UNIVERSITÀ DI BRESCIA
FACOLTÀ DI INGEGNERIA
Dipartimento di Elettronica per l'Automazione

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica Mobile
(Prof. Riccardo Cassinis)

Riprogrammazione del robot da mini-sumo

Elaborato di esame di:

Davide Barba, Andrea Marteletti

Consegnato il: **16 dicembre 2008**

Sommario

Mark III è un robot costruito per affrontare le gare di mini-sumo. Grazie ai sensori di cui è dotato, è in grado di muoversi all'interno di un'arena di colore nero, di forma circolare, delimitata da una striscia bianca di larghezza 2.5 cm. Nella configurazione iniziale il suo compito era quello di percorrere l'arena alla ricerca di un ostacolo e di cercare di spingerlo al di fuori della linea bianca. Non sempre questa risulta essere la tattica migliore di vittoria, quindi si vuole cercare una possibile alternativa ad essa e cioè si vuole fare in modo che Mark III si muova in modo casuale senza oltrepassare la linea di confine e senza che urti nessun ostacolo presente lungo il suo percorso.

1. Introduzione

Mark III è il successore di due robot progettati e realizzati dalla società "Portland Area Robotics Society" [1]. La versione iniziale fu realizzata per promuovere le prime competizioni, organizzate nel 2000, tra robot da minisumo [2]. Discorso analogo per la seconda versione del robot, prodotta nel 2001 in contemporanea con il secondo contest annuale; a differenza della prima versione, al secondo furono introdotti i sensori di prossimità a infrarossi che consentissero di rilevare la presenza dell'avversario.

Mark III [3] è stato realizzato sviluppando la seconda versione: è stata cambiata la CPU e sono stati migliorati tutti gli aspetti di Mark II (inclusi prezzo e prestazioni). Rispetto alla precedente versione ha il vantaggio di essere maggiormente espandibile e general-purpose.

Le gare di minisumo sono molto diffuse in Italia; in rete sono quindi facilmente reperibili programmi già scritti in diversi linguaggi per la gestione dei comportamenti desiderati del robot. I consigli forniti dagli utenti dei siti, che forniscono tali programmi, [11] e le loro guide hanno semplificato l'operazione di stesura del programma. È stato, infatti, facile individuare le cause dei diversi errori commessi e si è avuta a disposizione una struttura base poi sviluppata e personalizzata in funzione al nostro obiettivo finale. In particolare si voleva che il robot si muovesse in modo casuale all'interno dell'arena e che non urtasse nessun ostacolo presente su essa.

2. Il problema affrontato

La struttura del robot da mini-sumo in oggetto è approssimativamente modulare; si può rappresentare secondo lo schema a blocchi riportato in figura 1, in cui sono evidenziate le parti fondamentali.

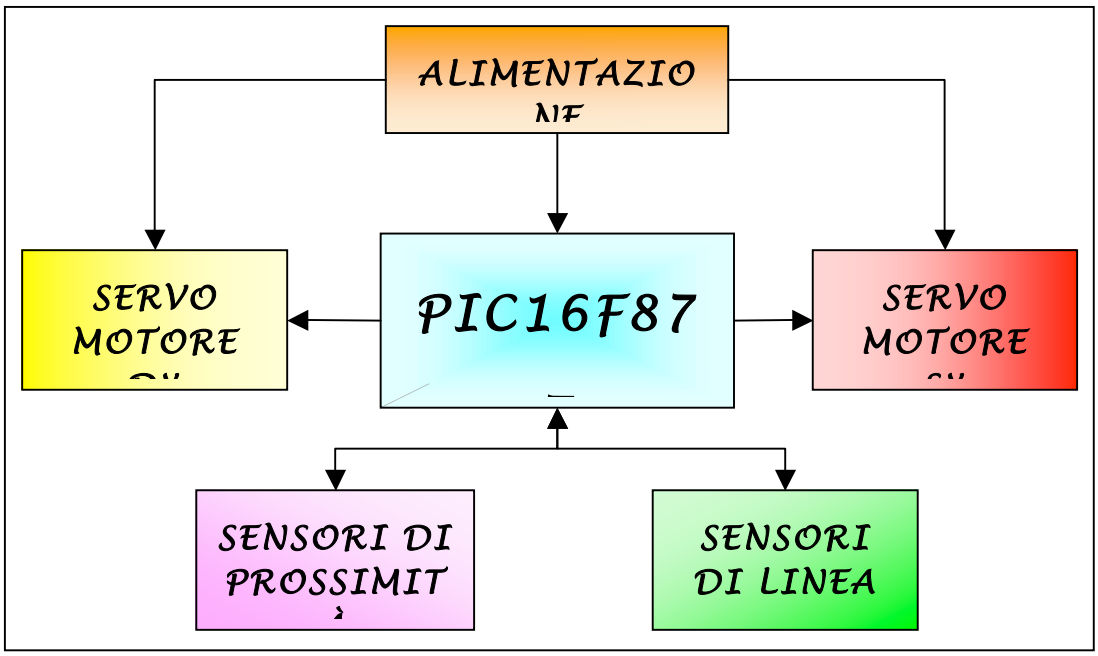


Fig. 1 - Schema a blocchi del robot

Vi è la necessità di analizzare il funzionamento di ciascun blocco in modo da determinarne l'effetto sul comportamento del robot, sfruttandone al meglio le caratteristiche per ottenere il miglior risultato possibile. Particolare attenzione va posta al blocco della CPU e a quello dei sensori; il motivo sarà analizzato nei paragrafi successivi.

2.1. Il PIC

La CPU è costituita da un PIC16F877 [4] prodotto dalla Microchip. Per le caratteristiche e funzionalità di questo dispositivo si rimanda al datasheet allegato.

All'interno della sua memoria è già salvato il PICLoader, realizzato da Rick Farmer [5], che rende il PIC in grado di comunicare direttamente con il computer e che consente il caricamento di nuovi programmi utilizzando semplicemente un cavo seriale RS232.

È importante notare che il PICLoader accetta esclusivamente file nel formato INHX8M (al fine di settare correttamente la lunghezza in byte della parola). Ogni tentativo di caricare file con altri formati porterà a un errore "Invalide record type".

- **Nella scelta del compilatore sarà quindi importante verificare che questo sia in grado di produrre un file compatibile, in caso contrario non sarà possibile caricare nessun programma.**

Il programma consigliato per dialogare con questo PIC è Hyper Terminal, disponibile su tutti i calcolatori Windows o facilmente reperibile online. Bisogna porre particolare attenzione ai settaggi e alle procedure da eseguire per ottenere la corretta comunicazione tra i dispositivi (come riportato nel paragrafo 4.2.1).

