Mobile Robot

Francesco Argentieri*, Alessandro Luchetti†, Luca Nicolodi‡

Università Degli Studi di Trento, Dipartimento Ingegneria Industriale
C.D.L.M Ingegneria Meccatronica
Corso di Informatica e Programmazione

Abstract

Nel seguente report si espongono i risultati raggiunti nello sviluppo del progetto *Mobile Robot*, un robot mobile con azionamento differenziale a due ruote che presenta queste caratteristiche: line follower, riconoscimento incroci ed ostacoli sul tracciato e comandi da remoto.
Si è scelto di implementarne il funzionamento mediante *macchina a stati finiti*.

Introduzione

Nella prima sezione del report si presentano le scelte preliminari effettuate al fine di soddisfare al meglio le caratteristiche di base richieste. Nella seconda sezione si descrive la struttura del codice realizzato soffermandosi sui punti chiave che ne hanno determinato lo sviluppo. Infine nella conclusione si riassumono i risultati raggiunti.

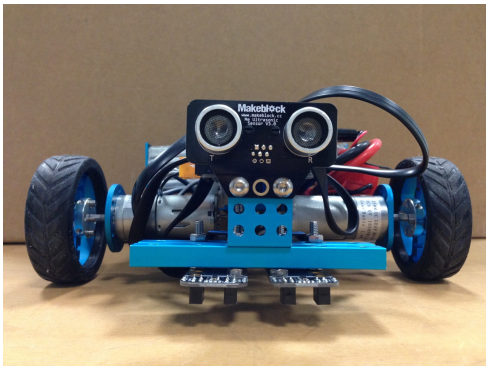
1 Scelte progettuali

Il progetto può essere suddiviso in due parti: una, inerente alla scelta ed assemblaggio dei componenti meccanici ed elettronici, mentre l'altra relativa allo sviluppo del software di controllo. Ciascuna di esse viene analizzata successivamente nei prossimi paragrafi.

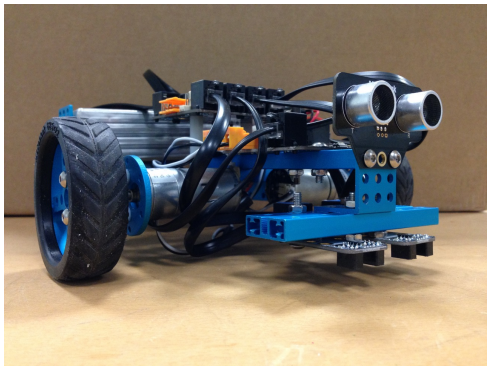
*ID: 183892 mail: francesco.argentieri@studenti.unitn.it
†ID: 180061 mail: alessandro.luchetti@studenti.unitn.it
‡ID: 179877 mail: luca.nicolodi@studenti.unitn.it

1.1 Hardware

Non avendo vincoli specifici si è scelto di utilizzare un struttura semplice come riportata in figura 1, ma che potesse al tempo stesso soddisfare tutte le funzionalità prefissate.



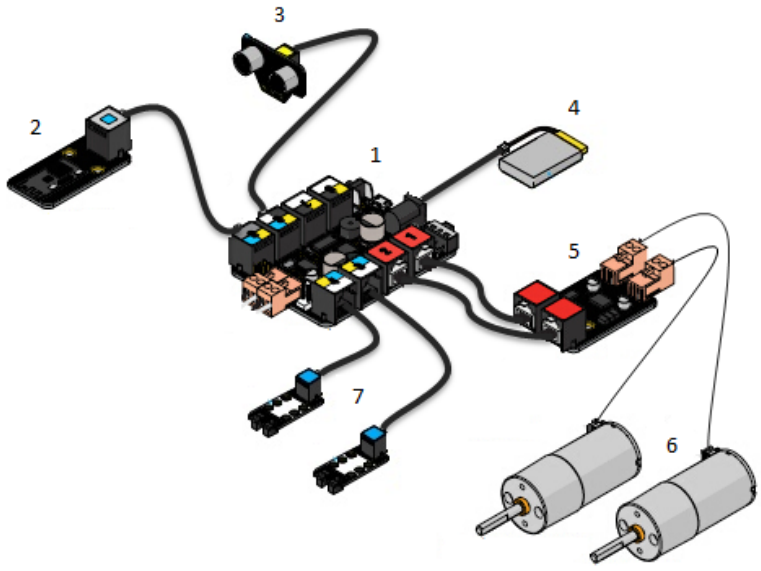
(a) *vista frontale*



(b) *scorcio*

Figura 1: Struttura robot

La parte elettronica è basata sulla scheda MeOrion baseboard, un modulo a ultrasuoni per il rilevamento degli ostacoli, due moduli per i sensori tipo MeLineFollower, un modulo bluetooth. In figura 2 è illustrato il dettaglio dei collegamenti. La scelta di adottare una coppia di sensori di linea è stata fatta per ottenere una migliore lettura del percorso, per una maggiore capacità di identificazione di casi particolari, come ad esempio curve a 90° o gli incroci e per un miglior controllo dei motori.



