**Versione N° 1.1**

# *LaurTec*

# Robot Domotino

**Autore :** *Mauro Laurenti*

**email:** info.laurtec@gmail.com

**ID:** PJ2003-IT

INFORMATIVA

Come prescritto dall'art. 1, comma 1, della legge 21 maggio 2004 n.128, l'autore avvisa di aver assolto, per la seguente opera dell'ingegno, a tutti gli obblighi della legge 22 Aprile del 1941 n. 633, sulla tutela del diritto d'autore.



AVVERTENZE

I progetti presentati non hanno la certificazione CE, quindi non possono essere utilizzati per scopi commerciali nella Comunità Economica Europea.

Chiunque decida di far uso delle nozioni riportate nella seguente opera o decida di realizzare i circuiti proposti, è tenuto pertanto a prestare la massima attenzione in osservanza alle normative in vigore sulla sicurezza.

L'autore declina ogni responsabilità per eventuali danni causati a persone, animali o cose derivante dall'utilizzo diretto o indiretto del materiale, dei dispositivi o del software presentati nella seguente opera.

Si fa inoltre presente che quanto riportato viene fornito cosi com'è, a solo scopo didattico e formativo, senza garanzia alcuna della sua correttezza.

L'autore ringrazia anticipatamente per la segnalazione di ogni errore.

## Introduzione

Domotino è un robot per applicazioni domestiche pronto per l'uso ma al tempo stesso concepito per poter essere modificato senza richiedere un'eccessiva conoscenza dell'elettronica. Il Robot si presenta come una piattaforma sperimentale che può essere facilmente integrata nelle vostre applicazioni e prendere la forma richiesta dalle vostre esigenze...

## Analisi del progetto

Partiamo dalla fine...!
In Figura 1 è quello che si otterrà a fine lavoro...Domotino!

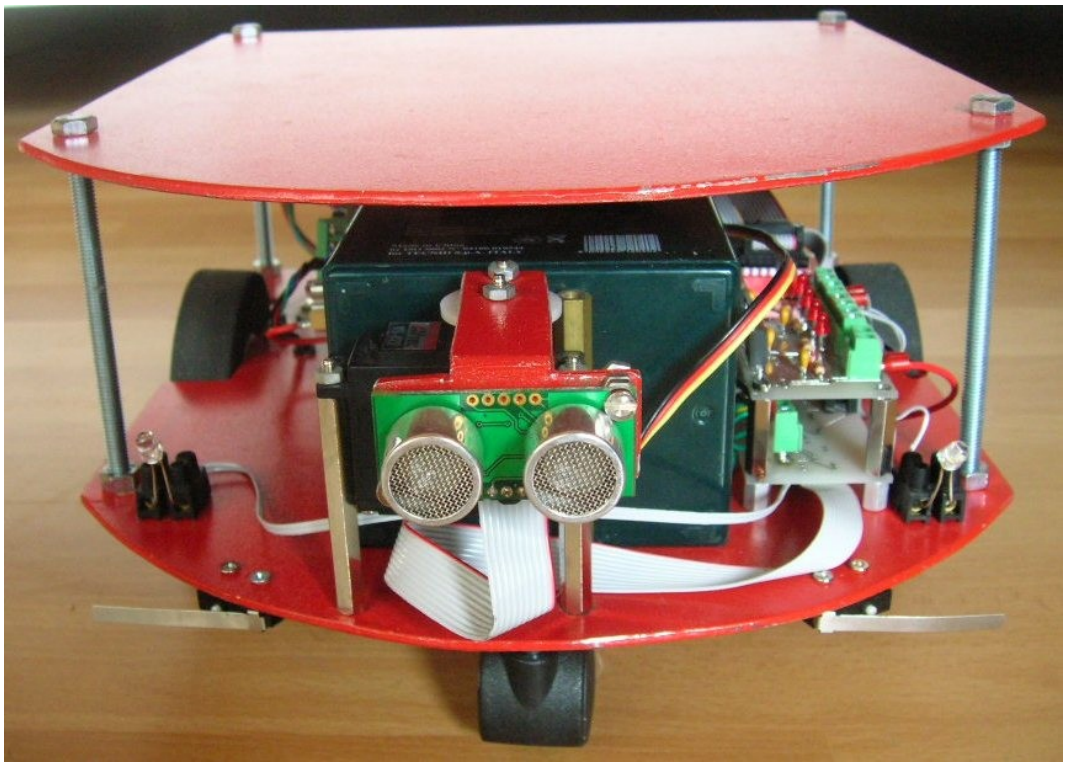


**Figura 1: Domotino**

Il nome Domotino discende dall'unione della parola domus e robottino, da cui l'applicazione principale del robot è gironzolare per la casa come un pipistrello...sfruttando la vista ad ultrasuoni. Il suo funzionamento è molto semplice dal momento che possiede un solo interruttore con il quale accendere e spegnere il robot stesso e un pulsante con cui modificare le sue modalità di funzionamento. Le modalità disponibili sono le seguenti:

- Visualizzazione parametri principali del sistema
- Explorer...ovvero scopa intelligente!
- Cane da guardia
- Luce di cortesia ON/OFF
- Luce di cortesia a tempo
- Controllo Remoto
- Visualizzazione parametri sveglia

L'utente, come verrà di seguito spiegato può facilmente aggiungere altre modalità senza un eccessivo sforzo d'integrazione. Le varie modalità vengono eseguite senza far ricorso all'utilizzo del multitasking in maniera da poter semplificare la modifica del programma da parte delle persone più inesperte.

Per semplificare la modifica del sorgente sono presenti delle variabili globali e funzioni speciali per mezzo delle quali è possibile conoscere e modificare lo stato di Domotino[1].

Le potenzialità di Domotino risiedono nel numero di periferiche che la scheda Freedom, che ne rappresenta il suo cervello, mette a disposizione. Domotino può essere telecomandato via cavo seriale RS232[2] tramite un computer, per mezzo del quale è anche possibile inviare altri comandi speciali con cui variare le caratteristiche del sistema. La connessione con il computer è ottenibile per mezzo del programma dedicato o per i più esperti per mezzo di un qualunque terminale seriale che permetta di inviare caratteri ASCII...

## Prendiamo il martello in mano

Se avete un martello in mano avete interpretato male il titolo; era solo un modo di dire...quel che vi serve è un piccone!

La descrizione per costruire fisicamente il robot verrà improntata come i documenti utilizzati in produzione, in modo da essere sintetici e chiari ma al tempo stesso confidando nell'esperienza del lettore per evitare la descrizione di come collegare il singolo filo. Si fa presente che i disegni non sono in scala e non possono dunque essere utilizzati come “stampini”.

Dal momento che la lavorazione dell'alluminio può creare superfici taglienti si consiglia di arrotondare tutte le superfici lavorate e ricoprirle con nastro. Per queste stesse ragioni è bene evitare che il robot sia un oggetto di gioco per bambini.

Il robot presenta molte parti verniciate per cui è bene che per diversi giorni non venga utilizzato dentro casa in modo da evitare inalazioni tossiche a causa del manto di vernice utilizzato.

La meccanica di Domotino è molto semplice dal momento che è ottenuta per mezzo di due soli piani in alluminio oppurtunatamente ritagliati. La forma consigliata può essere cambiata, ma al fine di evitare comportamenti diversi da quelli che ho potuto testare è bene non variare la dimensione delle due ruote motrici e la distanza tra loro. Tutte le parti meccaniche che verranno illustrate sono ottenute per mezzo di piani d' alluminio di spessore 3mm in modo da evitare che si possano piegare a causa del peso della batteria e di eventuali altri oggetti che dovessero essere da voi aggiunti. La struttura in alluminio e la potenza stessa di Domotino fa del robot un oggetto con il quale è meglio non giocare in presenza di oggetti delicati come vetreria e o altro mobilio di valore...Il robot può trainare anche una sedia...qualora non la dovesse vedere!

[1] Nelle descrizioni che seguiranno il Robot Domotino verrà anche chiamato più semplicemente robot o sistema.

[2] Si capisce che aggiungendo dei moduli wireless per connessioni seriali è possibile telecomandare Domotino senza fili.

Come prima cosa bisogna realizzare due piani in alluminio oppurtunatamente ritagliati come in Figura 2. Il raggio di curvatura della parte anteriore non è importante, se non avete un compasso un tegame andrà più che bene per tracciare l'arco.

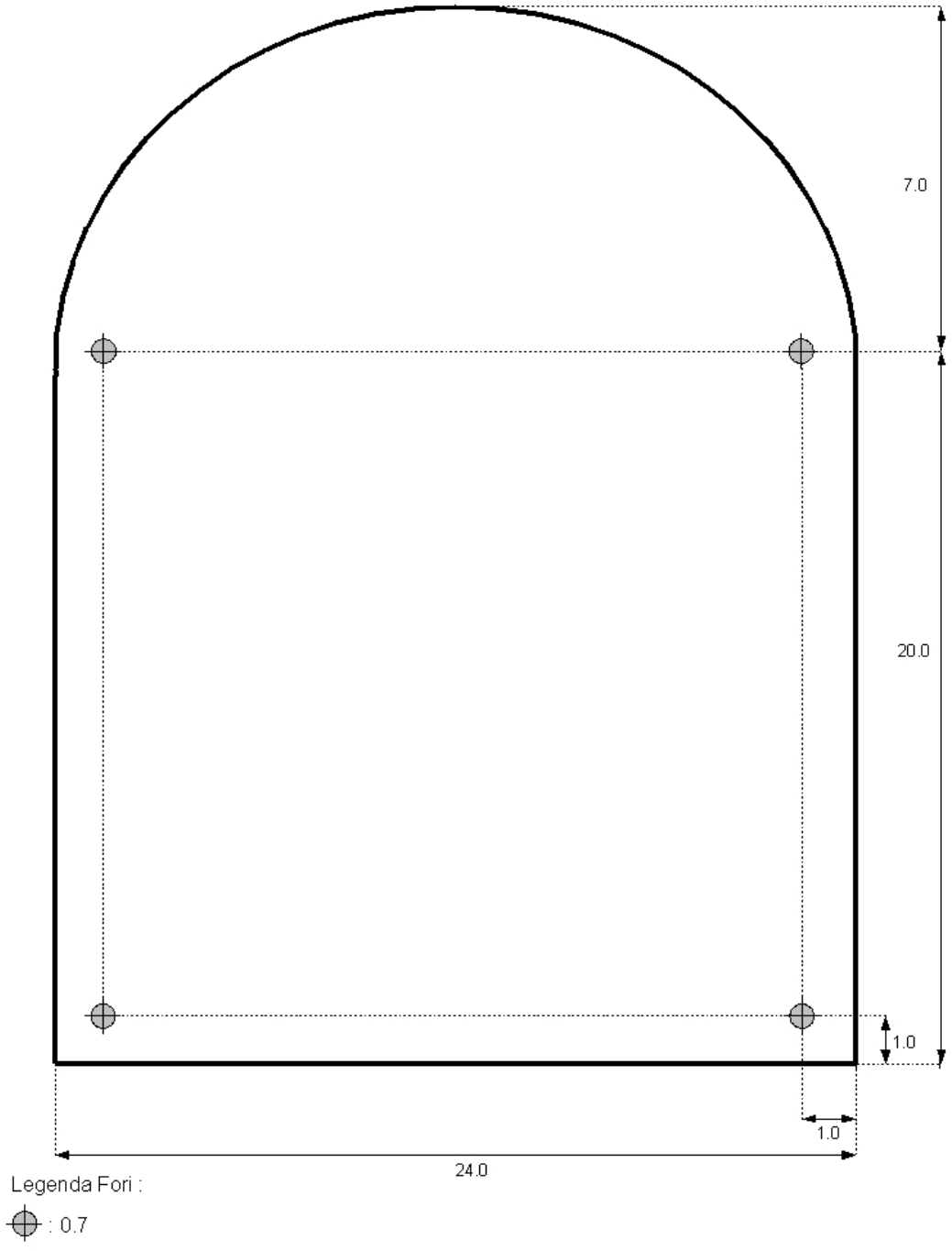



**Figura 2: Piano superiore di Domotino**

Con uno dei due piani ottenuti bisogna procedere ad ulteriori tagli in modo da ottenere il piano di Figura 3. La posizione dei fori da 0.5 cm può essere variata a seconda della disposizione dei circuiti sul piano stesso. Un'idea di come posizionare i circuiti è riportata in Figura 4.

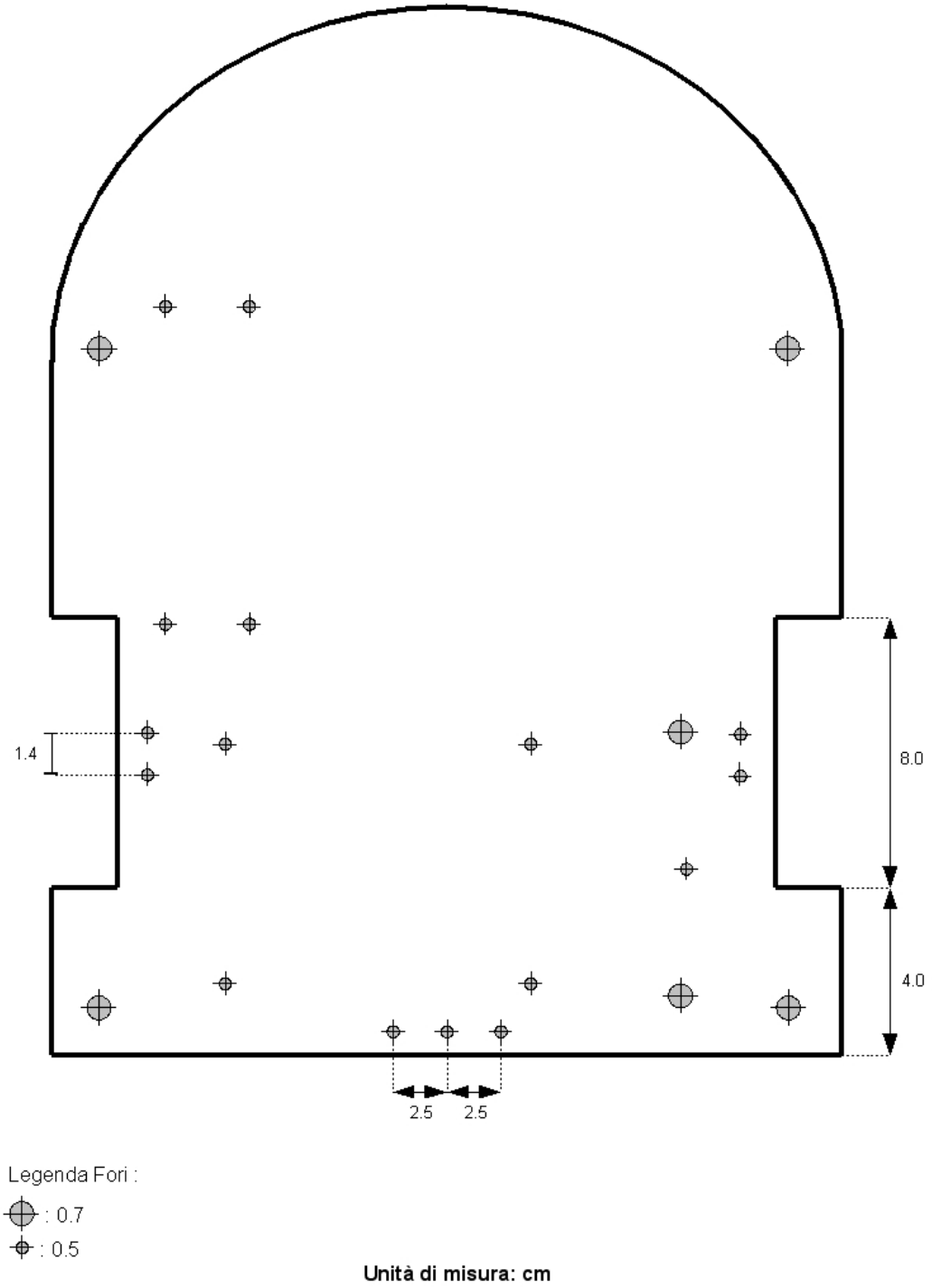



**Figura 3: Piano inferiore di Domotino**

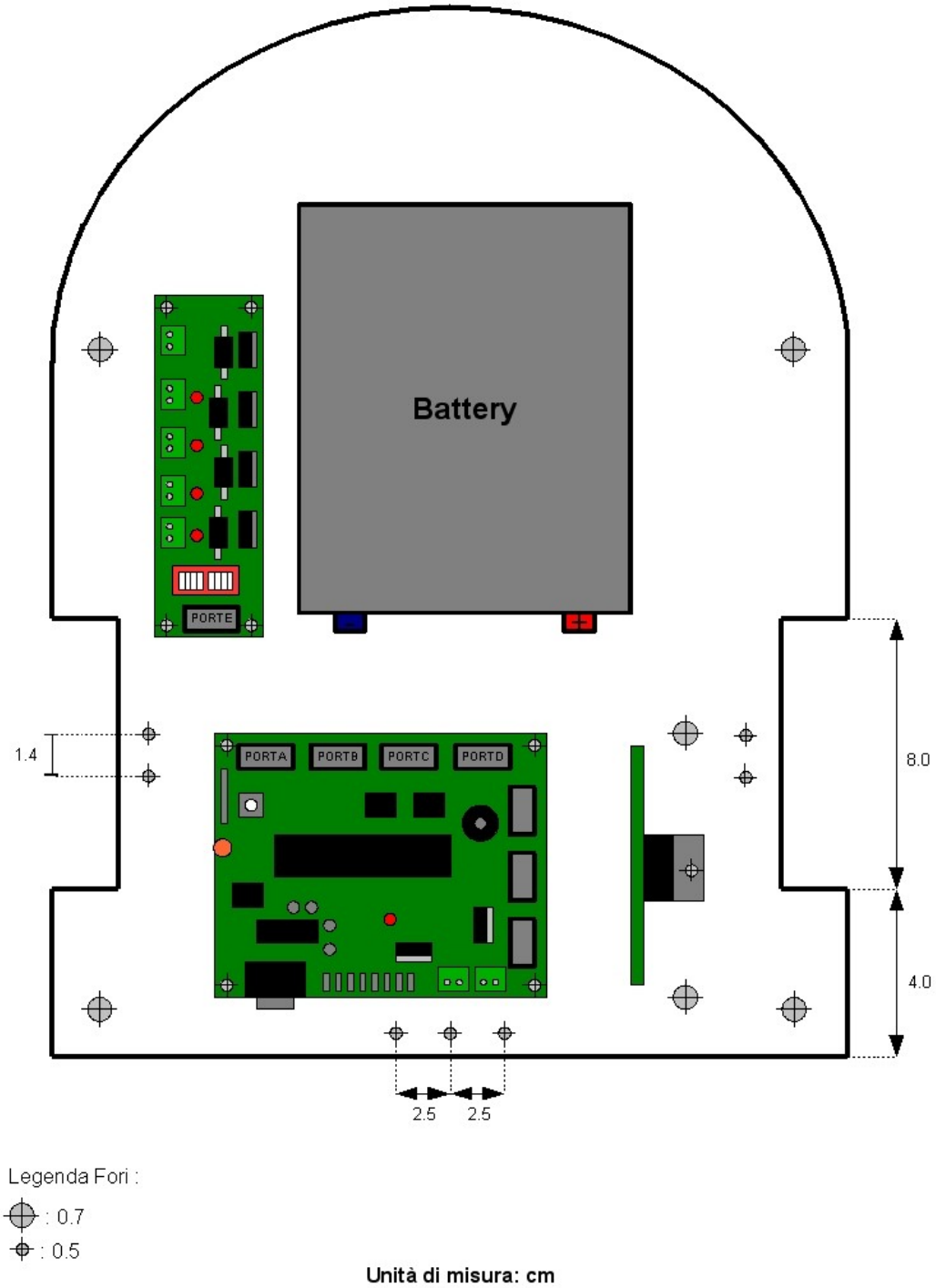
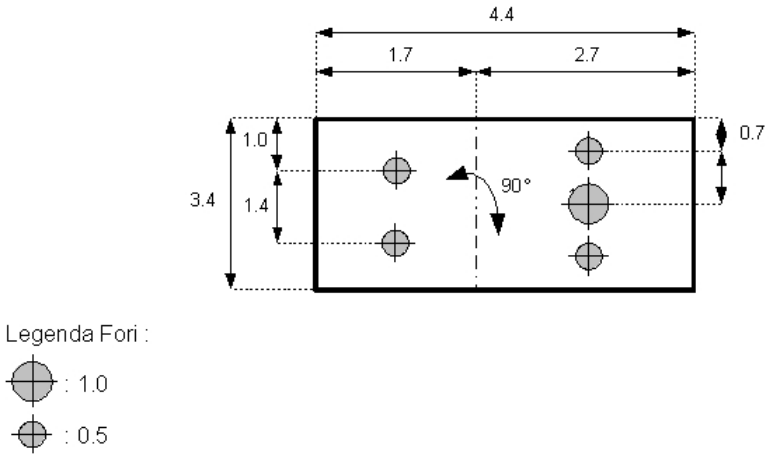



**Figura 4: Disposizione dei circuiti sul piano inferiore**

I due motori possono essere attaccati alla parte inferiore del secondo piano appena descritto, per mezzo di 2 L delle dimensioni di Figura 5. La parte è da piegare a 90° rispetto all'asse tratteggiato con punti e linee.