Una delle caratteristiche più interessanti di Mark III è quella di non richiedere software e ambienti di programmazione troppo sofisticati, permettendo anche a utenti inesperti di utilizzarne le funzionalità basilari. Contemporaneamente supporta linguaggi più articolati, consentendo ampia personalizzabilità ad utenti esperti. Questa flessibilità si traduce nella non necessità di avere software proprietari, e in particolare, in una ampia possibilità di scelta riguardante il linguaggio di programmazione che si desidera utilizzare:

- Ch Basis (proposto da Mark Gross; facile da imparare) [6];
- JAL (proposto da Brett Nelson; abbastanza facile da imparare, simile al Pascal) [7];
- C (proposto da James Farwell per il CC5X, da Pete Skeggs per il CCS C; entrambi i compilatori sono integrabili con l'ambiente di sviluppo MPLAB, il linguaggio C è più complesso rispetto al Basic, ma più flessibile);
- OOPic (proposto da Pete Skeggs; facile da imparare, è un linguaggio orientato agli oggetti) [8];
- PIC assembly (proposto da Rick Farmer; difficile da imparare ma necessario se si vogliono ottenere elevate prestazioni).

Il PIC è utilizzato, oltre che per l'esecuzione del programma, per l'acquisizione dei segnali generati dai sensori di prossimità e di linea. È importante quindi sfruttare correttamente il convertitore analogico/digitale. Per l'implementazione pratica si veda il paragrafo 3.1.

2.2. Il compilatore

Il primo quesito da affrontare riguarda il linguaggio di programmazione da utilizzare. Avere a disposizione un linguaggio flessibile è indubbiamente un vantaggio, poichè garantisce in futuro la possibilità di modificare e migliorare il programma in modo facile e veloce. Considerato inoltre che non vi è la necessità di utilizzare il linguaggio macchina, dato che non sono richieste prestazioni elevate, la scelta è ricaduta sul linguaggio C (linguaggio già ben appreso durante altri corsi di studio).

Indipendentemente dal linguaggio utilizzato, il file che deve essere caricato nel PIC deve essere di tipo esadecimale. Si rende quindi necessario l'utilizzo di un compilatore che traduca il programma scritto in C in un file compatibile con il PIC.

2.3. I sensori

Come evidenziato nello schema 1.1, il robot è dotato di due tipologie diverse di sensori: di prossimità e di linea. I primi sono utili per rilevare l'eventuale presenza di ostacoli lungo il percorso, i secondi (come dice il nome stesso) sono utilizzati per identificare la linea bianca, che rappresenta il confine dell'arena dalla quale il robot non deve uscire. La gestione delle due tipologie viene affrontata in modo distinto per la rilevante differenza di utilizzo e di programmazione.

2.3.1. Sensori di linea

I sensori di linea montati su Mark III sono prodotti dalla FAIRCHILD SEMICONDUCTOR e sono il modello QRB 1133 (figura 2) [9].

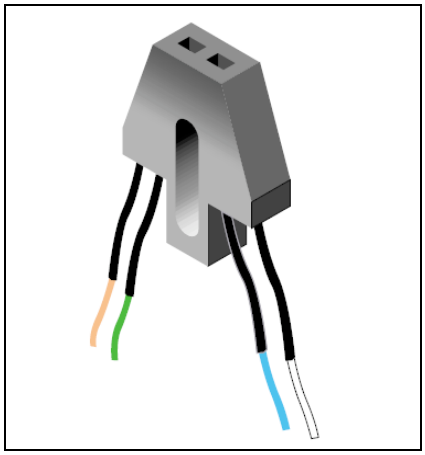


Fig. 2 - Disegno del sensore di linea

Essi sono costituiti da un diodo che emette un segnale infrarosso affiancato da un fototransistor NPN, come schematizzato in figura 3. Il fototransistor rileva la radiazione emessa solo quando un oggetto riflettente passa attraverso il campo visivo (che è approssimativamente un cerchio di diametro pari a 0,5cm).

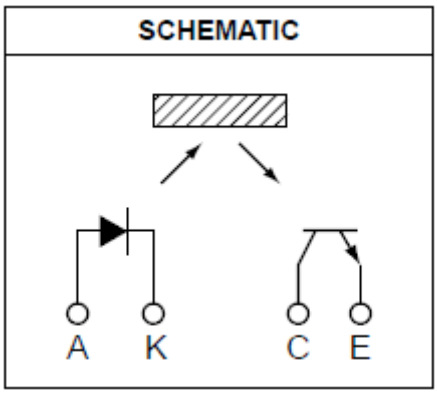


Fig. 3 - Schematizzazione del sensore

I sensori sono montati sotto la parte anteriore del robot, in modo da fornire al PIC una tensione differente a seconda che il robot si trovi su una parte nera (e quindi all'interno della pista) o sulla linea bianca (e quindi sul confine). In particolare il valore numerico associato alla misura (calcolato campionando il segnale fornito dal sensore) varia da 0 a 255. Tale valore non è inerente al codice RGB, in cui lo 0 corrisponde al nero e il 255 al bianco, bensì il nero corrisponde a un valore prossimo al 255 e il bianco a

uno prossimo allo 0. Nel paragrafo 3.3.1 sarà illustrato l'utilizzo di questi sensori per risolvere il problema.

2.3.2. Sensori di prossimità

In figura 4 è riportata un'immagine dei sensori di prossimità montati su Mark III: gli Sharp GP2D12 [10].

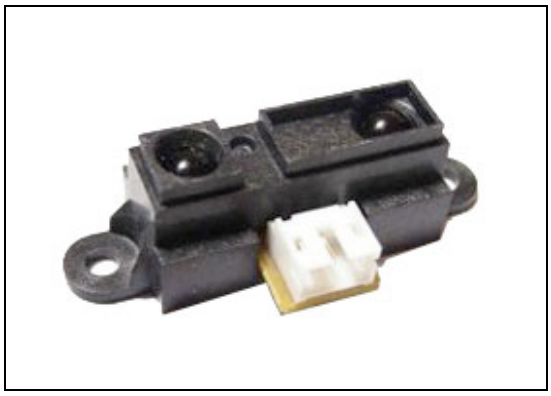


Fig. 4 - Sharp GP2D12

Essi sono dispositivi in grado di rilevare distanze di oggetti posti all'interno del proprio campo visivo (10-80cm). La misura di distanza si basa su un concetto semplice: un fotodiodo emette un fascio nella direzione in cui si vuole effettuare la misura; quando tale fascio colpisce un oggetto viene riflesso verso un foto rilevatore. Misurando il tempo di volo del fascio è possibile determinare la distanza dell'oggetto dal sensore. In figura 5 è riportato il valore della tensione di uscita fornita dal sensore al variare della distanza dell'oggetto.

