si fa uso sempre più frequentemente di antenne con una ¼ lunghezza (λ/4). La funzione di una antenna a λ /4 è la stessa di una antenna a dipolo λ /2, ma siccome da un lato la funzione è quella di commutare a terra, la lunghezza dell'antenna è divisa in due facendo un ¼ λ. Le antenne a frusta dei moduli radio, quelle dei telefoni mobili etc. usano questo meccanismo quando serve la funzione di terra (ground), cioè la compensazione della parte mancante che in un'antenna λ/2 invece compare(infatti la λ/4 è lunga la metà della λ/2).
Per il calcolo della lunghezza d'onda λ basta:

λ [m] = velocità della luce/frequenza dell'onda

esempio: per una frequenza di 850MHz λ [m] = 3 x 10^8 [m] / 850 x 10^6 = 0.353m
quindi per un antenna λ/2 la lunghezza è 17.75cm, mentre per una λ/4 è 8.88cm.



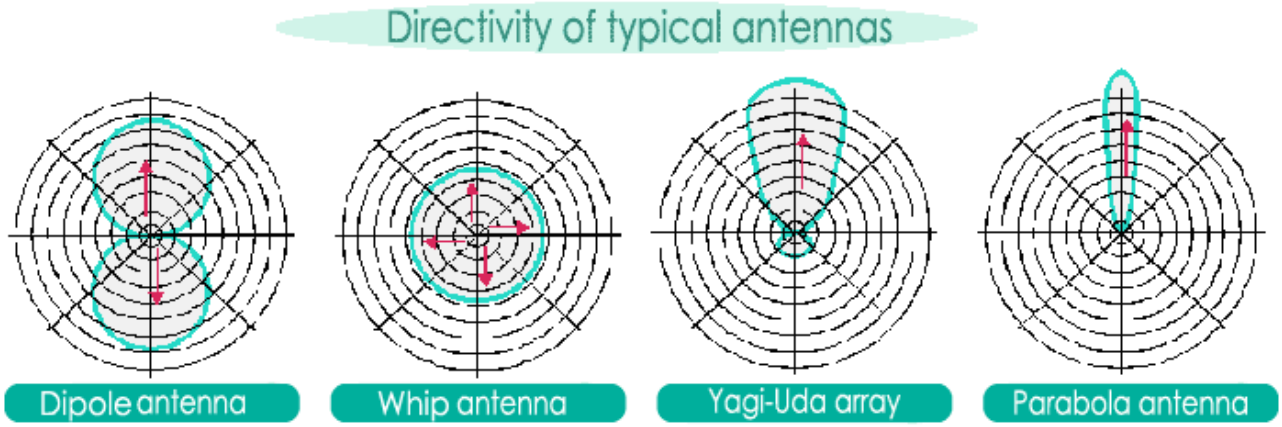
An antenna with a length half that of the wave length resonates most strongly.



A whip antenna forms λ/2 dipole antenna using an image made in the earth. The function of the earth can be changed to a vehicle body, case and so on.

6. Direzionabilità:

Ci sono antenne direzionali e on direzionali.
Le antenne direzionali possono essere utilizzate quando una delle parti in comunicazione ha postazione fissa. La direzionalità, oltre a consentire una trasmissione efficiente a bassa emissione, evita emissioni indesiderate nell'ambiente evitando la diffusione delle onde radio in altri sensi. Le onde radio che si irradiano in un senso specifico sono denominate beam (fascio). Le antenne non direzionali invece irradiano le onde radio indesiderabili nell'ambiente oltre a raccogliere il rumore da ogni senso. Tuttavia, con le antenne non direzionali la comunicazione è possibile in qualsiasi senso si trovino le parti in comunicazione e ciò le rende adatte nelle applicazioni mobili.



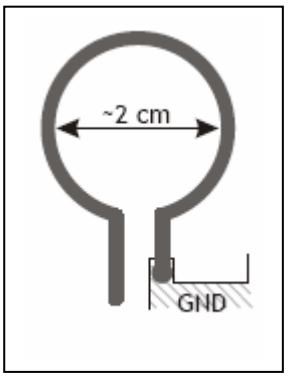
Se prendiamo come esempio l'antenna di allineamento Yagi-Uda , il fascio più grande di radiazione nel senso prospettato è il lobo principale e nel senso opposto la radiazione indesiderabile è denominata lobo laterale. Un lobo laterale nel senso frontale al lobo principale è denominato retro lobo.

Il rapporto fra il lobo principale ed il lobo posteriore denominato rapporto FB (Front/Back) esprime il livello di direttività dell'antenna, e questo livello è indicato in decibel (dB). Di conseguenza più è grande questo valore, migliori sono le prestazioni dell'antenna.

$$\text{FB ratio} = 20\log_{10}\left(\frac{\text{Main lobe (max.)}}{\text{Back lobe (max.)}}\right)$$

Dopo aver considerato i parametri sopraccitati, siamo passati alla scelta della antenne:

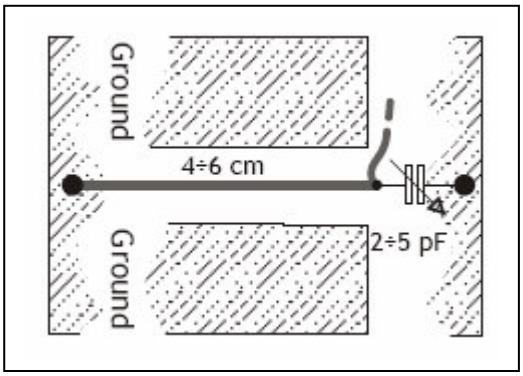
Antenne Loop



Tipo di antenna meno efficiente fra quelle disponibili. Ha il vantaggio di non richiedere punti di taratura e consiste in un loop di diametro approssimativo 2 cm, alimentato ad un'estremità e cortocircuitato a massa dall'altra.

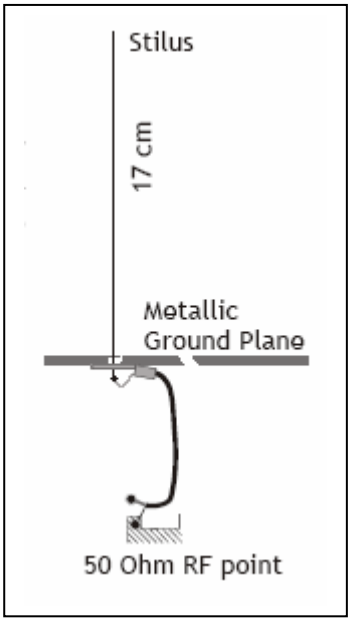
Si realizza direttamente sul circuito stampato, per cui è impiegata nei trasmettitori palmari ed è normalmente il carico del transistor oscillatore. Con riferimento all'antenna Stilo ha un'efficienza tipicamente inferiore di 20 dB il che comporta quasi un ordine di grandezza nella distanza teorica copribile.

Antenna Accordata (Su PCB)



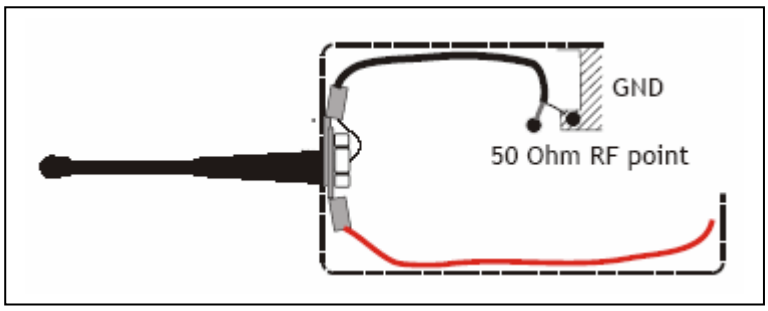
Altro tipo di antenna realizzabile direttamente su circuito stampato che richiede un punto di accordo ottenuto con capacità variabile.
In questo modo si ottiene l'impedenza vista dal dispositivo RF di 50 Ohm. La lunghezza fisica dell'elemento radiante è dai 4 ai 6 cm e l'accordo viene regolato per la massima efficienza di trasmissione o ricezione. Sempre con riferimento allo Stilo l'efficienza è tipicamente inferiore di 10 dB per cui la distanza teorica copribile cala di un fattore 3.

Antenna Stilo Verticale



È l'antenna più efficiente ottenibile in maniera semplice e ha come unico limite il dover utilizzare l'elemento radiante perpendicolare ad un piano di massa impegnando così un notevole spazio fisico.
Essendo facilmente ottenibile con le caratteristiche ottimali si consiglia comunque di impiegare questo tipo di antenna anche solo come verifica sperimentale delle caratteristiche elettriche dei ricevitori e trasmettitori collegati.
Sostanzialmente possiamo definire in 0 dB il guadagno di questa antenna per cui la potenza in ingresso viene riportata in energia radiante praticamente senza variazioni.
È possibile pertanto verificare le reali distanze che i vari moduli impiegati possono coprire con sicurezza decidendo di volta in volta quale tipo di antenna anche meno efficiente si può utilizzare.

Antenna Stilo Caricata



Per impieghi in cui il supporto non sia una superficie metallica adeguata come ad esempio il coperchio plastico di una scatola, è possibile utilizzare lo stilo caricato che riporta all'esterno l'elemento di antenna e costruendo una massa artificiale detta contrappeso.
Essendo di solito le dimensioni esterne più contenute per motivi funzionali dei 17 cm teorici l'efficienza è leggermente inferiore **[**max 2 dB**]** dello stilo ideale.

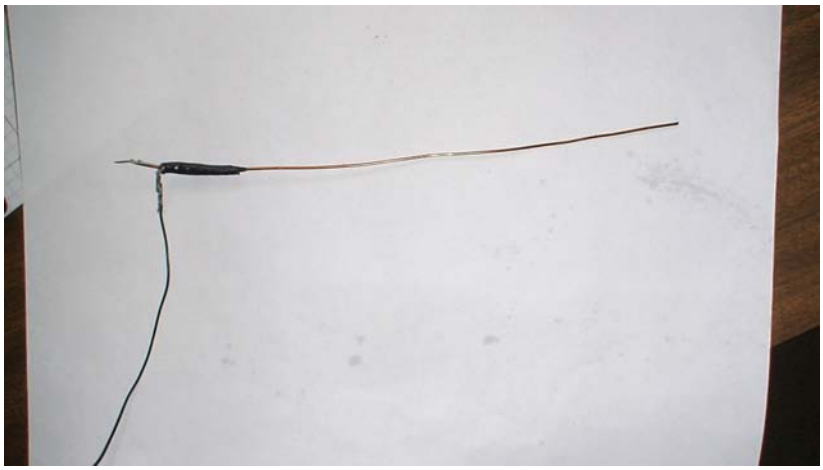
Antenne a frusta o asta
Sono le antenne utilizzate per i telefoni mobili. Antenne non direzionali con sensibilità uguale in qualsiasi senso.

Antenne di Allineamento o Yugi-Uda
Usate comunemente come antenne della televisione. Hanno forte direttività e devono essere allineate nel senso della stazione trasmittente. È un'antenna a bipolo che integra gli elementi di riflessione e direzione per dirigere e riflettere l'onda radio.

Antenne Paraboliche
Usate per la ricezione delle radiodiffusioni satellitari. Queste antenne hanno forte direttività, usano la potenza delle onde radio efficientemente e richiedono una registrazione molto accurata.

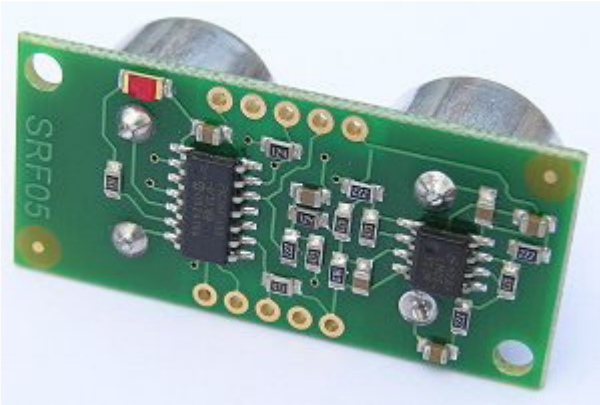
Antenna Dielettrica
Antenne che usano la ceramica dielettrica ad alta frequenza. Possono essere compatte e realizzare un elevato guadagno.

Dopo accurate ricerche abbiamo deciso di adottare un'antenna stilo non direzionabile, con una lunghezza totale di 16.5cm, ovvero λ/4, e un diametro di circa 1mm, realizzata in filo metallico di rame. Per realizzarla abbiamo utilizzato del conduttore in Cu usato anche per la costruzione degli avvolgimenti dei trasformatori. Essendo un'antenna λ/4, necessita della funzione di terra, quindi abbiamo schermato l'estremità con la calza del cavo coiassale. Le antenne sono posizionate in verticale per una trasmissione orizzontale e sono mantenute il più diritte possibile. Inoltre tutti i dispositivi sono lontani circa 5cm dal corpo dell'antenna. È molto importate che i circuiti vicino all'antenna siano circondati da un buon piano di massa per garantire l'immunità da possibili interferenze.

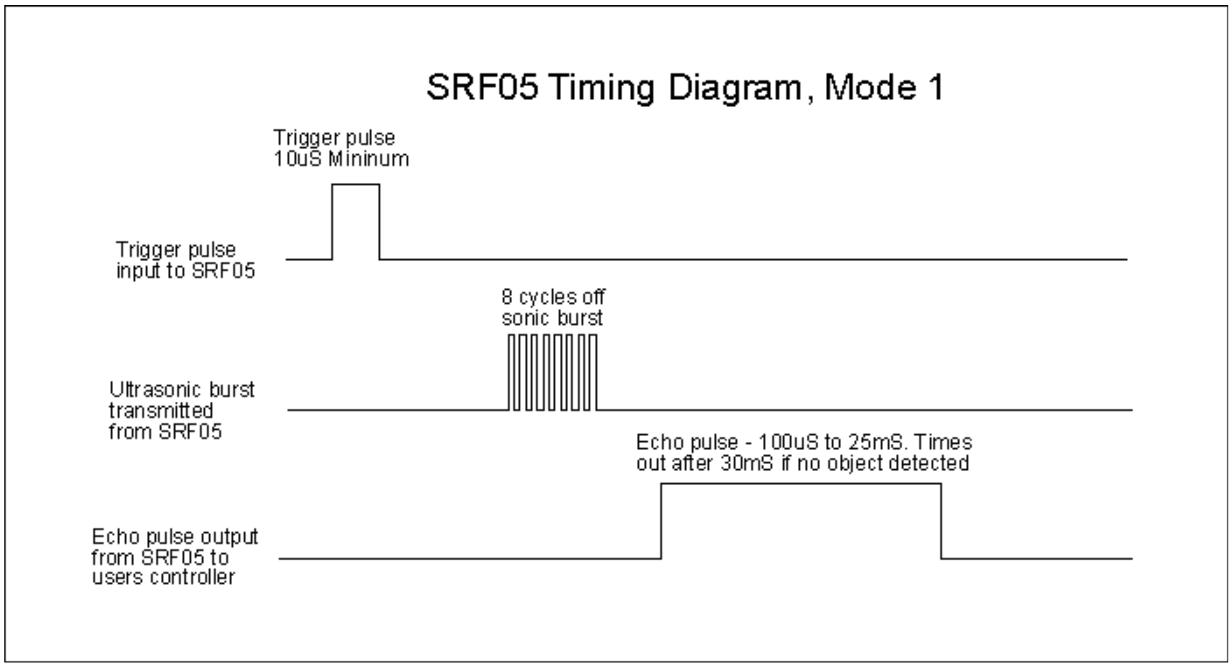


SENSORI

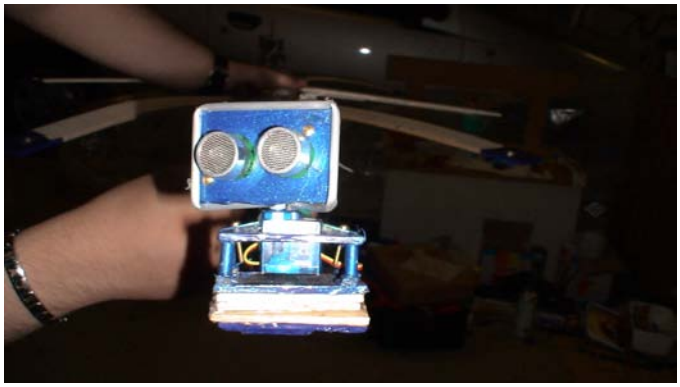
I sensori sono una delle caratteristiche che distinguono un robot da un semplice mezzo radiocomandato.
Il nostro dirigibile è dotato di parecchi sensori e dispositivi di misura, attraverso i quali rende l'utente consapevole della situazione ambientale circostante.
I sensori vengono gestiti da entrambi i PIC presenti nella mainboard della gondola.
Il PIC16F88 si occupa dei sensori più importanti, quelli per il rilevamento degli ostacoli.
Il dirigibile riesce a misurare la distanza a cui si trova l'ostacolo per mezzo di sensori ad ultrasuoni.



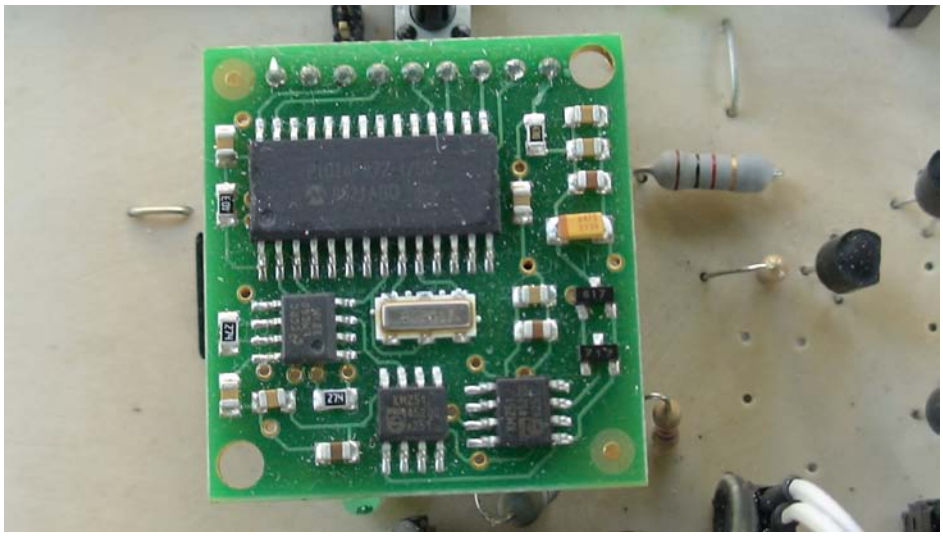
Due di tali sensori sono montati su un servomotore e fissati alla struttura, mentre il terzo è fermo e montato sotto la gondola per misurare l'altezza da terra. Questi sensori hanno una "vista" massima di 4m e comunicano con il pic per mezzo di un impulso TTL di durata variabile. Oltre all'alimentazione sono presenti due segnali: TRIGGER è un input del modulo: va inviato un impulso di 10mS su questo canale per iniziare una misura.
ECHO è la risposta del modulo (output), che consiste in un impulso alto con durata variabile e proporzionale alla distanza misurata dal dispositivo.
Per leggere il sensore si utilizzano i comandi PicBasic "Pulsin" e "Pulsout".



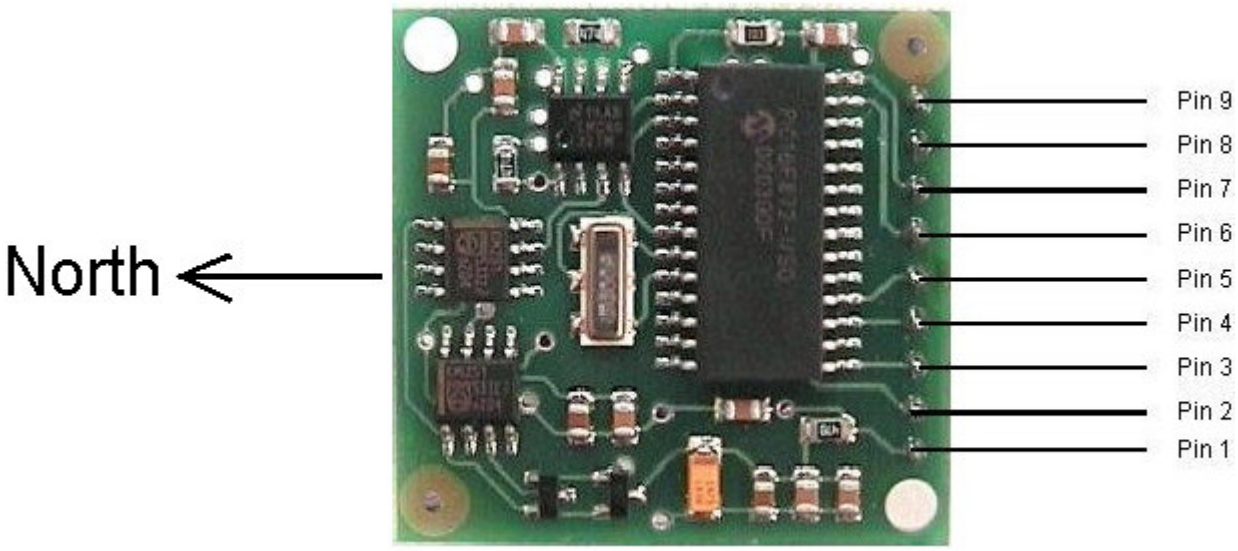
Come detto in precedenza, due di questi sensori sono montati su servomotori. Il motivo è semplice: con i servomotori possiamo effettuare misure su più direzioni (Sonar) semplicemente posizionando il servo prima di effettuare la misura. I sensori Sonar, battezzati con il nome di “Fuffy”, effettuano misurazioni nelle posizioni Centrale, Destra e Sinistra sia Davanti che Dietro il pallone.



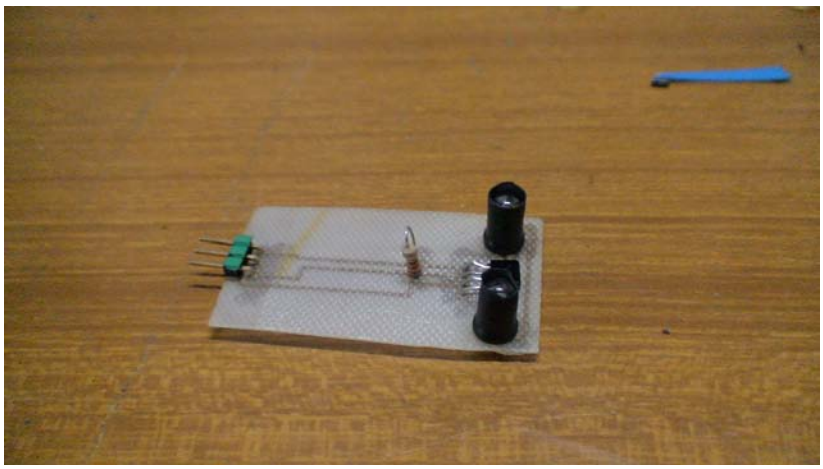
La misura dei sensori ed il posizionamento richiede tempo. Questo è uno dei motivi principali per cui è stato scelto un pic secondario (16F88) per la gestione di tutti i sensori Sonar-Ultrasuoni. L’esigenza di una comunicazione rapida e senza gestione delle interruzioni tra questo microcontrollore e quello principale è la principale motivazione dell’uso obbligato dell’I2C Ibrido e della SWAP-RAM.
Il velivolo è dotato di un altro importantissimo sensore: la Bussola Elettronica.

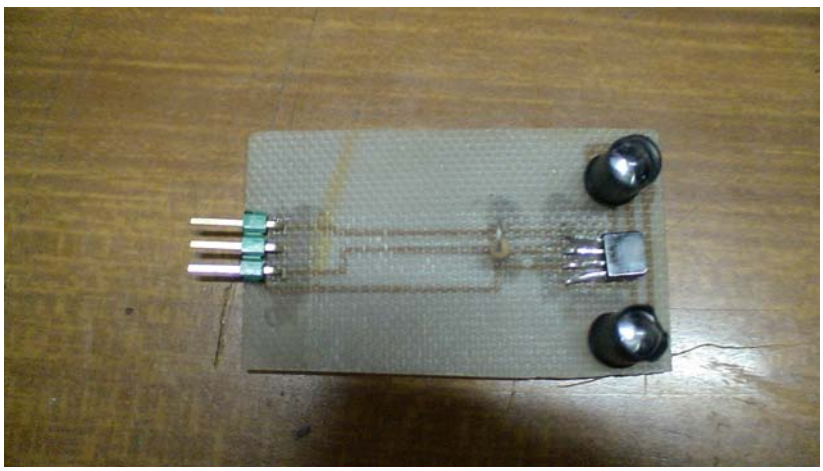


È per mezzo di tale componente che l'utente è reso consapevole dei cambi di rotta del dirigibile.
La bussola comunica attraverso lo standard I2C, quindi è stata collegata al bus tramite un connettore strip per renderla rimovibile. La bussola viene letta con la stessa procedura con cui si legge una memoria (RAM, EEPROM…), ma cambia il codice del dispositivo.
Per cui è impossibile sbagliarsi e leggere una delle memorie anziché la bussola. Attraverso l'I2C possiamo accedere alle locazioni contenenti i dati misurati, già pronti all'uso. La bussola fornisce il valore in formato Byte oppure Word (16 Bit). Basta leggere le locazioni corrispondenti (Formato Byte, Formato Word – Byte Basso, Formato Word – Byte Alto) per avere la posizione in gradi, con 0 riferito a Nord, del velivolo.

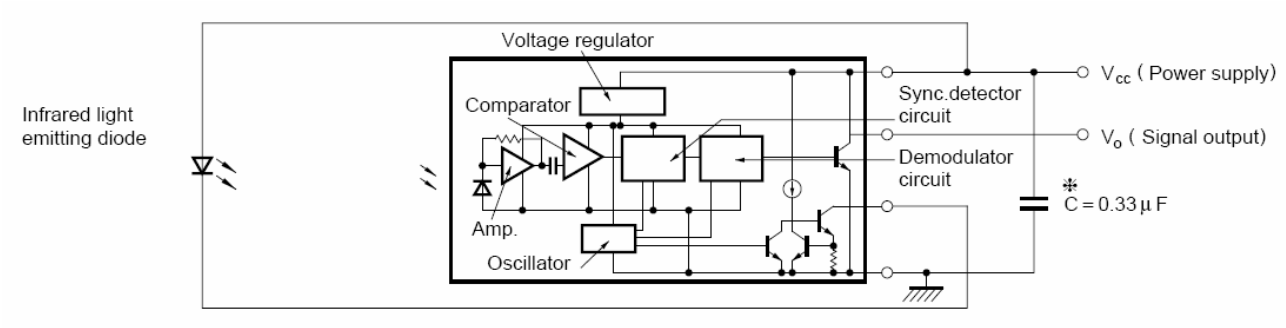


Il pic Centrale si occupa della lettura della bussola: viene letta la misura a 16 Bit per comodità, in modo tale che nel programma sia disponibile un valore preciso, ma convertibile facilmente in un Byte.
Il dato letto, nel formato word, va da 0, corrispondente ai 0° Nord, a 3599 corrispondenti ai 359,9° geometrici.
Il formato byte va invece da 0 a 255 ed ogni unità corrisponde a circa 1,4° geometrici, con lo zero corrispondente ai 0° Nord.
Attraverso la Bussola quindi il dirigibile prende consapevolezza di dove si sta dirigendo (con approssimazione media di 1°).
Il pic principale gestisce anche i sensori ad infrarossi: Il velivolo è dotato di due sensori, fissati sopra il pallone, uno davanti ed uno dietro, e connessi alle derivazioni della struttura per mezzo di un cavo flat, per la rilevazione dei soffitti.
È stato realizzato un modulo contenente il sensore, due led emettitori a luce infrarossa ed i componenti passivi (in sostanza una resistenza) necessari al funzionamento del dispositivo.



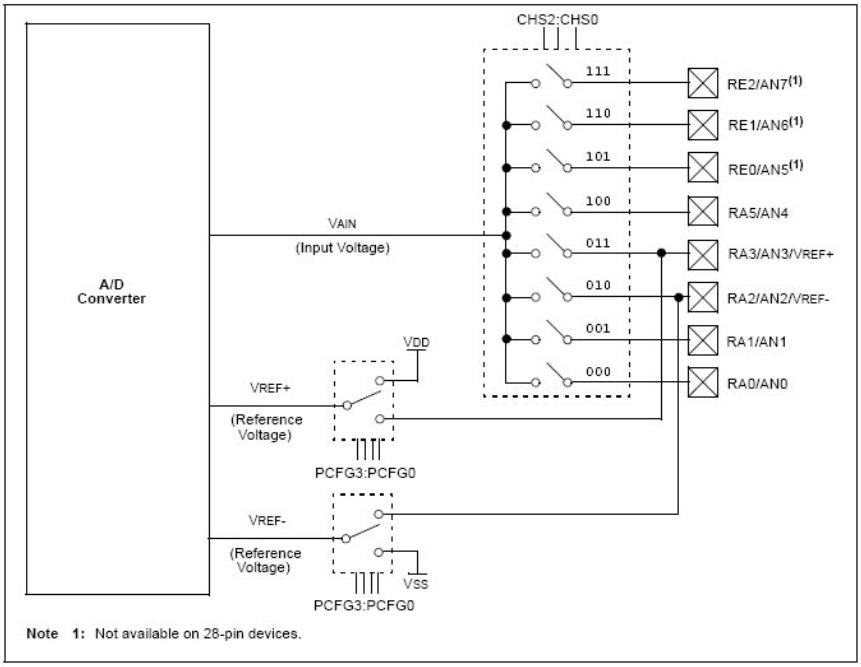


Il sensore ad infrarosso è dotato di un circuito interno per il pilotaggio dei led. I led vengono accesi e spenti con una certa frequenza, generata dal sensore. Il dispositivo "presta attenzione" solo a tale frequenza, per cui è molto basso il rischio di interferenze esterne.
Se il dirigibile incontra un soffitto la luce infrarossa emessa dai diodi viene riflessa ed il sensore la rileva. La portata del sensore, utilizzando la luce riflessa, varia dai 10cm ai 20cm a seconda del coefficiente di riflessione dell'ostacolo. Il sensore presenta in uscita un livello alto (TTL) che viene abbassato quando viene rilevato l'ostacolo.
Questo tipo di sensore è studiato apposta per applicazioni simili a questa e noi lo abbiamo scelto per un motivo ben preciso: è dotato di circuito di pilotaggio per i led.

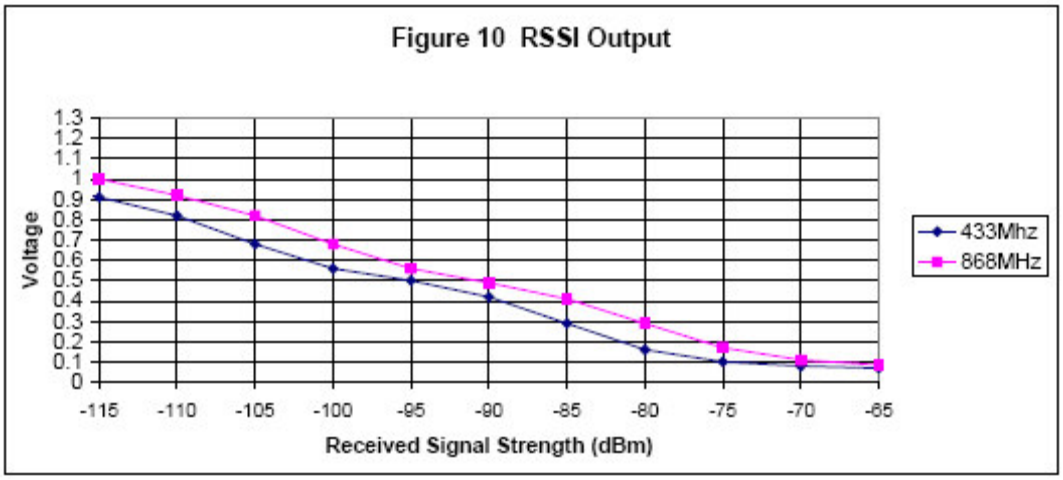


Infatti si poteva acquistare un ricevitore infrarosso comune, tarato per esempio sulla frequenza di 38Khz, ma poi sarebbe stato difficile generare con il pic tale frequenza in modo rapido per pilotare i led emettitori. PicBasic arriva solo a 32Khz, inoltre l'uso dei sensori descritti semplifica notevolmente anche l'interfaccia di collegamento ed aumenta la sicurezza sulla validità del risultato dell'operazione.

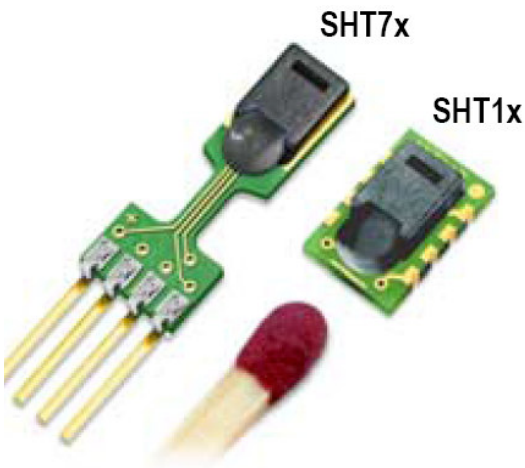
Anche se non sono dei sensori, gli ADC del pic possono essere considerati come tali, dato che vengono utilizzati per rilevare lo stato delle Batterie e la quantità di segnale radio.



Il dirigibile è alimentato da due celle con tensione 4,2V in serie. La tensione di riferimento superiore per la misura è di 5V, quindi la tensione della singola cella può essere misurata. La tensione della seconda cella non può essere misurata direttamente perché manca il riferimento a Gnd (inferiore). Per risalire alla tensione di quest'ultima basta misurare la tensione totale e sottrarre matematicamente (all'interno del software) la tensione della prima cella.
Le due celle sono in serie e la tensione totale è di 8,4V, superiore al riferimento superiore dell'ADC. Quindi è stato realizzato un partitore di tensione con resistenze di elevato valore (2,2MOhm) in modo tale da dividere il totale in due. Il valore misurato è quindi la tensione totale dimezzata, che verrà poi moltiplicata per due nel software prima di eseguire le altre operazioni.
Abbiamo così lo stato delle batterie: cella 1, cella 2 e totale.
L'ADC viene usato anche per misurare la tensione del segnale RSSI del modulo radio. Tale segnale indica la quantità di segnale portante ricevuto dal modulo. Con tale valore è possibile stimare la distanza a cui si trova il dirigibile rispetto al dispositivo trasmittente (la base).

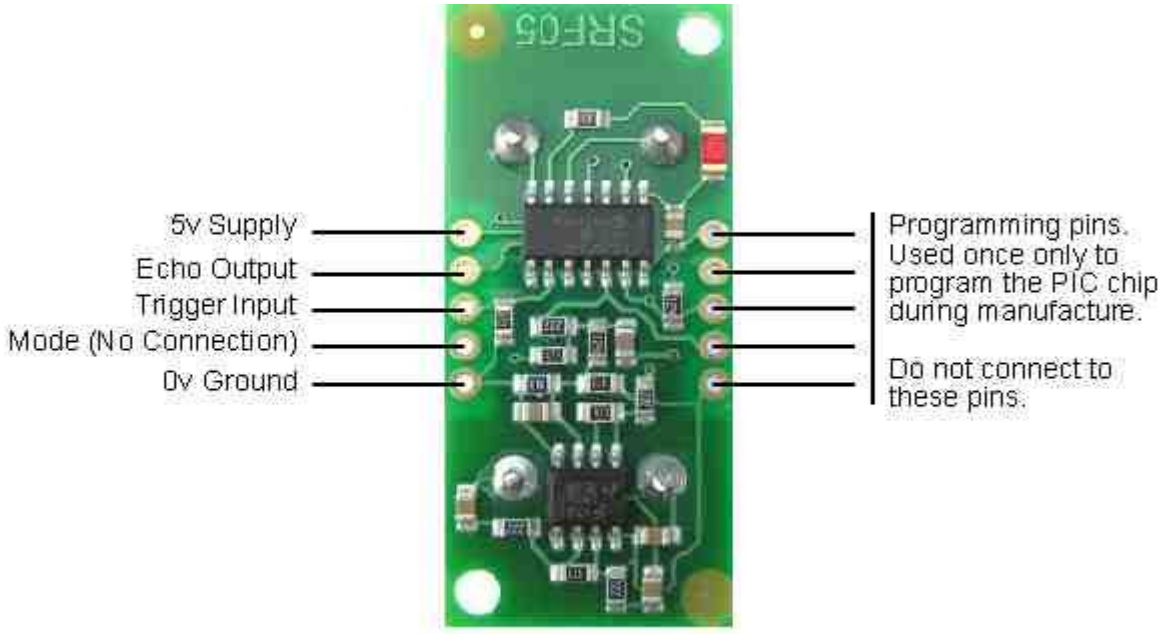


Il segnale RSSI è nettamente inferiore al riferimento dei 5V, quindi basta entrare con la rispettiva linea direttamente in un canal ADC del pic.
Il pic dispone di 8 canali ADC di cui ne sono utilizzati solamente tre. Gli altri sono stati portati agli slot di espansione per sviluppi futuri.
La gondola è dotata di 4 slot di espansione, di cui uno interno.
Alle quattro espansioni sono state portate tutte le tensioni di alimentazione, il bus I2C, un canale digitale ed uno analogico (ADC).
L'espansione occupata dal modulo della telecamera presenta un canale digitale in più.
Alle espansioni rimanenti è possibile collegare altre schede contenenti sensori di vario tipo. Sono messi a disposizione i canali analogico, il canale digitale ed il bus per la comunicazione con il pic centrale.
Attraverso le espansioni è possibile far diventare il nostro dirigibile una completa stazione meteo volante, per esempio introducendo un sensore di umidità e temperatura come l'**SHT11** della Sensirion (comunicazione seriale sincrona).



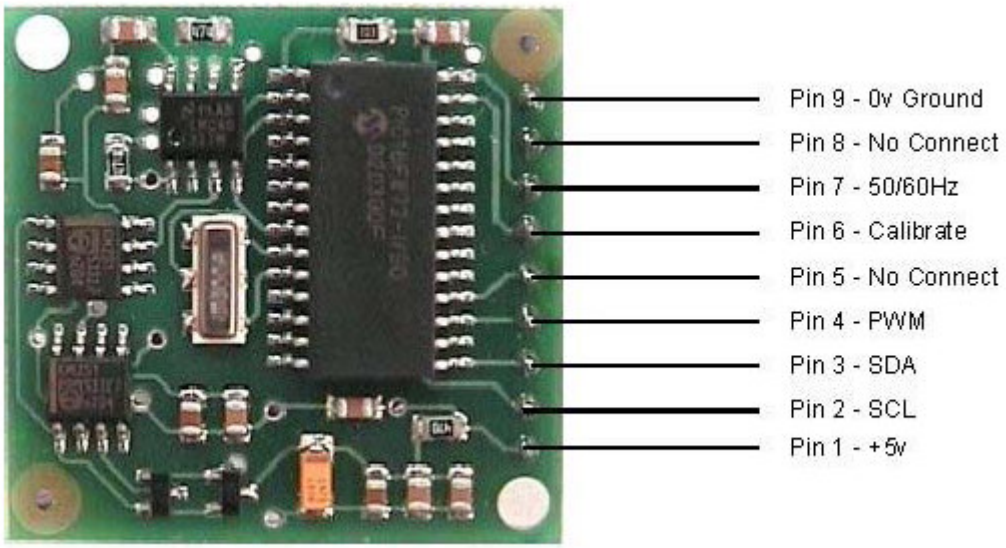
I canali ADC possono essere configurati anche come normali I/O digitali. Se il convertitore Analogico-Digitale non serve e fa comodo un canale digitale in più, questa operazione si rileva molto utile.
Ecco le caratteristiche dei sensori da noi utilizzati:

- **SRF05** sensore ad ultrasuoni prodotto dalla Devantech, dotato di microcontrollore che si occupa delle misure, massima distanza misurabile 4m con approssimazione 1cm, frequenza onda sonora 40Khz, comunica con un segnale TTL di durata proporzionale alla distanza misurata.

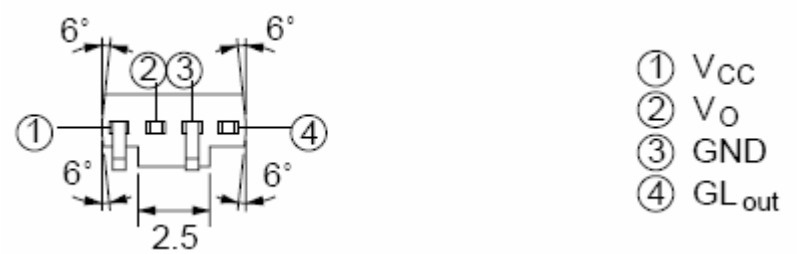


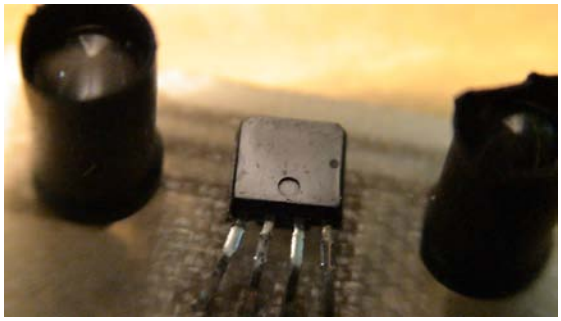
Connections for 2-pin Trigger/Echo Mode (SRF04 compatible)

- **CMPS03** bussola elettronica prodotta dalla Devantech, molto sensibile, riesce a misurare correttamente cambi di rotta di 1°, può comunicare via I2C oppure attraverso un segnale PWM con duty-cycle variabile in base al valore misurato, può essere calibrata in modo semplice grazie ad un pulsantino, dimensioni e consumi contenuti.



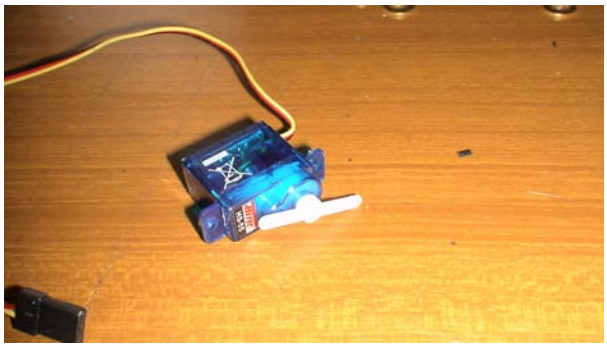
- **IS471F** sensore infrarosso prodotto dalla Sharp, circuito oscillante per il pilotaggio del led emettitore integrato, alimentabile da 4,5V fino a 16V, uscita TTL.

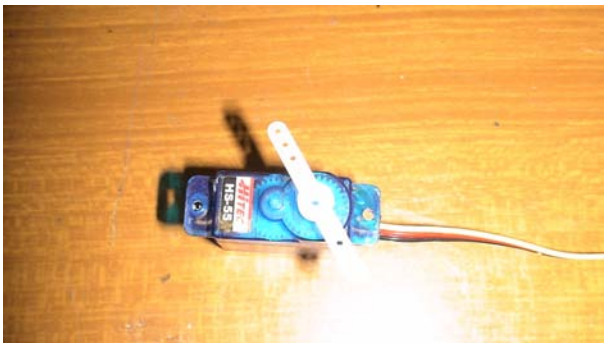


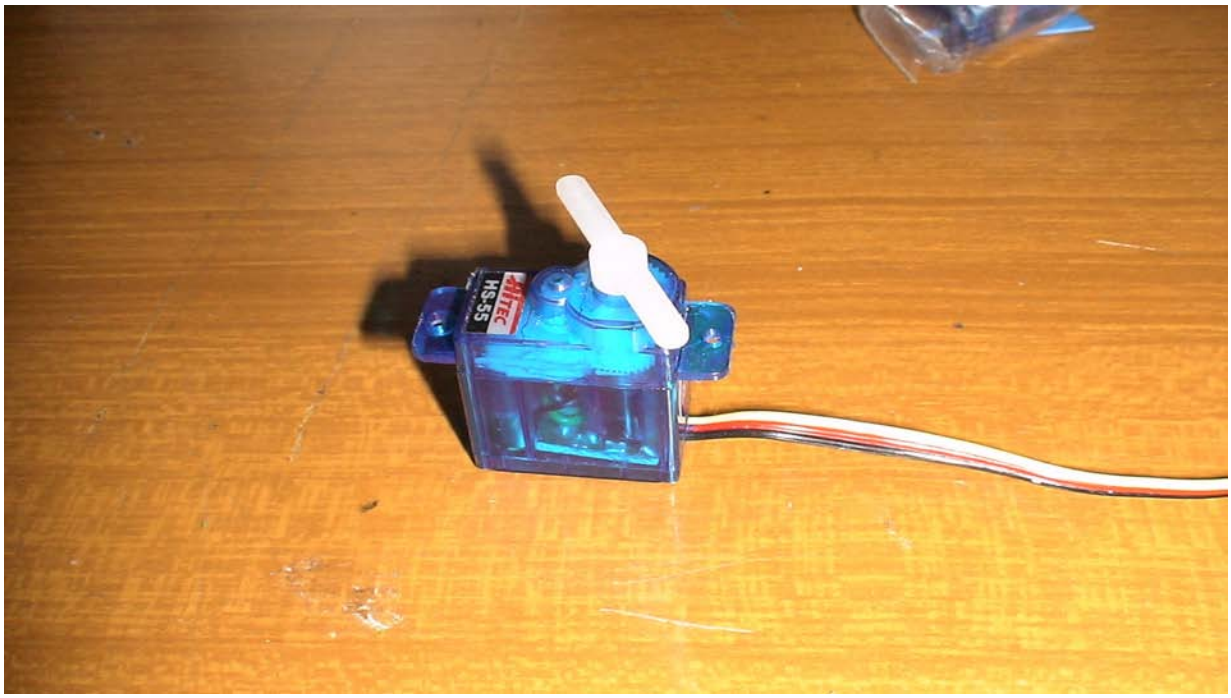


SERVOMOTORI

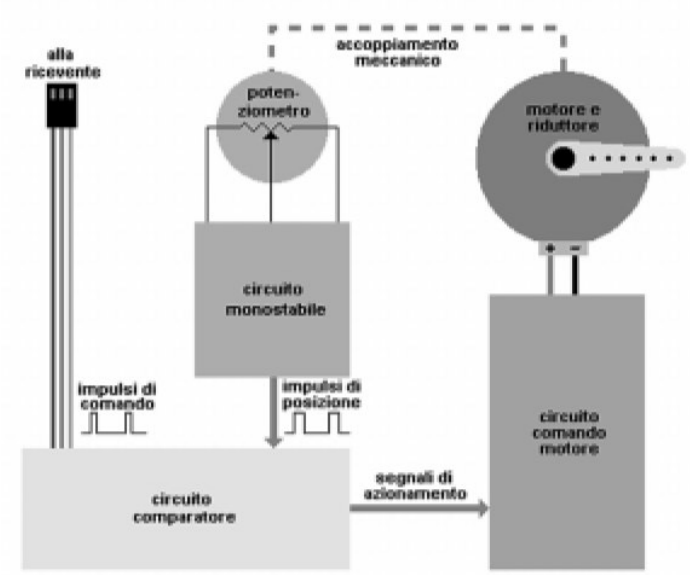
Nel nostro progetto facciamo uso di servomotori per il posizionamento dei sensori ad ultrasuoni. Gli altri “servomotori” presenti nel dirigibile in realtà erano servomotori, ma è stata tolta l’elettronica di comando per renderli dei semplici motoriduttori in DC (piccoli, leggeri ed economici). I veri due servomotori sono quelli dei due Fuffy.

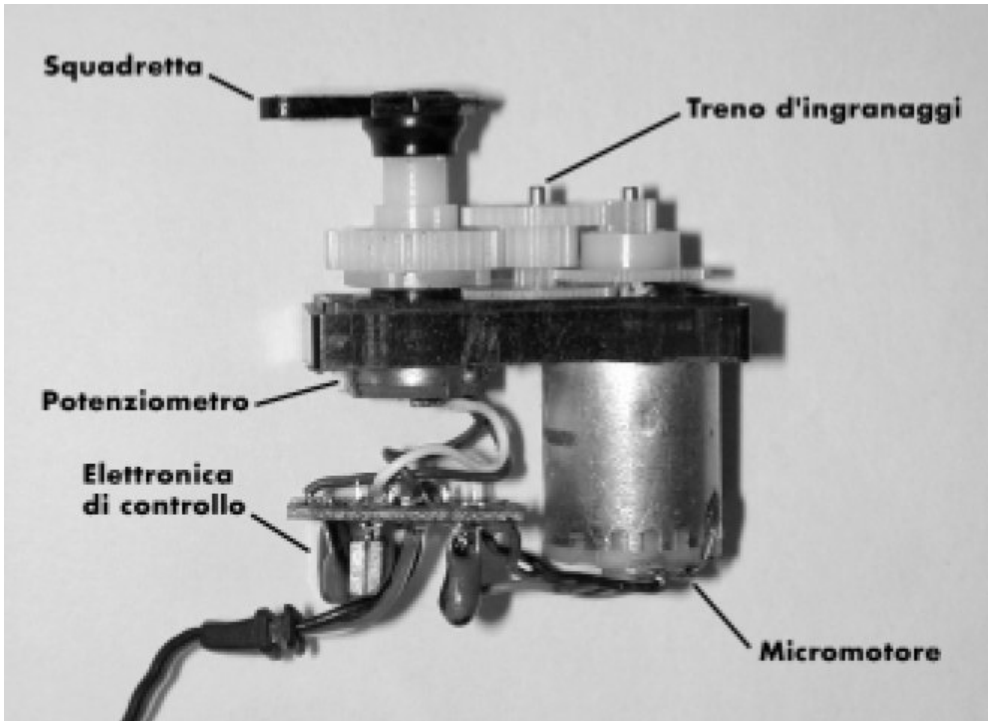




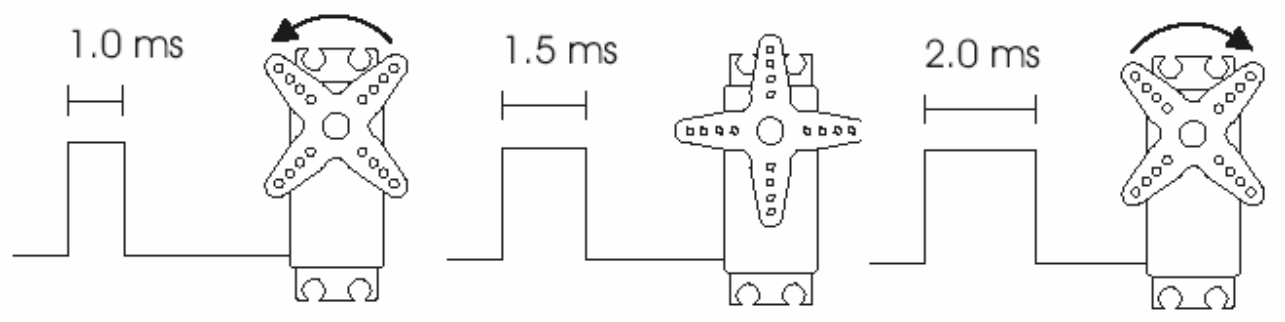


Il servomotore è composto da un motore in DC, da una riduzione meccanica del numero di giri e da un potenziometro usato come trasduttore di posizione. All’interno del servo vi è un circuito elettronico che accetta in ingresso un segnale pwm, usato per la regolazione della posizione. Il motore si ferma quando è stata raggiunta la posizione scelta: il potenziometro indica all’elettronica la posizione corrente, che viene confrontata con la posizione inviata al servo tramite il segnale di comando.

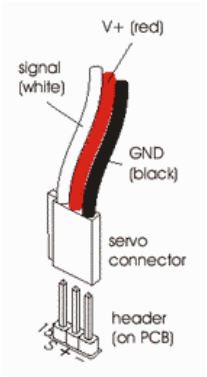




Il potenziometro agisce anche come blocco meccanico, assieme ai blocchi plastici contenuti nel servo, impedendo rotazioni maggiori di 180°. In poche parole il servomotore può effettuare solo mezzo giro. Vi sono in commercio anche dispositivi con angolo permesso maggiore, ma sono prodotti di non comune utilizzo.
Assumendo la posizione centrale del servo a 0°, il servo potrà spostarsi da 0° fino a -90° e da 0° fino a +90°.

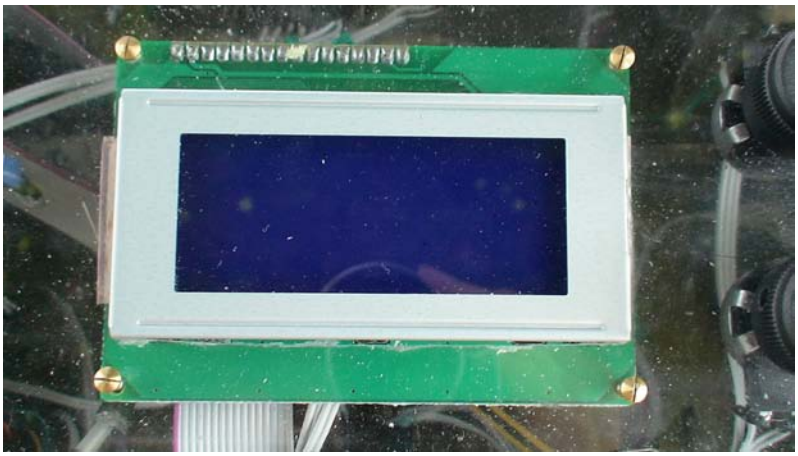


La posizione del servomotore viene impostata come detto sopra, attraverso un segnale pwm.
La parte alta dell'onda quadra indica la posizione: La durata dell'impulso varia da 1ms a 2ms corrispondente alla posizione da -90° a +90°. Per mettere il servomotore nella posizione centrale occorre inviare quindi impulsi dalla durata di 1,5ms.
La parte bassa dell'onda invece serve per variare la velocità di rotazione del servo: la durata minima è di 10ms perché l'elettronica del dispositivo "capisca" il comando inviato, il massimo consigliato non dovrebbe essere superiore ai 40ms-50ms, per lo stesso motivo indicato in precedenza. Normalmente si usa una durata di 20ms. È da notare che più lunga è la parte bassa dell'onda, più il motore diventa impreciso, poco potente e "cedevole" oltre che lento.
Se si intende generare il segnale pwm attraverso routines software anziché hardware, è necessario inviare almeno 50 periodi di comando, ad esempio utilizzando i comandi PicBasic "Pulsout" per l'impulso alto e "Pause" per la parte bassa dell'onda quadra.
Il servomotore normalmente è alimentato dai 4V ai 7V, l'interfaccia è a 3 cavi: +Vcc, Gnd, Segnale di comando (bianco o giallo).

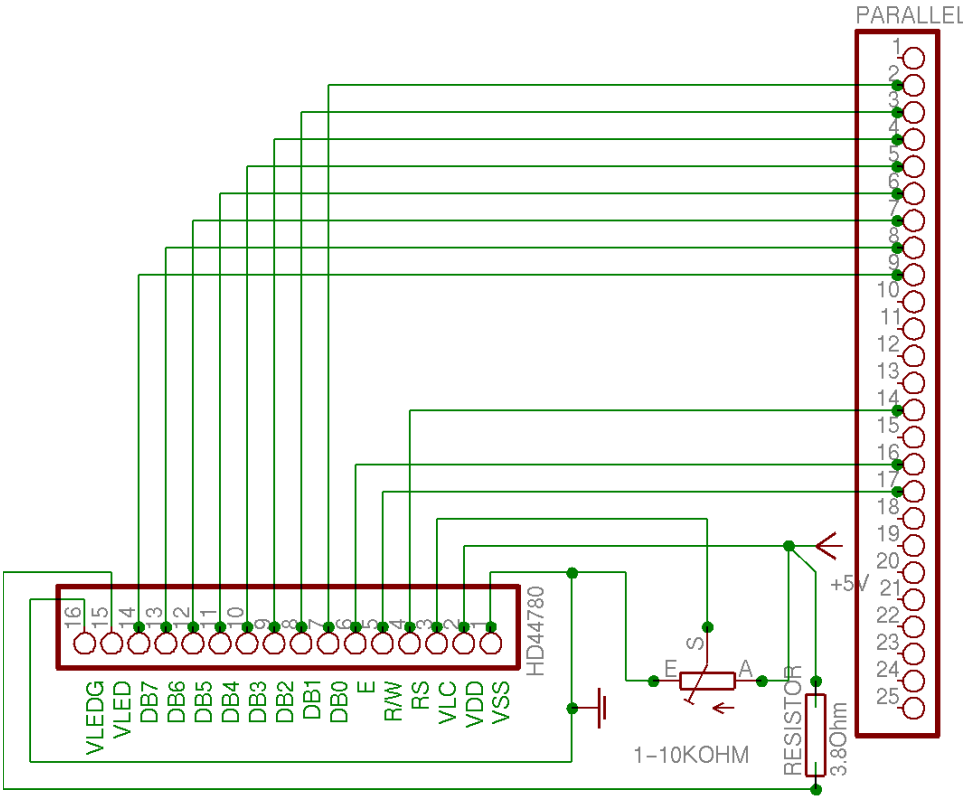


DISPLAY LCD PARALLELO
E CODIFICA HITACHI

Sulla base terrestre, per le segnalazioni di emergenza e di servizio, è stato installato un display lcd a 4 righe da 16 caratteri, retroilluminato a led blu, alimentato a 5V, con la possibilità di regolare il contrasto e l'intensità della retroilluminazione.



Il display presenta una piedinatura molto semplice:



1. VSS ovvero GND;
2. VDD, l'alimentazione del display che corrisponde a 5V;
3. VLC, collegato ad un potenziometro definisce il contrasto dello schermo;
4. RS, da cui il display dipende per avere istruzioni o dati;
5. R/W, imposta il display per la lettura/scrittura;
6. E, è il pin di abilitazione;

7-14 rappresentano il bus dati.

15-16 VLED e VLEDG non sono altro che i pin per la regolazione della retroilluminazione attraverso un comune potenziometro.

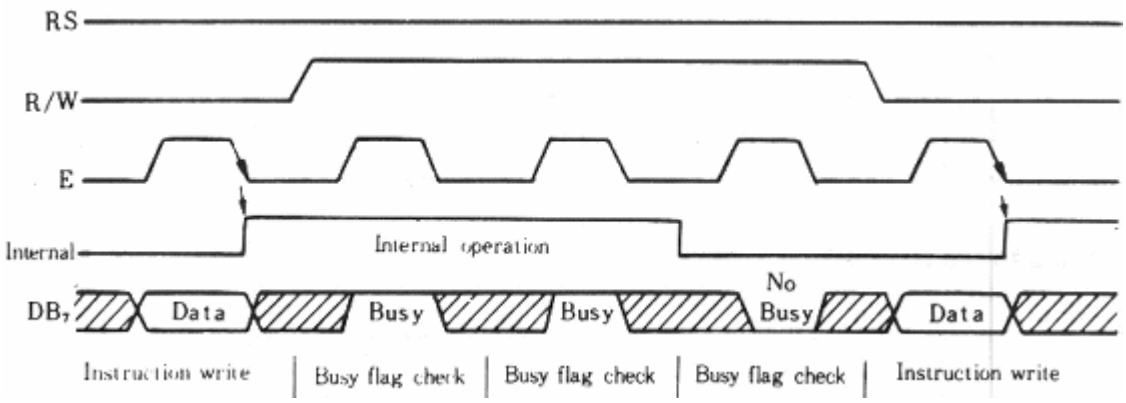
Tale display di basa sullo standard Hitachi HD44780 e non abbiamo incontrato particolari difficoltà nel comandarlo in quanto PicBasic gestisce in modo semplice l'invio dei dati al display.
PicBasic prevede due modalità di comunicazione con il display:

1. comunicazione con bus dati da otto linee (un bit per linea)
2. comunicazione con bus dati da quattro linee (due bit per linea)

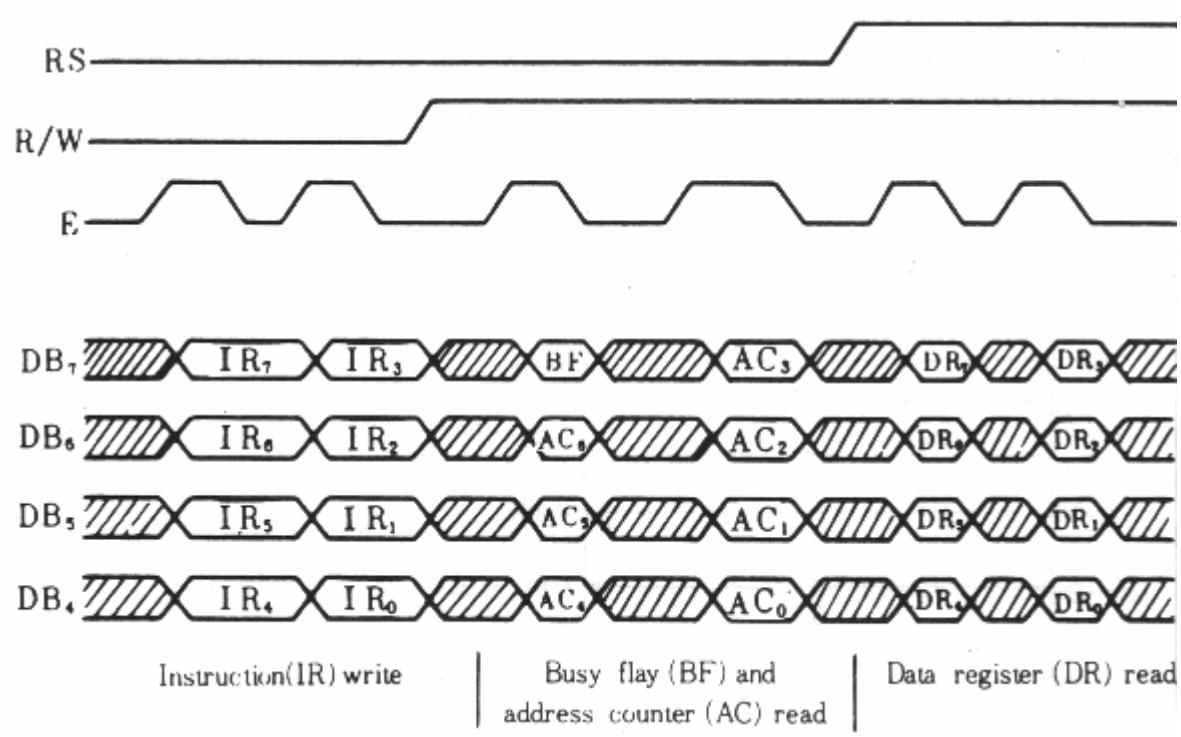
Basta fornire al compilatore i nomi dei pin a cui sono collegate le linee del display ed indicare la modalità di comunicazione. In caso venga usato un bus a quattro linee, i canali dati da collegare sono gli ultimi quattro (DB7, DB6, DB5, DB4).

8-bit interface

.



4-bit interface



Per maggiori dettagli sullo standard Hitachi HD44780 consultare il relativo datasheet.

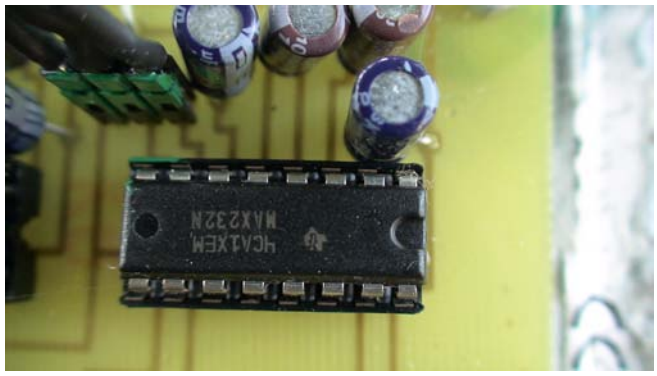
MAX232 E COLLEGAMENTO AL PC

Io nostro progetto prevede la possibilità di collegare la circuiteria del dirigibile con un computer per mezzo di un'interfaccia seriale.
Il protocollo rs232 prevede particolari parametri elettrici, infatti la tensione di uscita ad un trasmettitore RS232 deve essere compresa in valore assoluto tra 5V e 25V (quest'ultimo valore ridotto a 13V in alcune revisioni dello standard). A volte le tensioni in uscita sono intenzionalmente diminuite a +/- 6V anziché 12V per permettere minori emissioni EMC, peraltro sempre critiche, e favorire maggiori velocità di trasmissione.
Il ricevitore deve funzionare correttamente con tensioni di ingresso comprese, sempre in modulo, tra i 3V ed i 25V. Molti ricevitori commerciali considerano semplicemente una tensione di soglia al valore di +2V (sopra viene riconosciuto un segnale alto, sotto uno basso) anche se ciò non è pienamente aderente alla norme. E' però utile per effettuare una trasmissione "rs232" con livelli TTL...
L'impedenza di uscita del trasmettitore deve in ogni situazione essere maggiore di 300 ohm; l'impedenza di ingresso deve essere compresa tra i 3 ed i 7 kohm, anche a dispositivo spento. La corrente prelevabile in uscita mantenendo i corretti valori logici deve essere di almeno di 1.6 mA (potrebbe però essere maggiore, anche di un ordine di grandezza) e nel caso di corto circuito deve essere minore di 100mA. Infine lo slew-rate (cioè la pendenza del grafico del segnale nel passare da 1 a 0 o viceversa) deve essere minore di 30V/us per evitare eccessive emissioni elettromagnetiche.

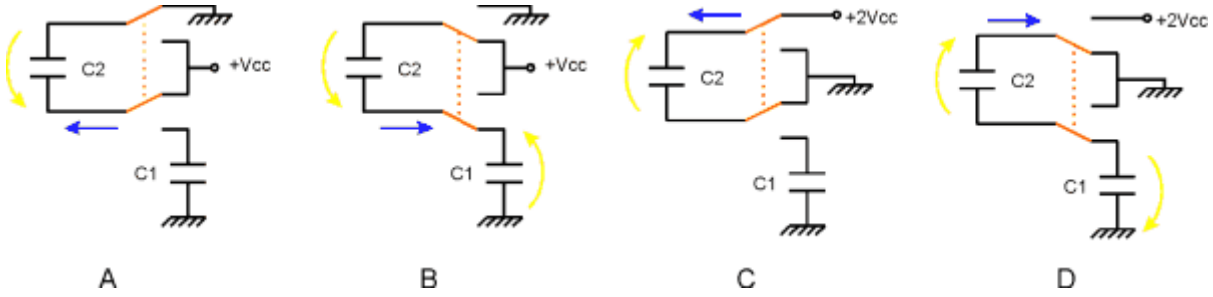
In genere i segnali utilizzati dai sistemi digitali variano tra 0 e 5V e non sono quindi direttamente compatibili con la standard RS232. In commercio esistono appositi *traslatori di livello* che hanno il compito di fornire sia in trasmissione che in ricezione gli opportuni livelli pur non modificando la forma del segnale trasmesso. Alcuni integrati (per esempio i classici 1488 e 1489, rispettivamente un trasmettitore ed un ricevitore, ambedue a quattro canali) sono molto usati in sistemi in cui è presente (oltre all'alimentazione logica di 5V o 3.3V) un'alimentazione duale a +/-12V. Questo integrato, come praticamente tutti i circuiti di questo tipo, contiene un inverter per ciascun canale e quindi nel segnale in uscita o in ingresso uno zero logico appare come 0 volt, cioè in quella che a molti sembra

essere la rappresentazione ovvia dei segnali digitali. L'uso di questi integrati è semplice ma non è sempre attuabile a causa della necessità di disporre di tre alimentazioni: si pensi per esempio alle apparecchiature alimentate a batteria.

Il MAX232 è un circuito integrato che permette il collegamento tra logica TTL o CMOS a 5V e le tensioni RS-232, partendo solo da un'alimentazione a 5V. Per ottenere la tensione positiva e negative necessarie per il funzionamento dell'integrato è usata una configurazione a pompa di carica, costituito da circuiti interni all'integrato e quattro condensatori esterni di circa 1 µF. La capacità effettiva dipende dal tipo di integrato e dalla relativa frequenza di commutazione; a volte i condensatori sono presenti all'interno dell'integrato stesso. Sono disponibili anche integrati che richiedono un'alimentazione di solo 3.3V (p.e. il MAX3232). La sezione ricevente del MAX232 è costituita da due porte invertenti che accettano in ingresso una tensione di +/- 12V (o altra tensione compatibile allo standard RS232) ed in uscita hanno un segnale TTL compatibile. La sezione trasmittente ha due driver invertenti con in ingresso TTL compatibile e capaci di erogare a vuoto una tensione di poco meno di +/- 10V, compatibile con lo standard RS232.



Per ricavare le tensioni positive e negative necessarie per garantire i livelli richiesti dalla RS232 è pratica comune utilizzare un duplicatore ed un invertitore di tensione a pompa di carica.



Le figure A e B mostrano come viene ottenuto il raddoppio della tensione. Una immagine che rende l'idea è quella di un contenitore (C2) che preleva acqua da una fonte e la riversa in un secondo contenitore (C1) posto a maggiore altezza. Più in dettaglio:

- Inizialmente il condensatore C2 viene connesso tra massa e Vcc; quindi la corrente (in blu) carica C2 alla tensione di alimentazione (in giallo). Quindi Vc2 = Vcc
- C2 viene successivamente connesso tra Vcc ed un secondo condensatore C1; la tensione ai capi di C1 deve essere uguale alla somma di Vcc e Vc2 e quindi C2 si scarica verso C1, che aumenta la propria tensione rispetto a massa
- Il processo è ripetuto fino a quando la tensione ai capi di C1 è uguale a 2Vcc: in questo caso infatti C2 non si può più scaricare.

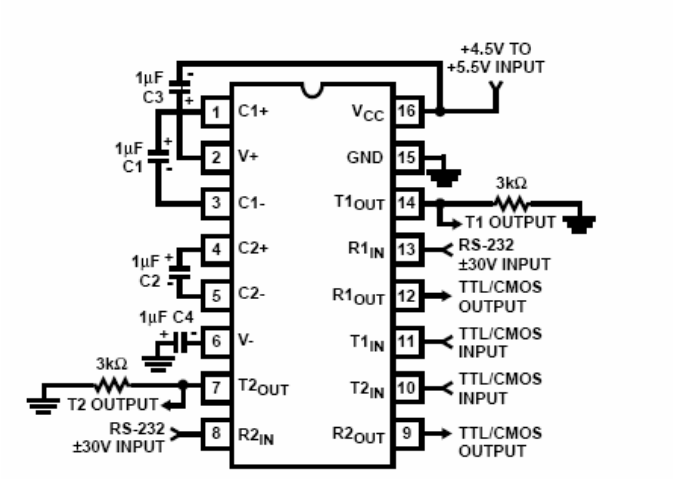
Da notare che, nel funzionamento normale, il processo non può mai interrompersi in quanto il carico collegato a C1, non disegnato, assorbe corrente e quindi tende a scaricare C2 stesso.

Analogamente le figure C e D mostra l'inversione di tensione:

- Inizialmente C2 è caricato alla tensione di alimentazione (magari, come nel disegno da 2Vcc, ricavata con il precedente circuito)
- Quindi C2 è connesso tra massa e C1 avendo cura di invertire le polarità. In questo modo C1 si carica a -2Vcc

Il limite dei circuiti a pompa di carica è la limitata quantità di corrente disponibile: infatti se prelevo corrente da C1 questo tende a scaricarsi, facendo scendere la tensione; la corrente generata da un circuito integrato tipo Max232 è generalmente tutta utilizzata per il solo funzionamento del driver e quindi non è disponibile per altri circuiti.

Il max232 necessita di alcuni componenti esterni collegati all'integrato come mostrato nella figura sotto.

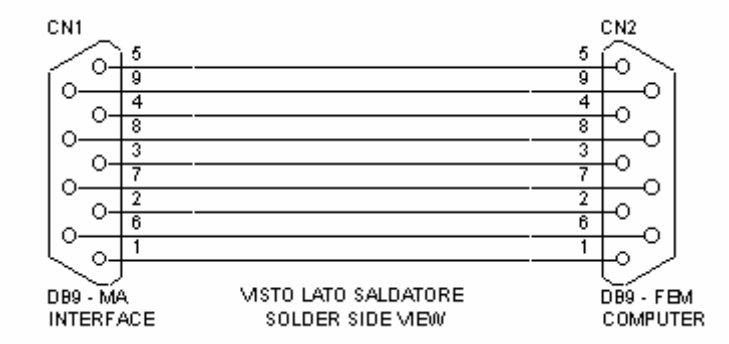


Nella base terrestre abbiamo installato una schedina dedicata alle trasmissioni seriali: vi sono 2 max232, ognuno con una coppia di convertitori TTL-RS232 ed una coppia di convertitori RS232-TTL, mettendo cosi a disposizione due porte seriali: il primo integrato gestisce una porta seriale (la principale) dotata di segnali RTS e CTS per la sincronizzazione dei dati (handshaking), il secondo integrato gestisce la porta ausiliaria, che non è dotata dei sopraccitati segnali.

CTS-RTS

Per spiegare come funziona l'handshake CTS-RTS è necessario sapere che il RTS è un'uscita ed il CTS è un ingresso, e che il CTR del primo dispositivo è connesso al RTS del secondo, ed il CTR del secondo dispositivo è connesso al RTS del primo. Per sincronizzare la comunicazione chi riceve deve dare il consenso a chi trasmette, ossia abbassa il segnale di RTS (lo pone a 0) cosicchè chi deve trasmettere sa che può farlo consultando lo stato del proprio CTS.
Quando un dispositivo apre la porta di comunicazione, come prima cosa deve abbassare il segnale RTS (segnale in uscita) per informare la controparte che è pronta a ricevere, e lo alza solo se il buffer di ricezione si satura. Di conseguenza il CTS (segnale in ingresso) è sempre basso, ossia con il consenso a trasmettere, e risulta alto solo se il ricevente è spento o non connesso, o perché il buffer di ricezione del ricevente è saturo.

PIEDINATURA DELLE PORTE SERIALI



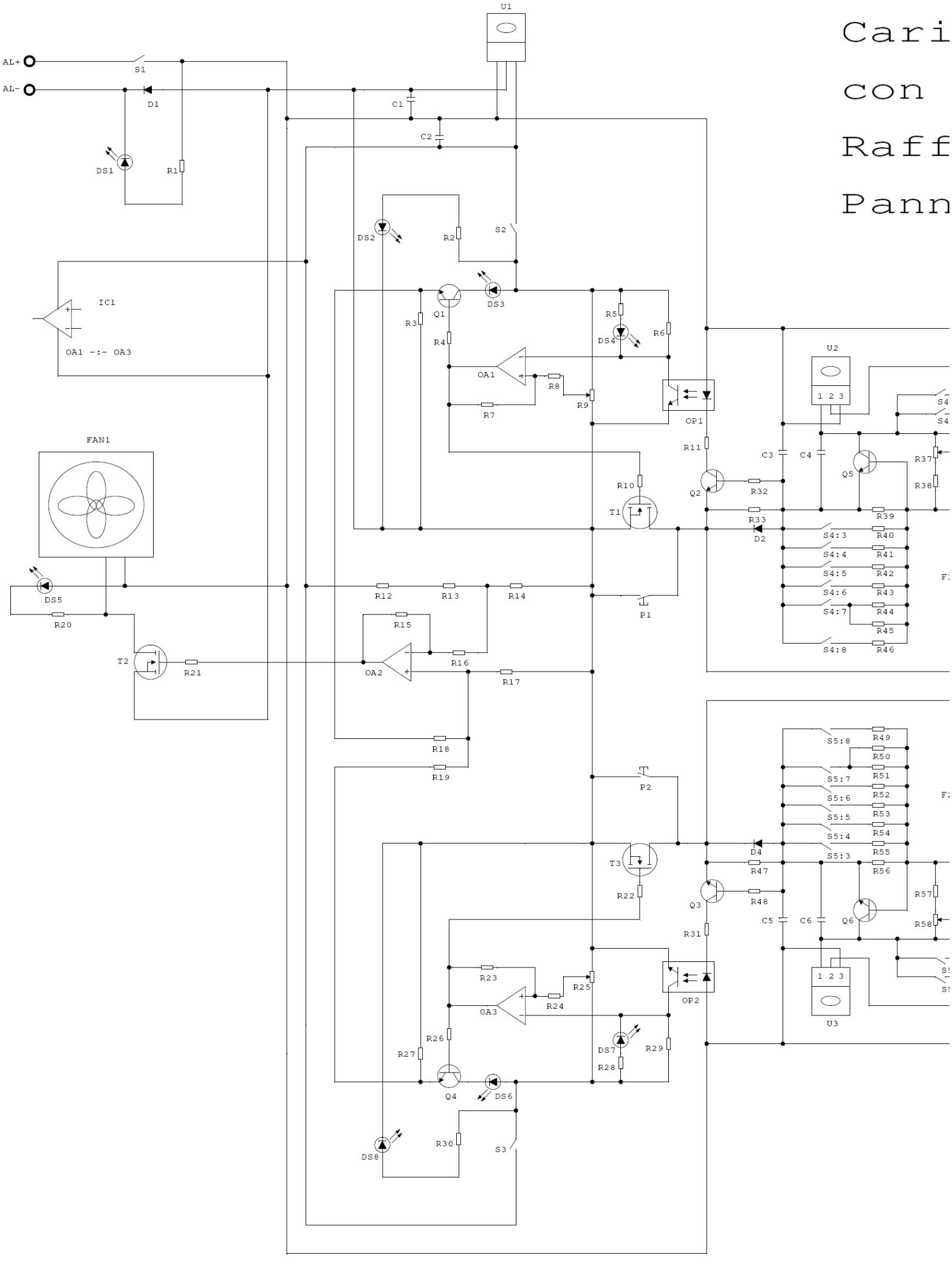
Connettore DTE 9 poli maschio (PC)			
PIN	**SEGNALE**	**NOME SEGNALE**	**TIPO (IN/OUT)**
2	RXD	Ricezione Dati	IN
3	TXD	Trasmissione Dati	OUT
4	DTR	Data Terminal Ready	OUT
5	GND	Ground (Massa segnali)	-
6	DSR	Data Set Ready	IN
7	RTS	Request To Send	OUT
8	CTS	Clear To Send	IN

Connettore DCE 9 poli femmina (modem)			
PIN	**SEGNALE**	**NOME SEGNALE**	**TIPO (IN/OUT)**
2	TXD	Trasmissione Dati	OUT
3	RXD	Ricezione Dati	IN
4	DSR	Data Set Ready	IN
5	GND	Ground (Massa segnali)	-
6	DTR	Data Terminal Ready	OUT
7	CTS	Clear To Send	IN
8	RTS	Request To Send	OUT

SCELTA BATTERIE
E COSTRUZIONE CARICABATTERIE

Per il nostro scopo servivano delle batterie che potessero fornire un'elevata quantità di energia e che fossero leggere. Il problema è stato risolto utilizzando delle batterie al litio: leggere, compatte, potenti, ma molto costose (anche 60€ per batteria) e pericolose se cortocircuitate. Fortunatamente abbiamo ricavato da una vecchia batteria per notebook quattro celle al li-ion (3.7V) da 2200 mAh e dal peso di 44g l'una, tutto ciò al modico prezzo di 10€. Sono delle batterie molto potenti, anche se, quando la gondola è a pieno carico, hanno una durata massima di 30min. Le calle al litio richiedono un transitorio di carica stabile e la corrente di carica non deve superare la quella nominale della batteria, nel nostro caso 2.2A. Abbiamo costruito un caricabatterie che prevede la ricarica di una o due celle con la possibilità di regolare sia la tensione che la corrente di carica. Inoltre ha un circuito di controllo che lo spegne automaticamente quando la batteria è carica. Il circuito è stato realizzato con un amplificatore operazionale usato come comparatore: finito il processo di carica in corrente, la tensione in ingresso al comparatore si azzera ed il relè viene diseccitato. Per il nostro progetto è previsto un caricabatteria "a due piazze" che integra anche una ventola centrale per il raffreddamento dei circuiti. In seguito verrà proposto lo schema elettrico e il PCB di quest'ultimo.