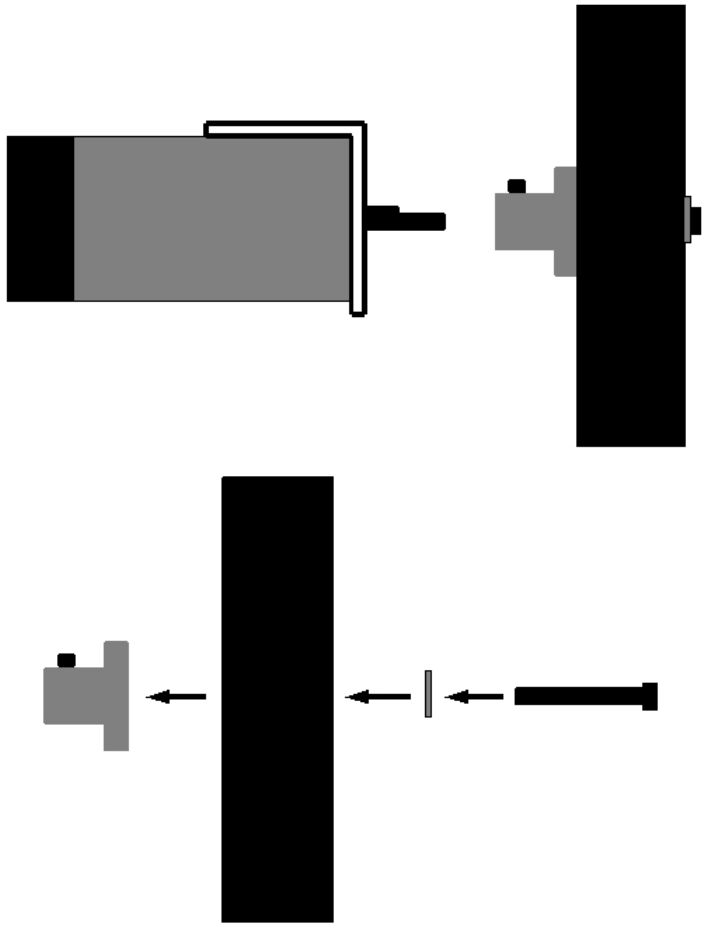



**Figura 5: Attacco ad L per i motori**

In Figura 5 è possibile anche vedere come attaccare le ruote ai motori. Come motori si è fatto uso di due motori da 12V con una coppia da 2.5Kg/cm, 75 giri al minuto e con encoder incorporato[3]. L'asse del motore è da 4mm e per poterlo attaccare alle ruote è necessario acquistare gli appositi mozzi da 4mm visibili in figura come due cilindri in grigio. Come ruote motrici si è fatto uso di due ruote al neoprene di diametro 7.6cm e spessore 1.9cm. Queste due ruote permettono di ottenere una buona presa su molti tipi di pavimento ma hanno il difetto di deformarsi facilmente.

Come ruota anteriore si è fatto uso di una comune ruota da armadietto, libera di ruotare a 360° intorno al proprio asse verticale. Il diametro di questa ruota più il suo piano di attacco dovrebbe essere di circa 5cm in maniera da mantenere in piano il robot. Il posizionamento dei motori e delle ruote è riportato in Figura 6.

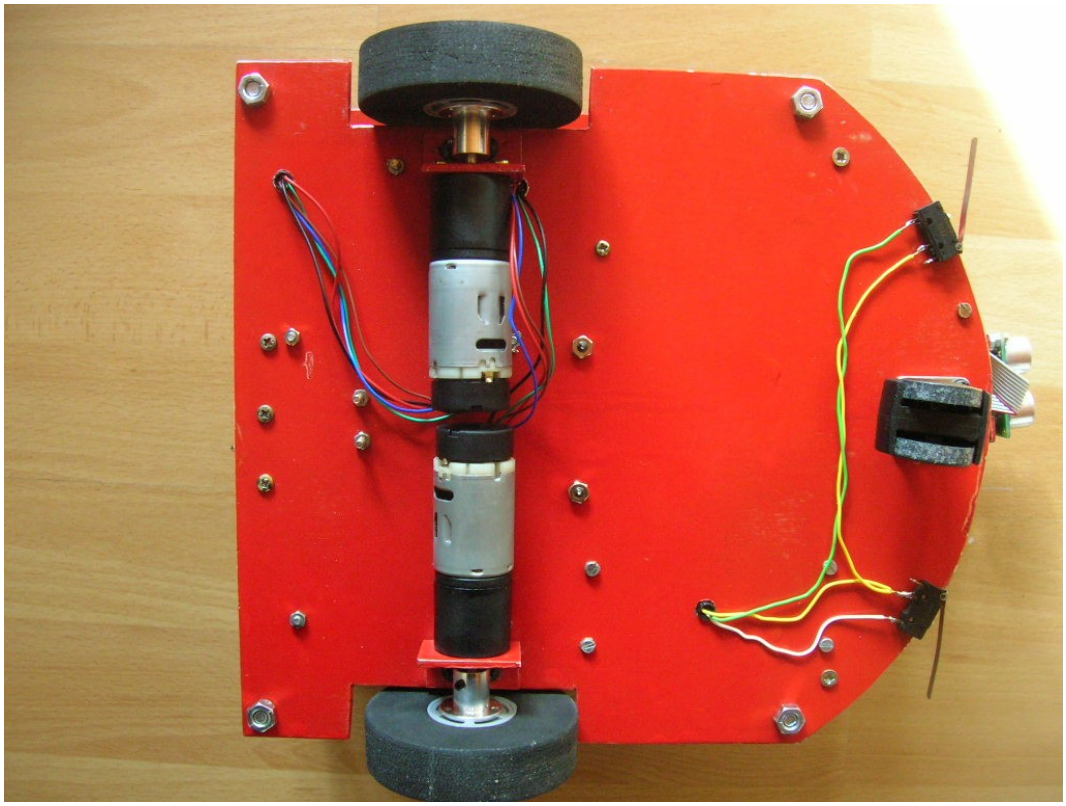


**Figura 6: Posizionamento dei motori e delle ruote**

Realizzati i supporti ad L per il motore è possibile passare al supporto ad L per il sensore ad ultrasuoni modello SRF05, come riportato in Figura 7. Il sensore ad ultrasuoni rappresenta l'organo visivo di Domotino. Al fine di ottenere una vista migliore si è posizionato il sensore con il suo supporto su di un servo in modo da poter cambiare la posizione dello “sguardo”. Questa soluzione rallenta lievemente la mobilità di Domotino rispetto all'uso di più sensori ad ultrasuoni, poiché bisogna continuamente spostare il sensore e misurare la distanza degli oggetti. L'utilizzo di più sensori richiede frequentemente l'uso di sensori con interfaccia I2C (per i quali Freedom è predisposto) i cui costi, per una soluzione a tre punti di osservazione, è superiore al sensore SRF05 più il servo.

Nonostante il sensore ad ultrasuoni possa guardare a destra e a sinistra quello che si ottiene non è un vero radar a causa del diagramma di radiazione che caratterizza il sensore stesso. Per tale ragione e al fine di limitare il tempo di misura, si controlla la distanza degli oggetti in soli tre punti: davanti, a

[3] In questa versione del robot non si è fatto uso degli encoder ma è consigliabile la loro presenza per compatibilità con versioni future del Firmware.

destra e a sinistra[4].

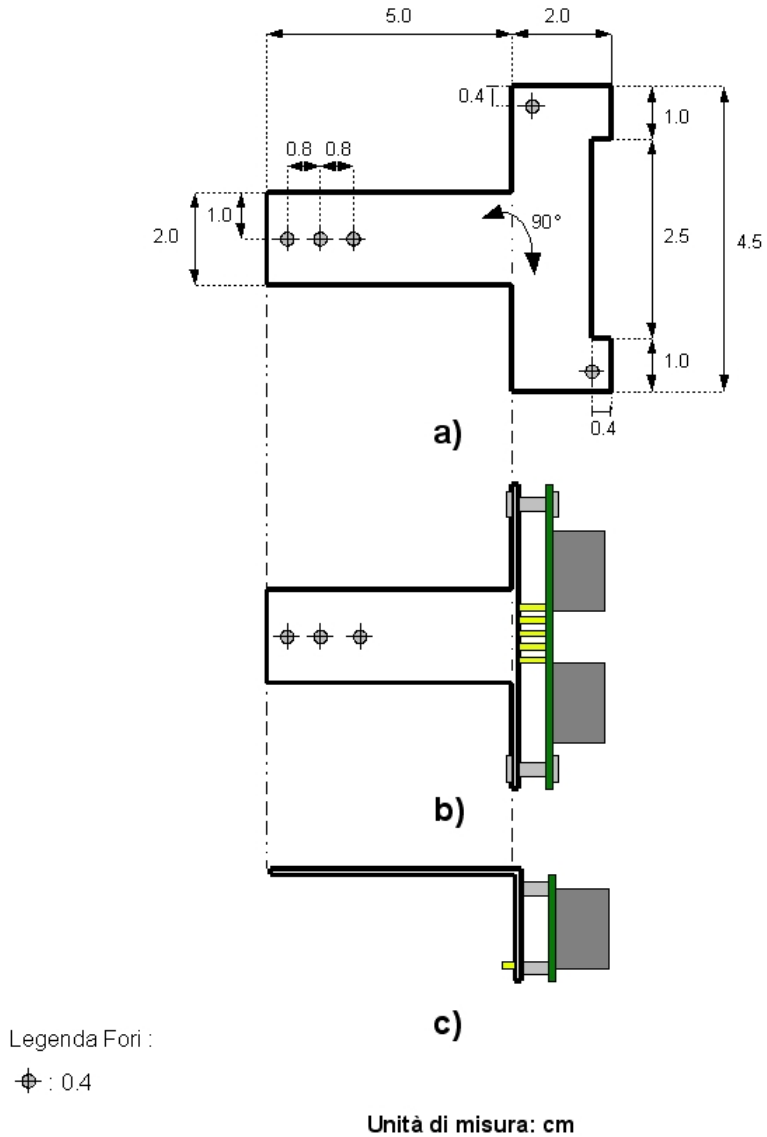



**Figura 7: Supporto per il sensore ad ultrasuoni**

[4] Le rotazioni sono da intendersi rispetto all'asse passante per il centro del robot.

Il supporto ora ottenuto deve essere attaccato al servo standard HS-422 il quale deve risultare attaccato al piano inferiore di Domotino per mezzo di rialzi da 5 cm. In Figura 8 è riportato un dettaglio di quanto esposto.

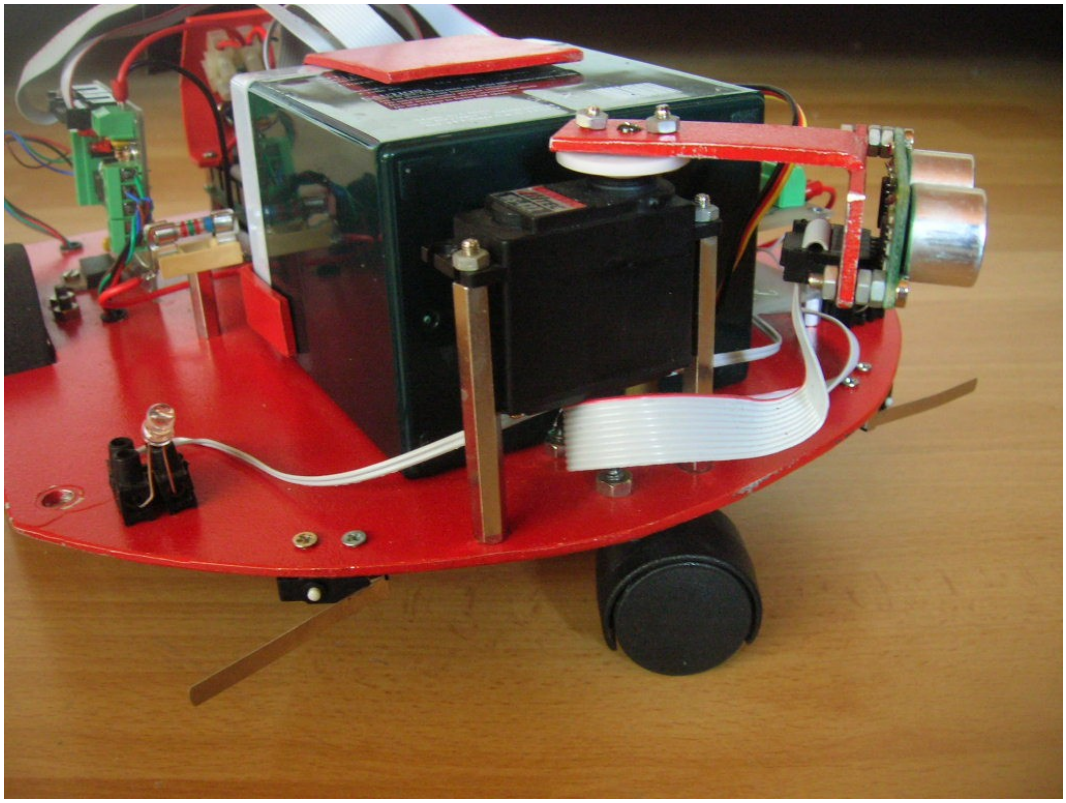


**Figura 8: Dettaglio del sensore ad ultrasuoni**

Realizzato il supporto per il sensore ad ultrasuoni si può procedere nella realizzazione della "cinta di sicurezza" per la batteria, come riportato in Figura 9; questa potrebbe variare a seconda della batteria che si utilizza[5]. La batteria utilizzata è al piombo da 12V 4.2Ah delle seguenti dimensioni 9x7x10. Generalmente le batterie al piombo possono essere ricaricate in maniera continua e senza limiti di corrente con tensioni comprese tra 13.5V e 13.8V[6]. Per limitare il peso e semplificare la struttura della cintura stessa questa non abbraccia la batteria nella parte anteriore. Per evitare che questa possa urtare il Servo è bene mettere un dado distanziatore, come quelli usati per il servo di Figura 8, che vincoli il piano della batteria non assicurato dalla "cintura".

[5] Il peso della batteria non deve essere superiore a 2.5Kg.
[6] Queste informazioni sono spesso stampate sulle batterie al piombo sigillate.

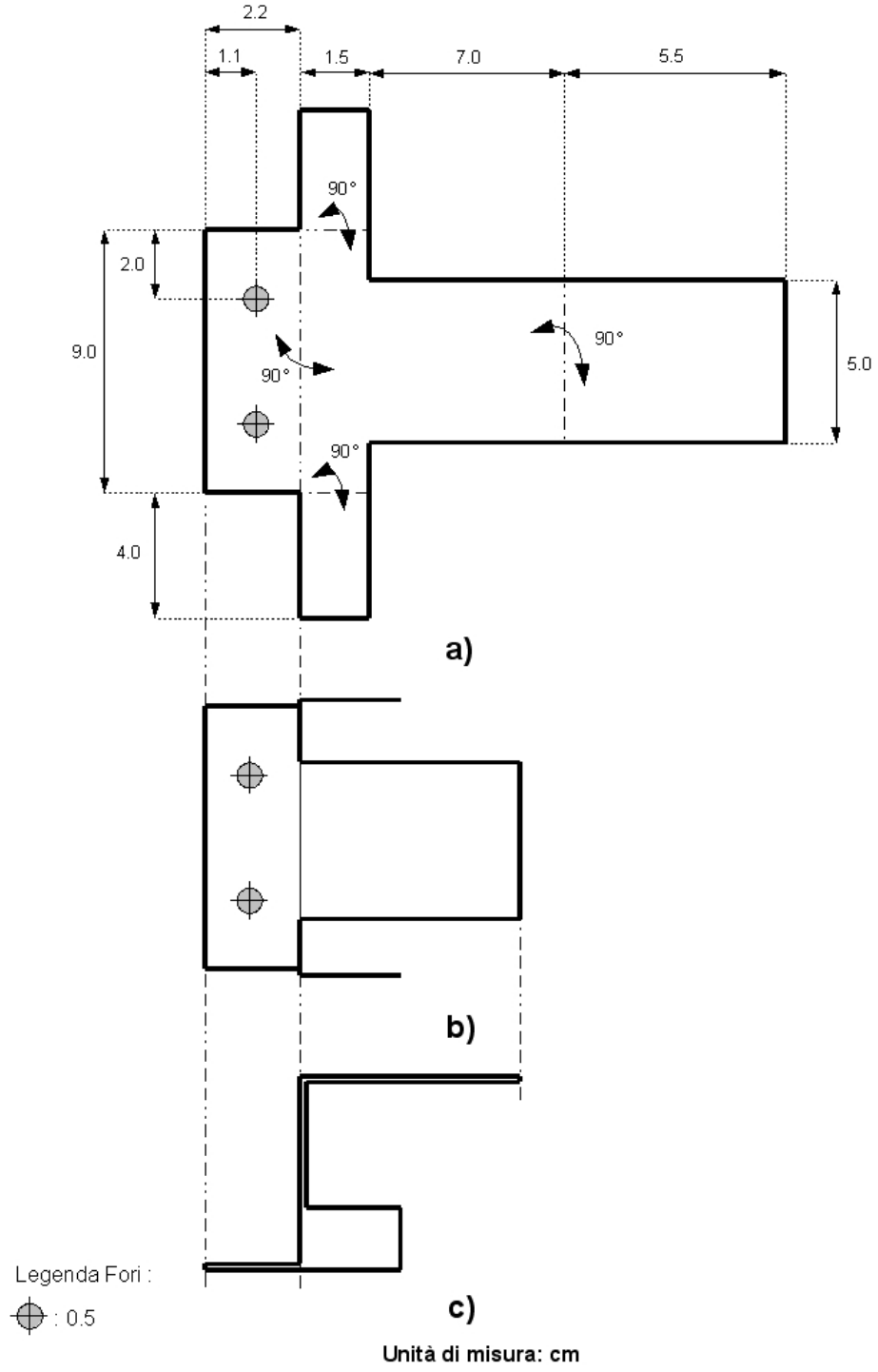



**Figura 9: Cintura di sicurezza per la batteria**

Dopo la "cintura di sicurezza" per la batteria si può procedere alla realizzazione del pannello utilizzato per il pulsante, l'interruttore, il connettore di ricarica della batteria e il display LCD 16x2[7]. La dimensione del pannello di Figura 10 possono essere facilmente variate a seconda del display LCD di cui si sta facendo uso. Le dimensioni dei fori possono variare a seconda del materiale utilizzato.

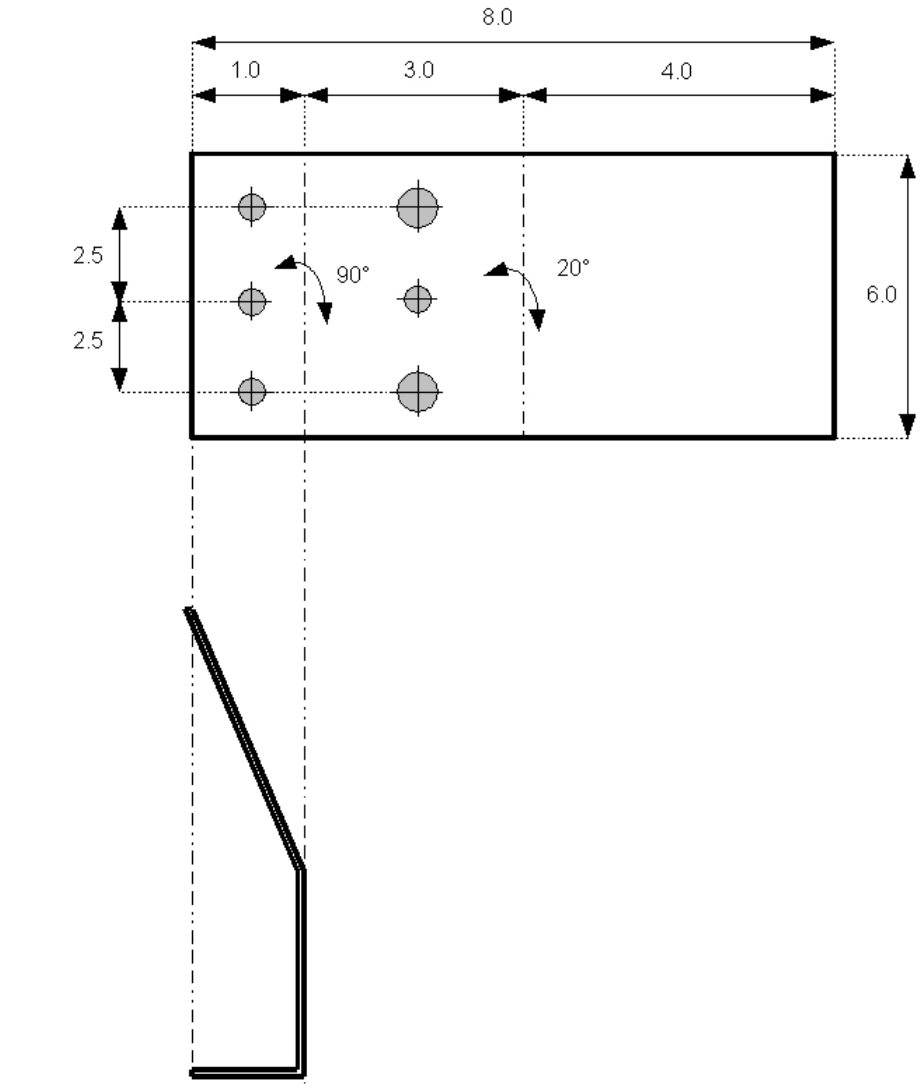



**Figura 10: Pannello di controllo**

[7] È possibile utilizzare anche altri tipi di display purché siano compatibili con Freedom e con il Firmware di Domotino.