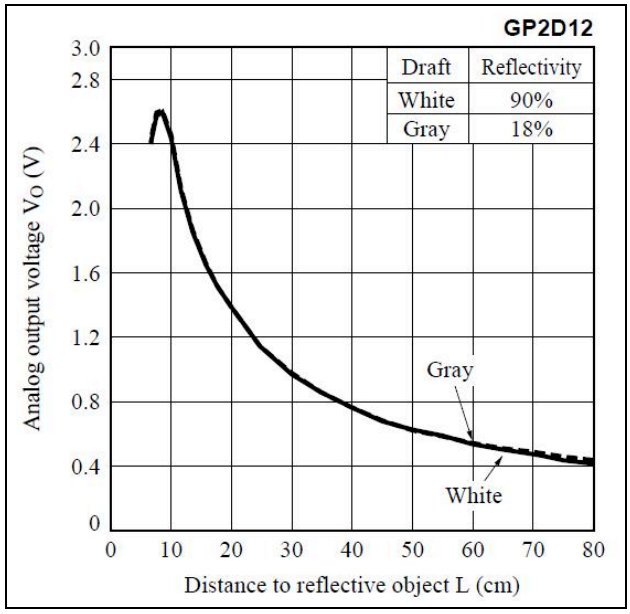


Fig. 5 - Andamento della tensione di uscita al variare della distanza dell'oggetto

Il valore di tensione non è molto indicativo, in quanto in fase di programmazione è necessario disporre del valore numerico associato alla misura di distanza (cioè del valore numerico determinato dal convertitore A/D). Considerando che tale valore non è immediatamente determinabile, è opportuno risalire ad esso sulla base di analisi sperimentali: si pone un oggetto in varie posizioni e a varie distanze dai sensori, registrando l'eventuale valore numerico associato alla misura (la motivazione di tale eventualità è inerente al campo visivo particolarmente ristretto). I risultati ottenuti sono riportati in figura

6; le linee rosse delimitano il cono di visione di ciascun sensore. Risulta immediato notare la prima problematica associata a tali sensori: la visibilità è limitata quasi esclusivamente agli oggetti che si trovano perfettamente di fronte al sensore, fattore che influirà notevolmente sulle prestazioni ottenibili.

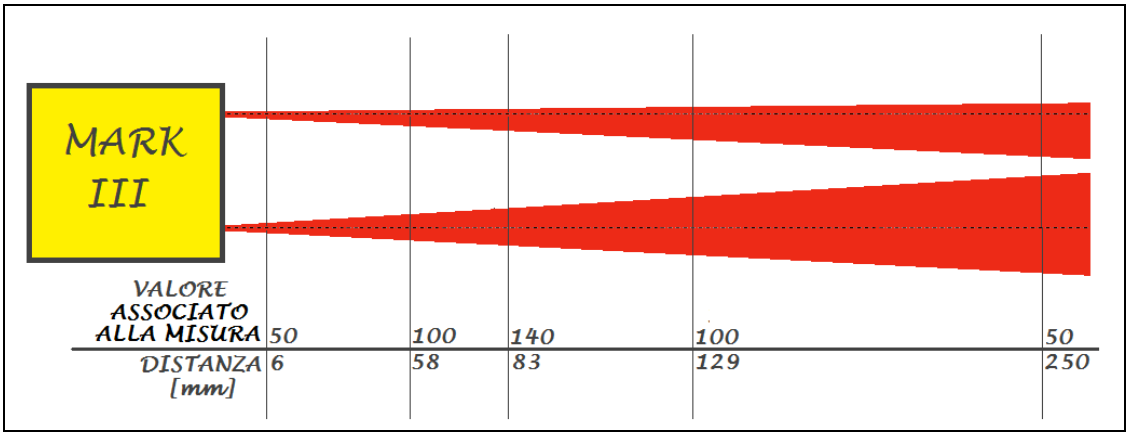


Fig. 6 - Risultato delle prove effettuate con i sensori

Per un maggiore dettaglio, tutti i dati rilevati sono inseriti nel grafico di figura 7. In ascissa è riportata la distanza dell'oggetto dal robot (in mm) e in ordinata il valore numerico associato alla misura.

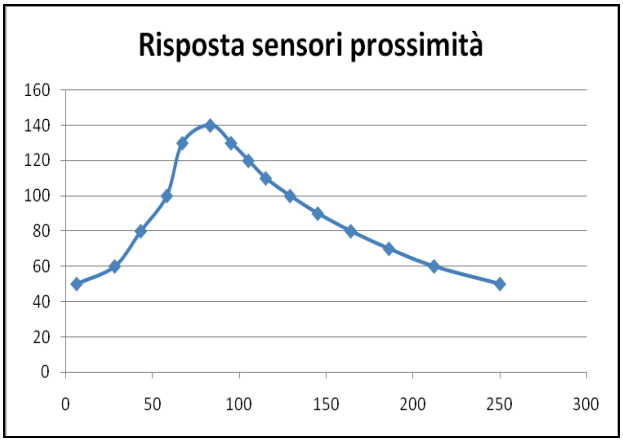


Fig. 7 - Grafico risposta dei sensori di prossimità

Dal grafico è possibile rilevare la seconda problematica associata a tali sensori: all'interno del range considerato vi è ambiguità sul valore associato alla misura, cioè un unico valore è riferibile a due differenti distanze. Ad esempio il valore 100 è associato sia alla distanza 60mm che a quella di 130mm. All'interno del range di funzionamento dei sensori, indicato dal costruttore, questo non avviene (il funzionamento è garantito per distanze comprese tra i 10 e gli 80cm); bisogna però anche ipotizzare che durante il movimento il robot possa effettuare una svolta e incappare in un oggetto a una distanza inferiore ai 10cm. Il valore associato alla misura sarebbe così prossimo al 50 (come si può osservare dal grafico); questo potrebbe essere interpretato come un oggetto posto a una distanza di circa 25cm, anzichè di 10cm. Sicuramente il robot urterebbe l'ostacolo, non rispettando quindi le specifiche richieste.

Nel paragrafo 3.3.2 sarà illustrata la soluzione adottata per risolvere tale problema.

2.4. La programmazione

Analizzate tutte le diverse componenti hardware, è ora possibile valutare i requisiti per la parte software. In primis è necessario determinare con esattezza il comportamento desiderato dal robot:

1. Il robot si dovrà muovere in modo casuale all'interno dell'arena

2. Il robot non dovrà oltrepassare la linea bianca che delimita l'arena
3. Il robot dovrà schivare gli eventuali ostacoli presenti lungo il suo percorso

Definite le specifiche-guida da osservare, è possibile realizzare un diagramma di flusso che riassuma il principio di funzionamento del robot; implementando ogni singola parte del diagramma si realizzerà il codice che garantirà il corretto funzionamento del robot.

3. La soluzione adottata

In questo capitolo si illustreranno le varie soluzioni adottate per risolvere i problemi elencati nel capitolo precedente.

3.1. Il PIC

Tra le varie funzionalità del PIC, particolarmente utile risulta essere quella del convertitore analogico/digitale.

Il modulo A/D è implementato secondo una delle tecniche più diffuse: il segnale analogico è campionato da un sample&hold; successivamente il convertitore elabora il segnale così generato, producendo il segnale digitale attraverso approssimazioni successive. Questo modulo è composto da quattro registri:

- A/D Result High Register (ADRESH);
- A/D Result Low Register (ADRESL);
- A/D Control Register0 (ADCON0);
- A/D Control Register1 (ADCON1).

Il registro ADCON0 controlla le operazioni del modulo A/D; ADCON1 configura la modalità di funzionamento dei porti di ingresso/uscita del PIC. Quando la conversione è completata, il risultato è caricato all'interno dei registri ADRESH:ADRESL e il bit GO/DONE del registro ADCON0 è resettato.

Per maggiori dettagli sui registri e sulla procedura completa per effettuare la conversione, si consiglia la consultazione del capitolo 11 del datasheet allegato.