Cari
con
Raff
Pann

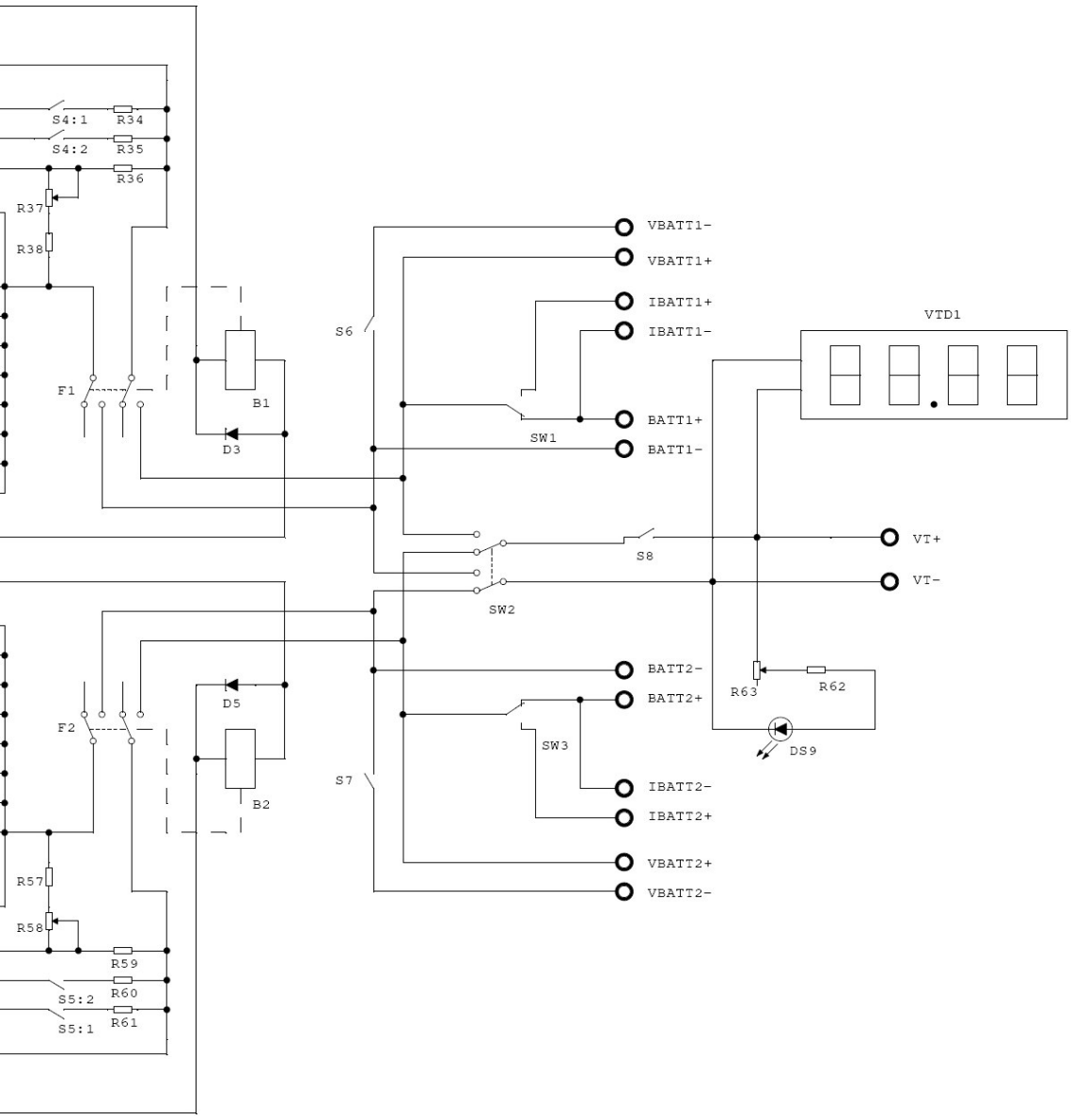


ricaBatterie Celle al Litio

n Gestore di carica,

ffreddamento Automatico e

nnello di Misura

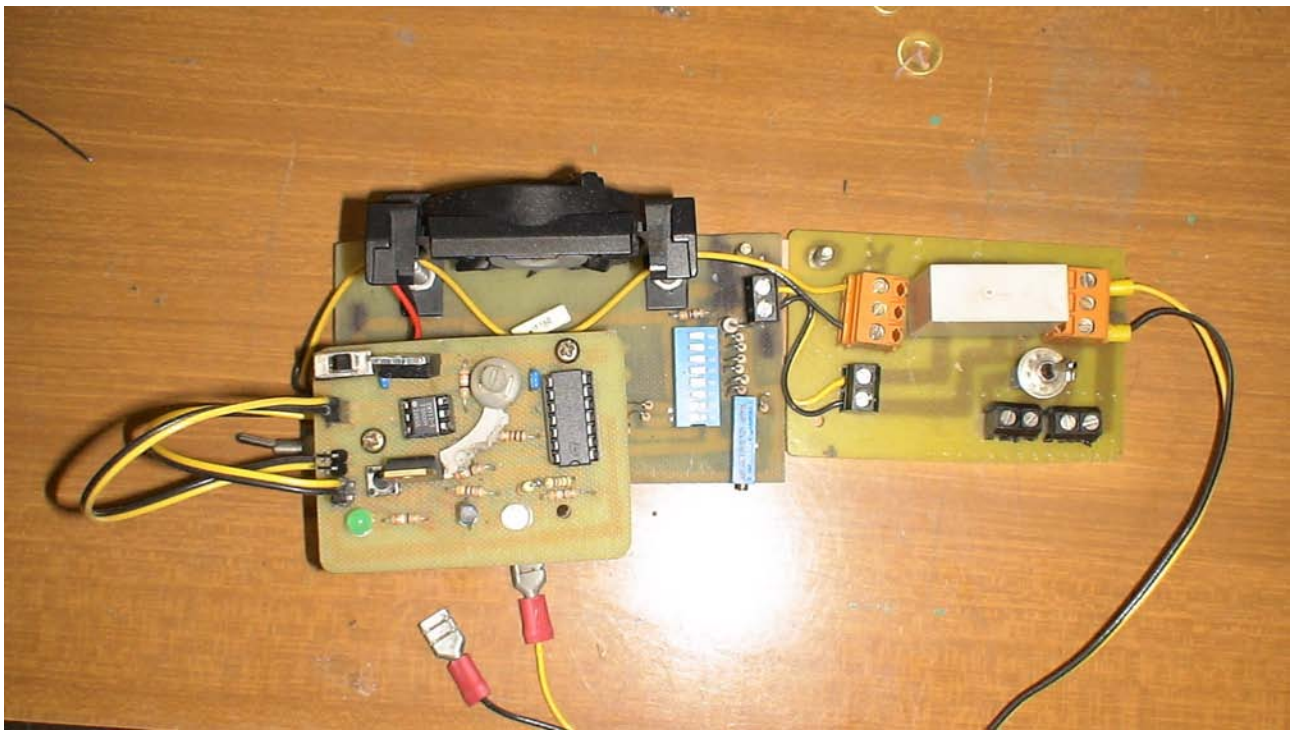


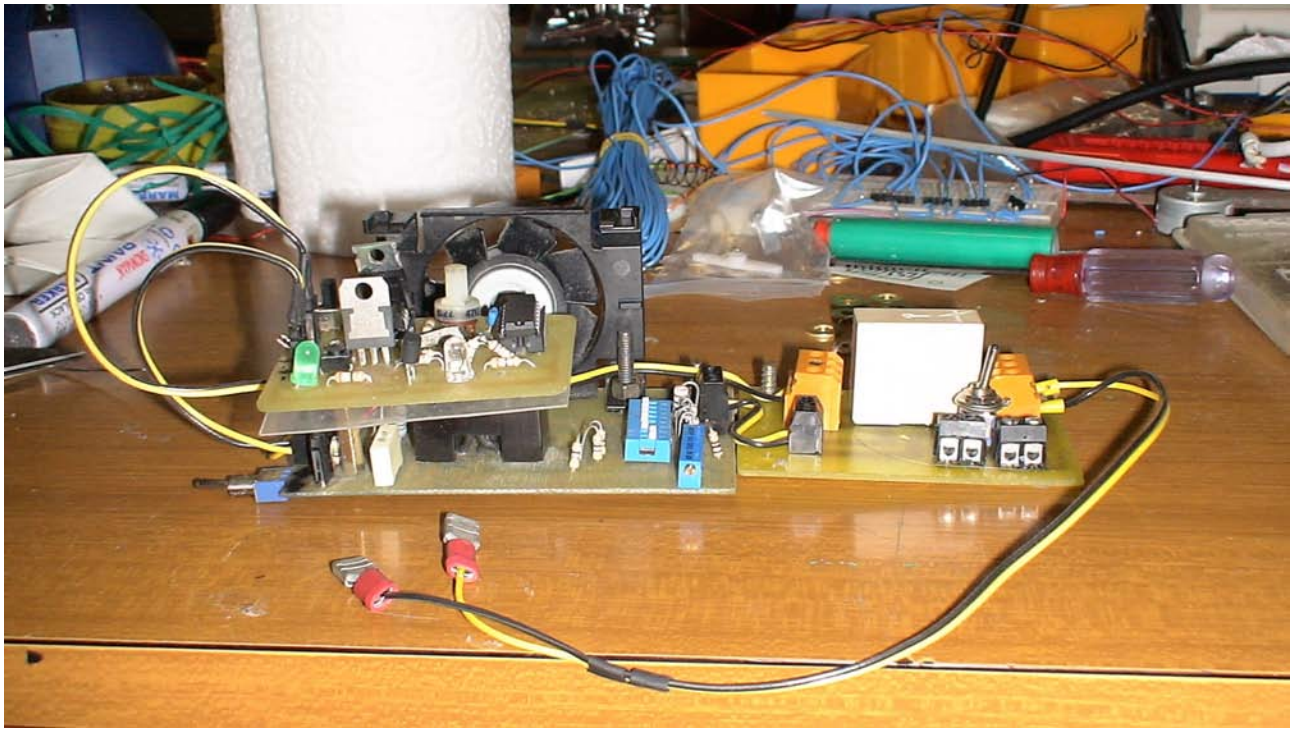
Design By Ruky

NOME	DESCRIZIONE	VALORE	NOTE
R1	Resistenza	930Ω	1/4 W
R2	Resistenza	340Ω	1/4 W
R3	Resistenza	340Ω	1/4 W
R4	Resistenza	4,7KΩ	1/4 W
R5	Resistenza	340Ω	1/4 W
R6	Resistenza	340Ω	1/4 W
R7	Resistenza	4,7MΩ	1/4 W
R8	Resistenza	10KΩ	1/4 W
R9	Trimmer Lineare	47KΩ	1/4 W
R10	Resistenza	340Ω	1/4 W
R11	Resistenza	470Ω	1/4 W
R12	Resistenza	2,2MΩ	1/4 W
R13	Resistenza	2,2MΩ	1/4 W
R14	Resistenza	2,2MΩ	1/4 W
R15	Resistenza	4,7MΩ	1/4 W
R16	Resistenza	10KΩ	1/4 W
R17	Resistenza	100KΩ	1/4 W
R18	Resistenza	4,7KΩ	1/4 W
R19	Resistenza	4,7KΩ	1/4 W
R20	Resistenza	930Ω	1/4 W
R21	Resistenza	340Ω	1/4 W
R22	Resistenza	340Ω	1/4 W
R23	Resistenza	4,7MΩ	1/4 W
R24	Resistenza	10KΩ	1/4 W
R25	Trimmer Lineare	47KΩ	1/4 W
R26	Resistenza	4,7KΩ	1/4 W
R27	Resistenza	340Ω	1/4 W
R28	Resistenza	340Ω	1/4 W
R29	Resistenza	340Ω	1/4 W
R30	Resistenza	340Ω	1/4 W
R31	Resistenza	470Ω	1/4 W
R32	Resistenza	47Ω	1/4 W
R33	Resistenza	47Ω	1/4 W
R34	Resistenza	470Ω	1/4 W
R35	Resistenza	470Ω	1/4 W
R36	Resistenza	470Ω	1/4 W
R37	Trimmer Multigiri	500Ω	1/4 W
R38	Resistenza	1KΩ	1/4 W
R39	Resistenza	47Ω	1/4 W
R40	Resistenza	47Ω	1/4 W
R41	Resistenza	10Ω	1/4 W
R42	Resistenza	10Ω	1/4 W
R43	Resistenza	10Ω	1/4 W
R44	Resistenza	10Ω	1/4 W
R45	Resistenza	10Ω	1/4 W
R46	Resistenza	1Ω	1 W
R47	Resistenza	47Ω	1/4 W
R48	Resistenza	47Ω	1/4 W
R49	Resistenza	1Ω	1 W
R50	Resistenza	10Ω	1/4 W
R51	Resistenza	10Ω	1/4 W
R52	Resistenza	10Ω	1/4 W
R53	Resistenza	10Ω	1/4 W
R54	Resistenza	10Ω	1/4 W
R55	Resistenza	47Ω	1/4 W
R56	Resistenza	47Ω	1/4 W
R57	Resistenza	1KΩ	1/4 W
R58	Trimmer Multigiri	500Ω	1/4 W
R59	Resistenza	470Ω	1/4 W
R60	Resistenza	470Ω	1/4 W
R61	Resistenza	470Ω	1/4 W
R62	Resistenza	340Ω	1/4 W
R63	Trimmer Lineare	1KΩ	1/4 W
C1	Condensatore	100nF	Poliestere
C2	Condensatore	100nF	Poliestere
C3	Condensatore	0,1µF	Poliestere
C4	Condensatore	0,1µF	Poliestere
C5	Condensatore	0,1µF	Poliestere
C6	Condensatore	0,1µF	Poliestere

NOME	DESCRIZIONE	VALORE	NOTE
D1	Diodo	1N4007	
D2	Diodo	1N4007	
D3	Diodo	1N4007	
D4	Diodo	1N4007	
D5	Diodo	1N4007	
DS1	Diodo Led	Verde	5mm - 10mA
DS2	Diodo Led	Verde	5mm - 10mA
DS3	Diodo Led	Rosso	5mm - 10mA
DS4	Diodo Led	Blu	5mm - 10mA
DS5	Diodo Led	Verde	5mm - 10mA
DS6	Diodo Led	Rosso	5mm - 10mA
DS7	Diodo Led	Blu	5mm - 10mA
DS8	Diodo Led	Verde	5mm - 10mA
DS9	Diodo Led	Blu	5mm - 10mA
Q1	Transistor BJT	BC337	Transistor NPN
Q2	Transistor BJT	2N2222A	Transistor NPN
Q3	Transistor BJT	2N2222A	Transistor NPN
Q4	Transistor BJT	BC337	Transistor NPN
Q5	Transistor BJT	2N2222A	Transistor NPN
Q6	Transistor BJT	2N2222A	Transistor NPN
T1	Transistor MOS	IRF520	N-MOS
T2	Transistor MOS	IRF520	N-MOS
T3	Transistor MOS	IRF520	N-MOS
U1	Regolatore Di Tensione	L7805	Fisso
U2	Regolatore Di Tensione	LM317TB	Variabile
U3	Regolatore Di Tensione	LM317TB	Variabile
OA1	Amplificatore Operazionale	LM324N	Integrato da 4 OP-AMP
OA2	Amplificatore Operazionale	LM324N	Integrato da 4 OP-AMP
OA3	Amplificatore Operazionale	LM324N	Integrato da 4 OP-AMP
IC1	Integrato 4 OP-AMP	LM324N	Alimentazione Set Operazionali (Integrato)
OP1	Optoisolatore	CNY17-2	Fototransistor Tipo NPN
OP2	Optoisolatore	CNY17-2	Fototransistor Tipo NPN
P1	Pulsante	Pulsante NA	
P2	Pulsante	Pulsante NA	
S1	Interruttore	Interruttore NA	
S2	Interruttore	Interruttore NA	
S3	Interruttore	Interruttore NA	
S4 (1:8)	Interruttore Dip-Switch	8 Interruttori NA	
S5 (1:8)	Interruttore Dip-Switch	8 Interruttori NA	
S6	Interruttore	Interruttore NA	
S7	Interruttore	Interruttore NA	
S8	Interruttore	Interruttore NA	
SW1	Deviatore	Deviatore	
SW2	Deviatore Doppio	Deviatore Doppio	
SW3	Deviatore	Deviatore	
F1	Deviatore Relè (Doppio)	Deviatore Doppio	Usato come Doppio Interruttore NA
F2	Deviatore Relè (Doppio)	Deviatore Doppio	Usato come Doppio Interruttore NA
B1	Bobina Relè	12V DC	Relè Monostabile
B2	Bobina Relè	12V DC	Relè Monostabile
VTD1	Voltmetro	Scala 0V - 5V	Miniaturizzato Digitale
FAN1	Ventola Raffreddamento	Ventola 12V DC	Dimensioni Minime: 10x10
AL+	Morsetti	Alimentazione +12V	Input Alimentazione 12V DC
AL-	Morsetti	Alimentazione 0V	Input Alimentazione 12V DC
VBATT1+	Morsetti	Misura Tensione + (Cella 1)	Connessione Voltmetro Esterno
VBATT1-	Morsetti	Misura Tensione - (Cella 1)	Connessione Voltmetro Esterno
IBATT1+	Morsetti	Misura Corrente + (Cella 1)	Connessione Amperometro Esterno
IBATT1-	Morsetti	Misura Corrente - (Cella 1)	Connessione Amperometro Esterno
VBATT2+	Morsetti	Misura Tensione + (Cella 2)	Connessione Voltmetro Esterno
VBATT2-	Morsetti	Misura Tensione - (Cella 2)	Connessione Voltmetro Esterno
IBATT2+	Morsetti	Misura Corrente + (Cella 2)	Connessione Amperometro Esterno
IBATT2-	Morsetti	Misura Corrente - (Cella 2)	Connessione Amperometro Esterno
VT+	Morsetti	Misura Tensione + (con switch)	Connessione Voltmetro Esterno
VT-	Morsetti	Misura Tensione - (con switch)	Connessione Voltmetro Esterno
BATT1+	Morsetti	Cella Litio 1 +	Connessione alle Batterie
BATT1-	Morsetti	Cella Litio 1 -	Connessione alle Batterie
BATT2+	Morsetti	Cella Litio 2 +	Connessione alle Batterie
BATT2-	Morsetti	Cella Litio 2 -	Connessione alle Batterie

A causa della scarsità di tempo a disposizione è stato costruito un prototipo di caricabatterie singolo (per una sola cella), dotato di operazionale di gestione carica, ventola di raffreddamento e relè di stacco.





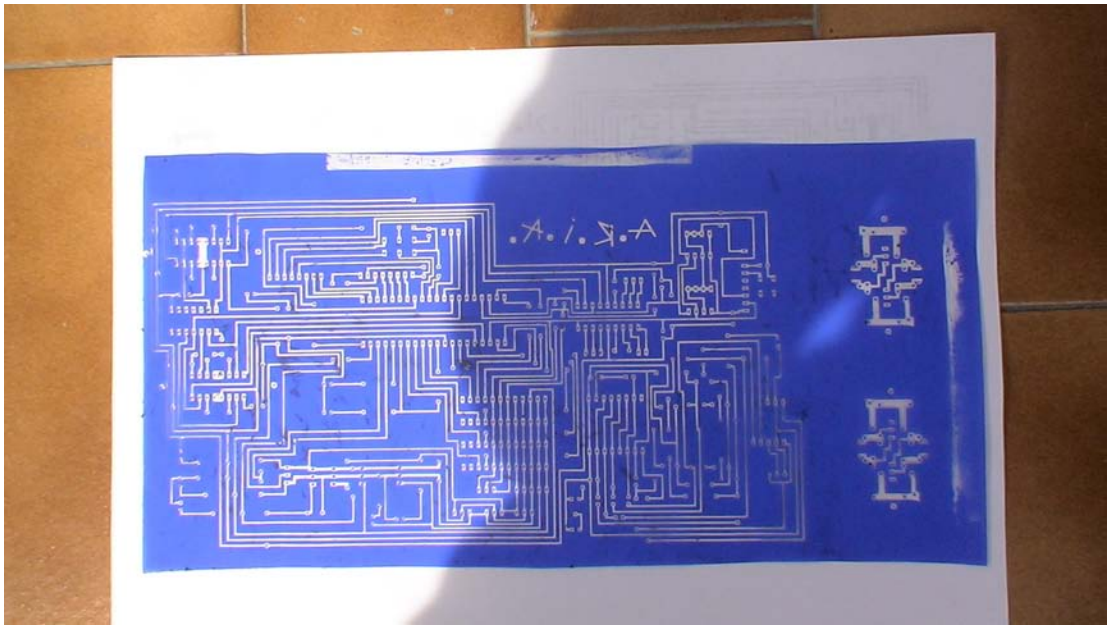
HARDWARE

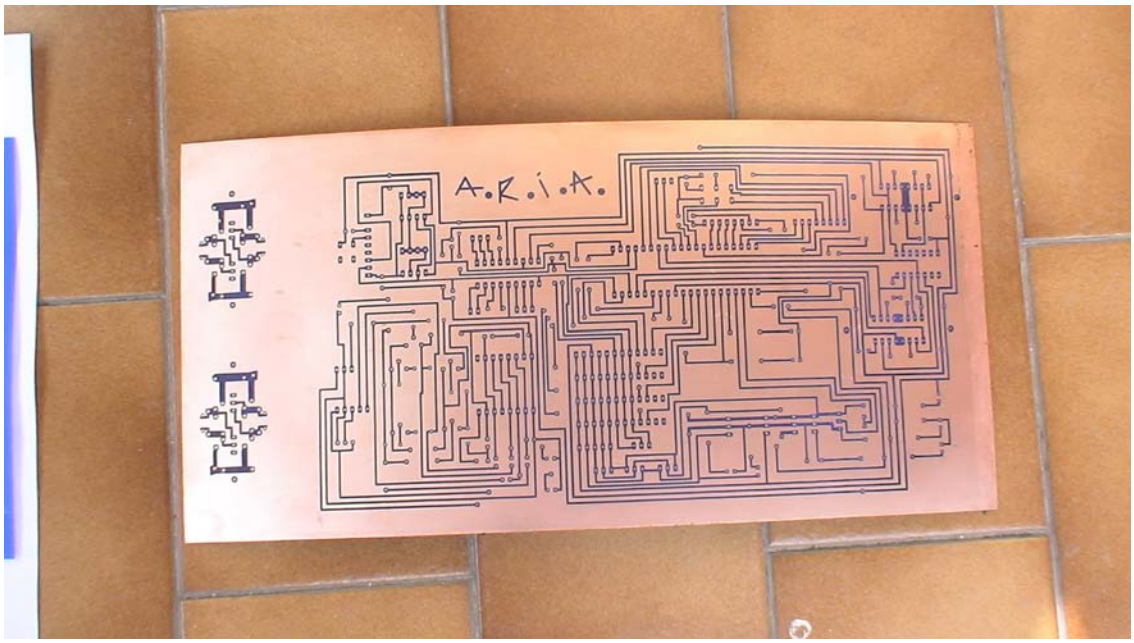
REALIZZAZIONE DELLE SCHEDINE

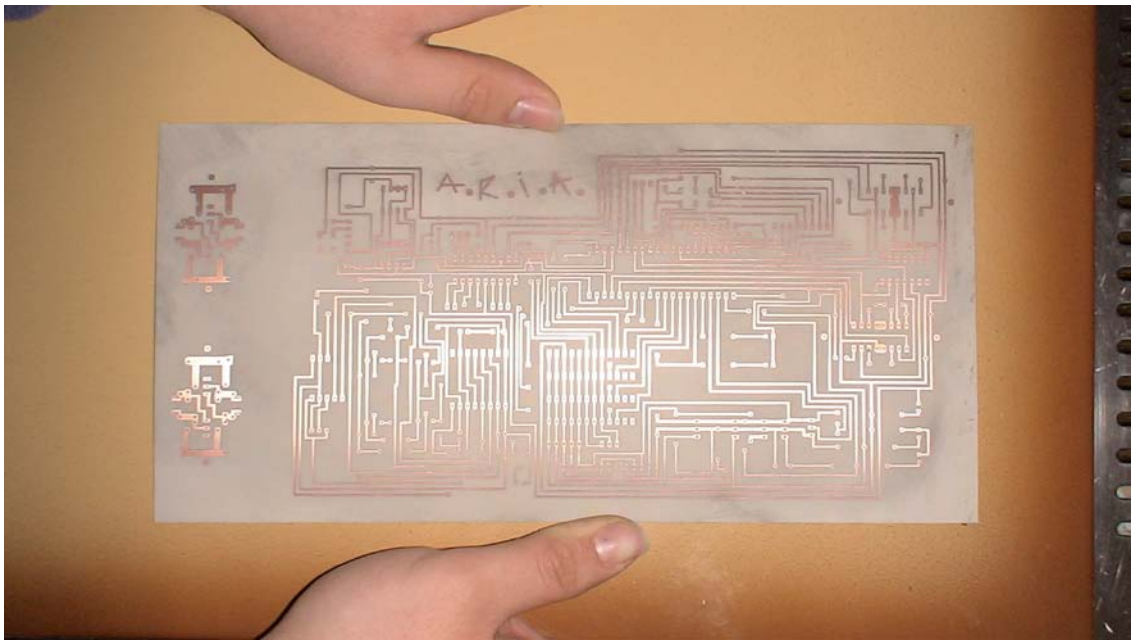
Le schedine del dirigibile sono state realizzate per mezzo di basette ramate.
Il circuito (PCB) è stato disegnato con FidoCad al PC e stampato su particolari fogli blu chiamati PnP, made in USA.




Il materiale blu di tali fogli reagisce con il toner di una stampante Laser. Attraverso un trasferimento a caldo, con un normale ferro da stiro, è poi possibile far aderire al rame della basetta il disegno delle piste da noi realizzato.
In seguito la basetta viene messa nell'acido, che "mangia il rame" ovunque tranne dove è presente la stampa.
Terminato il processo si pulisce la schedina con del diluente Nitro: sono rimaste solo le piste di rame.
I prossimi passi sono la foratura e la saldatura dei componenti.



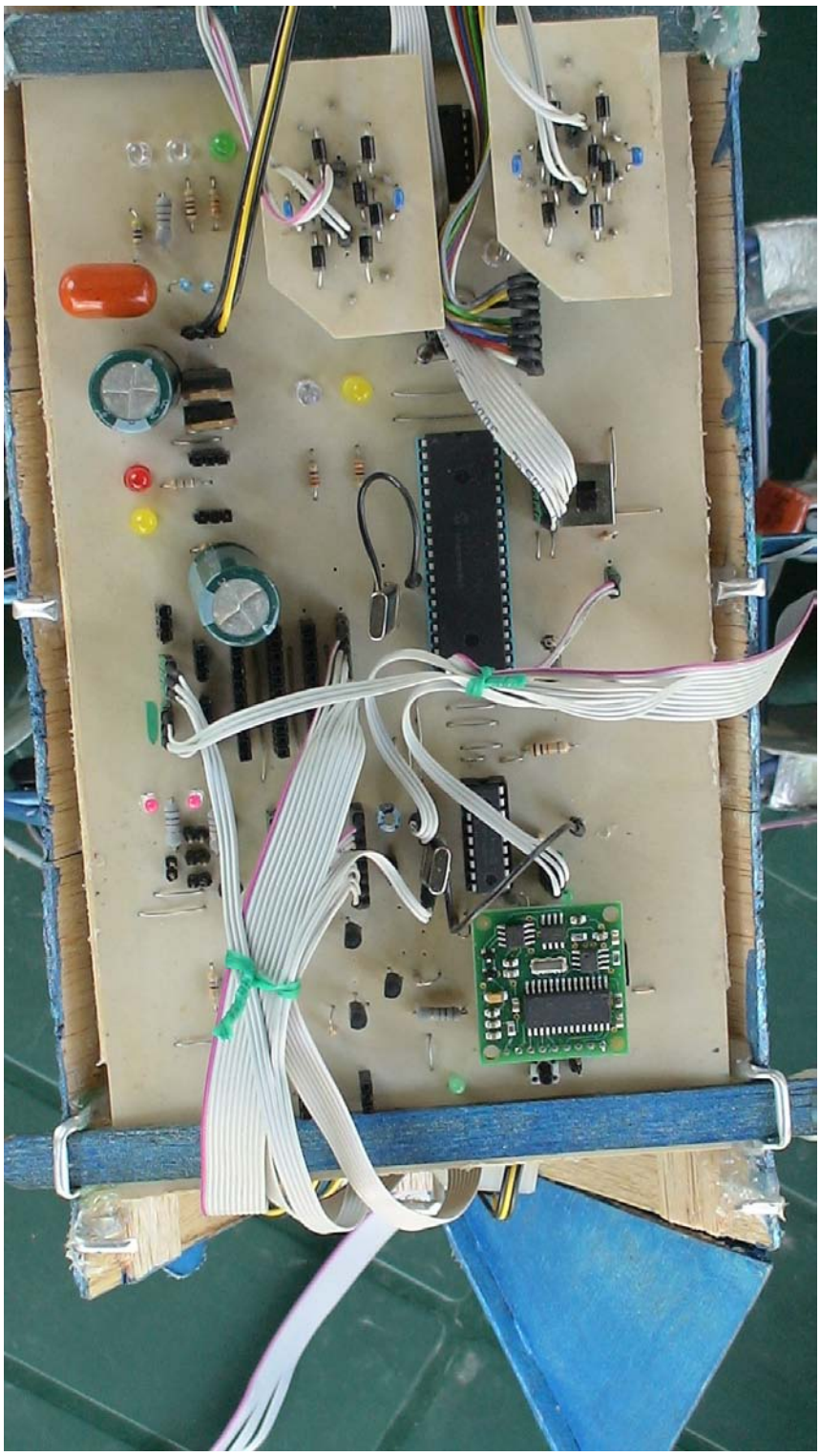




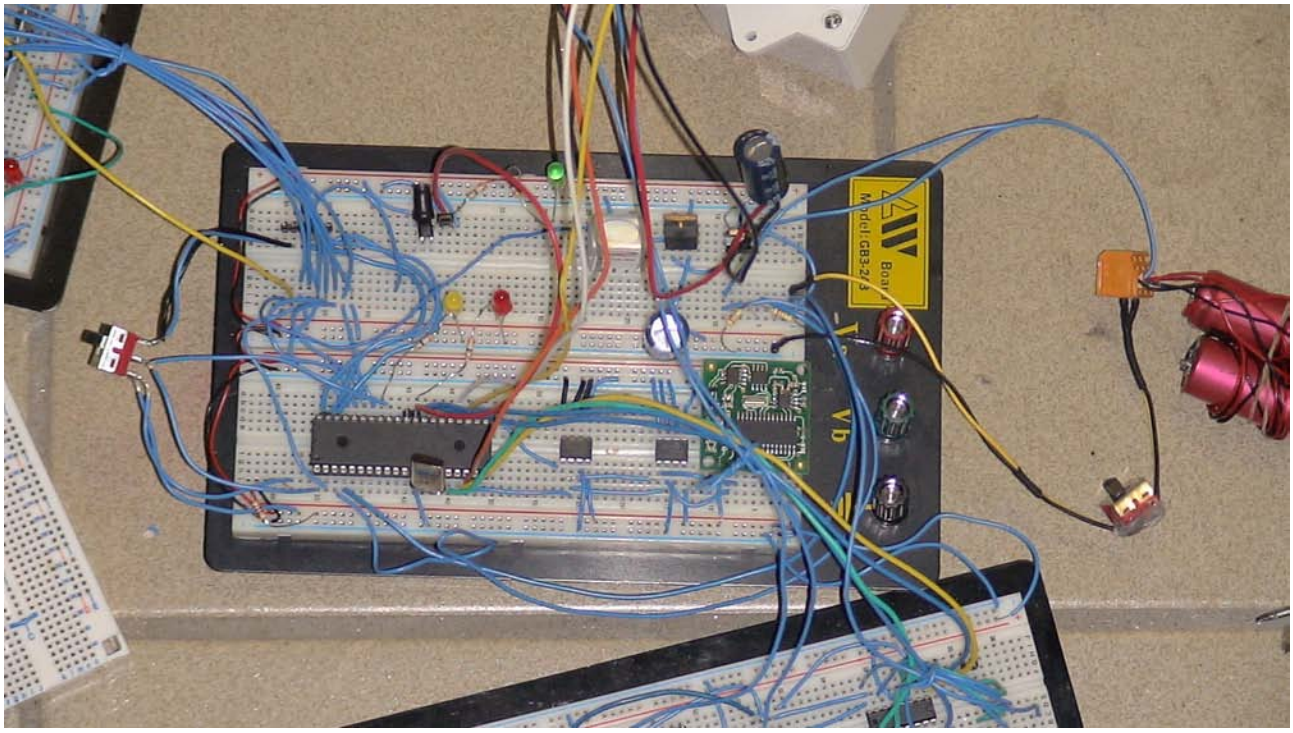
La maggior parte dei componenti delle nostre schedine sono estraibili e quindi sostituibili in caso di danneggiamento: stabilizzatori, mos, sensori, microcontrollori, integrati vari….

SCHEDINE DIRIGIBILE

Il pilotaggio del dirigibile avviene per mezzo di una scheda elettronica che occupa tutta la cabina.

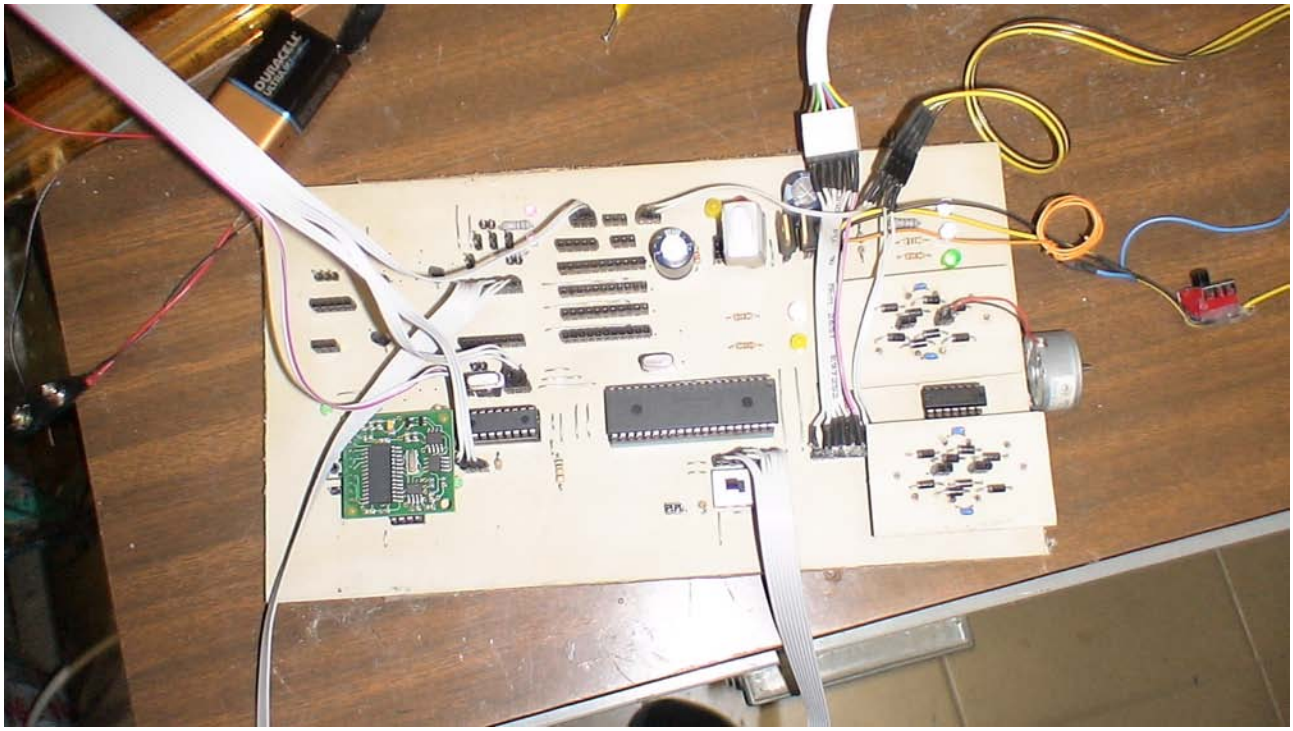


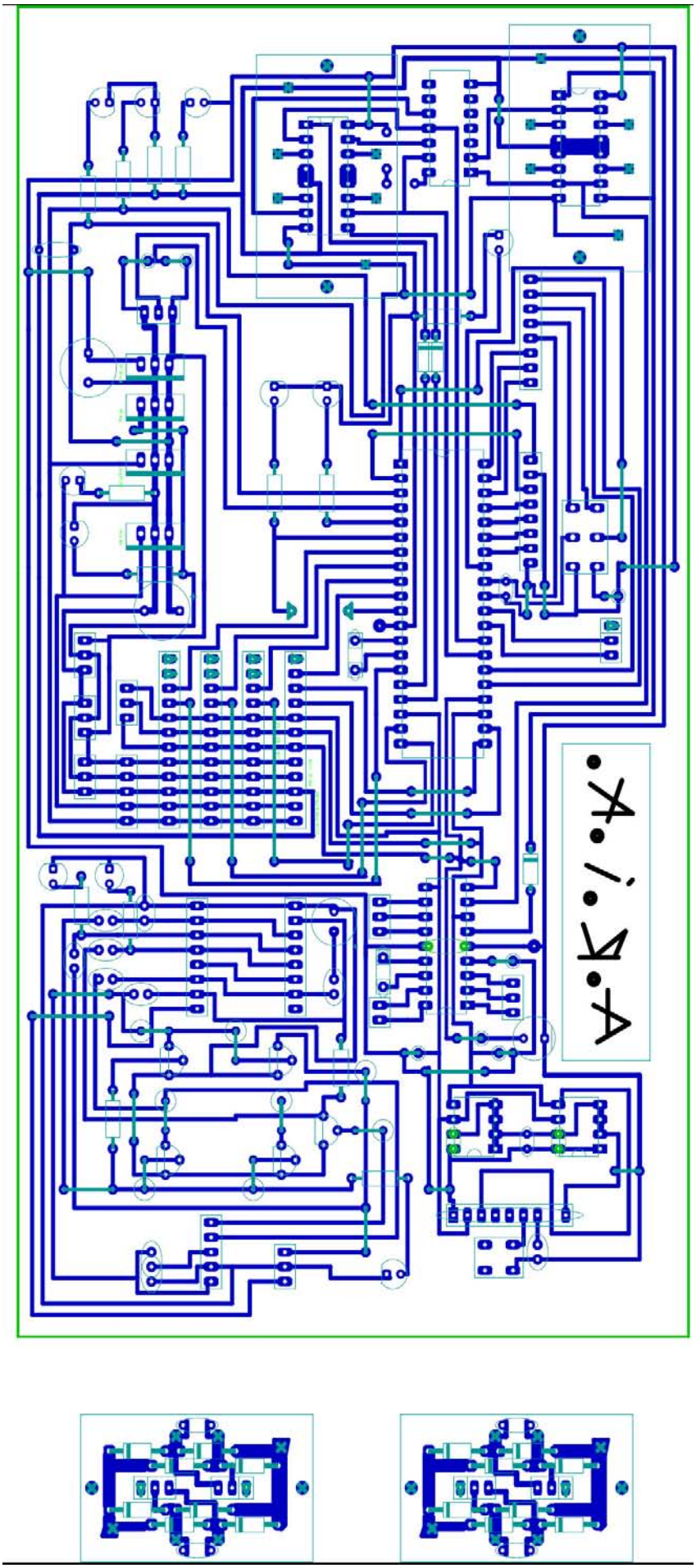
La scheda è stata realizzata in base alle conoscenze apprese realizzando i circuiti sulle bradbords.

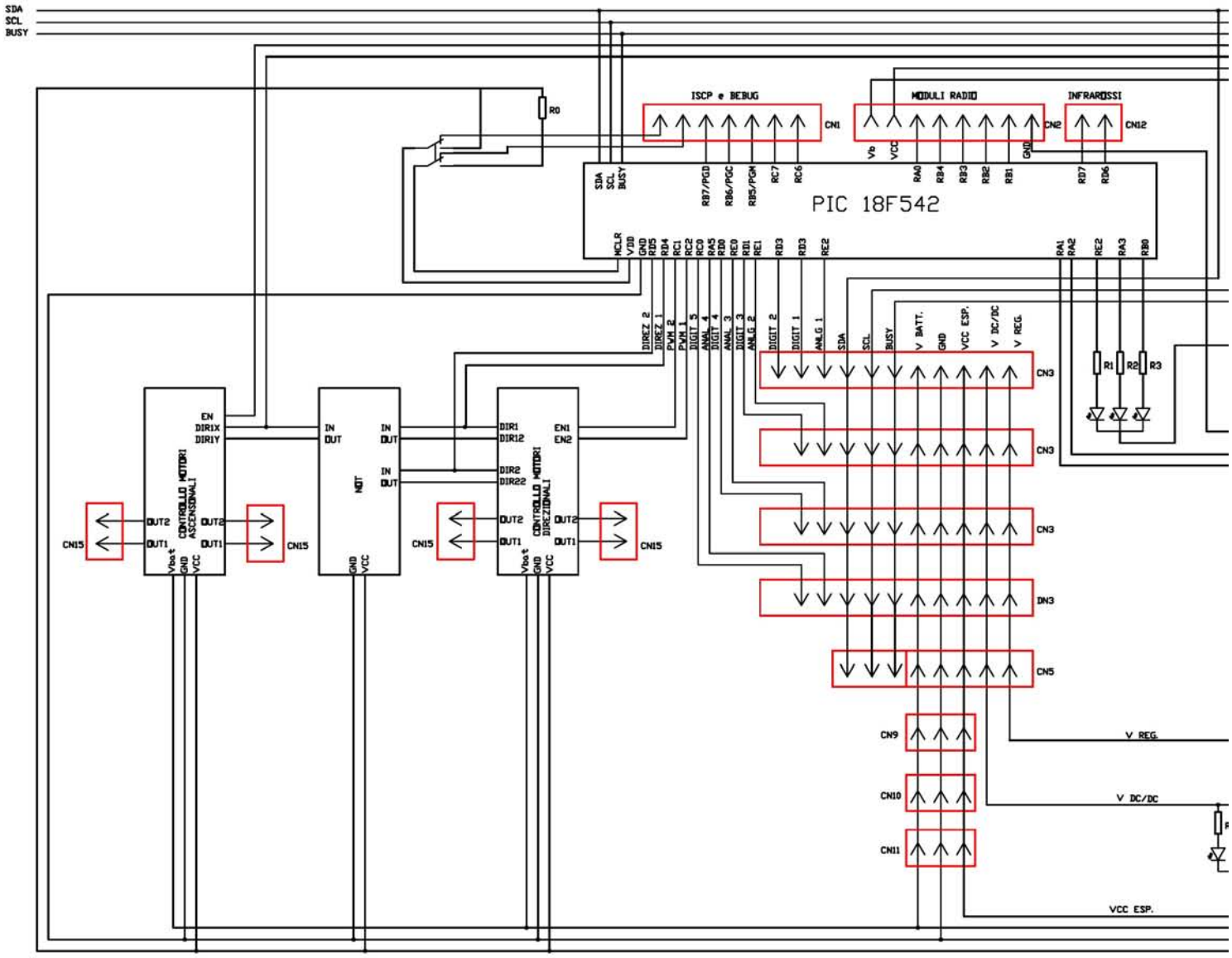


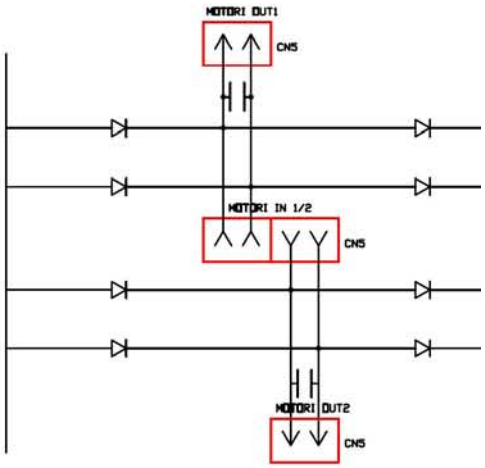
In alto nella scheda si possono osservare i connettori dei motori, sopraelevati rispetto al resto. È stata realizzata apposta una schedina che venisse inserita sopra i due chip di controllo motore per risparmiare spazio. Sotto infatti si notano gli integrati per il controllo dei motori: direzionali a sinistra e ascensionali a destra.
Tra i due è presente un altro integrato: è un inverter (74LS04) per il pilotaggio della direzione dei motori.
In basso troviamo il pic principale (core). Alla sua sinistra vi sono gli stabilizzatori per l'alimentazione dei vari dispositivi mentre alla sua destra i connettori per la programmazione (con relativo commutatore Esecuzione/Programmazione), per la comunicazione radio e per i sensori infrarossi.
Più in basso si trovano i connettori delle espansioni, a sinistra, il pic fuffy per la gestione dei sensori nel centro.
Ai suoi lati sono presenti i connettori per il collegamento dei sensori (derivazioni). Nella parte sinistra, subito dopo i connettori delle espansioni sono presenti i connettori dei moduli radio e l'elettronica per il controllo dei moduli ibridi Aurel (se usati).

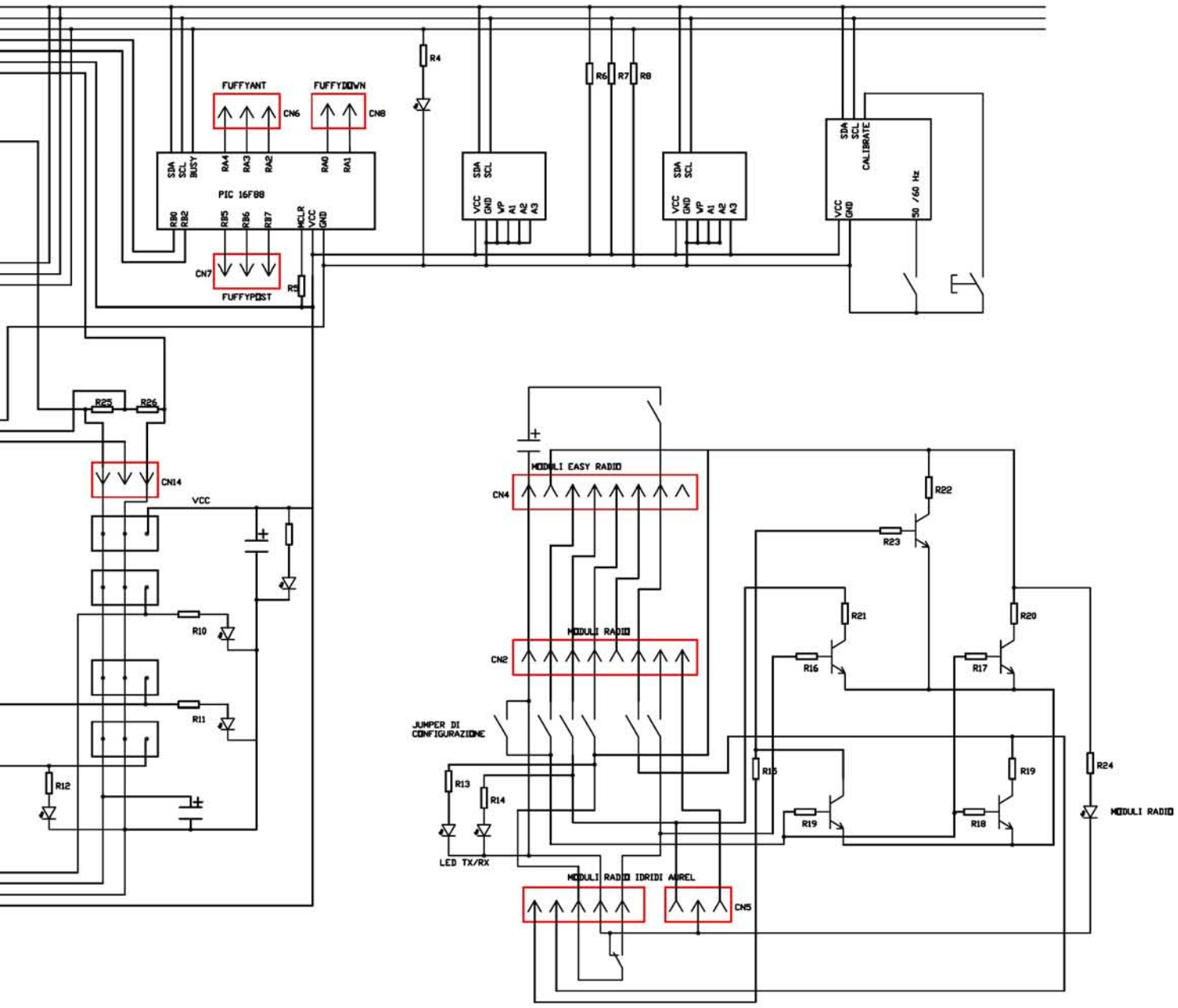
In centro, nella parte terminale della schedina si può vedere la bussola digitale, estraibile grazie all'apposito connettore. Sotto di essa vi sono le due memorie I2C: la RAM e la EEPROM.
Nella schedina sono sparsi diversi led di segnalazione colorati, alcuni di alimentazione ed altri comandati dal pic.
Osservando le piste della schedina si può facilmente notare il BUS I2C che la attraversa da cima a fondo, per diramarsi al centro verso le espansioni.
Si può anche osservare il bus di alimentazione che intercollega le espansioni con gli stabilizzatori.





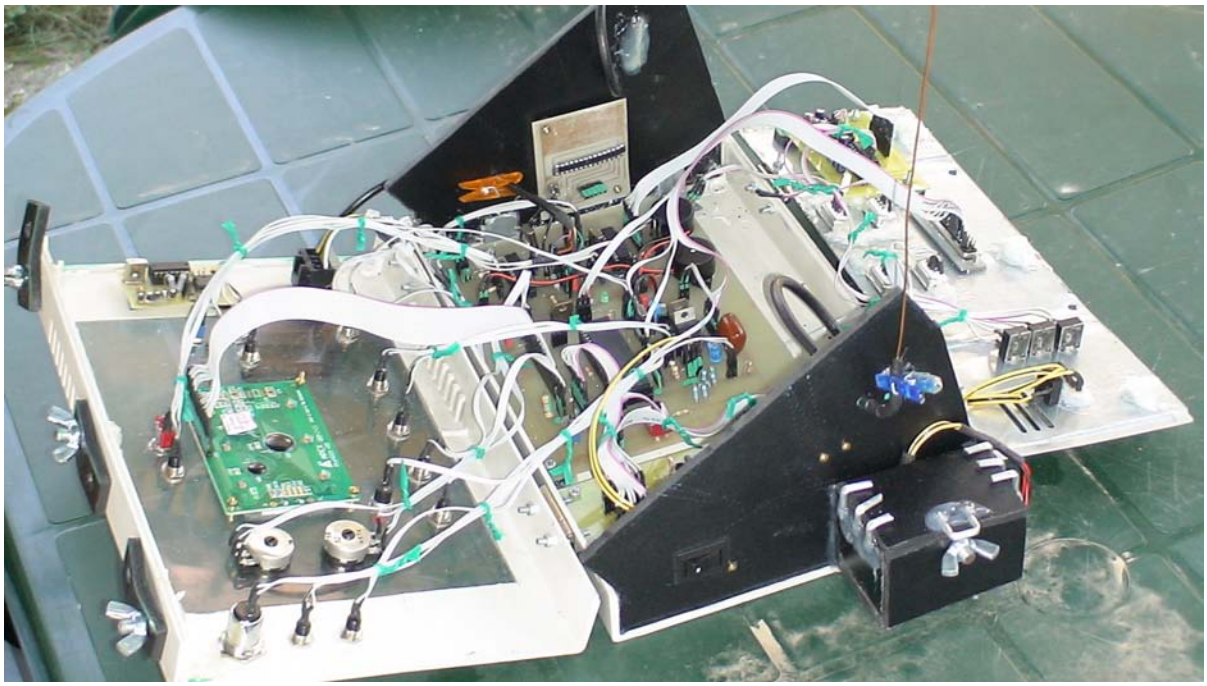

SDA
SCL
BUSY
ISCP e BEBUG
MODULI RADIO
INFRAROSSI
CN1
CN2
CN12
R0
PIC 18F542
CN3
CN5
CN9
CN10
CN11
CN15
V REG.
V DC/DC
VCC ESP.
CONTROLLO MOTORI ASCENSIONALI
CONTROLLO MOTORI DIREZIONALI
MOT


MOTORI OUT1
CN5
MOTORI IN 1/2
CN5
MOTORI OUT2
CN5

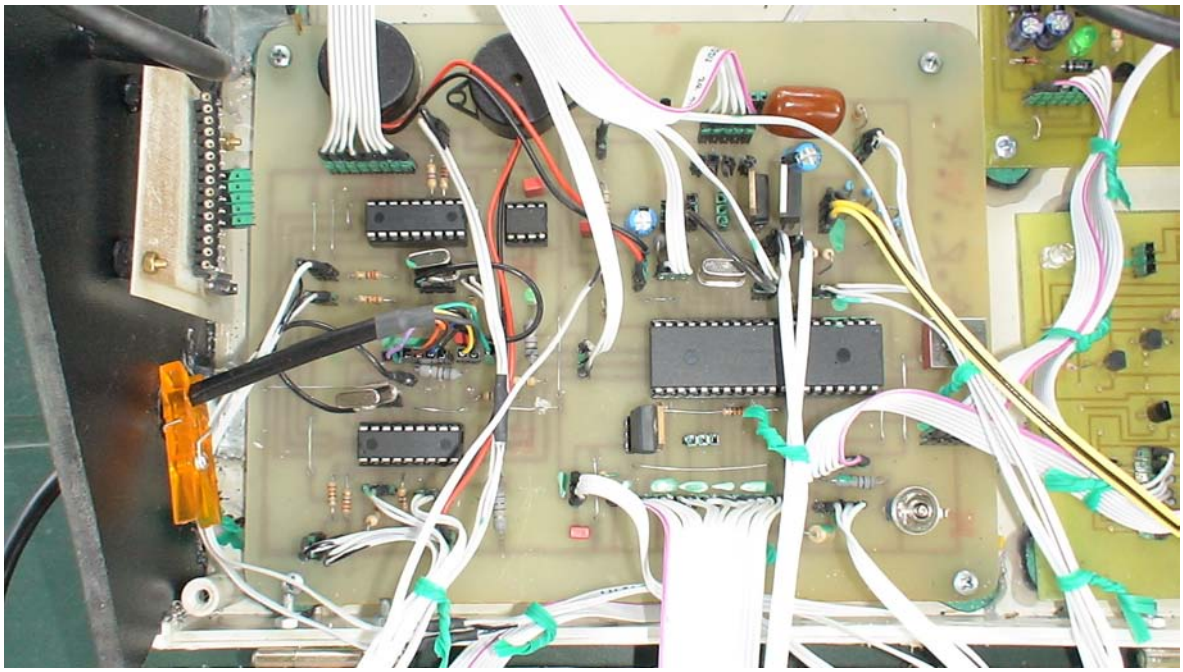

FUFFYANT
FUFFYDOWN
CN6
CN8
R4
R6
R7
R8
SDA
SCL
BUSY
RA4
RA3
RA2
RA0
RA1
PIC 16F88
RB0
RB2
RB5
RB6
RB7
MCLR
VCC
GND
CN7
FUFFYPOST
R9
SDA
SCL
VCC
GND
VP
A1
A2
A3
CALIBRATE
50 /60 Hz
R25
R26
CN14
VCC
R10
R11
R12
MODULI EASY RADIO
CN4
MODULI RADIO
CN2
JUMPER DI
CONFIGURAZIONE
R13
R14
LED TX/RX
MODULI RADIO IDRIDI AUREL
CN5
R22
R23
R21
R20
R16
R17
R15
R19
R18
R24
MODULI RADIO

SCHEDINE BASE

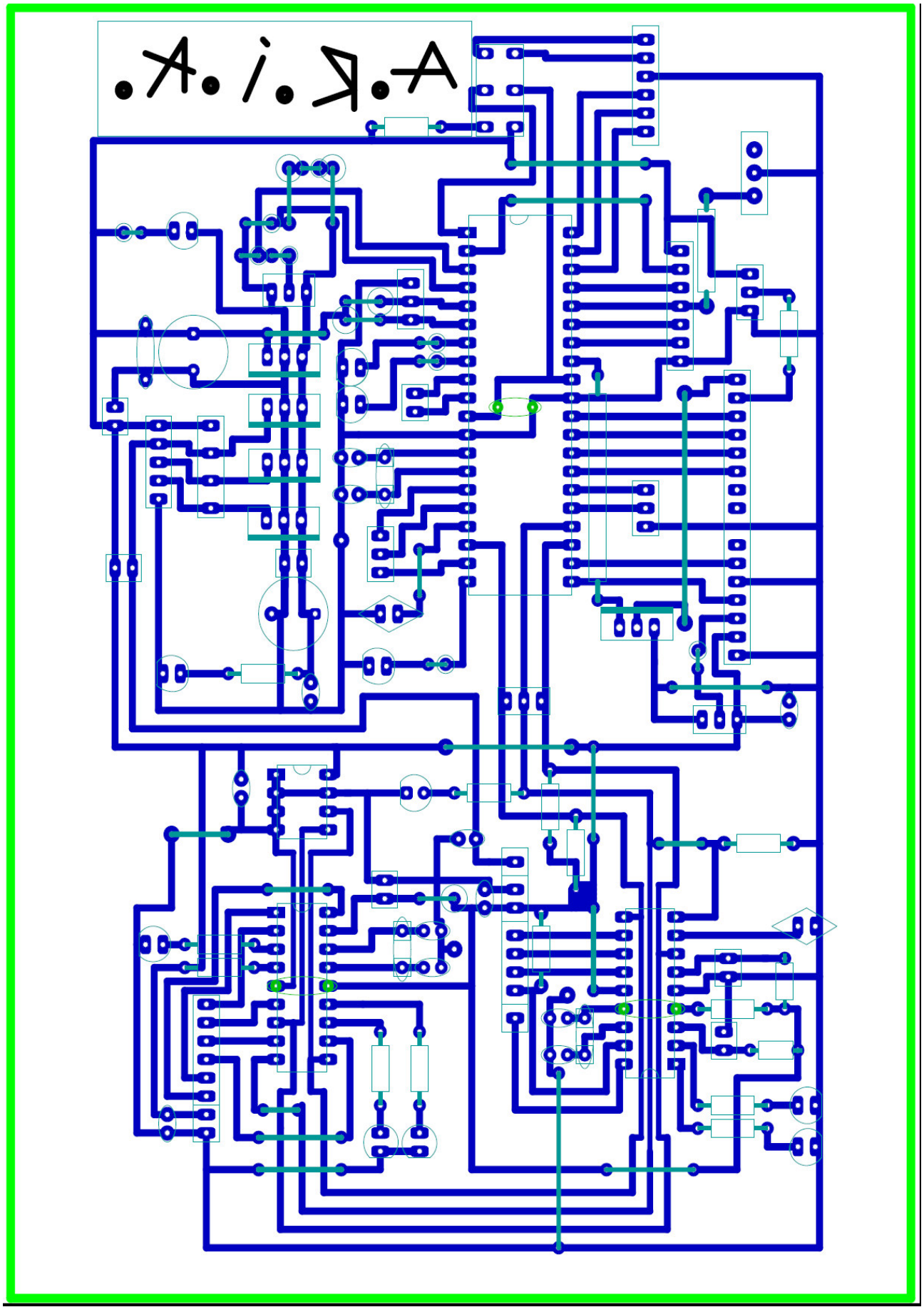
L’elettronica della base è stata suddivisa su più schede per ottimizzare gli spazi del contenitore, che altrimenti sarebbe stato troppo piccolo per contenere un’unica scheda.

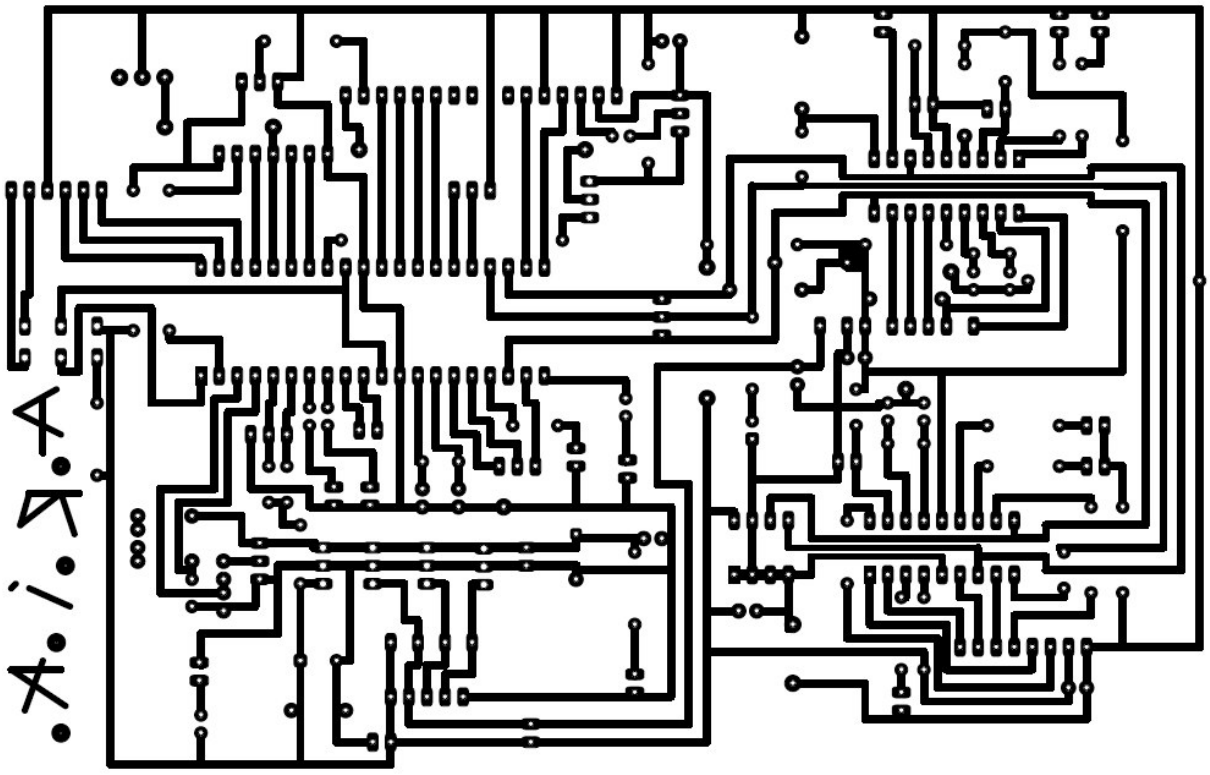


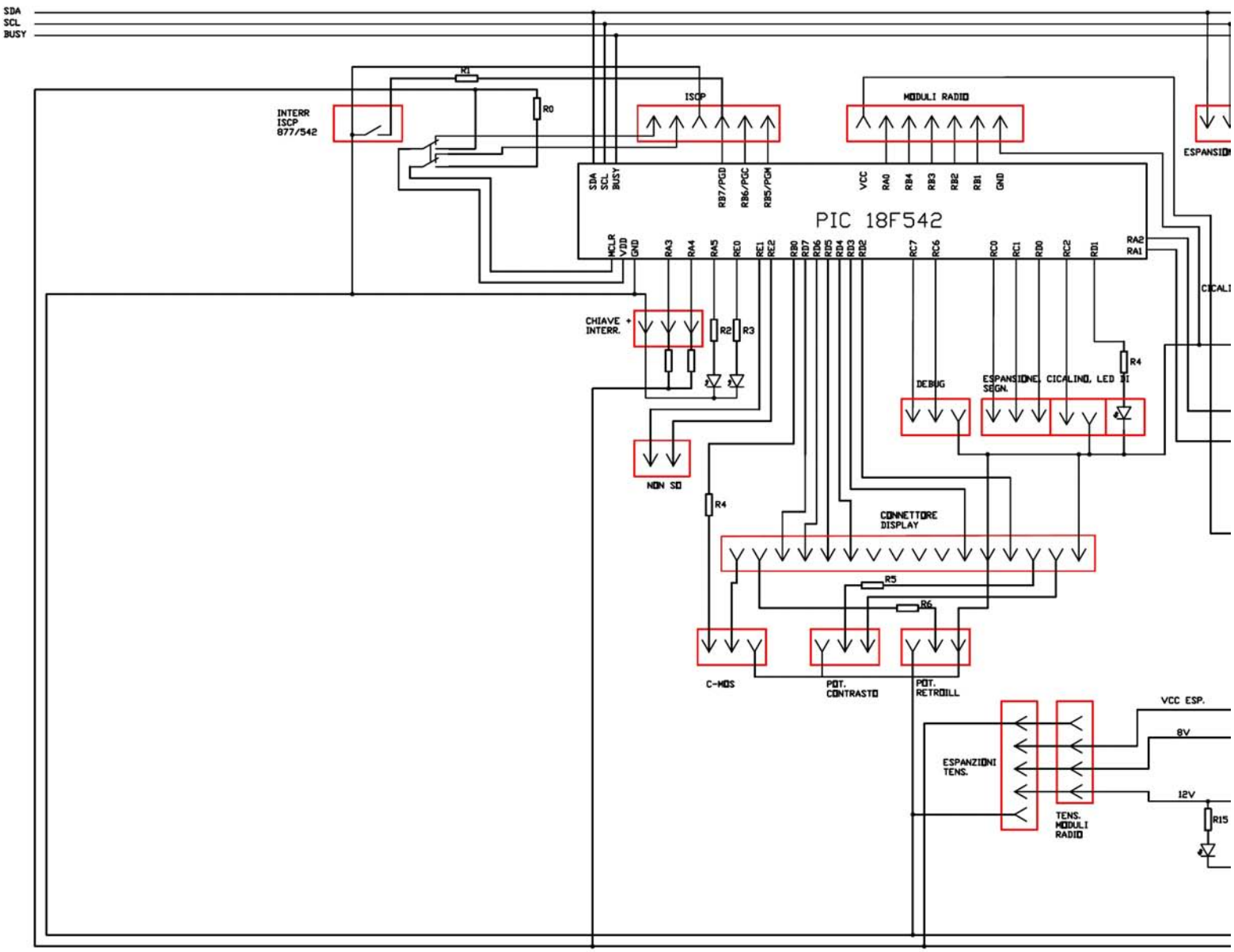
La scheda principale contiene tutti i componenti più importanti.

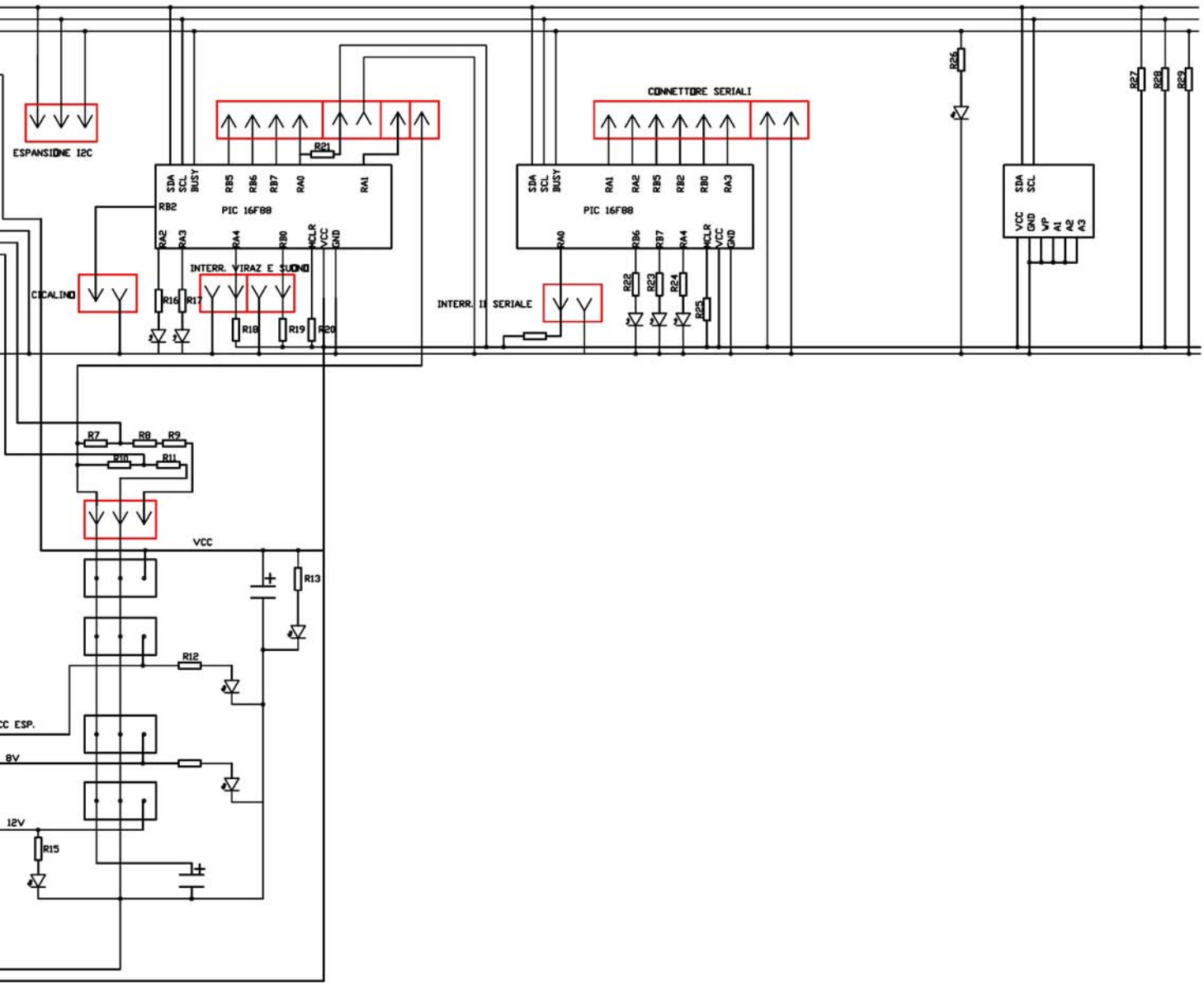


In alto a sinistra possiamo vedere il pic gestore delle comunicazioni tra i dispositivi di input/output e la memoria RAM di swap.
Subito sopra vi sono i connettori delle due porte seriali, una principale per il comando ed una ausiliaria per il debug esterno, che devono essere connessi alla schedina di interfaccia MAX232.
In basso, sulla sinistra invece possiamo vedere il pic del JoyPad, il cui compito è quello di leggere il Controller della Playstation. Nei suoi pressi possiamo notare il connettore del GamePad.
Entrambi i pic presentano dei connettori da due poli nelle vicinanze: sono i led di segnalazione e gli interruttori per abilitare varie opzioni.
Attraverso un interruttore è possibile abilitare la seriale 2 e un led segnalerà l'avvenuta modifica dell'impostazione.
Attraverso altri due interruttori si può abilitare/disabilitare la vibrazione e gli effetti sonori del joypad. Un connettore è quindi dedicato al cicalino.
Sulla destra, in centro vi è il pic principale. Esso gestisce la comunicazione radio, infatti possiamo vedere subito sotto il connettore della comunicazione radio. Può essere programmati via ICSP, come si può notare dalla presenza del connettore dedicato.
Il connettore più evidente è senza dubbio quello del display, da cui esce un gigantesco cavo flat. Il display è lcd è un mezzo di comunicazione con l'utente ed il pic principale gestisce tutte le segnalazioni utente.
Altri connettori presenti sono led di segnalazione, canali che vanno all'espansione della base, un ulteriore cicalino, tutti in prossimità del picmaster. Interessante è il connettore tripolare all'estremità superiore destra del pic: tale connettore collega al microcontrollore la chiave di avviamento ed il pulsante per far scorrere le schermate del display.
In alto ci sono gli stabilizzatori per l'alimentazione, da non confondere con il MOS per l'accensione del display in basso vicino al relativo connettore.
Il bus I2C attraversa la schedina formando una L: parte dal pic Principale e collega tutti gli altri fino alla memoria.

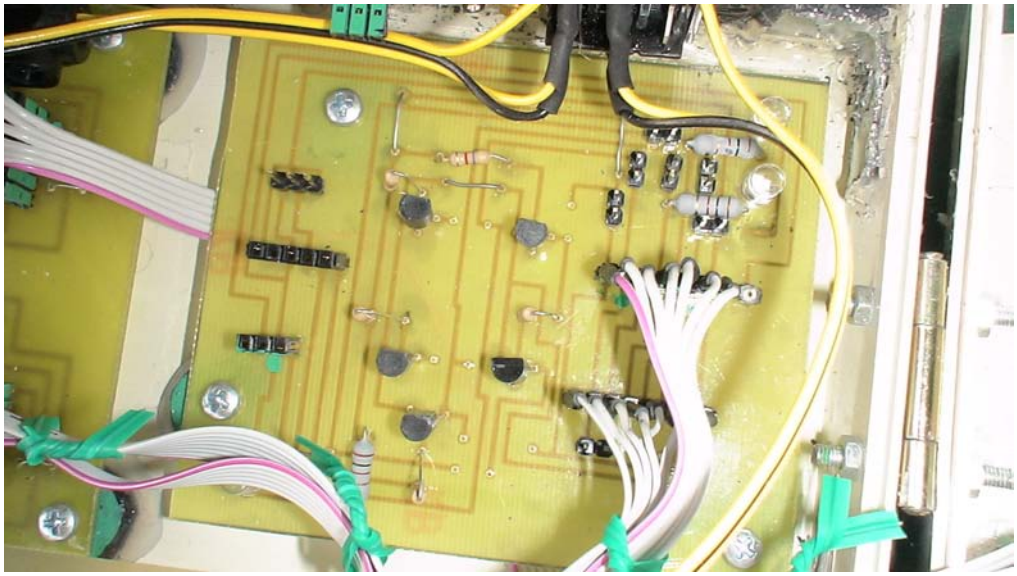




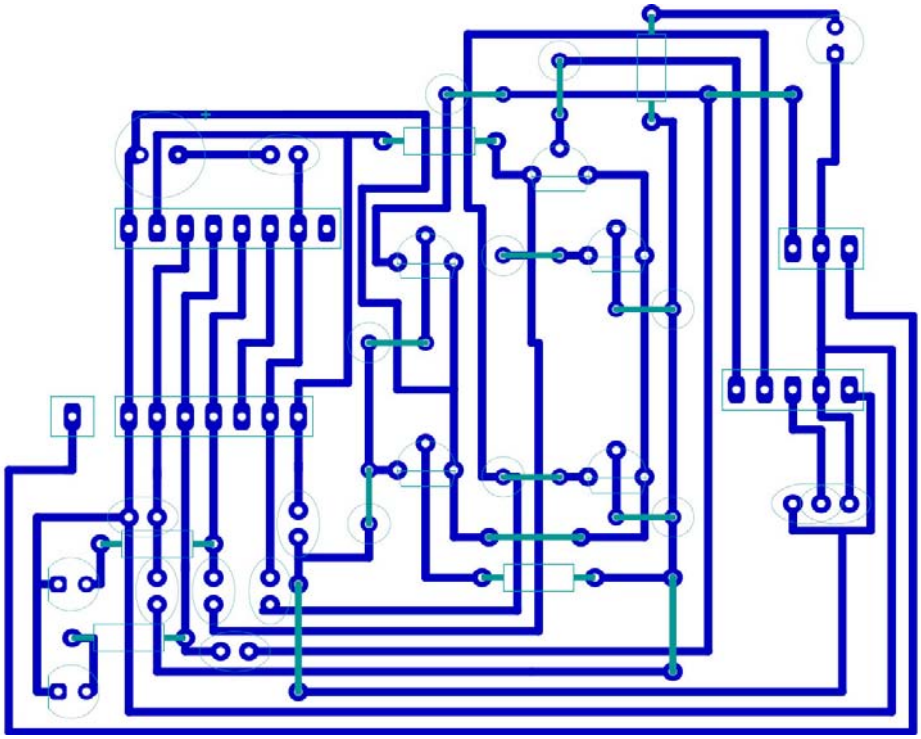

SDA
SCL
BUSY
R1
R0
INTERR
ISCP
877/542
ISCP
MODULI RADIO
ESPANSIONI
SDA
SCL
BUSY
RB7/PGD
RB6/PGC
RB5/PGM
VCC
RA0
RB4
RB3
RB2
RB1
GND
PIC 18F542
MCLR
VDD
GND
RA3
RA4
RA5
RE0
RE1
RE2
RB0
RD7
RD6
RD5
RD4
RD3
RB2
RC7
RC6
RC0
RC1
RD0
RC2
RD1
RA2
RA1
CICALI
CHIAVE +
INTERR.
R2
R3
R4
DEBUG
ESPANSIONE, CICALINO, LED
SEGN.
NON SO
R4
CONNETTORE
DISPLAY
R5
R6
C-MOS
POT.
CONTRASTO
POT.
RETROILL
VCC ESP.
8V
12V
ESPANZIONI
TENS.
TENS.
MODULI
RADIO
R15

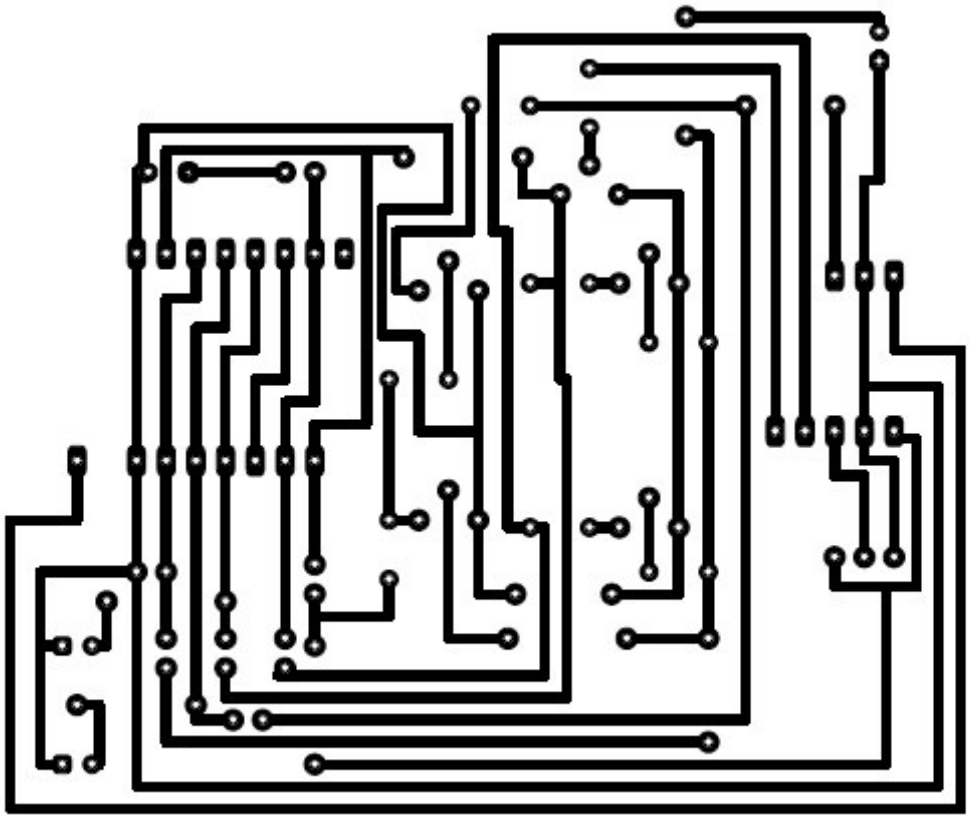

ESPANSIONE I2C
CONNETTORE SERIALI
PIC 16F88
PIC 16F88
INTERR. VIRAZ E SUONO
CICALINO
INTERR. II SERIALE
SDA
SCL
BUSY
VCC
GND
MCLR
R21
R22
R23
R24
R25
R26
R27
R28
R29
R16
R17
R18
R19
R20
R7
R8
R9
R10
R11
R12
R13
R15
VCC
8V
12V

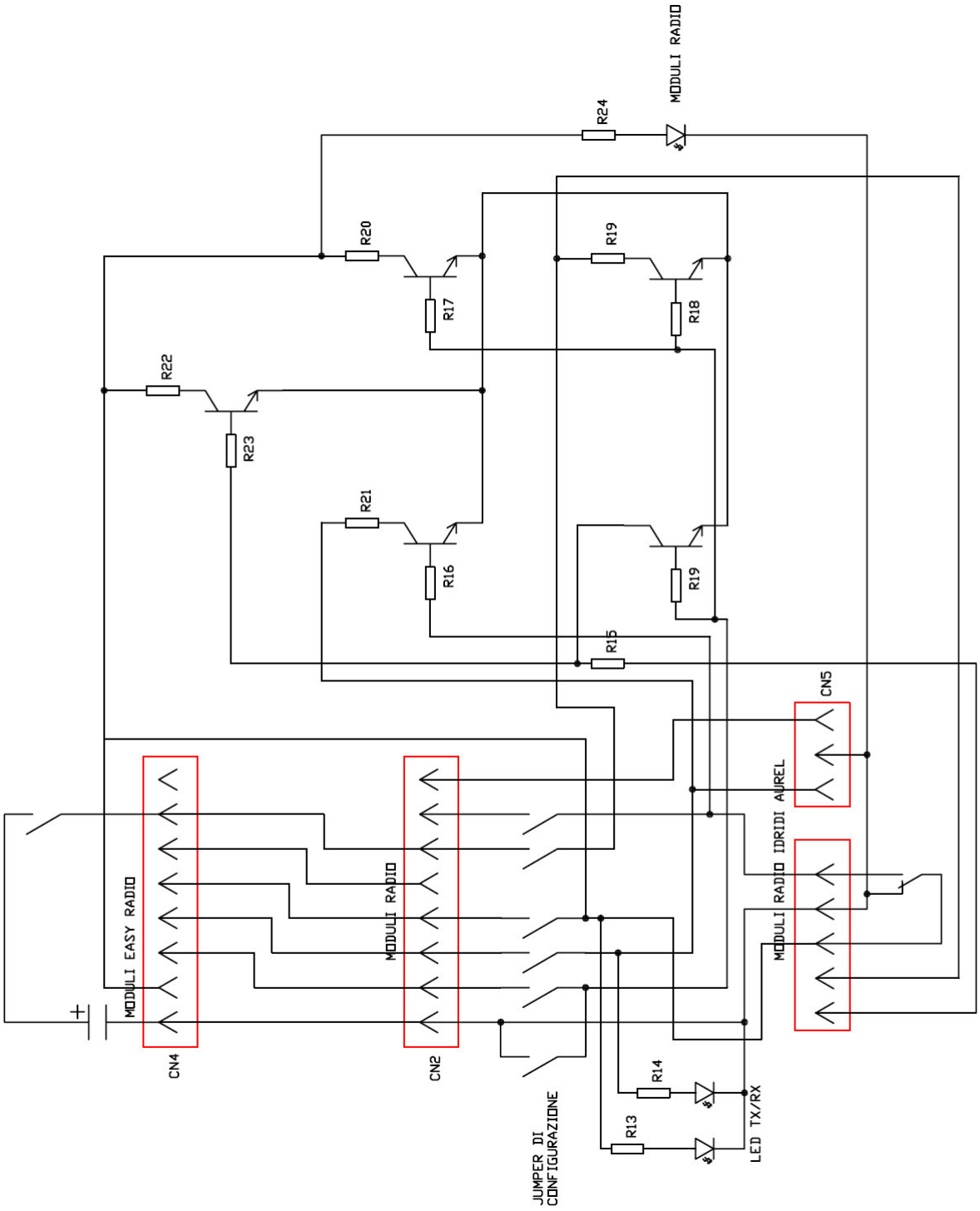
La schedina dei moduli radio gestisce i due tipi di moduli: quelli ibridi e quello easy-radio.