In Figura 11 è riportato una foto di come si presenta il pannello appena descritto. È possibile osservare che il pannello realizzato è sospeso per mezzo di tre distanziatori in maniera da rendere i cavi posteriori accessibili senza dover disassemblare il pannello stesso.

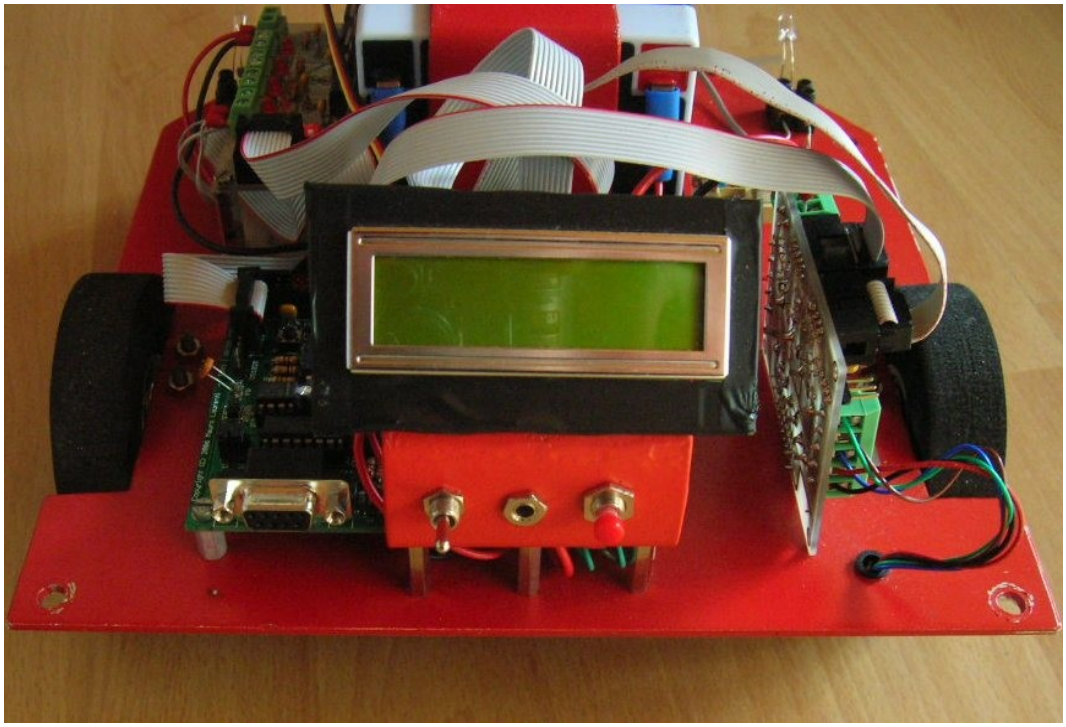


**Figura 11: Pannello di controllo di Domotino**

Un ultimo utensile da realizzare è rappresentato dal tool che verrà utilizzato per la pulizia della casa durante la modalità Explorer. Questo strumento è lasciato alla fantasia del lettore dicendo solo che non deve superare il peso di 1Kg e non deve ostacolare i movimenti del robot. Si capisce dunque che è possibile realizzare un vero e proprio aspirapolvere...

Per verificare la fattibilità di pulire casa ho realizzato un semplice strumento con le pezze che è possibile attaccare ai pulisci vetri. Questo tipo di pezza è risultata molto pratica nella sua rimozione e pulizia poiché alcuni modelli possono essere attaccati con semplici strip adesivi. Nonostante tale strumento sia particolarmente semplice[8] è possibile apprezzare una riduzione della sporcizia a terra tra il 60% e il 70%.

...realizzando strumenti più complessi Domotino non invidierà il suo fratello minore aspirapolvere. In Figura 12 e Figura 13 è riportato qualche dettaglio di come si presenta lo strumento di pulizia collegato sul retro di Domotino.

[8] La capacità di pulizia è anche legata all'algoritmo per il movimento di cui si fa uso. Ulteriori dettagli a riguardo verranno esposti a breve.

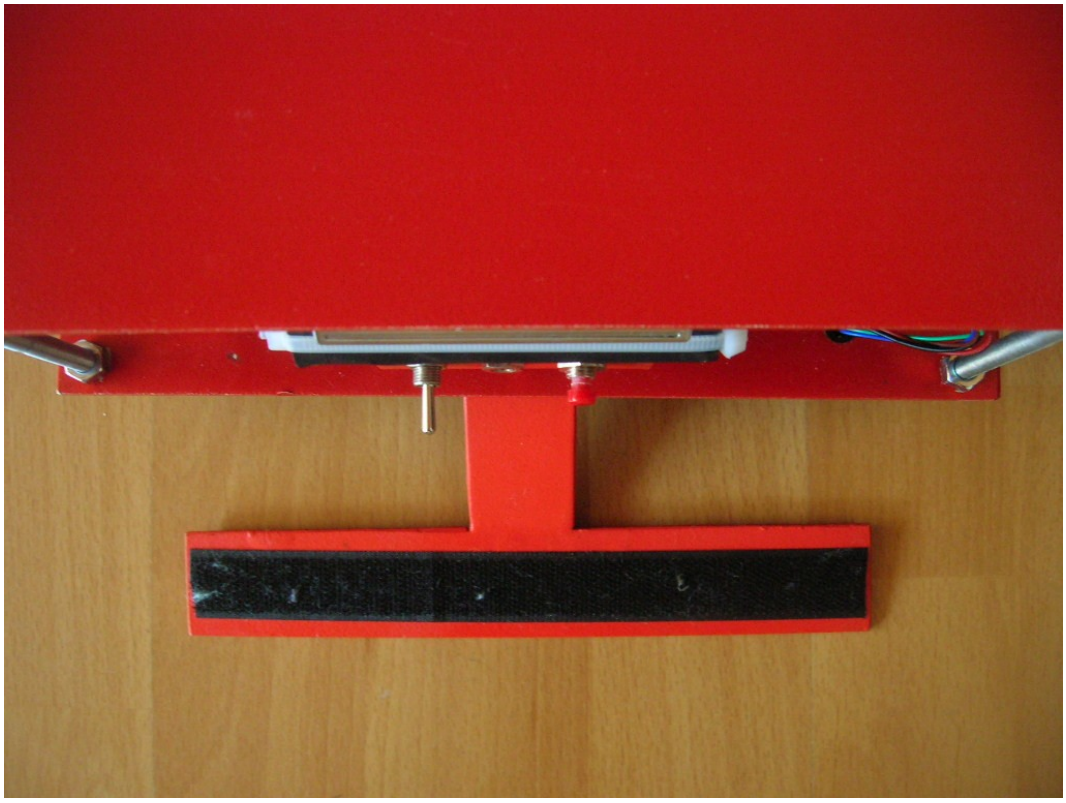


**Figura 12: Vista dello strumento di pulizia e lo strip adesivo**

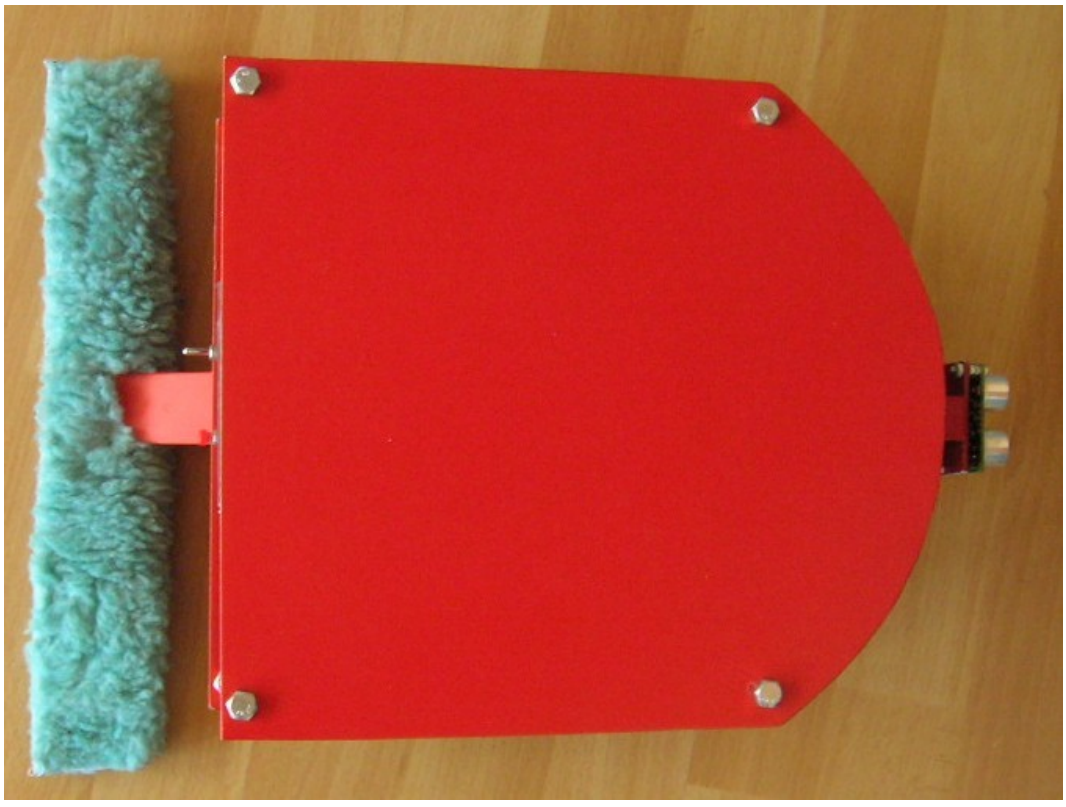


**Figura 13: Vista dello strumento di pulizia con la pezza attaccata**

Un'ultima nota merita la connessione dei due piani principali di Domotino. Come è stato già possibile vedere da alcune foto si è fatto uso di bulloni zincati. La lunghezza del bullone è da 10 cm e il suo diametro deve essere inferiore ai buchi da 0.7 cm fatti ai 4 angoli. Per il fissaggio dei 4 bulloni si è fatto uso di 12 dadi. I primi 8 dadi vanno avvitati sul piano superiore, come riporto in Figura 14, mentre gli altri 4 vanno avvitati alla base dei bulloni dopo averli fatti passare attraverso il piano inferiore di Domotino.

**Figura 14: Posizione dei primi 8 dadi sul piano superiore**

## Integrazione dei sistemi

Dopo una carrellata di foto e qualche callo sulle mani, finalmente la parte un po' più pulita. I sistemi che è necessario integrare su Domotino sono:

- Freedom
- Scheda di controllo motori DC e Stepper bipolari
- Scheda di controllo Open Drain e Motori Stepper unipolari da 3A
- Sensore ad ultrasuoni
- Display LCD[9]
- Servo

Per i dettagli tecnici su questi sistemi si rimanda alla relativa documentazione che è possibile scaricare dal sito www.LaurTec.com. In particolare la spiegazione del sensore ad ultrasuoni è contenuta nel progetto "Metro ad Ultrasuoni".

In Figura 18 è riportato lo schema a blocchi del sistema rappresentato dal nostro robot. Il cervello di Domotino è rappresentato da Freedom per mezzo del quale si controlla tutto il resto del sistema. In particolare i due motori utilizzati per il movimento sono collegati alla "Scheda di controllo motori DC e Stepper bipolari" che a sua volta è collegata a Freedom. La "Scheda di controllo Open Drain e Motori Stepper unipolari da 3A" è utilizzata come buffer per la PORTE di Freedom in modo da poter controllare lampade o sirene più potenti. Il sensore ad ultrasuoni, il display LCD e il servo sono direttamente collegati a Freedom.

Vediamo come impostare le singole schede affinché possano essere correttamente integrate nel sistema.

### Freedom

La scheda Freedom deve essere impostata per poter funzionare con i sensori ad ultrasuoni di tipo Trig/Echo. Per poter fare questo i jumper TRIG ed ECHO devono essere impostati come riportato in Figura 15.

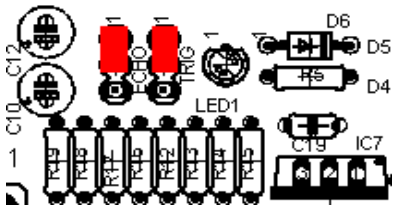



**Figura 15: Disposizione Jumper per Sensore ad ultrasuoni**

La scheda deve inoltre essere impostata per poter funzionare con la connessione RS232. Per fare questo è necessario impostare i jumper RX e TX come riportato in Figura 16.

Il microcontrollore utilizzato è il PIC18F4580 all'interno del quale deve essere caricato il firmware che è possibile scaricare direttamente dal sito www.LaurTec.com dove è possibile trovare ogni suo aggiornamento. Il quarzo della scheda deve essere di 20MHz mentre la memoria EEPROM deve essere una 24LC512.

[9] Il display utilizzato è 16x2 compatibile con il controllore Hitachi 44780.

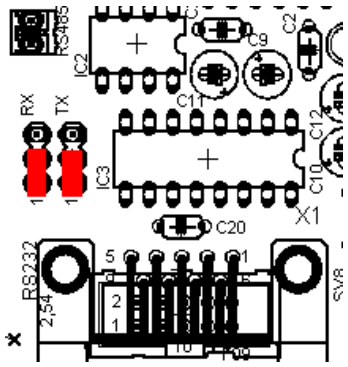



**Figura 16: Disposizione jumper per Trasmissione RS232**

### Scheda di controllo motori DC e Stepper bipolari

La scheda di controllo per i due motori DC deve essere impostata per funzionare con un solo segnale PWM. Per fare questo è necessario configurare il jumper PWM2 come riportato in Figura 17. Se la scheda non è posizionata come in Figura 11 ovvero con l'integrato L298 a contatto con il piano in alluminio inferiore, è necessario porre un'aletta di raffreddamento di opportune dimensioni sul L298.

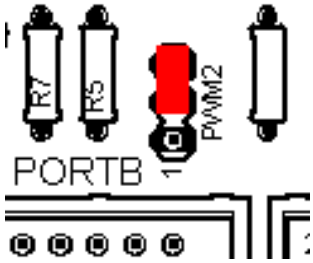



**Figura 17: 1 segnale PWM**

### Scheda di controllo Open Drain e Motori Stepper unipolari da 3A

Dal momento che tale scheda è collegata alla PORTE per la quale sono disponibili solo i pin RE0, RE1, RE2 i dip switch devono essere, partendo dal più significativo nella seguente configurazione: 00000111, cioè bisogna collegare solo i pin RE0, RE1, RE2.

Come riportato in Tabella 1 il pin RE1 è collegato al cicalino di sistema. Questo significa che questo output non potrà essere utilizzato per scopi generici ma potrà risultare utile per il collegamento di una sirena più potente in modo da rendere la modalità allarme o la sveglia, a prova di sordo!

### Display LCD

Il display LCD deve essere impostato in maniera da lavorare correttamente con Freedom ovvero in modalità 4bit. Per fare questo bisogna lasciare scollegati i bit D0, D1, D2, D3 del bus dati del controllore a bordo del display LCD[10].

Per quanto riguarda l'integrazione del sensore ad ultrasuoni e il servo, non è necessaria nessuna impostazione.

[10] Per maggior dettagli sulla corrispondenza tra i pin dell'LCD e il connettore disponibile su Freedom si rimanda alla documentazione del progetto Freedom.

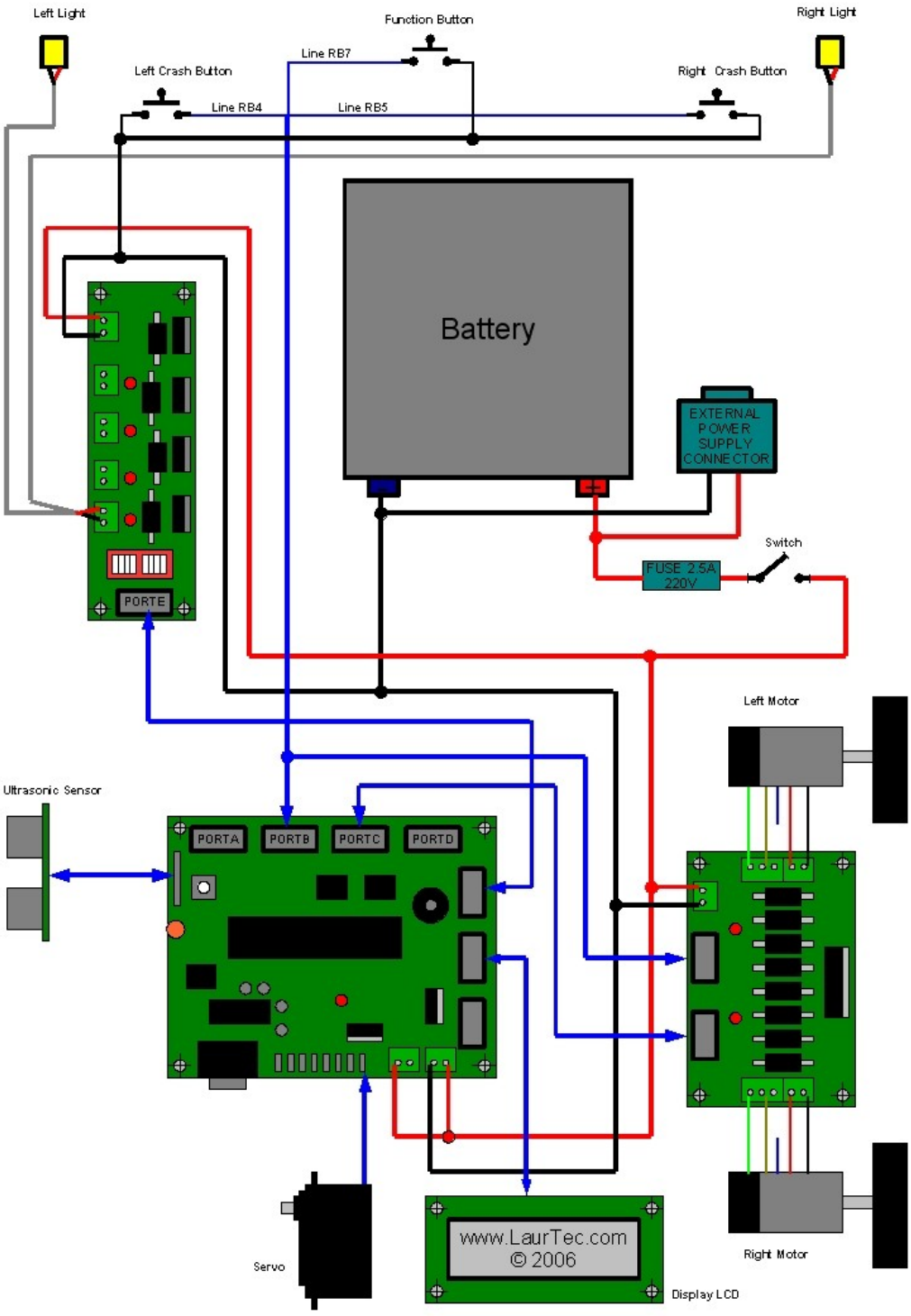



**Figura 18: Diagramma a blocchi del sistema**

Vediamo di comprendere lo schema a blocchi del nostro sistema. Le linee rosse e nere sono rispettivamente i +12V della batteria e la massa. La massa risulta collegata alla carcassa metallica del del robot. In particolare è possibile vedere che vicino al polo positivo della batteria è presente un fusibile da 2.5A 220V per mezzo del quale si protegge la circuiteria e la batteria stessa da eventuali correnti eccessive. Il polo positivo e negativo vanno anche all'External Power Supply Connector per mezzo del quale è possibile ricaricare la nostra batteria[11].

Le linee rosse e nere collegate al cavo grigio che va alla lampada destra e sinistra, rappresenta appunto il cavo di alimentazione per le stesse. Le lampade possono essere di qualunque potenza purché rientrino nei limiti delle specifiche di funzionamento della "Scheda di controllo Open Drain e Motori Stepper unipolari da 3A". Si sconsiglia l'utilizzo di lampade non eccessivamente potenti in modo da poter allungare l'autonomia del robot in termini energetici ed evitare l'utilizzo di alette di raffreddamento per i MOS della scheda. Qualora si faccia uso di led bianchi bisogna rispettare la polarità di collegamento, ovvero l'anodo deve essere connesso al terminale positivo e il catodo al terminale negativo, inoltre dal momento che l'alimentazione è circa 12V bisogna mettere una resistenza in serie ad ogni diodo LED pari a 560ohm, dunque o l'anodo o il catodo non risulterà collegato direttamente alla batteria.

I motori per il movimento del robot vanno collegati come riportato nello schema a blocchi ovvero lasciando scollegato il cavo centrale relativo all'uscita dell'encoder interno al motore stesso.

Le linee blu dello schema a blocchi rappresentano i bus, ovvero più fili. Gran parte delle connessioni blu sono rappresentate da ribbon cable ovvero cavetti piatti con 10 fili. La lunghezza di questi cavi verrà a dipendere dal posizionamento delle schede sulla piattaforma di alluminio. In particolare nel collegamento del sensore ad ultrasuoni bisogna fare attenzione a collegare al verso giusto il ribbon cable in maniera da evitare il danneggiamento del sensore stesso. Il servo deve essere collegato rispettando lo standard del connettore presente su Freedom.

Dal momento che l'alimentazione a batteria carica può essere di circa 14V e il servo si ferma e riparte in continuazione si hanno molti picchi di corrente e la d.d.p ai capi del regolatore è tale per cui il regolatore risulta termicamente stressato. Per questa ragione è bene provvedere una piccola aletta di raffreddamento per il regolatore di tensione dei servo.