3.2. Il compilatore

Come riportato nel paragrafo 2.2, i compilatori utilizzabili sono due: il CC5X [12] e il CCS C [13]. La scelta è caduta sul primo poiché è open source.

Le caratteristiche del CC5X sono riportate nel datasheet allegato. Ovviamente è in grado di produrre file nel formato INHX8M.

È possibile integrare il compilatore CC5X all'interno dell'ambiente MPLAB, in modo da poterne sfruttare appieno le potenzialità, velocizzando e semplificando le operazioni da svolgere. Naturalmente non è l'unico metodo corretto per creare il file hex; se si desidera, infatti, godere di una maggiore flessibilità è possibile utilizzare i due strumenti in due fasi separate: il file C viene scritto utilizzando MPLAB e successivamente compilato tramite CC5X.

Nel paragrafo 4.2 sono riportate le istruzioni di utilizzo di tale compilatore. Esse sono inserite all'interno del file "OPTIONS" (nella cartella contenente il compilatore stesso).

3.3. I sensori

Di seguito si esporranno le scelte progettuali che hanno permesso di risolvere le problematiche legate alle due tipologie di sensori utilizzate.

Il dettaglio del codice implementato è consultabile nei documenti allegati.

3.3.1. Sensori di linea

Mark III è dotato di tre sensori di linea posizionati nella parte anteriore del robot: uno centrale e gli altri agli estremi, come rappresentato in figura 8.

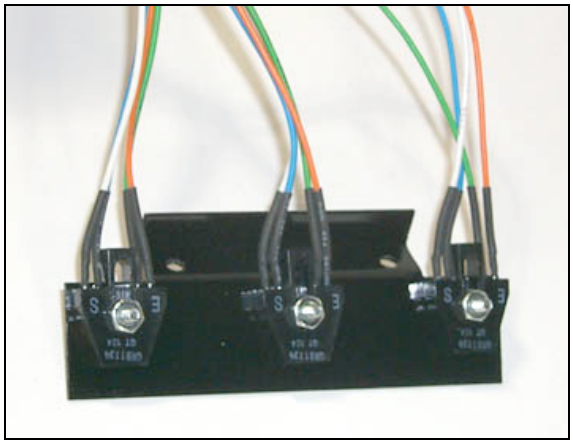


Fig. 8 - Sensori di linea

Tale disposizione dei sensori è particolarmente vantaggiosa, dato che consente di capire il movimento eseguito dal robot effettuando semplicemente un confronto tra i valori che essi forniscono. Al fine di realizzare un programma efficace, infatti, è importante verificare come il robot si avvicini al bordo dell'arena: a seconda dell'angolo di incidenza tra robot e linea si procederà ad attuare un controllo più o meno energico. Per il dettaglio delle strategie implementate si rimanda al paragrafo 3.4.

Nel paragrafo2.3.1 è descritto il metodo di funzionamento di tali sensori. È importante ora scegliere il valore da utilizzare come soglia, cioè determinare il valore con il quale stabilire se sia stata rilevata o meno la linea bianca. Impostare a priori questo valore non è conveniente, in quanto l'eventuale presenza di sporco nell'arena o la non uniformità di colore del fondo potrebbero influenzare il corretto comportamento del robot. La soluzione adottata è semplice: si calcola la media dei tre valori letti dai sensori e si setta la soglia a 2/3 della media; la lettura inferiore alla soglia indicherà che quel sensore ha identificato la linea bianca.

3.3.2. Sensori di prossimità

Le problematiche da risolvere sono principalmente due: lo scarso angolo di visione dei sensori e il valore minimo oltre il quale è possibile effettuare la misura.

La soluzione più immediata è quella di sostituire i sensori con altri più performanti, ma la necessità di mantenere la compatibilità con l'hardware, già implementato, ne sconsiglia la realizzazione. Per lo stesso motivo non è nemmeno possibile aggiungere ulteriori sensori che consentirebbero invece di ampliare il campo visivo. Considerata quindi l'impossibilità di modificare l'angolo di visione, si è scelto di riposizionare i sensori come in figura 9; in tal modo essi avranno una distanza minima da qualsiasi oggetto superiore al valore critico, individuato precedentemente.

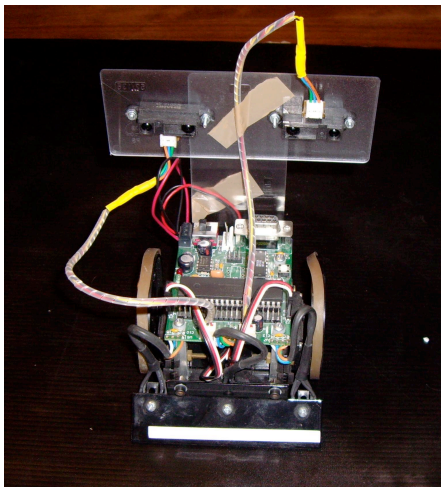


Fig. 9 - Modifiche apportate a Mark III

3.4. La programmazione

In questo paragrafo si illustrerà la struttura del programma realizzato, rimandando la visualizzazione dell'intero codice al file allegato.

All'avvio viene eseguita una procedura di inizializzazione dei porti di input/output del PIC e il calcolo del valore di soglia necessario per il corretto utilizzo dei sensori di linea. Terminata questa fase si procede all'esecuzione dell'algoritmo vero e proprio, inserito all'interno di un ciclo "while(1)", che si ripeterà all'infinito (fino allo spegnimento del robot).

L'algoritmo è costituito da varie fasi in sequenza, come rappresentato in figura 10.

