Università degli studi di Padova

Dipartimento di Tecnica e Gestione dei Sistemi Industriali

Laurea Triennale in ingegneria
Meccanica e Meccatronica
Curriculum Meccatronico

Realizzazione di un robot d'assistenza alla gente

RELATORE: **PROF. Roberto OBOE**

LAUREANDO: **KEUMEJIO Pris - Parfait**

ANNO ACCADEMICO: **2012/2013**

Sommario

pagine

Tabella delle figure

Introduzione

Questo progetto è stato realizzato durante il mio primo anno (2011-2012) di doppia laurea all'Ecole Centrale Lyon (Francia) nell'ambito del progetto TIME (Top Industrial Managers for Europe). Eravamo un gruppo di quattro studenti e abbiamo lavorato insieme sul progetto per otto mesi. Il progetto era la continuazione del lavoro cominciato da altri studenti per gli anni precedenti, l'obiettivo comune essendo la realizzazione di un robot d'aiuto alla gente.

Al fine di fornire un' assistenza quotidiana a persone anziane e disabili nella nostra società tecnologicamente avanzata, la robotica appare naturalmente come una soluzione interessante. Molti progetti per la realizzazione di robot d'aiuto alla gente sono già emersi: per esempio, ASIMO (figura 1) sviluppato da Honda. Attraverso sensori multipli (visivi, ultrasuoni, infrarossi, forza), è in grado di rilevare movimenti, posture, suoni, riconoscere visi e afferrare oggetti come vassoi.

Figura 1 : Robot ASIMO sviluppato da Honda.

L'obiettivo del progetto è di realizzare un robot che può identificare un dato oggetto, approcciarlo e quindi afferrarlo. Il robot è costituito da una base mobile , un braccio articolato e una fotocamera autonoma. La velocità d'approccio dell'oggetto non è un fattore rilevante. Inoltre, l'oggetto da afferrare è caratterizzato da dimensioni adattate alle dimensioni della pinza del braccio, da una forma "omogenea" (una sfera o un cubo, per esempio) e un colore chiaramente distinto dall'ambiente.

Il progetto è organizzato in tre capitoli:

Capitolo 1 : Descrizione del robot.

- La scheda madre.
- Il braccio.
- La fotocamera.
- La Base mobile.
- Il sistema di alimentazione.

Capitolo 2 : Le interconnessioni.

- Fotocamera - scheda madre.
- Braccio - scheda madre.
- Fotocamera - base mobile.

Capitolo 3 : Strategia per afferrare un oggetto.

- Inseguimento dell'oggetto dalla fotocamera.
- Controllo del braccio per afferrare l'oggetto.

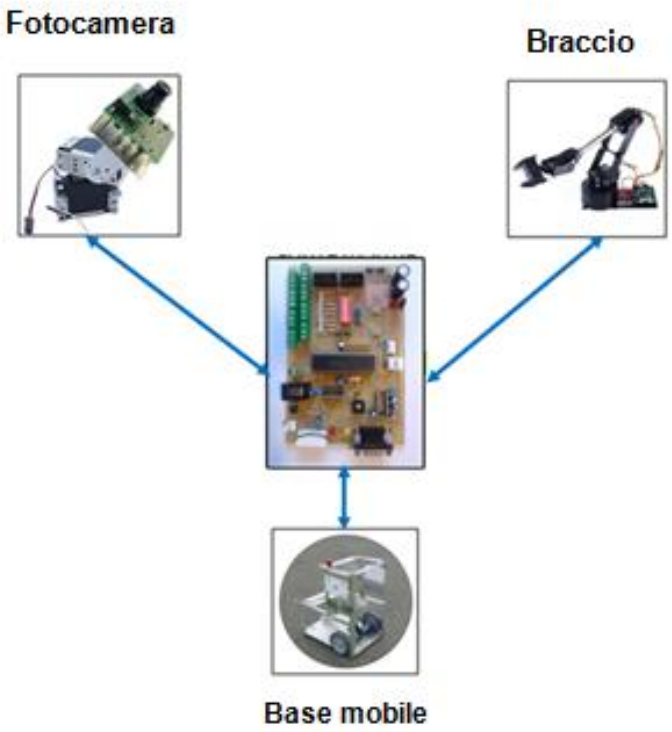


Figura 2 : Diversi elementi del robot con le interconnessioni.

CAPITOLO 1 : Descrizione del robot

L'obiettivo di questo capitolo è di presentare separatamente i diversi elementi che compongono il robot e il loro funzionamento.

1. La scheda madre

Questo paragrafo presenta la scheda detta "madre" e in particolare il microcontrollore PIC.

1.1. Presentazione e funzionamento

La cosiddetta scheda "madre" è la scheda elettronica centrale del robot. Essa gestisce la comunicazione tra la scheda elettronica del braccio, quella della fotocamera e quella della base mobile. Le schede elettroniche del braccio e della base mobile seguono ordini e quindi saranno chiamate schede "schiavi". La figura seguente rappresenta la scheda madre. Essa riceve le informazioni inviate dalla fotocamera, elabora ed invia i comandi alle ruote per il posizionamento della base mobile poi al braccio per afferrare un dato oggetto.