Per posizionare il supporto con il sensore ad ultrasuoni nella posizione centrale è bene non avvitare la vite tra il supporto e il servo fino a completamento dell'intero sistema. A sistema ultimato basterà attivare la modalità Explorer e il sistema posizionerà il servo nella posizione centrale. Entro tre secondi, ovvero prima che il servo cominci a guardare a destra e sinistra bisogna cambiare modalità in maniera da lasciare il servo nella posizione centrale. A questo punto basterà staccare il supporto per il sensore ad ultrasuoni ed allinearlo con l'asse centrale del robot. Solo a questo punto è possibile avvitare il supporto stesso al Servo.

Dallo schema a blocchi è possibile notare che il bus PORTB va collegato oltre che alla scheda di controllo dei motori anche ai baffi che rilevano eventuali urti nonché al pulsante per il cambio della modalità. Per questa ragione è bene prevedere una piccola mille fori con connettore ML10E per mezzo della quale è possibile prelevare le linee RB4, RB5 e RB7 dal bus stesso.

I buffi sono rappresentati da semplici pulsanti che vengono premuti da strutture (stecchette metalliche) la cui forma può variare a seconda delle esigenze. In particolare è possibile collegare per ogni pulsante (urto a destra o urto a sinistra) più pulsanti in serie in maniera da aumentare la superficie d'urto a cui il robot sarà sensibile.

Ora non ci resta che vedere come appariranno i collegamenti ad assemblaggio ultimato. In Figura 19 si può avere un idea di come sarà Domotino dopo l'integrazione dei vari sistemi. In particolare è possibile osservare come devono essere posizionati i dip switch della "Scheda di controllo Open Drain e Motori Stepper unipolari da 3A".

[11] Si fa notare che il connettore di ricarica della batteria non è protetto da nessun fusibile poiché si presuppone che il caricabatteria sia oppurtunatamente protetto con la propria circuiteria.

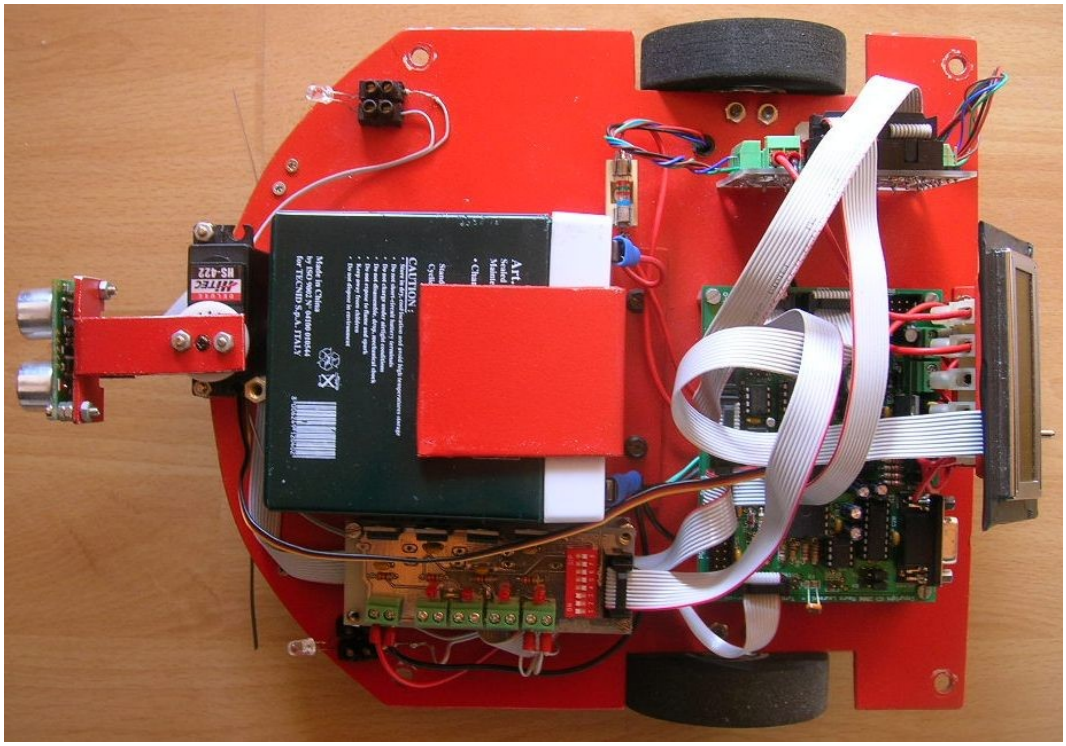


**Figura 19: Domotino a collegamenti ultimati**

Le caratteristiche tecniche del sistema sono:

**Tensione Alimentazione:** 12V
**Corrente a riposo:** 60mA
**Corrente massima a motori accesi:** 1250mA
**Corrente per LED luce bianca ON:** 50mA (incluso led Scheda di controllo Open Drain e Motori Stepper unipolari da 3A)

In Tabella 1 sono riportate le funzioni dei singoli pin del PIC all'interno del sistema Domotino. Questa tabella risulterà particolarmente utile qualora si dovesse personalizzare il software base per il robot.

| Pin | Funzione |
|---|---|
| **RA0** | Ingresso fotoresistenza FR |
| **RA1** | Ingresso livello Batteria |
| **RA2** | Ingresso Analogico disponibile |
| **RA3** | Ingresso Analogico disponibile |
| **RA4** | I/O Digitale disponibile |
| **RA5** | Ingresso Termometro analogico LM35DZ |
| **RA6** | Disponibile nei PIC in cui si faccia uso dell'oscillatore interno (se disponibile) |
| **RB0** | Linea d'interrupt per il Real Time Clock/Calendar |
| **RB1** | R*/W enable per trasmissione dati RS485 |
| **RB2** | Controllo verso rotazione motore destro |
| **RB3** | Controllo verso rotazione motore sinistro |
| **RB4** | Pulsante baffo sinistro |
| **RB5** | Pulsante baffo destro |
| **RB6** | Disponibile per uso encoder motore sinistro |
| **RB7** | Pulsante per cambio modalità |
| **RC0** | Segnale di TRIG per il sensore ad ultrasuoni |
| **RC1** | Possibile espansione per PWM |
| **RC2** | PWM |
| **RC3** | SCL per bus I2C; utilizzato dalla EEPROM e dal Real Time Clock/Calendar |
| **RC4** | SDA per bus I2C; utilizzato dalla EEPROM e dal Real Time Clock/Calendar |
| **RC5** | Segnale di ECO per il sensore ad ultrasuoni |
| **RC6** | Linea TX per trasmissioni RS232 o RS485 |
| **RC7** | Linea RX per trasmissioni RS232 o RS485 |
| **RD0** | Segnale per il controllo del Servo SV1 |
| **RD1** | Segnale per il controllo del Servo SV2 |
| **RD2** | Segnale RS LCD |
| **RD3** | Segnale E LCD |
| **RD4** | Segnale DB4 LCD |
| **RD5** | Segnale DB5 LCD |
| **RD6** | Segnale DB6 LCD |
| **RD7** | Segnale DB6 LCD |
| **RE0** | Lampada destra e sinistra |
| **RE1** | Uscita cicalino |
| **RE2** | Disponibile |

**Tabella 1: Descrizione delle funzioni associate ai singoli pin del PIC**

## Analisi del Firmware

Arrivati a questo punto Domotino ha preso forma e se il firmware è stato già caricato all'interno del microcontrollore, Domotino è anche in grado di compiere i primi passi! Unico problema è che il sistema è ancora una scatola nera...o meglio una scatola rossa! Cerchiamo ora di capire i pensieri di Domotino...

Il programma sorgente di Domotino è strutturato con il file principale main.c e i seguenti Header Files[12]:

- commands.h
- generic.h
- init.h
- mod.h
- motion.h
- motor.h
- sensors.h
- servo.h
- var.h

La ragione per cui il programma è strutturato in più parti discende dalla complessità del sistema che se da un lato la si è semplificata molto, inevitabilmente è rimasta complessa. La divisione in più file permette di avere sempre un certo ordine e di focalizzarsi sempre su un numero limitato di problemi.

Nel caso di un lavoro di gruppo, definite le specifiche con le quali le varie funzioni contenute in ogni file dovranno comunicare con l'esterno, può essere un modo per dividere il lavoro stesso!

La spiegazione che seguirà non entrerà nei dettagli della singola istruzione ma manterrà un distacco a livello funzionale in maniera da facilitare la comprensione e avere la possibilità di cambiare le singole funzioni mantenendo i contenuti di questa documentazione ancora validi. La trattazione che seguirà se pure non entrerà nel dettaglio del programma presuppone una conoscenza base dell'ANSI C e in particolare del C18[13]. Per ragioni concettuali la spiegazione non procederà secondo l'ordine alfabetico con cui le funzioni sono state precedentemente presentate. Ci siamo...

### File main.c

Il file main.c contiene le inizializzazioni base per il corretto funzionamento del PIC, in particolare si fa uso di varie direttive al fine di configurare i registri interni del PIC. Per mezzo della direttiva #include vengono inclusi tutti i file di libreria necessari per il corretto funzionamento di Domotino. I file inclusi sono in parte librerie associate al C18 e in parte librerie personalmente scritte per la gestione di periferiche di uso generale; queste funzioni sono :

```
#include "\library\PCF8563.h"
#include "\library\eeprom.h"
#include "\library\LCD_44780_Freedom.h"
```

Come si vede sono contenute nella directory libreria presente nella directory principale Robot Domotino, in cui è possibile trovare anche le altre librerie che ho scritto. Oltre a questi file si includono tutti gli Header file precedentemente elencati presenti nella directory Robot interna alla directory principale Robot Domotino. Ogni altra libreria che dovesse essere inserita da voi deve essere inclusa dopo questi file, in maniera d'esser certi di utilizzare funzioni precedentemente dichiarate all'interno del sistema.

[12] I file sorgente è possibile scaricarlo al sito www.LaurTec.com.

[13] Per le conoscenze del C18 necessarie ad una corretta comprensione del Firmware si rimanda al Tutorial "C18 step by step".

Dopo questa serie di direttive vi è la parte di codice per mezzo del quale è possibile gestire le interruzioni. Le interruzioni gestite sono quelle derivanti dal TIMER0, per mezzo del quale si scandiscono tutte le temporizzazioni del sistema. Gli eventi controllati da questo timer sono:

- l'impulso per il sensore ad ultrasuoni
- aggiornamento dell'orario e della data di sistema
- rinfresco della posizione del servo
- Time out per l'invio di un comando dal PC[14]
- Lettura dell'intensità luminosa per discriminare la notte dal giorno
- Lettura della temperatura
- Lettura dello stato della batteria

Per mezzo delle interruzioni si controlla inoltre la ricezione dei dati attraverso l'USART ovvero dei dati trasmessi dal PC. Un altro importante controllo gestito per mezzo delle interruzioni è quello dei pulsanti che segnalano il crash di Domotino con eventuali ostacoli. Anche il pulsante per il cambio della modalità è gestito per mezzo delle interruzioni. Maggiori dettagli sulle funzioni svolte quando vengono ricevuti degli interrupt si avranno più tardi. Dopo la gestione delle interruzioni inizia la funzione void main (void). All'interno di questa funzione vengono semplicemente impostati gli ingressi e le uscite delle varie porte del PIC, successivamente viene chiamata la funzione initialize ( ) per mezzo della quale vengono svolte altre inizializzazioni del sistema. Dopo questa inizializzazione inizia un loop infinito che rappresenta una scelta di progetto chiave.

```
while (1)       //ciclo infinito in attesa d'interrupt
  {
      if (modality = = MODA)
            modA ( );

      if (modality = = MODB)
            modB ( );

      if (modality = = MODC)
            modC ( );

      if (modality = = MODD)
            modD ( );

      if (modality = = MODE)
            modE ( );

      if (modality = = MODF)
            modF ( );

      if (modality = = MODG)
            modG ( );
  }
```

La variabile modality contiene il numero della modalità che è attualmente in esecuzione per mezzo dei controlli con le costanti MODA, MODB ecc... che altro non sono che numeri associati alle varie modalità. Non ci si è capito molto! Normale non c'è ancora la struttura del programma, per questo

[14] Domotino può ricevere comandi dal PC per mezzo di caratteri ASCII. Ogni comando è composto da 8 caratteri i quali devono essere inviati entro circa due secondi dall'invio del primo carattere.

ritornerò su questo punto quando avrò illustrato le variabili.

**File generic.h**

In questo file sono contenute alcune funzioni generiche che possono risultare utili nella personalizzazione del firmware. Oltre a queste funzioni è presente la funzione InitializationText ( ) utilizzata internamente al sistema per trasmettere tramite la porta seriale informazioni riguardanti al sistema stesso e i risultati di alcuni test. Dalla funzione InitializationText ( ), dopo il test sull'indirizzo della EEPROM 0x0000 viene eventualmente richiamata la funzione firstRun ( ) sempre parte delle funzioni di generic.h. La funzione firstRun ( ) permette di caricare in EEPROM i valori di default delle costanti di sistema, qualora si sia cambiata la EEPROM o sia la prima volta che si accende Domotino. Queste funzioni non dovrebbero essere utilizzate nelle vostre applicazioni. Le funzioni che possono invece ritornare utili sono:

| Funzione | Descrizione |
|---|---|
| **void** beep ( ) | Richiamando questa funzione viene generato un beep sul cicalino piezoelettrico collegato su RE1. |
| **void** Delay (int time) | Per mezzo di questa funzione è possibile avere dei ritardi in particolare Delay (1) corrisponde a circa 100ms, quindi per avere 1s di ritardo bisogna scrivere Delay (10). |
| **void** copyStr (**char** * dest, **char*** orig) | Per mezzo di questa funzione è possibile copiare una stringa origine all'interno di un'altra variabile stringa destinazione. |
| **unsigned char** deltaTime (**unsigned char*** Time1, **unsigned char*** Time2) | Per mezzo di questa funzione è possibile calcolare la differenza in minuti tra due orari. Il risultato deve essere al massimo 255 minuti. Gli orari devono appartenere allo stesso giorno solare altrimenti la variabile di ritorno avrà un valore negativo. |

**Tabella 2: Funzioni di uso generale**

**File init.h**

All'interno di questo file è contenuta la sola funzione void initialize ( ) per mezzo della quale viene iniziallizzato tutto l'hardware interno al PIC come per esempio l'USART, il bus I2C, i Timer e vengono inoltre caricate tutte le variabili con i dati contenuti in EEPROM. In questa funzione viene richiamata la funzione InitializationText ( ) non a caso dopo aver propriamente inizializzato l'USART. Per mezzo della funzione InitializationText ( ), oltre al semplice invio di testo alla porta seriale[15] vengono effettuati dei controlli di sistema.

Il primo controllo consiste nel verificare se il Firmware è stato avviato per la prima volta. Questo controllo avviene semplicemente leggendo l'indirizzo EEPROM 0x0000. Se il valore è diverso da 0x55 si considera che Domotino è appena nato, dunque il Firmware è alla sua prima esecuzione. Questo evento viene memorizzato all'indirizzo 0x0000 scrivendo 0x55 in modo tale che al secondo avvio il sistema riconosca “che era già vivo”!

Oltre a scrivere il valore 0x55 viene eseguita la funzione firstRun ( ) per mezzo della quale si caricano in EEPROM tutti i valori di default delle costanti di sistema. Queste variabili possono essere cambiate facendo uso degli opportuni comandi che verranno di seguito descritti.

Il secondo controllo consiste nel verificare se l'utente ha mai inizializzato le costanti di sistema in maniera da personalizzare il sistema. Domotino funzionerà anche con i valori di Default ma segnalerà una warning qualora l'operatore non abbia volontariamente cambiato le costanti interne. Il controllo se le costanti sono state o meno cambiate consiste semplicemente nel leggere il valore della EEPROM all'indirizzo 0x0001. Se questo valore dovesse essere diverso da 0xAA vorrebbe dire che le costanti non sono mai state modificate dall'utente. Il valore 0xAA all'indirizzo 0x0001 dovrebbe essere scritto

[15] Il testo inviato come verrà di seguito spiegato viene visualizzato per mezzo del programma Domotino GUI Controller.

manualmente dopo aver oppurtunatamente impostato le costanti con gli opportuni comandi o più semplicemente utilizzando Domotino GUI Controller che provvederà automaticamente a scrivere il valore 0xAA all'indirizzo 0x0001 dopo aver impostato le costanti di sistema[16].

Il terzo controllo consiste nel verificare se la EEPROM funziona correttamente[17]. Questo test è fatto leggendo il dato della EEPROM all'indirizzo 0x0002, incrementarlo e scrivere il suo incremento nuovamente all'indirizzo 0x0002. Dopo la scrittura il dato viene riletto e confrontato con il valore precedentemente scritto; se i due valori coincidono il test è passato altrimenti è fallito. Se per esempio all'indirizzo 0x0002 è presente il valore 0x0C, questo verrà incrementato ottenendo 0x0D. Questo nuovo valore verrà scritto all'indirizzo 0x0002 che dovrà dunque valere 0x0D e non più 0x0C.

Il quarto controllo consiste nel verificare se la dimensione della EEPROM è quella esatta, ovvero una 24LC512. Questo controllo è identico al controllo precedente ma effettuato all'ultimo indirizzo, ovvero 0xFFFF.

Il quinto controllo consiste nel verificare che il real time clock calendar funzioni correttamente. Il test consiste semplicemente nel leggere i secondi e aspettare che ne passino due[18].

Alla fine dei Test, il testo che viene trasmesso attraverso la porta seriale è il seguente:

```
**********************************************************

Firmware Robot Domotino
Version 1.1
Copyright 2006 by Mauro Laurenti

**********************************************************

Domotino first Execution.

WARNING : Constants setting is recomended

EEPROM Test: PASSED

EEPROM Size Test: PASSED

Clock/Calendar Test: PASSED

System Ready...
```

Qualora qualche test fallisca si otterrà il messaggio FAILED invece di PASSED. In alcuni casi eventuali errori causeranno lo stallo del programma senza la visualizzazione FAILED. Naturalmente il punto in cui il programma si blocca è un buon candidato all'essere la causa del problema.

Dopo le varie inizializzazioni e test il PIC viene abilitato al riconoscimento e alla gestione delle interruzioni. Questa funzione non deve essere modificata se non si ha una buona conoscenza del PIC...e una buona ragione!

### File servo.h

All'interno di questo file sono contenute le funzioni per comandare il Servo. In particolare è presente una funzione per centrare il servo, una funzione per guardare a sinistra e una funzione per guardare a destra. Queste funzioni sono utilizzati automaticamente dal sistema che provvede a fornire le distanze all'interno delle variabili di sistema distance, distanceRight e distanLeft. Il sistema provvede anche automaticamente a rinfrescare la posizione del servo. Per tali ragioni queste funzioni non devono essere utilizzate dal programmatore. Il programmatore deve far uso delle sole variabili di

[16] Alcune delle costanti di sistema possono essere cambiate solamente attraverso singoli comandi e non attraverso il Tab Setting.

[17] Il controllo è in realtà limitato ad un solo indirizzo ma se questo fallisce è bene cambiare la EEPROM.

[18] Si ricorda che il real time clock calendar possiede al suo interno alcuni registri che potrebbero essere letti per fini diagnostici.

sistema.

Qualora per ragioni meccaniche si volesse cambiare l'angolo di sguardo si deve opportunamente cambiare il valore intero nella funzione di delay interna alla funzione di controllo dei servi. Per calcolare tale valore bisogna far uso di questa equazione:

$$Delay = \frac{Degree \cdot 68}{90}$$

Dove Delay è il numero da passare alla funzione Delay100TCYx ( ). Si tenga conto che 90° è la posizione centrale, 65° è lo sguardo a sinistra e 115° è lo sguardo a destra. I valori 48 e 88 ottenuti per i rispettivi sguardi a destra e a sinistra sono valori approssimati per eccesso e difetto e piccola sfumatura ideologica.

**File motor.h**

All'interno del file motor.h sono presenti tutte le funzioni per il controllo del movimento. A differenza delle funzioni associate al Servo le funzioni associate ai motori sono concepite per poter essere utilizzate in modo da poter personalizzare il movimento del robot nella modalità Explorer o in modalità create dall'utente. Le funzioni Disponibili sono:

| Funzione | Descrizione |
|---|---|
| **void** Halt ( ) | Richiamando questa funzione vengono bloccati i motori |
| **void** Forward (**int** speed) | Richiamando questa funzione è possibile andare avanti. Il valore da passare determina la velocità di Domotino e deve essere compreso tra 0 e 1023. Si consideri però che un valore minimo consigliabile per il movimento è 700 e può variare in base a pesi aggiuntivi. I valori delle velocità possono essere memorizzati anche nella tabella di sistema speedTable. |
| **void** Backwards (**int** speed) | Richiamando questa funzione è possibile andare indietro. Il valore da passare determina la velocità di Domotino e deve essere compreso tra 0 e 1023. Si consideri però che un valore minimo consigliabile per il movimento è 700 e può variare in base a pesi aggiuntivi. I valori delle velocità possono essere memorizzati anche nella tabella di sistema speedTable. |
| **void** turnLeft (**int** speed) | Richiamando questa funzione è possibile girare a sinistra. Il valore da passare determina la velocità di Domotino e deve essere compreso tra 0 e 1023. Si consideri però che un valore minimo consigliabile per il movimento è 700 e può variare in base a pesi aggiuntivi. I valori delle velocità possono essere memorizzati anche nella tabella di sistema speedTable. Per ragioni legate all'inerzia del robot è consigliabile effetture le curve a bassa velocità evitando così di slittare. |
| **void** turnRight (**int** speed) | Richiamando questa funzione è possibile girare a destra. Il valore da passare determina la velocità di Domotino e deve essere compreso tra 0 e 1023. Si consideri però che un valore minimo consigliabile per il movimento è 700 e può variare in base a pesi aggiuntivi. I valori delle velocità possono essere memorizzati anche nella tabella di sistema speedTable. Per ragioni legate all'inerzia del robot è consigliabile effetture le curve a bassa velocità evitando così di slittare. |

**Tabella 3: Funzioni per il controllo del movimento**

Esempi di utilizzo di queste funzioni sono riportate nella spiegazione della modalità Explorer.