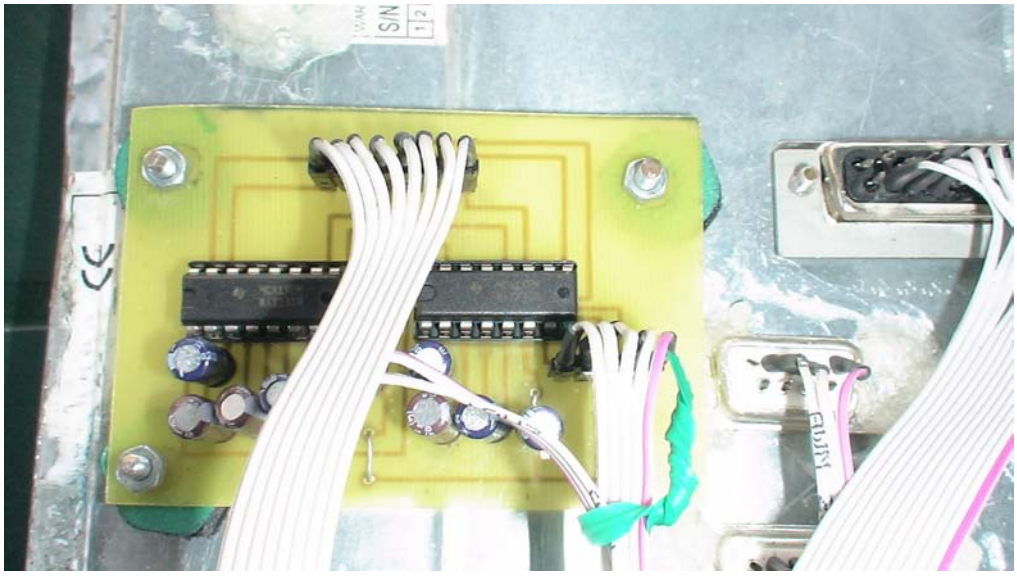
Presenta un connettore in ingresso che deve essere collegato a quello sulla scheda principale. Gli altri connettori vanno alle schedine di supporto dei moduli radio.
Si può notare l'interfaccia per la gestione dei moduli ibridi realizzata con transistors BC547.



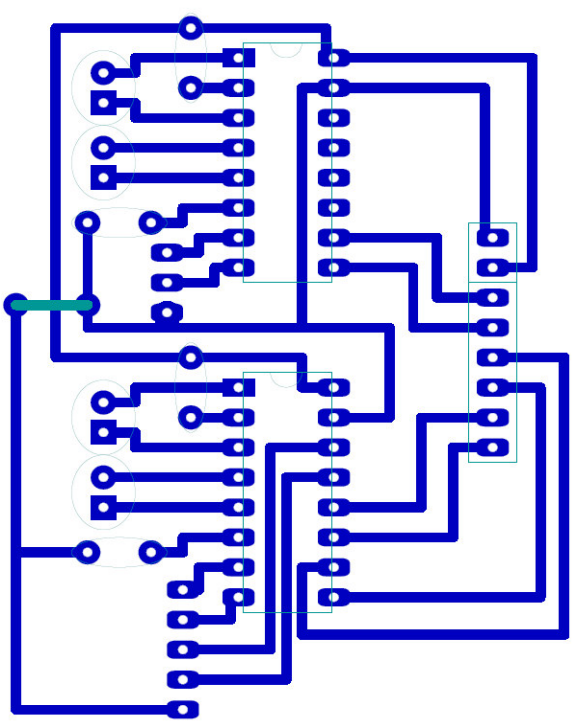


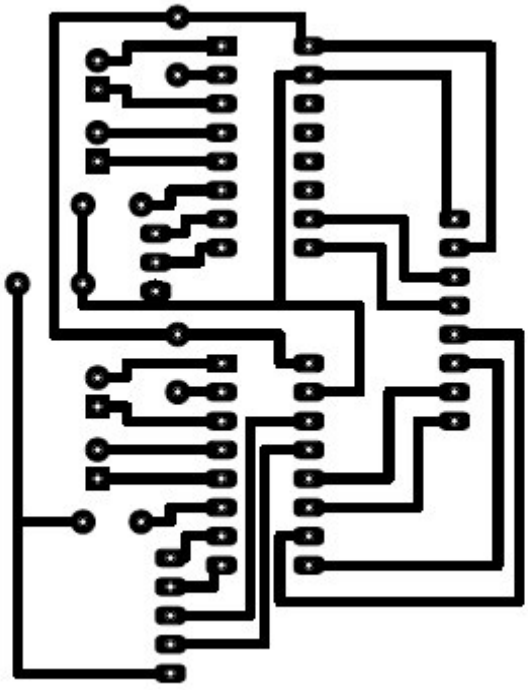

MODULI RADIO
R24
R20
R19
R17
R18
R22
R23
R21
R16
R19
R15
CN5
MODULI EASY RADIO
MODULI RADIO
MODULI RADIO IDRIDI AUREL
CN4
CN2
R14
R13
JUMPER DI CONFIGURAZIONE
LED TX/RX

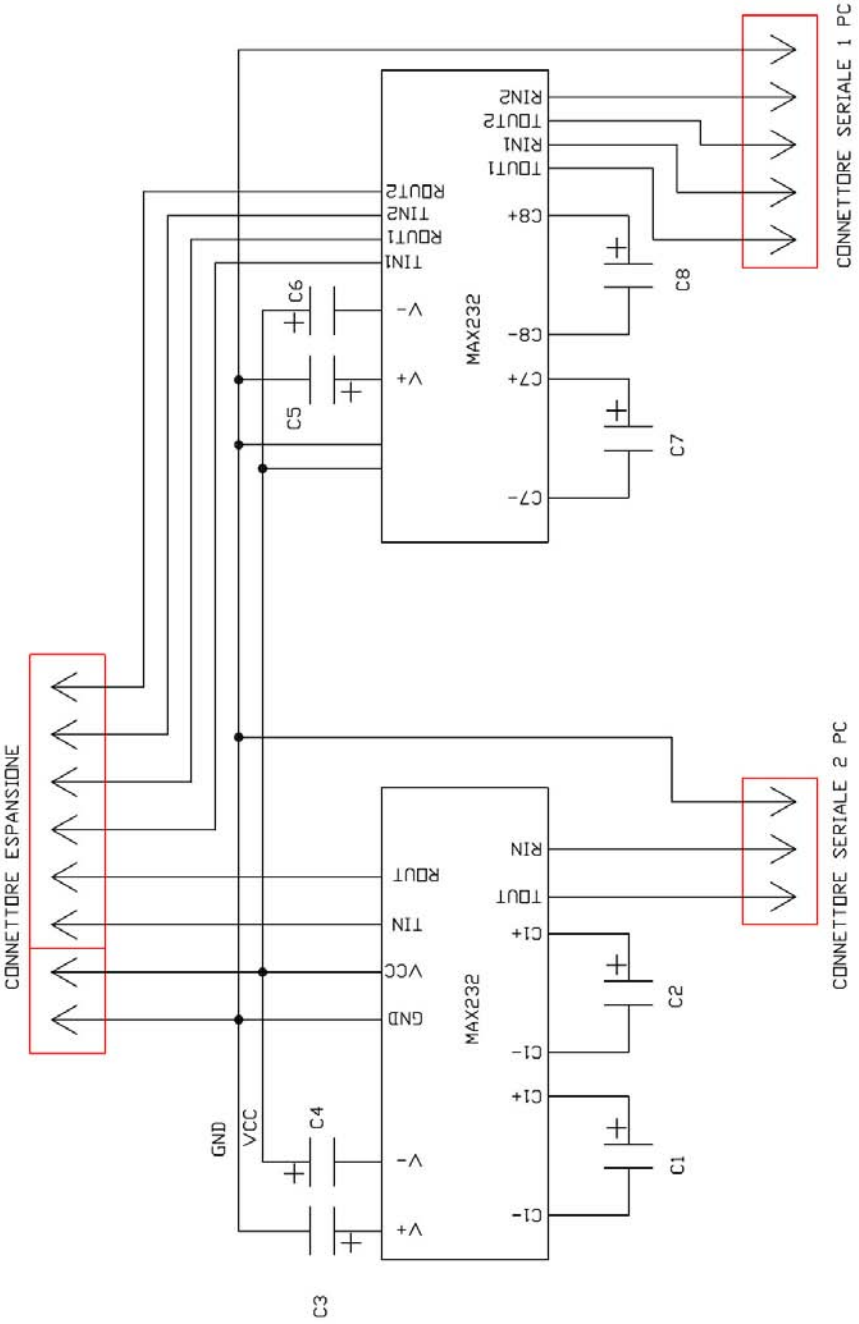
La scheda di interfaccia pc MAX232 converte i segnali TTL in segnali adeguati all'interfacciamento con la porta seriale del computer (segnali + e – 12V).



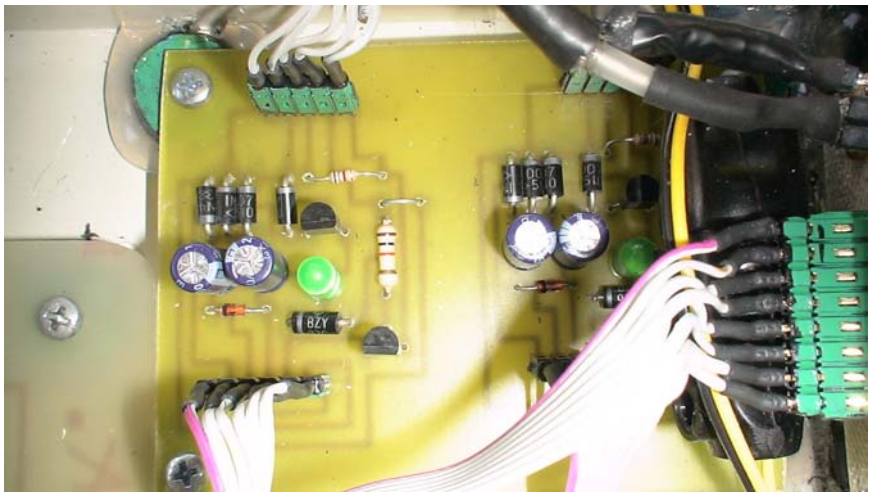
Si possono vedere i due integrati MAX232 ed i relativi condensatori da 1uF che costituiscono la Pompa di Carica/Scarica.
Si può vedere su un lato il connettore in arrivo, il cui cavo flat è connesso alla schedina principale, mentre sull'altro i connettori in uscita, diretti verso le DB9 (porte seriali).



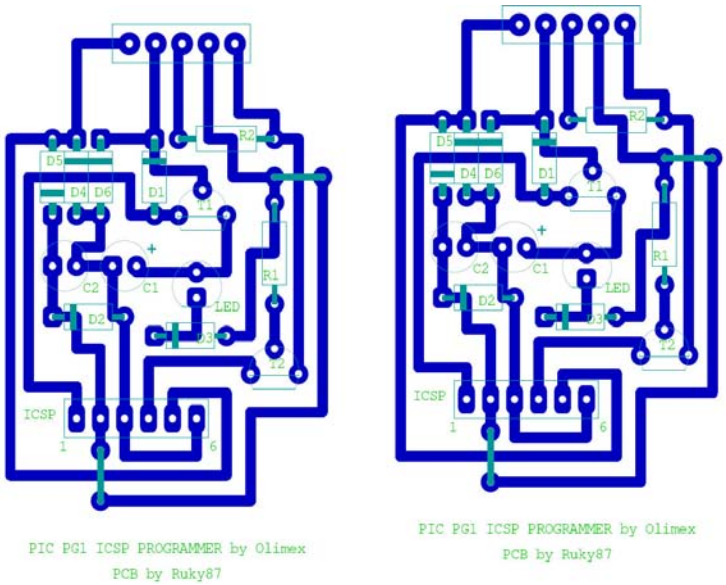


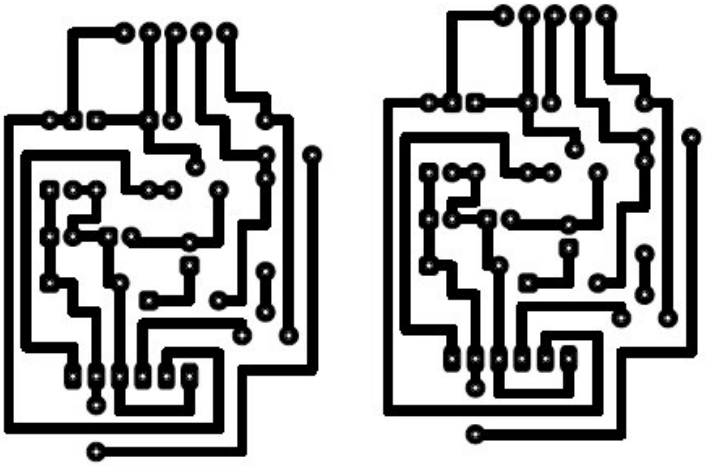

CONNETTORE ESPANSIONE
CONNETTORE SERIALE 1 PC
CONNETTORE SERIALE 2 PC
MAX232
RIN2
TOUT2
RIN1
TOUT1
C8+
C8-
C7+
C7-
ROUT2
TIN2
ROUT1
TIN1
V-
V+
C6
C5
C8
C7
MAX232
RIN
TOUT
C1+
C1-
C1+
C1-
ROUT
TIN
VCC
GND
C2
C1
GND
VCC
C4
C3

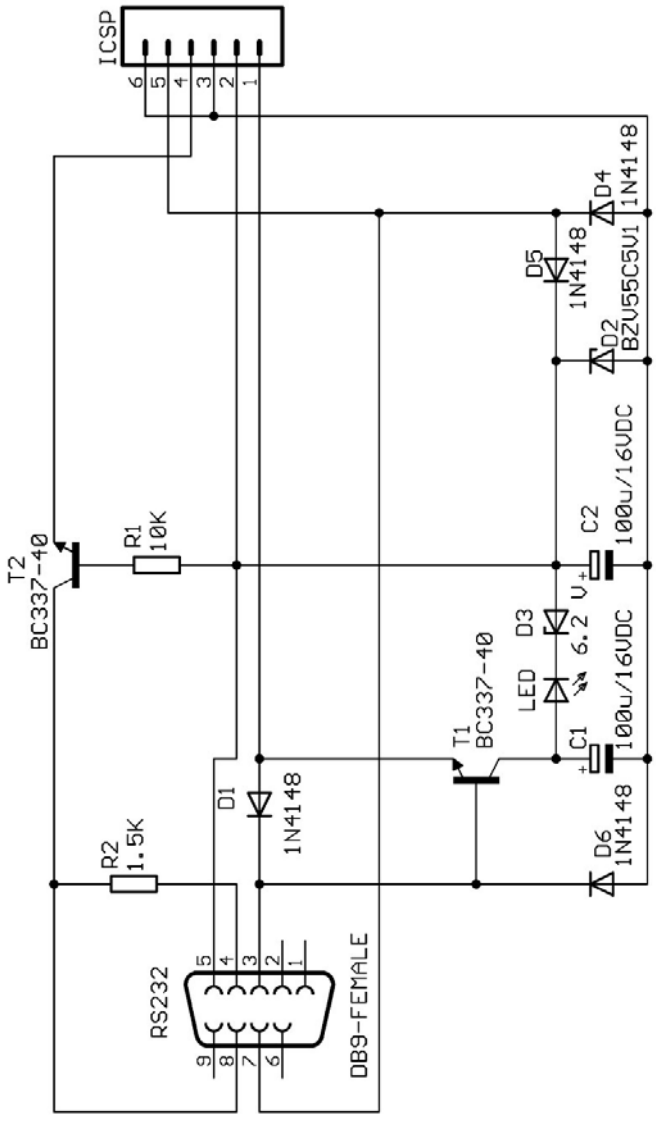
La schedina di programmazione contiene due programmatori JDM, alimentati direttamente dalla seriale del PC. Un programmatore serve per la programmazione del pic principale della base, mentre l'altro è collegato ad un connettore RJ45 (ethernet). Attraverso un cavetto è possibile collegare la base alla gondola e programmare il pic principale del dirigibile.



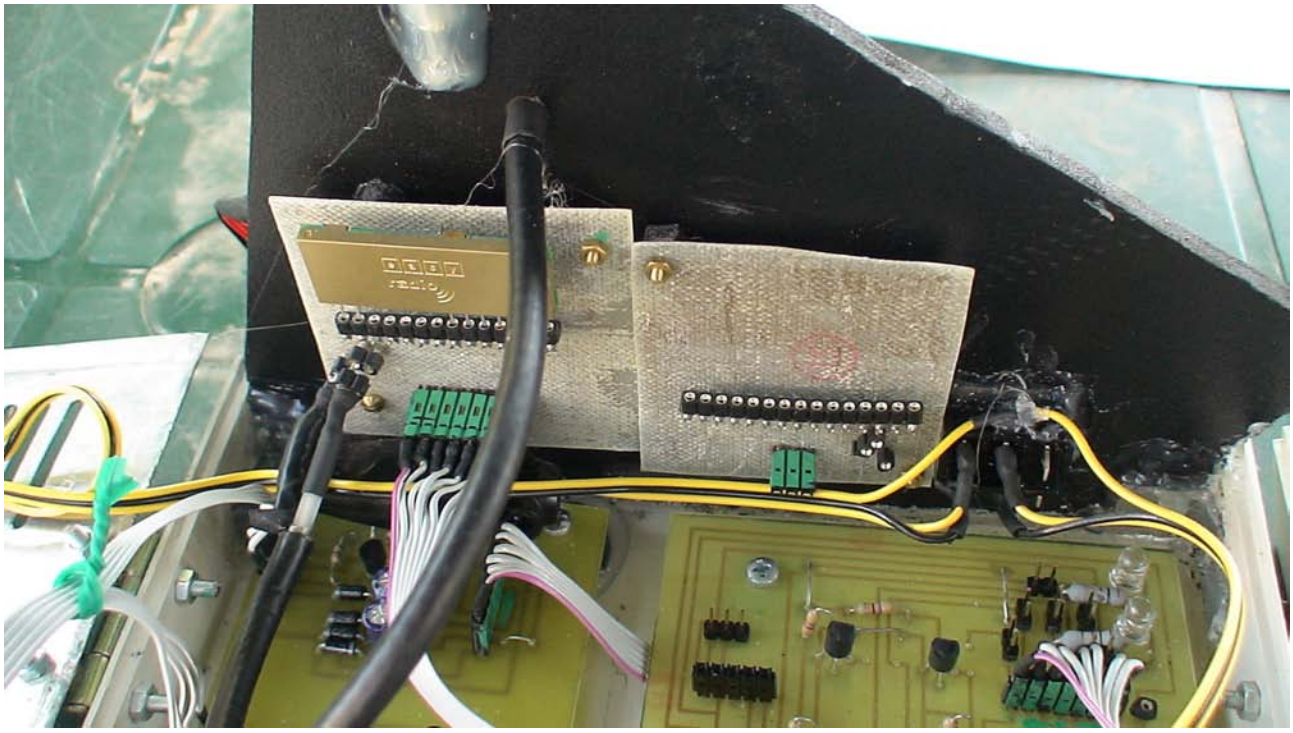
Si notano subito i due programmatori distinti. Entrambi con il rispettivo led di programmazione. In alto si può vedere il connettore che collega, tramite un flat, il programmatore al connettore DB9, mentre in basso un connettore ICSP per la programmazione In-Circuit.
Tale connettore è collegato con il rispettivo sulla schedina principale della base tramite un cavo flat.



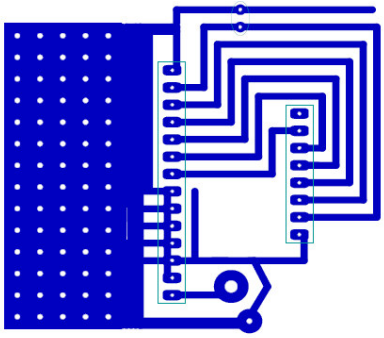


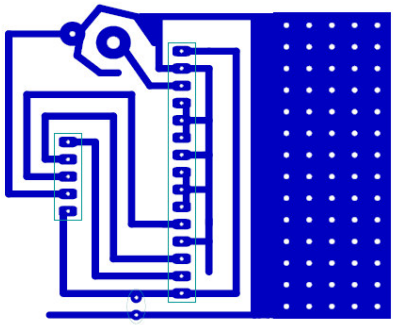

ICSP
6
5
4
3
2
1
D4
1N4148
D5
1N4148
D2
BZV55C5V1
C2
100u/16VDC
V+
6.2
D3
LED
C1
100u/16VDC
D6
1N4148
T1
BC337-40
T2
BC337-40
R1
10K
R2
1.5K
D1
1N4148
RS232
9
8
7
6
5
4
3
2
1
DB9-FEMALE

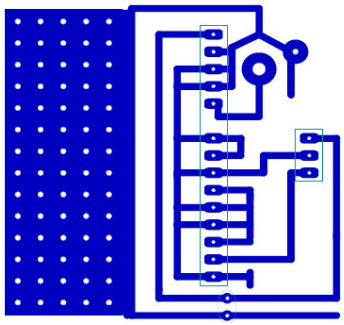
Sia per la base terrestre che per la gondola sono state realizzate delle **schedine di supporto per i moduli radio**.

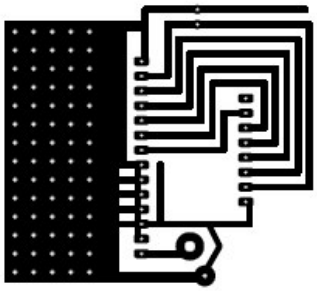


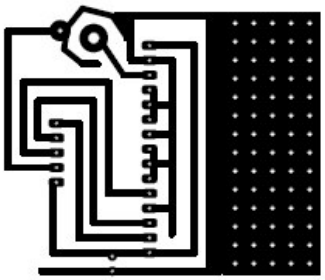
È stata realizzata una coppia di schedine per ogni modulo: l'easy radio, il trasmettitore ibrido ed il ricevitore ibrido.
Ogni coppia è diversa perché diversa è la piedinatura dei moduli.
Tutte le schedine comunque sono state studiate in modo da limitare il più possibile interferenze magnetiche.
In alto si può vedere il modulo radio che si incastra nell'apposito connettore a 90° realizzato con strip Tulipano, adatti ad alte frequenze.
In basso invece il connettore che collega il supporto alla schedina dei moduli radio, tramite l'apposito cavo Flat.

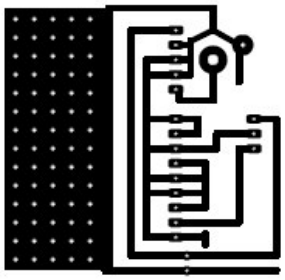






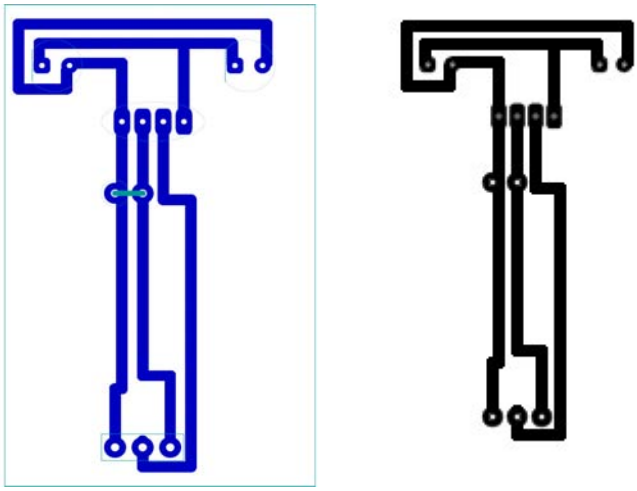


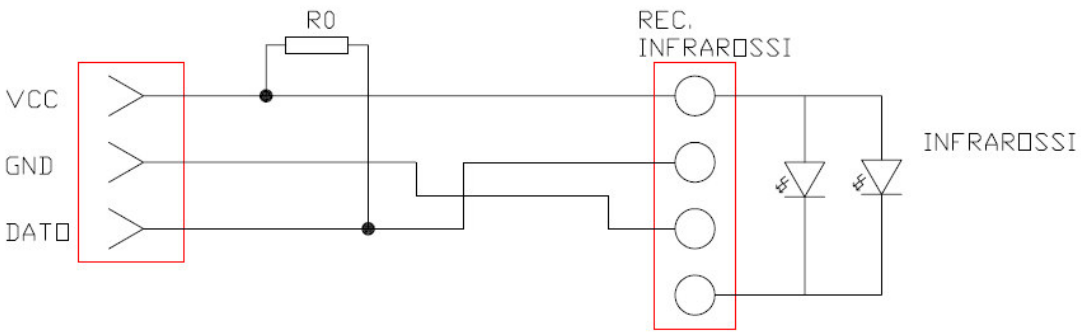




SCHEDINE INFRAROSSI

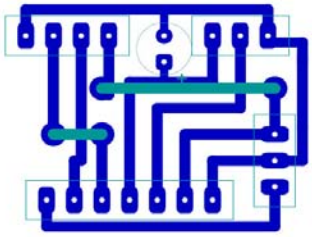
I sensori infrarossi necessitano alcuni componenti esterni: i led emettitori e una resistenza di pullup.
È stata realizzata una schedina contenenteil blocco sensore, che viene agganciata al pallone.

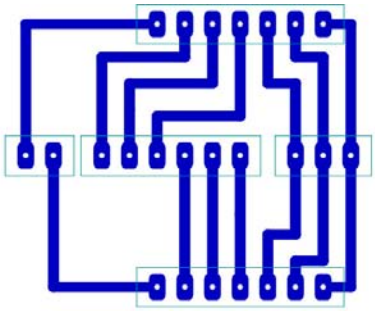


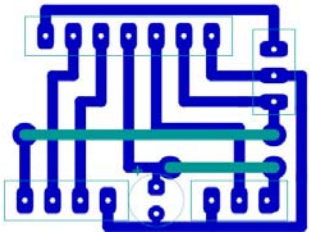


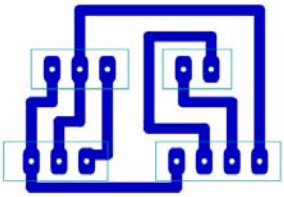
SCHEDINE DERIVAZIONI

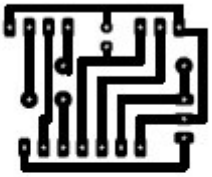
Sono le schedine delle derivazioni della struttura e della gondola.

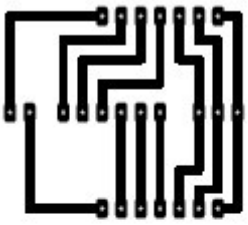


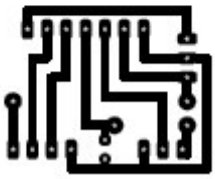






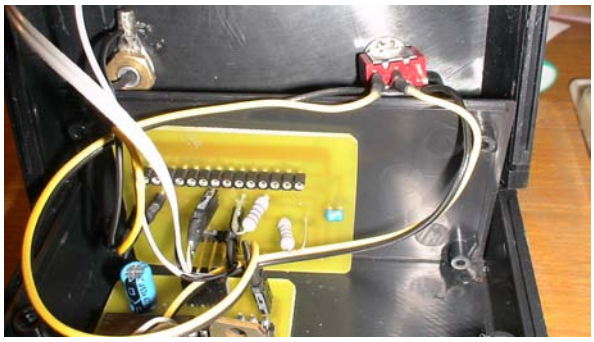


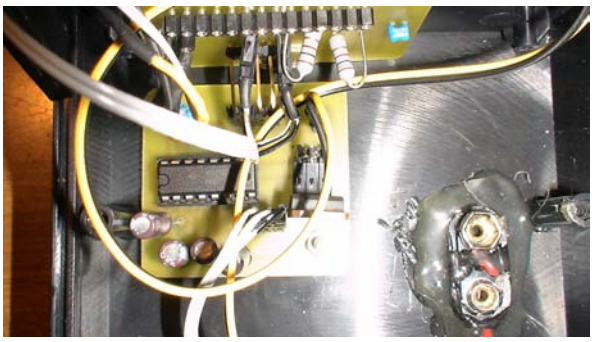


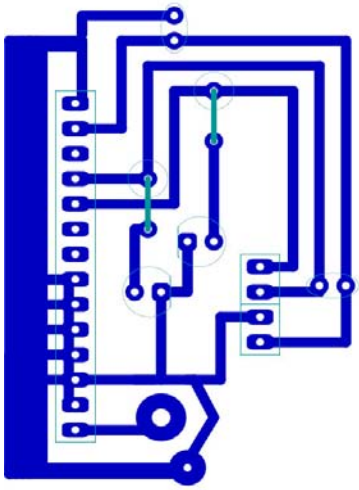


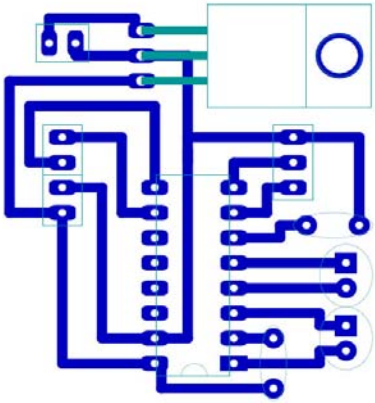
MINIBASE

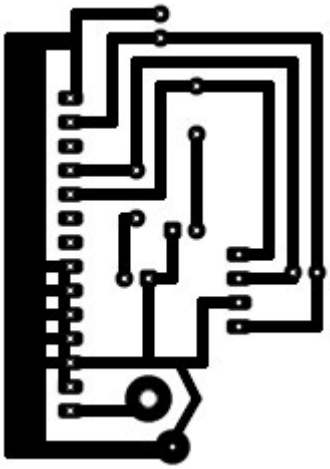
La minibase è realizzata da due circuiti stampati sui quali vi è un max232 e il connettore per il modulo radio. La schedina che ospita il max232 è corredata da una porta seriale per la comunicazione con il pc e da due led di segnalazione (RX-TX). Sulla seconda scheda vi è saldato un connettore, per l'inserimento del modulo, e il cavo schermato da 50 ohm per l'antenna. Per l'accensione e lo spegnimento vi è un interruttore su di un lato della scatola che contiene il tutto. L'alimentazione viene fornita da una pila da 9V acquistabile in qualsiasi supermercato.

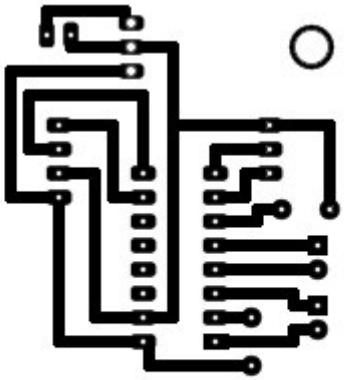






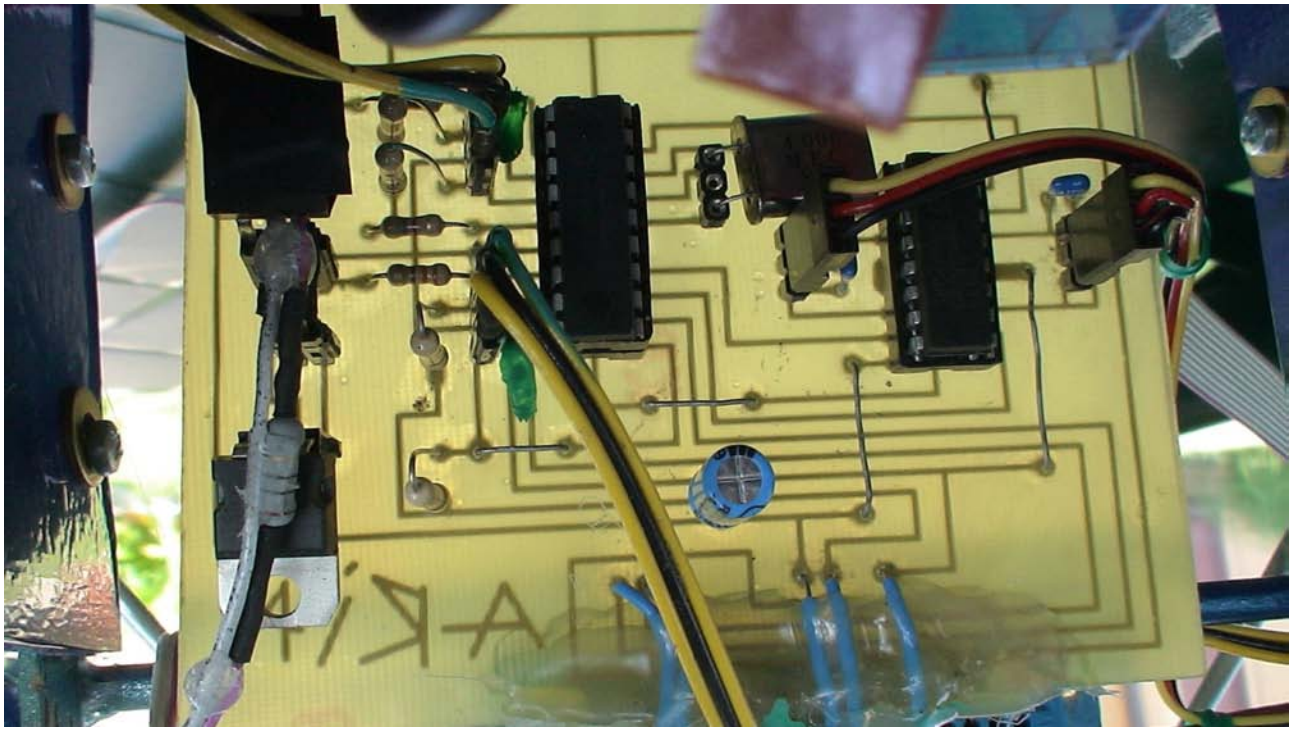






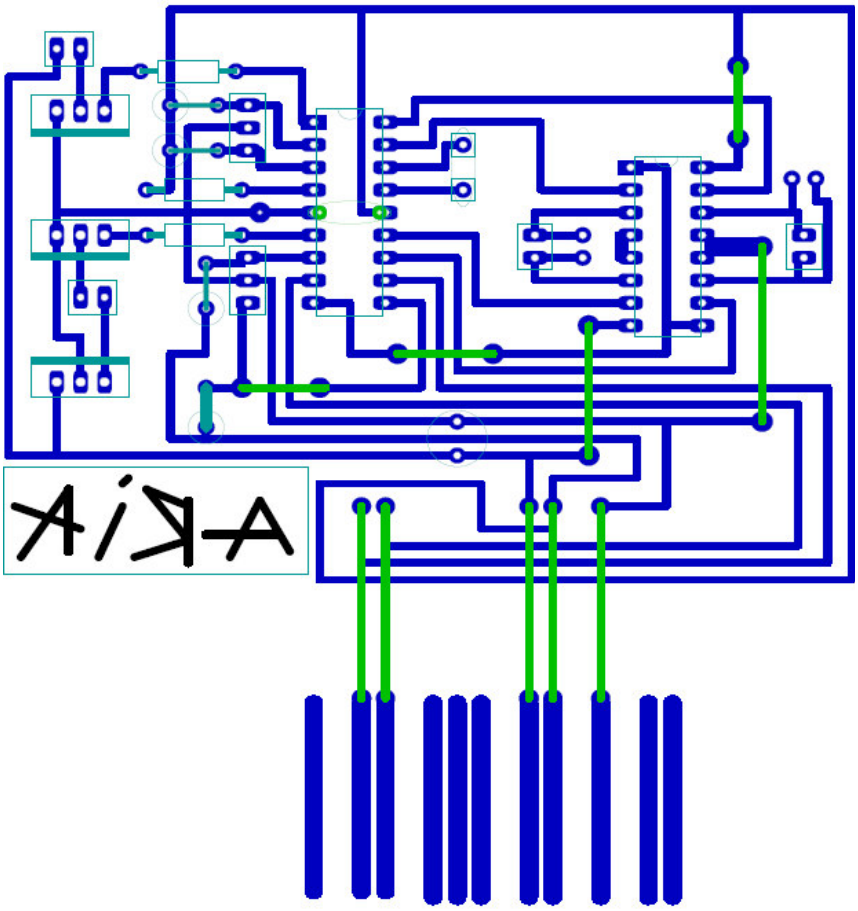
ESPANSIONI

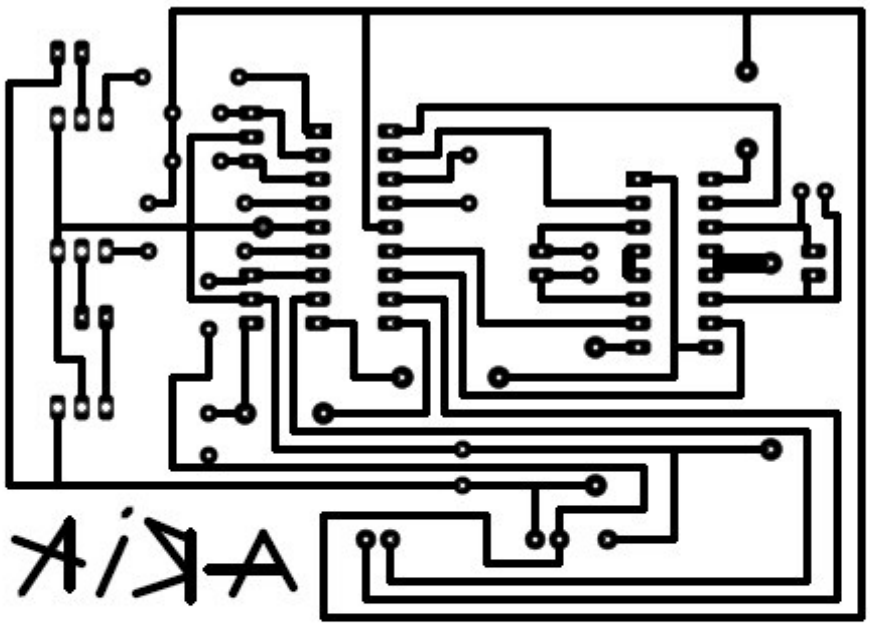
L’unica schedina realizzata per quanto riguarda le espansioni è quella di controllo della telecamera (modulo ciclopè).

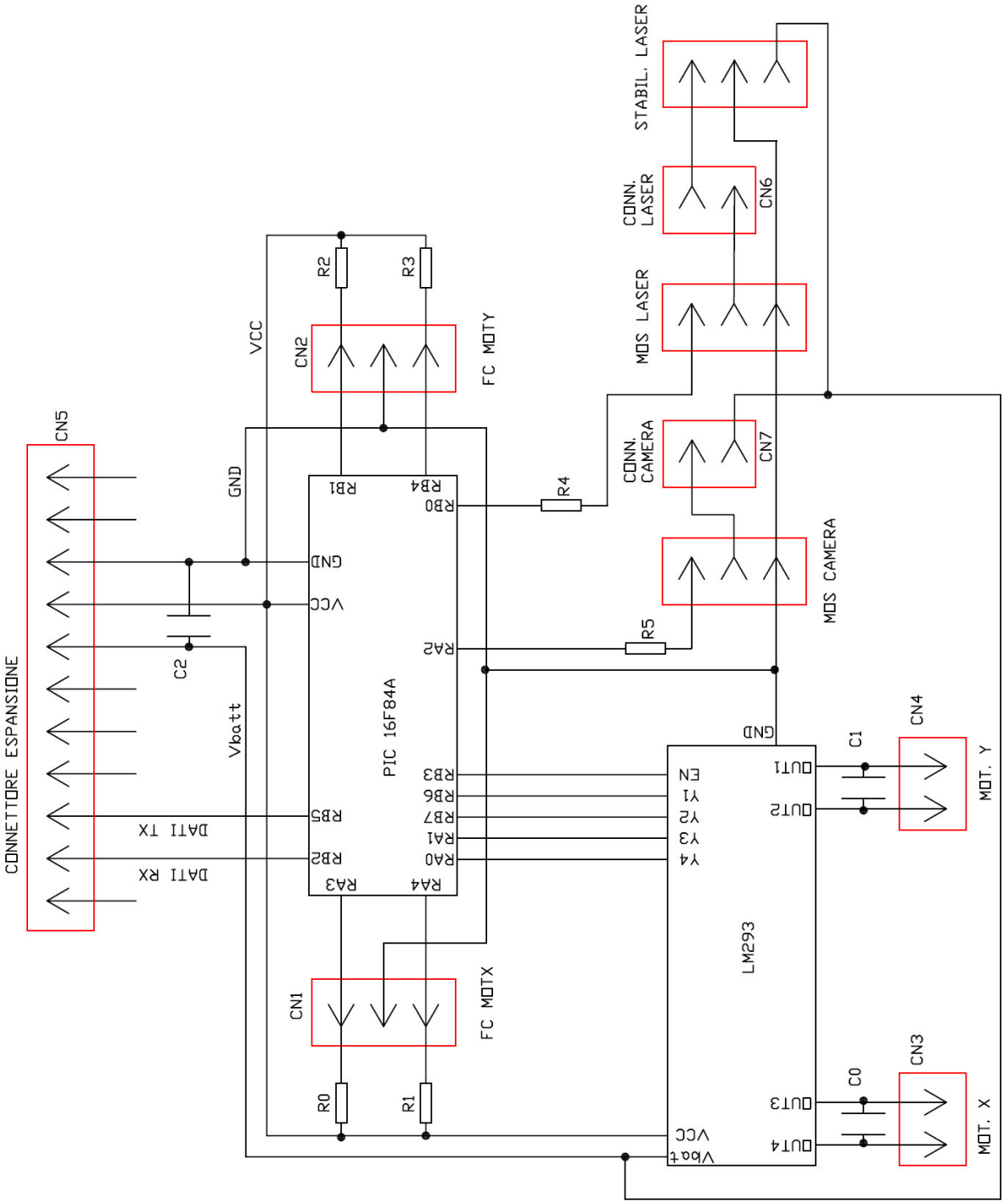


La schedina deve pilotare i due motori della telecamera, quello dell’asse x (al centro dell’asta) e quello dell’asse y (sul lato).
Ogni motore è dotato di due finecorsa, uno per la rotazione oraria l’altro per quella antioraria. Quando i finecorsa vengono premuti i motori devono arrestarsi per impedire il danneggiamento del modulo.
Il pic dovrà quindi leggere anche lo stato dei finecorsa prima di decidere se abilitare il movimento dei motori.
Si possono subito notare sulla schedina i connettori dei finecorsa alla sinistra del pic ed il chip di controllo motori con relativi connettori sulla destra.
Sulla schedina vi sono due Mos ed Uno stabilizzatore. I mos servono per comandare l’accensione e lo spegnimento della Telecamera e del Puntatore Laser. Lo stabilizzatore serve per alimentare il puntatore laser, che può anche essere sostituito da qualsiasi altro dispositivo (per esempio una sirena).

Si vedono subito i connettori per il collegamento dei due dispositivi appena citati.
Il pic gestisce quindi lo stato ON/OFF dei dispositivi, analizza i finecorsa e comanda i due motori. Il pilotaggio dei motori avviene per mezzo dei pin "direzione" dell'integrato di controllo motori. Attraverso essi è possibile mettere il motore in tre stati: fermo, direzione oraria, direzione antioraria. A differenza dei motori per la propulsione del dirigibile, quindi, il comando "motore fermo" è impostato per mezzo dei pin di controllo direzione anziché attraverso l'Enable dell'integrato.
I motori devono muoversi lentamente per dare tempo all'utente di regolare la posizione della telecamera. Questo non è possibile attraverso il pwm perché riducendo la velocità, si riduce anche la coppia.
I motori quindi vengono pilotati per mezzo di brevi impulsi. Gli impulsi determinano un breve spostamento con una coppia elevata (quella di spunto) necessaria per spostare il peso della telecamera. Gli impulsi vengono inviati su entrambi i canali Enable dell'integrato. Ecco perché è stato scelto di gestire la posizione "motore fermo" attraverso i canali Direzione.
In alto sulla schedina è incollato a 90° un connettore realizzato in vetronite (basetta ramata spessore 0,8). Il connettore è costituito da piste stagnate che fanno contatto con le lamelle del connettore dell'espansione (quello blu) quando viene inserito nella fessura. Il sistema è simile agli slot di espansione PCI di un PC. Il connettore oltre a stabilire il collegamento elettrico tra la scheda del modulo e la scheda principale garantisce anche il supporto ed il fissaggio della stessa. L'inserimento ad incastro rende massima l'estraibilità ed evita quindi una struttura aggiuntiva per il sostegno.
Il modulo della telecamera riceve la tensione logica (5V) e quella delle batterie per la telecamera ed i motori.
La comunicazione con il pic principale del dirigibile avviene attraverso due canali: il primo indica (livello alto) che il modulo è pronto a ricevere istruzioni, il secondo è quello attraverso il quale passano le istruzioni di controllo, secondo protocollo seriale asincrono standard a 2400 baud.





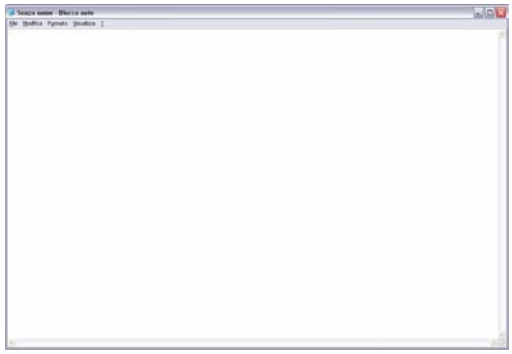

CONNETTORE ESPANSIONE
CN5
DATI RX
DATI TX
Vbatt
C2
GND
VCC
PIC 16F84A
RB1
RB4
RB0
RA2
RB3
RB6
RB7
RA1
RA0
RA3
RA4
RB5
RB2
R2
R3
CN2
FC MOTY
R4
R5
CN1
FC MOTX
R0
R1
STABIL. LASER
CONN. LASER
CN6
MOS LASER
CONN. CAMERA
CN7
MOS CAMERA
LM293
EN
Y1
Y2
Y3
Y4
OUT1
OUT2
OUT3
OUT4
Vbatt
VCC
GND
C1
C0
CN4
CN3
MOT. Y
MOT. X

SOFTWARE

Il software si può suddividere in 3 categorie: Software usato sul PC durante la fase di progettazione, Programmi realizzati per i PIC, Programmi realizzati per il PC.

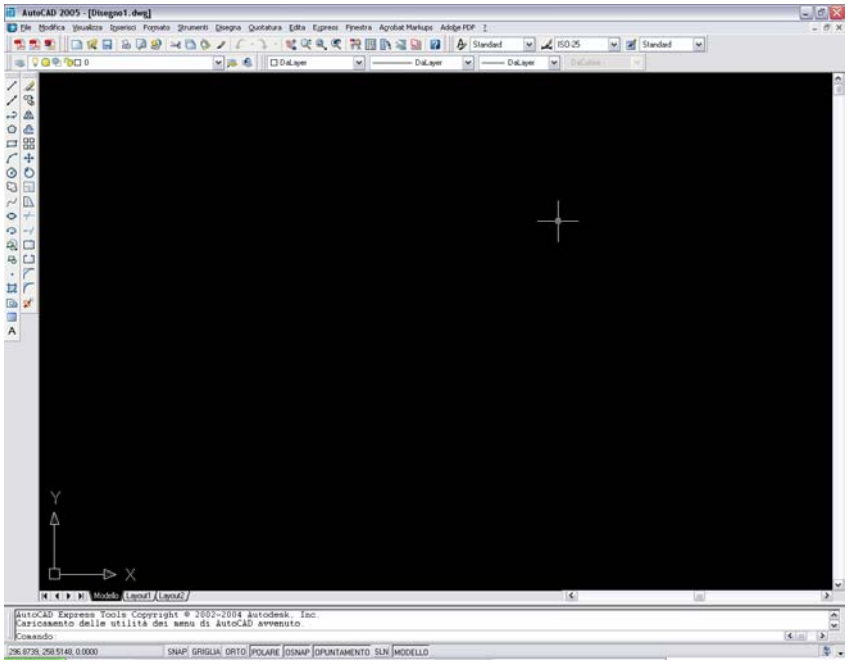
PROGRAMMI USATI

Nel corso della sperimentazione e della progettazione del dirigibile abbiamo fatto uso di numerosi software a cominciare dai semplici editor di testo come "**Blocco Note**".

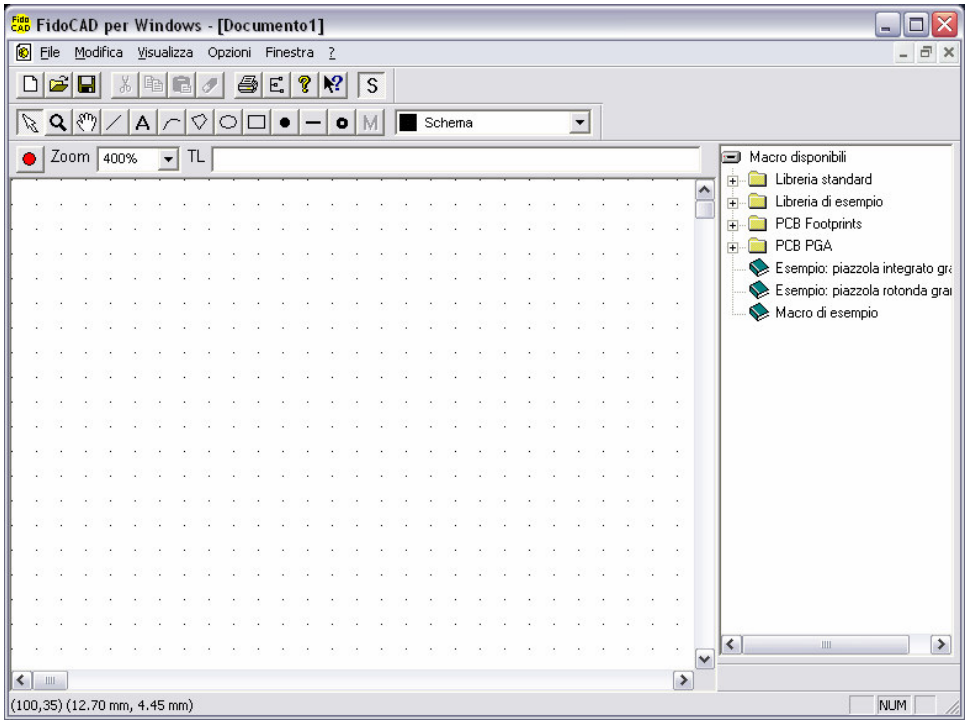


Abbiamo fatto largo uso di programmi di cad:

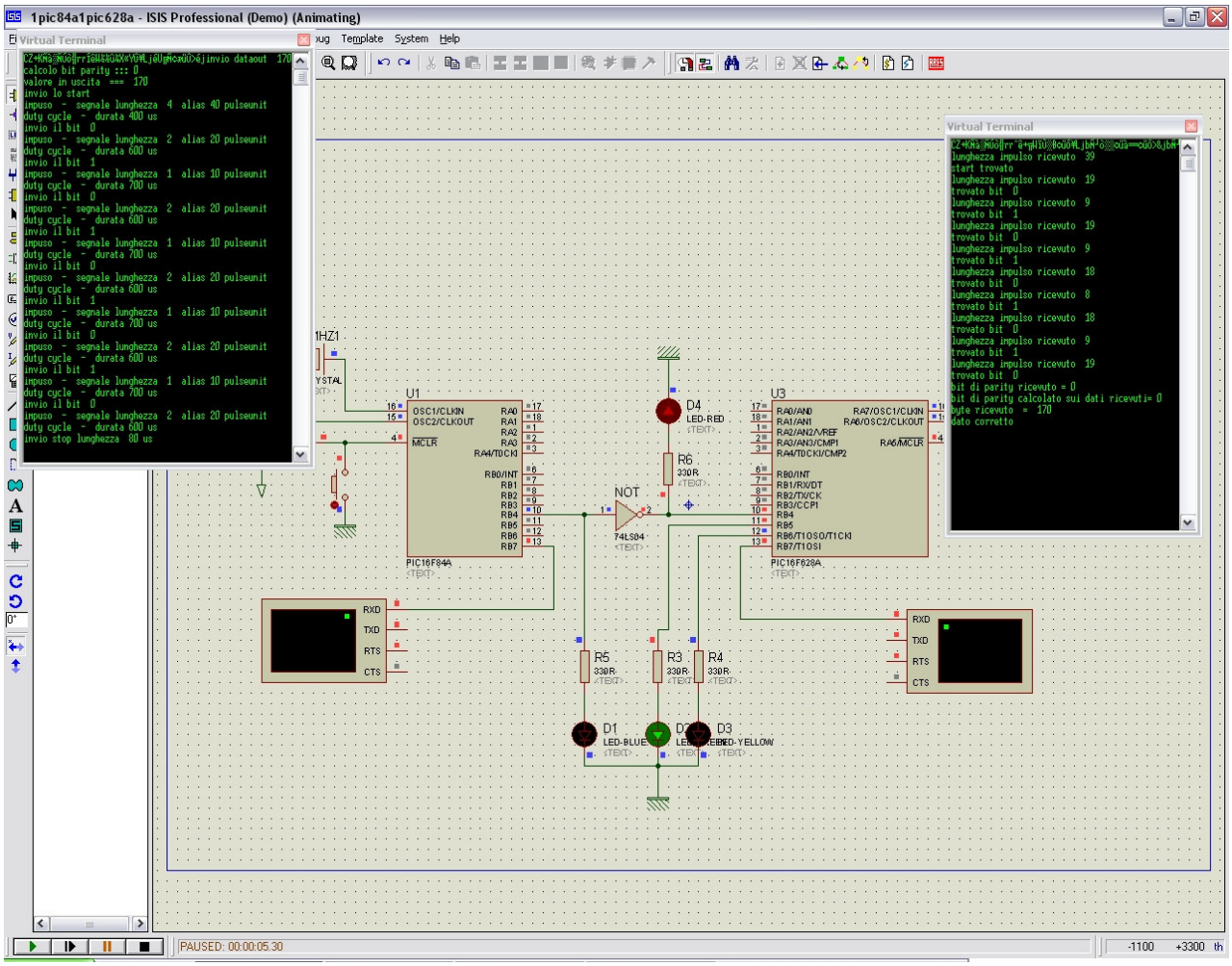
- **Autocad 2005** per la realizzazione di schemi elettrici e schemi meccanici



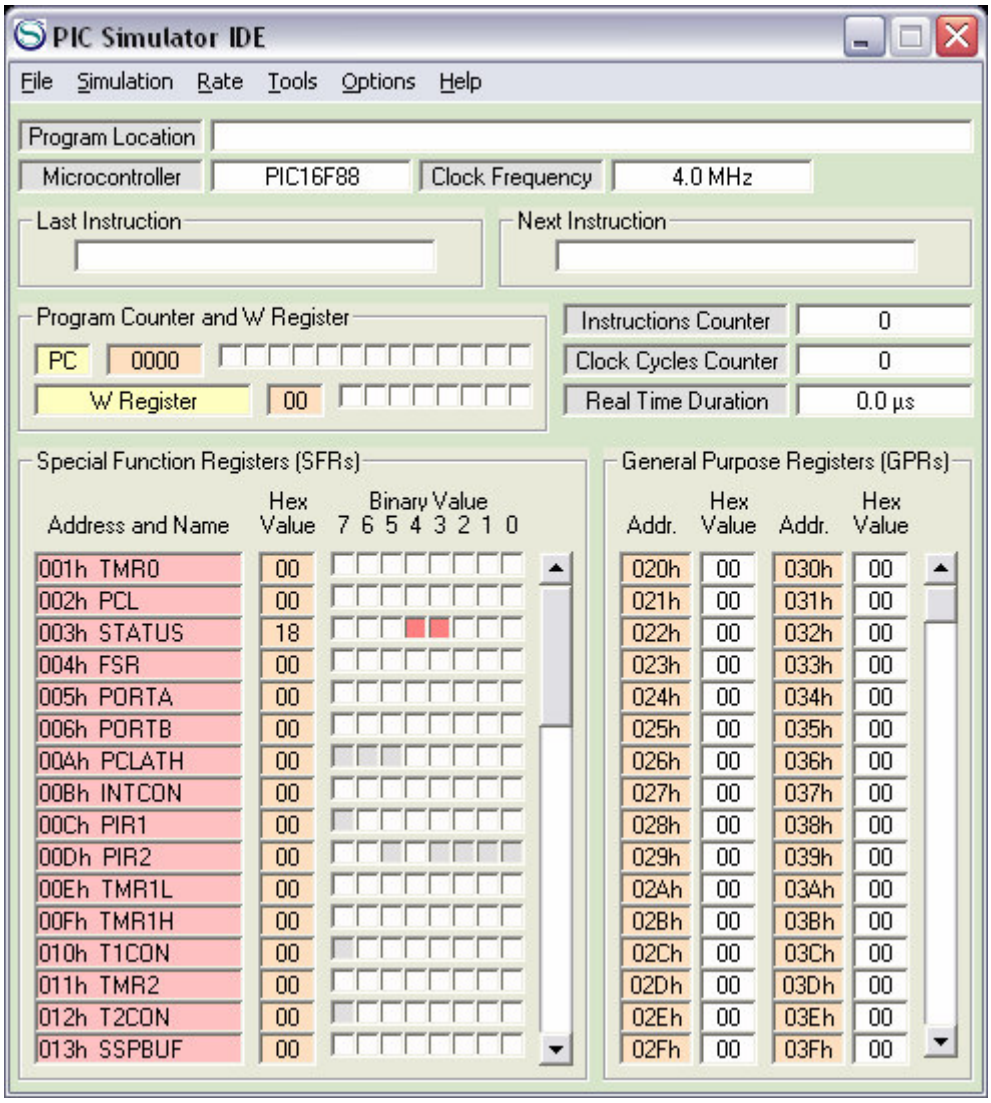
- **Fidocad** (Freeware) per la realizzazione dei PCB ed altro



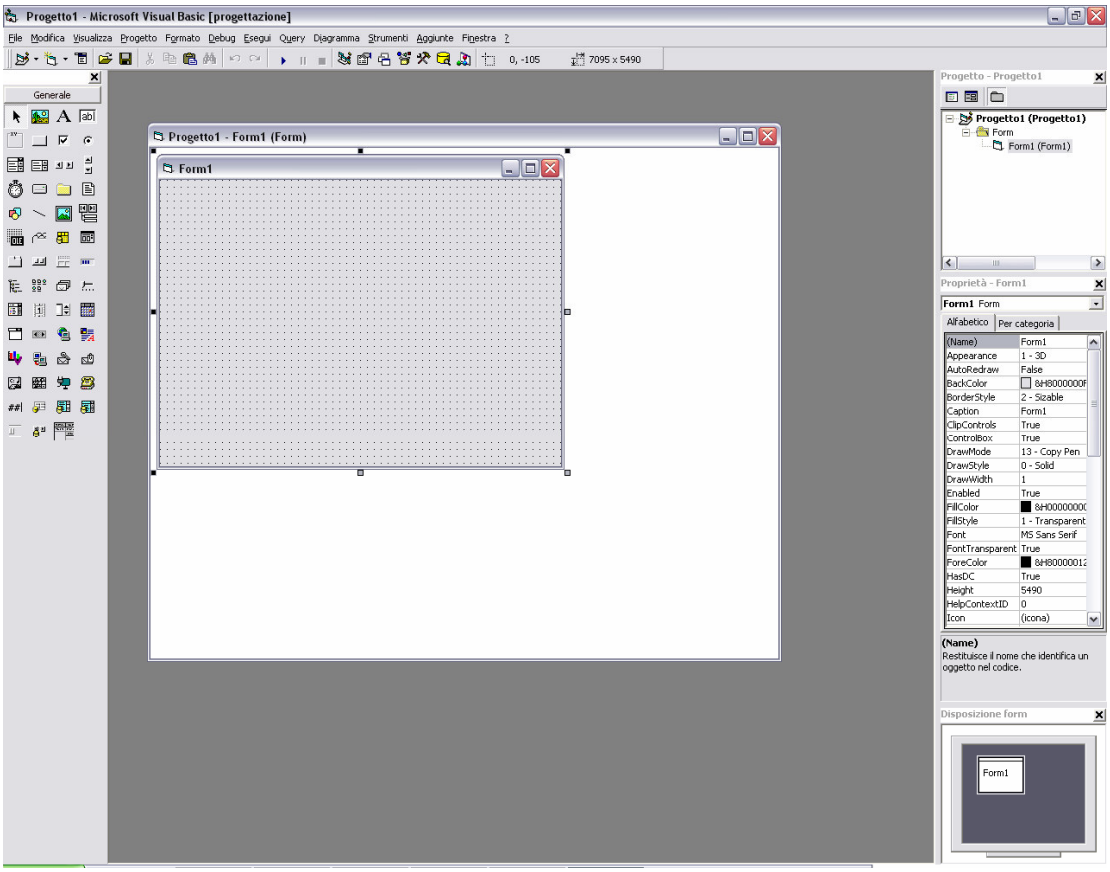
Per l'elaborazione di immagini abbiamo fatto uso di programmi di grafica e come editor di testo abbiamo usato Il pacchetto **Office 2003** di microsoft e la suite di **OpenOffice**, gratuita e potente.
Per simulare alcuni circuiti elettrici contenenti anche PIC abbiamo usato un programma chiamato "**PROTON**". Tale programma integra un potente simulatore di circuiti elettronici (**ISIS**), che lavora in real-time e supporta anche i microcontrollori PIC. Viene realizzato lo schema al pc e caricato nel pic virtuale il programma. Sul monitor si può osservare la simulazione a tempo reale di quello che accadrebbe sul circuito reale.



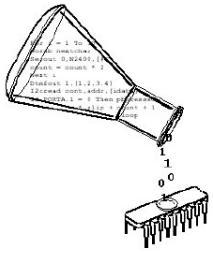
Un altro simulatore usato è stato **Pic Simulator Ide**. Con questo programma è possibile realizzare programmi sia in assembly che in un semplice linguaggio Basic per alcuni modelli PIC. Il programma simula anche l'esecuzione del programma, mostrando lo stato dei registri interni del pic. La simulazione non è real-time e viene eseguita solo sul microcontrollore senza possibilità quindi di studiarne il funzionamento all'interno di un circuito.



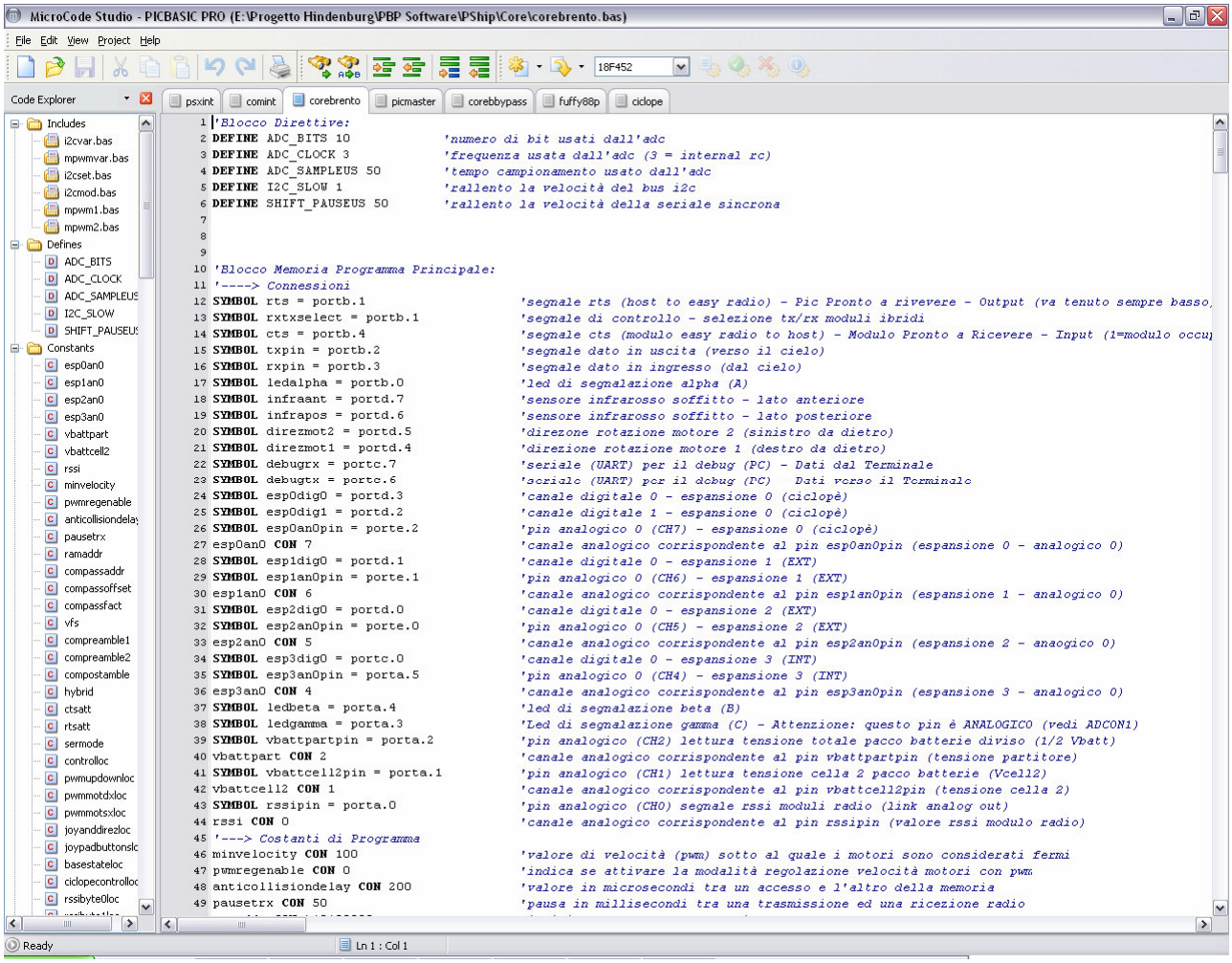
Per la realizzazione dei programmi per il PC inerenti al progetto e dell'interfaccia GUI della Base è stato usato il **Visual Basic**, linguaggio di programmazione a oggetti di casa Microsoft, molto versatile e di semplice utilizzo per realizzare applicazioni basate su interfacce grafiche.



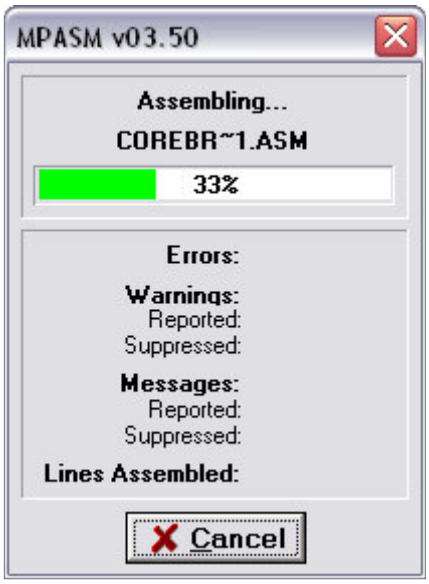
Per la realizzazione dei programmi dei PIC è stato usato un compilatore da Basic ad Assembly molto potente: il **PicBasic**.
Il linguaggio basic adottato è molto potente e ricco di istruzioni "già pronte", come ad esempio i comandi "serin" e "serout" per la gestione di una comunicazione seriali asincrona.
Il compilatore supporta tutti i PIC prodotti dalla Microchip e converte il codice scritto in basic nel linguaggio Assembly, usato nella programmazione a basso livello di questi microcontrollori.
Sarebbe un impresa indice della pazzia tentare di realizzare programmi complessi come quelli da noi ideati puramente in assemly.



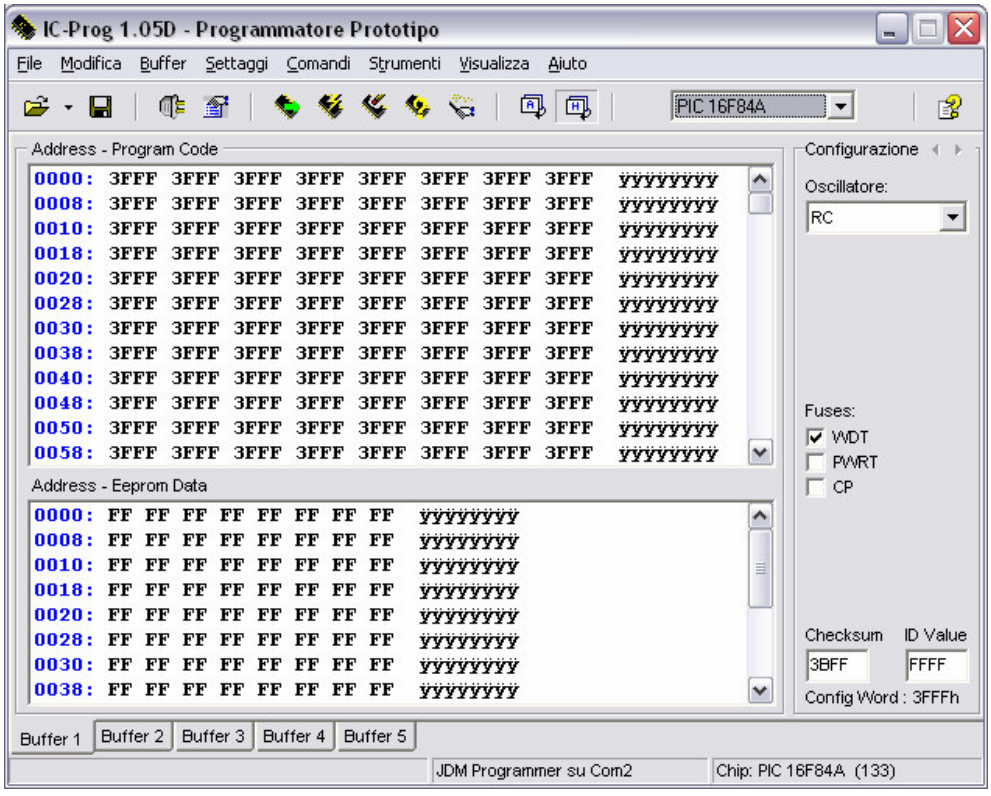
Il compilatore si appoggia ad un'interfaccia grafica, **MicroCode Studio**, che è usata anche come editor di testo per il codice che si va a scrivere. Attraverso la pressione di un tasto è possibile compilare il tutto.

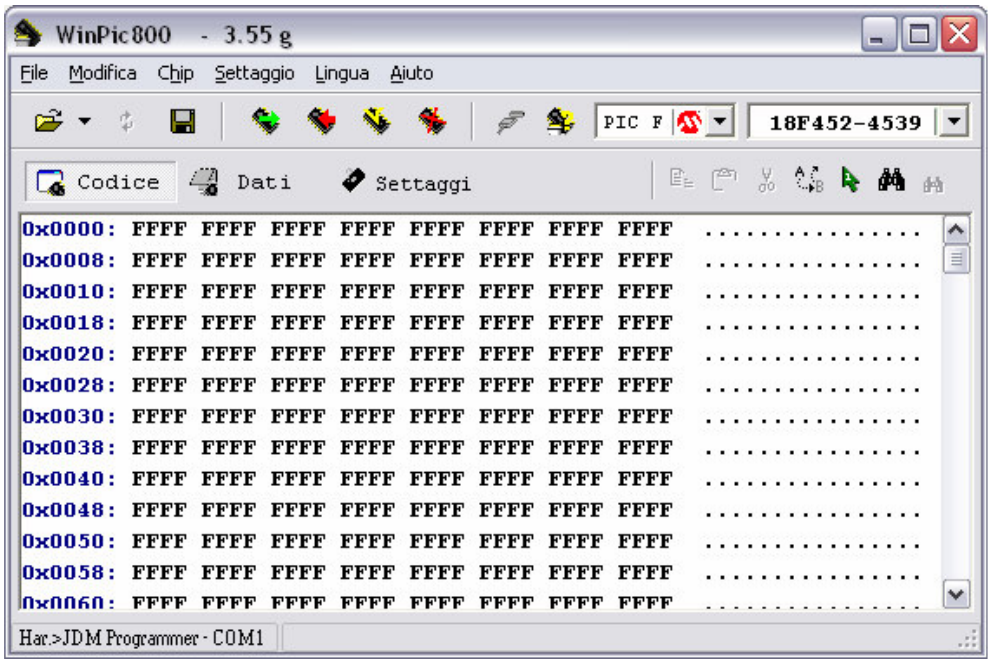


Il file contenente il codice assembly ".asm" generato dal compilatore deve essere convertito in un file binario o esadecimale contenente le istruzioni in linguaggio macchina del pic. Tale file andrà poi caricato nella memoria del dispositivo.
Per effettuare questa conversione si deve dare in pasto il file asm ad un programma detto assemblatore. Viene usato l'**MPASM** della Microchip, che supporta tutti i pic prodotti. Il risultato dell'operazione è un file ".hex". MicroCode Studio può essere configurato per richiamare l'MPASM e processare il file appena compilato in modo automatico.

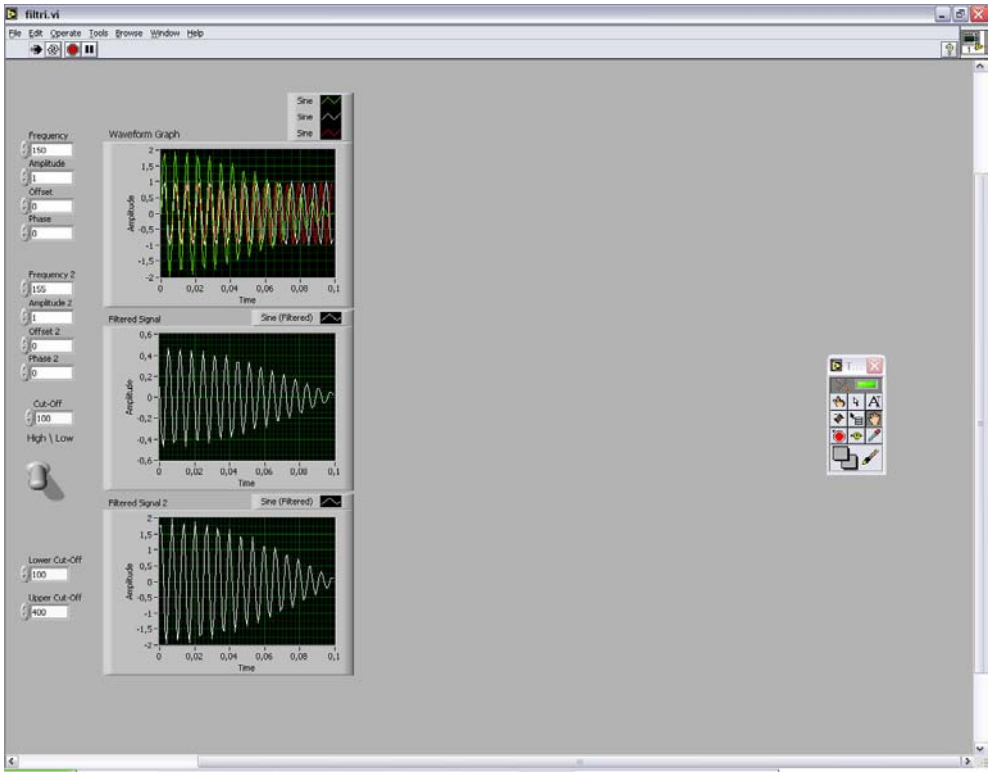


Per caricare il programma all'interno della memoria del pic è stato usato un programmatore di tipo JDM. Software compatibili con esso sono l'ormai famoso e conosciuto **IcProg**, che supporta una buona parte di PIC e può programmare anche le memorie, e **WinPic800**, che supporta tutti i PIC della Microchip e li programma molto velocemente.
Quest'ultimo programma è ancora in via di sviluppo, e potrebbe non essere compatibile con alcuni PC.
Quello che abbiamo usato maggiormente è proprio WinPic800





MicroCode Studio può essere configurato per aprire in modo automatico il WinPic800 al termine dell'assemblamento.
Questi elencati sono i programmi più importanti, usati nel nostro progetto. Sono comunque stati usati numerosi altri applicativi per operazioni di secondaria importanza. L'anno scorso è stato fatto uso anche di **LabView**, come strumento di acquisizione segnali analogici (lo abbiamo usato come oscilloscopio).



PICBASIC

Il compilatore PicBasic è prodotto dalla MicroEngineering Labs e rende semplice la realizzazione di programmi complessi per i microcontrollori PicMicro.
Il linguaggio Basic usato è compatibile con il "PBasic" della Parallax, uato per i microcontrollori "Basic Stamp".
Il compilatore è dotato di diverse librerie contenenti le subroutines base in assembly, che vengono opportunamente chiamate nel sorgente generato a seconda delle istruzioni basic di cui si è fatto uso.

```
' Example program to blink an LED connected to PORTB.0
about once a second

loop: High PORTB.0       ' Turn on LED
      Pause 500          ' Delay for .5 seconds

      Low PORTB.0        ' Turn off LED
      Pause 500          ' Delay for .5 seconds

      Goto loop          ' Go back to loop and blink
                        LED forever
      End
```

Noi dobbiamo occuparci di scrivere il programma in Basic, la creazione del file ".asm" attraverso le librerie è un compito che il compilatore svolge per conto suo.
Il linguaggio basic è abbastanza complesso: si possono creare diversi tipi di variabili (Bit, Byte, Word) ed agire su di esse: PicBasic mette a disposizione diversi operatori matematici per eseguire calcoli complessi.
È possibile lavorare su matrici di variabili ed utilizzare cicli come il "FOR…NEXT". Attraverso strutture come il "CASE…SELECT" e "IF…THEN…ELSE" è possibile optare scelte e discriminare le operazioni da far eseguire al programma. Questo permette di creare procedimenti logici basati sul test di una o più condizioni.
Il linguaggio è dotato di istruzioni che semplificano l'uso di alcuni moduli interni al pic, come ad esempio l'USART, il PWM e gli ADC.
Ma non è tutto. PicBasic integra anche routines software per la comunicazione seriale, per l'I2C, per la generazione di impulsi e frequenze….