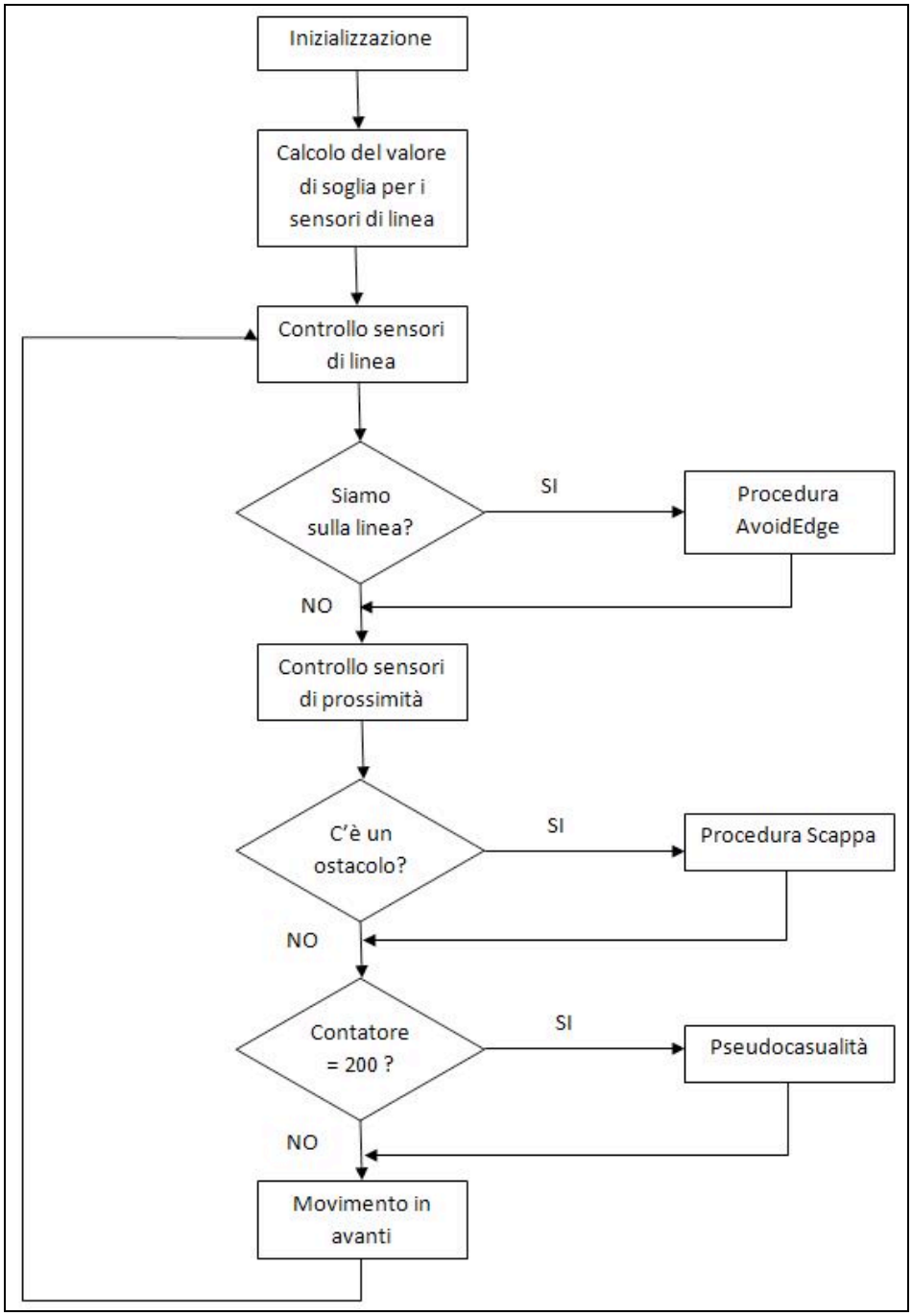


Fig. 10 - Diagramma riassuntivo del programma realizzato

Durante la prima fase avviene il controllo dello stato dei sensori di linea: quando almeno uno di questi fornisce un valore inferiore alla soglia precedentemente calcolata (ossia avviene l'identificazione della linea bianca di confine), viene richiamata la procedura di allontanamento, denominata AvoidEdge; se ciò non accade si procederà alla fase successiva.

Una volta accertato che il robot non si trovi in prossimità del bordo dell'arena, si verificherà che non vi siano ostacoli contro i quali questo possa urtare, ovvero acquisiti i valori dai sensori di prossimità, li si confronterà con la soglia impostata. Sarà eventualmente richiamata la procedura per la creazione di un percorso alternativo al fine di evitare la collisione.

Verificata l'assenza di ostacoli si procede ora a far muovere Mark III all'interno dell'arena. Per implementare la pseudo-casualità sono stati precedentemente introdotti due parametri: il contatore e la direzione. Il robot avanza fino a quando il valore di contatore risulta essere maggiore di 200, oltre il quale il robot cambia direzione. Il valore di contatore viene incrementato ogni volta che il ciclo while viene eseguito, contemporaneamente all'interno delle funzioni *AvoidEdge* e *Scappa* ne viene alterato il valore in modo da aumentarne la casualità. Per stabilire la nuova direzione del robot si effettua la differenza tra i valori forniti rispettivamente dal sensore di linea destro e quello sinistro; nel caso in cui il risultato sia negativo, Mark III girerà verso destra, in caso contrario girerà verso sinistra. In entrambi i casi l'entità della rotazione dipenderà dal modulo della differenza precedentemente calcolata.

La funzione *AvoidEdge* gestisce il comportamento del robot in base ad un opportuno parametro che determina il verificarsi di tre situazioni. La prima prevede che il robot rilevi la linea con tutti e tre i sensori, come riportato in figura 11.

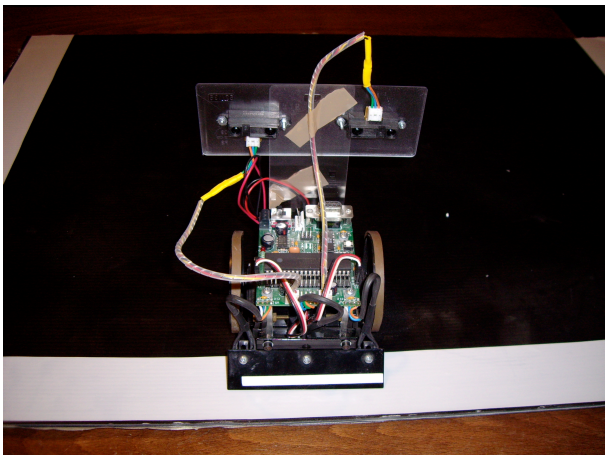


Fig. 11 - Avvicinamento perpendicolare al confine dell'arena, arretramento

In questo caso Mark III arretra, come riportato in figura 12.

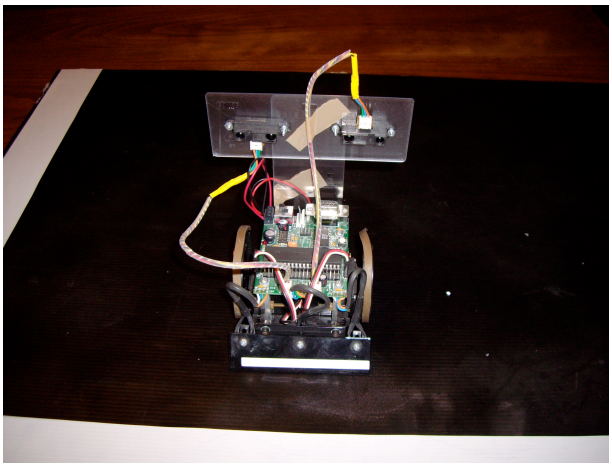


Fig. 12 - Arretramento

Successivamente il robot effettua una rotazione di 180°, allontanandosi così dal bordo dell'arena, come in riportato in figura 13.

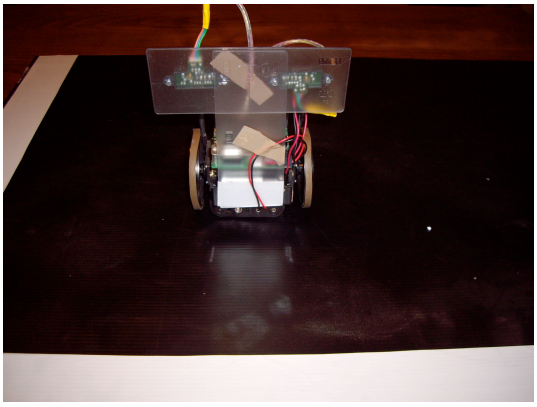


Fig. 13 - Rotazione di 180° e ripresa del movimento

La seconda situazione prevede che il robot rilevi la linea di confine solo con il sensore di sinistra, come in figura 12.