**File sensors.h**

All'interno di questo file sono contenute le funzioni associate ai sensori ausiliari di sistema, ovvero il sensore di luminosità e il sensore termico e il controllo dello stato della batteria. Tali funzioni non devono essere utilizzate dal programmatore poiché utilizzate dal sistema, il quale provvederà ad aggiornare automaticamente le variabili di sistema temperature, Night e batteryLevel. Nella prima è riportata la temperatura che deve essere compresa tra 0°C e 50°C, nella seconda è riportato lo stato notte giorno ovvero se Night vale 1 è notte mentre se vale 0 è giorno. Il livello per discriminare la notte dal giorno è contenuto nella EEPROM all'indirizzo 0x0071 (maggiori chiarimenti a riguardo saranno dati a breve). Nella variabile batteryLevel è contenuto il valore percentuale della carica della batteria. Al fine di ottenere informazioni sulla temperatura, luminosità o livello della batteria è bene far uso delle sole variabili di sistema. Nonostante ciò, visto l'interesse che tali funzioni possono suscitare le vedremo in maggior dettaglio. In particolare verranno spiegate le funzioni MeasureTemperature ( ) e MeasureBattery ( ), visto che richiedono un ritocco dell'hardware.

**MeasureTemperature ( )**

Come sensore di temperatura si è fatto uso del sensore LM35DZ. Per mantenere la compatibilità con Freedom è necessario rimuovere il resistore R10 da 4,7KΩ. Questo resistore di pull-up è invece necessario per il sensore digitale DS1820. Si fa presente che il sensore LM35DZ e DS1820 non sono tra loro compatibili ovvero la funzione MeasureTemperature ( ) non funziona con il sensore DS1820. Il sensore LM35DZ deve essere montato come riportato in Figura 20, ovvero con la mezza luna rivolta verso l'interno.

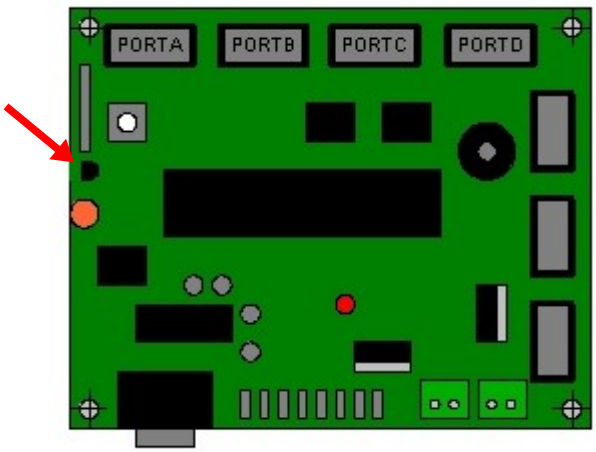



**Figura 20: Orientamento del sensore termico**

L'uscita del sensore LM35DZ è rappresentato dal pin centrale ed è collegata all'ingresso analogico AN4. L'uscita del sensore risulta pari a 10mV/°C ovvero aumenta di 10mV ogni grado centigrado. Dal momento che il convertitore analogico digitale interno al PIC è da 10bit e l'alimentazione del PIC è di 5V, si ha che il quanto[19] è di circa 5mV. Se la temperatura fosse di di 5°C si avrebbe una tensione in uscita al sensore di 50mV ovvero 10 quanti da 5mV. Questo significa che per ottenere il valore della temperatura con la precisione di 1°C basterà dividere per due il valore letto con l'ADC[20]. Vediamo con qualche altro dettaglio la funzione di seguito riportata. Come prima cosa si imposta ADCON1 per impostare tutti i pin della PORTA come

[19] Per ulteriori informazioni a riguardo si rimanda al Tutorial "Misure elettriche e tecniche di condizionamento del segnale".

[20] Si può vedere come in questo caso tutto quello che è descritto nel Tutorial "Misure elettriche e tecniche di condizionamento del segnale" non serve a nulla! Infatti la precisione che si riesce ad ottenere collegando direttamente il sensore termico in ingresso all'ADC è sufficiente ai nostri scopi pur sprecando molta della dinamica dell'ADC.

ingressi analogici (eccezion fatta per RA4). Come seconda cosa si imposta ADCON2 per i tempi di conversione e la giustificazione a sinistra del valore letto[21]. In ultimo si seleziona l'ingresso analogico AN4 sul quale effettuare la misura. La misura viene avviata solo successivamente attivando il bit GO di ACON0. Attivata la conversione si attende la fine della stessa per mezzo del ciclo `while( ADCON0bits.GO)` per mezzo del quale si attende che il bit GO ritorni al valore 0 segnalando cosi la fine conversione. Il valore della lettura avviene per mezzo della funzione ReadADC ( ). Il valore letto è sommato al valore della variabile data (che non sarà sfuggito essere statica). Ogni 8 chiamate della funzione MeasureTemperature ( ) viene calcolato il valore medio e aggiornato il valore temperature. La chiamata alla funzione MeasureTemperature ( ) avviene automaticamente grazie all'interrupt del TMR0.

```
void MeasureTemperature ()

{  static int data = 0;
   static unsigned char number = 0;

   ADCON1 = 0b00001010; //abilito ingressi analogici
   ADCON2 = 0b10111110; //imposto i tempi di conversione e giustificazione a destra
   ADCON0 = 0b00010001; //abilito l'ADC AN4

   ADCON0bits.GO = 1; // Avvio la conversione Analogico/Digitale
   while( ADCON0bits.GO); // Attendo la fine della conversione

   data = data + ReadADC(); // leggo il risultato della conversione

   number++;

   if (number==8)

   { data = data >> 3;   // con lo shift a destra divido per le letture
                         // fatte ottenendo il valore medio

       temperature = data >>1; //ho 10mV/°C e un quanto di 0,005mV quindi
                               // dividendo per 2 ho T
       data =0;
       number = 0;
   }
 }
```

La ragione per cui vengono fatte più letture serve solo per evitare fluttuazioni troppo rapide della variabile temperature. La media viene fatta su 8 letture in modo da poter fare la divisione semplicemente con uno shift[22] a destra di tre posizioni. Successivamente avviene un ulteriore schift di una posizione per effettuare la divisione per 2 in modo da ottenere la temperatura effettiva. Tale divisione si sarebbe naturalmente potuta effettuare con lo shift precedente. Per ragioni di chiarezza concettuale si è preferito separare i due shift.

**MeasureBattery ( )**

La funzione MeasureBattery ( ) risulta praticamente identica alla funzione MeasureTemperature ( ). Le uniche differenze consistono nel fatto che ADCON0 viene impostato per selezionare come ingresso analogico attivo AN1. Un'altra differenza sta nel fatto che il valore batteryLevel non viene ottenuto con una divisione per due ma sottraendo 796. Il risultato della sottrazione rappresenta il valore percentuale della carica della batteria. Quindi la batteria è al massimo se batteryLevel è 100% mentre è scarica se batteryLevel vale 0%. Vediamo da dove esce il numero 796 e cosa collegare all'ingresso AN4 (pin n. 7 del

[21] Per ulteriori informazioni in materia si rimanda al datasheet del PIC18F4580.
[22] Per ulteriori informazioni in materia si rimanda al Tutorial “C18 Step by Step”.

PIC18F4580). Come prima cosa si capisce che dal momento che la batteria avrà una tensione di circa 12V, mentre in ingresso al convertitore analogico digitale del PIC non posso mettere una tensione maggiore a quella dell'alimentazione del PIC stesso, dovrò ridurre in qualche modo il valore della tensione nel range 0V-5V[23].
La tecnica utilizzata fa uso di un semplice partitore di tensione 1:3 composta da tre resistori da 2,2KΩ come riportato in Figura 21.

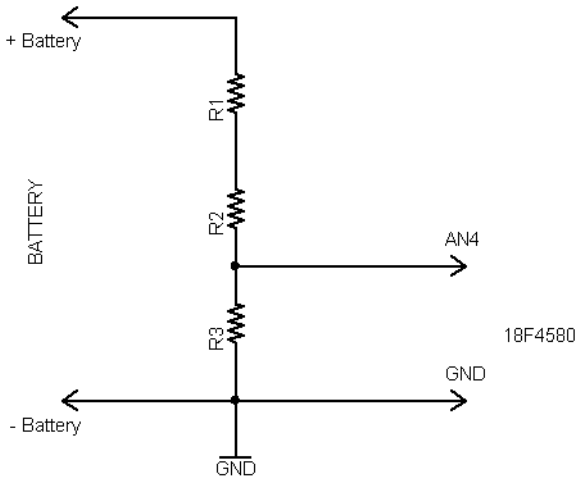



**Figura 21: Partitore 1:3 con resistori da 2,2KΩ**

Il valore della tensione della batteria diviso per tre viene applicato all'ingresso AN4. Si è scelto un divisore per 3 poiché il valore massimo in ingresso all'ADC è 5V che moltiplicato per 3 fa 15V. La batteria al piombo non avrà mai tensioni maggiori di 15V neanche nel momento di carica. Vediamo qualche altro dettaglio...
Le batterie al piombo con tensione nominale da 12V sono ottenute mettendo in serie celle elementari al piombo la cui tensione nominale è di 2,2V per cella. Questo significa che la tensione nominale della batteria al piombo da 12V è in realtà 13,2V. Il processo di carica della batteria avviene generalmente con tensioni pari a 13,8V. Il processo di carica delle celle elementari al piombo può avvenire in maniera reversibile sino a che il valore della cella elementare non scende al disotto di 1,8V ovvero la tensione dell'intera batteria non scenda al disotto di 10,8V[24]. Come livello limite sotto il quale la batteria è da considerarsi scarica si è scelto il valore 11,7V. A livelli inferiori Domotino può ancora funzionare ma si rischierebbe di raggiungere il punto di non ritorno della batteria. Vediamo ora in una tabella riassuntiva i valori appena descritti e i valori di tensione che si ottengono in ingresso al convertitore analogico digitale nonché il valore numerico con il quale l'ADC converte la tensione.

[23] Utilizzando la tecnica di condizionamento del segnale si potrebbe sottrarre 10V per poi analizzare solo le tensioni superiori a 10V, ma nel nostro caso si può semplificare il tutto.
[24] Per ulteriori informazioni in materia si rimanda al Tutorial “1000 domande 1000 risposte”.

| Livelli tensione | Partitore 1:3 | Valore ADC |
|---|---|---|
| 15,0V | 5.0V | 1024 |
| 13,8V | 4,6V | 939 |
| 13,2V | 4,4V | ~896[25] |
| 11,7V | 3,9V | 796 |
| 10,8V | 3,6V | 735 |

**Tabella 4: Tabella riassuntiva dei valori precedentemente citati**

Come è possibile vedere il valore 796 rappresenta il valore minimo della tensione ammissibile per la batteria mentre 896 è il valore massimo per una batteria carica. La differenza tra i due valori è proprio 100, dunque sottraendo 796 al valore letto si ottiene il valore percentuale dello stato della batteria. Si capisce dunque che il livello minimo 11,7V è stato scelto proprio per avere un calcolo della percentuale ottenibile con semplice sottrazione; 11,7V è però anche un valore ragionevole sotto il quale non bisogna scendere.

Dal momento che i valori in tabella sono anche oltre il range 11,7V-13,2V si capisce che il valore percentuale ottenibile può essere sia maggiore di 100 che inferiore di 0. Nella modalità modA per valori superiori a 100 si continua semplicemente a scrivere 100%. Questo è valido fino al livello 122% oltre il quale il sistema riconosce che la batteria è collegata all'alimentatore dunque scrive POW (Power) invece di 100%. Dalla Tabella 4 si capisce che il punto di non ritorno si ha per un livello di carica pari a -61%. Questo significa che in realtà si potrebbe scendere anche al disotto dello 0% avendo ancora la possibilità di ricaricare la batteria. In ogni modo si consiglia di ricaricare la batteria evitando di scendere al disotto dello 0%. Al fine di proteggere la batteria la modalità Explorer viene disattivata quando il livello della batteria scende al disotto del 30%.

A causa della semplicità del rilevatore di carica che si basa semplicemente sul livello di tensione può accadere che pur staccando l'alimentatore il display continui a visualizzare POW per qualche minuto. Se si fosse fatto un controllo sulla corrente di carica questo disagio non sarebbe presente. Il rilevatore di corrente di carica potrebbe essere ottenuto con un semplice diodo o resistore che venga attraversato dalla corrente di carica. Questo semplice circuito in aggiunta al precedente permetterebbe di risolvere il disagio ma richiederebbe un altro ingresso analogico...a voi il divertimento!

### File commands.h

Ogni volta che Domotino riceve un dato tramite la porta seriale lo memorizza in attesa che il numero di dati ricevuti sia 8. Otto byte compongono un comando ovvero un'istruzione da eseguire. Il comando può essere ottenuto inviando un carattere ASCII alla volta o direttamente una stringa di 8 caratteri ASCII. Gli 8 caratteri devono essere inviati entro un intervallo di circa due secondi dall'invio del primo. Se infatti dovessero passare più di due secondi il sistema considererà i byte precedenti come dei disturbi e li ignorerà. Per non dover impazzire ad inviare in velocità 8 caratteri ASCII si può far uso del programma Domotino.exe descritto successivamente. Nulla vieta l'utilizzo di altri terminali per l'invio dei comandi e la ricezione delle informazioni. Una volta inviato un comando bisogna attendere almeno 10ms prima di poter inviare un nuovo comando. Prima dell'esecuzione del comando Domotino rinvia il comando ricevuto[26].

La parte di programma associata alla ricezione degli 8 byte è scritta all'interno della funzione con cui sono gestiti gli interrupt, ovvero all'interno del file main.c. La funzione in cui vengono decodificati i comandi è invece all'interno del file commads.h. Si fa presente che ogni comando è composto da 8 byte, qualora un byte non sia utilizzato può assumere qualunque valore ma deve avere necessariamente

[25] In realtà il valore sarebbe 897 ma è approssimato a 896.

[26] Ricevendo il comando trasmesso è possibile implementare una semplice tecnica per rilevare errori in trasmissione.

un carattere in modo da raggiungere la lunghezza di 8 caratteri ovvero byte. I comandi che Domotino riconosce sono:

- Scrittura Orario di sistema
- Lettura Orario di sistema (con secondi)
- Scrittura Data di sistema
- Lettura Data di sistema
- Movimento in avanti
- Gira a destra
- Gira a sinistra
- Movimento retromarcia
- Blocco motori
- Controllo modalità attiva
- Avvio inizializzazione
- Cambio orario sveglia
- Scrittura EEPROM esterna
- Lettura EEPROM esterna
- Lettura Distanza
- Posizionamento servo
- Controllo MOS PORTE
- Lettura temperatura ambiente
- Lettura stato batteria
- Cambio Modalità

Qualora il comando ricevuto non sia riconosciuto tra quelli sopra elencati il sistema trasmette il messaggio Unknown Command, ovvero comando sconosciuto. I comandi riconosciuti dal sistema sono come detto composti da 8 caratteri ed in particolare il primo carattere contiene l'informazione relativa al tipo di comando[27] mentre il resto dei caratteri contengono l'eventuale informazione associata al comando stesso. Vediamo in dettaglio la sintassi dei singoli comandi:

- **Scrittura Orario di sistema**
  Per mezzo di questo comando è possibile aggiornare l'orario di sistema inclusi i secondi. La sintassi del comando è la seguente:

| **T** | H | H | M | M | S | S | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

dove :

H : rappresentano l'ora; deve valere 0 in caso di valore nullo
M : rappresentano i minuti; deve valere 0 in caso di valore nullo
S : rappresentano i secondi; deve valere 0 in caso di valore nullo

Es.
Per scrivere le 13:30 e 12 secondi il comando è: T1330120

- **Lettura Orario di sistema (con secondi)**
  Per mezzo di questo comando è possibile leggere direttamente l'orario di sistema. Il formato con cui viene trasmesso è HH:MM.SS. La sintassi del comando è la seguente:

[27] Il tipo di comando sarà riportato in grassetto per evitare confusione con il resto dei caratteri che assumono un significato diverso.

| t | X | X | X | X | X | X | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

Es.
Per richiedere l'ora di sistema il comando è: t0000000

Domotino risponderà con:

**Time: 11:53.59**

- **Scrittura Data di sistema**
  Per mezzo di questo comando è possibile aggiornare la data di sistema. La sintassi del comando è la seguente:

| D | D | D | M | M | Y | Y | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

dove :

D : rappresentano il giorno; deve valere 0 in caso di valore nullo
M : rappresentano il mese; deve valere 0 in caso di valore nullo
Y : rappresentano l'anno; deve valere 0 in caso di valore nullo

Es.
Per aggiornare la data al 1/1/2007 il comando è: D0101070

Domotino risponderà con:

**Date has been updated**

- **Lettura Data di sistema**
  Per mezzo di questo comando è possibile leggere direttamente la data di sistema. Il formato con cui viene trasmesso è DD/MM/YY. La sintassi del comando è la seguente:

| d | X | X | X | X | X | X | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

Es.
Per richiedere la data di sistema il comando è: d0000000

Domotino risponderà con:

**Date: 01/01/07**

- **Movimento in avanti**
  Per mezzo di questo comando è possibile muoversi in avanti ad una certa velocità. La sintassi del comando è la seguente:

| F | M | X | X | X | X | X | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

dove :

M : rappresenta la marcia tra 0 e 5[28]

Es.
Per andare in avanti al massimo della velocità il comando è: F5000000

Domotino risponderà con:

**Direction: Forward**
**Speed: 5**

- **Gira a destra**
  Per mezzo di questo comando è possibile girare a destra ad una certa velocità. La sintassi del comando è la seguente:

| **R** | M | X | X | X | X | X | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

dove :

M : rappresenta la marcia tra 0 e 5

Es.
Per girare a destra al minimo della velocità[29] il comando è: R1000000

Domotino risponderà con:

**Direction: Right**
**Speed: 1**

- **Gira a sinistra**
  Per mezzo di questo comando è possibile girare a sinistra ad una certa velocità. La sintassi del comando è la seguente:

| **L** | M | X | X | X | X | X | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

dove :

M : rappresenta la marcia tra 0 e 5

Es.
Per girare a sinistra al minimo della velocità il comando è: L1000000

Domotino risponderà con:

**Direction: Left**
**Speed: 1**

[28] Il valore delle velocità associate alle marce è memorizzato all'interno della tabella delle velocità. Alla marcia 0 la velocità è 0.
[29] Le manovre a destra e a sinistra si raccomanda di farle al minimo della velocità in maniera da evitare slittamenti.

- **Movimento retromarcia**
  Per mezzo di questo comando è possibile fare retromarcia ad una certa velocità. La sintassi del comando è la seguente:

| **B** | M | X | X | X | X | X | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

dove :

M : rappresenta la marcia tra 0 e 5

Es.
Per fare retromarcia a velocità intermedia il comando è: B3000000

Domotino risponderà con:

**Direction: Backward**
**Speed: 3**

- **Blocco motori**
  Per mezzo di questo comando è possibile arrestare i motori. La sintassi del comando è la seguente:

| **S** | X | X | X | X | X | X | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

Es.
Per bloccare i motori il comando è: S0000000

Domotino risponderà con:

**Domotino has been stopped**

- **Controllo modalità attiva**
  Per mezzo di questo comando è possibile richiedere la modalità che in è esecuzione. Il sistema trasmetterà come risposta un valore intero che corrisponde al numero della modalità attiva. La sintassi del comando è la seguente:

| **?** | X | X | X | X | X | X | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

Es.
Per richiedere la modalità il comando è: ?0000000

Domotino risponderà con:

**Modality: 1**

- **Avvio inizializzazione**

Per mezzo di questo comando è possibile richiedere una nuova inizializzazione delle periferiche di sistema.

La sintassi del comando è la seguente:

| **X** | X | X | X | X | X | X | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

Es.
Per richiedere l'inizializzazione del sistema il comando è: X0000000

Domotino risponderà con:

**System Ready...**

**System has been initialized**

- **Cambio orario sveglia**
  Per mezzo di questo comando è possibile impostare l'orario della sveglia. La sintassi del comando è la seguente:

| **W** | H | H | M | M | X | X | F |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

dove :

H : rappresentano l'ora; deve valere 0 in caso di valore nullo
M : rappresentano i minuti; deve valere 0 in caso di valore nullo
F : Flag che vale 1 per attivare la sveglia e 0 per disattivarla

Es.
Per impostare la sveglia alle 13:30 il comando è: W1330000

Domotino risponderà con:

**Alarm Time has been updated**

- **Scrittura EEPROM esterna**
  Per mezzo di questo comando è possibile scrivere su un qualunque indirizzo della memoria EEPROM esterna. Alcuni indirizzi non sono utilizzabili dal programmatore poiché usati dal sistema per memorizzare le impostazioni interne.