@	Insert one line of assembly language code.
ADCIN	Read on-chip analog to digital converter.
ASM..ENDASM	Insert assembly language code section.
BRANCH	Computed GOTO (equiv. to ON..GOTO).
BRANCHL	BRANCH out of page (long BRANCH).
BUTTON	Debounce and auto-repeat input on specified pin.
CALL	Call assembly language subroutine.
CLEAR	Zero all variables.
CLEARWDT	Clear (tickle) Watchdog Timer.
COUNT	Count number of pulses on a pin.
DATA	Define initial contents of on-chip EEPROM.
DEBUG	Asynchronous serial output to fixed pin and baud.
DEBUGIN	Asynchronous serial input from fixed pin and baud.
DISABLE	Disable ON DEBUG and ON INTERRUPT processing.
DISABLE DEBUG	Disable ON DEBUG processing.
DISABLE INTERRUPT	Disable ON INTERRUPT processing.
DTMFOUT	Produce touch-tone frequencies on a pin.
EEPROM	Define initial contents of on-chip EEPROM.
ENABLE	Enable ON DEBUG and ON INTERRUPT processing.
ENABLE DEBUG	Enable ON DEBUG processing.
ENABLE INTERRUPT	Enable ON INTERRUPT processing.
END	Stop program execution and enter low power mode.
ERASECODE	Erase block of code memory.
FOR..NEXT	Repeatedly execute statements in a loop.
FREQOUT	Produce 1 or 2 frequencies on a pin.
GOSUB	Call BASIC subroutine at specified label.
GOTO	Continue execution at specified label.
HIGH	Make pin output high.
HPWM	Output hardware pulse width modulated pulse train.
HSERIN	Hardware asynchronous serial input.
HSERIN2	Hardware asynchronous serial input, second port.
HSEROUT	Hardware asynchronous serial output.
HSEROUT2	Hardware asynchronous serial output, second port.
I2CREAD	Read from I²C device.
I2CWRITE	Write to I²C device.
IF..THEN..ELSE..ENDIF	Conditionally execute statements.
INPUT	Make pin an input.
LCDIN	Read from LCD RAM.
LCDOUT	Display characters on LCD.
{LET}	Assign result of an expression to a variable.
LOOKDOWN	Search constant table for value.
LOOKDOWN2	Search constant / variable table for value.
LOOKUP	Fetch constant value from table.
LOOKUP2	Fetch constant / variable value from table.
LOW	Make pin output low.
NAP	Power down processor for short period of time.
ON DEBUG	Execute BASIC debug monitor.
ON INTERRUPT	Execute BASIC subroutine on an interrupt.
OWIN	One-wire input.
OWOUT	One-wire output.
OUTPUT	Make pin an output.
PAUSE	Delay (1 millisecond resolution).
PAUSEUS	Delay (1 microsecond resolution).
PEEK	Read byte from register.
PEEKCODE	Read byte from code space.
POKE	Write byte to register.
POKECODE	Write byte to code space at device programming time.
POT	Read potentiometer on specified pin.
PULSIN	Measure pulse width on a pin.
PULSOUT	Generate pulse on a pin.
PWM	Output pulse width modulated pulse train to pin.
RANDOM	Generate pseudo-random number.
RCTIME	Measure pulse width on a pin.
READ	Read byte from on-chip EEPROM.
READCODE	Read word from code memory.
REPEAT..UNTIL	Execute statements until condition is true.
RESUME	Continue execution after interrupt handling.
RETURN	Continue at statement following last GOSUB.
REVERSE	Make output pin an input or an input pin an output.
SELECT CASE	Compare a variable with different values.
SERIN	Asynchronous serial input (BS1 style).
SERIN2	Asynchronous serial input (BS2 style).
SEROUT	Asynchronous serial output (BS1 style).
SEROUT2	Asynchronous serial output (BS2 style).
SHIFTIN	Synchronous serial input.
SHIFTOUT	Synchronous serial output.
SLEEP	Power down processor for a period of time.
SOUND	Generate tone or white-noise on specified pin.
STOP	Stop program execution.
SWAP	Exchange the values of two variables.
TOGGLE	Make pin output and toggle state.
USBIN	USB input.
USBINIT	Initialize USB.
USBOUT	USB output.
WHILE..WEND	Execute statements while condition is true.
WRITE	Write byte to on-chip EEPROM.
WRITECODE	Write word to code memory.
XIN	X-10 input.
XOUT	X-10 output.

Per la lista completa delle potenzialità e dei comandi PicBasic e della sintassi del linguaggio consultare il manuale del compilatore.
Molti programmi da noi realizzati fanno uso di istruzioni “particolari” incluse nel pacchetto, per esempio la comunicazione I2C si basa sulle istruzione già fornite dal linguaggio su cui è stata costruita una struttura decisionale più avanzata (gestione del canale BUSY).

PROGRAMMI PIC

STRUTTURA A MODULI:

I programmi che abbiamo realizzato sono molto complessi, nonostante il semplice linguaggio di programmazione.
I pic infatti si trovano a gestire un numero elevato di periferiche ed ogni periferica richiede diverse righe di codice per essere controllata.
Per questo motivo abbiamo deciso dall'inizio di suddividere il software in tanti "Moduli" diversificati per i loro compiti.
Nel programma principale, che deve gestire tutto l'hardware connesso viene fatto riferimento a tali moduli: La struttura diventa simile ad un programma in C, utilizzando l'istruzione "Include" è possibile integrare altri file all'interno del sorgente principale.
Con questo metodo si risparmia spazio nel listato consentendo una decifrabilità ottimale del codice.
Un sistema così strutturato diventa più comprensibile: provate ad immaginare un unico programma, già complicato e lungo di suo, contenente anche tutte le routines di comunicazione: questo creerebbe confusione perché si perderebbe di vista la struttura logica di funzionamento principale.
Il sistema "Modulare" è molto semplice da utilizzare ed è stato concepito per funzionare per usi generali, quindi non è progettato esclusivamente per il dirigibile.
I vari moduli si dividono i 3 file, da includere nel programma principale:

1. *Il file delle Variabili* di Modulo deve essere incluso subito dopo la dichiarazione delle variabili e delle costanti del programma principale (o "Programma Primo" secondo l'ordine gerarchico). Questo file contiene le variabili usate dai sottoprogrammi del modulo e le impostazioni dei pin del microcontrollore assegnati alle routines del modulo. È questo che permette di rendere il programma universale.
2. *Il file dei "Settaggi"* deve essere incluso subito dopo la sub "Main" del programma principale (la sub di partenza). Esso contiene le istruzioni per configurare il modulo con impostazioni standard, che possono essere modificate agendo sul codice di questo file.

Le impostazioni caricate possono essere modificate anche in runtime perché contenute all'interno di variabili. Alcuni moduli fanno uso di "pin variabili", cioè impostabili e modificabili in runtime. In quest'ultimo caso le impostazioni dei pin vengono fatte in questo file anziché nel precedente.
Le istruzioni di configurazione si trovano all'interno di una sub, per cui tale sub va richiamata all'inizio del programma primo, quando si intende effettuare la configurazione (di solito nella "Main").

3. *Il file delle "Routines"* o file "Del Modulo" contiene le subroutines di funzionamento del modulo, cioè gli insiemi di istruzioni che svolgono il compito per cui il modulo è progettato. Il file di solito è composto da molte routines, collegate ad una principale che gestisce il funzionamento di tutte. È proprio questa che deve essere chiamata dal Programma Primo ogni volta che si richiedono le funzionalità descritte dal modulo. La chiamata può essere fatta ovunque nel programma.
Talvolta il modulo per funzionare richiede la presenza di particolari etichette all'interno del programma principale. Ciò è dovuto al fatto che alcuni moduli fanno uso di diversi metodi per la segnalazione degli errori, tra cui il salto ad una sub generale di errore. Anche se non usata tale funzione, l'etichetta deve essere comunque definita.

È da ricordare che ogni modulo è diverso dall'altro: si consiglia di leggere la guida di ognuno prima di mettersi al lavoro. In seguito verranno proposti i singoli moduli realizzati per il progetto dirigibile ARIA, corredati di codice sorgente.

Modulo controllo Motori con PWM:

Questo modulo permette di controllare in velocità un motore/due motori attraverso i pwm hardware.

Guida al modulo:

File di Variabili ---> mpwmvar.bas
File programma (soubroutines) ---> mpwm1.bas (CANALE 1)
File programma (soubroutines) ---> mpwm2.bas (CANALE 2)

```
gosub mpwm1                    'per attivare disattivare e settare il pwm hardware - CANALE 1
gosub mpwm2                    'per attivare disattivare e settare il pwm hardware - CANALE 2
gosub mpwm1state               'per sapere lo stato dei registri - CANALE 1
gosub mpwm2state               'per sapere lo stato dei registri - CANALE 2
mpwmduty (variabile)           'byte valore del duty-cicle del pwm
monoffpwm (variabile)          'bit accendi/spegni pwm (1=on ; 0=off)
mpwmpr2 (variabile)            'byte valore del registro PR2 (periodo pwm)
mpwmpresc (variabile)          'byte valore prescaler (da 0 a 2) - divisore frequenza (vedi calcoli)
mpwmpin1 (symbol)              'pin pwm canale 1 (già impostato nel modulo variabili)
mpwmpin2 (symbol)              'pin pwm canale 2 (già impostato nel modulo variabili)
```

NOTE PWM CONTROLLO MOTORI:
Per maggiori dettagli consultare il DataSheet del PIC sotto la voce Capture/Compare/Pwm.
Per impostare i pin di uscita prestabiliti per il pwm hardware (consultare datasheet)
modificare i symbols del file di variabili inserendo il pin prestabilito per i due canali (max) del pwm.
Per i pic con un solo pwm, includere solamente il file di programma del canale 1, per i dispositivi
da due pwm, includere entrambi i files.
Nota bene che il file di variabili va incluso in entrambi i casi.
Per agire sui canali chiamare la sub corrispondente al canale scelto (vedi sopra). Le variabili di
impostazione sono le stesse per entrambe i canali (passaggio parametri).
Quindi si caricano i valori sempre nelle stesse variabili, ma verrà impostato con i valori
caricati solo il canale che si decide di impostare (richiamando la sub corrispondente)
La frequenza del pwm (il periodo e il prescaler) ovviamente è uguale per tutti e due
i canali, quello che cambia è il valore del duty-cicle.
Calcolo Del valore di PR2 in base alla frequenza:
La seguente equazione permette di ricavare il Periodo di PWM:
periodo PWM =[(PR2)+1]*4*Tosc*(TMR2 prescale value)
dove:
PR2 è il valore inserito nel registro PR2
Tosc è l'inverso della frequenza (in Hz) del quarzo esterno che fornisce il clock al PIC (1/Fosc)
TMR2 prescale value è il valore del prescaler (vedi valori prescaler)
Per calcolare il valore di PR2 sapendo la frequenza basta fare la formula inversa, quindi:
PR2 = periodo PWM/(4*Tosc*(TMR2 prescale value)) - 1
ricordandosi che: periodo PWM = 1/(frequenza pwm)
e sapendo che Fosc = 1 / Tosc
PR2 = Fosc/(frequenza PWM * 4 * TMR2 prescale value) - 1
dove Fosc è la frequenza dell'oscillatore del pic.
La frequenza può essere variata anche impostando il prescaler nel modo corretto.

Il valore del registro (mpwmpresc) può essere 0, 1 o 2. A seconda del valore del registro
i valori "efficaci" del prescaler, cioè quelli da sostituire nelle formule (TMR2 Prescale Value),
sono i seguenti:
mpwmpresc = 0 - Prescaler = 1
mpwmpresc = 1 - Prescaler = 4
mpwmpresc = 2 - Prescaler = 16
quando il valore di mpwmpresc = 0 il prescaler è disattivato. Esso corrisponde al
valore efficace 1, quindi questa impostazione è anche detta "prescaler unitario".
Da non confondere il valore "efficace" con il valore del registro. Per maggiori informazioni
consultare il datasheet del PIC alle voci "Capture/Compare/Pwm" e "Timer 2 Module".
Per sapere lo stato dei registri, sia quelli generali che quelli propri di ogni canale,
basta chiamare la sub mpwm1state per il primo canale o mpwm2state per il secondo.
Le variabili Buffer verranno riempite con i valori richiesti. I valori generali quali
periodo e prescaler saranno gli stessi a seconda che si chiami la sub "stato" del primo
o del secondo canale. La variabile del duty-cicle ed il flag on/off verranno caricati con i
valori di stato corrispondenti al primo o al secondo canale in base a quale sub di stato
è stata chiamata.
I symbols dei pin dei canali pwm possono essere usati quando il modulo pwm hardware è spento
per operazioni digitali sul pin.

Codice File Variabili:

```
'file di inizializzazione variabili

'per avere la stessa frequenza di 4096Mhz (prescaler unitario):
'per osc=4mhz --> pwmperiod = 255
'per osc=20Mhz --> pwmperiod = 52

'-------------------------------------------------------------------------------

mpwmbyte0 var byte                          'byte contenente le variabili BIT del programma
mpwmduty VAR BYTE                           'variabile valore duty-cicle
symbol mpwmpin1 = PORTB.3                   'pin corrispondente al canale 1 del pwm
symbol mpwmpin2 = PORTB.3                   'pin corrispondente al canale 2 del pwm
monoffpwm VAR mpwmbyte0.0                   'flag per attivare/disattivare (1/0) il pwm
mpwmpr2 VAR BYTE                            'valore del registro pr2 (vedi formule pwm)
mpwmpresc VAR BYTE                          'valore del prescaler (da 0 a 3)

'-------------------------------------------------------------------------------
```

Codice File Programma 1:

```
'sub di controllo motori con pwm hardware - canale 1

'-------------------------------------------------------------------------------

mpwm1:
T2CON = 0                                   'azzera il registro del timer
```

```
PR2 = mpwmpr2                          'imposto il periodo del pwm
T2CON.0 = mpwmpresc.0                  'imposto il valore del prescaler
T2CON.1 = mpwmpresc.1
T2CON.2 = 1                            'attivo il timer2
CCP1CON = 0                            'azzera il registro di configurazione
CCP1CON.5 = 1                          'imposto i bit meno significativi del dutycicle a 1
CCP1CON.4 = 1
CCPR1L = mpwmduty                      'imposto i restanti 8 bit del duty-cicle
IF monoffpwm = 0 Then                  'decido in base al flag se accendere o spegnere il pwm
CCP1CON = 0                            'fermo il pwm
input mpwmpin1                         'imposto il pin del pwm come ingresso
Else
Output mpwmpin1                        'imposto il pin del pwm come uscita
CCP1CON.3 = 1                          'imposto i due bit per attivare la modalità PWM
CCP1CON.2 = 1
EndIF                                  'fine condizione
Return                                 'ritorno alla sub chiamante

mpwm1state:                            'restituisce lo stato dei registri (caricandoli nei buffers)
mpwmpr2 = PR2                          'restituisce il valore generale del periodo
mpwmpresc.0 = T2CON.0                  'restituisce il valore generale del prescaler
mpwmpresc.1 = T2CON.1
mpwmduty = CCPR1L                      'restituisce il valore del duty-cicle (CH1) nel buffer
if CCP1CON = 0 then                    'controlla lo stato del registro per sapere se il pwm e on o off
monoffpwm = 0                          'restituisce il valore 0 se il pwm è spento (CH1)
else                                   'altrimenti
monoffpwm = 1                          'restituisce il valore 0 se il pwm è acceso (CH1)
endif
return                                 'ritorno alla sub chiamante

'-----------------------------------------------------------------------------
```

Codice File Programma 2:

```
'sub di controllo motori con pwm hardware - canale 2

'-----------------------------------------------------------------------------

mpwm2:
T2CON = 0                              'azzera il registro del timer
PR2 = mpwmpr2                          'imposto il periodo del pwm
T2CON.0 = mpwmpresc.0                  'imposto il valore del prescaler
T2CON.1 = mpwmpresc.1
T2CON.2 = 1                            'attivo il timer2
CCP2CON = 0                            'azzera il registro di configurazione
CCP2CON.5 = 1                          'imposto i bit meno significativi del dutycicle a 1
CCP2CON.4 = 1
CCPR2L = mpwmduty                      'imposto i restanti 8 bit del duty-cicle
IF monoffpwm = 0 Then                  'decido in base al flag se accendere o spegnere il pwm
```

```
CCP2CON = 0                          'fermo il pwm
input mpwmpin2                       'imposto il pin del pwm come ingresso
Else
Output mpwmpin2                      'imposto il pin del pwm come uscita
CCP2CON.3 = 1                        'imposto i due bit per attivare la modalità PWM
CCP2CON.2 = 1
EndIF                                'fine condizione
Return                               'ritorno alla sub chiamante

mpwm2state:                          'restituisce lo stato dei registri (caricandoli nei buffers)
mpwmpr2 = PR2                        'restituisce il valore generale del periodo
mpwmpresc.0 = T2CON.0                'restituisce il valore generale del prescaler
mpwmpresc.1 = T2CON.1
mpwmduty = CCPR2L                    'restituisce il valore del duty-cicle (CH2) nel buffer
if CCP2CON = 0 then                  'controlla lo stato del registro per sapere se il pwm e on o off
monoffpwm = 0                        'restituisce il valore 0 se il pwm è spento (CH2)
else                                 'altrimenti
monoffpwm = 1                        'restituisce il valore 0 se il pwm è acceso (CH2)
endif
return                               'ritorno alla sub chiamante

'--------------------------------------------------------------------------------
```

Modulo controllo JoyPad Playstation:

Attraverso questo modulo è possibile leggere il famoso GamePad di casa Sony ed attivarne la vibrazione e la modalità analogica.

Guida al modulo:

File di Variabili ---> psxvar.bas
File di Setup ---> psxset.bas
File programma (soubroutines) ---> psx.bas

ATTENZIONE: Nel circuito va messa una resistenza di PULL-UP sul segnale DAT perche tutto funzioni.
Inoltre se si intende far uso del segnale ACK, va messa una resistenza di pullup
anche su questo pin (resistenze da 1Kohm vanno più che bene).

```
gosub readpsx                   'per leggere il joypad senza impartire comandi alla vibrazione
gosub readpsxvib                'per leggere il joypad e impartire comandi ai motori vibranti
gosub psxchkmode                'per configurare la modalità analogica + vibrazione e controllare
                                'se tali modalità risultano attive. In caso contrario le attiva.
smallm (variabile)              'byte valore on/off (1/0) per il motore vibrante piccolo
largem (variabile)          'valore velocità (0-255) per il motore vibrante grande
digi1 (variabile)           'byte tasti digitali 1
digi2 (variabile)           'byte tasti digitali 2
analog1 (variabile)         'byte joy analogico destro 1 - l/r
analog2 (variabile)         'byte joy analogico destro 2 - u/d
analog3 (variabile)         'byte joy analogico sinistro 1 - l/r
analog4 (variabile)         'byte joy analogico sinistro 2 - u/d
psxid (variabile)           'byte che restituisce la modalità attiva del joypad
                                '$41 ---> Digitale  ;  $73 ---> Analogica
                                '$FF (decimale 255) ---> Problemi Hardware
                                'se si fa uso della sub "psxcheckmode"  $00 (decimale 0)
                                'indica che l'hardware non è connesso (al posto di $FF)
Alias per i tasti ed i joystik analogici:
keyX                        'alias per il tasto X
keyC                        'alias per il tasto Cerchio
keyQ                        'alias per il tasto Quadrato
keyT                        'alias per il tasto Triangolo
keyR1                       'alias per il tasto R1
keyL1                       'alias per il tasto L1
keyR2                       'alias per il tasto R2
keyL2                       'alias per il tasto L2
keyL                        'alias per il tasto Freccia Left
keyR                        'alias per il tasto Freccia Right
keyU                        'alias per il tasto Freccia Up
keyD                        'alias per il tasto Freccia Down
keyST                       'alias per il tasto Start
keySL                       'alias per il tasto Select
keyJL                       'alias per il tasto Joystik Left
keyJR                       'alias per il tasto Joystik Right
```

```
joyRX                   'alias valore asse X - Joystik Right
joyRY                   'alias valore asse Y - Joystik Right
joyLX                   'alias valore asse X - Joystik Left
joyLY                   'alias valore asse Y - Joystik Left
```

NOTE CONTROLLO JOYPAD PSX:
I pin di collegamento al joypad sono impostabili nel modulo delle variabili (symbols).
Per leggere la psx senza vibrare chiamare "readpsx", per vibrare "readpsxvib".
La virazione e la sua intensità viene impostata con le due variabili dei motori.
Se le variabili sono poste a 0 i la vibrazione è annullata.
Se si intende usare la modalità analogica e con vibrazione si deve chiamare la sub di impostazione
"psxcheckmode" dalla sub main del programma principale.
La sub "psxchekmode" deve essere chiamata anche ogni volta prima di una lettura, per verificare se
la vibrazione e la modalità analogica sono attive. Se non sono attive le attiva.
Tale sub verifica anche si il joypad non è connesso (modalità = $FF) e restituisce il valore 0
sulla variabile psxid. Se tale variabile vale quindi 0 dopo aver chiamato la sub "psxcheckmode"
significa che il joypad non è connesso o ci sono stati problemni hardware.
Se non si intende usare la sub di impostazione/check messa a disposizione e si vuole agire manualmente sulle
impostazioni del controller, leffere il documento delle impostazioni delle modalità per maggiori informazioni.
Il linea generale per modificare le impostazioni si deve:
1) Entrare nella modalità configurazione
2) Selezionare UNA configurazione (per esempio modalità analogica oppure modalità vibrazione)
3) Uscire dalla configurazione (non la Native)
4) Ripetere per ogni modalità da impostare
5) con la sub "querymode" è possibile sapere la modalità attiva (che può essere digitale, analogica, DS2 Native,
   oppure hardware non connesso se vale 255 ($FF)
6) Consultare il file di programma per la lista e descrizione delle sub di configurazione disponibili.
Gli alias dei tasti indicano se un tasto è premuto o no, oppure il valore dei joystik analogici
(da 0 a 255) per i due assi (dall'alto 0 al basso 255, da sinistra 0 a destra 255).
Per eseguire operazioni direttamente sui dati trasmessi (per esempio serout)
si possono usare le variabili "digi1, digi2, analog1, analog2 analog3, analog4"
che contengono i valori dei tasti e dei joystik analogici (vedi sistema di
trasmmissione psx controller). Lo schema dei bit delle variabili lette corrisponte agli alias
secondo le definizioni che si possono trovare nel modulo delle variabili.
NOTA BENE: Tutti i pulsanti sono attivi bassi (0 = premuto)
In caso si verificassero problemi di lettura/scrittura del joypad può essere necessario includere
nel programma principale la define DEFINE SHIFT_PAUSEUS 50 (utile per far comprendere bene al joypad i comandi)
Il consiglio è di mettere sempre la define, soprattutto se si fa uso della modalità analogica + vibrazione.

Codice file Variabili:

```
'-----------------------------------------------------------------------------

'Dichiarazione dei Symbols dei Pin
symbol cmd = PORTB.1            'pin uscita (comando)
symbol clk = PORTB.2            'pin clock-out (sincronizzazione) - low
symbol att = PORTB.7            'attivazione joypad (attivo a livello basso)
symbol dat = PORTB.3            'pin ingresso (ricezione dati) - Inverted Logic
symbol ack = PORTB.5            'pin ingresso risposta ACK (attivo basso)
```

```
'è necessaria una resistenza di pullup sul DAT

'Dichiarzioni Variabili e costanti
digi1 VAR BYTE                  'byte tasti digitali 1
digi2 VAR BYTE                  'byte tasti digitali 2
analog1 VAR BYTE                'byte joy analogico destro 1 - l/r
analog2 VAR BYTE                'byte joy analogico destro 2 - u/d
analog3 VAR BYTE                'byte joy analogico sinistro 1 - l/r
analog4 VAR BYTE                'byte joy analogico sinistro 2 - u/d
psxid VAR BYTE                  'byte dell'ID del controller (modalità attiva)
smallm VAR BYTE                 'on off motore piccolo (vibrazione)
largem VAR BYTE                 'valore velocità motore grande vibrazione
cont VAR BYTE                   'semplice contatore per cicli

'Alias Variabili
keyX VAR digi2.6                'alias per il tasto X
keyC VAR digi2.5                'alias per il tasto Cerchio
keyQ VAR digi2.7                'alias per il tasto Quadrato
keyT VAR digi2.4                'alias per il tasto Triangolo
keyR1 VAR digi2.3               'alias per il tasto R1
keyL1 VAR digi2.2               'alias per il tasto L1
keyR2 VAR digi2.1               'alias per il tasto R2
keyL2 VAR digi2.0               'alias per il tasto L2
keyL VAR digi1.7                'alias per il tasto Freccia Left
keyR VAR digi1.5                'alias per il tasto Freccia Right
keyU VAR digi1.4                'alias per il tasto Freccia Up
keyD VAR digi1.6                'alias per il tasto Freccia Down
keyST VAR digi1.3               'alias per il tasto Start
keySL VAR digi1.0               'alias per il tasto Select
keyJL VAR digi1.2               'alias per il tasto Joystik Left
keyJR VAR digi1.1               'alias per il tasto Joystik Right
joyRX VAR analog1               'alias valore asse X - Joystik Right
joyRY VAR analog2               'alias valore asse Y - Joystik Right
joyLX VAR analog3               'alias valore asse X - Joystik Left
joyLY VAR analog4               'alias valore asse Y - Joystik Left

'Settaggio Uscite/Ingressi
Output cmd
Output clk
Output dat
Input dat
Input ack

'-------------------------------------------------------------------------------
```

Codice file Settaggi:

```
psxchkmode:                     'sub per il controllo della modalità
GoSub querymode                 'leggo la modalità attiva
pause 1
```

```
if psxid = $FF then                 'se l'id restituito è 255(dec), il controller non risponde perchè è una modalità non contemplata
psxid = 0                           'azzero il psxid (errore hardware)
return                              'esco dal ciclo e torno alla sub chiamante
endif
IF psxid <> $73 Then                'se la modalità non è analogica
For cont = 1 T0 3                   'per 5 volte imposto la modalità analogica con vibrazione
GoSub confenter                     'entro nel modo configurazione
Pause 1                             'e faccio in modo che il codice inviato venga "capito"
GoSub confenter
Pause 1
GoSub setanalog                     'imposto la modalità analogica
Pause 1
GoSub setanalog
Pause 1
GoSub confexit                      'esco dalla configurazione
Pause 1
GoSub confexit
Pause 1
GoSub confenter                     'entro nella configurazione
Pause 1
GoSub confenter
Pause 1
GoSub setvibration                  'attivo la vibrazione
Pause 1
GoSub setvibration
Pause 1
GoSub confexit                      'esco dalla configurazione
Pause 1
GoSub confexit
Next cont                           'rieseguo il ciclo
GoTo psxchkmode                     'torno al check per verificare l'impostazione effettuata
Else                                'altrimenti         (se l'impostazione è corretta -- analogico)
Return                              'ritorno alla sub chiamante
EndIF                               'fine condizione
```

Codice file Programma:

```
'-------------------------------------------------------------------------------

'Sub dei comandi:
'Per le sequenze da inviare ed i valori delle modalità consultare il documento dei comandi.
'Per la modalità digitale le variabili analog hanno valore $0


confenter:                                          'entro nella configurazione
Low att                                             'attivo il controller
Shift0ut cmd,clk,4,[$1,$43,$0,$1,$0,$0,$0,$0,$0]    'invio in uscita la sequenza giusta (bytes)
High att                                            'disattivo il controller
Return                                              'torno alla sub chiamante
```

```
confexit:                                       'esco dalla configurazione
Low att
ShiftOut cmd,clk,4,[$01,$43,$00,$00,$00,$00,$00,$00,$00]
High att
Return

setvibration:                                   'attivo la vibrazione
Low att
ShiftOut cmd,clk,4,[$1,$4D,$0,$0,$1,$ff,$ff,$ff,$ff]
High att
Return

setanalog:                                      'setto la modalità analogica
Low att
ShiftOut cmd,clk,4,[$1,$44,$0,$1,$3,$0,$0,$0,$0]
High att
Return

querymode:                                      'controllo la modalità attiva
Low att
ShiftOut cmd,clk,4,[$1]
ShiftIn dat,clk,7,[psxid]                       'carico nella variabile il valore della modalità letta
High att
Return

readpsxvib:                                     'leggo il valore dei tasti e vibro
Low att
ShiftOut cmd,clk,4,[$1,$42,$0]                  'leggo i tasti digitali
ShiftIn dat,clk,7,[digi1,digi2]                 'carico nelle variabili digi1 e digi2 i valori dei tasti
High att
Low att
ShiftOut cmd,clk,4,[$1,$42,$0,smallm,largem]    'vibro secondo le impostazioni delle variabili dei due motori
ShiftIn dat,clk,7,[analog1,analog2,analog3,analog4]  'carico i valori dei joistik analogici nelle 4 variabili
High att
Return

readpsx:                                        'leggo i tasti della psx senza vibrare (sia digitali che analogici)
Low att
ShiftOut cmd,clk,4,[$1,$42,$0]
ShiftIn dat,clk,7,[digi1,digi2,analog1,analog2,analog3,analog4]
High att
Return

'-------------------------------------------------------------------------------
```

Modulo Comunicazione I2C:

Questo modulo permette la comunicazione su BUS I2C con canale BUSY. Il modulo gestisce le temporizzazioni per l'accesso al bus in modo da evitare scontri tra telegrammi in caso di più dispositivi master. Compatibile con tutte le periferiche I2C

Guida al Modulo:

File di Variabili ---> i2cvar.bas
File di Setup ---> i2cset.bas
File programma (soubroutines) ---> i2cmod.bas

```
gosub i2cset                    'inizializza il bus e attiva i parametri standard (vedi note)
gosub i2ctx                     'trasmette un dato sul bus (buffer in uscita = i2cout)
gosub i2crx                     'riceve un dato dal bus i2c (buffer in ingresso = i2cin)
gosub i2cmemorychk              'effettua il controllo in lettura e scrittura sulla memoria (v. note)
gosub i2copen                   'occupa il bus, per gestione personalizzata con comandi picbasic
gosub i2cclose                  'disoccupa il bus, gestione personalizzata (vedi note)
i2cackerror:                    'sub chiamata in caso di errore ack (vedi note)
i2cbusy:                        'sub chiamata in caso di errore "bus occupato" (vedi note)
i2cmemoryaccesserror:           'sub chiamata in caso di errore sul check della memoria
i2cin (variabile)               'buffer (recipiente) byte in ingresso (letto)
i2cout (variabile)              'buffer (recipiente) byte in uscita (da scrivere)
i2ccontrol (variabile)      'byte di controllo dispositivo (vedi dettagli inizializzazione)
i2caddr (variabile)             'word (oppure byte) indirizzo di accesso alla locazione di memoria
i2caddrbw (variabile)           'flag per selezionare tra indirizzo WORD (1) o BYTE (0) (i2caddr.byte0)
i2cwrep (variabile)             'flag che attiva (1) la modalità di attesa automatica del bus libero
i2cdelay (variabile)            'word tempo in us per l'esecuzione delle operazioni nel dispositivo slave
i2cbusycheck (variabile)        'word tempo in us tra un controllo e l'altro del segnale busy
i2cretflag (variabile)          'imposta la modalità "flag di ritorno" in caso di errore (vedi dettagli)
i2cflag (variabile)             'byte contenente i flags di errore (vedere gli alias per dettagli)
i2cackflag (alias)              'flag "errore ack non ricevuto" (1) corrispondente a i2cflag.1
i2cbusyflag (alias)             'flag "errore bus occupato" (1) corrispondente a i2cflag.0
i2cmemorychkflag (alias)        'flag "errore accesso memoria) (1) corrispondente a i2cflag.2
```

NOTE MODULO COMUNICAZION I2C:

Questi moduli permettono di avere accesso a dispositivi i2c compatibili aggiungendo numerose funzioni per una gestione decentralizzata della memoria condivisa in una rete di microprocessori.
Oltre alle linee dati e clock dell'i2c è stata introdotta una terza linea che indica se il bus è occupato (livello logico 1). Ogni processore verifica che il bus sia libero prima di eseguire ogni operazione i2c e durante l'esecuzione dell'operazione alza la linea "bus occupato" in modo che gli altri processori lo sappiano, evitando così collisioni tra telegrammi.
Il check del bus avviene due volte: se il bus è libero il programma attente per un certo tempo, impostabile con la variabile i2cbusycheck (in us) e poi esegue un altro controllo. Solo in caso di esito positivo esegue le operazioni successive, altrimenti attende di nuovo che il busy sia basso ricominciando dal primo check.
Quando sul bus ci sono più di 2 dispositivi questo può essere utile per aumentare la sicurezza ed evitare collisioni accidentali dei telegrammi: basta impostare per ogni pic un valore differente di pausa.
Usare i moduli i2c è facile. Si hanno a disposizione due buffer (uno per i dati un uscita ed uno per i dati in

ingresso) e due sub da chiamare rispettivamente per trasmettere o ricevere dati dal bus.
la variabile i2caddr indica l'indirizzo della locazione di memoria sul dispositivo slave su cui eseguire l'operazione. L'indirizzo è impostato di default (setup) come word (16 bit di indirizzo), ma si può lavorare su dispositivi con indirizzi da 8 bit (BYTE) agendo sul flag i2caddbw (1 per la WORD). In caso si utilizzasse un indirizzo BYTE caricare solo i primi otto bit della variabile i2caddr (che è dichiarata come word), quindi l'indirizzo a 8 bit sarà i2caddr.byte0.
sono stati aggiunti i comandi per aprire e chiudere il bus manualmente. Tali comandi agiscono solo sulla terza linea e possono essere usati per creare gestioni avanzate della comunicazione i2c (con i comandi picbasic) pur mantenendo la sicurezza contro le collisioni di telegrammi.
Nel programma principale è d'obbligo mettere una pausa anticollisione tra un accesso e l'altro al bus.
La pausa è dell'ordine delle decine o centinaia di microsecondi.
Questo può essere utile per inviare un treno di dati tenendo il bus occupato. Una volta occupato il bus, eseguire le sub di lettura/scrittura come si vuole, poi per abbassare il busy per rendere accessibile il bus ad altri dispositivi.
chiamando la sub di inizializzazione vengono settati i parametri di default previsti, come i pin delle linee ed i flags delle impostazioni interne. la variabile i2ccontrol è settata per default su %10100000, codice di controllo per l'accesso ad una memoria i2c standard. Anche il tempo di delay è impostato su 5ms (5000us).
Altre impostazioni predefinite sono i2caddbw = 1, i2cwrep = 1, i2cretflag = 0.
Questo modulo prevede una gestione degli errori con diverse modalità, utilizzabili a seconda delle esigenze, impostando correttamente alcuni flag.
Impostando il flag i2cretflag si attiva la modalità "flag di ritorno". Se la modalità è attiva e si verifica un errore di ack non ricevuto, un errore di bus occupato o un errore di check memoria il programma i2c verrà terminato senza eseguire altre operazioni e verrà impostato il flag errore corrispondente all'errore verificatosi (vedi i2cflag). Al ritorno dalla chiamata alla sub i2c (nel programma principale) sarà necessario quindi gestire eventuali errori basandosi sui flag restituiti.
Se la modalità "flag di ritorno" è disattivata, in caso di errore verrà invece chiamata la sub (goto) corrispondente al tipo di errore verificatosi (vedi sopra). Nota Bene che la sub deve essere sempre presente, anche se non utilizzata, indifferentemente dalla modalità di gestione errori scelta.
Con questa modalità non si può sapere però quale chiamata nel programma principale ha riscontrato un errore, quindi sarà necessaria una gestione errori avanzata esterna, per esempio usando un numero identificativo per ogni operazione.
Per ovviare all'incoveniente dell'errore "bus occupato" quando ci sono più di un dispositivo master che hanno l'accesso al bus (diversi processori sullo stesso bus) è stata introdotta una modalità di attesa automatica.
Questa modalità, attivabile con il flag i2cwrep (1), fa in modo che il programma attenda in automatico che il bus si liberi, in caso sia occupato, prima di eseguire l'operazione di lettura/scrittura richiesta (senza limite di tempo).
Questa modalità ha la priorità rispetto alle altre modalità di gestione errori, per quanto riguarda l'errore di bus occupato. Gli altri tipi di errori vengono gestiti normalmente con la modalità impostata.
Il modulo di comunicazioni i2c è dotato infine di un'utile strumento per le memorie i2c.
La sub i2cmemorychk può essere chiamata solo dall'esterno e può essere usata per verificare se una memoria i2c è accessibile. Il programma prova a scrivere e leggere la prima cella della memoria con due differenti valori per controllare se la memoria funziona. Se la memoria è scollegata dal bus, oppure è protetta da scrittura viene segnalato l'errore con uno dei due metodi impostati (sub comune oppure flag di ritorno). In caso il test venga superato, la prima locazione della memoria viene scritta con il valore letto prima del test (rescue del valore) ed il sottoprogramma i2cmemorychk viene terminato (return). Come nelle altre chiamate alle sub funzionali del modulo, sarà necessaria la gestione errori, diversa a seconda della modalità di gestione errori scelta.
Questa sub è stata progettata per essere caricata all'avvio del programma principale, nell'inizializzazione, per verificare l'hardware collegato.
Nota bene che durante l'esecuzione di i2cmemorychk viene automaticamente impostato il flag di attesa del bus libero e quello per la modalità flag di ritorno. Tali flag verranno ripristinati secondo le modalità impostate (rescue) alla fine del sottoprogramma. La gestione errori prosegue come impostata. Il motivo per cui vengono cambiati i flag è per fare in modo che il programma segnali anche l'errore ack come errore di check memoria. Nella sub di gestione dell'errore

di check si potrà verificare se la causa è l'ack controllando il relativo flag. Se invece è attiva la modalità "flag di ritorno" basta verificare se oltre al flag di check è presente anche il flag di errore ack. A seconda che si sia verificato o no un errore ack durante il test della memoria si può capire se è un problema di collegamento hardware o un problema di lettura/scrittura.
Per poter usufruire di più canali I2C sullo stesso pic è stata creata anche una versione modificata del modulo, che si appoggia ai moduli "pin variabili". In caso si usasse questa versione è necessario includere anche i suddetti moduli. Con questa versione è possibile scegliere i pin del bus tramite il passaggio parametri (vedi descrizione moduli pin variabili per maggiori info).
Per la versione modificata, le variabili dei pin sono:
sda (byte) --> pin data
scl (byte) --> pin clock
sbusy (byte) --> pin segnale "bus occupato"
In caso vi fossero dubbi, consultare i commenti ed il sorgente dei componenti del modulo.
Per bus con frequenza massima di 100Khz all'inizio del programma primo inserire la define
DEFINE I2C_SLOW 1

Codice File Variabili:

```
'Modulo delle variabili protocollo di comunicazione i2c

'variabili usate:
'DEFINE I2C_SLOW 1                                  'da usare in caso di problemi, con quarzi > 20Mhz
'DEFINE SHIFT_PAUSEUS 50                            'da usare in caso di problemi, con quarzi > 20Mhz
symbol sda = PORTC.4                                'pin linea dati
symbol scl = PORTC.3                                'pin linea clock
symbol sbusy = PORTC.5                              'pin linea "bus busy"
i2cbyte0 var byte                                   'byte contenente le variabili BIT del programma
i2cout VAR BYTE                                     'recipiente byte dato in uscita
i2cin VAR BYTE                                      'recipiente byte dato in ingresso
i2ccontrol VAR BYTE                                 'byte di controllo del dispositivo (il bit r/w è autoimpostante)
i2caddr VAR WORD                                    'indirizzo locazione su cui lavorare (byte0 oppure intera word)
i2caddrbw VAR i2cbyte0.0                            'flag di selezione tra "indirizzo byte" ed "indirizzo word" (0/1)
i2cwrep VAR i2cbyte0.1                              'flag per impostare l'attesa automatica della liberazione del bus
i2cretflag VAR i2cbyte0.2                           'in caso di errore restituisce il codice errore sul byte i2cflag
i2cdelay VAR WORD                                   'pausa funzionale impegno dispositivo slave (in microsecondi) - può
essere sostituito da una costante
i2cbusycheck var word                               'pausa tra un check è l'altro del canale busy (in microsecondi) - può
essere sostituito da una costante
i2cflag VAR BYTE                                    'flag stato ritorno dalla sub (disabilita le sub errore)
i2cexitflag VAR i2cbyte0.3                          'indica di uscire dal sottoprogramma i2c senza eseguire operazioni
i2cackflag VAR i2cflag.1                            'alias flag errore ack
i2cbusyflag VAR i2cflag.0                           'alias flag errore bus occupato
i2cmemorychkflag VAR i2cflag.2                      'alias flag errore controllo accesso memoria fallito (R/W)
i2cchkrescue VAR BYTE          'variabile temporanea per il ripristino del dato memorizzato prima del chk della memoria
i2cwreprescue VAR i2cbyte0.4   'variabile temporanea per il ripristino del valore del flag di attesa bus libero
i2cretflagrescue var i2cbyte0.5 'variabile temporanea per il ripristino del valore del flag "modalità flag di ritorno"
i2cextbusy var i2cbyte0.6      'indica che il bus è stato occupato da un comando esterno


'variabili utili:
```

```
'recipienti i2cout ed i2cin
'flags errori i2cflag (e relativi aliases)
'byte di contrllo i2ccontrol
'recipiente indirizzo i2caddr
'flag indirizzo BYTE o WORD i2caddrbw
'flag per l'attesa del bus libero automatica i2cwrep
'flag per impostare la modalità "flag errore" i2cretflag
'tempo di lavoro del dispositivo slave (delay operazione) i2cdelay
```

Codice File Settaggi:

```
'Modulo di configurazione e setup i2c.
'Contiene le impostazioni di default ed inizializza il bus.


i2cset:                                          'sub di setaggio e configurazione
Input sbusy                                      'pin "bus occupato" come ingresso
Output scl                                       'pin "i2c clock" come uscita
i2cextbusy = 0                                   'azzero il flag busy esterno
        'imposto i flag e le variabili di default:
i2cdelay = 10                                    'delay = 10us
i2cbusycheck = 5                                 'pausa check bus libero in us
i2cwrep = 1                                      'attesa automatica del bus libero attiva
i2cretflag = 1                                   'abilito l'opzione di ritorno flag in caso di errore
i2caddrbw = 0                                    'indirizzo dispositivo slave = byte
i2ccontrol = %10100000                           'codice di controllo ram i2c
Return
```

Codice File Programma:

```
'Programma Trasmissione/Ricezione su Bus i2c.
'Per memori i2c e simili.
'Per le variabili consultare il modulo delle variabili, per il funzionamento
'generale consultare la guida ai moduli.
'Il programma utilizza i comandi picbasic i2cRead ed i2cWrite.
'Il programma usa una gestione avanzata del bus (3 vie) con un canale che
'gestisce il transito dei telegrammi (occupazione del bus)
'Per scrivere nel dispositivo slave chiamare la sub i2ctx.
'Per leggere chiamare la sub i2crx.
'I due recipienti dei dati letti o scritti sono i2cin e i2cout.
'è inoltre presente il controllo avanzato per il check della memoria in lettura
'e scrittura (test hardware).
'Vedere la guida per istruzioni dettagliate.


'NOTE AGGIUNTIVE:

'sub extra di controllo errori:
'i2cackerror:
```

```
'i2cbusy:
'i2cmemoryaccesserror

'sub comandi:
'gosub i2cset
'gosub i2ctx
'gosub i2crx
'gosub i2copen
'gosub i2cclose
'gosub i2cmemorychk

'sub principalmente utili:
'gosub i2ctx (recipiente i2cout)
'gosub i2crx (recipiente i2cin)

'variabili utili:
'recipienti i2cout ed i2cin
'flags errori i2cflag (e relativi aliases)
'byte di contrllo i2ccontrol
'recipiente indirizzo i2caddr
'flag indirizzo BYTE o WORD i2caddrbw
'flag per l'attesa del bus libero automatica i2cwrep
'flag per impostare la modalità "flag errore" i2cretflag
'tempo di lavoro del dispositivo slave (delay operazione) i2cdelay



i2cbusytest:                            'controllo se il bus è occupato
i2cflag = 0                             'azzero i flags di ritorno
i2cexitflag = 0                         'azzero il flag di uscita
Input sbusy                             'imposta il pin "bus occupato" come input
IF sbusy = 1 Then                       'se il bus è effettivamente occupato
        IF i2cwrep = 1 Then             'se il flag di "attesa bus libero" è attivo
        GoTo i2cbusytest                'eseguo l'operazione automatica di attesa e quando il bus risulta libero
        Else                            'effettuo l'operazione, altrimenti
                IF i2cretflag = 1 Then          'se è impostata l'opzione "flag di ritorno"
                i2cexitflag = 1                 'imposto il flag di uscita dalle sub
                i2cflag.0 = 1                   'imposto il flag "errore bus occupato"
                GoTo i2cbusytestreturn          'vado al punto di ritorno
                Else                            'se l'opzione è disattivata
                GoTo i2cbusy                    'vado alla sub del "bus occupato"
                EndIF
        EndIF
EndIF
pauseus i2cbusycheck                    'effettuo una pausa prima di ricontrollare (pausa di priorità)
IF sbusy = 1 Then                       'se il bus è effettivamente occupato
        IF i2cwrep = 1 Then             'se il flag di "attesa bus libero" è attivo
        GoTo i2cbusytest                'eseguo l'operazione automatica di attesa e quando il bus risulta libero
        Else                            'effettuo l'operazione, altrimenti
                IF i2cretflag = 1 Then          'se è impostata l'opzione "flag di ritorno"
```

```
                i2cexitflag = 1                 'imposto il flag di uscita dalle sub
                i2cflag.0 = 1                   'imposto il flag "errore bus occupato"
                GoTo i2cbusytestreturn          'vado al punto di ritorno
                Else                            'se l'opzione è disattivata
                GoTo i2cbusy                    'vado alla sub del "bus occupato"
                EndIF
        EndIF
EndIF
i2cbusytestreturn:                      'punto di ritorno
Return

i2ctx:                                  'scrivo sul dispositivo slave
i2ccontrol.0 = 0                        'imposto il flag R/W su scrittura
if i2cextbusy = 0 then                  'se il busy non è controllato esternamente
GoSub i2cbusytest                       'controllo se il bus è occupato
endif
IF i2cexitflag = 1 Then GoTo i2ctxreturn        'esco dalla sub se il flag è true
High sbusy                              'segnalo che il bus è occupato
IF i2caddrbw = 1 Then                   'se è impostato l'utilizzo di un indirizzo a 16 bit
        IF i2cretflag = 1 Then          'se attivo i flag di ritorno NON chiamo la sub errore ack "esterna"
        I2CWrite sda,scl,i2ccontrol,i2caddr,[i2cout],i2cackexit         'scrivo nel dispositivo (addr = word)
        Else                                                            'altrimenti la chiamo
        I2CWrite sda,scl,i2ccontrol,i2caddr,[i2cout],i2cackfail         'scrivo nel dispositivo (addr = word)
        EndIF
        Else                                    'altrimenti uso un indirizzo a 8 bit
        IF i2cretflag = 1 Then                  'se attivo i flag di ritorno NON chiamo la sub errore ack "esterna"
        I2CWrite sda,scl,i2ccontrol,i2caddr.byte0,[i2cout],i2cackexit   'scrivo nel dispositivo (addr = byte)
        Else                                                            'altrimenti la chiamo
        I2CWrite sda,scl,i2ccontrol,i2caddr.byte0,[i2cout],i2cackfail   'scrivo nel dispositivo (addr = byte)
        EndIF
EndIF
i2cout = 0                      'azzero il recipiente in uscita
PauseUs i2cdelay                'pausa per l'esecuzione delle operazioni interne al dispositivo slave (delay)
if i2cextbusy = 0 then          'se il busy non è controllato esternamente
Low sbusy                       'abbassa la linea "bus occupato"
Input sbusy                     'imposta il pin "bus occupato" come input
endif
i2ctxreturn:                    'punto di ritorno
Return

i2crx:                          'leggo dal dispositivo slave
i2ccontrol.0 = 1                'imposto il flag R/W su lettura
i2cin = 0                       'azzera il recipiente in ingresso
if i2cextbusy = 0 then          'se il busy non è controllato esternamente
GoSub i2cbusytest               'controllo se il bus è occupato
endif
IF i2cexitflag = 1 Then GoTo i2crxreturn        'esco dalla sub se il flag è true
High sbusy                      'segnalo che il bus è occupato
IF i2caddrbw = 1 Then           'se è impostato l'utilizzo di un indirizzo a 16 bit
        IF i2cretflag = 1 Then  'se attivo i flag di ritorno NON chiamo la sub errore ack "esterna"
```

```
        I2CRead sda,scl,i2ccontrol,i2caddr,[i2cin],i2cackexit        'leggo dal dispositivo (addr = word)
        Else                                                         'altrimenti la chiamo
        I2CRead sda,scl,i2ccontrol,i2caddr,[i2cin],i2cackfail        'leggo dal dispositivo (addr = word)
        EndIF
        Else                                                 'altrimenti uso un indirizzo a 8 bit
        IF i2cretflag = 1 Then                               'se attivo i flag di ritorno NON chiamo la sub errore ack "esterna"
        I2CRead sda,scl,i2ccontrol,i2caddr.byte0,[i2cin],i2cackexit  'leggo dal dispositivo (addr = byte)
        Else                                                         'altrimenti la chiamo
        I2CRead sda,scl,i2ccontrol,i2caddr.byte0,[i2cin],i2cackfail  'leggo dal dispositivo (addr = byte)
        EndIF
EndIF
PauseUs i2cdelay                        'pausa per l'esecuzione delle operazioni interne al dispositivo slave (delay)
if i2cextbusy = 0 then                  'se il busy non è controllato esternamente
Low sbusy                               'abbassa la linea "bus occupato"
Input sbusy                             'imposta il pin "bus occupato" come input
endif
i2crxreturn:                            'punto di ritorno
Return

i2copen:                                'apro il bus (occupo) - per uso esterno
GoSub i2cbusytest                                   'controllo se il bus è occupato
IF i2cexitflag = 1 Then GoTo i2copenreturn          'esco dalla sub se il flag è true
High sbusy                                          'dichiaro il bus occupato
i2cextbusy = 1                                      'indico che il busy è stato impostato dall'esterno
i2copenreturn:                                      'punto di ritorno
Return


i2cclose:                                   'chiudo il bus (disoccupo) - per uso esterno
IF i2cexitflag = 1 Then GoTo i2cclosereturn 'esco dalla sub se il flag è true
Pauseus i2cdelay                            'pausa per l'esecuzione delle operazioni interne al dispositivo slave (delay)
Low sbusy                                   'abbassa la linea "bus occupato"
i2cextbusy = 0                              'indico che il controllo del busy è libero
Input sbusy                                 'imposta il pin "bus occupato" come input
i2cclosereturn:                             'punto di ritorno
Return

i2cackexit:                     'errore ack non ricevuto - flag di uscita
Pauseus i2cdelay                'pausa per l'esecuzione delle operazioni interne al dispositivo slave (delay)
Low sbusy                       'abbassa la linea "bus occupato"
Input sbusy                     'imposta il pin "bus occupato" come input
i2cflag.1 = 1                   'imposto il flag di ritorno su "errore ack"
return

i2cackfail:                     'chiudo il bus prima di saltare alla vera sub esterna "errore ack"
Pauseus i2cdelay                'pausa per l'esecuzione delle operazioni interne al dispositivo slave (delay)
Low sbusy                        'abbassa la linea "bus occupato"
Input sbusy                     'imposta il pin "bus occupato" come input
goto i2cackerror                'salta alla sub esterna per la gestione dell'errore ack non ricevuto
```

```
i2cmemorychk:                   'esegue il check di una memoria (in lettura e scrittura)
i2cwreprescue = i2cwrep         'salvo il valore impostato del flag "attesa automatica bus libero"
i2cretflagrescue = i2cretflag   'salvo il valore impostato del flag "modalità flag di ritorno"
i2cwrep = 1                     'attivo l'attesa automatica del bus libero
i2cretflag = 1                  'attivo la modalità flag di ritorno per ignorare l'errore ack
i2cchkrescue = 0                'azzera la variabile temporanea
i2caddr = 0                     'effettuo il check sulla prima locazione della memoria
GoSub i2crx                     'leggo il contenuto della locazione
i2cchkrescue = i2cin            'salvo il dato letto nella variabile temporanea
i2cout = %10101010              'imposto il buffer in uscita con il primo valore
GoSub i2ctx                     'scrivo il primo valore
GoSub i2crx                     'leggo il valore scritto
IF i2cin = %10101010 Then       'se il valore scritto è corretto (primo valore) passo alla fase successiva
        i2cout = %01010101      'scrivo nel buffer in uscita il secondo valore
        GoSub i2ctx             'scrivo nella memoria
        GoSub i2crx             'leggo dalla memoria
        IF i2cin = %01010101 Then       'confronto il valore scritto, se è corretto il chk è superato
        i2cout = i2cchkrescue           'scrivo nel buffer in uscita il valore che era presente nella locazione prima del chk
        GoSub i2ctx                     'scrivo nella memoria
        GoTo i2cmemorychkreturn         'vado al punto di ritorno della sub
        Else                            'in caso il secondo controllo fosse fallito
                i2cretflag = i2cretflagrescue   'ripristino il valore del flag "modalità flag di ritorno"
                IF i2cretflag = 1 Then          'se è attiva la modalità "flag di ritorno"
                i2cflag.2 = 1                   'imposto il flag "errore accesso memoria"
                GoTo i2cmemorychkreturn         'vado al punto di ritorno
                Else                            'in caso contrario
                i2cwrep = i2cwreprescue         'ripristino il precedente valore del flag di attesa bus libero
                GoTo i2cmemoryaccesserror       'salto alla sub "errore accesso memoria"
                EndIF
        EndIF                                   'fine secondo controllo
Else                                            'se il primo controllo fallisce
        i2cretflag = i2cretflagrescue           'ripristino il valore del flag "modalità flag di ritorno"
        IF i2cretflag = 1 Then                  'se è attiva la modalità "flag di ritorno"
        i2cflag.2 = 1                           'imposto il flag "errore accesso memoria"
        GoTo i2cmemorychkreturn                 'vado al punto di ritorno
        Else                                    'in caso contrario
        i2cwrep = i2cwreprescue                 'ripristino il precedente valore del flag di attesa bus libero
        GoTo i2cmemoryaccesserror               'salto alla sub "errore accesso memoria"
        EndIF
EndIF                                           'fine del controllo
i2cmemorychkreturn:                             'punto di ritorno della sub
i2cretflag = i2cretflagrescue                   'ripristino il valore del flag "modalità flag di ritorno"
i2cwrep = i2cwreprescue                         'ripristino il precedente valore del flag di attesa bus libero
Return
```

Modulo Servi R/C:

Questo modulo permette di posizionare un servomotore in modo semplice.

Guida al Modulo:

File di Variabili ---> servovar.bas
File di Setup ---> servoset.bas
File programma (soubroutines) ---> servomod.bas

```
srcpin (variabile pin)              'pin a cui è collegato il servo (pin variabile - valore 0-15)
srcosck (variabile)                 'byte coefficiente cambio quarzo (4Mhz --> 10 ; 20Mhz --> 2)
srcposus (variabile)                'word posizione in microsecondi
srcposgrad (variabile)              'word posizione in gradi (da -120 a +120)
srcunit (flag)                      'bit per la selezione dell'unità di misura (0 --> us ; 1 --> gradi)
srcvelocity (variabile)             'velocità del servo, da 0 a 10
srccycles (variabile)               'numero impulsi da inviare (frequenza) - da 10 a 80
gosub srcsetpos                     'sub da chiamare per effettuare il posizionamento del servomotore
gosub srcset                        'sub configurazione modulo servi (default)
```

NOTE MODULO SERVI R/C:
Questo modulo serve per posizionare in maniera semplice ma efficace un servo r/c.
la variabile srcpin è un valore da 0 a 15 che identifica il pin a cui è collegato il servo.
la variabile srcosck deve esser modificata all'inizio del programma primo oppure direttamente nella sub
di configurazione del modulo a seconda del quarzo che si intende usare con il pic (4Mhz o 20Mhz).
Per far muovere il servo basta settarne i parametri come velocità, numero cicli e posizione e poi richiamare la sub
srcsetpos.
Tramite la variabile srcvelocity è possibile selezionare la velocità di spostamento del servo con un valore da 0 a 10
Può andare benissimo il valore di default (6) impostato nel file di setup.
Con la variabile srccycles si può impostare il numero di impulsi da inviare al servo (utile per effettuare brevi, ma veloci
spostamenti). Leggere il tutorial sui servi per maggiori informazioni.
tramite il flag srcunit è possibile selezionare se il valore di posizione è dato in microsecondi oppure in gradi.
Una volta selezionata l'unità di misura si carica la variabile corrispondente con il valore di posizione. Una volta
eseguito il posizionamento, si potrà leggere il valore dell'altra variabile (unità di misura non usata) per sapere
il valore approsimativo convertito in quella unità.
Per il posizionamento in microsecondi, secondo lo standard dei servi (1ms - 2ms) caricare il valore (in us) nella
variabile srcposus (con srcunit = 0)
per il posizionamento in gradi (con srcunit = 1) caricare nella variabile srcposgrad un valore compreso tra -120
e +120, corrispondenti ai gradi di rotazione del servo, con zero centrale fissato su 1,5ms.
Di default è impostata l'unità di misura su microsecondi (più precisa).

Codice File Variabili:

```
'Variabili e symbols:
srcbyte0 var byte                   'byte contenente variabili BIT usate nel programma
srcpin var byte                     'pin comando servomotore (da impostare)
srcosck var byte                    'variabile coefficiente cambio quarzo
srcposus var word                   'posizione in microsecondi del servo
```

```
srcposgrad var word                    'posizione in gradi (da -120° a +120°)
srcunit var srcbyte0.0                 'flag scelta unità di misura (1 per i gradi)
srcvelocity var byte                   'velocità del servo (da 0 a 10)
srccycles var byte                     'numero di impulsi da inviare (da 10 a 80)
srccont var byte                       'variabile del contatore di cicli
srcpulse var word                      'variabile lunghezza impulso (unità pulseout)
srcpulsepause var byte                 'variabile pausa impulso (low side) - velocità
```

Codice File Settaggi:

```
'sub di settaggio iniziale:
srcset:                                'sub di configurazione iniziale di default
srcpin = 0                             'seleziono il pin di default
srcosck = 10                           'default il quarzo da 4Mhz
srcunit = 0                            'default l'unità in microsecondisrcposverse = 0

srcposgrad = 0                         'azzero la variabile per la scala in gradi
srcvelocity = 6                        'imposto una velocità normale (pause = 17ms)
srccycles = 45                         'imposto un numero di impulsi normale
srcposus = 1500                        'inizializzo la posizione centrale (per eventuale riposizionamento)
return
```

Codice File Programma:

```
'Programma di controllo per servi rc, con scala in microsecondi,
'secondo lo standard dei servi (1ms-1,5ms-2ms) e scala in gradi
'impostabile. Fissato il fattore di conversione 1° = 6us.
'Nota bene che la risoluzione cambia con il quarzo usato, quindi sarà
'necessario impostare il coefficiente per il cambio del quarzo per
'mantenere costanti le unità di misura (srcosck = 10 con i 4Mhz
'srcosck = 2 con i 20Mhz).

'sub di posizionamento:
srcsetpos:                             'sub di settaggio posizione
if srcunit = 1 then                    'se l'unità di misura è in gradi
srcposus = (6 * srcposgrad) + 1500     'converto il valore in microsecondi (standard servi rc)
else                                   'se l'unità di misura è in microsecondi
srcposgrad = (srcposus - 1500) / 6     'converto il valore in gradi (per informazione)
endif
srcpulse = srcposus / srcosck          'converto in unità di pulsout (secondo il quarzo impostato)
srcpulsepause = 1 + (10 - srcvelocity) * 4   'calcolo la pausa in base alla velocità scelta
for srccont = 1 to srccycles           'invia gli impulsi srccycles volte
pulsout srcpin,srcpulse                'invia l'impulso di comando (poizione)
pause srcpulsepause                    'invio la pausa
next srccont
return
```

Modulo Sensori Ultrasuoni:

Questo modulo garantisce una lettura semplice del sensore ad ultrasuoni, effettuando una media tra diverse letture e fornendo il valore in centimetri e metri.

Guida al Modulo:

File di Variabili ---> ultravar.bas
File di Setup ---> ultraset.bas
File programma (soubroutines) ---> ultramod.bas

```
ultratriggerpin (variabile pin)          'pin invio impulso Trigger (pin variabile - valore 0-15)
ultraechopin (variabile pin)             'pin ricezione echo sensore (pin variabile - valore 0-15)
ultraosck (variabile)                    'byte coefficiente cambio quarzo (4Mhz --> 10 ; 20Mhz --> 2)
ultraus (variabile)                      'word valore misurato restituito in microsecondi
ultracm (variabile)                      'word valore misurato restituito in centimetri
ultrametint (variabile)                  'byte valore rstituito in metri - parte intera
ultrametdec (variabile)                  'byte valore restituito in metri - parte decimale
ultraoverrange (flag)                    'bit che indica che l'oggetto è fuori portata del sensore
ultranoconnect (flag)                    'bit che indica che il sensore non è connesso (problemi hardware)
ultrasoglia (variabile)                  'word che indica la soglia (in cm) sotto la quale è impostato il flag "ostacolo"
ultraostacolo (flag)                     'bit "ostacolo rilevato" attivato dalla soglia ultrasoglia
ultrameasuresnumber (variabile)          'byte numero di misure consecutive da effettuare (ne verrà fatta la media)
ultradelay (variabile)                   'byte valore (in ms) di attesa tra una misura e l'altra
gosub ultraread                          'sub per effettuare l'operazione di lettura del sensore
gosub ultraset                           'sub di settaggio modulo sensori ad ultrasuoni (default)
```

NOTE MODULO ULTRASUONI:
Questo modulo è compatibile con i sensori SRF04 e SRF05 della Devantech.
Il modulo permette di rilevare ogetti con il sensore ad ultrasuoni e restituirne la distanza un diverse unità di misura.
Il valore è restituito in microsecondi (per uso avanzato), in centimetri ed in metri (parte intera e parte decimale separate).
Il sensore richiede due pin, uno per l'invio dell'impulso di inizio misura e uno per la lettura dell'impulso echo.
Questi pin sono variabili utilizzando il metodo con i numero da 0 a 15.
La variabile ultraosck serve per il cambio di quarzo, e deve essere impostata con valore relativo al quarzo usato.
Se il sensore non rileva alcun oggetto, perchè l'oggetto è troppo lontano o troppo vicino (meno di 1cm) il valore restituito sarà 0 (cm o metri) e verrà attivato il flag "overrange". In caso non vengano ricevuti impulsi dal sensore, significa che ci sono problemi hardware, percui verrà impostato il flag "noconnect".
Per modificare la soglia di overrange e le altre impostazioni consultare il datasheet del sensore ed il file programma prima di agire sul file di setup. Le configurazioni di default sono adatte ai due modelli specificati in precedenza.
la variabile ultrameasuresnumber serve per scegliere il numero di misure consecutive che verranno fatte fare al sensore. Al termine verrà eseguita la media tra tutte le misure. Di default il numero di misure è 2, per non rallentare troppo il programma.
con la variabile ultradelay è possibile impostare l'attesa tra una misura e l'altra (necessaria per una misura corretta). Comunque è consigliabile non modificare l'impostazione di default (10ms) per evitare di ottenere misure non corrette e/o una lentezza nell'esecuzione delle misure.
Una funzione aggiuntiva del modulo ultrasuoni consiste nella soglia ostacolo:
si può caricare un valore in centimetri nella variabile ultrasoglia. Quando il valore misurato è inferiore alla soglia verrà attivato il flag corrispondente (ultraostacolo). Questo è utile per la rilevazione di ostacoli semplificata

(c'è - non c'è).
L'unità di misura su cui si basa il modulo, cioè quella principale, è il centimetro. La misura in metri è stata introdotta soprattutto per il debug su terminali (o per avere una discriminazione iniziale sula distanza).
Il valore il microsecondi non è soggetto alle regole applicate ai centimetri ed ai metri, è presente solo per usi avanzati, per esempio per complicati calcoli diagnostici, si consiglia di non fare riferimento a tale valore per la rilevazione ed il calcolo della ditanza di oggetti.
Per dettagli consultare i sorgenti del modulo.

Codice File Variabili:

```
'Modulo Variabili


'Variabili:
ultratriggerpin var byte                'pin impulso Trigger (inizio rilevazione)
ultraechopin var byte                   'pin valore Echo del sensore (distanza)
ultraosck var byte                      'variabile coefficiente cambio di quarzo (4Mhz->10 ; 20Mhz->2)
ultraus var word                        'valore letto in microsecondi
ultracm var word                        'valore convertito in centimetri
ultrametint var byte                    'valore convertito in metri (parte intera)
ultrametdec var byte                    'valore convertito in metri (parte decimale)
ultratriggerus var byte                 'valore in microsecondi dell'impulso di trigger
ultratriggerpulse var byte              'valore (in unità pulsout) dell'impulso di trigger
ultraechopulse var word                 'valore restituito dal sensore (in unità di pulsin)
ultraoverrangeset var word              'valore (in us) oltre il quale non è rilevato alcun oggetto
ultraerrors var byte                    'byte flag di errore e segnalazioni
ultraoverrange var ultraerrors.0            'flag di "over range" della misura (oggetto fuori portata)
ultraminus var byte                         'valore in microsecondi minimo che il sensore può fornire
ultranoconnect var ultraerrors.1            'flag di errore che indica che l'hardware (il sensore) non è connesso (1)
ultrasoglia var word                        'soglia in centimetri per l'attivazione del flag "ostacolo"
ultraostacolo var ultraerrors.2             'flag superamento soglia (1 = valore letto < soglia)
ultrameasuresnumber var byte                'numero di misure da fare (verrà fatta la media)
ultracont var byte                          'contatore cilci di misura
ultradelay var byte                         'valore in millisecondi pausa tra una misura e l'altra
ultraustocm con 58                          'fattore di conversione tra us e cm (1cm = 58us)
```

Codice File Settaggi:

```
'Modulo di settaggio


ultraset:                               'sub di settaggio e configurazione iniziale
ultraechopin = 3                        'imposto il pin echo di default
ultratriggerpin = 2                     'imposto il pin trigger di default
ultraus = 0                             'azzero la variabile valore in microsecondi
ultracm = 0                             'azzero la variabile valore in centimetri
ultraoverrange = 0                      'azzero il flag "over range"
```

```
ultranoconnect = 0                      'azzero il flag "hardware non connesso"
ultraostacolo = 0                       'azzero il flag "ostacolo"
ultrametint = 0                         'azzero la variabile valore in metri (parte intera)
ultrametdec = 0                         'azzero la variabile valore in metri (parte decimale)
ultraosck = 10                          'imposto il coefficiente per il quarzo da 4Mhz
ultratriggerus = 10                     'imposto la durata dell'impulso di trigger
ultraoverrangeset = 30000               'imposto il valore per cui viene dichiarato l'"over range"
ultraminus = 100                        'imposto il valore minimo di microsecondi
ultrasoglia = 50                        'imposto la soglia "ostacolo" a 50cm
ultrameasuresnumber = 2                 'effettuo due misure (per poi farne la media)
ultradelay = 10                         'imposto il tempo tra una lettura e l'altra
return
```

Codice File Programma:

```
'Programma per la lettura della distanza da un sensore ad ultrasuoni.
'Compatibile con i sensori SRF04 e SRF05 Devantech.
'Il programma esegue un numero specificato di letture e ne esegue la media,
'poi converte il valore misurato in differenti unità comprensibili dall'utente.
'Il valore verrà restituito in microsecondi, in centimetri ed in metri (parte intera
'e parte decimale separate).
'La distanza misurata ha un'approsimazione del centimetro e la portata del sensore può arrivare a più
'di 4 metri (a seconda del modello). Se non viene rilevato un oggetto (perchè troppo distante) verrà
'considerata la distanza pari a 0 ed inoltre verrà impostato un flag di "over range".
'La distanza minima misurabile è 1cm (se si ottiene 0 si è in over range).
'Si consiglia di non fare riferimento al valore in microsecondi, destinato solo ad un uso avanzato.
'Sul valore in microsecondi non verranno applicate le regole che vengono applicate al valore in cm.
'Il valore in metri è stato introdotto specialmente per i debug, è un valore che viene restituito a
'partire dal valore in cm.
'è stata introdotta una variabile che funge da soglia (valore in centimetri). Quando i centimetri misurati
'risultano inferiori alla soglia, verrà attivato un flag corrispondente. Questo è utile per il rilevamento di
'ostacoli a raggio fisso o per funzioni di allarme.



ultraread:                              'sub di lettura sensore ad ultrasuoni
ultraechopulse = 0                      'azzero la variabile recipiente impulso in ricezione
ultraus = 0                             'azzero la variabile valore in microsecondi
ultracm = 0                             'azzero la variabile valore in centimetri
ultraoverrange = 0                      'azzero il flag "over range"
ultranoconnect = 0                      'azzero il flag "hardware non connesso"
ultraostacolo = 0                       'azzero il flag "ostacolo"
ultratriggerpulse = ultratriggerus / ultraosck          'converto il valore di trigger in unità di pulsout
for ultracont = 1 to ultrameasuresnumber                'eseguo il numero di misure impostato
        pulsout ultratriggerpin,ultratriggerpulse       'invio l'impulso trigger (interrogo il sensore)
        pulsin ultraechopin,1,ultraechopulse            'ricevo l'impulso (dati in unità di pulsin)
        ultraus = ultraus + ((ultraechopulse * ultraosck) / ultrameasuresnumber)        'effettuo la media tra le misure e converto in
                                                                                        microsecondi (tutto in uno)
        pause ultradelay                                'delay tra una misura e l'altra (per una misura corretta)
```

```
next ultracont                                          'ho finito le misurazioni
ultracm = ultraus / 58                                  'converto il valore da microsecondi a centimetri
if ultracm < 1 then                                     'se il valore misurato è inferiore al centimetro
ultracm = 1                                             'approssimo ad 1 centimetro
endif
if ultraus > ultraoverrangeset or ultraus < ultraminus then     'se il valore letto (in us) è maggiore del limite "over range" o
                                                                 minore del valore minimo
ultracm = 0                                             'imposto a 0 i centimetri restituiti
        if ultraus > ultraoverrangeset then             'in caso sia effettivamente maggiore del limite "over range"
        ultraoverrange = 1                              'imposto ad 1 il flag di "over range"
        else                                            'se invece il valore è minore del valore minimo, alias hardware non collegato
        ultranoconnect = 1                              'imposto a 1 il flag errore "hardware non connesso"
        endif
else                                            'se il valore rientra nel campo di misura
        if ultracm < ultrasoglia then           'se il valore misurato (cm) è inferiore alla soglia impostata
        ultraostacolo = 1                       'attivo il flag "ostacolo"
        endif
endif
ultrametint = ultracm / 100                     'calcolo i metri (dai centimetri) - Parte intera
ultrametdec = ultracm - (ultrametint * 100)     'calcolo la parte decimale (dopo la virgola) dei metri
return                                          'ritorno alla sub chiamante
```

Modulo Sonar:

Usato assieme al modulo servi ed al modulo ultrasuoni ne combina le funzioni per realizzare un sonar: posizionamento servo con successiva lettura del sensore.

Guida al Modulo:

```
MODULO SONAR:
File di Variabili  --->  sonarvar.bas
File programma (soubroutines)  --->  sonarmod.bas

sonardelay (variabile)                    'byte attesa (in millisecondi) tra il comando di posizionamento del servo
                                          'ed il comando di lettura del sensore ad ultrasuoni.
gosub sonarint                            'sub del sonar. Esegue il posizionamento del sensore (servo) e poi la misura

NOTE MODULO SONAR:
Il modulo sonar funziona appoggiandosi al modulo servo r/c ed al modulo per il sensore ad ultrasuoni.
Quindi nel programma primo devono essere inclusi anche tali moduli.
Il modulo sonar usa il modulo per il controllo dei servi r/c per posizionare il sensore, seguendo la sintassi di
posizionamento del servo (come se si pilotasse direttamente il servo).
Quindi è possibile usare sia il posizionamento in gradi che il posizionamento in microsenìcondi.
Dopo aver effettuato il posizionamento si attendono "sonardelay" millisecondi prima di eseguire la lettura del sensore.
Siccome il modulo sonar si appoggia al modulo per la lettura dei sensori ad ultrasuoni, il valore restituito sarà identico
(sintassi) a quello restituito se si usasse il modulo ultrasuoni singolarmente.
Quindi la misura può essere in centimetri o metri e saranno accessibile le funzioni extra quali la soglia ostacolo.
In definitiva questo modulo esegue un'operazione combinata sruttando i moduli servi ed ultrasuoni già creati.
In input viene data la posizione, si chiama la sub sonarint e si prelevano i valori misurati.                    '
```

Codice File Variabili:

```
'Modulo Variabili

'variabili:
sonardelay var byte                       'valore del delay tra posizionamento servo e lettura sensore (millisecondi)
```

Codice File Programma:

```
'Programma Modulo Sonar. Questo programma si appoggia a due moduli già esistenti:
'Il modulo per il comando dei servi e quello per la lettura del sensore ad ultrasuoni
'Nota bene che questi file devono essere inclusi nel programma principale perchè questo
'modulo funzioni. Ovviamente deve essere incluso anche quest'ultimo in ogni sua parte.
'Il programma non fa altro che chiamare la sub di posizionamento del servo e successivamente
'la lettura del sensore ad ultrasuoni. Quindi basta richiamare questo modulo dando come
'input i normali valori di posizionamento del modulo servorc (in gradi o us) e ritirare
'come output il valore del sensore, come si farebbe di solito (valore in us, cm, m, flags etc...).
'L'unica cosa impostabile di questo programma è il delay tra il posizionamento del servo e la lettura
```

'del sensore. Utile per servi lenti!

```
sonarint:                                   'programma sonar
gosub srcsetpos                             'chiamo la sub di posizionamento servo
pause sonardelay                            'eseguo il delay tra il posizionamento e la lettura del sensore
gosub ultraread                             'leggo il sensore
return                                      'torno alla sub chiamante
```

Moduli Comunicazione Radio:

Questi sono dei moduli sperimentali che permettono la trasmissione/ricezione di byte attraverso il protocollo da noi realizzato. I moduli gestiscono gli errori più semplici del protocollo, quindi si sconsiglia il loro uso senza un adeguato sovra-programma di gestione. Per realizzare un link bidirezionale è necessario includere entrambi i moduli per ogni parte. È stato anche realizzato un prototipo di sovra-programma che aumenta le prestazioni del protocollo, funzionante con i moduli ibridi testati in fase di progettazione (utilizza il circuito a transistor). Tale programma non verrà proposto in questo scritto, perché ancora in fase di creazione. I test effettuati finora hanno comunque garantito buoni risultati: nessuna perdita di dati su corte distanze.
Il codice di seguito proposto può essere utilizzato semplicemente, senza sovra-programmi per stabilire link unidirezionali a bassa velocità e su distanze non eccessive.

Guida ai Moduli:

TRASMISSIONE:
File di Variabili ---> travar.bas
File di Setup ---> traset.bas
File programma (soubroutines) ---> trasm.bas

```
gosub trasm                     'per trasmettere
dataout (varibile)              'byte in uscita
gosub traset                    'per configurare la trasmissione
```

RICEZIONE:
File di Variabili ---> ricvar.bas
File di Setup ---> ricset.bas
File programma (soubroutines) ---> ricez.bas

```
gosub ricez                     'per ricevere
datain  (variabile)             'byte in ingresso
gosub ricset                    'per configurare la ricezione
trasmfail:                      'sub eseguita in caso di errore nella ricezione
timeoutcont (variabile opz.)    'valore conteggio timeout (perdita portante) - valore del contatore
timeout (variabile opz.)        'imposto il valore di timeout per la perdita della portante.
errflags (variabile)            'ogni bit corrisponde a un errore riscontrato (debug per trasmfail)
```

Descrizione flags errore (variabile errflags):
bit0 ::: errore impulso lungo (1)
bit1 ::: errore impulso corto (2)
bit2 ::: errore riconoscimento bit (4)

bit0 + bit2 ::: errore bit, impulso lungo (5)
bit0 + bit2 ::: errore bit, impulso corto (6)
bit3 ::: errore di parità (8)
bit4 ::: errore timeout (16)

NOTE TRASMISSIONE E RICEZIONE:
Le basi dei tempi sono impostati nel file di setup. Per la ricezione i valori suno aumentati di solito di 1 o 2 unità rispetto a quelli della trasmissione (per sicurezza). I valori sono espressi in unità di pulsin da 10us.
I moduli utilizzano istruzioni (pulsin/pulsout) variabili con la frequenza del quarzo. per le istruzioni dettagliate su come modificare il moltiplicatore per il cambio quarzo (per i 20Mhz) vedere il file del programma del mudulo di trasmissione ed il file delle variabili del modulo della ricezione.
Questo permette di mantenere l'unità di misura di 1 unità = 10us.
Valori idonei ad una buona trasmissione/ricezione sono:

per la trasmissione:	per la ricezione:
tratimebase = 20	ricthigh = 22
traperiod = 90	rictlow = 42
tratstart = 4	rictstart = 82
tratlow = 2	rictoolow = 2
trathigh = 1	timeout = 1000

Per la gestione errori della ricezione si può usare la sub dedicata oppure inserendo un return in tale sub si potranno gestire gli errori con il metodo "flag di ritorno".