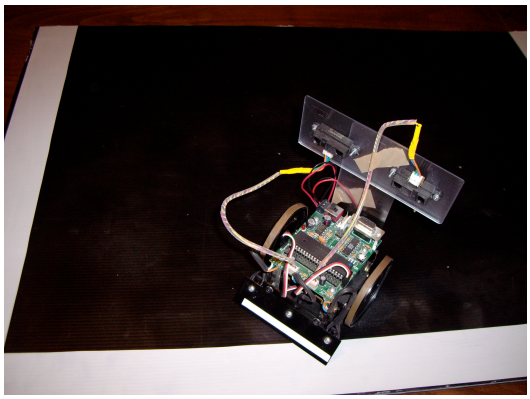


Fig. 14 - Avvicinamento trasversale al bordo dell'arena

Mark III prosegue quindi il suo movimento ruotando di circa 90° verso destra, in modo da allontanarsi dal bordo, come riportato in figura 15.

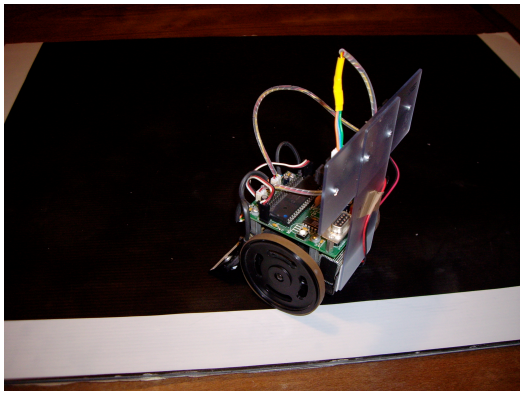


Fig. 15 - Rotazione di 90° e ripresa del movimento

La terza situazione prevede che il robot la rilevi solo con il sensore di destra (Mark III prosegue quindi il suo movimento ruotando di circa 90° verso sinistra). Il tutto viene gestito tramite controlli condizionali "if" in cascata.

La funzione *Scappa* viene richiamata qualora il robot incontri un ostacolo sul suo percorso. Al richiamo vengono passati due parametri: il valore del sensore di prossimità sinistro e il valore del sensore di prossimità destro. Il comportamento determinato da questa funzione è molto simile a quello della funzione AvoidEdge; anche in questo caso si distinguono tre diverse situazioni in base ai parametri adottati. La prima si verifica quando MARK III rileva l'ostacolo solo con il sensore di destra (come riportato nell'immagine sinistra in figura 16).

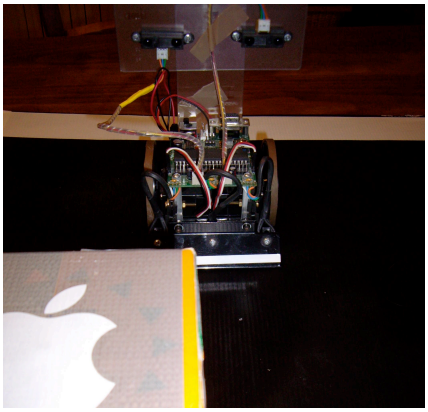


Fig. 16 - Ostacolo rilevato solo dal sensore destro

La seconda quando lo rileva solo con il sensore sinistro (come riportato in figura 17).

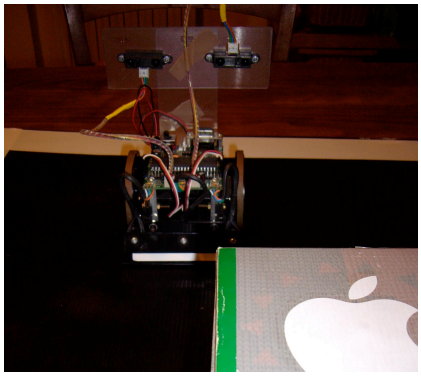


Fig. 17 - Ostacolo rilevato solo dal sensore sinistro

Infine la terza quando l'ostacolo si trova di fronte al robot (come riportato in figura 18). In questo caso entrambi i segnali dei sensori di prossimità saranno sopra della soglia prestabilita.

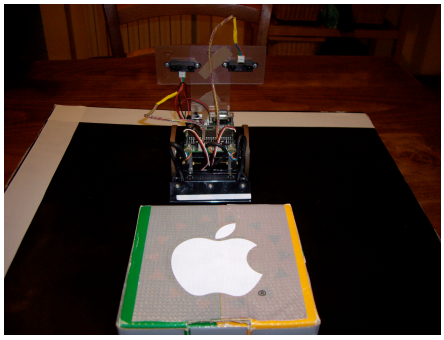


Fig. 18 - Ostacolo rilevato da entrambi i sensori

Nei primi due casi il robot devia il suo percorso nella direzione opposta rispetto a quella in cui si trova l'ostacolo, come nell'esempio di figura 14.

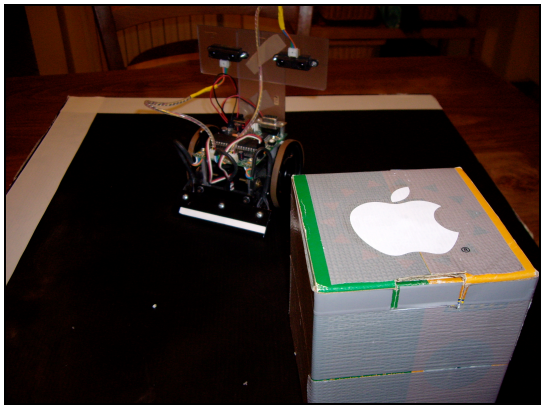


Fig. 19 - Ostacolo evitato con una rotazione nella direzione opposta

Nel terzo caso invece interrompe il suo movimento, indietreggia e ruota senza urtare l'ostacolo, come riportato in figura 15. Il senso di rotazione viene determinato confrontando i parametri: se il valore del sensore di sinistra è maggiore rispetto a quello del sensore di destra allora il robot ruoterà a destra, viceversa ruoterà a sinistra.

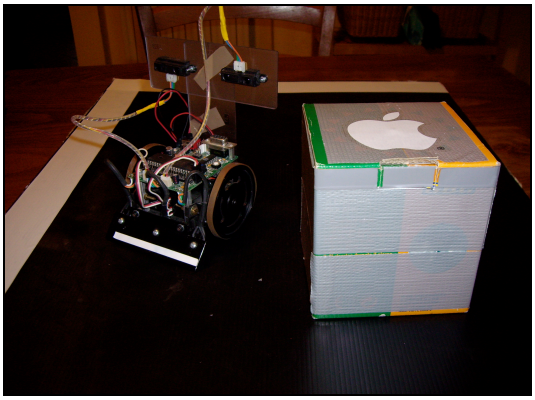


Fig. 20 - Ostacolo evitato con un arretramento e una rotazione nella direzione opposta

4. Modalità operative

In questo capitolo si descriverà la procedura di programmazione del robot: la scrittura del programma, la connessione fisica al calcolatore, il caricamento del programma in formato HEX e l'utilizzo di Mark III.