  L'organizzazione della memoria EEPROM è:

| Indirizzo | Contenuto |
|---|---|
| 0x0000 | Se diverso da 0x55 vuol dire che che il Firmware è al suo primo avvio. |
| 0x0001 | Se diverso da 0xAA vuol dire che che l'operatore non ha mai impostato le costanti interne al sistema. |
| 0x0002 | Contenuto variabile. Viene incrementato di 1 ad ogni Reset di sistema. Tale locazione è usata per controllare se la EEPROM funziona correttamente. |
| 0x0003... | Intervallo riservato per espansioni future. |
| ...0x0049 | Intervallo riservato per espansioni future. |
| 0x0050 | 0=OFF, 1=ON: Attiva il beep sulla modalità Lamp Service quando vengono accese o spente le luci. Il valore di Default al primo avvio è 1. |
| 0x0051 | 0=OFF, 1=ON: Attiva il beep quando si preme il pulsante per cambiare modalità. Il valore di Default al primo avvio è 1. |
| 0x0052 | 0=OFF, 1=ON: Attiva il beep quando avviene un urto. Il valore di Default al primo avvio è 1. |
| 0x0053 | 0=OFF, 1=ON: Attiva la modalità Explorer dopo la sveglia. Il valore di Default al primo avvio è 1. |
| 0x0054... | Intervallo riservato per espansioni Flag future. |
| ...0x006F | Intervallo riservato per espansioni Flag future. |
| 0x0070 | Contiene il valore della massima distanza misurabile dal sensore ad ultrasuoni. Il valore moltiplicato per due viene caricato nella variabile MAX_DISTANCE. Il valore di Default al primo avvio è 200 ovvero MAX_DISTANCE è pari a 400 cm. |
| 0x0071 | Contiene il valore con il quale il sistema discrimina la notte dal giorno. Il valore moltiplicato per due viene caricato nella variabile NIGHT_LEVEL. Il valore di Default al primo avvio è 175 ovvero NIGHT_LEVEL è pari a 350. |
| 0x0072 | Contiene il valore con il quale il sistema discrimina una variazione di distanza causata dal movimento di oggetti. Il valore viene caricato nella variabile DELTA_DISTANCE. Il valore di Default al primo avvio è 20 cm. |
| 0x0073 | Contiene il valore usato dal sistema per il l'isteresi del comparatore notte/giorno. Il valore viene caricato nella variabile LIGHT_FILTER. Il valore di Default al primo avvio è 50. |
| 0x0074 | Contiene il valore usato dal sistema per il numero di blocchi dopo il quale interpretare uno stallo. Il valore viene caricato nella variabile NUM_BLOCK. Il valore di Default al primo avvio è 8. |
| 0x0075 | Contiene il valore con il quale il sistema discrimina una variazione di distanza in caso di incastro con l'ambiente circostante. Il valore viene caricato nella variabile DELTA_BLOCK. Il valore di Default al primo avvio è 5 cm. |
| 0x0076 | Contiene il valore, espresso in minuti, del tempo di pulizia, ovvero del tempo per cui rimarrà attiva la modalità Explorer. Il valore di Default al primo avvio è 30 minuti. |
| 0x0077 | Contiene il valore, espresso in minuti, del tempo per cui rimarranno accese le luci nella modalità modF (Service Lamp BIS). Il valore di Default al primo avvio è 3 minuti. |
| 0x0078... | Intervallo riservato per espansioni future. |
| ...0x00FF | Intervallo riservato per espansioni future. |
| 0x0100 | Valore ora sveglia in formato BCD. |
| 0x0101 | Valore minuti sveglia in formato BCD. |

| Indirizzo | Contenuto |
|---|---|
| 0x0102 | 0=Sveglia OFF, 1=Sveglia ON . |
| 0x0103... | Intervallo riservato per espansioni future. |
| ...0xFFFE | Intervallo riservato per espansioni future. |
| 0xFFFF | Contenuto variabile. Viene incrementato di 1 ad ogni Reset di sistema. Tale locazione è usata per controllare se la dimensione della EEPROM è corretta. |

**Tabella 5: Mappatura della EEPROM esterna**

La sintassi del comando è la seguente:

| **E** | A3 | A2 | A1 | A0 | D1 | D0 | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

dove :

Ax : rappresentano il valore dell'indirizzo in esadecimale con A0 la parte meno significativa
Dx : rappresentano il valore del dato in esadecimale con D0 la parte meno significativa

Es.
Per scrivere il dato 0x03 all'indirizzo 0x0100 il comando è: E0100030

Domotino risponderà con:

**EEPROM has been written**

- **Lettura EEPROM esterna**
Per mezzo di questo comando è possibile leggere un qualunque indirizzo della memoria EEPROM esterna. La sintassi del comando è la seguente:

| **C** | A3 | A2 | A1 | A0 | X | X | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

dove :

Ax : rappresentano il valore dell'indirizzo in esadecimale con A0 la parte meno significativa

Es.
Per leggere l'indirizzo 0x03AC il comando è: C03AC000
Domotino risponderà con:

**Datum: 0xFF**

- **Lettura Distanza**
Per mezzo di questo comando è possibile leggere la distanza in centimetri degli oggetti posizionati di fronte a Domotino. Questo comando può essere usato solo quando la modalità Remote Control è attiva. La sintassi del comando è la seguente:

| **f** | X | X | X | X | X | X | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

Es.
Per richiedere la distanza il comando è: f0000000

Domotino risponderà con:

**Distance : 23 cm**

- **Posizionamento Servo**
  Per mezzo di questo comando è possibile posizionare il servo, ovvero il sensore ad ultrasuoni in modo da effettuare una misura a destra, a sinistra o al centro. Questo comando dovrebbe essere usato solo quando la modalità Remote Control è attiva. La sintassi del comando è la seguente:

| **s** | P | X | X | X | X | X | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

dove :

P : rappresenta la posizione che deve raggiungere il servo
I valori possibili sono:
0: Disattiva il servo e il rinfresco della posizione
1: Posizionamento a Sinistra
2: Posizionamento Frontale
3: Posizionamento a Destra

Es.
Per posizionare il servo a sinistra il comando è : s1000000

Domotino risponderà con:

**Ultrasonic Sensor : Left**

- **Controllo MOS PORTE**
  Per mezzo di questo comando è possibile controllare il valore assunto dalla PORTE ovvero della scheda con MOS. Si ricorda che la PORTE ha solo tre bit disponibili. Questo comando dovrebbe essere usato solo quando la modalità Remote Control è attiva. La sintassi del comando è la seguente:

| **P** | E2 | E1 | E0 | X | X | X | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

dove :
E2 : Rappresenta il bit PORTEbits.RE2 e può assumere il valore 0 o 1
E1 : Rappresenta il bit PORTEbits.RE1 e può assumere il valore 0 o 1
E0 : Rappresenta il bit PORTEbits.RE0 e può assumere il valore 0 o 1

Es.
Per accendere il cicalino il comando è : P0100000

Domotino risponderà con:

**PORTE has been updated**

- **Lettura temperatura ambiente**
  Per mezzo di questo comando è possibile leggere la temperatura dell'ambiente in cui si trova Domotino. Il valore in uscita rappresenta il valore della variabile globale temperature. La sintassi del comando è la seguente:

| **w** | X | X | X | X | X | X | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

Es.
Per richiedere la temperatura il comando è: w0000000

Domotino risponderà con:

**Temperature : 22 'C**

- **Lettura stato batteria**
  Per mezzo di questo comando è possibile leggere lo stato della batteria di Domotino espresso in percentuale. Il valore in uscita rappresenta il valore della variabile globale batteryLevel. La sintassi del comando è la seguente:

| **l** | X | X | X | X | X | X | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

Es.
Per richiedere la temperatura il comando è: l0000000

Domotino risponderà con:

**Battery Level : 79%**

- **Cambio Modalità**
  Per mezzo di questo comando è possibile cambiare la modalità attiva di Domotino. La sintassi del comando è la seguente:

| **M** | m | X | X | X | X | X | X |
|---|---|---|---|---|---|---|---|

X può assumere qualunque valore diverso dal carattere nullo

dove:
m : rappresenta il numero della modalità da attivare

Es.
Per attivare la modalità Remote Control il comando è: M6000000

Domotino risponderà con:

**Modality has been updated**

**File var.h**

Nel file var.h sono contenute tutte le variabili e costanti utilizzate dal sistema. Le costanti si dividono in non modificabili e modificabili mentre le variabili si dividono in non modificabili e utilizzabili. Non modificabile vuol dire che se ne sconsiglia di modificarle qualora non si abbia una buona conoscenza del sistema e una buona ragione. Le costanti modificabili sono invece quelle costanti che possono essere variate per personalizzare il sistema. Una volta che si cambia il valore di una costante, affinché possa avere effetto è necessario ricompilare il sorgente e reinstallare il firmware all'interno del PIC.

Le variabili utilizzabili sono quelle variabili speciali i cui valori sono aggiornati automaticamente dal sistema e dunque non bisogna preoccuparsi di aggiornarle. Grazie a queste variabili è possibile avere informazioni dell'ambiente esterno e decidere come il sistema deve reagire.

Queste variabili risultano particolarmente utili qualora si personalizzi l'algoritmo del movimento o si crei una modalità personale. Vediamo un riepilogo delle costanti modificabili e variabili utilizzabili[30].

| Costante | Utilizzo |
|---|---|
| #define NUM_MOD 7 | Per mezzo NUM_MOD viene definito il numero di modalità di funzionamento del sistema. Attualmente pari a 5. |
| #define MODA 1 | Definisco l'ordine con cui vengono attivate le varie modalità di funzionamento alla pressione del tasto Function Button. MODA sarà la prima funzione ad essere richiamata, MODB la seconda, MODC la terza, MODD la settima, MODE la quarta, MODF la quinta e MODG la sesta. |
| #define MODB 2 | |
| #define MODC 3 | |
| #define MODD 7 | |
| #define MODE 4 | |
| #define MODF 5 | |
| #define MODG 6 | |

**Tabella 6: Costanti di sistema modificabili**

[30] Il significato delle altre costanti e variabili è contenuto nei commenti del codice sorgente. In questo documento non vengono riportate in modo da scoraggiarne il loro utilizzo.

| Variabile | Utilizzo |
|---|---|
| unsigned char *Time | Questa variabile viene sempre aggiornata dal sistema con l'orario di sistema. |
| unsigned char *Date | Questa variabile viene sempre aggiornata dal sistema con la data di sistema. |
| int speedTable[6] | Questo array contiene il valore delle velocità alle varie marce. I valori di default vengono caricati all'interno dell'array durante la fase d'inizializzazione del sistema ovvero per mezzo della funzione initialize ( ) contenuta nel file init.h. I valori di default sono:<br><br>speedTable[0] = 0; //tabella della velocità<br>speedTable[1] = 700;<br>speedTable[2] = 750;<br>speedTable[3] = 800;<br>speedTable[4] = 850;<br>speedTable[5] = 1023; |
| int distance = 0 | Questa variabile contiene la distanza degli oggetti frontali. Viene aggiornata dal sistema solo se la modalità attiva é MODB, MODC o MODE[31]. |
| int distanceLeft = 0 | Questa variabile contiene la distanza degli oggetti a sinistra. Viene aggiornata dal sistema solo se la modalità attiva é MODB ovvero la modalità Explorer. |
| int distanceRight = 0 | Questa variabile contiene la distanza degli oggetti a destra. Viene aggiornata dal sistema solo se la modalità attiva é MODB ovvero la modalità Explorer. |
| unsigned char AlarmFlag = 0 | Questa variabile è un Flag e può assumere il valore 1 o 0. Quando vale 1 vuol dire che nella MODC si è verificato un allarme intrusi. |
| unsigned char timeAlarm = 0 | Questa variabile è un Flag e può assumere il valore 1 o 0. Quando vale 1 vuol dire che si è verificato l'allarme sveglia. |
| unsigned char Night = 0 | Questa variabile è un Flag e può assumere il valore 1 o 0. Quando vale 1 vuol dire che è notte. |
| int temperature = 0 | Questa variabile contiene il valore della temperatura dell'ambiente. Può essere letta da qualunque modalità e il suo valore è automaticamente aggiornato dal sistema. |
| int batteryLevel = 0 | Questa variabile contiene il valore dello stato della batteria espresso in percentuale. Può assumere valori negativi e superiori a 100, il programmatore si deve preoccupare di interpretare valori negativi o superiori a 100. Può essere letta da qualunque modalità e il suo valore è automaticamente aggiornato dal sistema. |
| unsigned char LeftCrash =0 | Questa variabile è un Flag e può assumere il valore 1 o 0. Quando vale 1 vuol dire che si è avuto un urto a sinistra. Dopo il suo utilizzo deve essere posto a 0 dal programma. |
| unsigned char RightCrash =0 | Questa variabile è un Flag e può assumere il valore 1 o 0. Quando vale 1 vuol dire che si è avuto un urto a destra. Dopo il suo utilizzo deve essere posto a 0 dal programma. |
| unsigned char LeftRoadBlock = 0 | Questa variabile è un Flag e può assumere il valore 1 o 0. Quando vale 1 vuol dire che si è avuto una predizione di stallo a sinistra. Dopo il suo utilizzo deve essere posto a 0 dal programma. |

[31] Il lampeggio del led sul sensore ad ultrasuoni è attivo in tutte le modalità poiché è un buon indice di salute del sistema stesso. Il suo lampeggio indica infatti che le interruzioni di sistema stanno lavorando correttamente, almeno per TMR0. Qualora non si abbia più il lampeggio vuol dire che il sistema non è più in grado di operare correttamente ed è necessario premere il pulsante di Reset di Freedom. Il lampeggio del led viene frequentemente utilizzato in sistemi elettronici complessi per indicare il battito “cardiaco del circuito”.

| Variabile | Utilizzo |
|---|---|
| unsigned char RightRoadBlock = 0 | Questa variabile è un Flag e può assumere il valore 1 o 0. Quando vale 1 vuol dire che si è avuto una predizione di stallo a destra. Dopo il suo utilizzo deve essere posto a 0 dal programma. |
| unsigned char FrontRoadBlock = 0 | Questa variabile è un Flag e può assumere il valore 1 o 0. Quando vale 1 vuol dire che si è avuto una predizione di stallo frontale. Dopo un suo utilizzo deve essere posto a 0 dal programma. |
| unsigned char leftCounter = 0 | Questa variabile conta il numero di volte che si è verificata una lettura di distanza uguale a sinistra. Deve essere posta a 0 dall'utente ogni volta che la rispettiva variabile xxxxRoadBlock viene posta a 0 dal programma. |
| unsigned char rightCounter = 0 | Questa variabile conta il numero di volte che si è verificata una lettura di distanza uguale a destra. Deve essere posta a 0 dall'utente ogni volta che la rispettiva variabile xxxxRoadBlock viene posta a 0 dal programma. |
| unsigned char frontCounter = 0 | Questa variabile conta il numero di volte che si è verificata una lettura di distanza uguale frontalmente. Deve essere posta a 0 dall'utente ogni volta che la rispettiva variabile xxxxRoadBlock viene posta a 0 dal programma. |
| int MAX_DISTANCE 400 | Per mezzo di questa costante si definisce la massima distanza in cm a cui può operare il sensore ad ultrasuoni e dopo la quale la misura perde di significato. Per il sensore SRF5 è stata impostata a 400cm ovvero 4m.Il contenuto della variabile viene caricato dalla EEPROM all'avvio del sistema. |
| unsigned char DELTA_DISTANCE | Per mezzo di questa costante è possibile impostare la variazione di distanza in cm, dopo la quale bisogna far attivare l'allarme MODC o accendere/spegnere le luci nella MODE. Il contenuto della variabile viene caricato dalla EEPROM all'avvio del sistema. |
| unsigned char NIGHT_LEVEL | Per mezzo di questa costante è possibile determinare il livello luminoso dopo il quale il sistema considererà l'ambiente circostante un ambiente notturno. Questo valore può dipendere dalla posizione del sensore luminoso (fotoresistenza). Valori piccoli corrispondono a “notti scure”. Il contenuto della variabile viene caricato dalla EEPROM all'avvio del sistema. |
| unsigned char LIGHT_FILTER | Per mezzo di questa variabile si crea un piccolo ciclo d'isteresi ovvero uno pseudo trigger di Smith. Questo permette di evitare che raggiunto il punto in cui il sistema riconosce la notte si possa avere un'instabilità della variabile Night legata a piccole perturbazioni luminose. Il contenuto della variabile viene caricato dalla EEPROM all'avvio del sistema. |
| unsigned char NUM_BLOCK | Per mezzo di questa costante si definisce il numero di volte dopo le quali il sistema, riscontrando misure di distanza uguali, o a destra o a sinistra o frontalmente, sospetta un eventuale stallo del robot stesso. Con un numero pari a 8 si è riscontrato una buona predizione di stallo. Valori troppo piccoli possono causare predizioni di stallo errato in caso il robot stia camminando parallelamente a qualche oggetto. Valori troppo grandi possono determinare la mancata predizione dello stallo poiché anche durante uno stallo ci possono essere delle variazioni di distanza tali per cui il sistema possa ripensare di essere libero. La predizione dello stallo verrà spiegata in maggior dettaglio della spiegazione della modalità MODB. Il contenuto della variabile viene caricato dalla EEPROM all'avvio del sistema. |
| int DELTA_BLOCK 5 | Per mezzo di questa costante si definisce la variazione minima della distanza, espressa in cm, con cui interpretare un eventuale stallo. Il contenuto della variabile viene caricato dalla EEPROM all'avvio del sistema. |

**Tabella 7: Variabili di sistema utilizzabili**

**File mod.h**

In questo file sono contenute tutte le funzioni delle modalità del sistema. Le varie modalità vengono attivate alternativamente per mezzo del pulsante Function Button. Le modalità presenti nel sistema sono:

- modA ( ) : schermata principale con visualizzazione ora, data e temperatura[32].
- ModB ( ) : modalità Explorer
- modC ( ) : modalità Alarm (cane da guardia)
- modD ( ) : schermata in cui viene visualizzato l'orario della sveglia e se questa è attiva o meno
- modE ( ) : modalità Lamp Service (luce di cortesia)
- modF ( ) : modalità Lamp Service BIS (luce a tempo)
- modG ( ) : modalità Remote Control

Il passaggio da una modalità all'altra avviene per mezzo della variabile modality che contiene il numero della modalità che deve essere attiva. All'interno di ogni modalità è presente un controllo della modalità che dovrebbe essere attiva. Questo controllo avviene controllando se il valore associato alla modalità per mezzo della costante MODx corrisponde alla modalità attiva. Per esempio nella funzione modA si avrà il seguente controllo:

```
if (modality != MODA)
            break;
```

Questo controllo permette di uscire dalla funzione modA rilasciando spazio allo stack...la cosa si fa complessa! Lo stack rappresenta una memoria speciale interna al PIC per mezzo del quale è possibile memorizzare, ogni volta che viene chiamata una funzione il valore del Program Counter[33] (ovvero il punto di chiamata della funzione) dove bisogna ritornare[34]. Il PIC18F4580 possiede 31 livelli di stack quindi sarà possibile svolgere un numero di funzioni annidate per un massimo di 31.

Supponiamo che il controllo precedente non fosse presente e premendo il pulsante si passasse alla modB. In questo caso nello stack rimarrebbe il valore di ritorno per la funzione modA ( ). Passando ancora dalla modB ( ) alla successiva nello stack rimarrebbe il valore di ritorno per la modB. Cambiando sempre modalità si andrebbero sempre ad accumulare valori di ritorno nello stack senza rilasciare lo stack, questo raggiunto, il livello massimo causerà il blocco del PIC o comunque comportamenti anomali[35]. Per evitare questo problema la chiamata ad ogni modalità è effettuata dalla funzione main ( ) per mezzo delle seguenti istruzioni:

```
while (1)       //ciclo infinito in attesa d'interrupt
   {
      if (modality = = MODA)
            modA ( );

      if (modality = = MODB)
            modB ( );

      if (modality = = MODC)
            modC ( );
```

[32] A causa della mancanza della libreria del sensore termico la temperatura visualizzata non è significativa e fissa sempre a 24°C.
[33] Il Program Counter o PC è un registro particolare che punta il valore di memoria che bisogna eseguire.
[34] A seconda dei modelli di microcontrollori oltre al PC possono essere salvati altri registri "vitali".
[35] Quando lo stack si esaurisce il PIC ricomincia il suo utilizzo dall'inizio rischiando di sovrascrivere a dati di ritorno utili.

```
        if (modality = = MODD)
                modD ( );

        if (modality = = MODE)
                modE ( );

        if (modality = = MODF)
                modF ( );

        if (modality = = MODG)
                modG ( );
}
```

Ogni volta che si preme il pulsante per il cambio della modalità, la modalità in esecuzione terminerà quello che stava facendo e nel momento in cui effettuerà il controllo della modalità attiva e vedrà che la modalità attiva non è più lei eseguirà l'istruzione break con cui uscirà dalla funzione rilasciando lo stack. Il valore di ritorno di tutte le modalità è il ciclo while sopra scritto dal momento che le funzioni modalità sono da qui richiamate. All'interno del ciclo while si controlla la modalità che deve esser richiamata confrontando quella attiva con il valore associato alle varie modalità per mezzo delle costanti:

```
#define MODA 1
#define MODB 2
#define MODC 3
#define MODD 7
#define MODE 4
#define MODF 5
#define MODG 6
```

Si capisce dunque che l'ordine non sarà necessariamente modA, modB, modC, modD, modE, modF, modG. Una volta che la variabile modality raggiunge un numero maggiore del numero di modalità presenti assume il valore 1 ovvero verrà richiamata la prima modalità. Vediamo ora in maggior dettaglio le caratteristiche principali delle varie modalità:

- **modA ( )**
  In questa modalità vengono visualizzate in forma di stringa le variabili di sistema dell'orario della data, della temperatura e dello stato della batteria. Questa modalità può essere considerata la modalità base con la quale si consiglia di avviare il sistema.