Codice File Variabili Trasmissione:

```
'Variabili ------------------------------------------------------------------------
'

trabyte0 var byte                       'byte contenente tutti i BIT
tratimebase VAR WORD                    'base dei tempi (durata unitaria)
traperiod VAR WORD                      'periodo del bit (impulso + tempo morto)
tratimebloc VAR BYTE                    'moltiplicatore base dei tempi
symbol tpin = PORTB.4                   'pin uscita (trasmettitore)
trasign VAR WORD                        'variabile calcolo durata impulso
traduty VAR WORD                        'variabile calcolo durata tempo morto
tratempbit VAR trabyte0.0               'bit temporaneo (accumulatore)

dataout VAR BYTE                        'byte da inviare
traparity VAR trabyte0.1                'calcolo bit di parità
tradatatosend VAR WORD                  'bits sparati in uscita al trasmettitore
tratstart VAR BYTE                      'moltiplicatore per il bit di start
tratlow VAR BYTE                        'moltiplicatore per il bit basso
trathigh VAR BYTE                       'moltiplicatore per il bit alto
tramoltk VAR BYTE                       'moltiplicatore per il cambio del quarzo
tratondis VAR BYTE                      'moltiplicatore impulso accensione modulo
```

'

'--

Codice File Settaggi Trasmissione:

```
'Inizializzazione


'Main - setup ------------------------------------------------------------------------------------------------------

traset:                                         'Sub di setup
tratimebase = 20                                'setto la base dei tempi
traperiod = 90                                  'setto la lunghezza del periodo (1 bit)
tratstart = 4                                   'setto il moltiplicatore per il bit di start (lunghezza)
tratlow = 2                                     'setto il moltiplicatore per il bit basso (lunghezza)
trathigh = 1                                    'setto il moltiplicatore per il bit alto (lunghezza)
tramoltk = 1                                    'setto il moltiplicatore per la base dei tempi con il quarzo da 4Mhz
'tratondis da abilitare in caso fosse abilitato l'impulso pre-start nel modulo di trasmissione
        'tratondis = 1                          'setto il moltiplicatore per l'impulso di accensione del trasmettitore
Return

'--------------------------------------------------------------------------------------------------------------------
```

Codice File Programma Trasmissione:

```
'--------------------------------------------------------------------------------------------------------------------

'Programma di trasmissione radio di un byte con bit di start e bit di parità.
'Velocità massima teorica 2500 baud, buona velocità 1250 baud.
'sono incluse instruzioni seriali per il debug tramite display o terminale (tabulazioni).
'il codice 13 nel SerOut viene usato per andare a capo nel terminale
'per il display sostituire con il valore "clear screen"
'Ricordarsi che perchè la ricezione funzioni, il segnale trasmesso deve essere negato
'(dopo il ricevitore).
'Il treno di impulsi viene inviato al trasmettitore attraverso il pin "tpin".

'BASE DEI TEMPI:
'Il valore tratimebase varia a seconda del quarzo usato. Attualmente 1 unità equivale a 10us
'il valore della variabile trasign segue questa scala. Ricordarsi di inserire un moltiplicatore
'nella formula in caso di cambio del quarzo (vedi subroutine sendpulse).
'Per il calcolo della variabile traduty si deve convertire la variabile trasign in us, per poter usare
'il valore con l'istruzione PauseUs. Il traperiod e la trasign devono essere
'convertiti, usando un moltiplicatore (10).
'Attualmente la variabile è convertita in us moltiplicando per 10 (quarzo da 4Mhz).
'è stato introdotto un moltiplicatore (la variabile tramoltk) che vale 1 per il quarzo a 4Mhz.
'Per gli altri quarzi il valore va cambiato a seconda della risoluzione del comando pulsout,
'per esempio per i 20Mhz tramoltk vale 5 perchè --> 1 unità (risoluzione 2us) * moltiplicatore 5 = 10
```

```
'se abbiamo 5 unità impostate (50us) --> risultato = 25 (perchè appunto 25*2us = 50us)
'quindi il risultato delle formule non cambia e così facendo l'unità di misura del tempo
'rimane uguale (1 unità = 10us). Per dettagli vedi la sub sendpulse.
'Quarzo da 4Mhz ::: tramoltk = 1
'Quarzo da 20Mhz ::: tramoltk = 5

'-----------------------------------------------------------------------------------------------------------

'Subroutines trasmissione byte ------------------------------------------------------------------------------

trasm:                                                  'sub di trasmissione
GoSub trasmoperations                                   'effettuo i calcoli preliminari...
GoSub sendstart                                         'invio il bit di start
tratempbit = tradatatosend.0                            'invio i bit-dati del byte partendo dall'LSB
GoSub sendbit                                           'chiamo la sub di invio del bit (sia 0 che 1)
tratempbit = tradatatosend.1
GoSub sendbit
tratempbit = tradatatosend.2
GoSub sendbit
tratempbit = tradatatosend.3
GoSub sendbit
tratempbit = tradatatosend.4
GoSub sendbit
tratempbit = tradatatosend.5
GoSub sendbit
tratempbit = tradatatosend.6
GoSub sendbit
tratempbit = tradatatosend.7
GoSub sendbit
tratempbit = tradatatosend.8                            'il bit numero 9 è il bit di parità (vedi calcoli preliminari)
GoSub sendbit                                           'lo invio come tutti gli altri
GoSub sendstop                                          '"invio" lo stop
Return                                                  'torno alla sub chiamante


trasmoperations:                                        'calcoli preliminari
'la trafila qui sotto serve solo per trovare il bit di parità sul byte da trasmettere,
'con il metodo delle exor (^). Se hai dubbi leggiti il libro di elettronica :)
traparity = ((dataout.0 ^ dataout.1) ^ (dataout.2 ^ dataout.3)) ^ ((dataout.4 ^ dataout.5) ^ (dataout.6 ^ dataout.7))
tradatatosend.byte0 = dataout                   'viene caricato la variabile in uscita con il byte da trasmettere
tradatatosend.8 = traparity                     'ma con l'aggiunta del bit di parità (bit numero 9)
Return                                          'torno alla sub chiamante

'-----------------------------------------------------------------------------------------------------------


'Invio singoli bit ------------------------------------------------------------------------------------------

sendstart:                      'invio bit di start
'può essere inviato un impulso precedente allo start (come attivazione modulo) togliendo l'apostrofo del commento
```

```
'alle due istruzioni seguenti. Il moltiplicatore tratondis è modificabile nella configurazione
        'tratimebloc = tratondis                 'imposto il moltiplicatore
        'GoSub sendpulse                        'invio l'impulso
tratimebloc = tratstart                         'imposto il moltiplicatore
GoSub sendpulse                                 'invio l'impulso
Return                                          'torno alla sub chiamante

sendstop:                                       '"invio" lo stop
PauseUs traperiod                               'attento per un tempo lungo come il periodo
Return                                          'torno alla sub chiamante

sendbit:                                        'invio bit-dati
IF tratempbit = 0 Then                          'a seconda che si voglia trasmettere uno 0 o un 1
tratimebloc = tratlow                           'viene impostato il moltipicatore
GoSub sendpulse                                 'ed inviato l'impulso
Else
tratimebloc = trathigh
GoSub sendpulse
EndIF
Return                                          'ritorno alla sub chiamante

'-------------------------------------------------------------------------------------------------


'Invio impulsi ---------------------------------------------------------------------------------


sendpulse:                                      'sub invio impulsi
Low tpin                                        'impostazione per inviare impulsi alti
trasign = tratimebase * tratimebloc * tramoltk  'calcolo la lunghezza dell'impulso da trasmettere a seconda del moltiplicatore
traduty = (traperiod * 10) - (trasign * 10)     'calcolo il valore del traduty cyle (tempo morto) in base al periodo e all'impulso
PulsOut tpin,trasign                            'invio l'impulso
PauseUs traduty                                 'invio il traduty-cycle (tempo morto)
Return                                          'torno alla sub chiamante

'-------------------------------------------------------------------------------------------------
'Fine Programma --------------------------------------------------------------------------------
'-------------------------------------------------------------------------------------------------
'Powered by Ruky87
```

Codice File Variabili Ricezione:

```
'Per il quarzo da 4Mhz ricmoltk deve essere impostato a 1
'Per il quarzo da 20Mhz ricmoltk deve essere impostato a 5
'(fare riferimento al file di setup).

'Descrizione flags errore:
'bit0 ::: errore impulso lungo    (1)
```

```
'bit1 ::: errore impulso corto    (2)
'bit2 ::: errore riconoscimento bit    (4)
'bit0 + bit2 ::: errore bit, impulso lungo    (5)
'bit0 + bit2 ::: errore bit, impulso corto    (6)
'bit3 ::: errore di parità    (8)
'bit4 ::: errore timeout    (16)


DEFINE PULSIN_MAX 500
'definisco sopra la massima attesa per il pulsin


'Variabili ----------------------------------------------------------------
'
ricbyte0 var byte                                   'byte contenente tutti i BIT
riclenght VAR WORD                                  'lunghezza impulso misurato
ricpflag VAR BYTE                                   'flag lettura impulso
symbol rpin = PORTB.4                               'collegamento al ricevitore
ricthigh VAR WORD                                   'tempo soglia bit 1
rictlow VAR WORD                                    'tempo soglia bit 0
rictstart VAR WORD                                  'tempo soglia bit di start
datain VAR BYTE                                     'byte ricevuto
rictempbit VAR ricbyte0.0                           'immagazzinatore temporaneo
ricparityric VAR ricbyte0.1                         'bit di parità ricevuto
ricparity VAR ricbyte0.2                            'bit di parità calcolato
ricdatarea VAR WORD                                 'bit letti (9) in ricezione
rictoolow VAR WORD                                  'soglia impulso troppo corto
errflags VAR BYTE                                   'flag errori ricezione
timeoutcont VAR WORD                                'variabile tempo timeout
timeout VAR WORD                                    'tempo massimo (timeout) no impulso
ricmoltk VAR BYTE                                   'moltiplicatore per cambio quarzo
ricerrocc var ricbyte0.3                            'variabile errore in corso
'
'-------------------------------------------------------------------------
```

Codice File Settaggi Ricezione:

```
'Sub di settaggio iniziale
'modificare i valori decimali per la sincronizzazione

'Main - Setup ------------------------------------------------------------------------------------------

ricset:                                     'sub di setup
ricmoltk = 1                                'imposto il moliplicatore per il cambio di quarzo (vedi variabili)
ricthigh = ricmoltk * 22                    'setto i tempi soglia in unità di pulse (ricordati del quarzo usato)
rictlow = ricmoltk * 42
rictstart = ricmoltk * 82
rictoolow = ricmoltk * 2
timeout = 1000                              'imposto il valore di timeout per la ricezione: perdita portannte
```

```
Return                                          'ritorno alla sub chiamante
```

Codice File Programma Ricezione:

```
'-------------------------------------------------------------------------------------------------------------

'Ricezione radio - velocità massima senza perdita di dati -->  2500 baud
'Il programma riceve sul pin "rpin" e comunica l'esito della trasmissione.
'Ricordarsi che perchè la ricezione funzioni, il segnale trasmesso deve essere negato.
'NOTE:
'il codice 13 nel SerOut viene usato per andare a capo nel terminale
'per il display sostituire con il valore "cler screen"

'-------------------------------------------------------------------------------------------------------------

'Routines di lettura byte ---------------------------------------------------------------------------------------

ricez:                                          'ricezione di un byte
ricerrocc = 0                                   'azzero il flag errore in corso
errflags = 0                                    'vengono azzerati i flags di errore
timeoutcont = 0                                 'azzero la variabile di incremento
GoSub findstart                                 'cerco il bit di start, e se lo trovo vado avanti
GoSub findbit                                   'cerco il primo bit, 0 oppure 1
ricdatarea.0 = rictempbit                       'e lo carico nel suo posto, partendo dall'LSB
GoSub findbit                                   'faccio lo stesso per tutti gli altri bit (sono 9)
ricdatarea.1 = rictempbit
GoSub findbit
ricdatarea.2 = rictempbit
GoSub findbit
ricdatarea.3 = rictempbit
GoSub findbit
ricdatarea.4 = rictempbit
GoSub findbit
ricdatarea.5 = rictempbit
GoSub findbit
ricdatarea.6 = rictempbit
GoSub findbit
ricdatarea.7 = rictempbit
GoSub findbit
ricdatarea.8 = rictempbit                       'leggo il bit numero 9 "alias" il bit di parità
PauseUs 10                                      'piccola pausa di sicronizzazione
GoSub ricezverify                               'vado a verificare la se ho capito bene (circa)
if ricerrocc = 1 then goto trasmfail            'se si è verificato un errore chiamo la sub di gestione errori
Return                                          'torno alla sub chiamante


ricezverify:                                    'sub di verifica byte
if ricerrocc = 1 then goto parityret            'se si è verificato un errore esco direttamente
ricparityric = ricdatarea.8                     'carico il bit di parità ricevuto
```

```
'la trafila qui sotto serve solo per trovare il bit di parità sul byte ricevuto, con il metodo delle exor (^)
'se hai dubbi leggiti il libro di elettronica :)
ricparity = ((ricdatarea.0 ^ ricdatarea.1) ^ (ricdatarea.2 ^ ricdatarea.3)) ^ ((ricdatarea.4 ^ ricdatarea.5) ^ (ricdatarea.6 ^ ricdatarea.7))
IF ricparity <> ricparityric Then                    'adesso confrontiamo se ci sono errori nel byte
datain = 0                                           'se si allora dobbiamo
errflags = 0                                         'azzero tutti i flag prima di impostare quello giusto
errflags.3 = 1                                       'attivo il flag "errore parità"
ricerrocc = 1                                        'attivo il flag errore in corso
goto parityret                                       'esco dalla rutine
Else                                                 'in caso contrario tutto bene, possiamo proseguire
datain = ricdatarea.byte0                            'il byte letto viene reso definitivo
timeoutcont = 0                                      'azzero la variabile di conteggio timeout
errflags = 0                                         'vengono azzerati i flags di errore
EndIF
parityret:                                           'punto di uscita
Return                                               'torno alla sub chiamante

'------------------------------------------------------------------------------------------------------------

'Routines di lettura bits ------------------------------------------------------------------------------------

findstart:                                           'cerco il bit di start
GoSub findpulse                                      'cerco e misuro l'impulso in arrivo
if ricerrocc = 1 then goto findstartret              'se si è verificato un errore esco direttamente
IF ricpflag <> 2 Then GoTo findstart                 'controllo se si tratta di uno "start"
findstartret:                                        'punto di uscita
Return                                               'torno alla sub chiamante


findbit:                                             'cerco i bit-dati
if ricerrocc = 1 then goto findbitret                'se si è verificato un errore esco direttamente
GoSub findpulse                                      'cerco l'impulso
if ricerrocc = 1 then goto findbitret                'se si è verificato un errore esco direttamente
Select Case ricpflag                                 'valuto di che si tratta...
Case 0
rictempbit = 0                                       'si tratta di un bit basso (0)
Case 1
rictempbit = 1                                       'si tratta di un bit alto (1)
Case Else                                            'e se non è ne 1 ne 0, ne vivo ne morto...
errflags.2 = 1                                       'errore riconoscimento bit
ricerrocc = 1                                        'attivo il flag errore in corso
goto findbitret                                      'esco dalla routine
End Select
findbitret:                                          'punto di uscita
Return                                               'torno alla sub chiamante

'------------------------------------------------------------------------------------------------------------


'Routine di Ricerca e Misurazione Impulso ---------------------------------------------------------------------
```

```
findpulse:                          'ricerca impulsi
riclenght = 0                       'azzero le due variabili per precauzione...
ricpflag = 0
PulsIn rpin,0,riclenght             'attento e misuro l'impulso (attivo basso, quindi negato)
Select Case riclenght               'a seconda della lunghezza (durata) faccio una scelta
Case 0                              'se non vi è stato alcun impulso (tempo morto, tempo di inattività)
timeoutcont = timeoutcont + 1       'incrementa la variabile di timeout
IF timeoutcont = timeout Then       'se raggiungo il tempo di timeout (alias, perdo portante)
errflags = 0                        'azzero tutti i flag prima di impostare quello giusto
errflags.4 = 1                      'attivo il flag "portante persa"
ricerrocc = 1                       'attivo il flag errore in corso
goto retpoint                       'esco dalla routine
EndIF
GoTo findpulse                      'mi fermo e continuo a cercare
Case Is < rictoolow                 'se l'impulso è troppo corto (per esempio interferenze)
errflags = 0                        'azzero tutti i flag prima di impostare quello giusto
errflags.1 = 1                      'errore "impulso troppo corto"
ricpflag = 3                        'setto il flag a 3 "alias" qualche problema
GoTo retpoint                       'vado al check-out
Case Is < ricthigh                  'se l'impulso è minore della soglia (1) (e maggiore di rictoolow)
ricpflag = 1                        'il flag viene messo a 1 "alias anche di" bit alto
GoTo retpoint                       'check-out
Case Is < rictlow                   'se è minore della soglia (0) (e maggiore di ricthigh)
ricpflag = 0                        'il flag viene messo a 0 "alias anche di" bit basso
GoTo retpoint                       'check-out
Case Is < rictstart                 'se infine è minore del tempo di start (ma maggiore di rictlow)
ricpflag = 2                        'il flag viene messo a 2 "alias anche di" questo è uno "start"
GoTo retpoint                       'vado al check-out
Case Else                           'in caso l'impulso sia più lungo dello start, c'è qualche problema...
errflags = 0                        'azzero tutti i flag prima di impostare quello giusto
errflags.0 = 1                      'errore "impulso troppo lungo"
ricpflag = 3                        'il flag è messo a 3... Sapete che significa?
End Select
retpoint:                           'arrivati al check-out! Punto di ritorno.
Return                              'torno alla sub chiamante

'-------------------------------------------------------------------------------------------------------
'Fine Programma ----------------------------------------------------------------------------------------
'-------------------------------------------------------------------------------------------------------
'Powered by Ruky87
```

PROGRAMMI PRINCIPALI:

Di seguito vengono proposti i programmi principali caricati nei pic. Ogni programma principale è costituito dal sorgente principale è dai vari moduli inclusi e configurati. Esso è vincolato al modello di PIC adottato, quindi questi programmi contengono la struttura di funzionamento dei vari microcontrollori presenti nei circuiti.
Di seguito verranno proposti i sorgenti per: il pic core (gondola), il pic fuffy (gondola), il picmaster (base), il pic del joypad (base), il pic gestore delle comunicazioni (base), ed il pic ciclopè (telecamera).
I sorgenti sono commentati ma si consiglia di fare uso del manuale del linguaggio PicBasic.
Per una migliore lettura dei programmi principali, essi sono stampati direttamente dall'editor, per mantenerne intatto il colore e le tabulazioni (file PDF). Infine i programmi vengono presentati separatamente, come capitoli a se stanti per organizzare meglio le numerose pagine di sorgente.

PROGRAMMA CORE

```
'Blocco Direttive:
DEFINE ADC_BITS 10              'numero di bit usati dall'adc
DEFINE ADC_CLOCK 3              'frequenza usata dall'adc (3 = internal rc)
DEFINE ADC_SAMPLEUS 50          'tempo campionamento usato dall'adc
DEFINE I2C_SLOW 1               'rallento la velocità del bus i2c
DEFINE SHIFT_PAUSEUS 50         'rallento la velocità della seriale sincrona


'Blocco Memoria Programma Principale:
'----> Connessioni
SYMBOL rts = portb.1                    'segnale rts (host to easy radio) - Pic Pronto a rivevere - Output (va
tenuto sempre basso)
SYMBOL rxtxselect = portb.1             'segnale di controllo - selezione tx/rx moduli ibridi
SYMBOL cts = portb.4                    'segnale cts (modulo easy radio to host) - Modulo Pronto a Ricevere -
Input (1=modulo occupato)
SYMBOL txpin = portb.2                  'segnale dato in uscita (verso il cielo)
SYMBOL rxpin = portb.3                  'segnale dato in ingresso (dal cielo)
SYMBOL ledalpha = portb.0               'led di segnalazione alpha (A)
SYMBOL infraant = portd.7               'sensore infrarosso soffitto - lato anteriore
SYMBOL infrapos = portd.6               'sensore infrarosso soffitto - lato posteriore
SYMBOL direzmot2 = portd.5              'direzone rotazione motore 2 (sinistro da dietro)
SYMBOL direzmot1 = portd.4              'direzione rotazione motore 1 (destro da dietro)
SYMBOL debugrx = portc.7                'seriale (UART) per il debug (PC) - Dati dal Terminale
SYMBOL debugtx = portc.6                'seriale (UART) per il debug (PC) - Dati verso il Terminale
SYMBOL esp0dig0 = portd.3               'canale digitale 0 - espansione 0 (ciclopè)
SYMBOL esp0dig1 = portd.2               'canale digitale 1 - espansione 0 (ciclopè)
SYMBOL esp0an0pin = porte.2             'pin analogico 0 (CH7) - espansione 0 (ciclopè)
esp0an0 CON 7                           'canale analogico corrispondente al pin esp0an0pin (espansione 0 -
analogico 0)
SYMBOL esp1dig0 = portd.1               'canale digitale 0 - espansione 1 (EXT)
SYMBOL esp1an0pin = porte.1             'pin analogico 0 (CH6) - espansione 1 (EXT)
esp1an0 CON 6                           'canale analogico corrispondente al pin esp1an0pin (espansione 1 -
analogico 0)
SYMBOL esp2dig0 = portd.0               'canale digitale 0 - espansione 2 (EXT)
SYMBOL esp2an0pin = porte.0             'pin analogico 0 (CH5) - espansione 2 (EXT)
esp2an0 CON 5                           'canale analogico corrispondente al pin esp2an0pin (espansione 2 -
anaogico 0)
SYMBOL esp3dig0 = portc.0               'canale digitale 0 - espansione 3 (INT)
SYMBOL esp3an0pin = porta.5             'pin analogico 0 (CH4) - espansione 3 (INT)
esp3an0 CON 4                           'canale analogico corrispondente al pin esp3an0pin (espansione 3 -
analogico 0)
SYMBOL ledbeta = porta.4                'led di segnalazione beta (B)
SYMBOL ledgamma = porta.3               'Led di segnalazione gamma (C) - Attenzione: questo pin è ANALOGICO (vedi
ADCON1)
SYMBOL vbattpartpin = porta.2           'pin analogico (CH2) lettura tensione totale pacco batterie diviso (1/2
Vbatt)
vbattpart CON 2                         'canale analogico corrispondente al pin vbattpartpin (tensione partitore)
SYMBOL vbattcell2pin = porta.1          'pin analogico (CH1) lettura tensione cella 2 pacco batterie (Vcell2)
vbattcell2 CON 1                        'canale analogico corrispondente al pin vbattcell2pin (tensione cella 2)
SYMBOL rssipin = porta.0                'pin analogico (CH0) segnale rssi moduli radio (link analog out)
rssi CON 0                              'canale analogico corrispondente al pin rssipin (valore rssi modulo
radio)
'---> Costanti di Programma
minvelocity CON 100                     'valore di velocità (pwm) sotto al quale i motori sono considerati fermi
pwmregenable CON 0                      'indica se attivare la modalità regolazione velocità motori con pwm
anticollisiondelay CON 200              'valore in microsecondi tra un accesso e l'altro della memoria
pausetrx CON 50                         'pausa in millisecondi tra una trasmissione ed una ricezione radio
ramaddr CON %10100000                   'indirizzo accesso alla memoria ram
compassaddr CON %11000000               'indirizzo accesso alla bussola digitale
compassoffset CON 5                     'offset per dare un errore di lettura sulla bussola digitale
compassfact CON 182                     'fattore di conversione per convertire il valore in gradi della bussola
in un byte
                                        '1 unità = 1,408°  -  valore da usare con l'operatore */  - per ritornare
in gradi
                                        'usare dall'altra parte il valore 361 sempre con l'operatore */
vfs CON 500                             'tensione di fondoscala in centivolts (riferimento adc)
compreamble1 CON %10101010              'Preambolo per le trasmissioni seriali / radio (primo)
compreamble2 CON %01010101              'Preambolo per le trasmissioni seriali / radio (secondo - complementare)
compostamble CON %10101010              'Postambolo per le trasmissioni seriali / radio
hybrid CON 0                            'impostazione moduli radio (se 1 significa che uso i moduli ibridi)
ctsatt CON 0                            'livello logico perchè cts sia attivo (attivo basso quindi)
rtsatt CON 0                            'livello logico perchè rts sia attivo (attivo basso quindi)
sermode CON 396                         'impostazione velocità di comunicazione serin2/serout2 (moduli easy radio)
'---> Variabili generiche di programma
tempcalc VAR WORD               'variabile temporanea per il calcolo con le variabili
compasstemp VAR tempcalc        'variabile per l'elaborazione del valore della bussola durante la lettura del
sensore
vbatttemp VAR tempcalc          'variabile per la lettura e compressione dei valori delle tensioni (per invio via
radio)
dataintemp VAR tempcalc.Byte0   'variabile temporanea lettura dati in ingresso


'Symbol-Alias per usi particolari (connessioni particolari):
SYMBOL cameratx = portd.3               'corrisponde a esp0dig0 --> canale seriale per pilotaggio modulo camera
(ciclopè)
SYMBOL cameracts = portd.2              'corrisponde a esp0dig1 --> canale cts modulo ciclopè (camera) - Ciclopè
to Core - attivo alto



'---> Pacchetti Radio in Ingresso
basestatep VAR basestatebyte            'dato byte di controllo base
joypadkeysp VAR joypadbuttons           'dato pulsanti joypad
```

```
joyanddirezp VAR joyanddirez            'croce joypad e direzione motori
motdxpwmp VAR pwmmotdx                  'dato velocità motore destro in ingresso
motsxpwmp VAR pwmmotsx                  'dato velocità motore sinistro in ingresso
motdownpwmp VAR pwmmotdown              'dato velocità motori ascensionali in ingresso


'---> Variabili Pacchetti Radio in Uscita
corebyte VAR coreword.Byte0           'Parte Bassa Word di Stato Core (Viene inviata via Radio)
fuffy1cenp VAR fuffy1cmcen            'dato fuffy1 centro
fuffy2cenp VAR fuffy2cmcen            'dato fuffy2 centro
fuffy1dxp VAR fuffy1cmdx              'dato fuffy1 destra
fuffy2dxp VAR fuffy2cmdx              'dato fuffy2 destra
fuffy1sxp VAR fuffy1cmsx              'dato fuffy1 sinistra
fuffy2sxp VAR fuffy2cmsx              'dato fuffy2 sinistra
fuffy3p VAR fuffy3cm                  'dato fuffy3 sotto
compassp VAR compassvalue.Byte0       'valore compresso della bussola digitale
sensorsp VAR sensors                  'valori dei sensori (infrarossi) e altro
vbattp VAR battcell1                  'livelli batterie pacchettati


'---> Variabili Dati in ingresso radio (prima dell'elaborazione) - Valori Spachettati
basestatebyte VAR BYTE                      'byte flags di stato base e controllo
       '---> Alias Bit Stato
       basestate VAR basestatebyte.0        'Flag di stato della base terrestre


joypadbuttons VAR BYTE                  'pulsanti joypad funzioni extra (vedi alias per descrizione dettagliata)
joyanddirez VAR BYTE                    'byte croce joypad e direzione motori
pwmmotdx VAR BYTE                       'valore velocità motore destro (visto da dietro la gondola)
pwmmotsx VAR BYTE                       'valore velocità motore sinistro (visto da dietro la gondola)
pwmmotdown VAR BYTE                     'valore velocità motori ascensionali
           '---> Alias
           keyup VAR joyanddirez.0              'bit tasto freccia in su
           keyright VAR joyanddirez.1           'bit tasto freccia a destra
           keydown VAR joyanddirez.2            'bit tasto freccia in basso
           keyleft VAR joyanddirez.3            'bit tasto freccia a sinistra
           keycroce VAR joypadbuttons.0         'bit tasto Croce
           keyquadrato VAR joypadbuttons.1      'bit tasto Quadrato
           keytriangolo VAR joypadbuttons.2     'bit tasto Triangolo
           keycerchio VAR joypadbuttons.3       'bit tasto Cerchio
           keyStart VAR joypadbuttons.4         'bit tasto Start
           keySelect VAR joypadbuttons.5        'bit tasto Select
           keyR1 VAR joypadbuttons.6            'bit tasto R1
           keyR2 VAR joypadbuttons.7            'bit tasto R2
           direzmotdx VAR joyanddirez.4         'direzione motore destro (motore 1)
           direzmotsx VAR joyanddirez.5         'direzione motore sinistro (motore 2)
           direzmotdown VAR joyanddirez.6       'direzione motori ascensionali (sotto)
                    '---> Alias Tasti Funzione
                    camerayup VAR keyup                 'motore asse y telecamera direzione alto
                    cameraxright VAR keyright           'motore asse x telecamera direzione destra
                    cameraydown VAR keydown             'motore asse y telecamera direzione basso
                    cameraxleft VAR keyleft             'motore asse x telecamera direzione sinistra
                    cameraacc VAR keyR1                 'accensione telecamera
                    laseracc VAR keyR2                  'accensione puntatore laser


'---> Variabili Modulo Fuffy
'I valori misurati sono in centimetri/2 quindi per avere il valore corretto in centimetri moltiplicare
(mentalmente)
'il valore delle variabili per 2 (risoluzione 2cm)
fuffy1cmcen VAR BYTE                    'buffer valore misurato fuffy 1 - posizione centrale anteriore
fuffy2cmcen VAR BYTE                    'buffer valore misurato fuffy 2 - posizione centrale posteriore
fuffy1cmdx VAR BYTE                     'buffer valore misurato fuffy 1 - posizione destra anteriore
fuffy2cmdx VAR BYTE                     'buffer valore misurato fuffy 2 - posizione destra posteriore
fuffy1cmsx VAR BYTE                     'buffer valore misurato fuffy 1 - posizione sinistra anteriore
fuffy2cmsx VAR BYTE                     'buffer valore misurato fuffy 2 - posizione sinistra posteriore
fuffy3cm VAR BYTE                       'buffer valore misurato fuffy 3 - sotto
'NB: Le direzioni sono riferite alla struttura vista da dietro.
controlbyte VAR BYTE                    'registro byte di controllo fuffy
        '---> Aliases
        corestate VAR controlbyte.0             'indica se il pic core è senziente (1)
        direzupdown VAR controlbyte.1           'direzione motori ascensionali fuffy

statebyte VAR BYTE                      'registro byte di stato fuffy
          'Aliases
          fuffyerr VAR statebyte.0              'bit di segnalazione errore fuffys
          fuffymodstate VAR statebyte.1         'indica se il pic fuffy ha inizializzato le prime misure

'---> Variabili Dati Sensori
sensors VAR BYTE                      'misure e stato altri sensori (infrared)
           '---> Alias
           infra1 VAR sensors.0         'valore infrarosso anteriore (infraant)
           infra2 VAR sensors.1         'valore infrarosso posteriore (infrapos)
compassvalue VAR WORD                       'valore letto dalla bussola elettronica (gradi)
             '---> Alias
             compassbyte0 VAR compassvalue.Byte0        'parte bassa del valore della bussola
             compassbyte1 VAR compassvalue.Byte1        'parte alta del valore della bussola


'---> Variabili e registri dati elaborati:
ciclopecontrolbyte VAR BYTE                     'byte di controllo modulo ciclopè
```

```
coreword VAR WORD                           'word flags di stato core (solo la parte bassa viene inviata alla
base)
        '--> Aliases
        corewordbyte0 VAR coreword.Byte0        'Parte Bassa word di stato Core
        corewordbyte1 VAR coreword.Byte1        'Parte Alta word di stato Core
                      '--> Alias Bits Stato
                      coreon VAR coreword.0                 'Flag che indica alla base che il core è attivo

                      coreinit VAR coreword.1               'Flag che indica alla base che il core è in stato
di inizializzazione
                      corericezerror VAR coreword.2         'Flag che indica che si è verificato un errore di
ricezione dato via radio
                      ciclopeunavable VAR coreword.3        'Flag che indica che il modulo ciclopè non è
disponibile al momento
                      positionflag VAR coreword.8           'Flag che indica se il programma si trova prima o
dopo l'autonav (rielaborazione)


'---> Misure Analogiche
rssiword VAR WORD                           'valore rssi elaborato
        '---> Aliases
        rssibyte0 VAR rssiword.Byte0        'parte bassa valore elaborato rssi
        rssibyte1 VAR rssiword.Byte1        'parte alta valore elaborato rssi
battcell1 VAR BYTE                          'valore elaborato tensione cella 1
battcell2 VAR BYTE                          'valore elaborato tensione cella 2

'---> Variabili Extra (o per espanioni)
exbyte0 VAR BYTE            'variabile extra 0
exbyte1 VAR BYTE            'variabile extra 1
exbyte2 VAR BYTE            'variabile extra 2
exbyte3 VAR BYTE            'variabile extra 3


'Costanti e variabili locazioni configurazioni (registri memoria) in lettura e scrittura:
'--> In scrittura --- Dati elaborati e non (vedi bit posizione):
controlloc CON 1                            'indirizzo locazione byte di controllo fuffy
pwmupdownloc CON 2                          'indirizzo locazione byte velocità motori ascensionali fuffy
pwmmotdxloc CON 3                           'indirizzo locazione byte velocità motore destro
pwmmotsxloc CON 4                           'indirizzo locazione byte velocità motore sinistro
joyanddirezloc CON 5                        'indirizzo locazione byte pulsanti joypad croce e direzioni motori
joypadbuttonsloc CON 6                      'indirizzo locazione byte pulsanti joypad letti
basestateloc CON 7                          'indirizzo locazione byte flags di stato base
ciclopecontrolloc CON 8                     'indirizzo locazione byte di controllo e dati modulo ciclopè
rssibyte0loc CON 9                          'indirizzo locazione byte parte bassa valore rssi
rssibyte1loc CON 10                         'indirizzo locazione byte parte alta valore rssi
battcell1loc CON 11                         'indirizzo locazione byte parte bassa valore cella 1
battcell2loc CON 12                         'indirizzo locazione byte parte bassa valore cella 2
sensorsloc CON 13                           'indirizzo locazione byte valori misurati dai sensori (infrarossi)
etc.
compassbyte0loc CON 14                      'indirizzo locazione byte parte bassa valore misurato dalla bussola
compassbyte1loc CON 15                      'indirizzo locazione byte parte alta valore misurato dalla bussola
corewordbyte0loc CON 16                     'indirizzo locazione byte parte bassa word di stato core (inviata via
radio)
corewordbyte1loc CON 17                     'indirizzo locazione byte parte alta word di stato core (interna)


'--> In lettura:
fuffy1cenloc CON 126                        'indirizzo locazione byte fuffy 1 - posizione centrale
fuffy2cenloc CON 127                        'indirizzo locazione byte fuffy 2 - posizione centrale
fuffy1dxloc CON 128                         'indirizzo locazione byte fuffy 1 - posizione destra
fuffy2dxloc CON 129                         'indirizzo locazione byte fuffy 2 - posizione destra
fuffy1sxloc CON 130                         'indirizzo locazione byte fuffy 1 - posizione sinistra
fuffy2sxloc CON 131                         'indirizzo locazione byte fuffy 2 - posizione sinistra
fuffy3loc CON 132                           'indirizzo locazione byte fuffy 3
stateloc CON 133                            'indirizzo locazione byte di stato fuffy


'Blocco Memoria Moduli:
INCLUDE "i2cvar.bas"
INCLUDE "mpwmvar.bas"



'Blocco Main - Configs:

main:
CLEAR                                   'azzera tutte le variabili
OUTPUT rts                              'rts è un output
IF hybrid = 0 THEN                      'se uso i moduli easyradio
OUTPUT rts                              'rts è un output
rts = rtsatt                            'abbasso il segnale rts (pic sempre pronto)
ELSE                                    'se uso i moduli ibridi
OUTPUT rxtxselect                       'rxtxselect è un output (lo stesso pin di radiorts)
ENDIF
INPUT cts                               'cts è un input
OUTPUT txpin                            'txpin è un output
INPUT rxpin                             'rxpin è un input
OUTPUT ledalpha                         'ledalpha è un output
INPUT infraant                          'infraant è un input
INPUT infrapos                          'infrapos è un output
OUTPUT direzmot2                        'direzmot2 è un output
LOW direzmot2                           'direzmot2 livello basso
OUTPUT direzmot1                        'direzmot1 è un output
LOW direzmot1                           'direzmot1 livello basso
```

```
INPUT debugrx                           'debugrx è un input
OUTPUT debugtx                          'debugtx è un output
OUTPUT ledbeta                          'ledbeta è un output
mpwmpr2 = 100                           'imposto pr2 a 100 con prescaler unitario
mpwmpresc = 0                           'frequenza ottenuta F=10KHz
mpwmduty = 0                            'azzero il duty-cycle
monoffpwm = 0                           'modulo pwm spento
GOSUB mpwm1                             'setto il pwm (pwm spento) - canale 1 - motore destro (da dietro)
GOSUB mpwm2                             'setto il pwm (pwm spento) - canale 2 - motore sinistro (da dietro)
GOSUB i2cset                            'setto le impostazioni di default del modulo i2cù
GOTO hardset                            'vado alla sub di settaggio ed inizializzazione dell'hardware


'Blocco Settaggi Moduli:
INCLUDE "i2cset.bas"


'Blocco Settaggio e Test Hardware:

hardset:                                'sub di impostazione e controllo hardware
PAUSE 1000                              'pausa accensione
GOSUB i2cmemorychk                      'test accesso alla memoria
IF i2cmemorychkflag = 1 THEN            'se il check memoria è fallito
GOTO hardset                            'continuo a verificare l'accesso alla memoria
ENDIF
GOSUB writeconfig                       'scrivo zeri nelle locazioni utilizzate
GOTO primaryinit                        'vado all'inizializzazione principale


'Blocco Inizializzazione Core:

'---> Inizializzazione Primaria (Setup Componenti)
primaryinit:                            'sub di inizializzazione Core
GOSUB readradio                         'ricevo dati
IF basestate = 1 THEN                   'se la base ha dato la conferma (base attiva)
coreinit = 1                            'setto lo stato del core come fase di inizializzazione
ELSE                                    'in caso non vi sia conferma (base non attiva)
GOTO primaryinit                        'eseguo nuovamente l'inizializzazione (ricerca base)
ENDIF
PAUSE pausetrx                          'pausa tra operazioni radio
GOSUB writeradio                        'invio dati
GOSUB unpackdata                        'spacchetto dati in ingresso
GOSUB readsense                         'leggo sensori locali
corestate = 1                           'indico che il core è attivo
GOSUB writeconfig                       'scrivo in ram
      fuffywait:                        'attesa che fuffy sia pronto
PAUSE 100                               'pausa per evitare che la ram (ed il bus) sclerino
GOSUB readconfig                        'legge la ram
IF fuffymodstate = 1 THEN               'se il modulo fuffy è pronto
GOTO secondaryinit                      'vado all'inizializzazione secondaria
ELSE                                    'altrimenti
GOTO fuffywait                          'attendo che fuffy sia pronto
ENDIF


'---> Sub per funzioni di inizializzazione successive alla dichiarazione dello stato attivo dei Componenti
secondaryinit:                          'sub di inizializzazione secondaria
'niente per ora
coreinit = 0                            'disattivo il flag "in stato di inizializzazione"
coreon = 1                              'indico che il pic core è attivo e in esecuzione
GOTO aria                               'vado al ciclo di programma


'Blocco Principale - Cicli di Sistema:

'---> Ciclo di comunicazione
aria:                                   'sub principale - gestione ciclo programma
GOSUB packdata                          'comprimo i dati in uscita
PAUSE pausetrx                          'pausa tra operazioni radio
GOSUB writeradio                        'invio i dati
GOSUB readradio                         'leggo i dati
GOSUB unpackdata                        'spacchetto i dati in ingresso
GOSUB readconfig                        'leggo la ram
GOSUB readsense                         'leggo i sensori locali
GOSUB senseoperations                   'eseguo ulteriori operazioni sui dati letti dai sensori
GOTO dataelabpre                        'vado all'elaborazione pre-autonav

'---> Ciclo di elaborazione pre-autonav
dataelabpre:                            'elabora i dati prima dell'autonav e scrive in ram
direzupdown = direzmotdown              'imposto la direzione dei motori ascensionali in base ai dati ricevuti
positionflag = 0                        'indico la posizione pre-autonav
corestate = 1                           'indico che il core è attivo
GOSUB writeconfig                       'scrivo la ram
GOTO autonav                            'vado al cuore logico del sistema

'---> Ciclo di elaborazione post-autonav
dataelabpost:                           'sub di elaborazione post-autonav
positionflag = 1                        'indico la posizione post-autonav
corestate = 1                           'indico che il core è attivo
GOSUB writeconfig                       'scrivo in ram
GOTO attuazione                         'attuo i cambiamenti
```

```
'Blocco AUTONAV - Il Centro Di Controllo

autonav:                                'cuore logico dei processi di sistema. Gestisce tutto il funzionamento di
ogni cosa
'passo i dati direttamente in uscita
GOTO dataelabpost                       'elaboro i dati dopo l'autonav e scrivo in ram i valori modificati
dall'autonavigazione



'Blocco Attuazione:

attuazione:                             'sub di gestione dei comandi diretti all'hardware (eseguo i comandi)
IF pwmregenable = 1 THEN                'se è attiva la modalità regolazione velocità motore con pwm
    'Decido se accendere o spegnere il modulo pwm (risparmio energia)
    IF pwmmotdx < minvelocity THEN          'se le velocità del motore destro è pressochè nulla
    pwmmotdx = 0                            'imposto la velocità a 0
    ENDIF
    IF pwmmotsx < minvelocity THEN          'se le velocità del motore sinistro è pressochè nulla
    pwmmotsx = 0                            'imposto la velocità a 0
    ENDIF
    IF pwmmotdx = 0 AND pwmmotsx = 0 THEN   'se entrambi i motori sono spenti
    monoffpwm = 0                           'spengo il modulo pwm
    ELSE                                    'altrimenti
    monoffpwm = 1                           'accendo il modulo pwm
    ENDIF
    'Imposto il motore destro (PWM)
    mpwmduty = pwmmotdx             'imposto il valore del duty-cycle in base alla velocità
    IF direzmotdx <> direzmot1 THEN         'se il verso di rotazione è cambiato
       monoffpwm = 0                        'spengo il pwm
       GOSUB mpwm1                          'imposto il modulo pwm (canale 1)
       PAUSE 10                             'piccola pausa per lasciare che il motore inverta il senso di
rotazione
       direzmot1 = direzmotdx               'modifico la direzione di rotazione dei motori
       monoffpwm = 1                        'accendo il pwm
    ENDIF
    GOSUB mpwm1                             'imposto il modulo pwm (canale 1)
    'Imposto il motore sinistro (PWM)
    mpwmduty = pwmmotsx             'imposto il valore del duty-cycle in base alla velocità
    IF direzmotsx <> direzmot2 THEN         'se il verso di rotazione è cambiato
       monoffpwm = 0                        'spengo il pwm
       GOSUB mpwm2                          'imposto il modulo pwm (canale 2)
       PAUSE 10                             'piccola pausa per lasciare che il motore inverta il senso di
rotazione
       direzmot2 = direzmotsx               'modifico la direzione di rotazione dei motori
       monoffpwm = 1                        'accendo il pwm
    ENDIF
    GOSUB mpwm2                             'imposto il modulo pwm (canale 2)
ELSE                                    'altrimenti non effettuo regolazioni in velocità (sempre velocità massima)
    'Imposto il motore destro
    monoffpwm = 0                           'spengo il pwm
    GOSUB mpwm1                             'imposto il modulo pwm
    OUTPUT mpwmpin1                         'imposto il pin di controllo motori come output
    IF direzmotdx <> direzmot1 THEN         'se il verso di rotazione è cambiato
       mpwmpin1 = 0                             'spengo il motore
       PAUSE 10                                 'piccola pausa per lasciare che il motore inverta il senso di
rotazione
       direzmot1 = direzmotdx                   'modifico la direzione di rotazione dei motori
       mpwmpin1 = 1                             'accendo il motore
    ENDIF
    IF pwmmotdx < minvelocity THEN          'se la velocità è pressochè nulla
    mpwmpin1 = 0                            'spengo il motore
    ELSE                                    'altrimenti
    mpwmpin1 = 1                            'accendo il motore
    ENDIF
    'Imposto il motore sinistro
    monoffpwm = 0                           'spengo il pwm
    GOSUB mpwm2                             'imposto il modulo pwm
    OUTPUT mpwmpin2                         'imposto il pin di controllo motori come output
    IF direzmotsx <> direzmot2 THEN         'se il verso di rotazione è cambiato
       mpwmpin2 = 0                             'spengo il motore
       PAUSE 10                                 'piccola pausa per lasciare che il motore inverta il senso di
rotazione
       direzmot2 = direzmotsx                   'modifico la direzione di rotazione dei motori
       mpwmpin2 = 1                             'accendo il motore
    ENDIF
    IF pwmmotsx < minvelocity THEN          'se la velocità è pressochè nulla
    mpwmpin2 = 0                            'spengo il motore
    ELSE                                    'altrimenti
    mpwmpin2 = 1                            'accendo il motore
    ENDIF
ENDIF
'Imposto il modulo ciclopè
GOSUB ciclopeset                        'imposto il modulo ciclopè
GOTO aria


ciclopeset:                             'impostazione modulo ciclopè
ciclopeunavable = 0             'indico che il modulo ciclope è disponibile a ricevere dati
'assegno alla variabile in uscita i valori dei bit di comando
ciclopecontrolbyte.0 = camerayup           'direzione motore y - su
ciclopecontrolbyte.1 = cameraxright        'direzione motore x - destra
ciclopecontrolbyte.2 = cameraydown         'direzione motore y - giù
```

```
ciclopecontrolbyte.3 = cameraxleft          'direzione motore x - sinistra
ciclopecontrolbyte.4 = cameraacc            'on/off telecamera
ciclopecontrolbyte.5 = laseracc             'on/off laser
ciclopecontrolbyte.6 = 0                    'non usato
ciclopecontrolbyte.7 = 0                    'non usato
IF cameracts = 0 THEN           'se il modulo ciclopè non è attivo oppure è occupato
ciclopeunavable = 1             'indico che il modulo ciclope non è attualmente disponibile
RETURN                          'esco dalla sub
ENDIF
SEROUT2 cameratx,sermode,[ciclopecontrolbyte]       'invia il byte di comando al modulo ciclopè
RETURN


segnalazione:                           'sub di segnalazione (leds ecc)
'niente per ora
RETURN


'Blocco lettura sensori e hardware:

'---> routines di gestione letture
readsense:                              'subroutine lettura sensori locali
GOSUB readcompass                       'legge la bussola digitale
GOSUB readinfra                         'legge gli infrarossi
GOSUB readanalog                        'legge gli adc utilizzati (tranne rssi), ad esempio le tensioni
RETURN

senseoperations:                        'esegue operazioni ci conversione eccetera sui sensori
'niente per ora
RETURN


'---> Routines di lettura hardware
readcompass:                            'subroutine lettura bussola digitale
compasstemp = 0                         'azzero la variabile temporanea
i2cdelay = 1000                         'delay = 1ms
i2caddrbw = 0                           'indirizzo dispositivo slave = byte
i2ccontrol = compassaddr                'codice di controllo bussola elettroica
i2caddr = 2                             'imposto l'indirizzo della parte alta del valore
GOSUB i2crx                             'leggo dalla bussola
compasstemp.Byte1 = i2cin               'scrivo nel registro temporaneo il valore della parte alta
i2caddr = 3                             'imposto l'indirizzo della parte bassa del valore
GOSUB i2crx                             'leggo dalla bussola
compasstemp.Byte0 = i2cin               'scrivo nel registro temporaneo il valore della parte bassa
compassvalue = compasstemp / 10         'scrivo il valore effettivo approssimato al grado
compasstemp = 0                         'azzero la variabile temporanea
i2caddr = 2                             'imposto l'indirizzo della parte alta del valore
GOSUB i2crx                             'leggo dalla bussola
compasstemp.Byte1 = i2cin               'scrivo nel registro temporaneo il valore della parte alta
i2caddr = 3                             'imposto l'indirizzo della parte bassa del valore
GOSUB i2crx                             'leggo dalla bussola
compasstemp.Byte0 = i2cin               'scrivo nel registro temporaneo il valore della parte bassa
compasstemp = compasstemp / 10          'approssimo il valore al grado
'effettuo la media con il secondo valore letto
compassvalue = compassvalue + compasstemp
compassvalue = compassvalue / 2
compasstemp = compassvalue - compassoffset    'imposto il valore d'errore massimo per la lettura
IF compassvalue < compasstemp THEN            'se la media letta è inferiore al valore d'errore (lettura bussola
fallita)
GOTO readcompass                              'rieffettua la lettura della bussola
ENDIF
'il valore della bussola va da 0 a 359 corrispondente a 0 - 359 gradi, in senso orario partendo da nord.
RETURN

readinfra:                             'legge i valori dei sensori infrarossi per gli ostacoli superiori
infra1 = infraant                      'legge il valore dell'infrarosso anteriore
infra2 = infrapos                      'legge il valore dell'infrarosso posteriore
RETURN

readanalog:                             'legge gli adc utilizzati (tranne l'rssi che è a parte)
'Le tensioni delle batterie vengono espresse in cv/2 quindi la tensione reale in cV è data dal valore della
variabile moltiplicato
'per 2. Se ne dovrà tenere conto (mentalmente) per eseguire calcoli sulle variabili delle tensioni delle celle.
GOSUB setanalog                         'attivo gli ingressi analogici e l'hardware
'Legge i valori di tensione delle celle ed esegue i calcoli necessari
vbatttemp = 0                           'azzero la variabile temporanea
INPUT vbattcell2pin                     'ingresso adc cella 2
ADCIN vbattcell2, vbatttemp             'leggo la tensione della cella 2
vbatttemp = (vbatttemp + 1)             'calcolo la tensione in cv riferita alla tensione di fondoscala
vbatttemp = vbatttemp  */ (vfs */ 64)
vbatttemp = vbatttemp / 2               'divido in 2 per far stare il valore in un byte (risoluzione 0,02V)
battcell2 = vbatttemp.Byte0             'carico il valore nel registro effettivo
vbatttemp = 0                           'azzero la variabile temporanea
INPUT vbattpartpin                      'ingresso adc partitore pacco batterie
ADCIN vbattpart, vbatttemp              'leggo la tensione dimezzata del pacco batterie (non serve dimezzare poi)
vbatttemp = (vbatttemp + 1)             'calcolo la tensione in cv riferita alla tensione di fondoscala
vbatttemp = vbatttemp  */ (vfs */ 64)
vbatttemp = vbatttemp - battcell2       'calcolo il valore in cv/2 della tensione della cella 1 (risoluzione 0,
02V)
battcell1 = vbatttemp.Byte0             'carico il valore nel registro effettivo
vbatttemp = 0                           'azzero la variabile temporanea
RETURN

readrssi:                               'legge il valore del segnale rssi corrispondente alla quantità di segnale
radio
```

```
GOSUB setanalog                          'attivo gli ingressi analogici e l'hardware
INPUT rssipin                            'ingresso adc cella 2
ADCIN rssi, rssiword                     'leggo la tensione della cella 2
rssiword = (rssiword + 1) */ (vfs */ 64)    'calcolo la tensione in cv riferita alla tensione di fondoscala
RETURN



'Blocco di compressione/decompressione pacchetti dati:

unpackdata:                      'sub per la decompressione dei dati in ingresso (ricevuti via radio)
'niente per ora
RETURN

packdata:                        'sub per la compressione dei dati in uscita (da inviare via radio)
'I valori di fuffy sono già compressi (occupano un byte). Così ottengo approsimazione 2cm, ma sufficienti per la
loro funzione
'I sensori infrarossi sono già inclusi in un byte (sensors) che può essere usato per altri sensori digitali o
comandi
'Comprimo il valore della bussola elettronica usando un fattore di conversione
compassp = compassvalue */ compassfact
'Creo un byte con i valori approssimati delle tensioni delle due celle. Il primo Nibble corrisponde alla prima
cella
'(nibble basso), il secondo alla seconda (nibble alto).
vbatttemp = 0                            'azzero la variabile temporanea
vbatttemp = battcell1 - 125              'sottraggo 250cV perchè la tensione minima per l'emergenza non è
inferiore ai 2,5V
vbatttemp = vbatttemp */ 30              'divido il valore per farcelo stare in 4 bit (0-15) - Risoluzione 0,16V
vbattp = 0                               'azzero la variabile batterie
vbattp.0 = vbatttemp.0
vbattp.1 = vbatttemp.1
vbattp.2 = vbatttemp.2
vbattp.3 = vbatttemp.3
vbatttemp = 0                            'azzero la variabile temporanea
vbatttemp = battcell2 - 125              'sottraggo 2500mV perchè la tensione minima per l'emergenza non è
inferiore ai 2,5V
vbatttemp = vbatttemp */ 30              'divido il valore per farcelo stare in 4 bit (0-15) - Risoluzione 0,16V
vbattp.4 = vbatttemp.0
vbattp.5 = vbatttemp.1
vbattp.6 = vbatttemp.2
vbattp.7 = vbatttemp.3
RETURN



'Blocco di gestione comunicazione radio:

readradio:                           'riceve dati via radio
corericezerror = 0                   'azzero il flag di errore lettura radio
IF hybrid = 1 THEN                   'se ho deciso di usare i moduli ibridi (aurel)
   GOSUB readhybrid                  'leggo i moduli ibridi
ELSE                                 'altrimenti uso i moduli easy-radio
   GOSUB readeasyradio               'leggo i moduli easyradio
ENDIF
RETURN

writeradio:                          'invia i dati via radio
IF hybrid = 1 THEN                   'se ho deciso di usare i moduli ibridi (aurel)
   GOSUB writehybrid                 'scrivo i moduli ibridi
ELSE                                 'altrimenti uso i moduli easy-radio
   GOSUB writeeasyradio              'scrivo i moduli easyradio
ENDIF
RETURN


'Blocco Comunicazione Moduli Easy-Radio:

readeasyradio:
dataintemp = 0               'azzero la variabile buffer in ricezione
GOSUB serin2easyradio        'ricevo il primo byte (preambolo1)
IF dataintemp = compreamble1 THEN         'se ho ricevuto il preambolo 1
    GOSUB serin2easyradio                 'ricevo il secondo byte (preambolo2)
    IF dataintemp = compreamble2 THEN         'se ho ricevuto il preambolo 2 leggo tutti i byte in ingresso in
sequenza e li carico nei registri interni
       GOSUB serin2easyradio                  'ricevo il terzo byte (basestatep - byte dato 1)
       basestatep = dataintemp
       GOSUB serin2easyradio                  'ricevo il quarto byte (joypadkeysp - byte dato 2)
       joypadkeysp = dataintemp
       GOSUB serin2easyradio                  'ricevo il quinto byte (joyanddirezp - byte dato 3)
       joyanddirezp = dataintemp
       GOSUB serin2easyradio                  'ricevo il sesto byte (motdxpwmp - byte dato 4)
       motdxpwmp = dataintemp
       GOSUB serin2easyradio                  'ricevo il settimo byte (motsxpwmp - byte dato 5)
       motsxpwmp = dataintemp
       GOSUB serin2easyradio                  'ricevo l'ottavo byte (motdownpwmp - byte dato 6)
       motdownpwmp = dataintemp
       GOSUB serin2easyradio                  'ricevo il nono byte (postambolo)
       IF dataintemp <> compostamble THEN         'se non ricevo il postambolo correttamente
          GOTO errreadeasyradio                   'vado alla sub errore lettura easy radio
       ENDIF
    ELSE                                   'se non ho ricevuto il preambolo 2
    GOTO errreadeasyradio                  'vado alla sub errore lettura easy radio
    ENDIF
ELSE                                       'se non ho ricevuto il preambolo 2
```

```
GOTO errreadeasyradio                    'vado alla sub errore lettura easy radio
ENDIF
RETURN
errreadeasyradio:                        'gestione errore lettura easyradio
corericezerror = 1                       'errore lettura radio
RETURN
serin2easyradio:
SERIN2 rxpin,sermode,[dataintemp]         'ricevo un byte
RETURN



writeeasyradio:
IF cts = ~ctsatt THEN        'se la linea cts del modulo indica che il modulo è occupato
   GOTO writeeasyradio       'attendo che il modulo si liberi
ENDIF
SEROUT2 txpin,sermode,[compreamble1,compreamble2]       'invio il preambolo 1 seguito dal suo complementare
(preambolo 2)
'Invio tutti i dati in uscita
SEROUT2 txpin,sermode,[corebyte,fuffy1cenp,fuffy2cenp,fuffy1dxp,fuffy2dxp,fuffy1sxp,fuffy2sxp,fuffy3p,compassp, _
        sensorsp,vbattp,exbyte0,exbyte1,exbyte2,exbyte3]
SEROUT2 txpin,sermode,[compostamble]                    'invio il postambolo
waitctsfree:                 'attesa che il modulo easy radio non sia più occupato
IF cts = ~ctsatt THEN        'se la linea cts del modulo indica che il modulo è occupato
   GOTO waitctsfree          'attendo che il modulo si liberi
ENDIF
RETURN



'Blocco Comunicazione Moduli Ibridi:

readhybrid:
'niente per ora
RETURN


writehybrid:
'niente per ora
RETURN


'Blocco Lettura/Scrittura Memoria Ram:

'---> Scrittura locazioni
writeconfig:                    'leggo dalla ram e carico i registri in ingresso
i2caddrbw = 0                   'indirizzo dispositivo slave = byte
i2ccontrol = ramaddr            'codice di controllo ram i2c
i2cdelay = 10                   'delay accesso ram = 10us
    writeloc0:                  'posizione blocco lettura - 0
i2caddr = controlloc            'seleziono la locazione da leggere
i2cout = controlbyte            'carico il buffer in uscita
GOSUB i2ctx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO writeloc0           'se si verifica un errore di lettura, rileggo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc1:                  'posizione blocco lettura - 1
i2caddr = pwmupdownloc          'seleziono la locazione da leggere
i2cout = pwmmotdown             'carico il buffer in uscita
GOSUB i2ctx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO writeloc1           'se si verifica un errore di lettura, rileggo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc2:                  'posizione blocco lettura - 2
i2caddr = pwmmotdxloc           'seleziono la locazione da leggere
i2cout = pwmmotdx               'carico il buffer in uscita
GOSUB i2ctx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO writeloc2           'se si verifica un errore di lettura, rileggo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc3:                  'posizione blocco lettura - 3
i2caddr = pwmmotsxloc           'seleziono la locazione da leggere
i2cout = pwmmotsx               'carico il buffer in uscita
GOSUB i2ctx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO writeloc3           'se si verifica un errore di lettura, rileggo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc4:                  'posizione blocco lettura - 4
i2caddr = joyanddirezloc        'seleziono la locazione da leggere
i2cout = joyanddirez            'carico il buffer in uscita
GOSUB i2ctx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO writeloc4           'se si verifica un errore di lettura, rileggo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc5:                  'posizione blocco lettura - 5
i2caddr = joypadbuttonsloc      'seleziono la locazione da leggere
i2cout = joypadbuttons          'carico il buffer in uscita
GOSUB i2ctx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO writeloc5           'se si verifica un errore di lettura, rileggo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc6:                  'posizione blocco lettura - 6
i2caddr = basestateloc          'seleziono la locazione da leggere
i2cout = basestatebyte          'carico il buffer in uscita
```

```
GOSUB i2ctx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO writeloc6          'se si verifica un errore di lettura, rileggo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc7:                  'posizione blocco lettura - 7
i2caddr = ciclopecontrolloc     'seleziono la locazione da leggere
i2cout = ciclopecontrolbyte     'carico il buffer in uscita
GOSUB i2ctx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO writeloc7          'se si verifica un errore di lettura, rileggo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc8:                  'posizione blocco lettura - 8
i2caddr = rssibyte0loc          'seleziono la locazione da leggere
i2cout = rssibyte0              'carico il buffer in uscita
GOSUB i2ctx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO writeloc8          'se si verifica un errore di lettura, rileggo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc9:                  'posizione blocco lettura - 9
i2caddr = rssibyte1loc          'seleziono la locazione da leggere
i2cout = rssibyte1              'carico il buffer in uscita
GOSUB i2ctx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO writeloc9          'se si verifica un errore di lettura, rileggo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc10:                 'posizione blocco lettura - 10
i2caddr = battcell1loc          'seleziono la locazione da leggere
i2cout = battcell1              'carico il buffer in uscita
GOSUB i2ctx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO writeloc10         'se si verifica un errore di lettura, rileggo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc11:                 'posizione blocco lettura - 11
i2caddr = battcell2loc          'seleziono la locazione da leggere
i2cout = battcell2              'carico il buffer in uscita
GOSUB i2ctx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO writeloc11         'se si verifica un errore di lettura, rileggo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc12:                 'posizione blocco lettura - 12
i2caddr = sensorsloc            'seleziono la locazione da leggere
i2cout = sensors                'carico il buffer in uscita
GOSUB i2ctx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO writeloc12         'se si verifica un errore di lettura, rileggo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc13:                 'posizione blocco lettura - 13
i2caddr = compassbyte0loc       'seleziono la locazione da leggere
i2cout = compassbyte0           'carico il buffer in uscita
GOSUB i2ctx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO writeloc13         'se si verifica un errore di lettura, rileggo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc14:                 'posizione blocco lettura - 14
i2caddr = compassbyte1loc       'seleziono la locazione da leggere
i2cout = compassbyte1           'carico il buffer in uscita
GOSUB i2ctx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO writeloc14         'se si verifica un errore di lettura, rileggo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc15:                 'posizione blocco lettura - 15
i2caddr = corewordbyte0loc      'seleziono la locazione da leggere
i2cout = corewordbyte0          'carico il buffer in uscita
GOSUB i2ctx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO writeloc15         'se si verifica un errore di lettura, rileggo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc16:                 'posizione blocco lettura - 16
i2caddr = corewordbyte1loc      'seleziono la locazione da leggere
i2cout = corewordbyte1          'carico il buffer in uscita
GOSUB i2ctx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO writeloc16         'se si verifica un errore di lettura, rileggo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
RETURN

'---> Lettura Locazioni
readconfig:                     'scrivo nella ram i valori dei registri in uscita
i2caddrbw = 0                   'indirizzo dispositivo slave = byte
i2ccontrol = ramaddr            'codice di controllo ram i2c
i2cdelay = 10                   'delay accesso ram = 10us
    readloc0:                   'posizione blocco scrittura - 0
i2caddr = fuffy1cenloc          'seleziono la locazione da scrivere
GOSUB i2crx                     'scrivo il dato nella locazione in ram
fuffy1cmcen = i2cin             'carico il valore in ingresso
IF i2cackflag = 1 THEN GOTO readloc0           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc1:                   'posizione blocco scrittura - 1
i2caddr = fuffy2cenloc          'seleziono la locazione da scrivere
GOSUB i2crx                     'scrivo il dato nella locazione in ram
fuffy2cmcen = i2cin             'carico il valore in ingresso
IF i2cackflag = 1 THEN GOTO readloc1           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
```

```
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc2:                   'posizione blocco scrittura - 2
i2caddr = fuffy1dxloc           'seleziono la locazione da scrivere
GOSUB i2crx                     'scrivo il dato nella locazione in ram
fuffy1cmdx = i2cin              'carico il valore in ingresso
IF i2cackflag = 1 THEN GOTO readloc2            'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc3:                   'posizione blocco scrittura - 3
i2caddr = fuffy2dxloc           'seleziono la locazione da scrivere
GOSUB i2crx                     'scrivo il dato nella locazione in ram
fuffy2cmdx = i2cin              'carico il valore in ingresso
IF i2cackflag = 1 THEN GOTO readloc3            'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc4:                   'posizione blocco scrittura - 4
i2caddr = fuffy1sxloc           'seleziono la locazione da scrivere
GOSUB i2crx                     'scrivo il dato nella locazione in ram
fuffy1cmsx = i2cin              'carico il valore in ingresso
IF i2cackflag = 1 THEN GOTO readloc4            'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc5:                   'posizione blocco scrittura - 5
i2caddr = fuffy2sxloc           'seleziono la locazione da scrivere
GOSUB i2crx                     'scrivo il dato nella locazione in ram
fuffy2cmsx = i2cin              'carico il valore in ingresso
IF i2cackflag = 1 THEN GOTO readloc5            'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc6:                   'posizione blocco scrittura - 6
i2caddr = fuffy3loc             'seleziono la locazione da scrivere
GOSUB i2crx                     'scrivo il dato nella locazione in ram
fuffy3cm = i2cin                'carico il valore in ingresso
IF i2cackflag = 1 THEN GOTO readloc6            'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc7:                   'posizione blocco scrittura - 7
i2caddr = stateloc              'seleziono la locazione da scrivere
GOSUB i2crx                     'scrivo il dato nella locazione in ram
statebyte = i2cin               'carico il valore in ingresso
IF i2cackflag = 1 THEN GOTO readloc7            'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
RETURN


'Blocco Subroutine Settaggio Extra:

setanalog:                  'attivo tutti gli ingressi analogici
'imposto le porte come ingressi analogici
adcon1.0 = 0
adcon1.1 = 0
adcon1.2 = 0
adcon1.3 = 0
'impiosto la giustificazione del risultato a destra
adcon1.7 = 1
RETURN

setdigital:                 'attivo tutti gli ingressi digitali (disattivo adc)
'imposto le porte come i/o digitali
adcon1.0 = 1
adcon1.1 = 1
adcon1.2 = 1
adcon1.3 = 0
RETURN


'Blocco Moduli - Routines:
INCLUDE "i2cmod.bas"
INCLUDE "mpwm1.bas"
INCLUDE "mpwm2.bas"



'Blocco Gestione Errori:

initfail:                               'errore nell'inizializzazione primaria  (primaryinit:)
'niente per ora
GOTO primaryinit                        'riesegue l'inizializzazione


'In caso per sbaglio si verificassero questi errori
i2cbusy:
i2cackerror:
i2cmemoryaccesserror:
GOTO main           'resetto l'esecuzione del programma
```

PROGRAMMA FUFFY

```
'Blocco Direttive:
DEFINE I2C_SLOW 1              'rallento la velocità del bus i2c
DEFINE SHIFT_PAUSEUS 50        'rallento la velocità della seriale sincrona


'Blocoo Memoria Main Program:
srcfuffy1 CON 7                          'pin servomotore fuffy 1 - anteriore
srcfuffy2 CON 10                         'pin servomotore fuffy 2 - posteriore
trigfuffy1 CON 6                         'pin trigger ultrasuoni fuffy 1
echofuffy1 CON 5                         'pin echo ultrasuoni fuffy 1
trigfuffy2 CON 11                        'pin trigger ultrasuoni fuffy 2
echofuffy2 CON 12                        'pin echo ultrasuoni fuffy 2
trigfuffy3 CON 9                         'pin trigger ultrasuoni fuffy 3 - atterraggio
echofuffy3 CON 8                         'pin echo ultrasuoni fuffy 3
SYMBOL motdirez = portb.2                'pin direzione motori ascensionali

srcposcen CON 1500                       'posizione centrale servo (in us)
srcposdx CON 2290                        'posizione destra servo (in us)
srcpossx CON 710                         'posizione sinistra servo (in us)
minvelocity CON 100                      'valore di velocità (pwm) sotto al quale i motori sono considerati
fermi
anticollisiondelay CON 200               'valore in microsecondi tra un accesso e l'altro della memoria
pwmregenable CON 0                       'indica se attivare la modalità regolazione velocità motori con pwm

fuffy1cmcen VAR WORD                     'buffer valore misurato fuffy 1 - posizione centrale anteriore
fuffy2cmcen VAR WORD                     'buffer valore misurato fuffy 2 - posizione centrale posteriore
fuffy1cmdx VAR WORD                      'buffer valore misurato fuffy 1 - posizione destra anteriore
fuffy2cmdx VAR WORD                      'buffer valore misurato fuffy 2 - posizione destra posteriore
fuffy1cmsx VAR WORD                      'buffer valore misurato fuffy 1 - posizione sinistra anteriore
fuffy2cmsx VAR WORD                      'buffer valore misurato fuffy 2 - posizione sinistra posteriore
fuffy3cm VAR WORD                        'buffer valore misurato fuffy 3 - sotto
'NB: Le direzioni sono riferite alla struttura vista da dietro.



'Costanti e variabili locazioni configurazioni:
'--> In lettura:
controlloc CON 1                         'indirizzo locazione byte di controllo
controlbyte VAR BYTE                     'registro byte di controllo
stateloc CON 133                         'locazione byte di stato
statebyte VAR BYTE                       'registro byte di stato
pwmupdownloc CON 2                       'indirizzo locazione byte velocità motori
pwmupdownbyte VAR BYTE                   'registro byte velocità motori ascensionali
        '---> Aliases Bit Particolari
        corestate VAR controlbyte.0      'indica se il pic core è senziente (1)
        direzupdown VAR controlbyte.1    'direzione motori ascensionali
        fuffyerr VAR statebyte.0         'bit di segnalazione errore fuffys
        fuffymodstate VAR statebyte.1    'indica se il pic fuffy ha inizializzato le prime misure

'--> In scrittura:
fuffy1cenloc CON 126                     'indirizzo locazione byte valore fuffy 1 - posizione centrale
fuffy2cenloc CON 127                     'indirizzo locazione byte valore fuffy 2 - posizione centrale
fuffy1dxloc CON 128                      'indirizzo locazione byte valore fuffy 1 - posizione destra
fuffy2dxloc CON 129                      'indirizzo locazione byte valore fuffy 2 - posizione destra
fuffy1sxloc CON 130                      'indirizzo locazione byte valore fuffy 1 - posizione sinistra
fuffy2sxloc CON 131                      'indirizzo locazione byte valore fuffy 2 - posizione sinistra
fuffy3loc CON 132                        'indirizzo locazione byte valore fuffy 3


        '---> Aliases Bit Particolari
        fuffy1cenbyte VAR fuffy1cmcen.Byte0      'registro byte valore fuffy 1 - posizione centrale
        fuffy2cenbyte VAR fuffy2cmcen.Byte0      'registro byte valore fuffy 2 - posizione centrale
        fuffy1dxbyte VAR fuffy1cmdx.Byte0        'registro byte valore fuffy 1 - posizione destra
        fuffy2dxbyte VAR fuffy2cmdx.Byte0        'registro byte valore fuffy 2 - posizione destra
        fuffy1sxbyte VAR fuffy1cmsx.Byte0        'registro byte valore fuffy 1 - posizione sinistra
        fuffy2sxbyte VAR fuffy2cmsx.Byte0        'registro byte valore fuffy 2 - posizione sinistra
        fuffy3byte VAR fuffy3cm.Byte0            'registro byte valore fuffy 3


'Blocco Memoria Moduli:
INCLUDE "i2cvar.bas"
INCLUDE "mpwmvar.bas"
INCLUDE "servovar.bas"
INCLUDE "ultravar.bas"
INCLUDE "sonarvar.bas"



'Blocco Main - Configurazioni:

main:
CLEAR                            'azzera tutte le variabili
ANSEL = 0                        'imposto le porte analogiche come digitali
OUTPUT motdirez                  'pin direzione motori ascensionali è un'uscita
LOW motdirez                     'motdirez livello basso
mpwmpr2 = 100                    'imposto pr2 a 100 con presc unitario
mpwmpresc = 0                    'frequenza ottenuta F=10Khz
mpwmduty = 0                     'azzero il duty-cycle
monoffpwm = 0                    'modulo pwm spento
GOSUB mpwm1                      'imposto il modulo pwm
```

```
GOSUB i2cset                    'chiamo la sub di settaggio del modulo i2c (v. i2cset.bas)
GOSUB srcset                    'chiamo la sub di settaggio dei servi r/c
GOSUB ultraset                  'chiamo la sub di settaggio ultrasuoni
sonardelay = 10                 'attesa tra il posizionamento e la lettura del sensore
PAUSE 1000                      'pausa di accensione
GOTO hwset                      'vado alla sub di configurazione dell'hardware

'Settaggio e Check dell Hardware:
hwset:
PAUSE 700                       'pausa di accensione
GOSUB i2cmemorychk              'test accesso alla memoria
IF i2cmemorychkflag = 1 THEN    'se il check memoria è fallito
GOTO hwset                      'continuo a verificare l'accesso alla memoria
ENDIF
GOSUB writeconfig               'scrivo zeri nelle locazioni utilizzate
GOTO initsub                    'vado all'inizializzazione


'Blocco Inizializzazione - INIT:
initsub:                        'inizializzazione
mpwmduty = 0                    'azzero il duty-cycle
monoffpwm = 0                   'modulo pwm spento
GOSUB mpwm1                     'imposto il modulo pwm
'---> Imposto tutti i servi nella posizione centrale:
'--->servo Fuffy 1:
srcpin = srcfuffy1              'seleziono il servo fuffy 1
srcposus = srcposcen            'imposto la posizione centrale
GOSUB srcsetpos                 'eseguo il posizionamento
'--->servo Fuffy 2:
srcpin = srcfuffy2              'seleziono il servo fuffy 2
srcposus = srcposcen            'imposto la posizione centrale
GOSUB srcsetpos                 'eseguo il posizionamento
GOTO prgmain                    'vado al blocco principale del programma


'Blocco Settaggi Moduli:
INCLUDE "i2cset.bas"
INCLUDE "servoset.bas"
INCLUDE "ultraset.bas"


'Blocoo Principale del Programma:
prgmain:                        'Blocco Principale
GOTO fuffyprelim                'inizio il blocco principale

fuffyprelim:                    'fase preliminare iniziale
'viene controllato se il core è attivo e poi vengono eseguite le misure
'di distanza in tutte le direzioni, salvate in ram e confermato lo stato
'attivo del modulo fuffy.
PAUSE 100                           'pausa check (per evitare di occupare troppo il bus)
GOSUB readconfig
IF corestate = 1 THEN               'se il core è attivo
   GOSUB missxdx                    'effettuo le misure anteriore-sx e posteriore-dx
   GOSUB misdxsx                    'effettuo le misure anteriore-dx e posteriore-sx
   GOSUB miscen                     'effettuo le misure anteriore-centrale e posteriore-centrale
   GOSUB misupdown                  'effettuo la misura sotto (atterraggio)
   fuffymodstate = 1                'indico che il modulo fuffy è pronto
   GOSUB writeconfig                'scrivo tutti i valori in ram
   fuffyerr = 0                     'azzero il flag errore
   GOSUB fuffyprinc                 'vado alla routine di programma del modulo fuffy
ENDIF
GOTO fuffyprelim                    'continua ad eseguire fino al verificarsi della condizione


fuffyprinc:                         'routine di programma modulo fuffy
'Il programma legge la memoria e controlla se il core è attivo prima di
'eseguire ogni operazione. In caso non vi siano errori vengono impostate
'direzione e velocità dei motori ascensionali in base ai dati letti dalla ram,
'vengono effettuate le letture dei Fuffys e salvati in ram i valori.
'Per motivi di velocità di risposta dei motori ascensionali, la lettura della ram
'(controlli inclusi) e l'impostazione dei motori ascensionali vengono fatte
'ad ogni direzione di misura dei fuffys (centro, dx, sx, sotto).
GOSUB readconfig                    'leggo la memoria ram
IF corestate = 0 THEN GOTO initsub          'se il core non è attivo ritorno all'inizializzazione
GOSUB pwmmotset                     'imposto velocità e direzione dei motori ascensionali
GOSUB missxdx                       'effettuo le misure anteriore-sx e posteriore-dx
GOSUB misupdown                     'effettuo la misura sotto (atterraggio)
GOSUB writeconfig                   'aggiorno i valori misurati in ram
fuffyerr = 0                        'azzero il flag errore
PAUSEUS 10                          'piccola pausa tra un read ed un write
GOSUB readconfig                    'leggo la memoria ram
IF corestate = 0 THEN GOTO initsub          'se il core non è attivo ritorno all'inizializzazione
GOSUB pwmmotset                     'imposto velocità e direzione dei motori ascensionali
GOSUB misdxsx                       'effettuo le misure anteriore-dx e posteriore-sx
GOSUB misupdown                     'effettuo la misura sotto (atterraggio)
GOSUB writeconfig                   'aggiorno i valori misurati in ram
fuffyerr = 0                        'azzero il flag errore
PAUSEUS 10                          'piccola pausa tra un read ed un write
GOSUB readconfig                    'leggo la memoria ram
IF corestate = 0 THEN GOTO initsub          'se il core non è attivo ritorno all'inizializzazione
GOSUB pwmmotset                     'imposto velocità e direzione dei motori ascensionali
GOSUB miscen                        'effettuo le misure anteriore-centrale e posteriore-centrale
GOSUB misupdown                     'effettuo la misura sotto (atterraggio)
GOSUB writeconfig                   'aggiorno i valori misurati in ram
```

```
fuffyerr = 0                        'azzero il flag errore
PAUSEUS 10                          'piccola pausa tra un read ed un write
GOSUB readconfig                    'leggo la memoria ram
IF corestate = 0 THEN GOTO initsub        'se il core non è attivo ritorno all'inizializzazione
GOSUB pwmmotset                     'imposto velocità e direzione dei motori ascensionali
GOSUB misupdown                     'effettuo la misura sotto (atterraggio)
GOSUB writeconfig                   'aggiorno i valori misurati in ram
fuffyerr = 0                        'azzero il flag errore
PAUSEUS 10                          'piccola pausa tra un read ed un write
GOTO fuffyprinc                     'riesegue all'infinito la routine principale


'Blocco impostazione motori ascensionali - pwm:

pwmmotset:                                  'sub controllo dei motori ascensionali
IF pwmregenable = 1 THEN                    'se è attiva la modalità pwm regolazione velocità
    mpwmduty = pwmupdownbyte                    'imposto il valore del duty-cycle in base alla velocità
    IF direzupdown <> motdirez THEN             'se il verso di rotazione è cambiato
       monoffpwm = 0                            'spengo il pwm
       GOSUB mpwm1                              'imposto il modulo pwm
       PAUSE 10                                 'piccola pausa per lasciare che il motore inverta il senso di
rotazione
       motdirez = direzupdown                   'modifico la direzione di rotazione dei motori
       monoffpwm = 1                            'riaccendo il pwm
    ENDIF
    IF pwmupdownbyte < minvelocity THEN         'se la velocità è pressochè nulla
    monoffpwm = 0                               'spengo il modulo pwm (risparmio energia)
    ELSE                                        'altrimenti
    monoffpwm = 1                               'accendo il modulo pwm
    ENDIF
    GOSUB mpwm1                                 'imposto il modulo pwm
ELSE                                        'altrimenti non effettuo regolazioni (max velocità)
    monoffpwm = 0                               'spengo il pwm
    GOSUB mpwm1                                 'imposto il modulo pwm
    OUTPUT mpwmpin1                             'imposto il pin di controllo motori come output
    IF direzupdown <> motdirez THEN             'se il verso di rotazione è cambiato
       mpwmpin1 = 0                                 'spengo il motore
       PAUSE 10                                     'piccola pausa per lasciare che il motore inverta il senso di
rotazione
       motdirez = direzupdown                       'modifico la direzione di rotazione dei motori
       mpwmpin1 = 1                                 'accendo il motore
    ENDIF
    IF pwmupdownbyte < minvelocity THEN         'se la velocità è pressochè nulla
    mpwmpin1 = 0                                'spengo il motore
    ELSE                                        'altrimenti
    mpwmpin1 = 1                                'accendo il motore
    ENDIF
ENDIF
RETURN


'Blocco routines di misura e posizionamento Fuffy:

miscen:                                     'misura distanza posizione centrale anteriore e posteriore
srcpin = srcfuffy1                          'seleziono i pin di fuffy anteriore - servomotore
ultratriggerpin = trigfuffy1                'seleziono i pin di fuffy anteriore - trigger
ultraechopin = echofuffy1                   'seleziono i pin di fuffy anteriore - echo
srcposus = srcposcen                        'seleziono la posizione centrale (del servomotore)
GOSUB sonarint                              'effettuo la misura alla posizione scelta (media 2 valori)
fuffy1cmcen = ultracm                       'scrivo il valore nel registro (anteriore centrale)
IF ultranoconnect = 1 THEN                  'se si è verificato un errore hardware sulla lettura
   fuffyerr = 1                             'attivo il flag errore lettura generale modulo fuffy
ENDIF
fuffy1cmcen = fuffy1cmcen / 2               'divido il valore per farlo stare in un byte (ricordarsi
mentalmente
                                            'che il valore va raddoppiato per avere i cm corretti)
                                            'La risoluzione dopo la compressione è di 2cm
srcpin = srcfuffy2                          'seleziono i pin di fuffy posteriore - servomotore
ultratriggerpin = trigfuffy2                'seleziono i pin di fuffy posteriore - trigger
ultraechopin = echofuffy2                   'seleziono i pin di fuffy posteriore - echo
srcposus = srcposcen                        'seleziono la posizione centrale (del servomotore)
GOSUB sonarint                              'effettuo la misura alla posizione scelta (media 2 valori)
fuffy2cmcen = ultracm                       'scrivo il valore nel registro (posteriore centrale)
IF ultranoconnect = 1 THEN                  'se si è verificato un errore hardware sulla lettura
   fuffyerr = 1                             'attivo il flag errore lettura generale modulo fuffy
ENDIF
fuffy2cmcen = fuffy2cmcen / 2               'divido il valore per farlo stare in un byte (ricordarsi
mentalmente
                                            'che il valore va raddoppiato per avere i cm corretti)
                                            'La risoluzione dopo la compressione è di 2cm
RETURN

misdxsx:                                    'misura distanza posizione anteriore-dx e posteriore-sx
srcpin = srcfuffy1                          'seleziono i pin di fuffy anteriore - servomotore
ultratriggerpin = trigfuffy1                'seleziono i pin di fuffy anteriore - trigger
ultraechopin = echofuffy1                   'seleziono i pin di fuffy anteriore - echo
srcposus = srcposdx                         'seleziono la posizione destra (del servomotore)
GOSUB sonarint                              'effettuo la misura alla posizione scelta (media 2 valori)

fuffy1cmdx = ultracm                        'scrivo il valore nel registro (anteriore destra)
IF ultranoconnect = 1 THEN                  'se si è verificato un errore hardware sulla lettura
   fuffyerr = 1                             'attivo il flag errore lettura generale modulo fuffy
ENDIF
fuffy1cmdx = fuffy1cmdx / 2                 'divido il valore per farlo stare in un byte (ricordarsi
mentalmente
```

```
                                        'che il valore va raddoppiato per avere i cm corretti)
                                        'La risoluzione dopo la compressione è di 2cm
srcpin = srcfuffy2                      'seleziono i pin di fuffy posteriore - servomotore
ultratriggerpin = trigfuffy2            'seleziono i pin di fuffy posteriore - trigger
ultraechopin = echofuffy2               'seleziono i pin di fuffy posteriore - echo
srcposus = srcposdx                     'seleziono la posizione destra (del servomotore)
GOSUB sonarint                          'effettuo la misura alla posizione scelta (media 2 valori)
fuffy2cmsx = ultracm                    'scrivo il valore nel registro (posteriore sinistra)
IF ultranoconnect = 1 THEN              'se si è verificato un errore hardware sulla lettura
   fuffyerr = 1                         'attivo il flag errore lettura generale modulo fuffy
ENDIF
fuffy2cmsx = fuffy2cmsx / 2             'divido il valore per farlo stare in un byte (ricordarsi
mentalmente
                                        'che il valore va raddoppiato per avere i cm corretti)
                                        'La risoluzione dopo la compressione è di 2cm
RETURN

missxdx:                                'misura distanza posizione anteriore-sx e posteriore-dx
srcpin = srcfuffy1                      'seleziono i pin di fuffy anteriore - servomotore
ultratriggerpin = trigfuffy1            'seleziono i pin di fuffy anteriore - trigger
ultraechopin = echofuffy1               'seleziono i pin di fuffy anteriore - echo
srcposus = srcpossx                     'seleziono la posizione sinistra (del servomotore)
GOSUB sonarint                          'effettuo la misura alla posizione scelta (media 2 valori)

fuffy1cmsx = ultracm                    'scrivo il valore nel registro (anteriore sinistra)
IF ultranoconnect = 1 THEN              'se si è verificato un errore hardware sulla lettura
   fuffyerr = 1                         'attivo il flag errore lettura generale modulo fuffy
ENDIF
fuffy1cmsx = fuffy1cmsx / 2             'divido il valore per farlo stare in un byte (ricordarsi
mentalmente
                                        'che il valore va raddoppiato per avere i cm corretti)
                                        'La risoluzione dopo la compressione è di 2cm
srcpin = srcfuffy2                      'seleziono i pin di fuffy posteriore - servomotore
ultratriggerpin = trigfuffy2            'seleziono i pin di fuffy posteriore - trigger
ultraechopin = echofuffy2               'seleziono i pin di fuffy posteriore - echo
srcposus = srcpossx                     'seleziono la posizione sinistra (del servomotore)
GOSUB sonarint                          'effettuo la misura alla posizione scelta (media 2 valori)
fuffy2cmdx = ultracm                    'scrivo il valore nel registro (posteriore destra)
IF ultranoconnect = 1 THEN              'se si è verificato un errore hardware sulla lettura
   fuffyerr = 1                         'attivo il flag errore lettura generale modulo fuffy
ENDIF
fuffy2cmdx = fuffy2cmdx / 2             'divido il valore per farlo stare in un byte (ricordarsi
mentalmente
                                        'che il valore va raddoppiato per avere i cm corretti)
                                        'La risoluzione dopo la compressione è di 2cm
RETURN

misupdown:
ultratriggerpin = trigfuffy3            'seleziono i pin di fuffy sotto - trigger
ultraechopin = echofuffy3               'seleziono i pin di fuffy sotto - echo
GOSUB ultraread                         'legge la distanza del sensore fuffy 3 - atterraggio
fuffy3cm = ultracm                      'scrivo il valore nel registro (atterraggio sotto)
IF ultranoconnect = 1 THEN              'se si è verificato un errore hardware sulla lettura
   fuffyerr = 1                         'attivo il flag errore lettura generale modulo fuffy
ENDIF
fuffy3cm = fuffy3cm / 2                 'divido il valore per farlo stare in un byte (ricordarsi
mentalmente
                                        'che il valore va raddoppiato per avere i cm corretti)
                                        'La risoluzione dopo la compressione è di 2cm
RETURN



'Blocco Lettura/Scrittura Memoria:

readconfig:                     'leggo dalla ram e carico i registri in ingresso
    readloc0:                   'posizione blocco lettura - 0
i2caddr = controlloc            'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc0            'se si verifica un errore di lettura, rileggo la locazione
corrente
controlbyte = i2cin             'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc1:                   'posizione blocco lettura - 1
i2caddr = pwmupdownloc          'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc1            'se si verifica un errore di lettura, rileggo la locazione
corrente
pwmupdownbyte = i2cin           'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
RETURN


writeconfig:                    'scrivo nella ram i valori dei registri in uscita
    writeloc0:                  'posizione blocco scrittura - 0
i2caddr = fuffy1cenloc          'seleziono la locazione da scrivere
i2cout = fuffy1cenbyte          'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc0           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc1:                  'posizione blocco scrittura - 1
i2caddr = fuffy2cenloc          'seleziono la locazione da scrivere
i2cout = fuffy2cenbyte          'carico il valore in uscita
```

```
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc1           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc2:                  'posizione blocco scrittura - 2
i2caddr = fuffy1dxloc           'seleziono la locazione da scrivere
i2cout = fuffy1dxbyte           'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc2           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc3:                  'posizione blocco scrittura - 3
i2caddr = fuffy2dxloc           'seleziono la locazione da scrivere
i2cout = fuffy2dxbyte           'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc3           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc4:                  'posizione blocco scrittura - 4
i2caddr = fuffy1sxloc           'seleziono la locazione da scrivere
i2cout = fuffy1sxbyte           'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc4           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc5:                  'posizione blocco scrittura - 5
i2caddr = fuffy2sxloc           'seleziono la locazione da scrivere
i2cout = fuffy2sxbyte           'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc5           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc6:                  'posizione blocco scrittura - 6
i2caddr = fuffy3loc         'seleziono la locazione da scrivere
i2cout = fuffy3byte         'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc6           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
  writeloc7:                    'posizione blocco scrittura - 7
i2caddr = stateloc              'seleziono la locazione da scrivere
i2cout = statebyte              'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc7           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
RETURN



'Blocoo Moduli - Routines:
INCLUDE "i2cmod.bas"
INCLUDE "servomod.bas"
INCLUDE "ultramod.bas"
INCLUDE "sonarmod.bas"
INCLUDE "mpwm1.bas"


'Blocoo Gestione errori:

'---> Sub di gestione errori non utilizzate
i2cbusy:
i2cackerror:
i2cmemoryaccesserror:
GOTO hwset              'in caso vengano chiamate per sbaglio rimando a hwset


'Blocco Extra:
```