4.1. Procedura generale per la programmazione del Robot

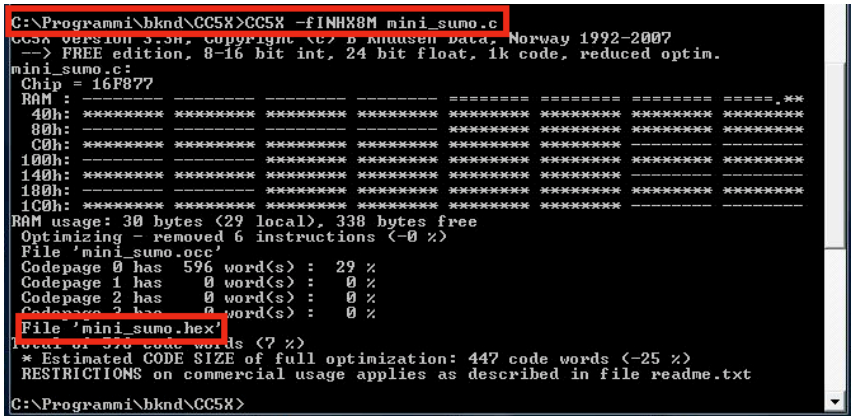
Per effettuare la programmazione del robot è necessario seguire la seguente procedura:

1. Scrivere il codice sorgente del programma (indipendentemente dal linguaggio di programmazione scelto) usando un qualsiasi ambiente di sviluppo (es. MPLab) o un editor (es. Windows Notepad). Salvare il file nel formato corretto (ad esempio .c, .asm ecc);
2. Compilare il codice sorgente utilizzando il compilatore adatto al linguaggio utilizzato;
3. Fare attenzione che il file prodotto dal compilatore sia compatibile con il PICLoader di Rick Farmer;
4. Collegare il robot al calcolatore usando un cavo seriale RS-232 a 9 pin (da un lato con un connettore femmina, dall'altro maschio);
5. Caricare il file hex dal calcolatore utilizzando BotLoader o Hyperterminal o qualsiasi altro programma analogo ad essi;
6. Scollegare il robot e testarne il funzionamento.

4.2. Procedura dettagliata per la programmazione del Robot

Nel paragrafo precedente è stata presentata una procedura generale che consenta di programmare Mark III; in questo paragrafo, invece, si riportata l'elenco dettagliato delle operazioni da noi eseguite.

1. Scrivere il programma utilizzando MPLab e salvarlo come *"mini_sumo.c"*;
2. Copiare il file all'interno della cartella in cui è installato il compilatore;
3. Aprire il prompt di ms-dos;
4. Portarsi nella cartella del compilatore e utilizzare la seguente istruzione: "CC5X -fINHX8M mini_sumo.c";



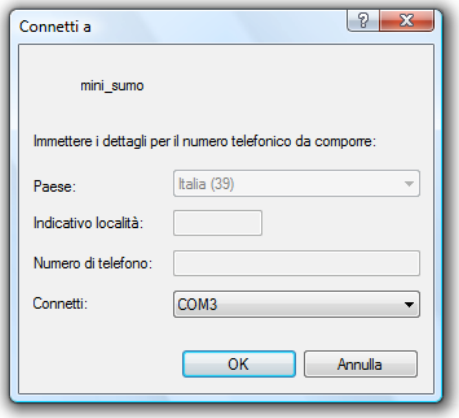
5. Caricare il programma utilizzando Hyper Terminal (per i dettagli si veda il paragrafo 4.2.1);
6. Scollegare il robot e testarne il funzionamento.

4.2.1. Hyper Terminal

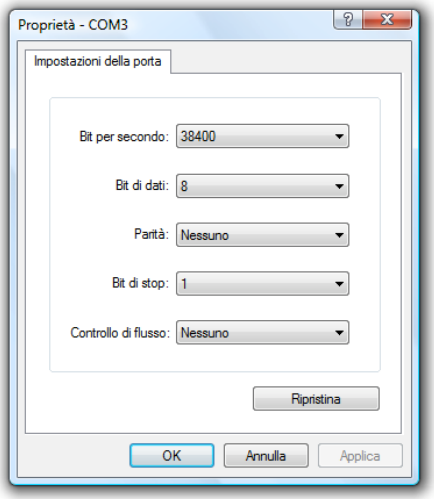
Per utilizzare correttamente Hyper Terminal si consiglia di seguire attentamente le seguenti istruzioni:

1. Collegare il robot alla porta seriale;
2. Aprire Hyper Terminal e nominare alla connessione (ad esempio mini_sumo);

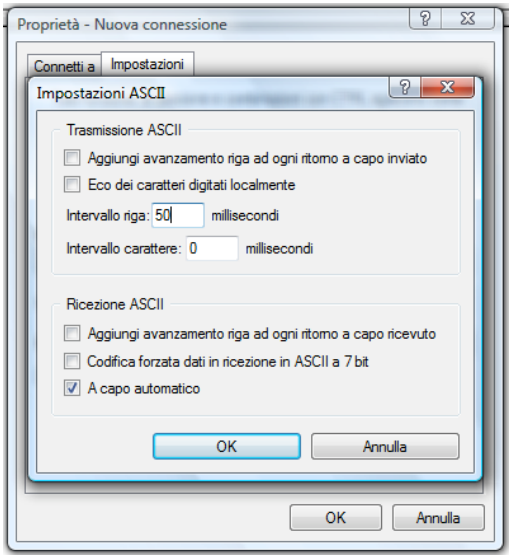
3. Selezionare la porta COM alla quale si desidera connettersi;



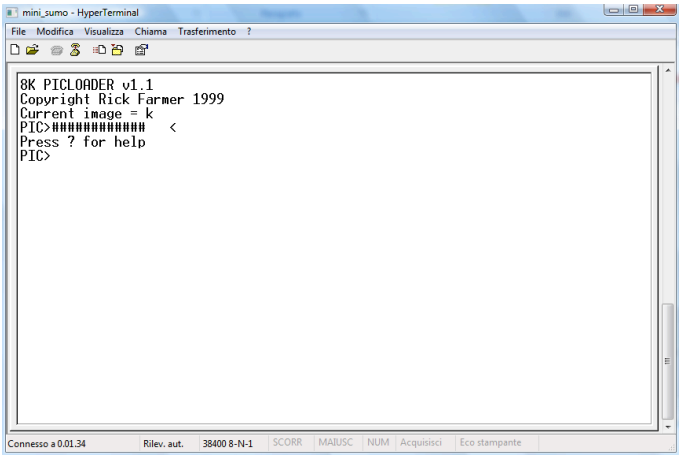
4. Settare le proprietà della porta COM come riportato in figura;