- **modB ( )**
  Quando viene richiamata questa modalità il sistema visualizza la scritta Explorer sul display LCD. La modalità diviene operativa dopo tre beep distanti tra loro circa 2 secondi. Dopo ogni beep viene controllata la modalità attiva in modo da permettere all'utente di cambiare la stessa anche durante i beep stessi. Questa modalità è concepita per esplorare l'ambiente circostante e poterlo pulire. L'abilità nell'esplorare un ambiente viene a dipendere direttamente dall'algoritmo di movimento che viene implementato nella funzione motionControl ( ) presente nel file motion.h. Quando è attiva questa modalità sono disponibili tutte le variabili sulle distanze e i Flag relativi ad eventuali stalli. Oltre a queste particolari variabili sono presenti tutte le altre precedentemente descritte in Tabella 7.
    Dopo l'inizializzazione di alcune variabili viene effettuata la misura delle distanze[36]

[36] Le distanze sono misurate allo stesso modo presentato nell'articolo “Metro ad ultrasuoni”.

posizionando in maniera opportuna il servo. Ad ogni misura viene confrontato il valore della nuova misura con il valore precedente per controllare e predire un eventuale stallo. Se la variazione della distanza è minore di DELTA_BLOCK[37] il sistema incrementerà il conteggio delle volte in cui la misura della distanza in una certa direzione non è variata. Se la distanza in una certa direzione dovesse essere maggiore di DELTA_BLOCK il relativo conteggio dello stallo verrebbe azzerato. Alla fine di ogni ciclo di misura, ovvero ogni volta che si aggiornano le variabili sulle distanze, viene richiamata la funzione motionControl ( ) per mezzo della quale viene fisicamente scelto il movimento. Grazie a questa separazione il movimento e la modalità sono due entità differenti. Per bloccare il Robot basta cambiare modalità.

Al fine di un risparmio energetico e al tempo stesso poter attivare Domotino in nostra assenza, la modalità Explorer rimane attiva per un tempo prefissato. Il tempo, espresso in minuti, è scritto in EEPROM all'indirizzo 0x76. Il controllo del tempo che passa è fatto memorizzando l'orario d'inizio pulizia e controllando l'orario attuale. La differenza in minuti tra i due orari è calcolata per mezzo della funzione deltaTime ( ); il tempo rimanente per la pulizia è inoltre visualizzato sul display LCD. Per evitare che il tempo di pulizia faccia consumare la batteria oltre un prefissato livello ritenuto di rischio, è presente anche un controllo sullo stato di carica della batteria stessa. La modalità Explorer viene disattivata, con il passaggio alla modA, sia quando il tempo di pulizia è terminato, sia quando il livello di carica è inferiore al 30%.

- **modC ( )**
  Questa modalità rappresenta il cane da guardia, dal momento che fa suonare il cicalino nel caso in cui venga rilevato un oggetto in avvicinamento o in allontanamento nel raggio di azione del sensore. Questa modalità entra in funzione dopo tre beep in modo da dare il tempo di allontanarsi dal Robot una volta attivato l'allarme. Dopo i tre beep viene misurata la distanza frontale e verrà memorizzata come riferimento rispetto al quale determinare se è presente un movimento di oggetti[38]. Qualora si dovesse attivare l'allarme per poterlo disattivare è necessario cambiare modalità.

- **modD ( )**
  Per mezzo di questa modalità viene semplicemente visualizzato l'orario della sveglia e se questa è attiva o meno. Per poter cambiare l'orario della sveglia è necessario l'utilizzo del computer per mezzo del quale bisognerà inviare gli opportuni comandi per aggiornare gli indirizzi della EEPROM associati alla sveglia. Per spegnere la sveglia bisogna cambiare modalità. Nel caso della modalità modE è possibile spegnere il cicalino facendo rilevare al sistema del movimento volontario e cosciente.

- **modE ( )**
  Per mezzo di questa modalità si attiva il Lamp Service ovvero la luce di cortesia. Questa modalità è molto simile alla modalità Alarm ma controlla solo eventuali oggetti in avvicinamento. Quando viene rilevato un movimento in avvicinamento vengono accese le lampade o led. Dopo tre secondi[39] se dovesse essere rilevato un ulteriore movimento le lampade vengono spente. Questa modalità risulta molto utile se Domotino viene posto non lontano dal letto. In questo modo è infatti possibile accendere le luci del sistema semplicemente mettendo la mano fuori dal letto o ogni volta che si scende dal letto stesso. In questo modo si evita di dover cercare il pulsante della lampada agevolando persone con eventuali problemi motori. Qualora la sveglia risultasse attiva è possibile spegnerla cambiando lo stato delle lampade

[37] Si ricorda che la costante DELTA_BLOCK è situata in EEPROM all'indirizzo 0x75.
[38] Per i più esperti, facendo riferimento alla modalità Explorer è possibile modificare la modalità Alarm facendo muovere anche il sensore. Per ragioni acustiche ho personalmente evitato questa opzione.
[39] Durante questi tre secondi non vengono rilevati i movimenti. In un certo qual modo questa pausa è un filtro antirimbalzo.

ovvero facendo rilevare del movimento al sistema.
Allo spegnimento della sveglia è possibile far attivare automaticamente la modalità Explorer in modo da iniziare le pulizie del mattino. La modalità Explorer verrà attivata se il flag contenuto nella EEPROM all'indirizzo 0x53 vale 1 mentre se questo flag vale 0 la modalità non verrà cambiata.

- **modF ( )**
  Questa modalità nominata "Lamp Service BIS" rappresenta la copia della modE con l'unica differenza che le luci una volta attivate si spegneranno automaticamente dopo un tempo prefissato. Questo valore può essere scelto dall'utilizzatore scrivendo in esadecimale il valore in minuti per cui vuole che le luci rimangano accese. Tale valore è scritto nella EEPROM all'indirizzo 0x77. Anche in questa modalità, allo spegnimento dell'allarme, viene attivata automaticamente la modalità Explorer. L'attivazione o meno della modalità Explorer può essere impostata dall'utente impostando il flag contenuto nella EEPROM all'indirizzo 0x53. Quando il flag vale 1 verrà attivata la modalità Explorer mentre se vale 0 la modalità non verrà cambiata.

- **modG ( )**
  Per mezzo di questa modalità si attiva il servizio Remote Control. Questa modalità rappresenta in realtà una modalità fittizia per mezzo della quale si agevola il controllo di Domotino per mezzo di un PC. In questa modalità viene offerto l' unico servizio della lettura della distanza. E' compito dell'operatore posizionare il sensore nella direzione che vuole misurare. In questa modalità dovrebbero essere utilizzati tutti i comandi di telecontrollo di Domotino.

**File motion.h**

Nel file motion.h è presente la parte di programma che implementa l'algoritmo per il movimento del Robot. Tutto quanto fin ora spiegato gira in parte attorno a questa funzione dando la possibilità in un solo punto di poter accedere all'intero sistema permettendo la personalizzazione del robot senza eccessivo sforzo. Si ricorda che il firmware scaricabile dal sito www.LaurTec.com è già funzionante e non necessita alcuna ricompilazione se si fa utilizzo della configurazione illustrata in questo articolo. Qualora si volesse personalizzare il file motion.h è necessario ricompilare il sorgente e reinstallare il programma all'interno del PIC18F4580.

Alla luce di quanto esposto non dovrebbe risultare particolarmente difficile la comprensione dell'algoritmo associato al movimento.
La funzione motionControl viene come detto richiamata dalla modB ogni volta che si finisce di aggiornare la lettura delle distanze nei vari punti d'interesse. Al suo interno è definita una sola variabile mySpeed per mezzo della quale si controlla la marcia che bisogna utilizzare nel controllo dei motori, ovvero la velocità.

La marcia viene impostata in funzione della distanza degli oggetti rilevati[40]. Più gli oggetti sono vicini minore sarà la marcia utilizzabile. La parte di codice d'interesse è il seguente:

```
//in funzione della distanza decido la velocità

if (distance > 100 && distanceLeft > 100 && distanceRight > 100)
     mySpeed = 4;
     else
     if (distance > 80 && distanceLeft > 80 && distanceRight >80 )
          mySpeed = 3;
     else
```

[40] Si sconsiglia l'utilizzo della velocità massima quando il Robot è in modalità Explorer, in modo da evitare eventuali danni derivanti da urti.

```
      if (distance > 30 && distanceLeft > 30 && distanceRight> 30 )
            mySpeed = 2;
      else
            mySpeed = 1;
```

Impostata la marcia si controlla se il Robot si è bloccato da qualche parte. Per fare questo un modo potrebbe essere controllare la somma dei flag che indica lo stallo in una certa direzione del robot come fatto nel seguente segmento di codice:

```
//Controllo se il Robot si è incastrato

if ((LeftRoadBlock+FrontRoadBlock+RightRoadBlock)>1)
{
  Halt ();
  Backwards(speedTable[mySpeed]);
  Delay (30);
  turnRight (speedTable[mySpeed]);
  Delay (20);

  LeftRoadBlock = 0; //azzero i flag che segnalano l'avvenuto incastro
  FrontRoadBlock = 0;
  RightRoadBlock = 0;

  leftCounter = 0; // contatori per verificare da quanto le distanze
non variano
  rightCounter = 0;
  frontCounter = 0;
}
```

Se la somma è maggiore di 1 vuol dire che un possibile stallo è su almeno due direzioni. In questo caso il Robot si blocca evitando di perseverare ad avanzare in situazione di stallo, per poi indietreggiare alla velocità impostata dalla tabella delle velocità per mezzo della marcia precedentemente scelta. Questo avviene per tre secondi dopodiché gira a destra per due secondi[41]. Alla fine delle operazioni per sbloccare il Robot è necessario resettare i Flag xxxxRoadBlock e i Flag xxxxCounter in modo da rilevare ulteriori stalli.

Si fa notare che con questo semplice algoritmo di predizione di stallo si può erroneamente considerare il robot in stallo se questo sta viaggiando parallelamente a due ostacoli posto uno a destra e uno a sinistra. In ogni caso si consiglia di non considerare il Robot in stallo solo se tutti e tre i flag di stallo sono ad 1, infatti durante uno stallo a causa del movimento del robot che tenta di avanzare in ogni modo in una direzione si potrebbe comunque avere una variazione di distanza sufficiente da non avere più lo stallo in tutte e tre le direzioni, pur essendo presente un vero stallo.

Dopo il controllo di stallo è bene controllare se si è urtato qualche oggetto o a destra o a sinistra come fatto nel seguente segmento di codice:

```
//Controllo se ho urtato qualche ostacolo a sinistra

if (LeftCrash)
{
  Backwards(speedTable[mySpeed]);
```

[41] Da quanto detto si capisce che non è presente alcuno controllo sulla distanza effettuata ma ci si sposta solo per un certo intervallo di tempo. Da questo discende che lo spazio percorso verrà a dipendere dallo stato di carica della batteria.

```
  Delay (20);
  turnRight (speedTable[mySpeed]);
  Delay (10);
  LeftCrash = 0;
}


//Controllo se ho urtato qualche ostacolo a destra

if (RightCrash)
{
  Backwards(speedTable[mySpeed]);
  Delay (20);
  turnLeft (speedTable[mySpeed]);
  Delay (10);
  RightCrash = 0;
}
```

Per fare questo controllo si controlla il valore delle variabili LeftCrash e RightCrash. Quando una delle due è uguale a 1 segnala l'avvenuto urto o a destra o a sinistra. Il sistema quando rileva un urto provvede automaticamente a bloccare i motori lasciando all'utente la scelta di come comportarsi dopo l'urto. Nei segmenti di codice scritti sopra si può vedere che la scelta consiste nell'indietreggiare per poi girare o a destra o a sinistra a seconda che l'ostacolo sia stato rispettivamente rilevato a sinistra o a destra. Alla fine del controllo i Flag LeftCrash e RightCrash vengono azzerati in modo da poter rilevare ulteriori urti.

Naturalmente gli ostacoli possono essere rilevati anche per mezzo degli ultrasuoni senza doverci per forza andare a sbattere! Ci sono altezze particolari e distanze tali per cui gli ultrasuoni non riescono a vedere alcuni oggetti. L'algoritmo per controllare gli ultrasuoni è molto semplice dunque facilmente migliorabile.

```
//Controllo eventuali ostacoli rilevati con gli ultrasuoni

if (distance < 12)
  turnRight (speedTable[mySpeed]);
  else
  if (distanceLeft < 15)
     turnRight (speedTable[mySpeed]);
  else
     if (distanceRight < 15)
          turnLeft (speedTable[mySpeed]);
     else
          Forward (speedTable[mySpeed]);
```

Se la distanza di un oggetto frontale è inferiore a 12cm il Robot gira a destra con la marcia mySpeed. Se la distanza a sinistra è minore di 15cm il Robot gira a destra. Se la distanza a destra è inferiore a 15cm il Robot gira a sinistra. Queste manovre vengono ripetute finché le distanze non sono maggiori di quelle citate, nel cui caso il Robot va semplicemente avanti. Oggetti che possono non essere rilevati dagli ultrasuoni sono mobili dell'altezza del sensore, raggi di ruote di bicicletta e tende a rete, in questi casi si ha generalmente uno stato di stallo del Robot che viene rilevato come precedentemente descritto. Per migliorare la "sensibilità" del Robot è possibile mettere dei baffi anche sul piano superiore...ovvero delle sopracciglie!

Durante il movimento del Robot può ritornare utile essere facilmente visibili in caso sia buio. Per fare questo si controlla semplicemente il Flag Night, se vale 1 si accendono le luci di sistema altrimenti si spengono. Questo controllo come gli altri viene eseguito in maniera continua durante l'esecuzione della modB.

```
//Controllo se accendere i LED

if (Night ==1)
  PORTE = 0x01;
 else
  PORTE = 0x00;
```

## Creare una propria modalità

Oltre a poter modificare facilmente l'algoritmo per il movimento è possibile integrare con altrettanta facilità modalità all'interno del sistema. Supponiamo che le modalità siano 5 e vogliamo aggiungerne una sesta che faccia ciò che volete. Ecco i passi:

- Come primo passo bisogna modificare il file var.h contenente le costanti per le modalità. Le costanti da variare sono :

  **#define** NUM_MOD 5

  che deve essere variata in 6 poiché avremo 6 modalità. Poi bisogna variare le costanti relative all'ordine di visualizzazione:

  **#define** MODA 1
  **#define** MODB 2
  **#define** MODC 3
  **#define** MODD 5
  **#define** MODE 4

  Per ragioni di simmetria chiameremo la nuova modalità MODF ma manterremo MODD l'ultima modalità da visualizzare. Si avrà dunque:

  **#define** MODA 1
  **#define** MODB 2
  **#define** MODC 3
  **#define** MODD 6
  **#define** MODE 4
  **#define** MODF 5

- Come secondo passo bisogna cambiare il file main.c dove si inserirà la chiamata alla nuova funzione ovvero modalità. Il ciclo while:

```
while (1)        //ciclo infinito in attesa d'interrupt
   {
       if (modality = = MODA)
       modA ( );

       if (modality = = MODB)
```

```
        modB ( );

        if (modality = = MODC)
        modC ( );

        if (modality = = MODD)
        modD ( );

        if (modality = = MODE)
        modE ( );
    }
```

deve essere modificato inserendo la chiamata alla nostra funzione che chiameremo modF.
Si otterrà:

```
while (1)       //ciclo infinito in attesa d'interrupt
{
        if (modality = = MODA)
        modA ( );

        if (modality = = MODB)
        modB ( );

        if (modality = = MODC)
        modC ( );

        if (modality = = MODD)
        modD ( );

        if (modality = = MODE)
        modE ( );

        if (modality = = MODF)
        modF ( );
}
```

- Come ultimo passo bisogna creare la funzione ovvero modalità all'interno del file mod.h. La modalità deve essere creata in coda a tutte le altre e deve essere scritta seguendo questo template:

```
//*************************************************************
//Modalità F
//*************************************************************

void modA ()
{
  AlarmFlag = 0;          //spengo cicalino in caso di allarme
  timeAlarm = 0;          //spengo cicalino sveglia
  ServoDirection = 0;     //disabilito il servo
  PORTE = 0;              //spengo i LED
  Halt ();                //spengo i motori
```

```
  ClearLCD ();

while (1)
 {
  if (modality != MODF)
     break;

  HomeLCD ();
  WriteStringLCD ("Modality modF" );

//Potete scrivere la vostra modalità

  }
```

Diamo un'occhiata al template. All'inizio della funzione si azzerano le variabili standard in maniera tale che il comportamento sia come le altre modalità, ovvero si spegne il cicalino al cambio di modalità, si blocca il servo, si spengono i led e si bloccano i motori. Dopo queste operazioni viene ripulito il display LCD[42]. Fatto questo inizia il ciclo infinito while all'interno del quale si effettuano in maniera continua le operazioni associate alla modalità e il controllo sulla modalità che deve essere attiva. In questo caso la modalità non fa che scrivere in continuazione Modality modF. Eventuali beep prima di attivare la modalità devono essere inseriti prima del ciclo while nel seguente modo:

```
Delay (20);
if (modality != MODF)
  return;
beep ();

Delay (20);
if (modality != MODF)
  return;
beep ();

Delay (20);
if (modality != MODF)
  return;
beep ();

Delay (20);
if (modality != MODF)
  return;
```

Si noti che in questo caso essendo fuori dal ciclo while per poter uscire dalla funzione modF non si esegue un break ma un return.

Qualora si volesse creare una modalità molto simile alle altre i primi due passi devono essere comunque seguiti come precedentemente descritto dopodiché basterà copiare la modalità che volete lievemente modificare e incollarla in coda alle altre. Quello che bisognerà poi cambiare sarà il nome della funzione, copiata e incollata, in modF e bisognerà cambiare il valore della costante utilizzata nel controllo della modalità facendo uso di MODF. Questo cambio deve essere fatto in tutti i punti della funzione in cui viene fatto questo controllo.

[42] Le funzioni per il controllo del display LCD derivano dall'aver incluso la libreria per il suo controllo. Per maggiori informazioni a riguardo si rimanda al Tutorial “C18 step by step”.

In questo modo è possibile facilmente scrivere più modalità Explorer ognuno con un diverso algoritmo per il movimento, ottimizzato per diversi ambienti o funzioni. Ogni volta che si crea una nuova modalità sarà necessario ricompilare il codice sorgente e reinstallarlo nel microcontrollore.

## Domotino GUI controller

Come visto Domotino può essere collegato direttamente alla porta seriale di un PC al fine di poter modificare le impostazioni di sistema o telecomandarlo. Un qualunque programma sia in ambiente Windows che Linux che possa trasmettere sequenze di caratteri ASCII può andare bene per inviare e leggere dati da Domotino. Al fine di semplificare l'invio e la lettura dati si è realizzata una semplice interfaccia grafica per mezzo della quale è anche possibile comandare il Robot.

Dopo aver installato il programma, eseguendo Domotino.exe (in ambiente Windows) si otterrà la finestra riportata in Figura 22. Il programma è in versione inglese ma la semplicità di quest'ultimo non farà della lingua un ostacolo. La schermata principale è divisa in 4 Tabs funzionali:

- Commands
- Driving
- Setting
- RS232

Per mezzo del Tab Commands è possibile inviare i comandi a Domotino. Per mezzo del Tab Driving è possibile guidare Domotino. Per mezzo del Tab Setting è possibile cambiare le impostazioni del sistema Domotino. Per mezzo del Tab RS232 è possibile configurare la porta seriale del computer, questa operazione deve essere la prima cosa da fare la prima volta che si esegue il programma. Vediamo i singoli Tab.

### Commands

Nell'angolo in basso a sinistra è possibile vedere lo stato connesso o disconnesso della porta seriale alla quale il programma si collegherà. Nell'angolo a destra è presente invece l'insieme delle impostazioni che caratterizzano la connessione[43].

Nella casella di testo nominata Commands è possibile scrivere i caratteri ASCII che bisogna inviare al Robot. Nella casella di testo più grande vengono invece scritti tutti i dati ricevuti da Domotino. Dal momento che Domotino rinvia la sequenza di dati ricevuti è possibile vedere il comando trasmesso come dato ricevuto.

Per aprire una connessione con Domotino ed inviare i comandi, ci si deve accertare che la porta seriale che si sta utilizzando è la stessa a cui è collegato Domotino[44] e la sua impostazione sia 19000,N,8,1. Fatto questo è possibile aprire la connessione per mezzo della voce Connect del menù Setting. Se tutto funziona a dovere Domotino si accorgerà dell'avvenuta connessione poiché il programma, dopo aver aperto la connessione trasmetterà il comando X0000000 per mezzo del quale Domotino eseguirà la funzione InitializationText ( ). Quello che si ottiene è quello riportato in Figura 23. Questa stessa schermata la si otterrà ogni volta che si resetta Domotino o si accenda Domotino dopo aver aperto la connessione.

[43] Per ulteriori informazioni sulla porta seriale e le sue impostazioni si rimanda al Tutorial “Il protocollo RS232”.

[44] Al fine di non danneggiare il PC è bene che la connessione e disconnessione del cavo seriale sia fatta a computer spento.