PROGRAMMA PICMASTER

```
'Blocco direttive:
'---> Configurazione Display LCD Base
DEFINE SHIFT_PAUSEUS 50
DEFINE LCD_DREG PORTD         'porta a cui è collegato l'lcd (canale dati)
DEFINE LCD_BITS 4             'uso un bus a 4 bit
DEFINE LCD_DBIT 4             'il bus dati del display è collegato agli ultimi 4 pin della portd
DEFINE LCD_RSREG PORTD        'porta dove è collegato il segnale RS del display
DEFINE LCD_RSBIT 2            'pin della porta a cui è collegato RS
DEFINE LCD_EREG PORTD         'porta dove è collegato il segnale ENABLE (E) del display
DEFINE LCD_EBIT 3             'pin della porta a cui è collegato E
DEFINE LCD_LINES 4            'linee del display LCD (Display a 4 linee)
DEFINE LCD_COMMANDUS 2000     'tempo di delay per i comandi del display
DEFINE LCD_DATAUS 50          'tempo di delay per i dati del display
DEFINE ADC_BITS 10            'numero di bit usati dall'adc
DEFINE ADC_CLOCK 3            'frequenza usata dall'adc (3 = internal rc)
DEFINE ADC_SAMPLEUS 50        'tempo campionamento usato dall'adc



'Blocco memoria Programma Principale:

'---> Symbols - Definizione Porte I/O
SYMBOL radiocts = portb.4                'segnale CTS (Input), Alto quando il modulo è occupato e non può ricevere
dati da spedire
SYMBOL radiorx = portb.3                 'segnale dati ricevuti via radio
SYMBOL radiotx = portb.2                 'segnale dati da trasmettere via radio
SYMBOL radiorts = portb.1                'segnale RTS (output), Alto per indicare che la base è occupata (Va messo
basso all'avvio del programma e lasciato così)
SYMBOL radioselect = portb.1             'segnale selezione trasmettitore/ricevitore (da usare solo con moduli
ibridi)
SYMBOL displayonoff = portb.0            'segnale accensione/spegnimento retroilluminazione display
SYMBOL debugrx = portc.7                 'uscita debug - dati in uscita
SYMBOL debugtx = portc.6                 'uscita debug - dati in ingresso
SYMBOL expled = portd.1                  'led segnalazione espansione base attiva - connesso a expdig0state
SYMBOL masterled1 = porte.0              'led segnalazione pic master 1
SYMBOL masterled2 = porta.5              'led segnalazione pic master 2
SYMBOL displaybutton = porta.4           'pulsante funzionale display (cambio pagina)
SYMBOL chiaveint = porta.3               'chiave di avviamento (lunica cosa che può far partire il programma)
SYMBOL cicalino = portc.2                'ciclaino per segnalazioni acustiche
SYMBOL expdig0state = portd.0            'canale digitale 0 - espansione base (segnala se l'espansione è connessa)
- connesso a expled
SYMBOL expdig1 = portc.1                 'canale digitale 1 - espansione base
SYMBOL expdig2 = portc.0                 'canale digitale 2 - espansione base
SYMBOL expan0pin = porte.2               'pin canale analogico 0 - espansione base
expan0 CON 7                             'canale di lettura analogico corrispondente al pin canale analogico 0 -
espansione base
SYMBOL expan1pin = porte.1               'pin canale analogico 1 - espansione base
expan1 CON 6                             'canale di lettura analogico corrispondente al pin canale analogico 1 -
espansione base
SYMBOL batttotquartipin = porta.2        'pin canale analogico lettura tensione totale batterie divisa in 4
(partitore)
batttotquarti CON 2                      'canale di lettura analogico corrispondente al pin canale analogico
tensione totale batterie
SYMBOL battcell2mezzipin = porta.1       'pin canale analogico lettura tensione cella 2 pacco batterie divisa in 2
(partitore)
battcell2mezzi CON 1                     'canale di lettura analogico corrispondente al pin canale analogico
tensione cella 2 (batterie)



'---> Variabili

      '---> Dati Dirigibile da leggere via radio e scrivere in ram
      corebyte VAR BYTE           'Parte Bassa Word di Stato Core
      fuffy1cenp VAR BYTE         'dato fuffy1 centro
      fuffy2cenp VAR BYTE         'dato fuffy2 centro
      fuffy1dxp VAR BYTE          'dato fuffy1 destra
      fuffy2dxp VAR BYTE          'dato fuffy2 destra
      fuffy1sxp VAR BYTE          'dato fuffy1 sinistra
      fuffy2sxp VAR BYTE          'dato fuffy2 sinistra
      fuffy3p VAR BYTE            'dato fuffy3 sotto
      compassp VAR BYTE           'valore compresso della bussola digitale
      sensorsp VAR BYTE           'valori dei sensori (infrarossi) e altro
      vbattp VAR BYTE             'livelli batterie pacchettati
      exbyte0 VAR BYTE            'byte aggiuntivo esterno 0 (espansioni)
      exbyte1 VAR BYTE            'byte aggiuntivo esterno 1 (espansioni)
      exbyte2 VAR BYTE            'byte aggiuntivo esterno 2 (espansioni)
      exbyte3 VAR BYTE            'byte aggiuntivo esterno 3 (espansioni)
      '---> Alias CoreByte
      coreon VAR corebyte.0               'Flag che indica alla base che il core è attivo
      coreinit VAR corebyte.1             'Flag che indica alla base che il core è in stato di inizializzazione
      corericezerror VAR corebyte.2       'Flag che indica che si è verificato un errore di ricezione dato via
radio (nel dirigibile)
      ciclopeunavable VAR corebyte.3      'Flag che indica che il modulo ciclopè non è disponibile al momento

      '---> Dati Dirigibile da leggere dalla ram e inviare via radio
      basestatep VAR BYTE           'dato byte di controllo base (per il dirigibile)
      joypadbuttonsp VAR BYTE       'dato pulsanti joypad (e comandi pc)
      joyanddirezp VAR BYTE         'croce joypad e direzione motori (e comandi pc)
      motdxpwmp VAR BYTE            'dato velocità motore destro
```

```
      motsxpwmp VAR BYTE              'dato velocità motore sinistro
      motdownpwmp VAR BYTE            'dato velocità motori ascensionali
              '---> Alias pulsanti joypad e direzioni
               keyup VAR joyanddirezp.0             'bit tasto freccia in su (telecamera asse y)
               keyright VAR joyanddirezp.1          'bit tasto freccia a destra (telecamera asse x)
               keydown VAR joyanddirezp.2           'bit tasto freccia in basso (telecamera asse y)
               keyleft VAR joyanddirezp.3           'bit tasto freccia a sinistra (telecamera asse x)
               keycroce VAR joypadbuttonsp.0        'bit tasto Croce
               keyquadrato VAR joypadbuttonsp.1     'bit tasto Quadrato
               keytriangolo VAR joypadbuttonsp.2    'bit tasto Triangolo
               keycerchio VAR joypadbuttonsp.3      'bit tasto Cerchio
               keyStart VAR joypadbuttonsp.4        'bit tasto Start (conferme autonav)
               keySelect VAR joypadbuttonsp.5       'bit tasto Select
               keyR1fun VAR joypadbuttonsp.6        'bit tasto R1 - funzione (telecamera on/off)
               keyR2fun VAR joypadbuttonsp.7        'bit tasto R2 - funzione (puntatore laser on/off)
               direzmotdx VAR joyanddirezp.4        'direzione motore destro (motore 1)
               direzmotsx VAR joyanddirezp.5        'direzione motore sinistro (motore 2)
               direzmotdown VAR joyanddirezp.6      'direzione motori ascensionali (sotto)
               'tutti i pulsanti attivi alti



      '---> Dati Base Terrestre da scrivere in ram
      segnalazbyte VAR BYTE       'byte flag di segnalazione (leds etc)
      picmasterstate VAR BYTE     'byte di stato del pic centrale (pic master) della base
      baseexpstate VAR BYTE       'byte di stato e comunicazione espansione base
      baseintbyte VAR BYTE        'byte per usi interni base
      basecell1 VAR BYTE          'byte tensione pacchettata in 8bit cella 1 pacco base
      basecell2 VAR BYTE          'byte tensione pacchettata in 8bit cella 2 pacco base
      baserssi VAR BYTE           'valore rssi base terrstre (valore tensione pacchettato in 8 bit)
      vibro VAR BYTE              'valore di vibrazione joypad
      '---> Dati Base Terrestre da leggere dalla ram
      devicesstate VAR BYTE       'stato dei dispositivi di comando
      statebytej VAR BYTE         'byte flags di stato pic psx
      prioritydev VAR BYTE        'byte di priorità con flags di priorità e controllo joypad
      pcstate VAR BYTE            'byte di stato del pc principale (COM1)
                  '---> Alias Stati Dispositivi
                  joystate VAR devicesstate.0       'stato del joypad (0 = non disponibile)
                  com1state VAR devicesstate.1      'stato della COM 1 (0 = non disponibile)
                  com2state VAR devicesstate.2      'stato della COM 2 (0 = non disponibile)
                  devicesfail VAR devicesstate.3    'nessuna periferica attiva (valore 1)
                  com1error VAR devicesstate.4      'errore lettura com1
                  com2error VAR devicesstate.5      'errore di lettura com2
                  comunicerror VAR devicesstate.6   'errore "lettura fallita da tutti i dispositivi connessi"
                  '---> Alias byte segnalazione
                  led1cstate VAR segnalazbyte.0     'stato del led 1 - pic comunicazioni
                  led2cstate VAR segnalazbyte.1     'stato del led 2 - pic comunicazioni
                  led1pmstate VAR segnalazbyte.2    'stato del led 1 - pic master
                  led2pmstate VAR segnalazbyte.3    'stato del led 2 - pic master
                  '---> Alias Stati pic master
                  picmasteron VAR picmasterstate.0         'indica che il pic master è attivo
                  picmastererrricez VAR picmasterstate.1   'indica che si è verificato un errore durante la
ricezione radio
                  '--> Alias flags di stato pic psx
                  picpsxon VAR statebytej.0          'indica che il pic psx è attivo
                  joynotconnected VAR statebytej.1   'indica che il joypad non è connesso o si sono verificati
problemi hardware
                  '---> Alias dei flag di priorità
                  centralstatej VAR prioritydev       'alias variabile usata per la comunicazione con il joypad
(in pratica uso il byte prioritydev)
                  centralon VAR prioritydev.0         'indica che il pic gestore delle comunicazioni è attivo
                  psxdisable VAR prioritydev.1        'interrompe l'esecuzione del programma (disabilita modulo
psx)
                  disablevib VAR prioritydev.2        'se attivo disabilita la vibrazione anche se attivata
manualmente
                  centralok VAR prioritydev.3         'indica che il pic gestore è pronto
                  priorityflag VAR disablevib         'il flag di priorità è lo stesso del flag per disabilitare
la vibrazione
                                                      'dato che se il joypad non viene letto, ne disabilito la
vibrazione
                                                      'se disabilitato per lungo periodo sul pc attivo anche
"psxdisable"
                  '---> Alias Variabile Stato PC
                  rxpcstate VAR pcstate.0             'valore stato pc ricevuto (1 = pc attivo)
                  replystate VAR pcstate.1            'valore stato pc confermato (il programma rileva la base)

      preamble1 VAR BYTE                          'valore letto preambolo 1
      preamble2 VAR BYTE                          'valore letto preambolo 2
      postamble VAR BYTE                          'valore letto postambolo
      schermatedisplay VAR BYTE                   'contatore per la scelta delle schermate (max 255 schermate)
      genpurpose VAR WORD                         'word per usi generali
      '---> Alias variabile per usi generali
      gencont VAR genpurpose.Byte0                'contatore per usi generali
      dataintemp VAR genpurpose.Byte1             'varibile temporanea buffer dati ricezione radio

      statusword  VAR WORD                        'flag interni, gestione errori base
      '---> Flags Status Word
      dispbuttonstate VAR statusword.0            'stato pulsante display memorizzato
      priorityscreen VAR statusword.1             'flag per la gestione di schermate prioritarie



'---> Costanti
    hybrid CON 0                                      'impostazione moduli radio (se 1 significa che uso i moduli
ibridi)
```

```
    ctsatt CON 0                                    'livello logico perchè cts sia attivo (attivo basso quindi)
    rtsatt CON 0                                    'livello logico perchè rts sia attivo (attivo basso quindi)
    sermode CON 396                                 'impostazione velocità di comunicazione serin2/serout2
(moduli easy radio)
    compreamble1 CON %10101010                      'valore standard preambolo 1
    compreamble2 CON %01010101                      'valore standard preambolo 2 (complementare)
    compostamble CON %10101010                      'valore standard postambolo
    radiorxtimeout CON 2000                         'tempo timeout per la ricezione dati radio (moduli easy
radio)
    pausetrx CON 50                                 'pausa tra operazioni radio (temporizzazione ciclo)
    anticollisiondelay CON 10                       'valore in microsecondi tra un accesso e l'altro al bus
    rampaketpause CON 70                            'valore in millisecondi di pausa tra ogni pacchetto
scritto/letto in ram
    antirimbalzo CON 200                            'pausa antirimbalzo per i tasti (in ms)
    ultimaschermatadati CON 1                       'numero schermate dati

'---> Comandi Display
      dcommand CON $FE              'invio il codice per un comando (da inviare prima di un "codice comando")
      dblinkon CON $0F              'comando attivazione cursore lampeggiante
      dblinkoff CON $0C             'comando disattivazione cursore lampeggiante (attivazione "nessun cursore")
      dclear CON 1                  'comando cancellazione schermata display
      dhome CON 2                   'comando che porta il cursore all'inizio del display (inizio prima riga)
      dline1 CON $80                'comando che porta il cursore all'inizio della prima riga
      dline2 CON $C0                'comando che porta il cursore all'inizio della seconda riga
      dline3 CON $90                'comando che porta il cursore all'inizio della terza riga
      dline4 CON $D0                'comando che porta il cursore all'inizio della quarta riga
      'il display ha 4 righe da 16 caratteri. Si può indicizzare un carattere incrementando il valore dei comandi
      'di inizio liea. Per esempio scrivendo dline1 + 6 si porterà il cursore sul carattere 6 (partendo a
scrivere da li)


'Blocco Locazioni Ram:

'---> Locazioni Joypad
    '---> In Lettura (Da usare solo per debug, quindi non verranno lette dalle routines di questo programma)
    statebytejloc CON 2                 'locazione byte di stato joypad  -  joypad
    joypadbuttonsjloc CON 3             'locazione byte tasti joypad  -  joypad
    joyanddirezjloc CON 4               'locazione byte tasti croce joypad e direzione motori  -  joypad
    pwmmotdxjloc CON 5                  'locazione velocità motore destro  -  joypad
    pwmmotsxjloc CON 6                  'locazione velocità motore sinistro  -  joypad
    pwmmotdownjloc CON 7                'locazione velocità motori ascensionali  -  joypad
    centralstatejloc CON 129            'locazione byte di stato pic centrale comunicazioni  -  joypad

    '---> In scrittura
    vibroloc CON 130                    'locazione valore vibrazione joypad (non usata in questo programma) -
joypad

'---> Locazioni Pic Master
    '---> In  Scrittura
    corebyteloc CON 10              'locazione Parte Bassa Word di Stato Core
    fuffy1cenploc CON 11            'locazione dato fuffy1 centro
    fuffy2cenploc CON 12            'locazione dato fuffy2 centro
    fuffy1dxploc CON 13             'locazione dato fuffy1 destra
    fuffy2dxploc CON 14             'locazione dato fuffy2 destra
    fuffy1sxploc CON 15             'locazione dato fuffy1 sinistra
    fuffy2sxploc CON 15             'locazione dato fuffy2 sinistra
    fuffy3ploc CON 17               'locazione dato fuffy3 sotto
    compassploc CON 18              'locazione valore compresso della bussola digitale
    sensorsploc CON 19              'locazione valori dei sensori (infrarossi) e altro
    vbattploc CON 20                'locazione livelli batterie pacchettati
    exbyte0loc CON 21               'locazione byte aggiuntivo esterno 0 (espansioni)
    exbyte1loc CON 22               'locazione byte aggiuntivo esterno 1 (espansioni)
    exbyte2loc CON 23               'locazione byte aggiuntivo esterno 2 (espansioni)
    exbyte3loc CON 24               'locazione byte aggiuntivo esterno 3 (espansioni)
    segnalazbyteloc CON 25          'locazione byte flag di segnalazione (leds etc)
    picmasterstateloc CON 26        'locazione byte di stato del pic centrale (pic master) della base
    baseexpstateloc CON 27          'locazione byte di stato e comunicazione espansione base
    basecell1loc CON 28             'locazione byte tensione cella 1 base
    basecell2loc CON 29             'locazione byte tensione cella 2 base
    baserssiloc CON 30              'locazione byte valore rssi base

    baseintbyteloc CON 31           'locazione byte per usi interni base (lettura e scrittura)

    '---> In Lettura
      basestateploc CON 135             'locazione dato byte di controllo base (per il dirigibile) - Valore
Definitivo
      joypadbuttonsploc CON 136         'locazione dato pulsanti joypad (e comandi pc) - Valore Definitivo
      joyanddirezploc CON 137           'locazione croce joypad e direzione motori (e comandi pc) - Valore
Definitivo
      motdxpwmploc CON 138              'locazione dato velocità motore destro - Valore Definitivo
      motsxpwmploc CON 139              'locazione dato velocità motore sinistro - Valore Definitivo
      motdownpwmploc CON 140            'locazione dato velocità motori ascensionali - Valore Definitivo
      prioritydevloc CON 141            'locazione byte di priorità con flags di priorità e controllo joypad -
Valore Definitivo
      pcstateloc CON 142                'locazione byte di stato del pc principale (COM1) - Valore Definitivo
      devicesstateloc CON 143           'locazione stato dei dispositivi di comando



'Blocco Memoria Moduli:
INCLUDE "i2cvar.bas"
```

```
'Blocco Main - Configurazione Iniziale:

main:                                   'prime istruzioni
FLAGS = 0                               'reinizializzo il display
CLEAR                                   'azzera le variabili
PAUSE 500                               'pausa iniziale
INPUT radiocts                          'radiocts è un input
INPUT radiorx                           'radiorx è un input
IF hybrid = 0 THEN                      'se uso i moduli easyradio
OUTPUT radiorts                         'radiorts è un output
radiorts = rtsatt                       'abbasso il segnale rts (pic sempre pronto)
ELSE                                    'se uso i moduli ibridi
OUTPUT radioselect                      'radioselect è un output (lo stesso pin di radiorts)
ENDIF
OUTPUT displayonoff                     'displayonoff è un output
LOW displayonoff                        'displayonoff lvello basso
INPUT displaybutton                     'displaybutton è un input
INPUT debugrx                           'debugrx è un input
OUTPUT debugtx                          'debugtx è un output
OUTPUT expled                           'expled è un output
OUTPUT cicalino                         'cicalino è un output
LOW cicalino                            'cicalino impostato livello basso
INPUT expdig0state                      'expdig0state è un input
schermatedisplay = 3                    'seleziono la schermata di benvenuto
GOSUB display                           'visualizzo la schermata di benvenuto
PAUSE 2500                              'pausa iniziale per la sincronizzazione i2c
GOSUB i2cset                            'imposto il modulo i2c
GOTO hardcheck                          'vado al check della memoria


'Blocco Configurazione Moduli:
INCLUDE "i2cset.bas"



'Blocco configurazione Programma Principale:

hardcheck:                              'check della memoria
GOSUB i2copen                           'occupo il bus
GOSUB i2cmemorychk                      'test accesso alla memoria
GOSUB i2cclose                          'libero il bus
PAUSE rampaketpause                     'pausa contro le collisioni sul canale busy
IF i2cmemorychkflag = 1 THEN            'se il check memoria è fallito
GOTO hardcheck                          'continuo a verificare l'accesso alla memoria
ENDIF
GOSUB writeconfig                       'scrivo zeri e configurazioni iniziali nelle locazioni utilizzate
GOSUB chiavecheck                       'setto lo stato del programma
schermatedisplay = 2                    'attivo la schermata di off
GOSUB display                           'visualizzo la schermata di off
GOTO initprg



initprg:                                'inizializzazione
GOSUB writeconfig                       'scrivo la ram
GOSUB chiavecheck                       'setto lo stato del programma
IF picmasteron = 0 THEN GOTO initprg                'se il pic master è disattivato continuo l'inizializzazione
   checkcentral:                                    'attendo che la centrale di comunicazione sia pronta
GOSUB readconfig                        'leggo la ram
IF centralok = 0 THEN                   'se il pic delle comunicazioni non è pronto (incluso anche il pic psx)
   GOTO checkcentral                    'attendo che la centrale comunicazioni sia pronta
ENDIF
schermatedisplay = 0                    'attivo la schermata di on
GOSUB display                           'visualizzo la schermata di on
GOTO maincicle                          'vado al ciclo di programma



maincicle:                              'ciclo di programma
PAUSE pausetrx                          'temporizzazione ciclo (trasmissioni radio)
GOSUB writeradio                        'spedisco via radio
GOSUB readradio                         'ricevo via radio
waitchiave:                             'attendo che la chiave sia in posizione AVV
GOSUB chiavecheck                       'controllo la chiave
IF picmasteron = 0 THEN                 'se il pic master è disattivato
    PAUSE 10                                'piccola pausa per evitare che il display scleri
    GOSUB writeconfig                       'scrivo la ram
    schermatedisplay = 2                    'attivo la schermata di off
    GOSUB display                           'visualizzo la schermata di off
    schermatedisplay = 0                    'attivo la schermata di on
    priorityscreen = 1                      'attivo come schermata ad alta priorità
    GOTO waitchiave                         'attendo che il pic master sia attivato
ENDIF
GOSUB statecheck                        'controllo i dati sugli stati ricevuti via radio (flags e emergenze)
GOSUB writeconfig                       'scrivo la ram
GOSUB readconfig                        'leggo la ram
IF centralok = 0 THEN GOTO initprg      'se il pic delle comunicazioni non è pronto (incluso anche il pic psx)
```

```
GOSUB readmasterdevices              'legge le periferiche del pic master (per esempio tensione batterie)
GOSUB elabdata                       'elaboro i dati ricevuti
GOSUB basenav                        'eseguo l'autonav della base
GOSUB segnalazioni                   'eseguo la sub delle segnaazioni (all'interno sub gestione display)
GOSUB writeconfig                    'scrivo in ram
GOTO maincicle                       'continua ad eseguire il ciclo di programma


readradio:                           'riceve dati via radio
picmastererrricez = 0                'azzero il flag errore ricezione radio
IF hybrid = 1 THEN                   'se ho deciso di usare i moduli ibridi (aurel)
   GOSUB readhybrid                  'leggo i moduli ibridi
ELSE                                 'altrimenti uso i moduli easy-radio
   GOSUB readeasyradio               'leggo i moduli easyradio
ENDIF
RETURN

writeradio:                          'invia i dati via radio
basestatep = 255
IF hybrid = 1 THEN                   'se ho deciso di usare i moduli ibridi (aurel)
   GOSUB writehybrid                 'scrivo i moduli ibridi
ELSE                                 'altrimenti uso i moduli easy-radio
   GOSUB writeeasyradio              'scrivo i moduli easyradio
ENDIF
RETURN


readeasyradio:
'serout2 txpin,sermode,[corebyte,fuffy1cenp,fuffy2cenp,fuffy1dxp,fuffy2dxp,fuffy1sxp,fuffy2sxp,fuffy3p,compassp, _
        'sensorsp,vbattp,exbyte0,exbyte1,exbyte2,exbyte3
dataintemp = 0               'azzero la variabile buffer in ricezione
GOSUB serin2easyradio        'ricevo il primo byte (preambolo1)
IF dataintemp = compreamble1 THEN         'se ho ricevuto il preambolo 1
    GOSUB serin2easyradio                 'ricevo il secondo byte (preambolo2)
    IF dataintemp = compreamble2 THEN         'se ho ricevuto il preambolo 2 leggo tutti i byte in ingresso in
sequenza e li carico nei registri interni
       GOSUB serin2easyradio                  'ricevo il terzo byte (corebyte - byte dato 1)
       corebyte = dataintemp
       GOSUB serin2easyradio                  'ricevo il quarto byte (fuffy1cenp - byte dato 2)
       fuffy1cenp = dataintemp
       GOSUB serin2easyradio                  'ricevo il quinto byte (fuffy2cenp - byte dato 3)
       fuffy2cenp = dataintemp
       GOSUB serin2easyradio                  'ricevo il sesto byte (fuffy1dxp - byte dato 4)
       fuffy1dxp = dataintemp
       GOSUB serin2easyradio                  'ricevo il settimo byte (fuffy2dxp - byte dato 5)
       fuffy2dxp = dataintemp
       GOSUB serin2easyradio                  'ricevo l'ottavo byte (fuffy1sxp - byte dato 6)
       fuffy1sxp = dataintemp
       GOSUB serin2easyradio                  'ricevo il nono byte (fuffy2sxp - byte dato 7)
       fuffy2sxp = dataintemp
       GOSUB serin2easyradio                  'ricevo il decimo byte (fuffy3p - byte dato 8)
       fuffy3p = dataintemp
       GOSUB serin2easyradio                  'ricevo l'undicesimo byte (compassp - byte dato 9)
       compassp = dataintemp
       GOSUB serin2easyradio                  'ricevo il dodicesimo byte (sensorsp - byte dato 10)
       sensorsp = dataintemp
       GOSUB serin2easyradio                  'ricevo il tredicesimo byte (vbattp - byte dato 11)
       vbattp = dataintemp
       GOSUB serin2easyradio                  'ricevo il quattordicesimo byte (exbyte0 - byte dato 12)
       exbyte0 = dataintemp
       GOSUB serin2easyradio                  'ricevo il quindicesimo byte (exbyte1 - byte dato 13)
       exbyte1 = dataintemp
       GOSUB serin2easyradio                  'ricevo il sedicesimo byte (exbyte2 - byte dato 14)
       exbyte2 = dataintemp
       GOSUB serin2easyradio                  'ricevo il diciasettesimo byte (exbyte3 - byte dato 15)
       exbyte3 = dataintemp
       GOSUB serin2easyradio                  'ricevo il diciottesimo byte (postambolo)
       IF dataintemp <> compostamble THEN         'se non ricevo il postambolo correttamente
          GOTO errreadeasyradio                   'vado alla sub errore lettura easy radio
       ENDIF
    ELSE                                      'se non ho ricevuto il preambolo 2
    GOTO errreadeasyradio                     'vado alla sub errore lettura easy radio
    ENDIF
ELSE                                          'se non ho ricevuto il preambolo 2
GOTO errreadeasyradio                        'vado alla sub errore lettura easy radio
ENDIF
RETURN
errreadeasyradio:                            'gestione errore lettura easyradio
picmastererrricez = 1                        'errore lettura radio
RETURN
serin2easyradio:
SERIN2 radiorx,sermode,radiorxtimeout,rxeasytimeout,[dataintemp]          'ricevo un byte
RETURN
rxeasytimeout:
picmastererrricez = 1                        'errore lettura radio
RETURN


writeeasyradio:
'if radiocts = ~ctsatt then  'se la linea cts del modulo indica che il modulo è occupato
'   goto writeeasyradio      'attendo che il modulo si liberi
'endif
SEROUT2 radiotx,sermode,[compreamble1,compreamble2]         'invio il preambolo 1 seguito dal suo complementare
(preambolo 2)
```

```
'Invio tutti i dati in uscita
SEROUT2 radiotx,sermode,[basestatep,joypadbuttonsp,joyanddirezp,motdxpwmp,motsxpwmp,motdownpwmp]
SEROUT2 radiotx,sermode,[compostamble]                    'invio il postambolo
'waitctsfree:                'attesa che il modulo easy radio non sia più occupato
'if radiocts = ~ctsatt then  'se la linea cts del modulo indica che il modulo è occupato
   'goto waitctsfree             'attendo che il modulo si liberi
'endif
RETURN


'Blocco Comunicazione Moduli Ibridi:

readhybrid:
'niente per ora
RETURN


writehybrid:
'niente per ora
RETURN


statecheck:                         'legge i flags di vitale importanza in arrivo dal ponte radio
'Niente per ora
RETURN


readmasterdevices:                   'legge le periferiche del pic master
'Niente per ora
RETURN


elabdata:                            'elaborazione dati ricevuti (ram e radio)
'niente per ora
RETURN

basenav:                             'autonavigazione della base
'niente per ora
RETURN


segnalazioni:                        'sub segnalazioni
IF priorityscreen = 1 THEN           'se devo visualizzare una schermata di alta priorità
   priorityscreen = 0                'azzero il flag di alta priorità
   GOSUB display                     'visualizzo la schermata
   RETURN                            'esco dalla sub
ENDIF
IF dispbuttonstate <> displaybutton THEN  'se è cambiato lo stato del pulsante del display
   dispbuttonstate = displaybutton
   IF displaybutton = 0 THEN              'se il pulsante è stato premuto
       PAUSE antirimbalzo                    'pausa antirimbalzo
       schermatedisplay = schermatedisplay + 1         'passo alla schermata successiva (1 è la prima schermata)
       IF schermatedisplay > ultimaschermatadati THEN    'se sono arrivato all'ultima schermata disponibile per i
dati
          schermatedisplay = 0                   'vado alla prima schermata
       ENDIF
       GOSUB display                                    'organizzo le schermate e le visualizzo
   ENDIF
ENDIF
RETURN


display:                             'visualizza pagine sul display
'se la schermata è la numero 0, visualizzo la schermata di benvenuto. Altre schermate possono essere aggiunte
HIGH displayonoff                                    'attivo la retroilluminazione
BRANCHL schermatedisplay,[stateon,screen1,stateoff,benvenuto]              'aggiungere altre schermate
semplicemente al branch
RETURN

benvenuto:                      'schermata di benvenuto
LCDOUT dcommand,dblinkon,dcommand,dclear,dcommand,dhome," PROGETTO  ARIA",dcommand,dline2,"   Beta  Test", _
       dcommand,dline3,"  Version 1.0a "
RETURN

stateoff:                       'schermata di benvenuto
LCDOUT dcommand,dblinkon,dcommand,dclear,dcommand,dhome," PROGETTO  ARIA",dcommand,dline2,"   Beta  Test", _
       dcommand,dline3,"  Version 1.0a",dcommand,dline4,"   STATE:  OFF "
RETURN

stateon:                        'schermata di benvenuto
LCDOUT dcommand,dblinkon,dcommand,dclear,dcommand,dhome," PROGETTO  ARIA",dcommand,dline2,"   Beta  Test", _
       dcommand,dline3,"  Version 1.0a",dcommand,dline4,"   STATE:  ON "
RETURN

screen1:                        'schermata 1
LCDOUT dcommand,dblinkon,dcommand,dclear,dcommand,dhome," PROGETTO  ARIA",dcommand,dline2,"   Beta  Test", _
       dcommand,dline3,"   SCHERMATA",dcommand,dline4,"    DI PROVA "
RETURN



chiavecheck:
GOSUB setdigital                    'imposto ingressi digitali
IF chiaveint = 1 THEN               'se la chiave non è girata
```

```
   picmasteron = 0                  'indico che il pic master non è attivo
   basestatep = 0                   'la base non è attiva
ELSE                                'se la chiave è girata
   picmasteron = 1                  'indico che il pic master è attivo
   basestatep = 1                   'indico che la base è attiva
ENDIF
RETURN


'Blocco Programma Principale:




'Blocco Subroutines Moduli:
INCLUDE "i2cmod.bas"



'Blocco Subroutines Secondarie Programma Principale:




'Blocco Lettura/Scrittura Memoria RAM:

'---> Lettura Memoria
readconfig:                     'Leggo i dati dalla Ram
GOSUB i2copen                   'occupo il bus
    readloc0:                   'posizione blocco lettura - 0
i2caddr = baseintbyteloc        'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc0            'se si verifica un errore di lettura, rileggo la locazione
corrente
baseintbyte = i2cin             'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc1:                   'posizione blocco lettura - 1
i2caddr = basestateploc         'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc1            'se si verifica un errore di lettura, rileggo la locazione
corrente
basestatep = i2cin              'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc2:                   'posizione blocco lettura - 2
i2caddr = joypadbuttonsploc     'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc2            'se si verifica un errore di lettura, rileggo la locazione
corrente
joypadbuttonsp = i2cin          'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc3:                   'posizione blocco lettura - 3
i2caddr = joyanddirezploc       'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc3            'se si verifica un errore di lettura, rileggo la locazione
corrente
joyanddirezp = i2cin            'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc4:                   'posizione blocco lettura - 4
i2caddr = motdxpwmploc          'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc4            'se si verifica un errore di lettura, rileggo la locazione
corrente
motdxpwmp = i2cin               'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc5:                   'posizione blocco lettura - 5
i2caddr = motsxpwmploc          'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc5            'se si verifica un errore di lettura, rileggo la locazione
corrente
motsxpwmp = i2cin               'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc6:                   'posizione blocco lettura - 6
i2caddr = motdownpwmploc        'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc6            'se si verifica un errore di lettura, rileggo la locazione
corrente
motdownpwmp = i2cin             'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc7:                   'posizione blocco lettura - 7
i2caddr = prioritydevloc            'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc7            'se si verifica un errore di lettura, rileggo la locazione
corrente
prioritydev = i2cin             'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc8:                   'posizione blocco lettura - 8
i2caddr = pcstateloc            'seleziono la locazione da leggere
```

```
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc8            'se si verifica un errore di lettura, rileggo la locazione
corrente
pcstate = i2cin                 'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc9:                   'posizione blocco lettura - 9
i2caddr = devicesstateloc       'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc9            'se si verifica un errore di lettura, rileggo la locazione
corrente
devicesstate = i2cin            'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
GOSUB i2cclose                  'libero il bus
PAUSE rampaketpause             'pausa contro le collisioni sul canale busy
RETURN


'---> Scrittura Memoria
writeconfig:                    'Scrivo i dati in Ram
GOSUB i2copen                   'Occupo il bus
    writeloc0:                  'posizione blocco scrittura - 0
i2caddr = corebyteloc           'seleziono la locazione da scrivere
i2cout = corebyte               'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc0           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc1:                  'posizione blocco scrittura - 1
i2caddr = fuffy1cenploc         'seleziono la locazione da scrivere
i2cout = fuffy1cenp             'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc1           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc2:                  'posizione blocco scrittura - 2
i2caddr = fuffy2cenploc         'seleziono la locazione da scrivere
i2cout = fuffy2cenp             'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc2           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc3:                  'posizione blocco scrittura - 3
i2caddr = fuffy1dxploc          'seleziono la locazione da scrivere
i2cout = fuffy1dxp              'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc3           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc4:                  'posizione blocco scrittura - 4
i2caddr = fuffy2dxploc          'seleziono la locazione da scrivere
i2cout = fuffy2dxp              'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc4           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc5:                  'posizione blocco scrittura - 5
i2caddr = fuffy1sxploc          'seleziono la locazione da scrivere
i2cout = fuffy1sxp              'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc5           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc6:                  'posizione blocco scrittura - 6
i2caddr = fuffy2sxploc          'seleziono la locazione da scrivere
i2cout = fuffy2sxp              'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc6           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc7:                  'posizione blocco scrittura - 7
i2caddr = fuffy3ploc            'seleziono la locazione da scrivere
i2cout = fuffy3p                'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc7           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc8:                  'posizione blocco scrittura - 8
i2caddr = compassploc           'seleziono la locazione da scrivere
i2cout = compassp               'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc8           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc9:                  'posizione blocco scrittura - 9
i2caddr = sensorsploc           'seleziono la locazione da scrivere
i2cout = sensorsp               'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc9           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc10:                 'posizione blocco scrittura - 10
i2caddr = vbattploc             'seleziono la locazione da scrivere
i2cout = vbattp                 'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
```

```
IF i2cackflag = 1 THEN GOTO writeloc10          'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc11:                 'posizione blocco scrittura - 11
i2caddr = exbyte0loc            'seleziono la locazione da scrivere
i2cout = exbyte0                'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc11          'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc12:                 'posizione blocco scrittura - 12
i2caddr = exbyte1loc            'seleziono la locazione da scrivere
i2cout = exbyte1                'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc12          'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc13:                 'posizione blocco scrittura - 13
i2caddr = exbyte2loc            'seleziono la locazione da scrivere
i2cout = exbyte2                'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc13          'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc14:                 'posizione blocco scrittura - 14
i2caddr = exbyte3loc            'seleziono la locazione da scrivere
i2cout = exbyte3                'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc14          'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc15:                 'posizione blocco scrittura - 15
i2caddr = segnalazbyteloc       'seleziono la locazione da scrivere
i2cout = segnalazbyte           'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc15          'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc16:                 'posizione blocco scrittura - 16
i2caddr = picmasterstateloc     'seleziono la locazione da scrivere
i2cout = picmasterstate         'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc16          'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc17:                 'posizione blocco scrittura - 17
i2caddr = baseexpstateloc       'seleziono la locazione da scrivere
i2cout = baseexpstate           'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc17          'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc18:                 'posizione blocco scrittura - 18
i2caddr = basecell1loc          'seleziono la locazione da scrivere
i2cout = basecell1              'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc18          'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc19:                 'posizione blocco scrittura - 19
i2caddr = basecell2loc          'seleziono la locazione da scrivere
i2cout = basecell2              'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc19          'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc20:                 'posizione blocco scrittura - 20
i2caddr = baserssiloc           'seleziono la locazione da scrivere
i2cout = baserssi               'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc20          'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc21:                 'posizione blocco scrittura - 21
i2caddr = baseintbyteloc        'seleziono la locazione da scrivere
i2cout = baseintbyte            'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc21          'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc22:                 'posizione blocco scrittura - 22
i2caddr = vibroloc              'seleziono la locazione da scrivere
i2cout = vibro                  'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc22          'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
GOSUB i2cclose                  'libero il bus
PAUSE rampaketpause             'pausa contro le collisioni sul canale busy
RETURN


'Blocco Subroutine Settaggio Extra:
```

```
setanalog:                    'attivo tutti gli ingressi analogici
'imposto le porte come ingressi analogici
adcon1.0 = 0
adcon1.1 = 0
adcon1.2 = 0
adcon1.3 = 0
'impiosto la giustificazione del risultato a destra
adcon1.7 = 1
PAUSE 10
RETURN

setdigital:                   'attivo tutti gli ingressi digitali (disattivo adc)
'imposto le porte come i/o digitali
adcon1.0 = 1
adcon1.1 = 1
adcon1.2 = 1
adcon1.3 = 0
PAUSE 10
OUTPUT masterled1             'masterled1 è un output
OUTPUT masterled2             'masterled2 è un output
INPUT displaybutton           'displaybutton è un input
PAUSE 10
INPUT chiaveint               'chiaveint è un input
PAUSE 50
RETURN


'Blocco Gestione Errori:
'---> Sub di gestione errori non utilizzate
i2cbusy:
i2cackerror:
i2cmemoryaccesserror:
GOTO hardcheck              'in caso vengano chiamate per sbaglio rimando al check dell'hardware
```

PROGRAMMA PIC PSX (JOYPAD)

```
'Blocco Direttive:
DEFINE SHIFT_PAUSEUS 50            'rallenta le istruzioni shiftin/shiftout


'Blocco Memoria Main Program:

'---> Symbols Porte I/O
SYMBOL cicalino = portb.2            'cicalino piezoelettrico
SYMBOL led1 = porta.2                'led di segnalazione 1 (virtualmente collegato a interr1) - disabilita
vibrazione
SYMBOL led2 = porta.3                'led di segnalazione 2 (virtualmente collegato a interr2)
SYMBOL interr1 = porta.4             'interruttore 1 (virtualmente collegato a led1) - disabilita vibrazione
SYMBOL interr2 = portb.0             'interruttore 2 (virtualmente collegato a led2)

'---> Variabili
genpurpose VAR WORD                  'word per usi generali
           '---> Alias
           gencont VAR genpurpose.Byte0    'contatore cicli for di uso generale

joymotstopdx VAR BIT                      'bit per la segnalazione dello stato di stop del motore destro
joymotstopsx VAR BIT                      'bit per la segnalazione dello stato di stop del motore sinistro
joymotmotodx VAR BIT                      'bit per la segnalazione dello stato di moto del motore destro
joymotmotosx VAR BIT                      'bit per la segnalazione dello stato di moto del motore sinistro


tonehigh VAR BYTE                    'tono alto 1 (tasti funzione, motori...)
tonehigh2 VAR BYTE                   'tono alto 2 (velocità motori ascensionali)
tonehigh3 VAR BYTE                   'tono alto 2 (velocità motori ascensionali)
tonelow VAR BYTE                     'tono basso (tasti funzione, motori...)
tonepush VAR BYTE                    'tono pressione pulsante non funzionale
tonecroce VAR BYTE                   'tono pressione tasti direzionali (con ripetizione)
tonestse VAR BYTE                    'tono pressione tasti start e select
tonemin VAR BYTE                     'tono minimo (per il jingle)
tonemax VAR BYTE                     'tono massimo(per il jingle)

statebyte VAR BYTE                    'byte flags di stato pic psx
          '--> Alias
          picpsxon VAR statebyte.0           'indica che il pic psx è attivo
          joynotconnected VAR statebyte.1    'indica che il joypad non è connesso o si sono verificati problemi
hardware



vibro VAR BYTE                        'byte vibrazione joypad
centralstate VAR BYTE                 'byte di stato centrale di comunicazione (pic comunicazioni)
             '---> Alias
             centralon VAR centralstate.0     'indica che il pic gestore delle comunicazioni è attivo
             psxdisable VAR centralstate.1        'interrompe l'esecuzione del programma (disabilita modulo psx)
             disablevib VAR centralstate.2        'se attivo disabilita la vibrazione anche se attivata
manualmente

joypadbuttons VAR BYTE                'pulsanti joypad funzioni extra (vedi alias per descrizione dettagliata)
joyanddirez VAR BYTE                  'byte croce joypad e direzione motori
pwmmotdx VAR BYTE                     'valore velocità motore destro (visto da dietro la gondola)
pwmmotsx VAR BYTE                     'valore velocità motore sinistro (visto da dietro la gondola)
pwmmotdown VAR BYTE                   'valore velocità motori ascensionali
           '---> Alias
           keyup VAR joyanddirez.0              'bit tasto freccia in su (telecamera asse y)
           keyright VAR joyanddirez.1           'bit tasto freccia a destra (telecamera asse x)
           keydown VAR joyanddirez.2            'bit tasto freccia in basso (telecamera asse y)
           keyleft VAR joyanddirez.3            'bit tasto freccia a sinistra (telecamera asse x)
           keycroce VAR joypadbuttons.0         'bit tasto Croce
           keyquadrato VAR joypadbuttons.1      'bit tasto Quadrato
           keytriangolo VAR joypadbuttons.2     'bit tasto Triangolo
           keycerchio VAR joypadbuttons.3       'bit tasto Cerchio
           keyStart VAR joypadbuttons.4         'bit tasto Start (conferme autonav)
           keySelect VAR joypadbuttons.5        'bit tasto Select
           keyR1fun VAR joypadbuttons.6         'bit tasto R1 - funzione (telecamera on/off)
           keyR2fun VAR joypadbuttons.7         'bit tasto R2 - funzione (puntatore laser on/off)
           direzmotdx VAR joyanddirez.4         'direzione motore destro (motore 1)
           direzmotsx VAR joyanddirez.5         'direzione motore sinistro (motore 2)
           direzmotdown VAR joyanddirez.6       'direzione motori ascensionali (sotto)
'Il joypad fornisce i valori dei tasti invertiti come pure i valori delle leve analogiche.
'Il programma inverte tutto, quindi i pulsanti sono da intentersi premuti con un livello alto.



'---> Costanti
sogliasmallm CON 20                   'soglia per l'attivazione del motore vibrante piccolo
anticollisiondelay CON 10             'valore in microsecondi tra un accesso e l'altro della memoria
rampaketpause CON 70                      'valore in millisecondi di pausa tra ogni pacchetto scritto/letto in ram
antirimbalzo CON 60                   'pausa antirimbalzo per i tasti (in ms)
veldown1 CON 80                       'velocità 1 motori ascensionali
veldown2 CON 180                      'velocità 2 motori ascensionali
veldown3 CON 255                      'velocità 3 motori ascensionali
direzsu CON 0                         'valore direzione per salire
direzgiu CON 1                        'valore direzione per scendere
direzavanti CON 0                     'valore direzione per andare avanti
direzindietro CON 1                   'valore direzione per andare indietro
campoav CON 30                        'capo avanti (la direzione avanti scatta per un valore della leva analogica
> della metà + campoav)
campoind CON 30                       'capo indietro (la direzione indietro scatta per un valore della leva
analogica < della metà - campoind)
'Suoni e tempi segnalazione cicalino
```

```
tonesilence CON 0                   'silenzio
tonehighset CON 112                 'impostazione tono alto 1 (tasti funzione, motori...)
tonehigh2set CON 118                'impostazione tono alto 2 (velocità motori ascensionali)
tonehigh3set CON 121                'impostazione tono alto 2 (velocità motori ascensionali)
tonelowset CON 110                  'impostazione tono basso (tasti funzione, motori...)
tonepushset CON 115                 'impostazione tono pressione pulsante non funzionale
tonecroceset CON 123                'impostazione tono pressione tasti direzionali (con ripetizione)
tonestseset CON 105                 'impostazione tono pressione tasti start e select
toneminset CON 100                  'impostazione tono minimo (per il jingle)
tonemaxset CON 126                  'impostazione tono massimo(per il jingle)
toneduration CON 9                  'durata dei toni (in unità di 12ms)
silenceduration CON 1               'durata del silenzio (in unità di 12ms)
silencedurationms CON 10            'durata del silenzio (in ms)
silencejingle CON 5                 'durata del silenzio tra le note del jingle (in unità di 12ms)
pausejingle CON 300                 'pausa effetto per il jingle (in ms)


'Blocco Locazioni Ram:
'---> In scrittura
statebyteloc CON 2                   'locazione byte di stato
joypadbuttonsloc CON 3               'locazione byte tasti joypad
joyanddirezloc CON 4                 'locazione byte tasti croce joypad e direzione motori
pwmmotdxloc CON 5                    'velocità motore destro
pwmmotsxloc CON 6                    'velocità motore sinistro
pwmmotdownloc CON 7                  'velocità motori ascensionali

'---> In lettura
centralstateloc CON 129              'locazione byte di stato pic centrale
vibroloc CON 130                     'locazione valore vibrazione joypad


'Blocco memoria moduli:
INCLUDE "psxvar.bas"                 'variabili modulo psx
INCLUDE "i2cvar.bas"                 'variabili modulo i2c



'Blocco Main - Configurazione princpale:

main:                                'programma di configurazione iniziale
PAUSE 500                            'pausa iniziale
CLEAR                                'azzero le variabili
ANSEL = 0                            'imposto le prte analogiche come i/o digitale
OUTPUT cicalino                      'cicalino è un output
LOW cicalino                         'cicalino livello basso
OUTPUT led1                          'led1 è un output
OUTPUT led2                          'led2 è un output
INPUT interr1                        'interr1 è un input
INPUT interr2                        'interr2 è un imput
GOSUB psxchkmode                     'setto il joypad psx su modalità analogica
IF psxid = 0 THEN                    'se il joypad non è connesso
joynotconnected = 1                  'indico che il joypad non è connesso
ELSE                                 'altrimenti
joynotconnected = 0                  'indico che il joypad è connesso
ENDIF
GOSUB i2cset                         'setto il modulo i2c
GOTO hardcheck                       'vado al check dell'hardware


'Blocco Configurazione Moduli:
INCLUDE "psxset.bas"                 'settaggio modulo psx
INCLUDE "i2cset.bas"                 'settaggio modulo i2c


'Blocco Configurazione programma principale:

hardcheck:                           'test hardware
GOSUB i2copen                        'occupo il bus
GOSUB i2cmemorychk                   'test accesso alla memoria
GOSUB i2cclose                       'libero il bus
PAUSE rampaketpause                  'pausa contro le collisioni sul canale busy
IF i2cmemorychkflag = 1 THEN         'se il check memoria è fallito
GOTO hardcheck                       'continuo a verificare l'accesso alla memoria
ENDIF
GOSUB writeconfig                    'scrivo zeri e configurazioni iniziali nelle locazioni utilizzate
GOTO initprg                         'inizializzazione del programma


initprg:                             'sub di inizializzazione
GOSUB readconfig                     'leggo la ram
IF centralon = 0 OR psxdisable = 1 THEN             'se il pic gestore non è attivo o il modulo psx è stato
disabilitato
GOTO initprg                         'ritorno all'inizializzazione
ENDIF
GOSUB cicalstart                     'esegue un jingle iniziale con il cicalino (se l'impostazione è attiva)
led1 = interr1                       'visualizzo l'impostazione attivata con l'interruttore 1 con il led
corrispondente
led2 = interr2                       'visualizzo l'impostazione attivata con l'interruttore 2 con il led
corrispondente
GOTO maincicle                       'vado al ciclo di programma
```

```
'Blocco Programma principale:

maincicle:                          'ciclo di programma - esegue la struttura del programma
GOSUB readconfig                    'leggo la ram
IF centralon = 0 OR psxdisable = 1 THEN             'se il pic gestore non è attivo o il modulo psx è stato
disabilitato
GOTO initprg                        'ritorno all'inizializzazione
ENDIF
GOSUB setvib                        'imposto i valori dei motori vibranti in base al valore letto in ram
GOSUB setsound                      'attivo/disattivo gli effetti sonori
GOSUB psxchkmode                    'setto il joypad psx su modalità analogica
IF psxid = 0 THEN                   'se il joypad non è connesso
joynotconnected = 1                 'indico che il joypad non è connesso
GOTO exitwithoutread                'non leggo il joypad (perchè non connesso) ma scrivo i dati in ram (comunico
stato)
ELSE                                'altrimenti
joynotconnected = 0                 'indico che il joypad è connesso
ENDIF
GOSUB readpsxvib                    'leggo il controller e lo faccio vibrare
GOSUB elabdata                      'elaboro i dati ricevuti
exitwithoutread:                    'punto di scrittura dati in ram (salto della lettura del joypad)
picpsxon = 1                        'indico che il pic psx è attivo
GOSUB writeconfig                   'scrivo i dati in ram
GOTO maincicle                      'rieseguo il ciclo di programma all'infinito


elabdata:                           'elaboro i dati letti dal joypad e segnalo le impostazioni con il cicalino
(se attivo)
'Inverto i valori letti quindi: tasti attivi alti, valori analogici da 255 a 0
digi1 = ~digi1
digi2 = ~digi2
analog1 = ~analog1
analog2 = ~analog2
analog3 = ~analog3
analog4 = ~analog4
'emetto un suono se viene premuto uno dei pulsanti non funzionali
IF keycroce = 0 AND keyX = 1 THEN         'se viene premuto il pulsante
SOUND cicalino, [tonepush,toneduration]         'emetto un suono
LOW cicalino                                    'disattivo il cicalino
ENDIF
IF keyquadrato = 0 AND keyQ = 1 THEN      'se viene premuto il pulsante
SOUND cicalino, [tonepush,toneduration]         'emetto un suono
LOW cicalino                                    'disattivo il cicalino
ENDIF
IF keytriangolo = 0 AND keyT = 1 THEN     'se viene premuto il pulsante
SOUND cicalino, [tonepush,toneduration]         'emetto un suono
LOW cicalino                                    'disattivo il cicalino
ENDIF
IF keycerchio = 0 AND keyC = 1 THEN       'se viene premuto il pulsante
SOUND cicalino, [tonepush,toneduration]         'emetto un suono
LOW cicalino                                    'disattivo il cicalino
ENDIF
IF keystart = 0 AND keyST = 1 THEN        'se viene premuto il pulsante
SOUND cicalino, [tonestse,toneduration]         'emetto un suono
LOW cicalino                                    'disattivo il cicalino
ENDIF
IF keyselect = 0 AND keySL = 1 THEN       'se viene premuto il pulsante
SOUND cicalino, [tonestse,toneduration]         'emetto un suono
LOW cicalino                                    'disattivo il cicalino
ENDIF
'se viene premuto uno dei tasti direzionali emetto un suono anche se mantenuto premuto
IF keyU = 1 OR keyD = 1 OR keyL = 1 OR keyR = 1 THEN    'se viene premuto un tasto direzionale
SOUND cicalino, [tonecroce,toneduration,tonesilence,silenceduration]        'emetto un suono ripetutamente
LOW cicalino                                                                'disattivo il cicalino
ENDIF
'assegno i valori dei tasti non funzionali alle variabili in uscita
keycroce = keyX
keyquadrato = keyQ
keytriangolo = keyT
keycerchio = keyC
keystart = keyST
keyselect = keySL
keyup = keyU
keydown = keyD
keyleft = keyL
keyright = keyR
'Assegno i valori dei tasti funzionali (modulo ciclopè) alle variabili in uscita
IF keyR1 = 1 THEN           'se viene premuto il tasto R1
keyr1fun = ~keyr1fun        'inverto il valore del tasto funzioe R1
        IF keyr1fun = 1 THEN        'se il tasto è "attiva funzione"
        SOUND cicalino, [tonelow,toneduration,tonesilence,silenceduration,tonehigh,toneduration]         'emetto
una scala positiva
        LOW cicalino                                    'disattivo il cicalino
        ELSE
        SOUND cicalino, [tonehigh,toneduration,tonesilence,silenceduration,tonelow,toneduration]         'emetto
una scala negativa
        LOW cicalino                                    'disattivo il cicalino
        ENDIF
        'il suono emesso fa anche da pausa antirimbalzo
ENDIF
IF keyR2 = 1 THEN           'se viene premuto il tasto R2
keyr2fun = ~keyr2fun        'inverto il valore del tasto funzioe R2
```

```
         IF keyr2fun = 1 THEN        'se il tasto è "attiva funzione"
         SOUND cicalino, [tonelow,toneduration,tonesilence,silenceduration,tonehigh,toneduration]         'emetto
una scala positiva
         LOW cicalino                                    'disattivo il cicalino
         ELSE
         SOUND cicalino, [tonehigh,toneduration,tonesilence,silenceduration,tonelow,toneduration]         'emetto
una scala negativa
         LOW cicalino                                    'disattivo il cicalino
         ENDIF
         'il suono emesso fa anche da pausa antirimbalzo
ENDIF
'Scelgo la velocità e la direzione dei motori ascensionali in base alle pressioni dei tasti L1 e L2 (3 velocità)
IF keyL1 = 1 THEN             'se è stato premuto il tasto "motori ascensionali su"
   IF pwmmotdown = 0 THEN     'se la velocità è 0
   direzmotdown = direzsu     'imposto la direzione per la salita
   ENDIF
   IF direzmotdown = direzsu THEN      'se la direzione è quella di salita
      IF pwmmotdown = 0 THEN               'se la velocità è 0
      pwmmotdown = veldown1                'imposto il livello di velocità 1
      SOUND cicalino, [tonehigh,toneduration]       'emetto il suono di "velocità uno"
      LOW cicalino                                  'disattivo il cicalino
      GOTO exitsetvelL1                             'esco da''impostazione della velocità
      ENDIF
      IF pwmmotdown = veldown1 THEN        'se la velocità è "velocità 1"
      pwmmotdown = veldown2                'imposto il livello di velocità 2
      SOUND cicalino, [tonehigh2,toneduration]      'emetto il suono di "velocità due"
      LOW cicalino                                  'disattivo il cicalino
      GOTO exitsetvelL1                             'esco da''impostazione della velocità
      ENDIF
      IF pwmmotdown = veldown2 THEN        'se la velocità è "velocità 2"
      pwmmotdown = veldown3                'imposto il livello di velocità 3
      SOUND cicalino, [tonehigh3,toneduration]      'emetto il suono di "velocità tre"
      LOW cicalino                                  'disattivo il cicalino
      GOTO exitsetvelL1                             'esco da''impostazione della velocità
      ENDIF
      IF pwmmotdown = veldown3 THEN        'se la velocità è "velocità 3"
      pwmmotdown = veldown2                'imposto il livello di velocità 2
      SOUND cicalino, [tonehigh2,toneduration]      'emetto il suono di "velocità due"
      LOW cicalino                                  'disattivo il cicalino
      GOTO exitsetvelL1                             'esco da''impostazione della velocità
      ENDIF
      exitsetvelL1:         'uscita settaggio velocità L1
   ELSE                                 'se la direzione è quella di discesa
   pwmmotdown = 0                       'azzero la velocità
   SOUND cicalino, [tonelow,toneduration]       'emetto il suono di "velocità zero"
   LOW cicalino                                 'disattivo il cicalino
   ENDIF
PAUSE antirimbalzo                              'pausa antirimbalzo
ENDIF
IF keyL2 = 1 THEN             'se è stato premuto il tasto "motori ascensionali giù"
   IF pwmmotdown = 0 THEN     'se la velocità è 0
   direzmotdown = direzgiu    'imposto la direzione per la discesa
   ENDIF
   IF direzmotdown = direzgiu THEN     'se la direzione è quella di discesa
      IF pwmmotdown = 0 THEN               'se la velocità è 0
      pwmmotdown = veldown1                'imposto il livello di velocità 1
      SOUND cicalino, [tonehigh,toneduration]       'emetto il suono di "velocità uno"
      LOW cicalino                                  'disattivo il cicalino
      GOTO exitsetvelL2                             'esco da''impostazione della velocità
      ENDIF
      IF pwmmotdown = veldown1 THEN        'se la velocità è "velocità 1"
      pwmmotdown = veldown2                'imposto il livello di velocità 2
      SOUND cicalino, [tonehigh2,toneduration]      'emetto il suono di "velocità due"
      LOW cicalino                                  'disattivo il cicalino
      GOTO exitsetvelL2                             'esco da''impostazione della velocità
      ENDIF
      IF pwmmotdown = veldown2 THEN        'se la velocità è "velocità 2"
      pwmmotdown = veldown3                'imposto il livello di velocità 3
      SOUND cicalino, [tonehigh3,toneduration]      'emetto il suono di "velocità tre"
      LOW cicalino                                  'disattivo il cicalino
      GOTO exitsetvelL2                             'esco da''impostazione della velocità
      ENDIF
      IF pwmmotdown = veldown3 THEN        'se la velocità è "velocità 3"
      pwmmotdown = veldown2                'imposto il livello di velocità 2
      SOUND cicalino, [tonehigh2,toneduration]      'emetto il suono di "velocità due"
      LOW cicalino                                  'disattivo il cicalino
      GOTO exitsetvelL2                             'esco da''impostazione della velocità
      ENDIF
      exitsetvelL2:         'uscita settaggio velocità L2
   ELSE                                 'se la direzione è quella di salita
   pwmmotdown = 0                       'azzero la velocità
   SOUND cicalino, [tonelow,toneduration]       'emetto il suono di "velocità zero"
   LOW cicalino                                 'disattivo il cicalino
   ENDIF
PAUSE antirimbalzo                              'pausa antirimbalzo
ENDIF
'Calcolo la velocità dei motori direzionali in base al valore delle leve analogiche
IF analog2 > 127 THEN               'se il valore della leva analogica destra è > di 127
pwmmotdx = analog2 - 128            'sottraggo 127 ottenendo una scala da 0 a 127
pwmmotdx = pwmmotdx * 2             'moltiplico il valore ottenuto per 2 così da ottenere una scala da 0 a 254
(di 2 in 2)
ELSE                                'se il valore della leva è inferiore a 127
pwmmotdx = analog2 * 2              'la scala è già da 0 a 127 quindi moltiplico solo per 2
pwmmotdx = ~ pwmmotdx               'inverto la scala perchè la leva va verso il basso
```

```
ENDIF
IF analog4 > 127 THEN                'se il valore della leva analogica sinistra è > di 127
pwmmotsx = analog4 - 128             'sottraggo 127 ottenendo una scala da 0 a 127
pwmmotsx = pwmmotsx * 2              'moltiplico il valore ottenuto per 2 così da ottenere una scala da 0 a 254
(di 2 in 2)
ELSE                                 'se il valore della leva è inferiore a 127
pwmmotsx = analog4 * 2               'la scala è già da 0 a 127 quindi moltiplico solo per 2
pwmmotsx = ~ pwmmotsx                'inverto la scala perchè la leva va verso il basso
ENDIF
'Scelgo la direzione dei motori in base alle leve analogiche (con campo di inattività intermedio)
IF analog2 > (127 + campoav) THEN    'se il valore della leva analogica destra è maggiore della metà più il campo
alto
    joymotstopdx = 0                     'imposto a 0 la variabile di stop motore destro
    direzmotdx = direzavanti             'imposto la direzione avanti
    IF joymotmotodx = 0 THEN             'se la variabile di moto motore destro è 0
    SOUND cicalino, [tonehigh,toneduration]     'emetto il suono di "motore in movimento"
    LOW cicalino                                'disattivo il cicalino
    ENDIF
    joymotmotodx = 1                            'imposto la variabile di moto motore destro a 1
ELSE
    IF analog2 < (127 - campoind) THEN   'se il valore della leva analogica destra è minore della metà meno il
campo basso
        joymotstopdx = 0                    'imposto a 0 la variabile di stop motore destro
        direzmotdx = direzindietro          'imposto la direzione indietro
        IF joymotmotodx = 0 THEN            'se la variabile di moto motore destro è 0
        SOUND cicalino, [tonehigh,toneduration]     'emetto il suono di "motore in movimento"
        LOW cicalino                                'disattivo il cicalino
        ENDIF
        joymotmotodx = 1                            'imposto la variabile di moto motore destro a 1
    ELSE                                'altrimenti
        joymotmotodx = 0                    'azzero la variabile di moto motore destro
        pwmmotdx = 0                        'azzero la velocità del motore destro
        IF joymotstopdx = 0 THEN                'se il valore della variabile di stop motore destro è 0
        SOUND cicalino, [tonelow,toneduration]      'emetto il suono di "motore non in movimento"
        LOW cicalino                                'disattivo il cicalino
        ENDIF
        joymotstopdx = 1                            'imposto la variabile di stop motore destro a 1
    ENDIF
ENDIF
IF analog4 > (127 + campoav) THEN    'se il valore della leva analogica sinistra è maggiore della metà più il
campo alto
    joymotstopsx = 0                     'imposto a 0 la variabile di stop motore sinistro
    direzmotsx = direzavanti             'imposto la direzione avanti
    IF joymotmotosx = 0 THEN             'se la variabile di moto motore sinistro è 0
    SOUND cicalino, [tonehigh,toneduration]     'emetto il suono di "motore in movimento"
    LOW cicalino                                'disattivo il cicalino
    ENDIF
    joymotmotosx = 1                            'imposto la variabile di moto motore sinistro a 1
ELSE
    IF analog4 < (127 - campoind) THEN   'se il valore della leva analogica sinistra è minore della metà meno il
campo basso
        joymotstopsx = 0                    'imposto a 0 la variabile di stop motore sinistro
        direzmotsx = direzindietro          'imposto la direzione indietro
        IF joymotmotosx = 0 THEN            'se la variabile di moto motore sinistro è 0
        SOUND cicalino, [tonehigh,toneduration]     'emetto il suono di "motore in movimento"
        LOW cicalino                                'disattivo il cicalino
        ENDIF
        joymotmotosx = 1                            'imposto la variabile di moto motore sinistro a 1
    ELSE                                'altrimenti
        joymotmotosx = 0                    'azzero la variabile di moto motore sinistro
        pwmmotsx = 0                        'azzero la velocità del motore sinistro
        IF joymotstopsx = 0 THEN                'se il valore della variabile di stop motore sinistro è 0
        SOUND cicalino, [tonelow,toneduration]      'emetto il suono di "motore non in movimento"
        LOW cicalino                                'disattivo il cicalino
        ENDIF
        joymotstopsx = 1                            'imposto la variabile di stop motore sinistro a 1
    ENDIF
ENDIF
RETURN



'Blocco Subroutines Moduli:
INCLUDE "psx.bas"                    'modulo psx - subroutines
INCLUDE "i2cmod.bas"                 'modulo i2c - soubroutines




'Blocco subroutines secondarie Programma Principale:


setvib:                          'imposto i valori di vibrazione
led1 = interr1                   'visualizzo l'impostazione attivata con l'interruttore 1 con il led corrispondente
vibro = 255
IF interr1 = 0 THEN              'se l'interruttore è chiuso (disattiva vibrazione)
vibro = 0                        'azzero la vibrazione
ENDIF
IF disablevib = 1 THEN           'se la vibrazione è da disabilitare (dalle alte sfere)
vibro = 0                        'azzero la vibrazione
led1 = 0                         'visualizzo l'assenza di vibrazione
ENDIF
largem = vibro                   'la potenza del motore grande è data dal valore letto dalla ram
```

```
IF vibro > sogliasmallm THEN      'se la velocità impostata è superiore alla soglia
smallm = 1                      'accendo il motore piccolo
ELSE                                'altrimenti
smallm = 0                      'lo spengo
ENDIF
RETURN


setsound:                       'attivo/disattivo gli effetti sonori
led2 = interr2                  'visualizzo l'impostazione attivata con l'interruttore 2 con il led corrispondente
IF interr2 = 0 THEN             'se l'interruttore è chiuso (disattiva suono)
'imposto un suono vuoto (silenzio) per tutti i suoni del programma
tonehigh = tonesilence
tonehigh2 = tonesilence
tonehigh3 = tonesilence
tonelow = tonesilence
tonepush = tonesilence
tonecroce = tonesilence
tonestse = tonesilence
tonemin = tonesilence
tonemax = tonesilence
ELSE                            'altrimenti lo attivo
'carico nelle variabili le impostazioni dei suoni impostate
tonehigh = tonehighset
tonehigh2 = tonehigh2set
tonehigh3 = tonehigh3set
tonelow = tonelowset
tonepush = tonepushset
tonecroce = tonecroceset
tonestse = tonestseset
tonemin = toneminset
tonemax = tonemaxset
ENDIF
RETURN



cicalstart:                     'esegue un jingle con il cicalino
'carico nelle variabili le impostazioni dei suoni impostate
tonemin = toneminset
tonemax = tonemaxset
FOR gencont = tonemin TO tonemax
SOUND cicalino, [gencont,toneduration,tonesilence,silencejingle]
LOW cicalino
NEXT gencont
PAUSE pausejingle
SOUND cicalino, [tonemin,toneduration]
LOW cicalino
RETURN

'Blocco Lettura/Scrittura Ram i2c:

writeconfig:                      'scrittura memoria ram
GOSUB i2copen
    writeloc0:                    'posizione blocco scrittura - 0
i2caddr = statebyteloc            'seleziono la locazione da scrivere
i2cout = statebyte           'carico il valore in uscita
GOSUB i2ctx                       'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc0           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay        'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc1:                    'posizione blocco scrittura - 1
i2caddr = joypadbuttonsloc        'seleziono la locazione da scrivere
i2cout = joypadbuttons            'carico il valore in uscita
GOSUB i2ctx                       'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc1           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay        'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc2:                    'posizione blocco scrittura - 2
i2caddr = joyanddirezloc          'seleziono la locazione da scrivere
i2cout = joyanddirez              'carico il valore in uscita
GOSUB i2ctx                       'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc2           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay        'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc3:                    'posizione blocco scrittura - 3
i2caddr = pwmmotdxloc             'seleziono la locazione da scrivere
i2cout = pwmmotdx                 'carico il valore in uscita
GOSUB i2ctx                       'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc3           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay        'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc4:                    'posizione blocco scrittura - 4
i2caddr = pwmmotsxloc             'seleziono la locazione da scrivere
i2cout = pwmmotsx                 'carico il valore in uscita
GOSUB i2ctx                       'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc4           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay        'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc5:                    'posizione blocco scrittura - 5
i2caddr = pwmmotdownloc           'seleziono la locazione da scrivere
i2cout = pwmmotdown               'carico il valore in uscita
GOSUB i2ctx                       'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc5           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
```

```
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
GOSUB i2cclose
PAUSE rampaketpause
RETURN


readconfig:                     'lettura memoria ram
GOSUB i2copen                   'occupo il bus
    readloc0:                   'posizione blocco lettura - 0
i2caddr = centralstateloc       'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc0            'se si verifica un errore di lettura, rileggo la locazione
corrente
centralstate = i2cin            'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc1:                   'posizione blocco lettura - 1
i2caddr = vibroloc              'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc1            'se si verifica un errore di lettura, rileggo la locazione
corrente
vibro = i2cin                   'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
GOSUB i2cclose                  'libero il bus
PAUSE rampaketpause             'pausa contro le collisioni sul canale busy
RETURN


'Blocco gestione errori:
'---> Sub di gestione errori non utilizzate
i2cbusy:
i2cackerror:
i2cmemoryaccesserror:
GOTO hardcheck              'in caso vengano chiamate per sbaglio rimando al check dell'hardware
```