(a) *collegamenti*

1	Me Orion
2	Me Bluetooth
3	Line Follower
4	Me Ultrasonic Sensor
5	Batteria Li-Po
6	Me Dual DC MotorDriver
7	Me DCMotor

(b) *lista dei compomenti*

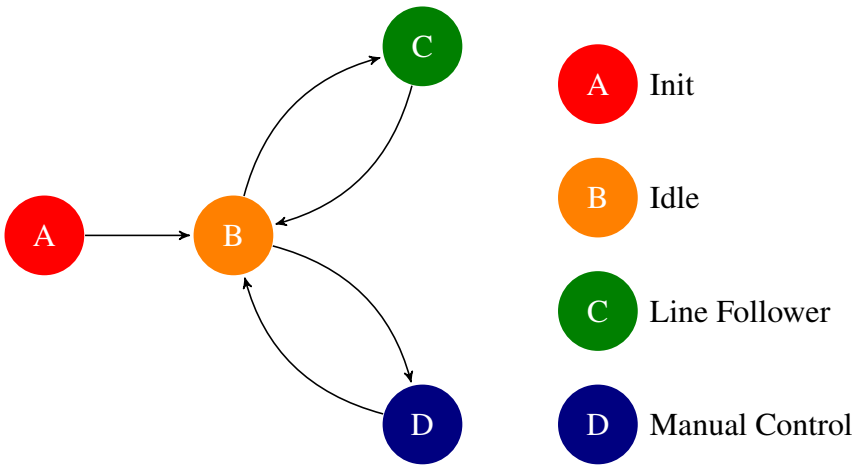
Figura 2: Componenti elettronici

1.2 Software

L'implementazione del software, basata sul C++, è il cuore del progetto. Ci si è avvalsi delle librerie proprietarie Makeblock, della classe macchina a stati finiti, precedentemente realizzata, e di metodi appositamente elaborati per una corretta gestione delle funzionalità richieste.

2 Macchina a stati finiti

Come riferimento in figura 3 sono stati individuati 4 stati per la gestione dell'automa. Ogni stato chiama le funzioni che vengono eseguite per espletare il proprio compito ed appositi metodi regolano la transizione da uno stato all'altro al verificarsi di specifiche condizioni.
Line Follower e *Manual Control* rappresentano i due stati principali che permettono rispettivamente una guida autonoma sul tracciato o un controllo manuale. Lo stato *Idle* è uno stato di attesa del robot in cui in ogni momento è possibile accedervi su decisione dell'utente e in cui lo stesso può scegliere quale modalità di funzionamento adottare. Lo stato *Init*, infine, è solamente uno stato di inizializzazione all'avvio del robot.



(a) *schema concettuale* (b) *legenda*

Figura 3: Macchina a stati finiti

2.1 Line Follower

Nella modalità line follower in ogni istante viene effettuata la verifica della presenza ostacoli mediante sensore ad ultrasuoni. Qualora non venisse riscontrato un pericolo di collisione, entro una prefissata distanza di sicurezza, l'algoritmo prosegue con la lettura del tracciato altrimenti impartisce al robot il comando di invertire il moto. Al contempo la coppia di sensori di linea

istantaneamente legge il tracciato e lo codifica secondo la tabella 1, attribuendo un errore che serve a correggere la velocità dei motori mediante un controllo PID[1].

		DX SX	SX DX	Errore
GO FORWARD	==	1, 2	O X X O	0
TURN LEFT VERY SOFT	==	3, 2	O X O O	1
TURN LEFT SOFT	==	1, 0	X X X O	2
TURN LEFT HARD	==	3, 0	X X O O	3
TURN LEFT VERYHARD	==	3, 1	X O O O	4
TURN RIGHT VERY SOFT	==	1, 3	O O X O	-1
TURN RIGHT SOFT	==	0, 2	O X X X	-2
TURN RIGHT HARD	==	0, 3	O O X X	-3
TURN RIGHT VERYHARD	==	2, 3	O O O X	-4
NO LINE	==	3, 3	O O O O	5
CROSS	==	0, 0	X X X X	6

Tabella 1: Codifica input dei sensori di linea (X = tracciato nero; O = sfondo bianco)

Per garantire una maggiore robustezza al programma sono state considerate anche le casistiche eccezionali che coprono eventuali guasti o malfunzionamenti (ad esempio linea a contrasto invertito, sporco sui sensori o sul tracciato), come mostrato in tabella 2.

		DX SX	SX DX	Errore
GO FORWARD bis	==	2, 1	X O O X	0
EXCEPTION1	==	2, 0	X X O X	1
EXCEPTION2	==	0, 1	X O X X	-1
EXCEPTION3	==	2, 2	O X O X	0
EXCEPTION4	==	1 , 1	X O X O	0

Tabella 2: Codifica CASI ECCEZIONALI (X = tracciato nero; O = sfondo bianco)

2.1.1 Controller PID

Per permettere al robot di seguire in maniera fluida il tracciato è stato implementato un controllo PID come illustrato in figura 4.