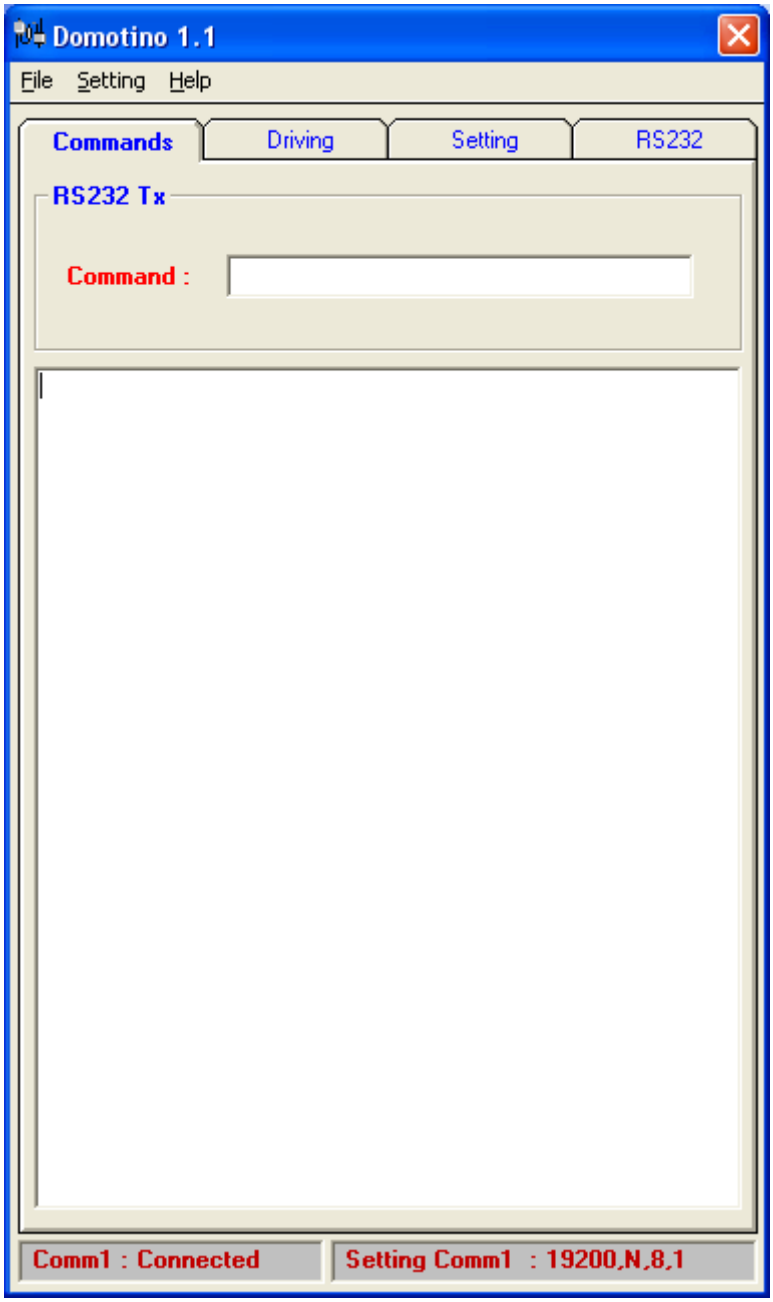



**Figura 22: Schermata principale del programma Domotino**

A questo punto è possibile scrivere sulla riga Commands un qualunque comando ed inviarlo semplicemente premendo invio. Si ricorda che la lunghezza del comando deve essere sempre di 8 caratteri. Questi possono essere anche inviati uno alla volta avendo però il vincolo dello scadere del timeout di circa 2 secondi. Se il comando verrà inviato sempre con la lunghezza corretta non ci si deve preoccupare del timeout.

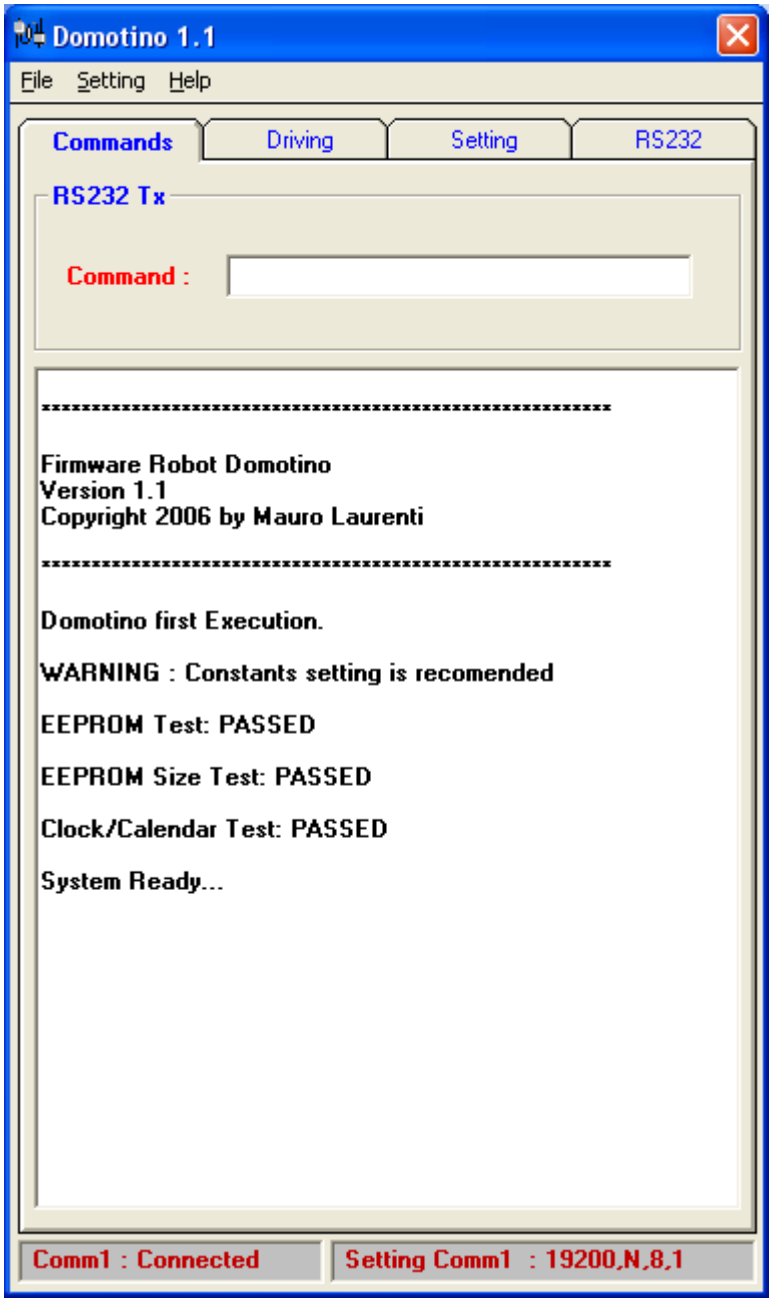



**Figura 23: Schermata principale dopo aver attivato la connessione**

## Driving

Per mezzo del Tab Driving è possibile inviare i comandi per controllare la velocità e la direzione di Domotino semplicemente premendo dei pulsanti. Sul Tab Driving riportato in Figura 24, è possibile vedere la presenza di 5 pulsanti e di un variatore di velocità[45].

Il pulsante F permette di andare avanti, il pulsante R permette di girare a destra, il pulsante L permette di girare a sinistra, il pulsante B permette di fare retromarcia mentre il pulsante S ferma i motori. La velocità deve essere impostata prima dell'invio di ogni comando ovvero pressione di pulsanti poiché si ricorda che ogni comando di direzione contiene al suo interno la velocità con cui Domotino eseguirà il cambio di direzione o semplicemente di velocità. Le velocità disponibili sono

[45] La modalità viene automaticamente cambiata su Remote Control ogni qual volta venga selezionato il Tab Driving.

rispettivamente 0,1,2,3,4,5. Con la velocità 0 Domotino si ferma come se fosse stato eseguito il comando per bloccare i motori[46].

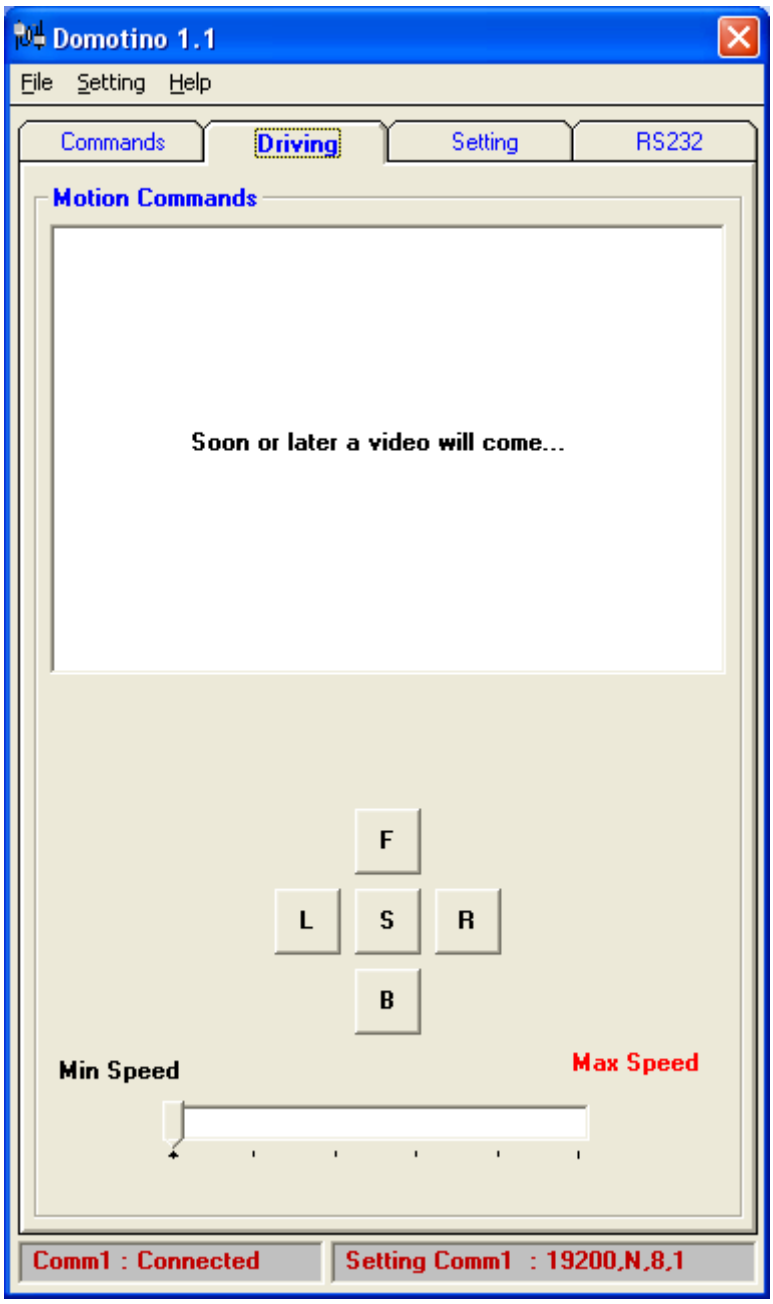



**Figura 24: Schermata per pilotare Domotino**

### Setting

Per mezzo del Tab Setting è possibile inviare automaticamente i comandi di lettura e scrittura della EEPROM di sistema senza doversi preoccupare di inviare i singoli comandi per mezzo del Tab Commands. Dal Tab Setting riportato in Figura 25 è possibile vedere che nel momento in cui ci si sposta sul Tab, qualora la connessione con Domotino sia stata precedentemente instaurata, viene visualizzato lo stato dell'inizializzazione di sistema contenuto nella EEPROM. Variando una qualunque impostazione il tasto Apply diviene attivo e premendolo si trasferiscono i dati visualizzati

[46] In realtà il blocco dei motori avviene poiché si sta eseguendo un cambio di direzione a velocità 0m/s.

sul Tab all'interno della EEPROM.

Per mezzo della prima opzione è possibile attivare un beep ogni volta che si accendono o spengono le lampade. Per mezzo della seconda opzione è possibile attivare un beep ogni volta che si preme il pulsante per cambiare modalità. Per mezzo della terza opzione è possibile attivare un beep ogni volta che si rileva un urto con i pulsanti frontali. Per mezzo della quarta opzione è possibile attivare la modalità Explorer ogni volta che venga spento, dalle modalità modE e modF, l'allarme della sveglia.

Per mezzo delle caselle di testo alla destra di Date è possibile impostare la data di sistema. Con le caselle di testo alla destra di Time è possibile impostare l'ora di sistema inclusi i secondi. Per mezzo delle caselle alla destra di Alarm è possibile impostare l'ora e i minuti della sveglia, la quale può essere attivata o meno per mezzo dell'opzione ON/OFF.

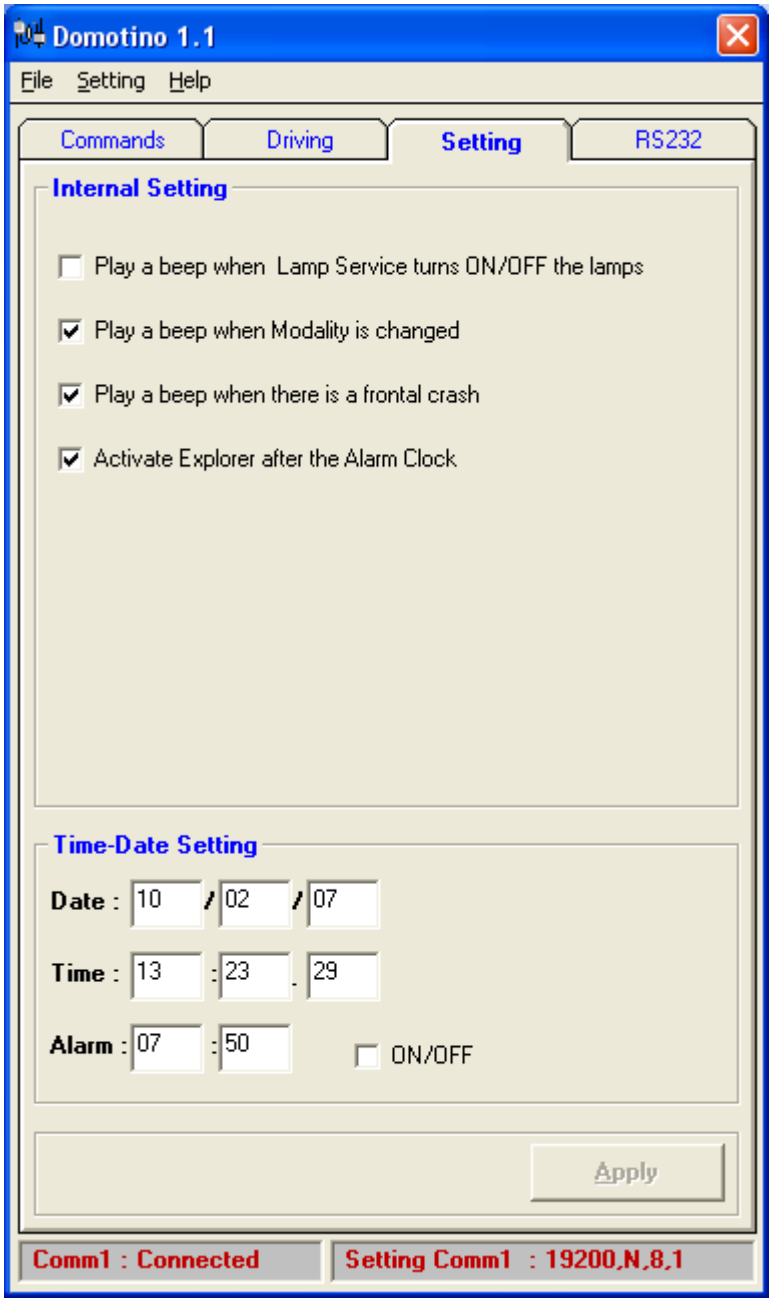



**Figura 25: Schermata per il cambio delle impostazioni di sistema**

Le caselle di testo devono essere riempite rispettando il formato di Figura 25 ovvero scrivendo

sempre decine e unità; dunque, qualora le decine siano pari a 0 bisognerà sempre scrivere lo zero. Si fa presente che il programma non svolge nessuna operazione di Debug per controllare se i dati trasmessi siano stati ricevuti correttamente. Un modo semplice per controllare se i dati sono stati cambiati in maniera opportuna consiste semplicemente nel cambiare Tab e ritornare poi nel Tab Setting permettendo di rileggere i dati nella EEPROM.

**RS232**

Per mezzo del Tab RS232 è possibile cambiare la porta seriale e le sue impostazioni. Una volta che il programma viene chiuso le impostazioni vengono salvate nel file setting contenuto della directory dove è installato Domotino.exe. Questo significa che se si collega Domotino sempre alla stessa porta seriale non ci si dovrà preoccupare molto delle impostazioni della porta seriale anche se una sbirciata all'angolo destro è sempre consigliabile al fine d'essere certi che le impostazioni siano corrette. Il Tab RS232 è riportato in Figura 26.

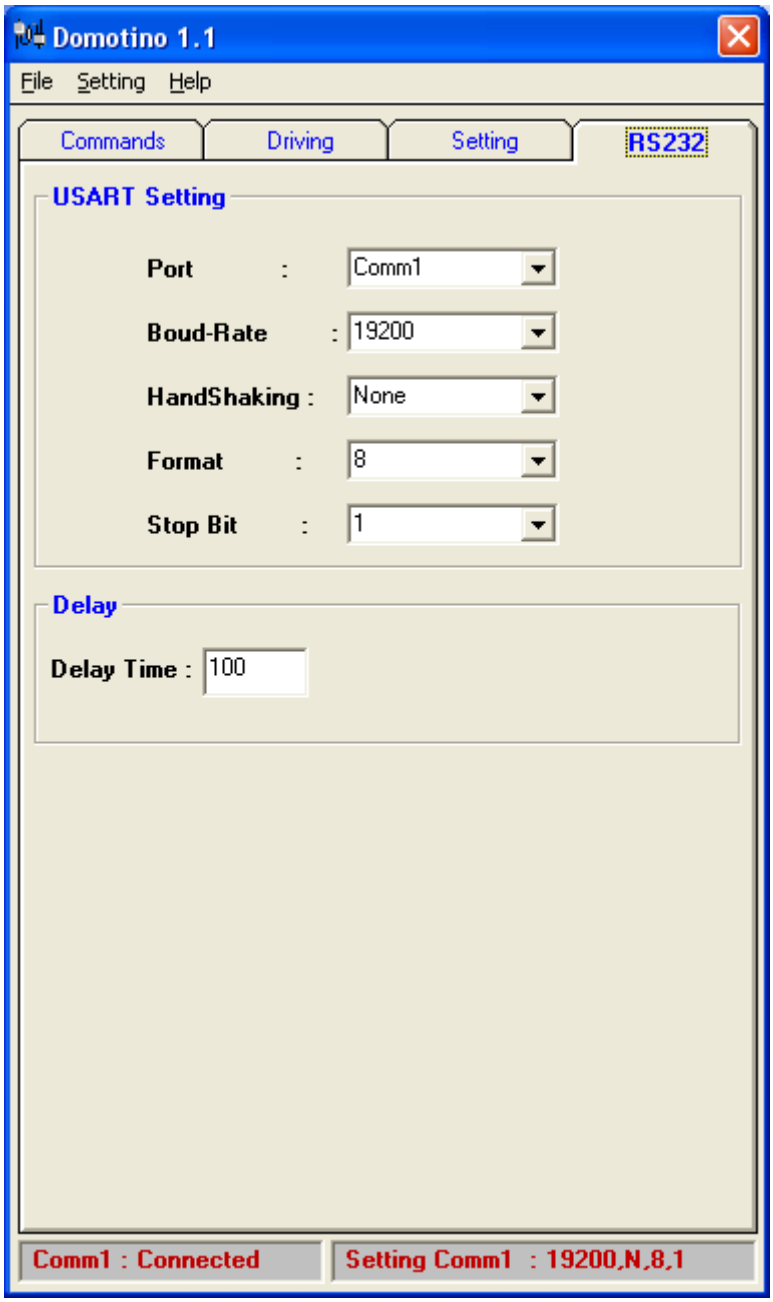



**Figura 26: Schermata per le impostazioni della Porta RS232**

E' possibile osservare che è possibile cambiare tutte le impostazioni della porta seriale del PC. Questa possibilità discende dal fatto che è possibile utilizzare questo programma anche in altre applicazioni in cui siano richieste altre impostazioni. Le configurazioni della porta[47] per poter comunicare con Domotino devono essere come in Figura 26. Unico parametro che può cambiare è la variabile Delay Time che può assumere un valore compreso tra 1 e 2000. Il valore può cambiare da computer a computer e deve essere scelto in maniera da far funzionare correttamente il programma. In particolare valori troppo piccoli possono causare il blocco del programma stesso o di Domotino. Se si nota che i dati ricevuti da Domotino non riempiono in maniera opportuna le caselle di testo del Tab Setting vuol dire che bisogna aumentare il valore di Delay Time. Valori alti hanno come effetti collaterali solo quello di rallentare la trasmissione e non quello di causare errori.

[47] Ogni volta che viene cambiato un parametro della porta, ad eccezione del Delay, la connessione viene interrotta ed è necessario riattivarla se si è certi che le impostazioni siano corrette. Qualora le impostazioni non siano supportate dal sistema la connessione non verrà attivata e si avrà una segnalazione d'errore.

## Bibliografia

www.LaurTec.com : sito di elettronica dove poter scaricare gli altri articoli menzionati, aggiornamenti e progetti.