PROGRAMMA PIC GESTORE COMUNICAZIONE DISPOSITIVI

```
'Blocco direttive:
DEFINE SHIFT_PAUSEUS 50            'rallenta le istruzioni shiftin/shiftout



'Blocco memoria Programma Principale:

'---> Symbols - Definizione Porte I/O
SYMBOL led1 = portb.6              'led di segnalazione 1 (attivato via ram)
SYMBOL led2 = portb.7              'led di segnalazione 1 (attivato via ram)
SYMBOL ledcom2 = porta.4           'led di segnalazione com 2 attiva (collegato virtualmente a intcom2)
SYMBOL intcom2 = porta.0           'interruttore attivazione com 2 (collegato virtualmente a ledcom2)
SYMBOL com1tx  = portb.5           'dati in uscita (verso il pc) - Porta 1
SYMBOL com1rx = portb.2            'dati in ingresso (dal pc) - Porta 1
SYMBOL com1cts = portb.0           'segnale CTS Locale (Input) - PC Pronto a Ricevere (e attivo) --> RTS del PC
(Out)
SYMBOL com1rts = porta.3           'segnale RTS Locale (Output) - PIC Pronto a Ricevere (e attivo) --> CTS del PC
(In)
SYMBOL com2rx = porta.2            'dati in uscita (verso il pc) - Porta 2
SYMBOL com2tx = porta.1            'dati in ingresso (dal pc) - Porta 1


'---> Variabili
      '---> Modulo joypad
      statebytej VAR BYTE                    'byte flags di stato pic psx
          '--> Alias
          picpsxon VAR statebytej.0          'indica che il pic psx è attivo
          joynotconnected VAR statebytej.1   'indica che il joypad non è connesso o si sono verificati problemi
hardware


      '---> Pacchetti di comunicazione

      datapach VAR BYTE[23]                'pacchetto dati dirigibile e base inviati al pc
      '---> Alias Pacchetto
      '---> Dati Dirigibile
      corebyte VAR datapach[0]             'Parte Bassa Word di Stato Core
      fuffy1cenp VAR datapach[1]           'dato fuffy1 centro
      fuffy2cenp VAR datapach[2]           'dato fuffy2 centro
      fuffy1dxp VAR datapach[3]            'dato fuffy1 destra
      fuffy2dxp VAR datapach[4]            'dato fuffy2 destra
      fuffy1sxp VAR datapach[5]            'dato fuffy1 sinistra
      fuffy2sxp VAR datapach[6]            'dato fuffy2 sinistra
      fuffy3p VAR datapach[7]              'dato fuffy3 sotto
      compassp VAR datapach[8]             'valore compresso della bussola digitale
      sensorsp VAR datapach[9]             'valori dei sensori (infrarossi) e altro
      vbattp VAR datapach[10]              'livelli batterie pacchettati
      exbyte0 VAR datapach[11]             'byte aggiuntivo esterno 0 (espansioni)
      exbyte1 VAR datapach[12]             'byte aggiuntivo esterno 1 (espansioni)
      exbyte2 VAR datapach[13]             'byte aggiuntivo esterno 2 (espansioni)
      exbyte3 VAR datapach[14]             'byte aggiuntivo esterno 3 (espansioni)
      '---> Dati Base Terrestre
      devicesstate VAR datapach[15]        'stato dei dispositivi di comando
      segnalazbyte VAR datapach[16]        'byte flag di segnalazione (leds etc)
      picmasterstate VAR datapach[17]      'byte di stato del pic centrale (pic master) della base
      baseexpstate VAR datapach[18]        'byte di stato e comunicazione espansione base
      baseintbyte VAR datapach[19]         'byte per usi interni base
      basecell1 VAR datapach[20]           'byte tensione pacchettata in 8bit cella 1 pacco base
      basecell2 VAR datapach[21]           'byte tensione pacchettata in 8bit cella 2 pacco base
      baserssi VAR datapach[22]            'valore rssi base terrstre (valore tensione pacchettato in 8 bit)
                  '---> Alias Stati Dispositivi
                  joystate VAR devicesstate.0       'stato del joypad (0 = non disponibile)
                  com1state VAR devicesstate.1      'stato della COM 1 (0 = non disponibile)
                  com2state VAR devicesstate.2      'stato della COM 2 (0 = non disponibile)
                  devicesfail VAR devicesstate.3    'nessuna periferica attiva (valore 1)
                  com1error VAR devicesstate.4      'errore lettura com1
                  com2error VAR devicesstate.5      'errore di lettura com2
                  comunicerror VAR devicesstate.6   'errore "lettura fallita da tutti i dispositivi connessi"

                  '---> Alias byte segnalazione
                  led1cstate VAR segnalazbyte.0     'stato del led 1 - pic comunicazioni
                  led2cstate VAR segnalazbyte.1     'stato del led 2 - pic comunicazioni
                  led1pmstate VAR segnalazbyte.2    'stato del led 1 - pic master
                  led2pmstate VAR segnalazbyte.3    'stato del led 2 - pic master

                  '---> Alias Stati pic master
                  picmasteron VAR picmasterstate.0         'indica che il pic master è attivo
                  picmastererrricez VAR picmasterstate.1   'indica che si è verificato un errore durante la
ricezione radio


      cmdpach VAR BYTE[8]                  'pacchetto comandi inviati alla base ed al dirigibile (incluso stato pc)
      '---> Alias Pacchetto                'questi dati vengono rispediti anche alle periferiche terminali come
riconferma
      '---> Comandi Dirigibile
      basestatep VAR cmdpach[0]            'dato byte di controllo base (per il dirigibile)
      joypadbuttonsp VAR cmdpach[1]        'dato pulsanti joypad (e comandi pc)
      joyanddirezp VAR cmdpach[2]          'croce joypad e direzione motori (e comandi pc)
      motdxpwmp VAR cmdpach[3]             'dato velocità motore destro
      motsxpwmp VAR cmdpach[4]             'dato velocità motore sinistro
      motdownpwmp VAR cmdpach[5]           'dato velocità motori ascensionali
      '---> Comandi e variabili locali
```

```
      prioritydev VAR cmdpach[6]          'byte di priorità con flags di priorità e controllo joypad
      pcstate VAR cmdpach[7]              'byte di stato del pc principale (COM1)
              '---> Alias pulsanti joypad e direzioni
               keyup VAR joyanddirezp.0             'bit tasto freccia in su (telecamera asse y)
               keyright VAR joyanddirezp.1          'bit tasto freccia a destra (telecamera asse x)
               keydown VAR joyanddirezp.2           'bit tasto freccia in basso (telecamera asse y)
               keyleft VAR joyanddirezp.3           'bit tasto freccia a sinistra (telecamera asse x)
               keycroce VAR joypadbuttonsp.0        'bit tasto Croce
               keyquadrato VAR joypadbuttonsp.1     'bit tasto Quadrato
               keytriangolo VAR joypadbuttonsp.2    'bit tasto Triangolo
               keycerchio VAR joypadbuttonsp.3      'bit tasto Cerchio
               keyStart VAR joypadbuttonsp.4        'bit tasto Start (conferme autonav)
               keySelect VAR joypadbuttonsp.5       'bit tasto Select
               keyR1fun VAR joypadbuttonsp.6        'bit tasto R1 - funzione (telecamera on/off)
               keyR2fun VAR joypadbuttonsp.7        'bit tasto R2 - funzione (puntatore laser on/off)
               direzmotdx VAR joyanddirezp.4        'direzione motore destro (motore 1)
               direzmotsx VAR joyanddirezp.5        'direzione motore sinistro (motore 2)
               direzmotdown VAR joyanddirezp.6      'direzione motori ascensionali (sotto)
               'tutti i pulsanti attivi alti
               '---> Alias dei flag di priorità
               centralstatej VAR prioritydev       'alias variabile usata per la comunicazione con il joypad (in
pratica uso il byte prioritydev)
               centralon VAR prioritydev.0         'indica che il pic gestore delle comunicazioni è attivo
               psxdisable VAR prioritydev.1        'interrompe l'esecuzione del programma (disabilita modulo psx)
               disablevib VAR prioritydev.2        'se attivo disabilita la vibrazione anche se attivata
manualmente
               centralok VAR prioritydev.3         'indica che il pic gestore è pronto
               priorityflag VAR disablevib         'il flag di priorità è lo stesso del flag per disabilitare la
vibrazione
                                                   'dato che se il joypad non viene letto, ne disabilito la
vibrazione
                                                   'se disabilitato per lungo periodo sul pc attivo anche
"psxdisable"
               '---> Alias Variabile Stato PC
               rxpcstate VAR pcstate.0             'valore stato pc ricevuto (1 = pc attivo)
               replystate VAR pcstate.1            'valore stato pc confermato (il programma rileva la base)


      preamble1 VAR BYTE                           'valore letto preambolo 1
      preamble2 VAR BYTE                           'valore letto preambolo 2
      postamble VAR BYTE                           'valore letto postambolo




'---> Costanti
ctsatt CON 0                                       'livello logico perchè cts sia attivo (attivo basso quindi)
rtsatt CON 0                                       'livello logico perchè rts sia attivo (attivo basso quindi)
com1mode CON 396                                   'velocità porta com1 (vedi impostazioni serin2)
com1timeout CON 100                                'tempo in (ms) di timeout per la serin2 della com1
com2mode CON 396                                   'velocità porta com2 (vedi impostazioni serin2)
com2timeout CON 200                                'tempo in (ms) di timeout per la serin2 della com2
compreamble1 CON %10101010                         'valore standard preambolo 1
compreamble2 CON %01010101                         'valore standard preambolo 2 (complementare)
compostamble CON %10101010                         'valore standard postambolo
comdatarequest CON %10101010                       'valore inviato dalla com2 per richiedere l'invio di dati
anticollisiondelay CON 10                          'valore in microsecondi tra un accesso e l'altro al bus
rampaketpause CON 70                                   'valore in millisecondi di pausa tra ogni pacchetto
scritto/letto in ram

'Blocco Locazioni Ram:

'---> Locazioni Joypad
    '---> In lettura
    statebytejloc CON 2                 'locazione byte di stato joypad  -  joypad
    joypadbuttonsjloc CON 3             'locazione byte tasti joypad  -  joypad
    joyanddirezjloc CON 4               'locazione byte tasti croce joypad e direzione motori  -  joypad
    pwmmotdxjloc CON 5                  'locazione velocità motore destro  -  joypad
    pwmmotsxjloc CON 6                  'locazione velocità motore sinistro  -  joypad
    pwmmotdownjloc CON 7                'locazione velocità motori ascensionali  -  joypad

    '---> In scrittura
    centralstatejloc CON 129            'locazione byte di stato pic centrale comunicazioni  -  joypad
    vibroloc CON 130                    'locazione valore vibrazione joypad (non usata in questo programma) -
joypad

'---> Locazioni Pic Master
    '---> In  Lettura
    corebyteloc CON 10          'locazione Parte Bassa Word di Stato Core
    fuffy1cenploc CON 11        'locazione dato fuffy1 centro
    fuffy2cenploc CON 12        'locazione dato fuffy2 centro
    fuffy1dxploc CON 13         'locazione dato fuffy1 destra
    fuffy2dxploc CON 14         'locazione dato fuffy2 destra
    fuffy1sxploc CON 15         'locazione dato fuffy1 sinistra
    fuffy2sxploc CON 15         'locazione dato fuffy2 sinistra
    fuffy3ploc CON 17           'locazione dato fuffy3 sotto
    compassploc CON 18          'locazione valore compresso della bussola digitale
    sensorsploc CON 19          'locazione valori dei sensori (infrarossi) e altro
    vbattploc CON 20            'locazione livelli batterie pacchettati
    exbyte0loc CON 21           'locazione byte aggiuntivo esterno 0 (espansioni)
    exbyte1loc CON 22           'locazione byte aggiuntivo esterno 1 (espansioni)
    exbyte2loc CON 23           'locazione byte aggiuntivo esterno 2 (espansioni)
    exbyte3loc CON 24           'locazione byte aggiuntivo esterno 3 (espansioni)
```

```
    segnalazbyteloc CON 25          'locazione byte flag di segnalazione (leds etc)
    picmasterstateloc CON 26        'locazione byte di stato del pic centrale (pic master) della base
    baseexpstateloc CON 27          'locazione byte di stato e comunicazione espansione base
    basecell1loc CON 28             'locazione byte tensione cella 1 base
    basecell2loc CON 29             'locazione byte tensione cella 2 base
    baserssiloc CON 30              'locazione byte valore rssi base

    baseintbyteloc CON 31           'locazione byte per usi interni base (lettura e scrittura)

    '---> In Scrittura
      basestateploc CON 135           'locazione dato byte di controllo base (per il dirigibile) - Valore
Definitivo
      joypadbuttonsploc CON 136       'locazione dato pulsanti joypad (e comandi pc) - Valore Definitivo
      joyanddirezploc CON 137         'locazione croce joypad e direzione motori (e comandi pc) - Valore
Definitivo
      motdxpwmploc CON 138            'locazione dato velocità motore destro - Valore Definitivo
      motsxpwmploc CON 139            'locazione dato velocità motore sinistro - Valore Definitivo
      motdownpwmploc CON 140          'locazione dato velocità motori ascensionali - Valore Definitivo
      prioritydevloc CON 141          'locazione byte di priorità con flags di priorità e controllo joypad -
Valore Definitivo
      pcstateloc CON 142              'locazione byte di stato del pc principale (COM1) - Valore Definitivo
      devicesstateloc CON 143         'locazione stato dei dispositivi di comando



'Blocco Memoria Moduli:
INCLUDE "i2cvar.bas"
INCLUDE "modedefs.bas"




'Blocco Main - Configurazione Iniziale:

main:                               'configurazione iniziale - Prime istruzioni
CLEAR                               'azzero le variabili
PAUSE 1500                          'pausa iniziale
ANSEL = 0                           'imposto le porte analogiche come i/o digitale (se uso il pic 16F88)
'CMCON = 7                          'imposto le porte analogiche (comparatori) come i/o digitale (se uso il pic
16F628)
'VRCON = 0                          'disattivo il modulo tensioni (se uso il pic 16F628)
OUTPUT led1                         'led1 è un output
OUTPUT led2                         'led2 è un output
OUTPUT ledcom2                      'ledcom2 è un output
INPUT intcom2                       'intcom2 è un input
OUTPUT com1tx                       'com1tx è un output
INPUT com1rx                        'com1rx è un input
OUTPUT com2tx                       'com2tx è un output
INPUT com2rx                        'com2rx è un input
OUTPUT com1rts                      'com1rts è un output
com1rts = ~rtsatt                   'segnale rts disattivato (default perchè rts è attivo basso)
INPUT com1cts                       'com1cts è un input
GOSUB i2cset                        'setto il modulo i2c
GOTO hardcheck                      'vado al check dell'hardware



'Blocco Configurazione Moduli:
INCLUDE "i2cset.bas"



'Blocco configurazione Programma Principale:

hardcheck:                          'test dell'hardware (memoria)
GOSUB i2copen                       'occupo il bus
GOSUB i2cmemorychk                  'test accesso alla memoria
GOSUB i2cclose                      'libero il bus
PAUSE rampaketpause                 'pausa contro le collisioni sul canale busy
IF i2cmemorychkflag = 1 THEN        'se il check memoria è fallito
GOTO hardcheck                      'continuo a verificare l'accesso alla memoria
ENDIF
GOSUB writeconfig                   'scrivo zeri e configurazioni iniziali nelle locazioni utilizzate
GOTO initprg                        'inizializzazione del programma


initprg:                            'sub di inizializzazione principale
    centralon = 1                       'indico che il pic delle comunicazioni è attivo
    GOSUB writeconfig                   'scrivo la ram
    PAUSE 10                            'attendo un attimo
          picpsxcheck:
    GOSUB readconfig                    'leggo la ram
    IF picpsxon = 0 THEN                'se il pic psx non è ancora attivo
       GOTO picpsxcheck                 'attendo la conferma che sia attivo
    ENDIF
    centralok = 1                       'indica che il pic gestore è pronto
    GOSUB writeconfig                   'scrivo la ram
    GOTO maincicle
GOTO initprg                        'altrimenti continua a verificare lo stato del picmaster
(inizializzazione)
```

```
'Blocco Programma Principale:


maincicle:                          'ciclo principale di programma
GOSUB readconfig                    'leggo la ram
GOSUB viewsegn                      'visualizzo le segnalazioni
GOSUB readcomms                     'leggo i dispositivi
GOSUB elabdata                      'elaboro i dati ricevuti
centralon = 1                       'indico che il pic gestore delle comunicazioni è attivo
centralok = 1                       'indica che il pic gestore è pronto
GOSUB writeconfig                   'scrivo la ram
GOSUB writecomms                    'scrivo i dispositivi
GOTO maincicle                      'rieseguo il ciclo di programma



readcomms:                          'leggo i dispositivi e ne organizzo priorità ed errori
devicesfail = 0                     'azzero il flag "nessuna periferica attiva"
comunicerror = 0                    'azzero il flag errore "lettura fallita da tutti i dispositivi connessi"
IF intcom2 = 1 THEN                 'se la com2 è attiva da impostazioni manuali
   com2state = 1                          'indico che la com2 è attiva
ELSE                                'in caso la che la com2 non sia stata attivata manualmente
   com2state = 0                          'indico la com2 non attiva
ENDIF
'inizialmente la priorità è quella standard (vaolore 0), prima leggo il pc poi il joypad che ha priorità più alta
(sostituisce)
IF com1cts = ctsatt THEN            'se la porta com1 (principale) è attiva
com1state = 1                       'indico che la porta com1 è attiva
GOSUB readcom1                      'leggo la com1 è metto i dati nelle variabili definitive
IF priorityflag = 1 THEN            'se il pc ha cambiato la priorità ed ha preso il controllo
   'in automatico viene disabilitata la vibrazione del joypad quando si arriva al writeconfig
   'in automatico se impostato dal pc, viene disattivato l'intero programma del joypad
   'i dati vengono lasciati nelle variabili definitive ed il joypad non viene letto
   RETURN           'esco dalla sub
ENDIF
ELSE                'in caso il com1 non risponda
   com1state = 0        'indico che la porta com1 non è attiva
ENDIF
psxdisable = 0      'in caso di errori azzero il flag di disattivazione del modulo del joypad
IF joynotconnected = 0 THEN             'se il joypad è collegato
    joystate = 1                            'indico che il joypad è attivo
    GOSUB readjoy                           'leggo il joypad e sostituisco i valori definitivi (priorità standard)
    RETURN                                  'esco dalla sub
ELSE                                    'se vi sono problemi hardware o il joypad non è collegato
    joystate = 0                            'il joypad non è attivo
    IF com1error = 1 THEN                   'se si è verificato un errore di lettura sulla com1
       comunicerror = 1                         'attivo il flag errore "lettura fallita da tutti i dispositivi
connessi"
       GOSUB clearinbuffer                      'azzero il buffer dei dati in ingresso
    ENDIF
ENDIF
IF joystate = 0 AND com1state = 0 THEN      'se entrambi i dispositivi principali sono fuori uso
   IF com2state = 1 THEN                    'se la com2 risulta attiva
      GOSUB readcom2                        'leggo la com2, ricordandomi di verificare il timeout
      RETURN                                'esco dalla sub
   ELSE                                     'in caso la che la com2 non sia stata attivata manualmente
   devicesfail = 1                          'attivo il flag "nessuna periferica attiva"
   GOSUB clearinbuffer                      'azzero il buffer dei dati in ingresso
   ENDIF
ENDIF
RETURN                                      'esco dalla sub


writecomms:                                 'invio la telemetria e gli stati ai terminali
'Il pacchetto inviato è composto da: preambolo1 + preambolo 2 + 31 byte di dati (vedi serout2 sottostante per
dettagli) + postambolo
IF com1state = 1 THEN                       'se la com1 è attiva invio i dati alla com1
SEROUT2 com1tx,com1mode,[compreamble1, compreamble2]            'invio i preamboli
SEROUT2 com1tx,com1mode,[corebyte,fuffy1cenp,fuffy2cenp,fuffy1dxp,fuffy2dxp,fuffy1sxp,fuffy2sxp,fuffy3p, _
        compassp,sensorsp,vbattp,exbyte0,exbyte1,exbyte2,exbyte3]           'invio i dati del dirigibile
SEROUT2 com1tx,com1mode,[devicesstate,segnalazbyte,picmasterstate,baseexpstate,baseintbyte,basecell1,basecell2,
baserssi]  'invio i dati della base
SEROUT2 com1tx,com1mode,[basestatep,joypadbuttonsp,joyanddirezp,motdxpwmp,motsxpwmp,motdownpwmp,prioritydev,
pcstate]      'invio la copia dei dati effettivi
SEROUT2 com1tx,com1mode,[compostamble]                  'invio il postambolo
ELSE                                        'se la com1 non è attiva
pcstate = 0                                 'azzero la variabile dello stato pc (non serve perchè com1 è scollegato)
prioritydev = 0                             'azzero la variabile con i flags di priorità (com 1 non connessa)
centralon = 1                               'indico che il pic delle comunicazioni è attivo
centralok = 1                               'indica che il pic gestore è pronto
ENDIF
IF com2state = 1 THEN                       'se la com2 è attiva invio i dati anche alla com 2
SEROUT2 com2tx,com2mode,[compreamble1, compreamble2]            'invio i preamboli
SEROUT2 com2tx,com2mode,[corebyte,fuffy1cenp,fuffy2cenp,fuffy1dxp,fuffy2dxp,fuffy1sxp,fuffy2sxp,fuffy3p, _
        compassp,sensorsp,vbattp,exbyte0,exbyte1,exbyte2,exbyte3]           'invio i dati del dirigibile
SEROUT2 com2tx,com2mode,[devicesstate,segnalazbyte,picmasterstate,baseexpstate,baseintbyte,basecell1,basecell2,
baserssi]  'invio i dati della base
SEROUT2 com2tx,com2mode,[basestatep,joypadbuttonsp,joyanddirezp,motdxpwmp,motsxpwmp,motdownpwmp,prioritydev,
pcstate]      'invio la copia dei dati effettivi
SEROUT2 com2tx,com1mode,[compostamble]                  'invio il postambolo
ENDIF
```

```
RETURN                                  'esco dalla sub


readcom1:                               'leggo la porta com1
com1error = 0                           'disattivo il flag "errore lettura com1"
com1rts = rtsatt                        'alzo l'rts, chiedo al pc che mi vengano spediti i dati
'Riceve Il pacchetto dati dal pc. L'istruzione attende i dati per un certo timeout. Se non riceve niente chiama
la sub
'assegnata al timeout (errore ricezione)
SERIN2 com1rx,com1mode,com1timeout,com1timeoutstate,[preamble1,preamble2]   'ricevo i preamboli
com1rts = ~ rtsatt                                                          'abbasso il segnale rts
IF preamble1 = compreamble1 AND preamble2 = compreamble2 THEN               'se ho ricevuto i preamboli
correttamente
SERIN2 com1rx,com1mode,com1timeout,com1timeoutstate,[basestatep,joypadbuttonsp,joyanddirezp,motdxpwmp, _
       motsxpwmp,motdownpwmp,prioritydev,pcstate]                           'ricevo i dati
SERIN2 com1rx,com1mode,com1timeout,com1timeoutstate,[postamble]             'ricevo il postambolo
    IF postamble <> compostamble THEN                                       'se il postambolo è diverso dal
valore impostato
       com1error = 1                                                        'attivo il flag di errore lettura
com1
    ENDIF
RETURN                                  'esco dalla sub
ELSE                                    'se i preamboli non corrispondono c'è un errore di lettura
com1timeoutstate:                       'se il timeout scade allora c'è un errore di lettura
com1error = 1                           'attivo il flag di errore lettura com1
com1rts = ~ rtsatt                      'abbasso l'rts
ENDIF
RETURN                                  'esco dalla sub


readcom2:                               'leggo la porta com2
com2error = 0                           'azzero il flag "errore lettura com2"
SEROUT2 com2tx,com2mode,[comdatarequest]          'invio la richiesta di invio dati
SERIN2 com2rx,com2mode,com2timeout,com2timeoutstate,[preamble1,preamble2]   'ricevo i preamboli
IF preamble1 = compreamble1 AND preamble2 = compreamble2 THEN               'se ho ricevuto i preamboli
correttamente
SERIN2 com2rx,com2mode,com2timeout,com2timeoutstate,[basestatep,joypadbuttonsp,joyanddirezp,motdxpwmp, _
       motsxpwmp,motdownpwmp]                                               'ricevo gli altri dati
SERIN2 com2rx,com2mode,com2timeout,com2timeoutstate,[postamble]             'ricevo il postambolo
    IF postamble <> compostamble THEN                                       'se il postambolo è diverso dal
valore impostato
        com2error = 1                           'attivo il flag "errore lettura com 2"
        comunicerror = 1                        'attivo il flag errore "lettura fallita da tutti i dispositivi
connessi"
       GOSUB clearinbuffer                      'azzero il buffer dei dati in ingresso
    ENDIF
RETURN                                  'esco dalla sub
ELSE                                    'se i preamboli non sono corretti
com2error = 1                           'attivo il flag "errore lettura com 2"
comunicerror = 1                        'attivo il flag errore "lettura fallita da tutti i dispositivi connessi"
GOSUB clearinbuffer                     'azzero il buffer dei dati in ingresso
RETURN                                  'esco dalla sub
ENDIF
com2timeoutstate:                       'in caso il timeout scada, intendo la com2 come "non attiva"
com2state = 0                           'indico la com2 come non attiva
devicesfail = 1                         'attivo il flag "nessuna periferica attiva"
GOSUB clearinbuffer                     'azzero il buffer dei dati in ingresso
RETURN                                  'esco dalla sub


readjoy:                                'leggo il joypad
GOSUB readconfig                        'leggo la ram e quindi i dati del joypad (vengono sostituiti alla
variabili definitive)
RETURN                                  'esco dalla sub


elabdata:                               'elaborazione dati rivecuti
replystate = rxpcstate                  'confermo lo stato del pc (reply) perchè il programma sul pc si renda
conto che la base risponde
RETURN                                  'esco dalla sub


'Blocco Subroutines Secondarie Programma Principale:


viewsegn:                       'sub di visualizzazione delle segnalazioni (leds)
led1 = led1cstate               'visualizzo la segnalazione corrispondente al led 1
led2 = led2cstate               'visualizzo la segnalazione corrispondente al led 2
ledcom2 = intcom2               'visualizzo l'impostazione dell'interruttore sul led corrispondente
RETURN


clearinbuffer:                  'azzera tutte le variabili lette dalle periferiche (in caso di errori/stati
fatali)
basestatep = 0
joypadbuttonsp = 0
joyanddirezp = 0
motdxpwmp = 0
motsxpwmp = 0
motdownpwmp = 0
prioritydev = 0
pcstate = 0
```

```
RETURN


'Blocco Subroutines Moduli:
INCLUDE "i2cmod.bas"



'Blocco Lettura/Scrittura Memoria RAM:

'---> Lettura Memoria
readconfig:                     'Leggo i dati dalla Ram
GOSUB i2copen                   'Occupo il bus
    readloc0:                   'posizione blocco lettura - 0
i2caddr = statebytejloc         'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc0            'se si verifica un errore di lettura, rileggo la locazione
corrente
statebytej = i2cin              'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc1:                   'posizione blocco lettura - 1
i2caddr = joypadbuttonsjloc     'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc1            'se si verifica un errore di lettura, rileggo la locazione
corrente
joypadbuttonsp = i2cin          'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc2:                   'posizione blocco lettura - 2
i2caddr = joyanddirezjloc       'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc2            'se si verifica un errore di lettura, rileggo la locazione
corrente
joyanddirezp = i2cin            'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc3:                   'posizione blocco lettura - 3
i2caddr = pwmmotdxjloc          'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc3            'se si verifica un errore di lettura, rileggo la locazione
corrente
motdxpwmp = i2cin               'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc4:                   'posizione blocco lettura - 4
i2caddr = pwmmotsxjloc          'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc4            'se si verifica un errore di lettura, rileggo la locazione
corrente
motsxpwmp = i2cin               'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc5:                   'posizione blocco lettura - 5
i2caddr = pwmmotdownjloc        'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc5            'se si verifica un errore di lettura, rileggo la locazione
corrente
motdownpwmp = i2cin             'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc6:                   'posizione blocco lettura - 6
i2caddr = corebyteloc           'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc6            'se si verifica un errore di lettura, rileggo la locazione
corrente
corebyte = i2cin                'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc7:                   'posizione blocco lettura - 7
i2caddr = fuffy1cenploc         'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc7            'se si verifica un errore di lettura, rileggo la locazione
corrente
fuffy1cenp = i2cin              'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc8:                   'posizione blocco lettura - 8
i2caddr = fuffy2cenploc         'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc8            'se si verifica un errore di lettura, rileggo la locazione
corrente
fuffy2cenp = i2cin              'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc9:                   'posizione blocco lettura - 9
i2caddr = fuffy1dxploc          'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc9            'se si verifica un errore di lettura, rileggo la locazione
corrente
fuffy1dxp = i2cin               'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc10:                  'posizione blocco lettura - 10
i2caddr = fuffy2dxploc          'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc10           'se si verifica un errore di lettura, rileggo la locazione
corrente
fuffy2dxp = i2cin               'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc11:                  'posizione blocco lettura - 11
i2caddr = fuffy1sxploc          'seleziono la locazione da leggere
```

```
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc11           'se si verifica un errore di lettura, rileggo la locazione
corrente
fuffy1sxp = i2cin               'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc12:                  'posizione blocco lettura - 12
i2caddr = fuffy2sxploc          'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc12           'se si verifica un errore di lettura, rileggo la locazione
corrente
fuffy2sxp = i2cin               'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc13:                  'posizione blocco lettura - 13
i2caddr = fuffy3ploc            'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc13           'se si verifica un errore di lettura, rileggo la locazione
corrente
fuffy3p = i2cin                 'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc14:                  'posizione blocco lettura - 14
i2caddr = compassploc           'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc14           'se si verifica un errore di lettura, rileggo la locazione
corrente
compassp = i2cin                'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc15:                  'posizione blocco lettura - 15
i2caddr = sensorsploc           'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc15           'se si verifica un errore di lettura, rileggo la locazione
corrente
sensorsp = i2cin                'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc16:                  'posizione blocco lettura - 16
i2caddr = vbattploc             'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc16           'se si verifica un errore di lettura, rileggo la locazione
corrente
vbattp = i2cin                  'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc17:                  'posizione blocco lettura - 17
i2caddr = exbyte0loc            'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc17           'se si verifica un errore di lettura, rileggo la locazione
corrente
exbyte0 = i2cin                 'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc18:                  'posizione blocco lettura - 18
i2caddr = exbyte1loc            'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc18           'se si verifica un errore di lettura, rileggo la locazione
corrente
exbyte1 = i2cin                 'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc19:                  'posizione blocco lettura - 19
i2caddr = exbyte2loc            'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc19           'se si verifica un errore di lettura, rileggo la locazione
corrente
exbyte2 = i2cin                 'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc20:                  'posizione blocco lettura - 20
i2caddr = exbyte3loc            'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc20           'se si verifica un errore di lettura, rileggo la locazione
corrente
exbyte3 = i2cin                 'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc21:                  'posizione blocco lettura - 21
i2caddr = segnalazbyteloc       'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc21           'se si verifica un errore di lettura, rileggo la locazione
corrente
segnalazbyte = i2cin            'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc22:                  'posizione blocco lettura - 22
i2caddr = picmasterstateloc     'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc22           'se si verifica un errore di lettura, rileggo la locazione
corrente
picmasterstate = i2cin          'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc23:                  'posizione blocco lettura - 23
i2caddr = baseexpstateloc       'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc23           'se si verifica un errore di lettura, rileggo la locazione
corrente
baseexpstate = i2cin            'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc24:                  'posizione blocco lettura - 24
i2caddr = baseintbyteloc        'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc24           'se si verifica un errore di lettura, rileggo la locazione
corrente
```

```
baseintbyte = i2cin             'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc25:                  'posizione blocco lettura - 25
i2caddr = basecell1loc          'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc25           'se si verifica un errore di lettura, rileggo la locazione
corrente
basecell1 = i2cin               'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc26:                  'posizione blocco lettura - 26
i2caddr = basecell2loc          'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc26           'se si verifica un errore di lettura, rileggo la locazione
corrente
basecell2 = i2cin               'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    readloc27:                  'posizione blocco lettura - 27
i2caddr = baserssiloc           'seleziono la locazione da leggere
GOSUB i2crx                     'leggo la locazione selezionata
IF i2cackflag = 1 THEN GOTO readloc27           'se si verifica un errore di lettura, rileggo la locazione
corrente
baserssi = i2cin                'carico il registro
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
GOSUB i2cclose                  'Libero il bus
PAUSE rampaketpause             'pausa contro le collisioni sul canale busy
RETURN


'---> Scrittura Memoria
writeconfig:                    'Scrivo i dati in Ram
GOSUB i2copen                   'occupo il bus
    writeloc0:                  'posizione blocco scrittura - 0
i2caddr = baseintbyteloc        'seleziono la locazione da scrivere
i2cout = baseintbyte            'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc0           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc1:                  'posizione blocco scrittura - 1
i2caddr = centralstatejloc      'seleziono la locazione da scrivere
i2cout = centralstatej          'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc1           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc2:                  'posizione blocco scrittura - 2
i2caddr = basestateploc         'seleziono la locazione da scrivere
i2cout = basestatep             'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc2           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc3:                  'posizione blocco scrittura - 3
i2caddr = joypadbuttonsploc     'seleziono la locazione da scrivere
i2cout = joypadbuttonsp         'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc3           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc4:                  'posizione blocco scrittura - 4
i2caddr = joyanddirezploc       'seleziono la locazione da scrivere
i2cout = joyanddirezp           'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc4           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc5:                  'posizione blocco scrittura - 5
i2caddr = motdxpwmploc          'seleziono la locazione da scrivere
i2cout = motdxpwmp              'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc5           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc6:                  'posizione blocco scrittura - 6
i2caddr = motsxpwmploc          'seleziono la locazione da scrivere
i2cout = motsxpwmp              'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc6           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc7:                  'posizione blocco scrittura - 7
i2caddr = motdownpwmploc        'seleziono la locazione da scrivere
i2cout = motdownpwmp            'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc7           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc8:                  'posizione blocco scrittura - 8
i2caddr = prioritydevloc        'seleziono la locazione da scrivere
i2cout = prioritydev            'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc8           'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
```

```
    writeloc9:                  'posizione blocco scrittura - 9
i2caddr = pcstateloc            'seleziono la locazione da scrivere
i2cout = pcstate                'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc9          'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
    writeloc10:                 'posizione blocco scrittura - 10
i2caddr = devicesstateloc       'seleziono la locazione da scrivere
i2cout = devicesstate           'carico il valore in uscita
GOSUB i2ctx                     'scrivo il dato nella locazione in ram
IF i2cackflag = 1 THEN GOTO writeloc10          'se si verifica un errore di scrittura, riscrivo la locazione
corrente
PAUSEUS anticollisiondelay      'pausa per prevenire collisioni accidentali tra i telegrammi
GOSUB i2cclose                  'libero il bus
PAUSE rampaketpause             'pausa contro le collisioni sul canale busy
RETURN



'Blocco Gestione Errori:
'---> Sub di gestione errori non utilizzate
i2cbusy:
i2cackerror:
i2cmemoryaccesserror:
GOTO hardcheck              'in caso vengano chiamate per sbaglio rimando al check dell'hardware
```

PROGRAMMA MODULO CICLOPÈ

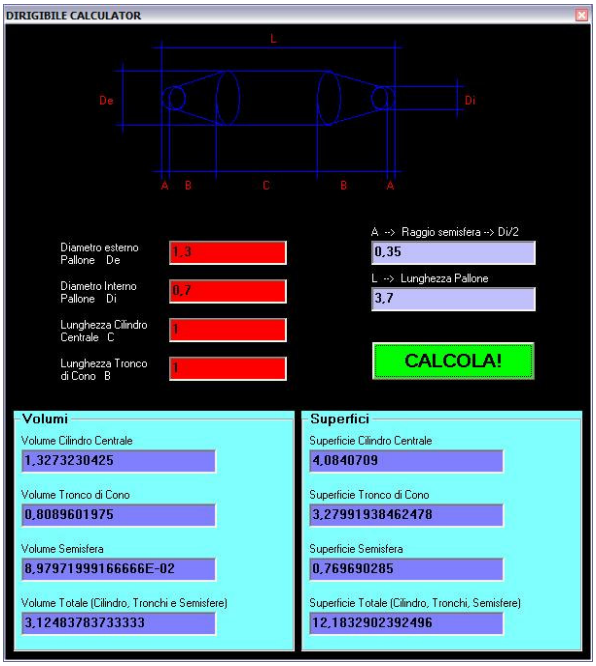
```
'Blocco direttive:


'Blocco memoria Programma Principale:

'---> Definizione Pins I/O
SYMBOL teleconoff = porta.2                 'pin accensione/spegnimento telecamera
SYMBOL laseronoff = portb.0                 'pin accensione/spegnimento puntatore laser
SYMBOL serial = portb.2                     'pin ricezione dati seriale
SYMBOL cicloperts = portb.5                 'pin Request to Send (corrispondente al CTS sul core) - indica lo
stato del modulo - attivo alto
SYMBOL finecdx = porta.3                    'finecorsa destro da dietro
SYMBOL finecsx = porta.4                    'finecorsa sinistro da dietro
SYMBOL finecup = portb.1                    'finecorsa alto da dietro
SYMBOL finecdown = portb.4                  'finecorsa basso da dietro
SYMBOL direzdxsx1 = portb.7                 'comando motore direzione/off 1 - motore posizione destra/sinistra
SYMBOL direzdxsx2 = porta.0                 'comando motore direzione/off 2 - motore posizione destra/sinistra
SYMBOL direzupdown1 = portb.6               'comando motore direzione/off 1 - motore posizione su/giù
SYMBOL direzupdown2 = porta.1               'comando motore direzione/off 2 - motore posizione su/giù
SYMBOL motpulse = portb.3                   'comando movimento motori (a impulsi)


'---> Costanti
sermode CON 396                             'impostazione velocità di comunicazione serin2
sertimeout CON 5000                         'timeout ricezione dati perchè il core sia considerato non attivo
(in ms)



'---> Variabili
command VAR BYTE                            'comandi ricevuti via seriale
        '---> Alias
        camerayup VAR command.0                 'motore asse y telecamera direzione alto
        cameraxright VAR command.1              'motore asse x telecamera direzione destra
        cameraydown VAR command.2               'motore asse y telecamera direzione basso
        cameraxleft VAR command.3               'motore asse x telecamera direzione sinistra
        cameraacc VAR command.4                 'accensione telecamera
        laseracc VAR command.5                  'accensione puntatore laser
intstate VAR BYTE                           'flags di stato interni

pulsetime VAR WORD                          'durata impulso motori (in millisecondi)
pulsepause VAR WORD                         'pausa popo l'impulso (tempo tra un impulso e l'altro (in
millisecondi)



'Blocco Main - Configurazione Iniziale:

main:                           'istruzioni prime
CLEAR                           'azzera le variabili
PAUSE 500                       'pausa di accensione
'ANSEL = 0                      'imposto le porte analogiche come i/o digitale (se uso il pic 16F88)
'CMCON = 7                      'imposto le porte analogiche (comparatori) come i/o digitale (se uso il pic 16F628)
'VRCON = 0                      'disattivo il modulo tensioni (se uso il pic 16F628)
PAUSE 500                       'pausa di accensione
OUTPUT teleconoff               'teleconoff è un output
LOW teleconoff                  'telecamera spenta
OUTPUT laseronoff               'laseronoff è un output
LOW laseronoff                  'laser spento
INPUT serial                    'serial è un input
OUTPUT cicloperts               'cicloperts è un output
LOW cicloperts                  'cicloperts disattivato
INPUT finecdx                   'finecdx è un imput
INPUT finecsx                   'finecsx è un input
INPUT finecup                   'finecup è un input
INPUT finecdown                 'finecdown è un imput
OUTPUT direzdxsx1               'direzdxsx1 è un output
OUTPUT direzdxsx2               'direzdxsx2 è un output
OUTPUT direzupdown1             'direzupdown1 è un output
OUTPUT direzupdown2             'direzupdown2 è un output
OUTPUT motpulse                 'motpulse è un output
GOTO init                       'vado all'inizializzazione



'Blocco configurazione Programma Principale:

init:                           'inizializzazione programma
pulsetime = 50                  'imposto la durata dell'impulso inviato ai motori (in millisecondi)
pulsepause = 200                'pausa tra un impulso e l'altro (in ms)
GOSUB initposiz                 'porto la telecamera nella posizione iniziale
GOTO prgcic                     'vado al ciclo di programma



'Blocco Programma Principale:


prgcic:                         'ciclo principale di programma
GOSUB indata                    'ricevo il byte di comando
```

```
GOSUB setdirez                  'imposto le direzioni dei motori x ed y
GOSUB setonoff                  'attivo/disattivo telecamera e/o laser
GOSUB finecheck                 'controllo lo stato dei finecorsa e blocco i motori
GOSUB pulsemot                  'invio un impulso di posizionamento ai motori
GOTO prgcic                     'rieseguo all'infinito il ciclo di programma


indata:                         'riceve i dati dalla seriale
HIGH cicloperts                 'attivo il segnale rts (ricezione comandi disponibile)
SERIN2 serial,sermode,sertimeout,coredown,[command]          'ricevo la variabile di comando
LOW cicloperts                  'disattivo il segnale rts (ricezione comandi non disponibile)
RETURN                          'esco dalla sub
coredown:                       'se scade il timeout della serin2 significa che il core non è attivo
LOW cicloperts                  'disattivo il segnale rts (ricezione comandi non disponibile)
GOSUB initposiz                 'porto la telecamera nella posizione iniziale
command = 0                     'azzero la variabile di comando
RETURN                          'esco dalla sub


setdirez:                   'imposta le direzioni dei motori
IF cameraxright = 1 THEN            'se ho rivevuto il comando "telecamera verso destra (da dietro)"
direzdxsx1 = 1                      'imposto i pin per la direzione destra (motore x)
direzdxsx2 = 0
ENDIF
IF cameraxleft = 1 THEN             'se ho rivevuto il comando "telecamera verso sinistra (da dietro)"
direzdxsx1 = 0                      'imposto i pin per la direzione sinistra (motore x)
direzdxsx2 = 1
ENDIF
IF cameraxright = 0 AND cameraxleft = 0 THEN          'se ho rivevuto il comando "telecamera destra/sinistra
ferma"
direzdxsx1 = 0                                        'imposto i pin per motore fermo (motore x)
direzdxsx2 = 0
ENDIF
IF camerayup = 1 THEN               'se ho rivevuto il comando "telecamera verso l'alto (da dietro)"
direzupdown1 = 1                    'imposto i pin per la direzione su (motore y)
direzupdown2 = 0
ENDIF
IF cameraydown = 1 THEN             'se ho rivevuto il comando "telecamera verso il basso (da dietro)"
direzupdown1 = 0                    'imposto i pin per la direzione giù (motore y)
direzupdown2 = 1
ENDIF
IF camerayup = 0 AND cameraydown = 0 THEN             'se ho rivevuto il comando "telecamera su/giù ferma"
direzupdown1 = 0                                      'imposto i pin per motore fermo (motore y)
direzupdown2 = 0
ENDIF
RETURN                              'esco dalla sub


setonoff:                   'accende e spegne laser e telecamera
teleconoff = cameraacc      'accendo o spengo la telecamera
laseronoff = laseracc       'accendo o spengo il laser
RETURN                      'esco dalla sub


finecheck:                  'controlla i finecorsa e blocca i motori
IF finecdx = 0 AND cameraxright = 1 THEN              'se il finecorsa destro è chiuso ed intendo andare a destra
direzdxsx1 = 0                                        'imposto i pin per motore fermo (motore x)
direzdxsx2 = 0
ENDIF
IF finecsx = 0 AND cameraxleft = 1 THEN               'se il finecorsa sinistro è chiuso ed intendo andare a
sinistra
direzdxsx1 = 0                                        'imposto i pin per motore fermo (motore x)
direzdxsx2 = 0
ENDIF
IF finecup = 0 AND camerayup = 1 THEN                 'se il finecorsa alto è chiuso ed intendo andare in su
direzupdown1 = 0                                      'imposto i pin per motore fermo (motore y)
direzupdown2 = 0
ENDIF
IF finecdown = 0 AND cameraydown = 1 THEN             'se il finecorsa basso è chiuso ed intendo andare in giù
direzupdown1 = 0                                      'imposto i pin per motore fermo (motore y)
direzupdown2 = 0
ENDIF
RETURN                                                'esco dalla sub


pulsemot:                   'invia un impulso ai motori
IF direzdxsx1 = 0 AND direzdxsx2 = 0 AND direzupdown1 = 0 AND direzupdown2 = 0 THEN          'se tutti i motori
sono fermi
RETURN                      'esco dalla sub
ENDIF
HIGH motpulse               'invio un impulso
PAUSE pulsetime             'di "pulsetime" millisecondi
LOW motpulse
PAUSE pulsepause            'tra un impulso ed il successivo c'è un delay pari a "pulsepause" millisecondi
RETURN                      'esco dalla sub


'Blocco Subroutines Secondarie Programma Principale:


initposiz:                  'mette la telecamera nella posizione iniziale
'niente per ora
RETURN                      'esco dalla sub
```

PROGRAMMI PC

Nel corso della progettazione abbiamo fatto uso di Visual Basic per creare diversi applicativi necessari alla semplificazione di alcuni lavori.
Per esempio è stato creato un timer-countdown per “dare la temporizzazione” al personale lavorativo durante la fase di saldatura del pallone: il timer emette due suoni distinti, uno indica l’inizio della saldatura con i ferri da stiro e l’altro indica la fine. Tra ogni saldatura vi è un breve tempo morto per dare il tempo di posizionarsi sul successivo punto da saldare.

Grazie a tale programma siamo riusciti a costruire in maniere veloce il pallone, garantendo lo stesso tempo per ogni saldatura.
Un altro programmino realizzato è il già discusso “Dirigibile Calcolator”, per la semplificazione dei calcoli di dimensionamento del pallone.



La Base Terrestre comunica con il computer attraverso una porta seriale. Il computer può pilotare il dirigibile e visualizzare i valori dei sensori del velivolo. Era necessaria un'interfaccia grafica per il comando e la visualizzazione dei dati. Visual Basic è stato a capo dell'operazione.

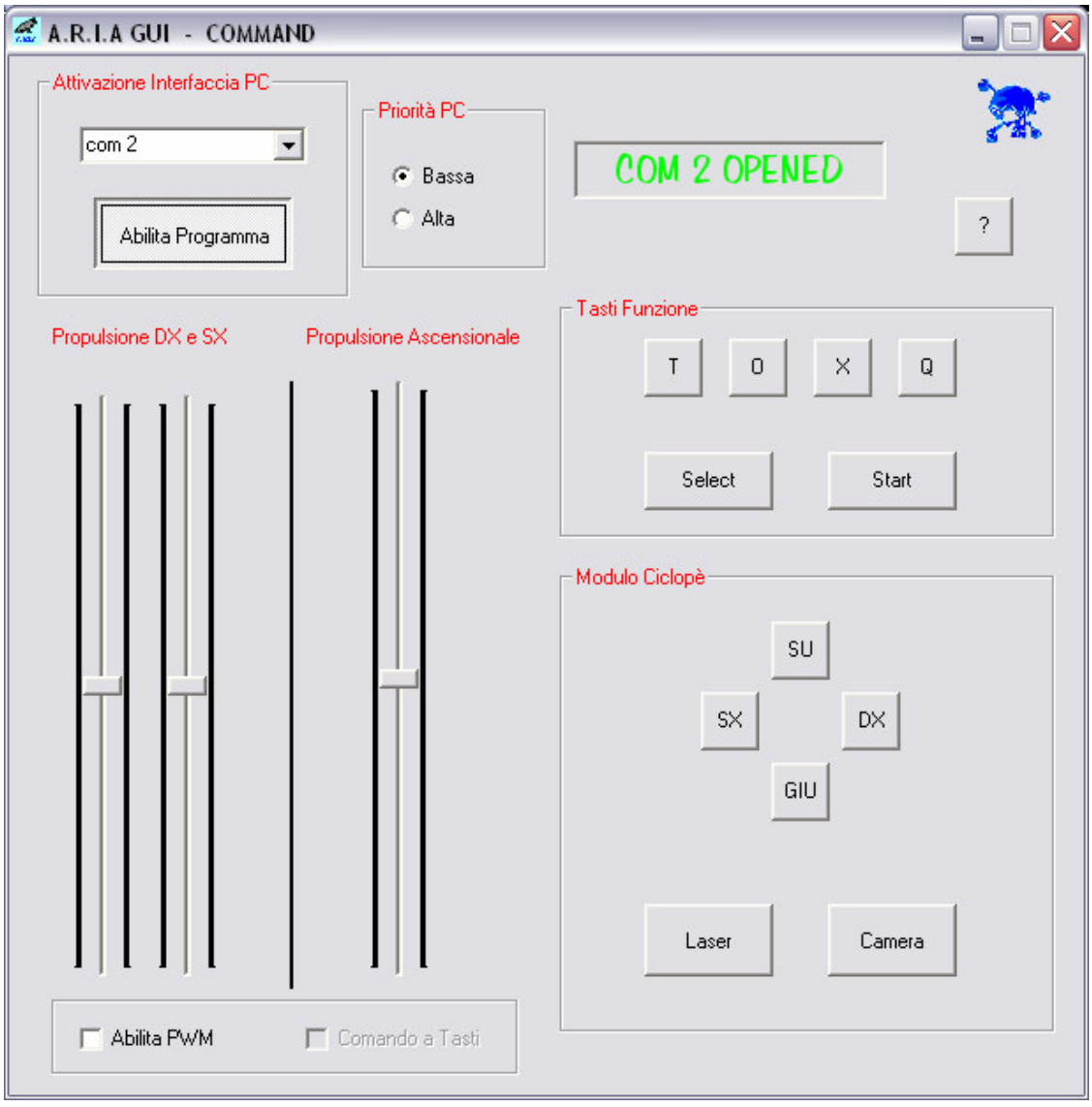
ARIA GUI – INTERFACCIA GRAFICA:

Visual Basic mette a disposizione un elevato numero di controlli grafici controllabili in run-time. Inoltre integra uno strumento per la comunicazione seriale molto semplice da usare: L'MSComm.

Il programma è stato suddiviso su tre finestre: la prima è quella di pilotaggio del dirigibile, la seconda visualizza i dati dei sensori (telemetria) e la terza è un echo dei comandi inviati alla base (sia dal pc che dal joypad).
Per ogni finestra è stata creata un'interfaccia grafica dedicata.
Nel form di comando si possono agire su delle slide per impartire comandi ai motori e su dei pulsanti per il resto delle funzioni (laser, telecamera…).
I comandi possono essere impartiti anche premendo i tasti della tastiera assegnati ad ogni funzione (frecce direzionali per i motori…).
All'interno di questo form ci sono anche le configurazioni per la comunicazione con il PC e l'abilitazione generale del programma.
Nel form della telemetria si possono osservare i flags di stato: si può sapere in ogni momento se il link radio è attivo, se la base è accesa e quale dispositivo di comunicazione ha la priorità.
Nella parte centrale dell'interfaccia vi sono dei riquadri contenenti le distanze misurate dai sensori sonar e dal sensore ultrasonico dell'altitudine. Questi valori sono indici della presenza di ostacoli in prossimità del velivolo.
Altri riquadri contengono segnalazioni, altri sensori, livelli delle batterie…
È da notare il riquadro contenente la rosa dei venti. Questa è l'attrazione migliore del programma: la lancetta della rosa si sposta, indicando la direzione cardinale verso la quale sta facendo rotta il dirigibile.

Nell'ultimo form invece è stato riprodotto graficamente il JoyPad della Playstation. I tasti grafici si illuminano quando un comando è impartito al dirigibile, confermando all'utente la scelta effettuata. Se i comandi arrivano dal PC, il joypad virtuale visualizza i comandi inviati come se provenissero da un joypad reale. In poche parole viene visualizzato lo stesso effetto che si avrebbe impartendo lo stesso comando attraverso il GamePad.
Per la comunicazione con il pc viene usato l'oggetto MSComm, che semplifica notevolmente la comunicazione seriale. È stato realizzato un modulo che si occupa di caricare in memoria i byte inviati e separare i vari pacchetti. Ogni pacchetto inizia con due byte "preambolo", il secondo complementare del primo, e si conclude con un postambolo, posto uguale al primo preambolo. Il modulo preleva i byte contenuti tra i preamboli ed il postambolo e li divide inserendoli in una matrice (un telegramma per indice).
Non si deve fare altro che chiamare la sub del modulo ogni volta che si riceve un singolo byte dalla seriale (il modulo è per uso generale).
All'interno del programma, nei vari form, basta accedere al telegramma richiesto, contenuto nella matrice, contenente i dati da visualizzare.
Per la separazione dei singoli bit, i flags di stato, è stato creato un modulo che opera la conversione da decimale a binario e viceversa.
Di seguito verrà proposto il programma realizzato, comprensivo di interfaccia grafica e codice sorgente (sorgenti divisi per ottimizzare la lettura). Nota che per ricostruire il programma non basta ricreare l'interfaccia e ricopiare il codice! In caso richiedere il file zip contenente il progetto completo.

FORM COMMAND



```
Option Explicit                 'le variabili devono essere tutte dichiarate
Private Declare Function InitCommonControls Lib "Comctl32.dll" () As Long
'attivo la possibilità di usare la grafica XP (manifest)
'Variabili valore tasti di comando
Public tastot As Integer, tastoq As Integer, tastox As Integer, tastoo As Integer,
tastost As Integer, tastosl As Integer
Public tastoup As Integer, tastodown As Integer, tastoleft As Integer, tastoright As
Integer
'Variabili valori direzione motori
Public direzdx As Integer, direzsx As Integer, direzdown As Integer
Public pccmd As Integer                         'Variabile acquisizione priorità (su
bassa priorità)
Public watchdog As Long                         'contatore "watchdog" per rilevazione
base terrestre
Const compre1 = 170, compre2 = 85, compost = 170        'Preamboli e Postamboli per
trasmissione/ricezione seriale
Const dogtimeout = 500                          'Timeout per il contatore "watchdog"



Private Sub camera_Click()              'Click sull'interruttore camera
If camera.Value = 1 Then                'se l'interruttore è chiuso
    pccmd = 1                               'acquisisco la priorità
Else                                    'altrimenti
    pccmd = 0                               'rilascio la priorità
End If
End Sub

Private Sub cameradx_MouseDown(Button As Integer, Shift As Integer, x As Single, Y As
Single)       'premo il tasto "freccia dx"
tastoright = 1              'attivo il tasto
pccmd = 1                   'acquisisco la priorità
End Sub

Private Sub cameradx_MouseUp(Button As Integer, Shift As Integer, x As Single, Y As
Single)         'rilascio il tasto "freccia dx"
tastoright = 0              'disattivo il tasto
pccmd = 0                   'rilascio la priorità
End Sub

Private Sub cameragiu_MouseDown(Button As Integer, Shift As Integer, x As Single, Y As
Single)      'premo il tasto "freccia giù"
tastodown = 1               'attivo il tasto
pccmd = 1                   'acquisisco la priorità
End Sub

Private Sub cameragiu_MouseUp(Button As Integer, Shift As Integer, x As Single, Y As
Single)        'rilascio il tasto "freccia giù"
tastodown = 0               'disattivo il tasto
pccmd = 0                   'rilascio la priorità
End Sub

Private Sub camerasu_MouseDown(Button As Integer, Shift As Integer, x As Single, Y As
Single)       'premo il tasto "freccia su"
tastoup = 1                 'attivo il tasto
pccmd = 1                   'acquisisco la priorità
End Sub

Private Sub camerasu_MouseUp(Button As Integer, Shift As Integer, x As Single, Y As
Single)         'rilascio il tasto "freccia su"
tastoup = 0                 'disattivo il tasto
pccmd = 0                   'rilascio la priorità
End Sub

Private Sub camerasx_MouseDown(Button As Integer, Shift As Integer, x As Single, Y As
Single)       'premo il tasto "freccia sx"
tastoleft = 1               'attivo il tasto
pccmd = 1                   'acquisisco la priorità
End Sub

Private Sub camerasx_MouseUp(Button As Integer, Shift As Integer, x As Single, Y As
```

```
Single)         'rilascio il tasto "freccia sx"
tastoleft = 0               'disattivo il tasto
pccmd = 0                   'rilascio la priorità
End Sub


Private Sub Com1_OnComm()                           'eventi generati durante la comunicazione
seriale
On Error Resume Next                                'in caso di errori proseguo
If Com1.CTSHolding = True Then                      'se ho ricevuto una richiesta di invio
comandi
    watchdog = 0                                        'azzero il watchdog
    Call trasmpack                                      'trasmetto i comandi
End If
If Com1.CommEvent = comEvReceive Then               'se ho ricevuto qualcosa
    Call CommSet.InPacket(Com1)                         'aggiungo il byte ricevuto al buffer
di elaborazione e separo i pacchetti (ed i telegrammi)
End If
End Sub

Private Sub Command1_Click()                        'se premo il tasto della guida "?"
MsgBox "La guida non è ancora stata implementata", vbInformation, "Info"
'messaggio: la guida non è stata ancora implementata
End Sub

Private Sub focus_Change()                          'se per sbaglio scrivo qualcosa ne
textbox per lo stato attivo
focus.Text = ""                                     'cancello il textbox
End Sub

Private Sub focus_GotFocus()                        'se il controllo textbox nascosto riceve
lo stato attivo
Check2.Value = 1                                    'abilito i comandi da tastiera
End Sub

Private Sub focus_LostFocus()                       'se il controllo textbox nascosto perde
lo stato attivo (pressione tab)
Check2.Value = 0                                    'disattivo i comandi da tastiera
End Sub

Private Sub Form_Initialize()                       'attivo la grafica xp se è presente il
file manifest
    Dim x As Long
    x = InitCommonControls
End Sub

Private Sub Form_KeyUp(KeyCode As Integer, Shift As Integer)            'rilascio un
pulsante
On Error Resume Next                                'in caso di errori proseguo
If Check2.Value = 0 Then Exit Sub                   'se non è abilitato il comando da
tastiera esco dalla sub
If Label3.Visible = False Then                      'se il pc non ha la priorità (label
invisibile)
    Beep                                                'emetto un beep
End If
pccmd = 0                                           'azzero la variabile di acquisizione
priorità
If KeyCode = vbKeyZ Then                            'se è stato rilasciato il tasto Z
    cameragiu.Value = False                             'disattivo il pulsante "freccia giu"
    tastodown = 0
End If
If KeyCode = vbKeyA Then                            'se è stato rilasciato il tasto A
    camerasx.Value = False                              'disattivo il pulsante "freccia sx"
    tastoleft = 0
End If
If KeyCode = vbKeyS Then                            'se è stato rilasciato il tasto S
    cameradx.Value = False                              'disattivo il pulsante "freccia dx"
    tastoright = 0
End If
If KeyCode = vbKeyW Then                            'se è stato rilasciato il tasto W
    camerasu.Value = False                              'disattivo il pulsante "freccia su"
```

```
    tastoup = 0
End If
If KeyCode = vbKeyQ Then                        'se è stato rilasciato il tasto Q
    TastQ.Value = False                             'disattivo il pulsante "Q"
    tastoq = 0
End If
If KeyCode = vbKeyO Then                        'se è stato rilasciato il tasto O
    TastO.Value = False                             'disattivo il pulsante "O"
    tastoo = 0
End If
If KeyCode = vbKeyT Then                        'se è stato rilasciato il tasto T
    tastT.Value = False                             'disattivo il pulsante "T"
    tastot = 0
End If
If KeyCode = vbKeyX Then                        'se è stato rilasciato il tasto X
    tastX.Value = False                             'disattivo il pulsante "X"
    tastox = 0
End If
If KeyCode = vbKeyD Then                        'se è stato rilasciato il tasto D
    Me.select.Value = False                         'disattivo il pulsante "select"
    tastosl = 0
End If
If KeyCode = vbKeyF Then                        'se è stato rilasciato il tasto F
    Start.Value = False                             'disattivo il pulsante "start"
    tastost = 0
End If
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)              'il
programma sta per essere chiuso
On Error Resume Next                            'in caso di errori proseguo
If Com1.PortOpen = True Then                    'se la porta com è aperta
    Com1.PortOpen = False                           'la chiudo
End If
ricezione.totunload = True                      'abilito la chiusura del form ricezione
joy.totunload = True                            'abilito la chiusura del form del joypad
Unload comandi                                  'scarico i forms dalla memoria
Unload joy
Unload Me
End                                             'tolgo brutalmente il programma dalla ram
(in caso di problemi)
End Sub

Private Sub laser_Click()                       'se clicco sull'interruttore del laser
If laser.Value = 1 Then                         'se l'interruttore è chiuso
    pccmd = 1                                       'acquisisco la priorità
Else                                            'altrimenti
    pccmd = 0                                       'rilascio la priorità
End If
End Sub

Private Sub abprg_Click()                       'se clicco sull'interruttore di
abilitazione programma
On Error Resume Next                            'in caso di errori proseguo
If Combo1.ListCount = 0 Then                    'se non è stata rilevata alcuna seriale
    MsgBox "nessuna seriale trovata!!!", vbCritical, "ATTENZIONE"       'visualizzo un
messaggio di avvertimento
    abprg.Value = 0                                 'apro l'interruttore di abilitazione
programma
    Exit Sub                                        'esco dalla sub
End If
If abprg.Value = 1 Then                         'se l'interruttore è chiuso
    Com1.CommPort = Right(Combo1.List(Combo1.ListIndex), 1)         'attivo la porta
selezionata dal combo box
    Com1.PortOpen = True                                            'apro la porta
    If Err Then                                                     'se si è verificato
un errore durante l'apertura
        MsgBox "Impossibile Aprire la porta Selezionata!", vbExclamation, "Attenzione"
'visualizzo un messaggio
        abprg.Value = 0                                                 'apro
l'interruttore
```

```
        Err = 0                                                         'azzero la
variabile di errore
        Exit Sub                                                        'esco dalla sub
    End If
    Text1.Text = "COM " & Right(Combo1.List(Combo1.ListIndex), 1) & " OPENED"
'visualizzo lo stato e la porta aperta
    Text1.ForeColor = vbGreen
'coloro la scritta di verde
Else                                            'se l'interruttore è aperto
    Com1.PortOpen = False                           'chiudo la porta aperta
    Text1.Text = "PORTE CHIUSE"                     'visualizzo lo stato delle porte
    Text1.ForeColor = vbRed                         'coloro il testo di rosso (porte
chiuse)
End If
End Sub


Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)              'premo un
tasto
On Error Resume Next                        'in caso di errori proseguo
If KeyCode = vbKeyControl Then              'se è stato premuto il tasto control (ctrl)
    Check2.Value = 1                            'attivo il comando da tastiera
    focus.SetFocus                              'passo lo stato ativo al text box nascosto
(così posso usare i tasti)
End If
If Check2.Value = 0 Then Exit Sub           'se i comandi da tastiera non sono attivi esco
dalla sub
If Label3.Visible = False Then              'se la priorità non è stata già acquisita
    Beep                                        'emetto un beep
End If
pccmd = 1                                   'acquisisco la priorità
If KeyCode = vbKeyP Then                    'se è stato premuto il tasto P
    If Check1.Value = 1 Then                    'se la modalità pwm è attiva
        Check1.Value = 0                            'disattivo la modalità pwm
    Else                                        'se la modalità pwm non è attiva
        Check1.Value = 1                            'attivo la modalità pwm
    End If
End If
If Check1.Value = 1 Then                    'se è attiva la modalità pwm
    Select Case KeyCode                         'discrimino la pressione dei vari tasti
        Case vbKeyDown                              'premuta la freccia in giù
            If pwsx.Value <> pwdx.Value Then        'se il valore del motore dx è diverso
dal valore del motore sx
                pwsx.Value = 0                          'azzero il valore dei due motori
(dx e sx)
                pwdx.Value = 0
            End If
            pwdx.Value = pwdx.Value + 1             'incremento il valore del motore dx
di una unità
            pwsx.Value = pwsx.Value + 1             'incremento il valore del motore sx
di una unità
            'NB: le slide sono inverse (il minimo si trova verso l'alto)
        Case vbKeyUp                            'premuta la freccia in su
            If pwsx.Value <> pwdx.Value Then        'se il valore del motore dx è diverso
dal valore del motore sx
                pwsx.Value = 0                          'azzero il valore dei due motori
(dx e sx)
                pwdx.Value = 0
            End If
            pwdx.Value = pwdx.Value - 1             'decremento il valore del motore dx
di una unità
            pwsx.Value = pwsx.Value - 1             'decremento il valore del motore sx
di una unità
        Case vbKeyRight                         'premuta freccia destra
            pwdx.Value = pwdx.Value - 1             'decremento il valore del motore dx
di una unità
            pwsx.Value = pwsx.Value + 1             'incremento il valore del motore sx
di una unità
        Case vbKeyLeft                          'premuta freccia sinistra
            pwdx.Value = pwdx.Value + 1             'incremento il valore del motore dx
di una unità
```

```
            pwsx.Value = pwsx.Value - 1             'decremento il valore del motore sx
di una unità
        Case vbKeyPageUp                            'premuto il tasto pag-su
            pwsugiu.Value = pwsugiu.Value - 1       'decremento il valore dei motori
ascensionali di una unità
        Case vbKeyPageDown                          'premuto il tasto pag-giu
            pwsugiu.Value = pwsugiu.Value + 1       'incremento il valore dei motori
ascensionali di una unità
        Case vbKeyEnd                               'premuto il tasto End
            pwdx.Value = 0                          'azzero il valore dei motori dx e sx
            pwsx.Value = 0
        Case vbKeyHome                              'premuto il tasto home
            pwsugiu.Value = 0                       'azzero il valore dei motori
ascensionali
    End Select
Else                                        'se non è attiva la modalità pwm
Select Case KeyCode                         'discrimino la pressione dei vari tasti
    Case vbKeyDown                              'premuta la freccia in giù
            If pwsx.Value <> pwdx.Value Then        'se il valore del motore dx è diverso
dal valore del motore sx
                pwsx.Value = 0                          'azzero il valore dei due motori
(dx e sx)
                pwdx.Value = 0
            End If
            pwdx.Value = 255                    'imposto il valore del motore destro
al massimo (indietro)
            pwsx.Value = 255                    'imposto il valore del motore
sinistro al massimo (indietro)
            'NB: le slide sono inverse (indietro si trova verso l'alto)
        Case vbKeyUp                            'premuta la freccia in su
            If pwsx.Value <> pwdx.Value Then        'se il valore del motore dx è diverso
dal valore del motore sx
                pwsx.Value = 0                          'azzero il valore dei due motori
(dx e sx)
                pwdx.Value = 0
            End If
            pwdx.Value = -255                   'imposto il valore del motore destro
al massimo (avanti)
            pwsx.Value = -255                   'imposto il valore del motore
sinistro al massimo (avanti)
        Case vbKeyRight                         'premuta freccia destra
            pwdx.Value = -255                       'imposto il valore del motore destro
al massimo (avanti)
            pwsx.Value = 255                        'imposto il valore del motore
sinistro al massimo (indietro)
        Case vbKeyLeft                          'premuta freccia sinistra
            pwdx.Value = 255                        'imposto il valore del motore destro
al massimo (indietro)
            pwsx.Value = -255                       'imposto il valore del motore
sinistro al massimo (avanti)
        Case vbKeyPageUp                        'premuto il tasto pag-su
            pwsugiu.Value = -255                    'imposto il valore dei motori
ascensionali al massimo (avanti)
        Case vbKeyPageDown                      'premuto il tasto pag-giu
            pwsugiu.Value = 255                     'imposto il valore dei motori
ascensionali al massimo (indietro)
        Case vbKeyEnd                           'premuto il tasto End
            pwdx.Value = 0                          'azzero il valore dei motori dx e sx
            pwsx.Value = 0
        Case vbKeyHome                          'premuto il tasto home
            pwsugiu.Value = 0                       'azzero il valore dei motori
ascensionali
End Select
End If
If KeyCode = vbKeyZ Then                    'se è stato premuto il tasto Z
    cameragiu.Value = True                      'attivo il tasto "freccia giù"
    tastodown = 1
End If
If KeyCode = vbKeyA Then                    'se è stato premuto il tasto A
    camerasx.Value = True                       'attivo il tasto "freccia sx"
    tastoleft = 1
```

```
End If
If KeyCode = vbKeyS Then                    'se è stato premuto il tasto S
    cameradx.Value = True                       'attivo il tasto "freccia dx"
    tastoright = 1
End If
If KeyCode = vbKeyW Then                    'se è stato premuto il tasto W
    camerasu.Value = True                       'attivo il tasto "freccia su"
    tastoup = 1
End If
If KeyCode = vbKeyQ Then                    'se è stato premuto il tasto Q
    TastQ.Value = True                          'attivo il tasto "Q"
    tastoq = 1
End If
If KeyCode = vbKeyO Then                    'se è stato premuto il tasto O
    TastO.Value = True                          'attivo il tasto "O"
    tastoo = 1
End If
If KeyCode = vbKeyT Then                    'se è stato premuto il tasto T
    tastT.Value = True                          'attivo il tasto "T"
    tastot = 1
End If
If KeyCode = vbKeyX Then                    'se è stato premuto il tasto X
    tastX.Value = True                          'attivo il tasto "X"
    tastox = 1
End If
If KeyCode = vbKeyD Then                    'se è stato premuto il tasto D
    Me.select.Value = True                      'attivo il tasto "select"
    tastosl = 1
End If
If KeyCode = vbKeyF Then                    'se è stato premuto il tasto F
    Start.Value = True                          'attivo il tasto "start"
    tastost = 1
End If
If KeyCode = vbKeyL Then                    'se è stato premuto il tasto L
    If laser.Value = 1 Then                     'se l'interruttore del laser è chiuso
        laser.Value = 0                             'lo apro
    Else                                        'altrimenti
        laser.Value = 1                             'lo chiudo
    End If
End If
If KeyCode = vbKeyC Then                    'se è stato premuto il tasto C
    If camera.Value = 1 Then                    'se l'interruttore della telecamera è
chiuso
        camera.Value = 0                            'lo apro
    Else                                        'altrimenti
        camera.Value = 1                            'lo chiudo
    End If
End If
End Sub


Private Sub Form_Load()                     'al caricamento del form
On Error Resume Next                        'in caso di errori proseguo
Dim contat As Integer                       'dichiaro una variabile contatore
Dim nport As Integer                        'contatore porte seriali
ReDim Valori(0 To CommSet.PackNumber - 1)   'ridefinisco la matrice dei telegrammi
ricevuti con grandezza pari al numero di telegrammi del pacchetto
For contat = 0 To UBound(Valori)            'carico il valore zero in ogni indice
    Valori(contat) = 0
Next contat
ricezione.Show                              'visualizzo il form dei sensori in
ricezione
ricezione.WindowState = vbMinimized         'e lo riduco a icona
joy.Show                                    'visualizzo il form del joypad
joy.WindowState = vbMinimized               'e lo riduco a icona
Com1.Settings = "2400,n,8,1"                'setto l'oggetto com: baudrate 2400, 8
bit, 1 bit di stop, nessuna parità
Com1.Handshaking = comRTS                   'attivo l'handshaking RTS
Com1.InputLen = 1                           'posso prelevare solo un byte per volta
dal buffer di ricezione
Com1.RThreshold = 1                         'genero un evento di ricezione ad ogni
```

```
byte ricevuto
Com1.RTSEnable = True                           'abilito il canale RTS
wdg.Enabled = True                              'attivo il watchdog
For nport = 1 To 20                             'ipotizzo un numero massimo di 20 porte
seriali e ne eseguo la scansione
    Com1.CommPort = nport                           'seleziono una porta
    Com1.PortOpen = True                            'la apro
    If Not CBool(Err) Then                          'se non si verifica alcun errore
(porta aperta)
        Combo1.AddItem "com " & nport                   'aggiungo la porta nella lista
combobox (con il prefisso com)
        Com1.PortOpen = False                           'chiudo la porta aperta
    End If
    Err = 0                                         'azzero il flag di errore (e non
inserisco niente)
Next nport                                          'analizzo tutte le porte
Combo1.ListIndex = 0                            'seleziono il primo elemento della lista
combo
'nascondo tutte le frecce direzionali degli slide (per la direzione dei motori)
Line2(0).Visible = False
Line3(0).Visible = False
Line4(0).Visible = False
Line5(0).Visible = False
Line6(0).Visible = False
Line7(0).Visible = False
Line2(1).Visible = False
Line3(1).Visible = False
Line4(1).Visible = False
Line5(1).Visible = False
Line6(1).Visible = False
Line7(1).Visible = False
Line2(2).Visible = False
Line3(2).Visible = False
Line4(2).Visible = False
Line5(2).Visible = False
Line6(2).Visible = False
Line7(2).Visible = False
Label3.Visible = False              'nascondo la scritta di acquisizione priorità
Combo1.Text = Combo1.List(0)        'scrivo nel campo di testo del combobox la stringa
del primo elemento
If Combo1.Text = "" Then Combo1.Text = "Porte Seriali"      'se non vi sono elementi
nella lista visualizzo il nome della lista
End Sub


Private Sub pwdx_Change()           'se il valore della slide del motore destro cambia
On Error Resume Next                'in caso di errori prosegue
Select Case pwdx.Value              'discriminazione valore slide
    Case Is < 0                         'il valore è minore di zero (verso l'alto)
        direzdx = 0                     'la direzione è avanti
        pccmd = 1                       'acquisisco la priorità
        'visualizzo la freccia verso l'alto e nascondo quella verso il basso
        Line2(1).Visible = True
        Line3(1).Visible = True
        Line4(1).Visible = True
        Line5(1).Visible = False
        Line6(1).Visible = False
        Line7(1).Visible = False
    Case Is > 0                         'il valore è maggiore di zero (verso il basso)
        direzdx = 1                         'la direzione è indietro
        pccmd = 1                           'acquisisco la priorità
        'visualizzo la freccia verso il basso e nascondo quella verso l'alto
        Line5(1).Visible = True
        Line6(1).Visible = True
        Line7(1).Visible = True
        Line2(1).Visible = False
        Line3(1).Visible = False
        Line4(1).Visible = False
    Case 0                              'il valore è zero (in centro)
        direzdx = 0                         'la direzione è avanti
        pccmd = 0                           'rilascio la priorità
```

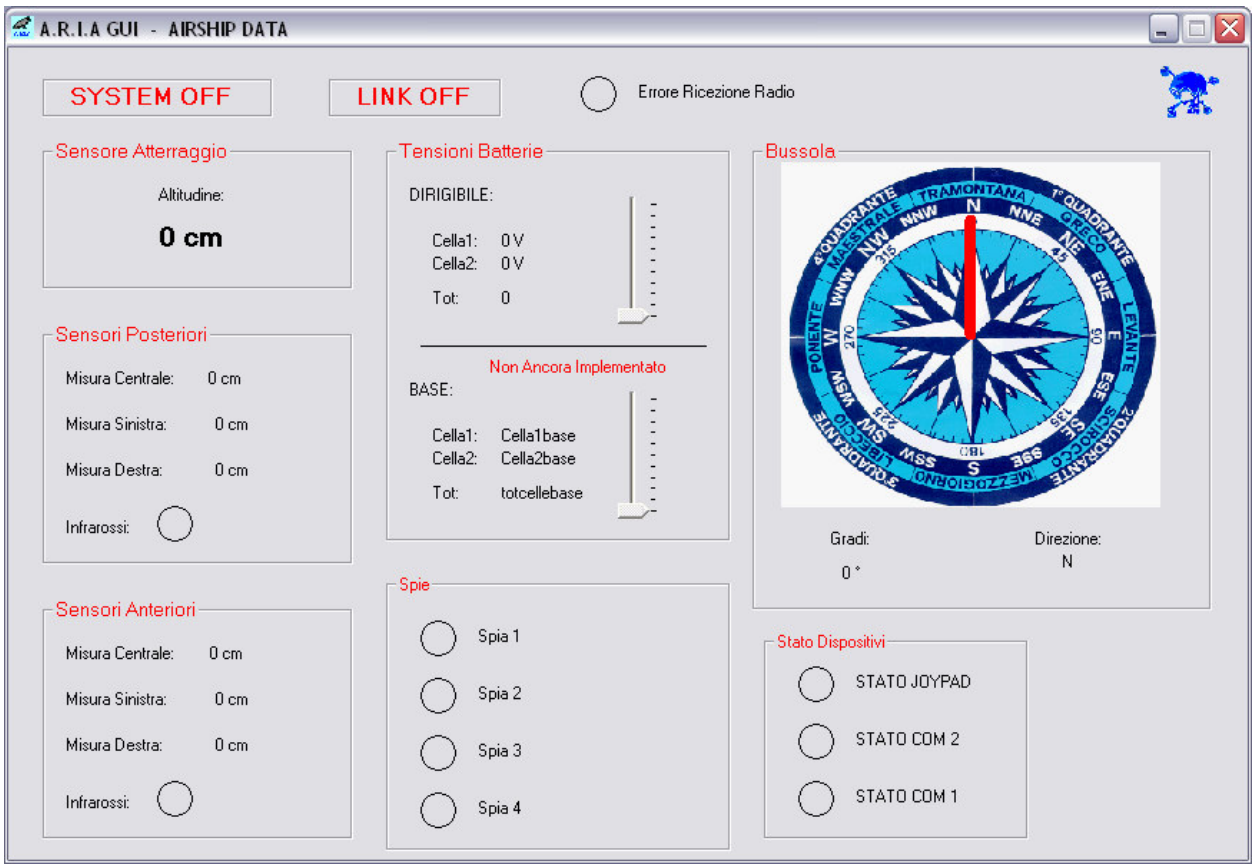
```
        'Nascondo le frecce
        Line5(1).Visible = False
        Line6(1).Visible = False
        Line7(1).Visible = False
        Line2(1).Visible = False
        Line3(1).Visible = False
        Line4(1).Visible = False
End Select
End Sub


Private Sub pwsugiu_Change()        'se il valore della slide dei motori ascensionali
cambia
On Error Resume Next                'in caso di errori prosegue
Select Case pwsugiu.Value           'discriminazione valore slide
    Case Is < 0                         'il valore è minore di zero (verso l'alto)
        direzdown = 0                       'la direzione è salita
        pccmd = 1                           'acquisisco la priorità
        'visualizzo la freccia verso l'alto e nascondo quella verso il basso
        Line2(2).Visible = True
        Line3(2).Visible = True
        Line4(2).Visible = True
        Line5(2).Visible = False
        Line6(2).Visible = False
        Line7(2).Visible = False
    Case Is > 0                         'il valore è maggiore di zero (verso il basso)
        direzdown = 1                       'la direzione è discesa
        pccmd = 1                           'acquisisco la priorità
        'visualizzo la freccia verso il basso e nascondo quella verso l'alto
        Line5(2).Visible = True
        Line6(2).Visible = True
        Line7(2).Visible = True
        Line2(2).Visible = False
        Line3(2).Visible = False
        Line4(2).Visible = False
    Case 0                              'il valore è zero (in centro)
        direzdown = 0                       'la direzione è salita
        pccmd = 0                           'rilascio la priorità
        'Nascondo le frecce
        Line5(2).Visible = False
        Line6(2).Visible = False
        Line7(2).Visible = False
        Line2(2).Visible = False
        Line3(2).Visible = False
        Line4(2).Visible = False
End Select
End Sub


Private Sub pwsx_Change()       'se il valore della slide del motore sinistro cambia
On Error Resume Next            'in caso di errori proseguo
Select Case pwsx.Value          'discriminazione valore slide
    Case Is < 0                     'il valore è minore di zero (verso l'alto)
        direzsx = 0                     'la direzione è avanti
        pccmd = 1                       'acquisisco la priorità
        'visualizzo la freccia verso l'alto e nascondo quella verso il basso
        Line2(0).Visible = True
        Line3(0).Visible = True
        Line4(0).Visible = True
        Line5(0).Visible = False
        Line6(0).Visible = False
        Line7(0).Visible = False
    Case Is > 0                     'il valore è maggiore di zero (verso il basso)
        direzsx = 1                     'la direzione è indietro
        pccmd = 1                       'acquisisco la priorità
        'visualizzo la freccia verso il basso e nascondo quella verso l'alto
        Line5(0).Visible = True
        Line6(0).Visible = True
        Line7(0).Visible = True
        Line2(0).Visible = False
        Line3(0).Visible = False
```

```
        Line4(0).Visible = False
    Case 0                          'il valore è zero (in centro)
        direzsx = 0                     'la direzione è avanti
        pccmd = 0                       'rilascio la priorità
        'Nascondo le frecce
        Line5(0).Visible = False
        Line6(0).Visible = False
        Line7(0).Visible = False
        Line2(0).Visible = False
        Line3(0).Visible = False
        Line4(0).Visible = False
End Select
End Sub


Private Sub select_MouseDown(Button As Integer, Shift As Integer, x As Single, Y As
Single)     'Premo il tasto Select
tastosl = 1         'attivo il tasto
pccmd = 1           'acquisisco la priorità
End Sub

Private Sub select_MouseUp(Button As Integer, Shift As Integer, x As Single, Y As Single)
'Rilascio il tasto Select
tastosl = 0         'disattivo il tasto
pccmd = 0           'rilascio la priorità
End Sub

Private Sub Start_MouseDown(Button As Integer, Shift As Integer, x As Single, Y As
Single)      'Premo il tasto Start
tastost = 1         'attivo il tasto
pccmd = 1           'acquisisco la priorità
End Sub

Private Sub Start_MouseUp(Button As Integer, Shift As Integer, x As Single, Y As Single)
'Rilascio il tasto Start
tastost = 0         'disattivo il tasto
pccmd = 0           'rilascio la priorità
End Sub

Private Sub TastO_MouseDown(Button As Integer, Shift As Integer, x As Single, Y As
Single)      'Premo il tasto O
tastoo = 1          'attivo il tasto
pccmd = 1           'acquisisco la priorità
End Sub

Private Sub TastO_MouseUp(Button As Integer, Shift As Integer, x As Single, Y As Single)
'Rilascio il tasto O
tastoo = 0          'disattivo il tasto
pccmd = 0           'rilascio la priorità
End Sub

Private Sub TastQ_MouseDown(Button As Integer, Shift As Integer, x As Single, Y As
Single)      'Premo il tasto Q
tastoq = 1          'attivo il tasto
pccmd = 1           'acquisisco la priorità
End Sub

Private Sub TastQ_MouseUp(Button As Integer, Shift As Integer, x As Single, Y As Single)
'Rilascio il tasto Q
tastoq = 0          'disattivo il tasto
pccmd = 0           'rilascio la priorità
End Sub

Private Sub tastT_MouseDown(Button As Integer, Shift As Integer, x As Single, Y As
Single)      'Premo il tasto T
tastot = 1          'attivo il tasto
pccmd = 1           'acquisisco la priorità
End Sub

Private Sub tastT_MouseUp(Button As Integer, Shift As Integer, x As Single, Y As Single)
'Rilascio il tasto T
```

```
tastot = 0          'disattivo il tasto
pccmd = 0           'rilascio la priorità
End Sub

Private Sub tastX_MouseDown(Button As Integer, Shift As Integer, x As Single, Y As
Single)      'Premo il tasto X
tastox = 1          'attivo il tasto
pccmd = 1           'acquisisco la priorità
End Sub

Public Sub trasmpack()              'Trasmette i comandi alla base terrestre attraverso
la porta com aperta (previa richiesta di invio)
On Error Resume Next                'In caso di errori prosegue
'Variabili Per il calcolo e l'assegnamento dei valori definitivi inviare alla base
(comandi dati)
Dim basestate As Integer, joypadbuttons As String, joyanddirez As String, pwmmotdx As
Integer, pwmmotsx As Integer, pwmmotdown As Integer, prioritydev As String, pcstate As
String
'Variabili per la gestione della priorità dei dispositivi della base terrestre (comandi
priorità)
Dim lowpriority As Integer, highpriority As Integer
'Variabile Buffer in uscita (Pacchetto completo sparato fuori dalla com)
Dim outbuffer As String
basestate = 255                     'flags di stato della base (inviati al velivolo) -
255 = tutti i flags on
pwmmotsx = Abs(pwsx.Value)          'calcolo ed assegno la velocità del motore sinistro
pwmmotdx = Abs(pwdx.Value)          'calcolo ed assegno la velocità del motore destro
pwmmotdown = Abs(pwsugiu.Value)     'calcolo ed assegno la velocità dei motori
ascensionali
'Costruisco il byte dei tasti funzione (tasti joypad) - Include tasti funzione e
attivazione microcamera e laser (modulo ciclopè)
joypadbuttons = CStr(laser.Value) & CStr(camera.Value) & CStr(tastosl) & CStr(tastost) &
CStr(tastoo) & CStr(tastot) & CStr(tastoq) & CStr(tastox)
'Costruisco il byte delle direzioni - Include il pilotaggio della telecamera (direzioni)
e le direzioni dei motori
joyanddirez = "0" & CStr(direzdown) & CStr(direzsx) & CStr(direzdx) & CStr(tastoleft) &
CStr(tastodown) & CStr(tastoright) & CStr(tastoup)
'Setto la priorità
If Option2.Value = True Then        'se è stata selezionata la priorità alta
(disattivazione completa joypad)
    lowpriority = 1                     'attivo il flag di bassa priorità (computer come
dispositivo comunicante)
    highpriority = 1                    'attivo il flag di alta priorità (causa la
disattivazione del joypad)
Else                                'se è stata selezionata la priorità bassa (il
computer ha la priorità solo quando si inviano comandi)
    highpriority = 0                    'disattivo il flag di alta priorità
(disabilitazione joypad)
    If pccmd = 1 Then                   'se è stata acquisita la priorità
        lowpriority = 1                     'attivo il flag di bassa priorità (controllo
da pc solo durante invio comandi)
    Else                                'se la priorità non è ascquisita (per esempio è
stata rilasciata)
        lowpriority = 0                     'disattivo il flag di bassa priorità
    End If
End If
'Costruisco il byte di controllo priorità (gli zeri sono i bit non implementati"
prioritydev = "0" & "0" & "0" & "0" & "0" & CStr(lowpriority) & CStr(highpriority) & "0"
'Invio i flags di stato del pc (gli zeri sono i bit non implementati)
pcstate = "0" & "0" & "0" & "0" & "0" & "0" & "0" & "1"
'Carico tutto nel buffer di uscita (in formato stringa) - Il buffer di uscita è composto
da tutti i telegrammi in sequenza
'che assieme formano il pacchetto comandi in uscita - Il pacchetto inizia con 2
preamboli, seguono i dati e poi un postambolo
outbuffer = Chr(compre1) & Chr(compre2) & Chr(basestate) &
Chr(binary.bintodec(joypadbuttons)) & Chr(binary.bintodec(joyanddirez)) & _
Chr(pwmmotdx) & Chr(pwmmotsx) & Chr(pwmmotdown) & Chr(prioritydev) & Chr(pcstate) &
Chr(compost)
'Carico i dati nel buffer hardware (invio automatico)
Com1.Output = outbuffer         'invio i dati (comandi) alla base terrestre
End Sub
```

```
Private Sub tastX_MouseUp(Button As Integer, Shift As Integer, x As Single, Y As Single)
'Rilascio il tasto X
tastox = 0          'disattivo il tasto
pccmd = 0           'acquisisco la priorità
End Sub

Private Sub Text1_GotFocus()        'in caso venga per sbaglio messo a fuoco il textbox
dello stato porta seriale
abprg.SetFocus                      'sposto lo stato attivo sull'interruttore di
abilitazione programma
End Sub

Private Sub Timer1_Timer()          'timer per visualizzazione segnalazioni etc
'Se gli slide dei motori hanno valore diverso da zero ed il laser oppure la telecamera
sono accesi
If pwdx.Value <> 0 Or pwsx.Value <> 0 Or pwsugiu.Value <> 0 Or laser.Value <> 0 Or
camera.Value <> 0 Then
    pccmd = 1           'forzo l'acquisizione della priorità
End If
If pccmd = 1 Then           'se la priorità è acquisita
    Label3.Visible = True       'visualizzo la label indicativa
Else                        'altrimenti
    Label3.Visible = False      'nascondo la label indicativa
End If
If Option2.Value = True Then    'se è attiva l'alta priorità
    Label3.Visible = True           'visualizzo la label di "priorità acquisita"
End If
End Sub

Private Sub wdg_Timer()             'Timer watchdog (indica che la base non è collegata)
On Error Resume Next                'In caso di errori proseguo
Dim cont As Integer                 'dichiaro una variabile contatore
watchdog = watchdog + 1             'incremento il valore del watchdog (timer impostato a
1ms) ad ogni ciclo
If watchdog = dogtimeout Then       'se ho raggiunto il timeout
    watchdog = 0                        'azzero il watchdog (i conteggio ricomincia da 0)
    For cont = 0 To UBound(CommSet.Valori)      'per ogni indice della matrice contenente
i telegrammi del pacchetto (dati)
        CommSet.Valori(cont) = 0                    'azzero il valore (azzero dati in
ingresso)
    Next cont                                   'conto fino all'ultimo indice della
variabile matriciale
End If
End Sub
```