Tale controllo agisce sulla velocità delle singole ruote apportando una correzione che dipende dall'errore e dai guadagni K_p, K_d, K_i determinati sperimentalmente. L'attribuzione di un valore a questi parametri ha richiesto numerosi test, data la loro elevata influenza sull'andatura del robot. Poiché alcuni di essi risultano migliori per un determinato tracciato piuttosto che un altro si è cercato di trovare il compromesso che garantisse una maggiore flessibilità nell'affrontare differenti percorsi. Tali guadagni permettono di calcolare la correzione da apportare ai motori

[1]Proporzionale-Integrativo-Derivativo, comunemente abbreviato come PID

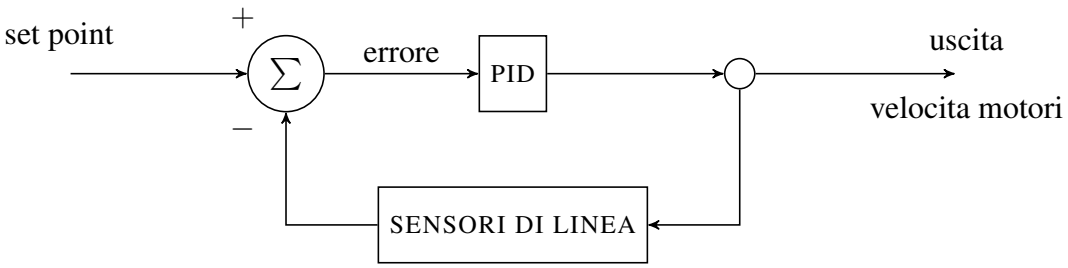


Figura 4: Schema a blocchi PID

mediante la seguente equazione 1:

$$PID_{value} = K_p\, e(t) + K_d \frac{de(t)}{dt} + K_i \int_0^t e(\tau)d\tau \tag{1}$$

Grazie a questo controllo il robot viene mantenuto allineato al percorso come si può graficamente apprezzare in un esempio di figura 5.

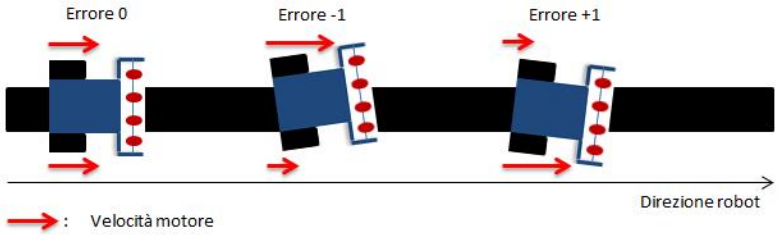


Figura 5: Esempio correzione mediante PID

2.1.2 Riconoscimento incroci e curve ad angolo retto

Un incrocio viene individuato quando tutti i sensori rilevano la linea. In tale condizione all'utente viene richiesto, mediante apposito comando, di scegliere la direzione da far intraprendere al robot.

Un problema particolare si è presentato nel riconoscimento delle curve ad angolo retto nelle quali, i sensori di linea, a causa della velocità del robot, uscivano brevemente dal tracciato perdendone la lettura. Per garantire comunque una velocità sostenuta del robot è stato necessario dotare il sistema di una memoria che tenesse conto della direzione di uscita e ne garantisse il rientro dalla parte corretta.

2.2 Manual Control

In tale modalità l'utente può controllare da remoto il robot tramite la connettività bluetooth. Mediante l'utilizzo della tastiera del Pc è possibile guidare il robot nelle 4 direzioni principali

agendo direttamente sui motori. Il sistema garantisce continuamente un controllo attivo della presenza ostacoli ed in tale condizione ignora un comando errato dell'utente invertendo il moto ed evitando la collisione.

2.2.1 Trasmissione comandi da remoto

Per l'interfaccia con l'utente si è deciso di non utilizzare il serial monitor dell'ambiente Arduino ma di utilizzare il terminale Putty che permette un invio sequenziale dei comandi senza richiedere la pressione del tasto Enter.

3 Conclusione

Lo sviluppo del software ha permesso di soddisfare le richieste di line following infatti il robot riesce autonomamente ad adattarsi a diversi tipi di tracciato e muoversi anche nei casi più difficili. Il setup utilizzato nel controllore PID permette di spingere il robot al 55% della velocità massima consentita dai motori senza incorrere in slittamenti o sbandate. Permette all'utente di interagirvi con la possibilità di scegliere la direzione di svolta nel caso di incrocio e pilotarlo da remoto tramite conessione bluetooth. Infine in tutti i casi il robot evita autonomamente l'impatto con gli ostacoli.

Riferimenti bibliografici

1. J. Hespanha, *Linear Systems Theory* (Princeton University Press, 2009).
2. http://makeblock.com/en.
3. http://en.cppreference.com/w/.
4. M. Schmidt, *Il manuale di Arduino*, Guida completa (Apogeo, 2011).