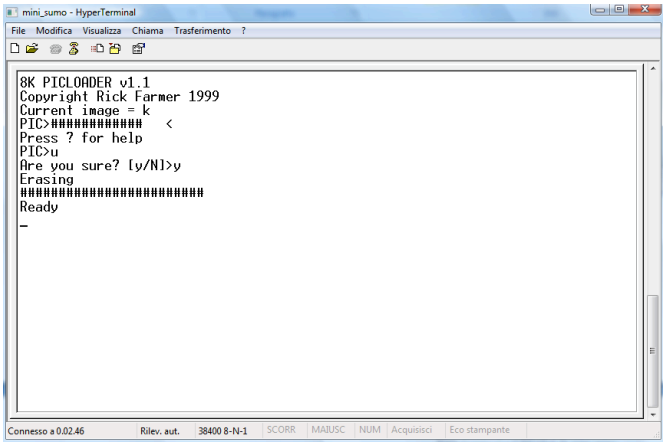
5. Settare l'intervallo di tempo di trasmissione tra una riga e la successiva (file – proprietà – impostazioni – impostazioni ASCII – intervallo riga = 50);



6. Accendere il robot;
7. Premere '?' entro 5 secondi;

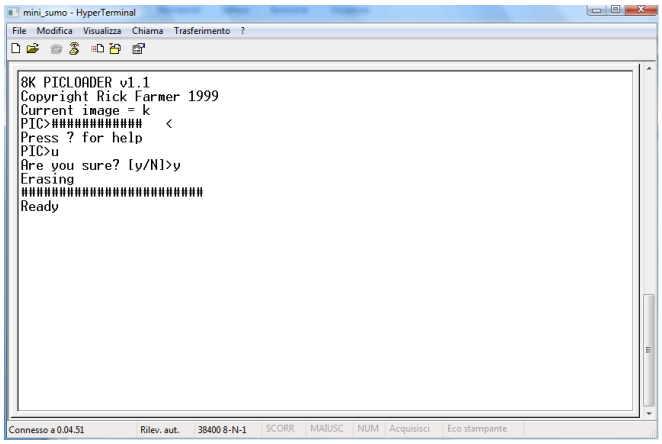


8. Premere 'u' (comando per resettare il contenuto della memoria) e poi confermare con 'y';

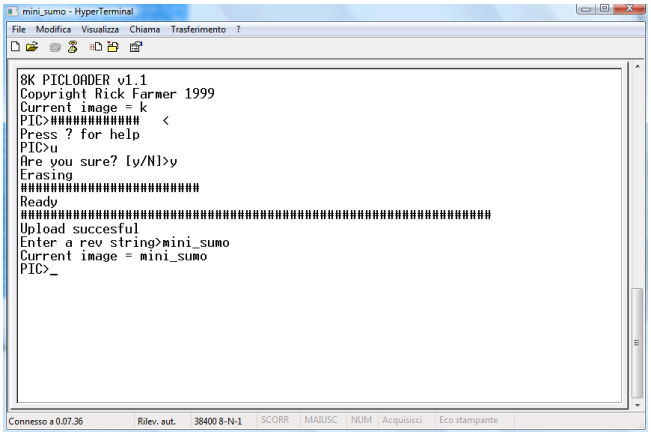


9. Aprire il file 'mini_sumo.hex', selezionare e copiarne tutto il contenuto;

10. Caricare il file nella memoria del PIC premendo il tasto destro e selezionando ‘incolla su host’;



11. Assegnare il nome al programma caricato;



4.3. Componenti necessari

Per lo sviluppo del progetto si sono utilizzati strumenti standard, classificabili in componenti hardware e software:

- **componenti hardware**: robot Mark III, cavo seriale per la connessione tra il calcolatore e il robot, arena di colore nero con una linea bianca di larghezza 2,5cm che ne delimiti i confini;
- **componenti software**: editor per la scrittura di un programma in C (nel nostro caso si è scelto MPLAB), compilatore C (CC5X), programma per comunicare con il robot tramite porta seriale (HyperTerminal).

4.4. Avvertenze

Durante la realizzazione del progetto non si sono riscontrati significativi problemi, nonostante alcuni fattori avrebbero potuto influenzare negativamente l’esito della programmazione.

Mark III sfrutta due diverse tipologie di batterie: 4 stilo di tipo AA da 1,5V, utilizzate per alimentare i motori e una batteria da 9V, utilizzata per la parte circuitale (sensori, PIC, ecc). La presenza di un led rosso, che segnali il livello di carica della batteria da 9V, permette di mantenere la stabilità della connessione PIC-calcolatore in modo da effettuare il caricamento del programma all’interno del PIC.

È inoltre fondamentale verificare la corretta impostazione de Hyper Terminal e del formato del file prodotto dal compilatore.

5. Conclusioni e sviluppi futuri

Lo scopo del progetto è stato quello di realizzare un programma specifico per Mark III; in particolare il robot doveva muoversi in maniera casuale all'interno di un'arena delimitata da una linea bianca e non urtare nessun tipo di ostacolo.

In considerazione delle numerose prove effettuate, durante le quali il robot è stato sottoposto alle condizioni più critiche (partenza sulla linea bianca, presenza di numerosi ostacoli nell'arena, etc) è possibile affermare che la soluzione adottata soddisfa le specifiche richieste.

È importante però evidenziare che l'esiguo numero di sensori a disposizione restringe notevolmente il campo visivo del robot. La situazione potrebbe sensibilmente migliorare con l'aggiunta di almeno due sensori di prossimità installati ai lati, grazie ai quali sarebbe possibile perfezionare l'algoritmo che consente di individuare ed evitare gli ostacoli. Inoltre se Mark III fosse dotato di sensori di prossimità posti nella parte posteriore, avrebbe a disposizione un campo visivo di ben 360°.

Infine, considerato che Mark III è un robot progettato per disputare le gare di mini-sumo, potrebbe essere vantaggioso includere il nostro algoritmo all'interno di una strategia di gara più complessa; si potrebbe, ad esempio, far evitare l'avversario quando si trova lontano dai bordi (in quanto sarebbe difficoltoso spingerlo al di fuori dell'arena), sferrando così l'attacco solo in determinate zone della pista.

Bibliografia

[1] Portland Area Robotics Society (PARTS): *http://www.portlandrobotics.org*.

[2] Competizioni minisumo: *http://www.minisumo.net*.

[3] Mark III User Guide: *http://www.junun.org/MarkIII/Manual/index.jsp*.

[4] Pic16F87x DataSheet: *http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf*.

[5] Rick Farmer PICLoader: *http://www.dontronics.com/rfarmer.html*.

[6] CH Basic: *http://thegnar.org/mIII/mIII_tool_chain.htm*.

[7] JAL: *http://www.voti.nl/jal/*.

[8] OOPic: *http://www.oopic.com/*.

[9] Sensori di linea: *http://www.junun.org/MarkIII/datasheets/QRB113x.pdf*.

[10] Sensori di prossimità: *http://www.junun.org/MarkIII/datasheets/GP2D12_15.pdf*.

[11] Gruppo MinisumoMarkIII: *http://tech.groups.yahoo.com/group/MiniSumoMarkIII/*.

[12] CC5x: *http://www.bknd.com/cc5x/*.

[13] CCS C: *http://www.ccsinfo.com/content.php?page=compilers*.

Indice