FORM AIRSHIP DATA

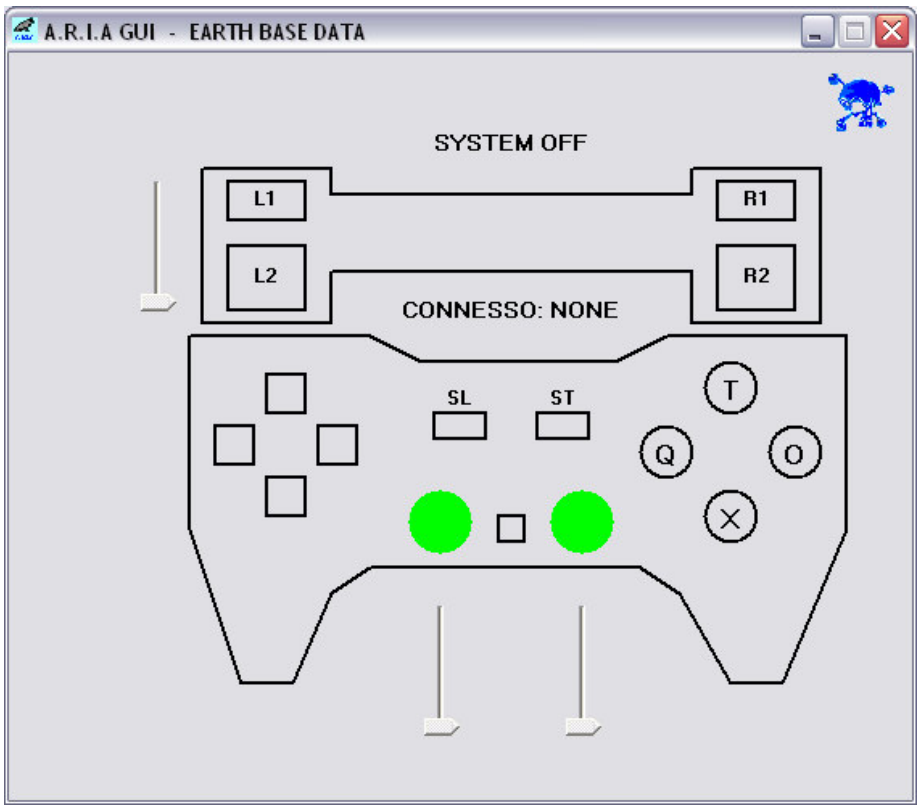


```
Option Explicit             'le variabili devono essere tutte dichiarate
Private Declare Function InitCommonControls Lib "Comctl32.dll" () As Long
'attivo la possibilità di usare la grafica XP (manifest)
'variabili dati da visualizzare
Public compass, celladir1 As String, celladir2 As String, cella1e2 As String, sensors As
String, cell1battdir As Double, cell2battdir As Double
'Variabile consenso unload form
Public totunload As Boolean
'Costanti posizione in gradi rosa dei venti
Const N = 0, NNE = 22.5, NE = 45, ENE = 77.5, E = 90, ESE = 112.5, SE = 135, SSE = 157.5,
S = 180, SSO = 202.5, SO = 225, OSO = 247.5, O = 270, ONO = 292.5, NO = 315, NNO = 337.5,
INTERVALL = 11.25


Private Sub Form_Initialize()           'attiva la grafica xp in caso sia presente il
file manifest
    Dim x As Long
    x = InitCommonControls
End Sub

Private Sub Form_Load()                 'al caricamento del form
levdir.Value = 10       'azzero gli slide delle tensioni
levbase.Value = 10
'spengo i led degli ostacoli rilevati dagli infrarossi
infrapost.FillStyle = 1
infraant.FillStyle = 1
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)      'alla
chiusura del programma
On Error Resume Next                'in caso di errori prosegue
If totunload = False Then           'se non c'è il consenso per uscire
    Cancel = 1                          'non esco
Else                                'altrimenti
    Cancel = 0                          'esco
End If
End Sub

Private Sub Timer1_Timer()              'carica, elabora e visualizza i dati inviati dal
velivolo alla base
On Error Resume Next                    'in caso di errori prosegue
'Variabili bytes flags di segnalazione e di stato letti
Dim segnalaz As String, devices As String, pcstate As String, picmaster As String
'spengo tutti i led di segnalazione
Shape1(0).FillStyle = 1
Shape1(1).FillStyle = 1
Shape1(2).FillStyle = 1
Shape1(3).FillStyle = 1
Shape2(1).FillStyle = 1
Shape2(2).FillStyle = 1
Shape3.FillStyle = 1
If comandi.abprg.Value = 1 Then                 'se il programma è attivo (porta aperta)
    If Label21.Caption = "SYSTEM OFF" Then      'se il sistema non è attivo (base spenta)
        Shape2(0).FillStyle = 1                     'spengo il led "com1 attiva"
    Else                                        'altrimenti
        Shape2(0).FillStyle = 0                     'accendo il led "com1 attiva"
    End If
Else                                            'se il programma è attivo
    Shape2(0).FillStyle = 1                         'spengo il led "com 1 attiva"
End If
'Calcolo posizione velivolo (bussola digitale)
compass = Fix(1.408 * CommSet.Valori(8))        'Leggo il byte della bussola - TEL 8
Select Case compass                             'discriminazione dei gradi letti per
visualizzare la sigla della rosa dei venti
    'Visualizzo nella label le sigle dei punti della rosa dei venti a seconda delle fasce
in cui si trova la lancetta
    'della bussola. Uso i simboli inglesi (N-S-E-W e combinazioni)
    Case N - INTERVALL To N + INTERVALL
        Label26.Caption = "N"
    Case NNE - INTERVALL To NNE + INTERVALL
        Label26.Caption = "NNE"
```

```
    Case NE - INTERVALL To NE + INTERVALL
        Label26.Caption = "NE"
    Case ENE - INTERVALL To ENE + INTERVALL
        Label26.Caption = "ENE"
    Case E - INTERVALL To E + INTERVALL
        Label26.Caption = "E"
    Case ESE - INTERVALL To ESE + INTERVALL
        Label26.Caption = "ESE"
    Case SE - INTERVALL To SE + INTERVALL
        Label26.Caption = "SE"
    Case SSE - INTERVALL To SSE + INTERVALL
        Label26.Caption = "SSE"
    Case S - INTERVALL To S + INTERVALL
        Label26.Caption = "S"
    Case SSO - INTERVALL To SSO + INTERVALL
        Label26.Caption = "SSW"
    Case SO - INTERVALL To SO + INTERVALL
        Label26.Caption = "SW"
    Case OSO - INTERVALL To OSO + INTERVALL
        Label26.Caption = "WSW"
    Case O - INTERVALL To O + INTERVALL
        Label26.Caption = "W"
    Case ONO - INTERVALL To ONO + INTERVALL
        Label26.Caption = "WNW"
    Case NO - INTERVALL To NO + INTERVALL
        Label26.Caption = "NW"
    Case NNO - INTERVALL To NNO + INTERVALL
        Label26.Caption = "NNW"
End Select
'Visualizzo il valore in gradi della bussola nella label
Label19.Caption = 1.408 * CommSet.Valori(8) & " °"
'Valcolo la posizione dell'ago della bussola e visualizzo la lancetta nel disegno della
rosa dei venti
Call Bussola.Bussola(1.408 * CommSet.Valori(8), 1200, Line1)
'Leggo i valori di fuffy (devono essere moltiplicati per 2 per avere l'unità in cm) e li
visualizzo
'Le posizioni dei fuffy sono indicate in riferimento al pallone visto da dietro
fuffypostcent.Caption = CommSet.Valori(2) * 2 & " cm"       'Misura Post-Centro - TEL 2
fuffypostsx.Caption = CommSet.Valori(6) * 2 & " cm"         'Misura Post-Sinistra - TEL 6
fuffypostdx.Caption = CommSet.Valori(4) * 2 & " cm"         'Misura Post-Destra - TEL 4
fuffyantcent.Caption = CommSet.Valori(1) * 2 & " cm"        'Misura Ant-Centro - TEL 1
Fuffyantsx.Caption = CommSet.Valori(5) * 2 & " cm"          'Misura Ant-Sinistra - TEL 5
Fuffyantdx.Caption = CommSet.Valori(3) * 2 & " cm"          'Misura Ant-Destra - TEL 3
fuffysotto.Caption = CommSet.Valori(7) * 2 & " cm"          'Misura Sotto - TEL 7
'Lettura valori batterie dirigibile
cella1e2 = binary.dectobin(CLng(CommSet.Valori(10)))    'Byte Celle - TEL 10
celladir1 = Right(cella1e2, 4)      'cella 1 - Nibble Basso
celladir2 = Left(cella1e2, 4)       'cella 2 - Nibble Alto
If Label25.Caption = "LINK OFF" Then        'se il link radio non è attivo
    cell1battdir = 0                            'azzero il livello delle tensioni (slide
azzerato)
    cell2battdir = 0
Else                                        'se il link radio è attivo
    'eseguo il calcolo per risalire alla tensione (approssimata) delle celle del
dirigibile
    'moltiplico per 0,166666666 e sommo 2,5
    cell1battdir = Left(binary.bintodec(celladir1) * 0.16666666 + 2.5, 4)
    cell2battdir = Left(binary.bintodec(celladir2) * 0.16666666 + 2.5, 4)
End If
'Visualizzo i valori delle tensioni del dirigibile nelle label
Cella1dir.Caption = cell1battdir & " V"
Cella2dir.Caption = cell2battdir & " V"
totcelledir.Caption = cell1battdir + cell2battdir
'Visualizzo il livello delle batterie negli slide
levdir.Value = 10 - totcelledir
'Carico i valori dei sensori extra (infrarossi) e visualizzo
sensors = binary.dectobin(CLng(CommSet.Valori(9)))    'Byte sensori - TEL 9
infrapost.FillStyle = Val(Mid(sensors, 7, 1))         'Stato Infrarosso Posteriore - Bit
7
infraant.FillStyle = Val(Mid(sensors, 8, 1))          'Stato Infrarosso Anteriore - Bit 8
'Carico i flags di segnalazione
```

```
segnalaz = binary.dectobin(CLng(CommSet.Valori(16)))        'Byte flags di segnalazione
(spie) - TEL 16
'Visualizzo i led di segnalazione
Shape1(0).FillStyle = Not CBool(Val(Mid(segnalaz, 8, 1)))   'Led 1 - Bit 8
Shape1(1).FillStyle = Not CBool(Val(Mid(segnalaz, 7, 1)))   'Led 2 - Bit 7
Shape1(2).FillStyle = Not CBool(Val(Mid(segnalaz, 6, 1)))   'Led 3 - Bit 6
Shape1(3).FillStyle = Not CBool(Val(Mid(segnalaz, 5, 1)))   'Led 4 - Bit 5
'Leggo i flags di stato dei dispositivi della base e visualizzo
devices = binary.dectobin(CLng(CommSet.Valori(15)))         'Byte dispositivi - TEL 15
Shape2(1).FillStyle = Not CBool(Val(Mid(devices, 6, 1)))    'Stato porta com2 Base - Bit
6
Shape2(2).FillStyle = Not CBool(Val(Mid(devices, 8, 1)))    'Stato Joypad
(connesso/disconnesso) - Bit 8
'Carico i flags di segnalazione replicati per la conferma dello stato della base
pcstate = binary.dectobin(CLng(CommSet.Valori(30)))     'Byte stato pc - TEL 30
'Carico i flags di stato del pic master della Base Terrestre
picmaster = binary.dectobin(CLng(CommSet.Valori(17)))   'Byte Picmaster - TEL 17
If Val(Mid(pcstate, 7, 1)) = 1 Then     'se il flag replicato dello stato pc è attivo -
Bit 7
    Label21.Caption = "SYSTEM ON"           'indico che il sistema è attivo
Else                                    'altrimenti
    Label21.Caption = "SYSTEM OFF"          'indico che il sistema non è attivo
End If
If Val(Mid(picmaster, 8, 1)) = 1 Then   'se il picmaster è attivo - Bit 8
    Label25.Caption = "LINK ON"             'indico che il link è attivo
Else                                    'altrimenti
    Label25.Caption = "LINK OFF"            'indico che il link non è attivo
End If
'Segnalo errori nella ricezione radio
Shape3.FillStyle = Not CBool(Val(Mid(picmaster, 7, 1)))        'Flag errore ricezione
radio - Bit 7
'se il sistema non è attivo oppure il link non è attivo
If Label21.Caption = "SYSTEM OFF" Or Label25.Caption = "LINK OFF" Then
    'spengo i led di errore radio e segnalazione infrarossi (perchè negati)
    Shape3.FillStyle = 1
    infrapost.FillStyle = 1
    infraant.FillStyle = 1
End If
If Label21.Caption = "SYSTEM OFF" Then          'se il sistema non è attivo
    Label21.ForeColor = vbRed                       'visualizzo la scritta in rosso
Else                                            'altrimenti
    Label21.ForeColor = vbGreen                     'visualizzo la scritta in verde
End If
If Label25.Caption = "LINK OFF" Then            'se il link non è attivo
    Label21.ForeColor = vbRed                       'visualizzo la scritta in rosso
Else                                            'altrimenti
    Label21.ForeColor = vbGreen                     'visualizzo la scritta in verde
End If
End Sub
```

FORM EARTH BASE DATA



```
Option Explicit                 'le variabili devono essere tutte dichiarate
Private Declare Function InitCommonControls Lib "Comctl32.dll" () As Long
'attivo la possibilità di usare la grafica XP (manifest)
'Variabili dati letti dalla base
Public joypadbuttons As String, joyanddirez As String, pwmmotsx As Integer, pwmmotdx As
Integer, pwmmotdown As Integer, priority As String
'Variabile consenso unload del form
Public totunload As Boolean

Private Sub Form_Initialize()       'attiva la grafica xp in caso sia presente il file
manifest
    Dim x As Long
    x = InitCommonControls
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, UnloadMode As Integer)          'quando
esco dal programma
On Error Resume Next                    'in caso di errori proseguo
If totunload = False Then               'se non ho il permesso di uscire
    Cancel = 1                              'blocco la chiusura del programma
Else                                    'altrimenti
    Cancel = 0                              'il programma può essere chiuso
End If
End Sub

Private Sub Timer1_Timer()              'timer di lettura e visualizzazione dati
dispositivi di comando
On Error Resume Next                    'in caso di errori prosegue
Label1.Caption = ricezione.Label21.Caption      'Visualizzo lo stato del sistema (base
terrestre accesa o spenta)
If Label1.Caption = "SYSTEM ON" Then            'se la base è accesa
    Frame1.Visible = True                           'visualizzo l'icona del pc collegato
all'interfaccia
    Line32.Visible = True
Else                                            'se la base è spenta
    Frame1.Visible = False                      'visualizzo l'icona del pc collegato
all'interfaccia
    Line32.Visible = False
End If
'Spengo tutti gli indicatori (pulsanti e spie)
funx.FillStyle = 1
funq.FillStyle = 1
funo.FillStyle = 1
funt.FillStyle = 1
selectt.FillStyle = 1
start.FillStyle = 1
laser.FillStyle = 1
camera.FillStyle = 1
camerasu.FillStyle = 1
cameradx.FillStyle = 1
cameragiu.FillStyle = 1
camerasx.FillStyle = 1
analdxsu.FillStyle = 1
analsxsu.FillStyle = 1
analdxgiu.FillStyle = 1
analsxgiu.FillStyle = 1
motascsu.FillStyle = 1
motascgiu.FillStyle = 1
'Leggo il byte dei tasti del joypad (lo converto in stringa binaria per separare i bit) -
TEL 24
joypadbuttons = binary.dectobin(CLng(CommSet.Valori(24)))
'Segnalo i tasti premuti (gli shape hanno logica inversa)
funx.FillStyle = Not CBool(Val(Mid(joypadbuttons, 8, 1)))       'tasto X - bit 8
funq.FillStyle = Not CBool(Val(Mid(joypadbuttons, 7, 1)))       'tasto Q - bit 7
funo.FillStyle = Not CBool(Val(Mid(joypadbuttons, 5, 1)))       'tasto O - bit 5
funt.FillStyle = Not CBool(Val(Mid(joypadbuttons, 6, 1)))       'tasto T - bit 6
selectt.FillStyle = Not CBool(Val(Mid(joypadbuttons, 3, 1)))    'tasto select - bit 3
start.FillStyle = Not CBool(Val(Mid(joypadbuttons, 4, 1)))      'tasto start - bit 4
laser.FillStyle = Not CBool(Val(Mid(joypadbuttons, 1, 1)))      'tasto laser (R2) - bit 1
camera.FillStyle = Not CBool(Val(Mid(joypadbuttons, 2, 1)))     'tasto camera (R1) - bit
2
```

```
'Leggo il byte delle direzioni (lo converto in stringa binaria per separare i bit) - TEL
25
joyanddirez = binary.dectobin(CLng(CommSet.Valori(25)))
'Segnalo il tasto premuto (freccette) e quindi la direzione della telecamera
camerasu.FillStyle = Not CBool(Val(Mid(joyanddirez, 8, 1)))     'tasto su - bit 8
cameradx.FillStyle = Not CBool(Val(Mid(joyanddirez, 7, 1)))     'tasto dx - bit 7
cameragiu.FillStyle = Not CBool(Val(Mid(joyanddirez, 6, 1)))    'tasto giu - bit 6
camerasx.FillStyle = Not CBool(Val(Mid(joyanddirez, 5, 1)))     'tasto sx - bit 5
'Carico i bytes contenenti la velocità dei motori (utile con la modalità pwm)
pwmmotdx = CInt(CommSet.Valori(26))         'motore destro - TEL 26
pwmmotsx = CInt(CommSet.Valori(27))         'motore sinistro - TEL 27
pwmmotdown = CInt(CommSet.Valori(28))       'motori sotto - TEL 28
'Visualizzo sugli slide la velocità (gli slide sono rovesci)
Slider1.Value = 255 - pwmmotdown
Slider2.Value = 255 - pwmmotsx
Slider3.Value = 255 - pwmmotdx
'In base alla direzione dei motori ascensionali segnalo con i tasti L1 ed L2 (pilotaggio
motori sotto)
If pwmmotdown > 0 Then                  'se la velocità non è zero
    motascsu.FillStyle = Val(Mid(joyanddirez, 2, 1))                'carico e imposto i
led dei pulsanti
    motascgiu.FillStyle = Not CBool(Val(Mid(joyanddirez, 2, 1)))    'Direzione Motori
sotto - Bit 2
Else                                    'altrimenti
    motascsu.FillStyle = 1                                          'spengo i segnalatori
(motori spenti)
    motascgiu.FillStyle = 1
End If
'In base alla direzione del motore destro segnalo con le levette analogiche (leva destra)
If pwmmotdx > 0 Then                    'se la velocità non è zero
    analcentdx.FillColor = vbYellow         'il pallino centrale è giallo
    analdxsu.FillStyle = Val(Mid(joyanddirez, 4, 1))                    'accendo il
pallino della direzione letta
    analdxgiu.FillStyle = Not CBool(Val(Mid(joyanddirez, 4, 1)))        'Direzione DX -
Bit 4
Else                                    'se la velocità è zero
    analcentdx.FillColor = vbGreen          'il pallino centrale è verde (leva in centro)
    analdxsu.FillStyle = 1                  'spengo i segnalatori della leva avanti e
della leva indietro
    analdxgiu.FillStyle = 1
End If
'In base alla direzione del motore sinistro segnalo con le levette analogiche (leva
sinistra)
If pwmmotsx > 0 Then                    'se la velocità non è zero
    analcentsx.FillColor = vbYellow         'il pallino centrale è giallo
    analsxsu.FillStyle = Val(Mid(joyanddirez, 3, 1))                        'accendo il
pallino della direzione letta
    analsxgiu.FillStyle = Not CBool(Val(Mid(joyanddirez, 3, 1)))            'Direzione SX
- Bit 3
Else                                    'se la velocità è zero
    analcentsx.FillColor = vbGreen          'il pallino centrale è verde (leva in centro)
    analsxsu.FillStyle = 1                  'spengo i segnalatori della leva avanti e
della leva indietro
    analsxgiu.FillStyle = 1
End If
'Leggo il byte dei flags di priorità e lo converto in una stringa binaria per separare i
bit - TEL 29
priority = binary.dectobin(CLng(CommSet.Valori(29)))
If Val(Mid(priority, 6, 1)) = 1 Then        'se il flag è attivo (il PC ha la priorità) -
Bit 6
    Label2.Caption = "CONNESSO: PC"             'indico che il pc è connesso come
dispositivo di comando
    Shape21.FillStyle = 1                       'spengo il led di segnalazione "joypad
acceso"
Else                                        'se il flag indica che il pc non ha la
priorità
    If ricezione.Shape2(2).FillStyle = 0 Then       'se il joypad è connesso
        Label2.Caption = "CONNESSO: PSX"                'indico che il dispositivo di
comando è il joypad
        Shape21.FillStyle = 0                           'accendo il segnalatore "joypad
acceso"
```

```
    Else                                                    'se il joypad non è connesso
        If Label1.Caption = "SYSTEM OFF" Then                   'se la base è spenta o non
risponde
            Label2.Caption = "CONNESSO: NONE"                       'indico che nessun
dispositivo è connesso e attivo
            Shape21.FillStyle = 1                                   'spengo il led di
segnalazione "joypad acceso"
        Else                                                    'se la base è accesa, ma il
joypad è scollegato
            Label2.Caption = "CONNESSO: PC"                         'indico che il pc è connesso
            Shape21.FillStyle = 1                                   'spengo il led di
segnalazione "joypad acceso"
        End If
    End If
End If
End Sub
```

MODULO COMMSET (DIVISIONE PACCHETTI)

```
'Buffer per la ricezione e l'elaborazione dei pacchetti
Dim Buffer As String, Buffer2 As String, Buffer3 As String
'valori dei preamboli e postambolo usati nella comunicazione
Const Pre1 = "170", Pre2 = "85", Post = "170"
'Costanti altre impostazioni
Global Const PackNumber = 31        'numero telegrammi che compongono il pacchetto
'matrice contenente i dati del pacchetto (contenente gli ultimi ricevuti)
Global Valori() As String
'variabili flags di stato per la gestione di errori di comunicazione e di pacchetto
Global ErrPackNumber As Boolean        'numero telegrammi ricevuti non corrispondente


Public Function InPacket(CommCTRL As MSComm)        'gestisce il flusso dati
On Error Resume Next            'in caso di errori proseguo
Dim occorr As Double            'variabile posizione stringa cercata
Buffer = Buffer & "." & CStr(Asc(CommCTRL.Input))       'accumulo dati nel buffer
(separando i bytes con un punto)
'calcolo l'occorrenza della stringa di stop del pacchetto, iniziando la ricerca alla fine
del buffer meno il totale
'di caratteri pari alla lunghezza della stringa da cercare.
occorr = InStr(Len(Buffer) - Len(Pre1) - Len(Post) - 4, Buffer, "." & Post & "." & Pre1 &
".")
If occorr > 0 Then          'se è stata trovata la stringa di stop
    Buffer2 = Left(Buffer, occorr - 1)          'carico nel buffer 2 la parte del buffer
che precede lo stop trovato
    Buffer = Right(Buffer, Len(Pre1) + Len(Pre2) + 1)       'tolgo dal buffer la parte di
dati caricata nel buffer 2
End If
'cerco l'occorrenza della stringa di start all'interno del buffer 2
occorr = InStr(1, Buffer2, Pre1 & "." & Pre2 & ".")
If occorr > 0 Then          'se è stata trovata la stringa di start
    'carico nel buffer 3 il contenuto del buffer successivo alla stringa di start, pari
    'al pacchetto dati (senza start e stop)
    Buffer3 = Right(Buffer2, Len(Buffer2) - occorr - (Len(Pre1) + Len(Pre2) + 1))
    Buffer2 = ""                'azzero il buffer 2
    Valori = Split(Buffer3, ".")        'divido i vari pacchetti (tolgo i punti) e li
metto in una matrice
    Buffer3 = ""                'azzero il buffer 3
    If UBound(Valori) <> (PackNumber - 1) Then       'se il pacchetto non contiene tutti
i telegrammi
        ErrPackNumber = True                        'attivo il flag
    Else                            'altrimenti
        ErrPackNumber = False       'azzero il flag errore numero telegrammi
    End If
End If
End Function
```

MODULO CONVERSIONE BINARIA (BINARY)

```
Public Function dectobin(decnumber As Long) As String          'converte da decimale a
binario (stringa)
On Error Resume Next                'in caso di errori inprecisati continuo l'esecuzione
Dim rest As Byte                    'resto della divisione
Dim numbertoconv As Long            'numero da convertire
numbertoconv = decnumber            'carico il numero da convertire
If numbertoconv <> 0 Then              'se il valore da convertire è diverso da 0
    While numbertoconv <> 1               'finchè non rimane 1 dalle divisioni
        rest = numbertoconv Mod 2             'salvo il resto della divisione
        numbertoconv = Fix(numbertoconv / 2)    'divido in due e salvo il risultato con
approsimazione per eccesso
        dectobin = rest & dectobin            'aggiorno il risultato
    Wend
End If
dectobin = numbertoconv & dectobin           'scrivo l'lsb del byte
While Len(dectobin) <> 8                    'finchè la lunghezza della stringa non è 8
digit
    dectobin = "0" & dectobin                   'aggiungo uno zero dalla parte dell'MSB
Wend
End Function



Public Function bintodec(binnumber As String) As Long       'converte da binario
(stringa) a decimale (long)
On Error Resume Next                'in caso di errori inprecisati continuo l'esecuzione
Dim contdigit As Byte               'contatore cifre
Dim numbertoconv As String          'numero da convertire
numbertoconv = binnumber            'carico il numero da convertire
If Len(numbertoconv) > 0 Then       'se la lunghezza della stringa binaria non è zero
    For contdigit = 0 To Len(numbertoconv) - 1              'per il numero di caratteri
della stringa (con indice 0)
        'sommo al risultato la potenza 2^n, dove n è il peso del bit (determinato dal
contatore caratteri), moltiplicata per la variabile binaria
        'del digit corrispondente al peso puntato.
        bintodec = bintodec + (2 ^ contdigit) * CByte(Left(Right(numbertoconv, contdigit
+ 1), 1))
    Next contdigit
Else                                'se la lunghezza è 0
    bintodec = 0                    'il risultato è zero
End If
End Function


Public Function binsetbit(binnumber As String, bitnumber As Byte, bitvalue As Byte) As
String       'imposta un singolo bit in una stringa binaria
On Error Resume Next                'in caso di errori inprecisati continuo l'esecuzione
If binnumber = "" Then              'se la stringa binaria è nulla
    binnumber = "00000000"              'creo una stringa binaria byte con valore 0
End If
If bitnumber > Len(binnumber) - 1 Then Exit Function        'se il digit corrispondente
alll'indice del bit supera la lunghezza della stringa binaria, esco senz modifiche
binsetbit = Left(binnumber, Len(binnumber) - bitnumber - 1) & CStr(bitvalue) &
Right(binnumber, bitnumber)          'modifico il bit puntato dall'indice
End Function


Public Function binbitcont(binnumber As String) As Byte             'restituisce il
numero di bit che compongono una stringa binaria
On Error Resume Next                'in caso di errori inprecisati continuo l'esecuzione
binbitcont = Len(binnumber)                 'numero di bit che compongono la stringa
End Function
```

MODULO BUSSOLA

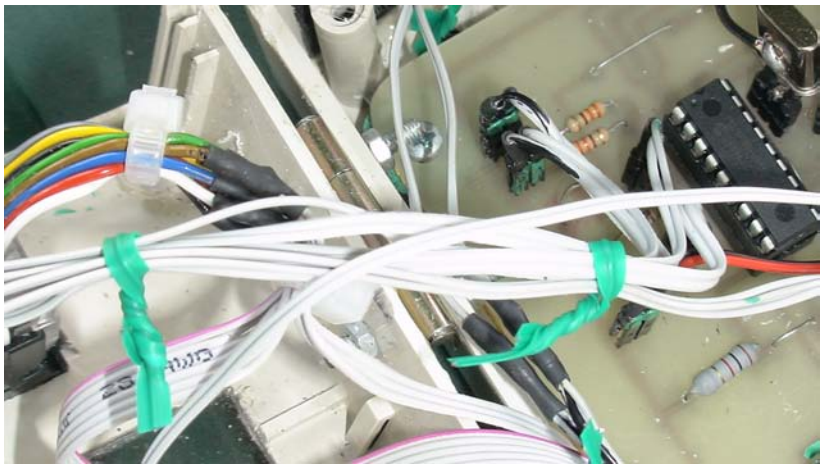
```
Public k1 As Double, k2 As Double, k3 As Double         'variabili per il calcolo della
posizione della lancetta
Const pi = 3.141592654                                  'valore della costante pi greco


Public Function Bussola(degrees As Integer, lunghezza As Double, tline As Line)
'calcola  la posizione della lancetta
'Vengono passati come parametri i gradi (partendo da nord in senso orario) della
posizione dell'ago della busola
'la lunghezza dell'ago e la linea che viene utilizzata come ago. Il punto centrale
dell'ago è il punto superiore
'della linea usata, messa in posizione verticale (punta verso sud)
On Error Resume Next                    'in caso di errori proseguo
'uso le funzioni seno e coseno per fissare la posizione della punta dell'ago a seconda
dei gradi inseriti (che sono poi
'convertiti in radianti) e della lunghezza dell'ago.
k3 = -2 * pi / 360 * (degrees + 180)
k1 = Sin(k3) * lunghezza
k2 = Cos(k3) * lunghezza
'fisso le posizioni dell'ago
tline.X1 = tline.X2 + k1
tline.Y1 = tline.Y2 + k2
End Function
```

FASE 3

MESSA IN SERVIZIO

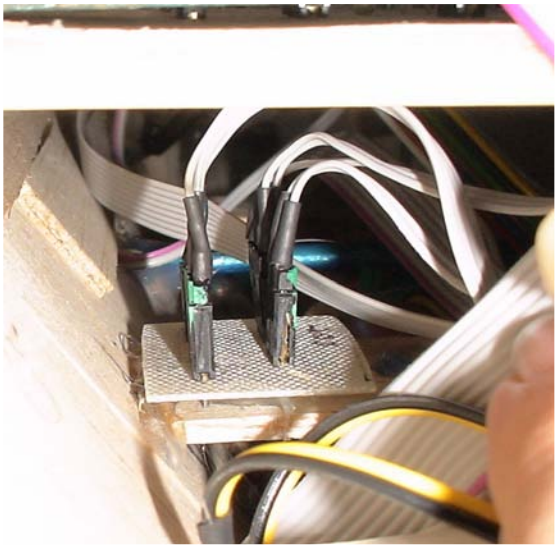
PREPARAZIONE AL PRIMO TEST SUL CAMPO

Ora è arrivato il momento di assemblare tutto, sia l'elettronica che la struttura.
Come prima cosa sono stati realizzati tutti i cavetti per interconnettere i dispositivi. La realizzazione dei cavetti è stata fatta utilizzando del cavo flat e dei normalissimo connettori strip da circuito stampato. Ogni singolo cavo è stato spellato e stagnato al rispettivo contatto sul connettore, usando della guaina termorestringente per nascondere ed isolare le saldature.
La maggior parte dei cavetti presentano ad entrambe le estremità dei contatti di tipo maschio, mentre le femmine sono sui circuiti stampati.
Sono stati eseguiti i vari collegamenti della base. L'elevato numero di cavi ha implicato un'adeguata operazione di cablaggio per organizzare al meglio lo spazio. Come si può notare i cavi sono più ordinati.



Anche sulla gondola vi sono numerosi cavi, che vengono fatti passare sotto la schedina e fuoriescono attraverso i buchi sui lati della cabina.
Un unico cavo collega i connettori dei sensori fuffy e degli infrarossi alla scheda di derivazione principale, situata al centro della struttura. Il cavo passa attraverso un foro sulla sommità della cupola.
Dalla derivazione principale si estendono i cavi che collegano quest'ultima alle derivazioni periferiche, a cui sono direttamente connessi i sensori fuffy (servo + ultrasuono). Da queste derivazioni partono anche i cavi diretti ai sensori infrarossi situati sopra il pallone.
Un'altra derivazione è invece all'interno della gondola, sotto la scheda principale: il suo compito è quello di unire l'alimentazione ed i canali dati

del sensore ultrasonico per l'atterraggio. In uscita abbiamo quindi un unico cavo che si collega a quest'ultimo, portando sia l'alimentazione che i dati.

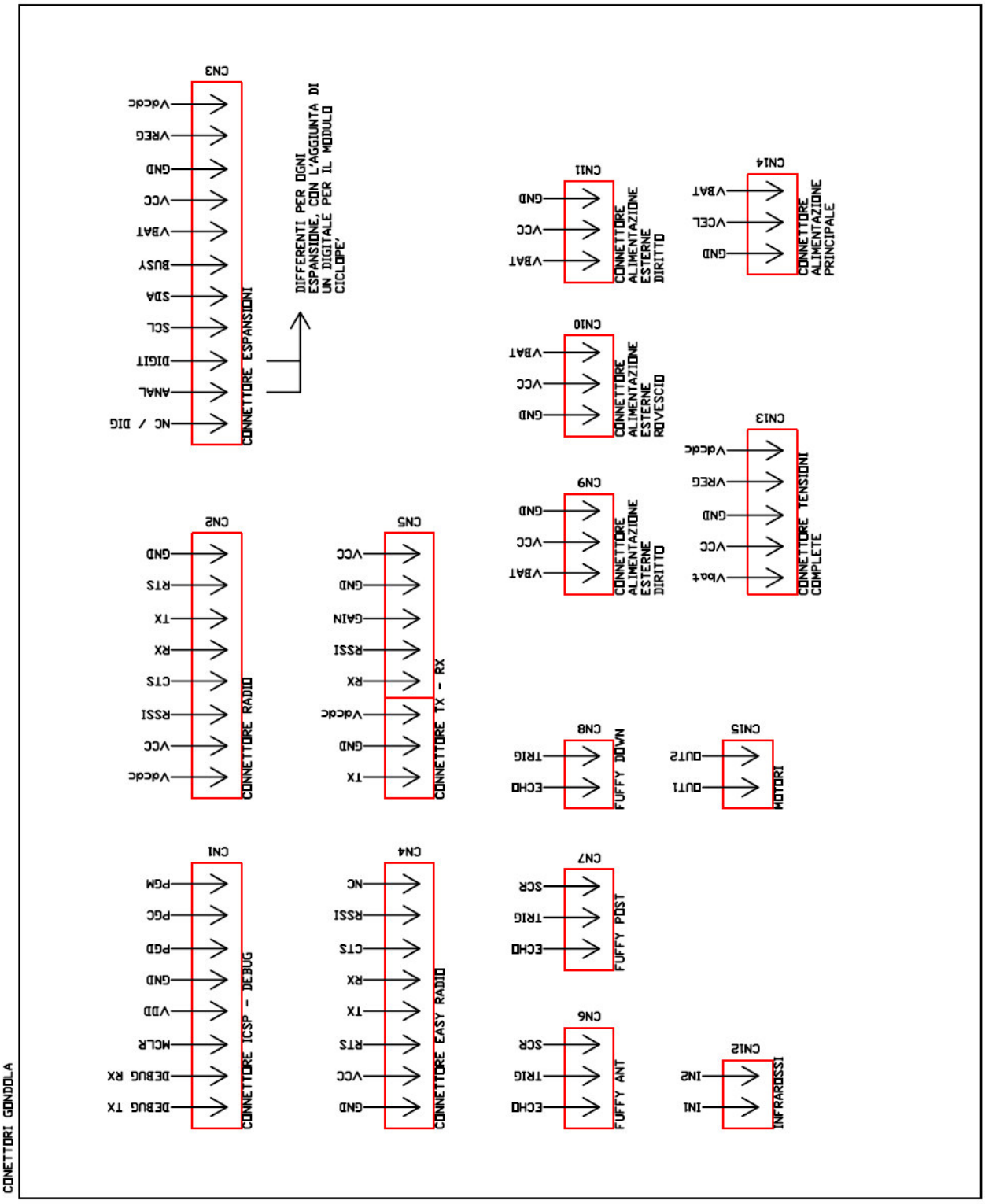


Nella punta della gondola è stata fissata la schedina di supporto per il modulo easy radio. Le eventuali schedine per i moduli ibridi verranno fissate una per lato (sempre sulla punta).

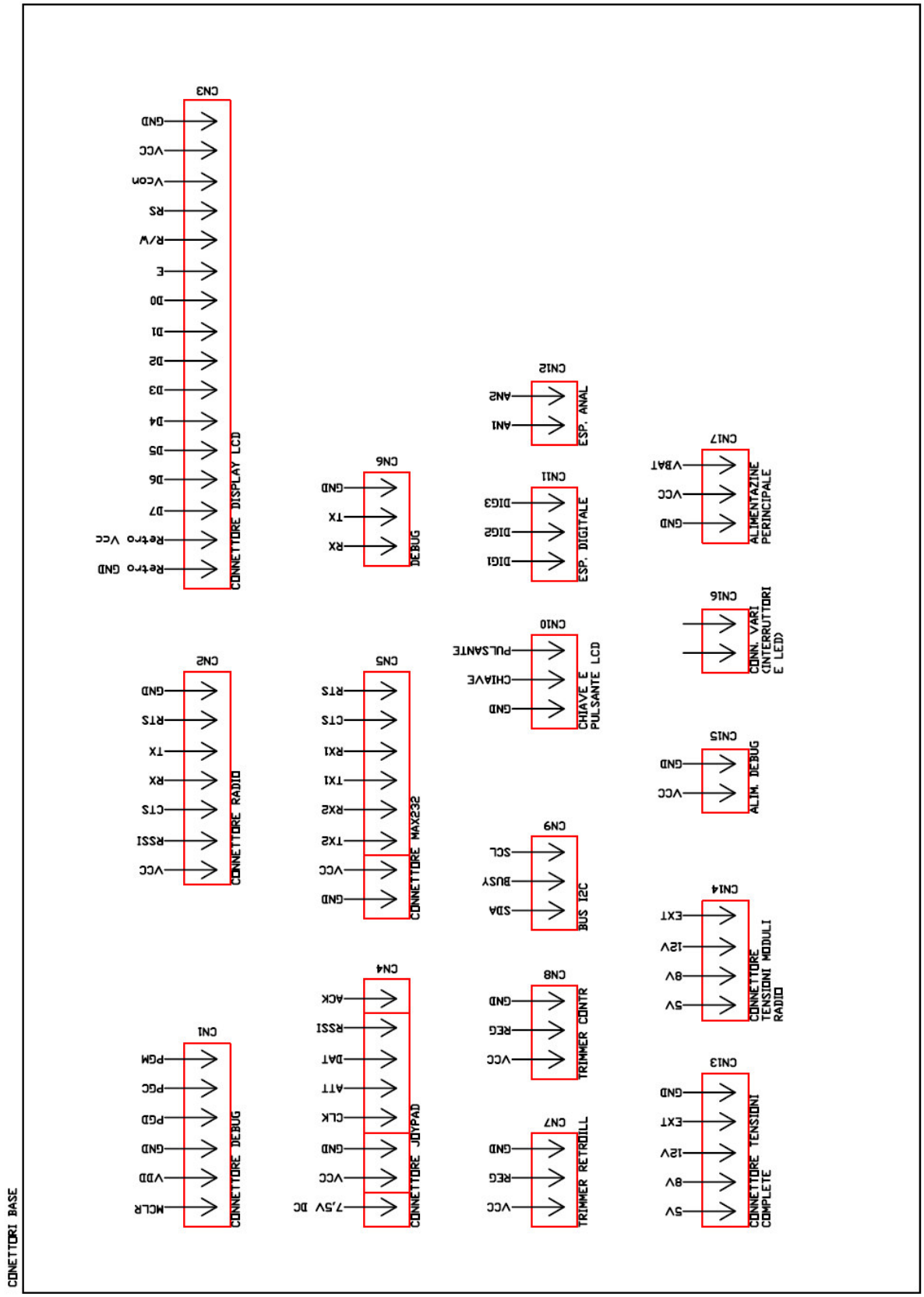
Tutti i collegamenti fatti devono rispettare la polarità dei connettori, percui è stata realizzata una mappatura completa dei tali, realizzando uno schema di guida ai connettori.

SCHEMI CONNETTORI
E CONFIGURAZIONE SCHEDINE

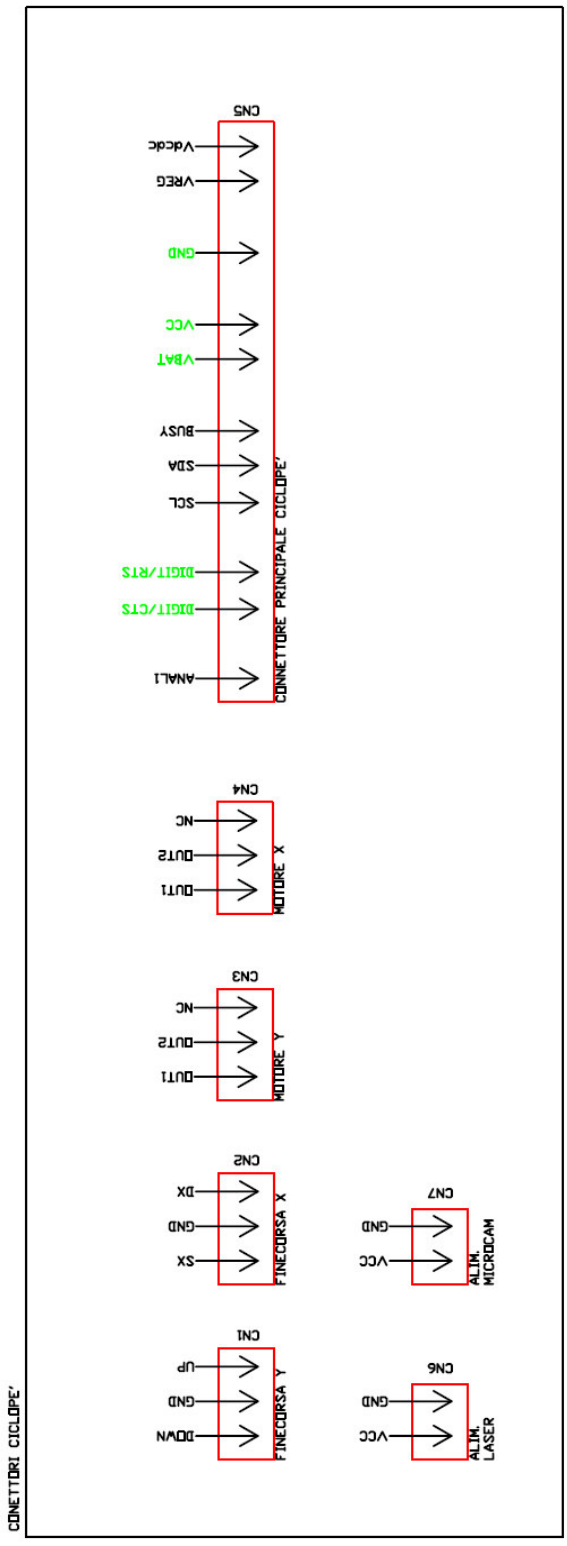
CONNETTORI SCHEDA GONDOLA

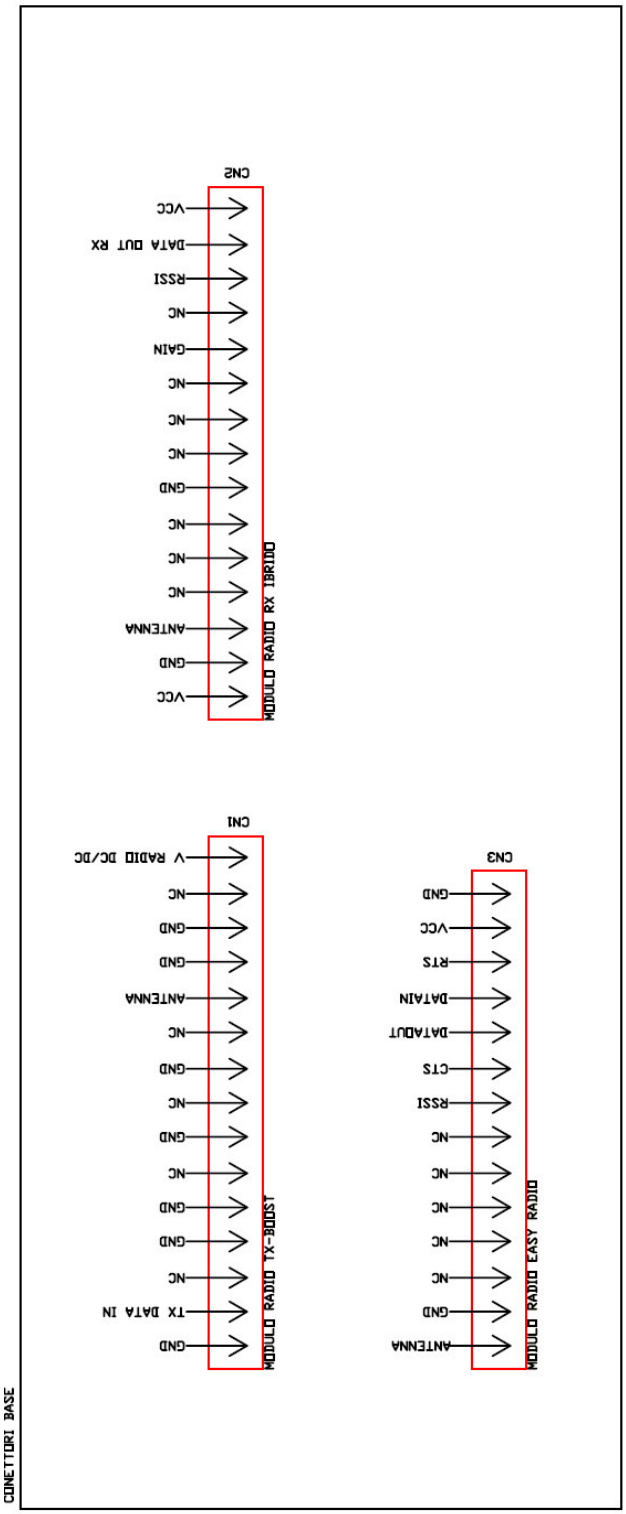


CONNETTORI SCHEDE BASE

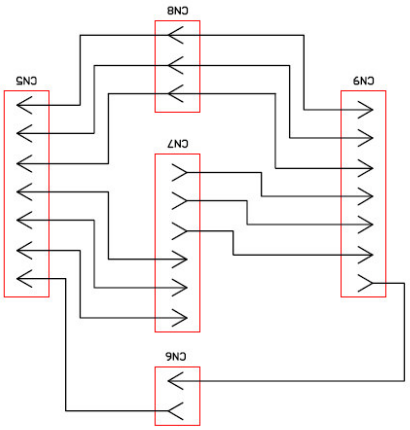


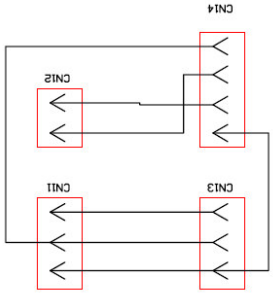
CONNETTORI MODUO CICLOPÈ
E MODULI RADIO

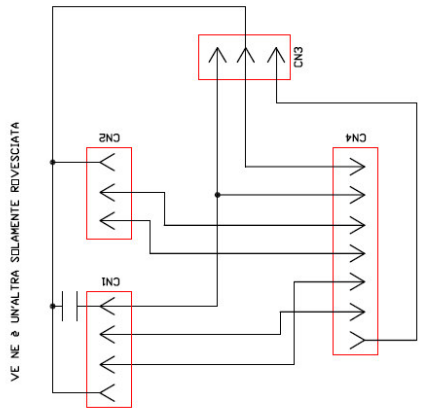




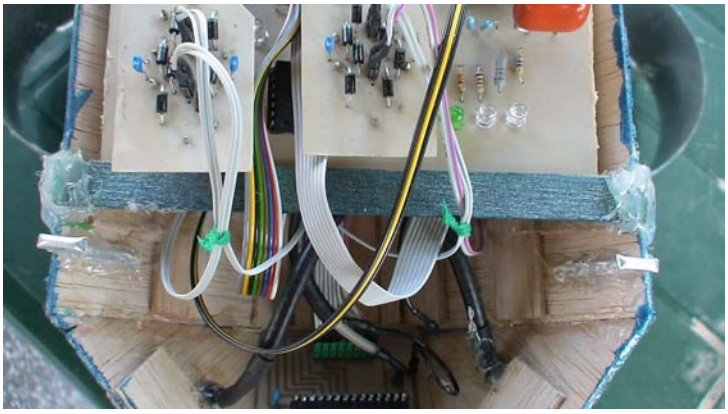
CONNETTORI ESPANSIONI





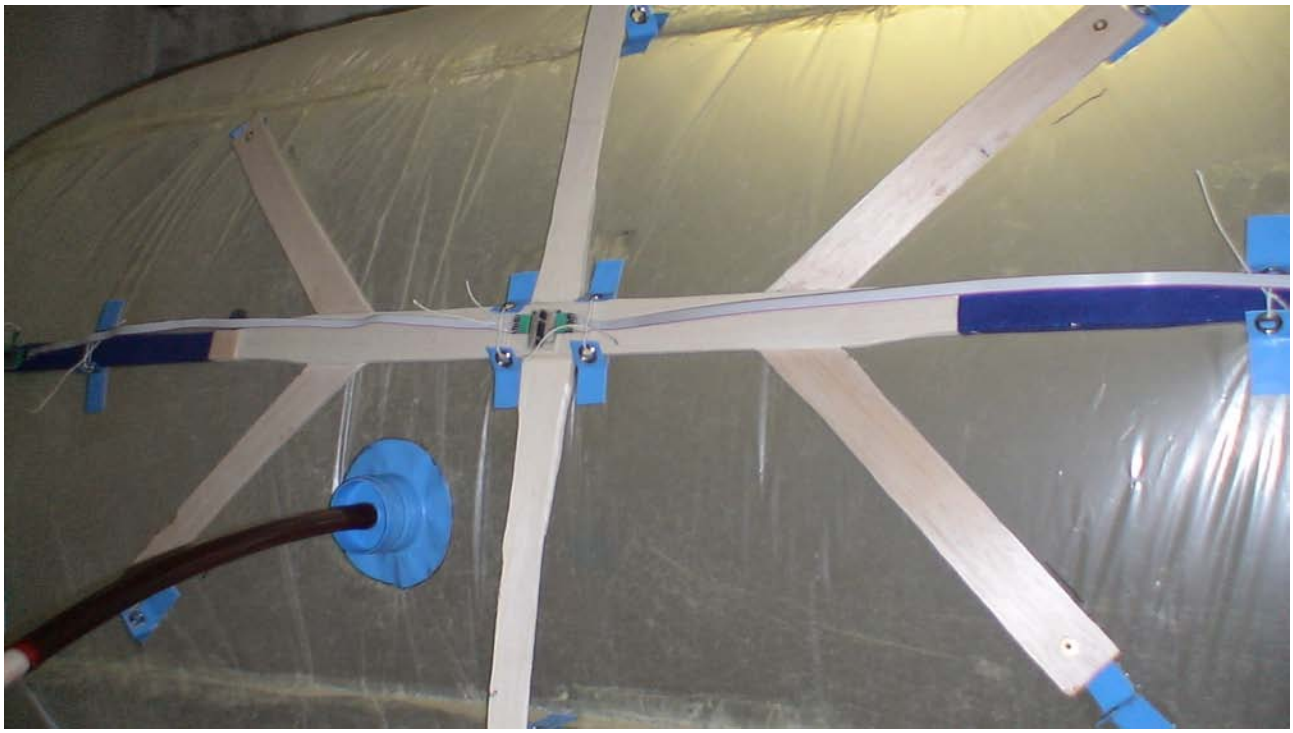


La scheda principale del dirigibile è rimovibile: infatti si innesta ad incastro nella gondola e viene fissata attraverso un passante di legno.



Le schede della base invece sono tutte fissate per mezzo di viti e distanziate dal fondo della scatola e dalle pareti attraverso degli spessori in PVC espanso.
Il computer viene collegato alla porta COM1 della base. Tale porte gestisce anche l'Handsacking, ed è l'unica che può comunicare con l'interfaccia grafica software.
La seconda seriale può essere connessa ad un terminale ausiliario oppure ad un PLC.
L'espansione consente di espandere le potenzialità della base in futuro ed i connettori jack femmina allineati sono dedicati al debug in fase di sviluppo del software dei pic principali (uno è connesso all'RJ45).
Sono presenti anche altre due seriali: sono i programmatori per i PIC, uno collegato alla PicMaster della Base ed uno per il pic Core del velivolo (attraverso l'RJ45).
I restanti connettori (di cui uno jack femmina) sono destinati all'alimentazione della base.
Le batterie sono contenute dell'alloggiamento posto sul lato sinistro della scatola (2 da 9V).
La base presenta due antenne, anche se in realtà si fa uso di una soltanto. In caso venissero usati i moduli Ibridi sarebbe richiesto l'uso di entrambe. Due antenne tuttavia migliorano anche l'aspetto della consolle.
Il pannello frontale della base presenta gli interruttori per abilitare/disabilitare le varie opzioni: quello in alto per abilitare la COM2,

quelli sulla sinistra per Accendere/Spegnere la Vibrazione e gli “effetti sonori” del Joypad. Ogni interruttore ha il proprio led di segnalazione.
In basso vi sono i led comandati dai pic per segnalare i vari stati del velivolo. Al centro c’è il display, che visualizza qualsiasi tipo di dato inerente alla base ed al velivolo. Attraverso il pulsantino si scorrono le varie schermate, mentre con i potenziometri si aggiustano retroilluminazione e contrasto.
In basso a sinistra c’è il connettore RJ45 che permette di connettere la scheda del dirigibile alla base, in modo da poter usufruire dei connettore di debug e della programmazione In-Circuit.
Sulla destra, in alto c’è la chiave per l’abilitazione del sistema (chiave accensione) e sotto i led di alimentazione.
Sul lato sinistro della base esce il cavo con il connettore di collegamento del Gamepad della PlayStation, sul lato destro vi è l’interruttore generale (alimentazione).
Il dirigibile è pilotabile sia con il computer che con il joypad, ma il PC ha una priorità più alta di quest’ultimo in caso siano connessi (e abilitati) entrambi.
Il prossimo passo è il fissaggio della gondola alla struttura ed infine al pallone.
Il pallone è stato gonfiato ad aria e sono state ricercate eventuali perdite.
Successivamente sono stati applicati dei ganci realizzati con delle strisce di PVC (tessuto) forate ad una estremità. Il foro è stato rinforzato con degli anelli metallici ed infine la striscia è stata incollata sul pallone (mylar) per mezzo di un collante apposito, lo stesso usato per la costruzione del siluro.
I ganci sostengono la struttura, che viene fissata al pallone con dello spago.
Altri ganci servono per il bilanciamento della struttura e per impedirne movimenti oscillatori durante il volo. Tali ganci sono visibili in prossimità dei rami del supporto.

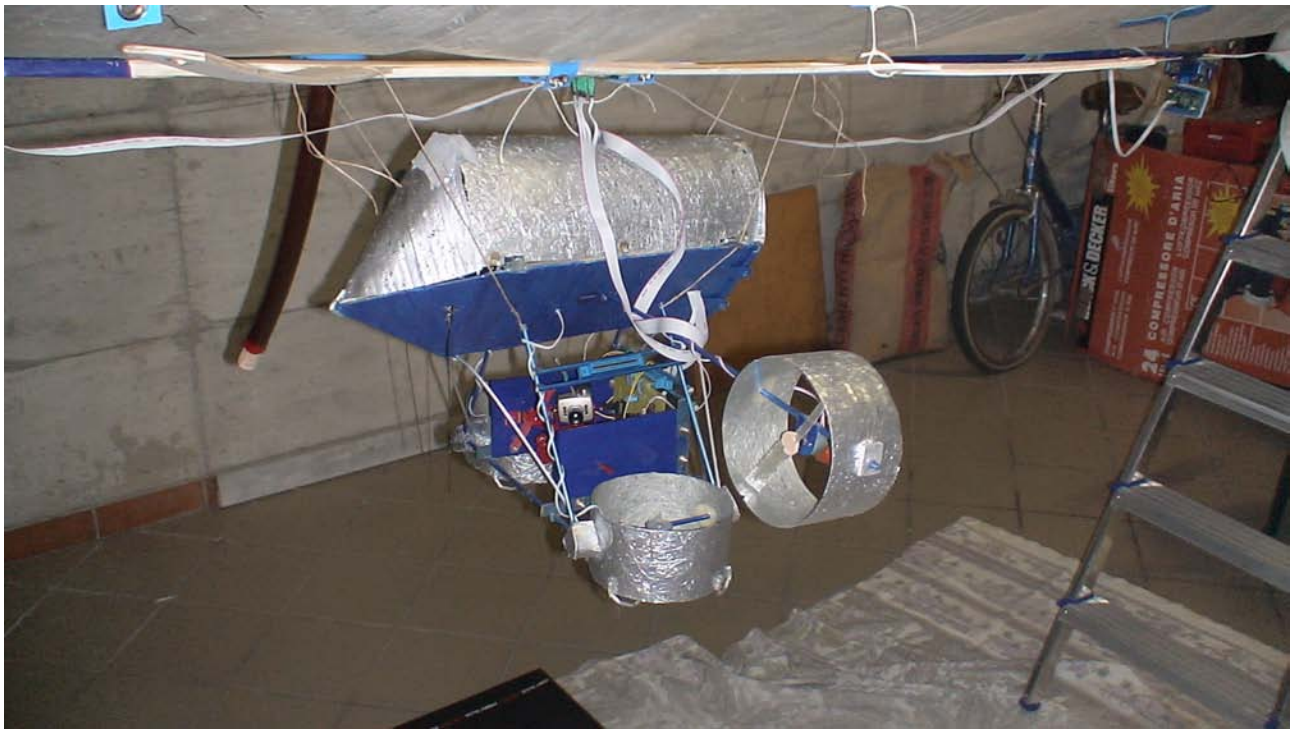


Ganci realizzati nello stesso modo vengono usati anche per il fissaggio di loghi pubblicitari: due sono incollati sulla parte superiore del pallone e due sotto la struttura, incollati alla balsa. Per mezzo un filo il logo viene fatto aderire al pallone (il filo viene passato negli anelli dei ganci).
Per fissare i moduli dei sensori infrarossi, che devono guardare verso il soffitto, si usa ancora il PVC. Sono state tagliate due striscioline di materiale ed incollate alle estremità. La strisciolina inferiore è quindi stata incollata al pallone. Tra le due strisce vi è uno spazio attraverso il quale viene passata la schedina degli infrarossi.

La gondola viene poi fissata alla struttura per mezzo di tiranti in corda: i tiranti sono fissati da una parte sulle aste inclinate della struttura dei motori ascensionali, subito sotto la cabina della gondola, è dall'altra al centro della struttura, nella parte più resistente e spessa (quella rivestita con il termoretraibile).

Adesso devono essere connessi i sensori infrarossi alle derivazioni rispettive e la derivazione generale alla motherboard attraverso il cavo principale.

La gondola è fissata al pallone, adesso manca la coda. La coda è composta da tre triangoli di balsa, sui quali sono stati realizzati dei fori, rinforzati poi con anelli metallici. La coda principale, quella che viene messa in verticale ha una base in balsa che ne garantisce la posizione quando il pallone è gonfio. Le pinne laterali invece vengono mantenute orizzontali grazie a dei tiranti che la collegano alla principale. Ecco lo scopo dei fori: sostenere le parti della coda.
I tre triangoli sono fermati al pallone per mezzo di ganci in PVC, identici a quelli usati per la struttura, e spago.
Il dirigibile è pronto a volare.

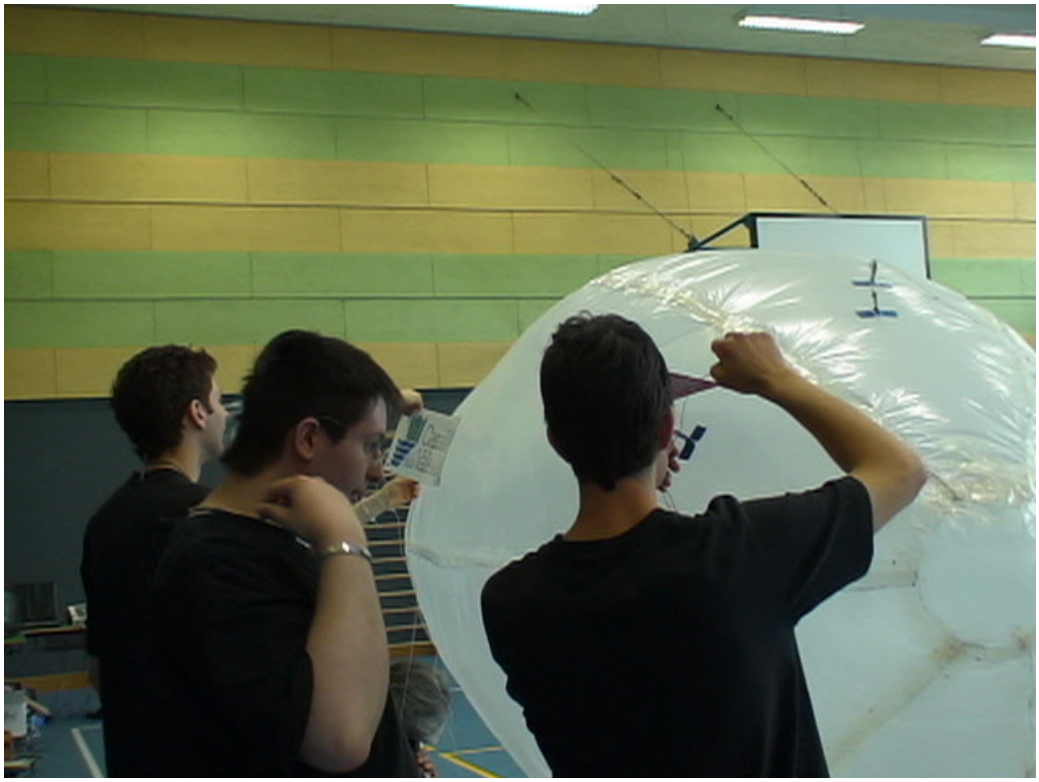

COMPRESSORE D'ARIA

IL VOLO

Il pallone è ovviamente stato sgonfiato per essere poi rigonfiato con il giusto Gas: L'Elio.

Il pallone si è rilevato un po' più grande di quanto calcolato: il volume è maggiore perché durante la costruzione il siluro è risultato più cilindrico del previsto.
È stato abbastanza difficile tenerlo a terra, una volta gonfio, infatti abbiamo dovuto legarlo ad alcune sedie

L’aggancio della struttura, della gondola e della coda sono stati eseguiti al momento perché è stato necessario smontare tutto in precedenza per il trasporto.



Nella gondola sono stati inseriti dei pesi, nell'apposito scomparto, come zavorra.
Ma questo non basta. È necessaria anche una zavorra sulla struttura. Per equilibrare il dirigibile, in modo che non vada ne verso l'alto, ne verso il basso, stando a mezz'aria, sono state usate delle bottigliette d'acqua da ½ litro. Sono state fissate con della corda sulla struttura, una in avanti ed una in dietro rispetto alla gondola.
Per calibrare l'equilibrio del velivolo, man mano che si sgonfiava si beveva un po' d'acqua dalle bottigliette di zavorra, ritornando ad una situazione di stallo.

Il pallone si è stabilizzato a mezz'aria. Attraverso la base terrestre venivano pilotati i motori.
Il dirigibile si muoveva: saliva e scendeva senza problemi, andava avanti, indietro e cambiava rotta.
Ci eravamo riusciti!
Sul televisore si vedevano le immagini riprese dalla telecamera. Da terra, attraverso il JoyPad oppure il PC, venivano trasmessi i dati.
La telecamera girava, inquadrando le persone, che noi potevamo vedere sullo schermo.

Anche il laser funzionava alla grande: quando veniva acceso si poteva chiaramente vedere il bollino rosso sulla testa della persone sotto, al centro della visuale della telecamera.
Tutto ha funzionato come previsto in fase di progetto, regalandoci un magnifico spettacolo.
Dal volo abbiamo ricavato alcune informazioni importantissime per lo sviluppo del sistema di autonavigazione.
Abbiamo potuto constatare la velocità (o meglio la lentezza) del velivolo e l'inerzia. Spegnendo i motori infatti il dirigibile proseguiva per un bel po' sulla stessa rotta prima di fermarsi.
Questo ci permette di calcolare in quanto tempo avviene la frenatura (ordine di grandezza) mandando i motori al massimo della potenza nella direzione opposta.
La durata delle batterie è stata breve, come previsto, anche se maggiore di quanto ipotizzato all'inizio: tutti i motori sono stati fatti andare al massimo per tutto il tempo.
I sensori del dirigibile hanno continuato ad effettuare misurazioni. Sul monitor del PC si potevano vedere i dati relativi: distanze degli oggetti in prossimità del pallone, altezza da terra, direzione (cardinale), presenza del soffitto (infrarossi), tensione e livello batterie…
Da terra si aveva il completo quadro della situazione.
Gli obiettivi principali sono stati raggiunti in pieno!


NEW YORK

Un dirigibile in palestra

E' uno dei lavori realizzati dagli studenti dell'Iti

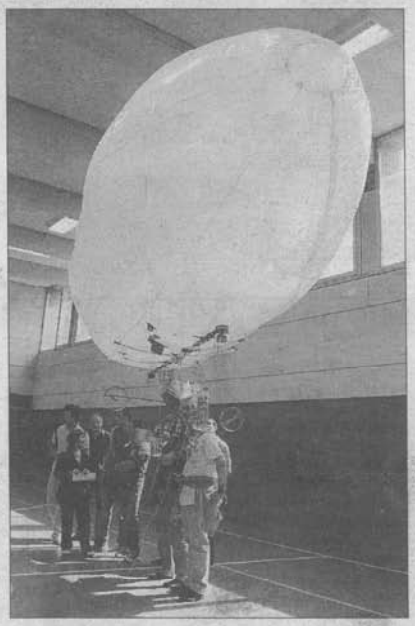
Un dirigibile in volo in palestra. È accaduto ieri presso l'Iti «Buonarroti» di Trento durante la presentazione di alcuni lavori realizzati dagli studenti nell'ambito dell'area di progetto, il settore pluridisciplinare che consente di mettere in pratica le conoscenze acquisite a lezione.

La perfetta riproduzione di un dirigibile rappresenta il risultato più curioso e interessante di tali lavori. Realizzato in un anno da Daniele Rucatti, Omar Bottesi, Michele Conci e Debora Lageder della 5 ECO, il velivolo ha dato prova di perfetta manovrabilità. Il pallone – riempito di elio - è lungo quasi 4 metri e ha un diametro di 1,30 metri; sorregge una gondola con quattro eliche in grado di muovere il dirigibile in ogni direzione; due sensori a ultrasuoni segnalano eventuali ostacoli; a bordo una telecamera orientabile trasmette in diretta le immagini relative al percorso. Le manovre vengono decise da terra con un telecomando radio fino alla distanza di cento metri.

«L'idea – spiega Rucatti – ci è venuta durante una gita. Abbiamo reperito sia le informazioni per il progetto sia parte del materiale in Internet. Abbiamo inoltre utilizzato materiali di scarto. Un aiuto finanziario ci è stato garantito dalla Cassa Rurale della Valle dei Laghi». Il dirigibile potrebbe essere usato per riprese video dall'alto, come sonda meteo o mezzo pubblicitario. L'intero progetto sarà disponibile in rete. Informazioni possono già essere ottenute scrivendo un'e-mail a ruky87@gmail.com.

A settembre il dirigibile sarà esposto al museo Zeppelin di Friedrichshafen grazie all'interessamento dell'istituto di elettronica di Tettnang con cui l'Iti ha avviato una collaborazione. Klaus Hesse, dirigente della scuola tedesca, era presente al varo dell'aeromobile che ha seguito con notevole attenzione.

Fra gli altri progetti in mostra ieri, vanno segnalati un ponte mobile e un Un ascensore in miniatura. Andrea Nardin, Tomas Pradella e Stefano Zurlo della 5 ELC hanno realizzato un modello di ponte levatoio che si alza e si abbassa automaticamente all'arrivo di un'imbarcazione.



IN VOLO. Il dirigibile (foto Dandrea)

Thomas Grisenti, Francesco Bortolotti, Luca Lorenzetti, Gianluigi Rosa e Daniele Toller sono invece gli studenti che hanno costruito un ascensore in minitura perfettamente in grado di funzionare.

Il progetto più significativo è quello che ha visto protagonisti Alessandro Casna, Andrea Malfatti e Matteo Pompermaier guidati dal professor Maurizio Casagranda di «Ingegneri senza frontiere». A scuola i tre studenti hanno realizzato un gruppo di continuità che garantirà l'erogazione senza interruzione di energia elettrica a un laboratorio per l'analisi delle acque reflue a Mahajanga, la seconda città del Madagascar. In luglio due allievi dell'Iti (Tomas Pradella e Michele Conci) si recheranno nel paese africano per l'installazione e il collaudo del prezioso apparecchio.

Il dirigibile, il ponte mobile, l'ascensore e il gruppo di continuità costituiscono quattro esempi delle ottime conoscenze acquisite dagli studenti del «Buonarroti» durante il loro percorso di studi.

P. B.

Il progetto è costato quasi un anno di lavoro. Ma alla fine quanta soddisfazione

Un dirigibile all'Iti Buonarroti

Quattro studenti hanno creato un «mini Zeppelin»

di Nicola Baldo

TRENTO. Un intero anno scolastico di lavoro, di giorno e di notte chiusi nella propria cameretta, ma ieri il "suppostone" ha potuto svolgere il suo viaggio inaugurale. Un vero dirigibile, lungo quattro metri, realizzato da quattro studenti dell'Iti Buonarroti si è finalmente librato in volo. Un primo viaggio fra le mura della palestra di via Brigata Acqui per il progetto portato avanti da Daniele Rucatti, Omar Bottesi, Michele Conci e Debora Lageder, tutti alunni della quinta elettro C. Un lavoraccio il loro, portato avanti praticamente da soli eccezion fatta per un contributo economico della Cassa Rurale della valle dei Laghi, che ha permesso di creare da zero questo vero zeppelin in miniatura.

Dotato di quattro motori, telecamera con puntamento laser, batterie al litio e radiocomandato a terra grazie ad un joystick della Playstation riadattato per l'occasione, il dirigibile è una vera chicca della tecnica. La telecamera permette al dirigibile di trasmettere ad uno schermo a terra tutto ciò che si vede da lassù, non fra le nuvole ovviamente ma comunque ad altitudini difficilmente raggiungibili da noi esseri umani. Adesso a disposizione di tutti, scuole ed enti, che vogliano utilizzarlo per determinati scopi. Ma cosa ha spinto quattro ragazzi che in testa dovrebbero avere solo la maturità a sacrificare serate e weekend per allestire questo mini zeppelin? La voglia di cimentarsi con un progetto ambizioso e allo stesso tempo stimolante, che permetterà loro di andare ospiti dal 12 al 16 settembre al Museo dello Zeppelin in Germania e dell'industria che ancora costruisce questi signori del cielo. Un'idea nata quasi per caso circa un anno fa, al ritorno dopo un viaggio d'istruzione all'istituto di Bodensee, nei pressi del lago di Costanza. Da allora tanto, tanto lavoro sino all'inaugurazione di ieri. Con il naso all'insù di tanti studenti dell'Iti per guardare questo dirigibile tutto "made in Buonarroti".

Gli studenti dell'Iti Buonarroti con il dirigibile (foto Panato)

Ma ben presto un altro importante progetto coinvolgerà alcuni studenti dell'Iti. Il 23 luglio, infatti, Michele Conci e Thomas Pradella partiranno per Mahajanga, in Madagascar, per un progetto di solidarietà. Insieme al loro insegnante Maurizio Casagranda, grazie ad un progetto di Ingegneria Senza Frontiere, realizzerano un gruppo di continuità, misura e protezione elettrica che permetta di caricare le batterie indispensabili per avere l'elettricità anche in questo, sfortunato, angolo di Africa. In una zona dove l'elettricità salta (troppo) spesso e la gente vive in condizioni di vita disperate.c Dopo la maturità, per entrambi, un mese di lavoro in Madagascar che proseguire su un percorso di aiuto e sostegno con l'Africa che è iniziato ormai nel lontano 2001.

I prossimi obiettivi da realizzare sono l'implementazione del sistema di navigazione automatica AUTONAV.
Il sistema permetterebbe al velivolo di destreggiarsi autonomamente nello spazio, evitando oggetti, percorrendo determinate rotte e trovando la strada di casa. Insomma il sottoprogramma incrementerebbe notevolmente l'autonomia dell'aeronave.
I sensori sono stati messi per un preciso scopo, che era appunto quello di poter creare, in futuro, il sistema AUTONAV.
Infatti il dirigibile può appoggiarsi ai sensori per rilevare ostacoli, altitudine, direzione ed agire in merito.
Inoltre sa lo stato delle batterie ed il livello di segnale radio, con cui può tracciare la ritta di casa.
L'espandibilità introdotta grazie alle 4 espansioni, 3 disponibili di cui una interna, permettono l'implementazione di numerosi sensori esterni o di altre schedine dedicate a particolari funzioni.
Per poter iniziare la creazione dell'AUTONAV, dovevamo prima però sapere la velocità di spostamento del velivolo e la sua inerzia oltre a verificare il completo funzionamento di tutti i dispositivi.
Grazie al volo inaugurale questo è stato possibile, quindi le basi per creare un robot indipendente sono già state date.
I software sono già predisposti per l'aggiornamento, infatti la struttura di questi è stata organizzata per integrare un giorno il sottosistema autonav.
L'unica cosa da fare è creare l'algoritmo di navigazione e poi testarlo con ulteriori voli.
Questo progetto non è "concluso" ma può essere ampliato con nuove funzionalità. Le stesse schedine elettroniche possono essere usate per altri scopi. Nulla vieta di usare la MainBoard del dirigibile come scheda di controllo di un Rover esploratore su ruote.

CONCLUSIONI

Siamo arrivati felicemente alla conclusione di questa saga che ci ha coinvolto per ben un anno e mezzo. Sembrerà che il percorso è stato facile, leggendo queste pagine…
In realtà la realizzazione di questo progetto è stata molto più complicata di quello che è all'apparenza.
Molti week-end trascorsi a lavorare. Sere trascorse davanti ai computer fino a tardi. Intere giornate a fare ipotesi e a risolvere problemi.
Sempre un quaderno sotto mano per annotare intuizioni ed idee.
Questo progetto non è stato un passatempo. Per niente. Certo, ci siamo anche divertiti nel costruire il dirigibile!
Comunque questo lavoro è un modo per dimostrare a tutti il nostro valore. Siamo partiti dal Nulla, e da soli siamo riusciti a costruire qualcosa che funziona.
Qualcosa che non solo si muove, ma interagisce con noi!
Dal Nulla.
Mettendo in pratica i frutti delle numerose ricerche, concretizzando tutte quelle teorie, tutte quelle idee che ci balenavano nella testa.
La collaborazione e la tenacia ci hanno permesso di giungere ad una conclusione con ottimi risultati.
Grazie a questo progetto siamo riusciti ad aumentare le nostre conoscenze, non solo nelle discipline appartenenti al nostro corso di studi. Ma la cosa più importante è che abbiamo imparato è a lavorare assieme, per rendere reale ciò che è sulla carta.
Tutto il nostro lavoro è disponibile a tutti, deve essere patrimonio del Mondo, per quanto poco possa valere. È questo il suo valore. Non sono i componenti che lo costituiscono. Perché il vero traguardo che abbiamo raggiunto non è stato dimostrare a tutti che il nostro progetto Funziona, ma il fatto di sapere più di prima. Di sapere cose che altri non sanno.
Prima di terminare questo scritto dobbiamo però fare un ringraziamento a tutte le persone che ci hanno sostenuto, senza le quali forse non saremo giunti ad una conclusione.
Primi fra tutti i nostri genitori, sempre disponibili a dare una mano ed a farci il pranzo!
Grazie alla loro pazienza, costante sebbene la casa sia perennemente sotto-sopra, e grazie al loro sostegno, senza il quale non avemmo potuto superare le difficoltà.

In seconda posizione la Grande Prof. Luciani detta “Tony”, che ci ha seguito in ogni passo della realizzazione del progetto, ci ha sostenuti, e, si deve pur dirlo, ha fatto in modo che venissimo pubblicati sul giornale!
Un grazie anche a tutti gli altri prof che ci hanno aiutato e sostenuto, consapevoli fin dall’inizio che il nostro sarebbe un grande progetto.
Grazie alla Cassa Rurale della Valle dei Laghi che ci ha finanziati economicamente, pagando più dei ¾ dei componenti, mantenendo sempre la fiducia in noi e nel nostro lavoro.
Grazie a tutte quelle persone che ci hanno sostenuto, a scuola e fuori dalla scuola.
Grazie a tutti quelli che ci hanno fatto i complimenti con sincerità.
Grazie, infine, a noi, membri del gruppo e collaboratori, che siamo riusciti a sopportarci, a lavorare senza “andar fuor di testa”. Nonostante le incomprensioni e le divergenze, le idee diverse, siamo riusciti a trovare un punto medio, una via comune.
Ci siamo sostenuti a vicenda, e siamo riusciti a portare a termine qualcosa di grandioso. Adesso possiamo tirare un sospiro di sollievo, rilassarci e pensare ai bei momenti passati.
Grazie a tutti.

RUCATTI DANIELE
BOTTESI OMAR

Per qualsiasi informazione, per gli schemi, le guide, i programmi…
Visitate il sito www.rozzate.com
Contatto e-mail ruky87@gmail.com (Daniele)
Il progetto è completamente OpenSource!

ALLEGATI