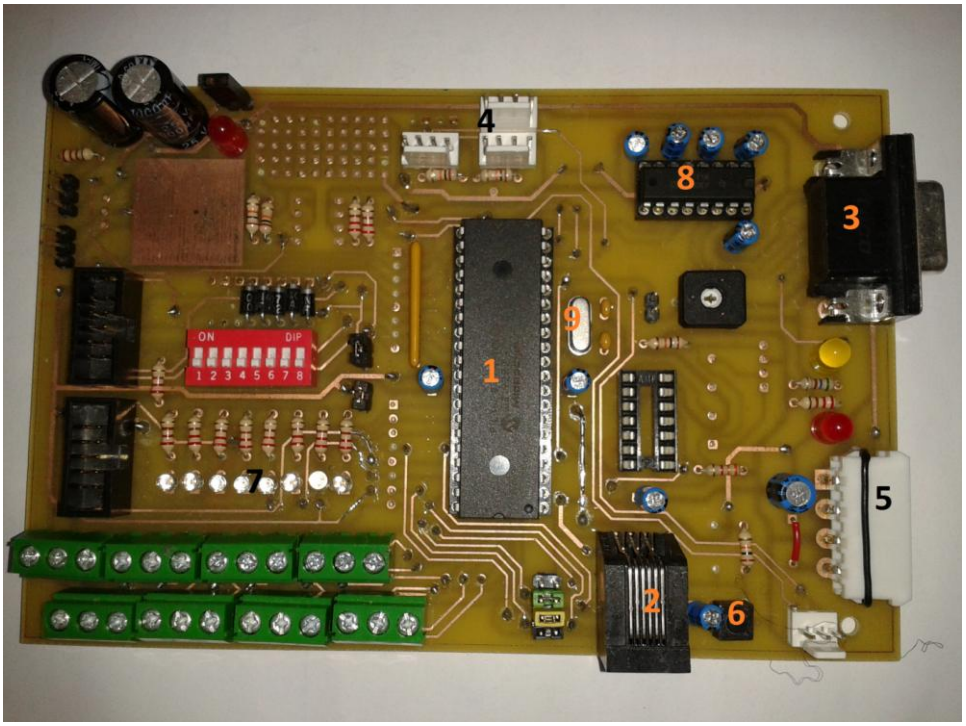


Figura 3 : Fotografia della scheda madre.

Molti elementi elettronici compongono la scheda madre:

1- **Il microcontrollore PIC16F877A :** è il "cervello" della scheda. Esso contiene il programma in grado di elaborare le informazioni. Con i suoi quaranta piedi (ingressi, uscite) chiamati “pins” può ricevere e inviare informazioni. È programmabile nel linguaggio C collegandolo al computer.

2- **Il connettore RJ11 :** permette di trasmettere il programma dal software MPLAB (installato nel computer) verso il microcontrollore.

3- **La porta seriale RS232 :** permette la comunicazione con la scheda di controllo della fotocamera.

4- **I connettori I^2C** (Inter Integrated Circuit) : permettono la comunicazione tra le schede di controllo del braccio e della base mobile.

5- **La presa Molex :** fornisce alimentazione alla scheda. Il cavo di alimentazione è composto da uno strato di quattro fili: il filo rosso di +5 V, il filo giallo di +12 V e due fili neri a terra.

6- **Il tasto RESET :** permette di ripristinare il programma già presente nel microcontrollore.

7- **LED** (Light Emitting Diode) : I diodi di colore bianco presenti al numero 7 emettono un segnale luminoso quando un segnale viene inviato ai “pins” dove sono connessi. Essi sono utilizzati per verificare il funzionamento del programma e identificare potenziali problemi.

8- **Convertitore Max232 :** converte i segnali che arrivano attraverso la porta seriale (-12V /+12V) in segnali leggibili dal microcontrollore (0 V / +5 V)

9- **Quarzo :** è un oscillatore a 20MHz (20.10^6 Hertz) di frequenza che permette di calcolare più precisamente il tempo di ciclo del PIC (cioè il tempo di esecuzione di una operazione).

Tanti altri elementi sono presenti su questa scheda (le resistenze, i condensatori, connettori per segnali analogici, jumpers, diodi che segnalano se la scheda è sotto tensione).

1.2. Il microcontrollore PIC16F877A

1.2.1. Generalità sui microcontrollori PIC

I microcontrollori PIC sono una famiglia di microcontrollori della la società Microchip. Un microcontrollore è una unità di elaborazione del tipo microprocessore al quale si sono aggiunti dispositivi interni per renderlo autonomo (può operare in modo indipendente dopo la programmazione). In questo senso, il PIC è particolarmente ben dotato, perché contiene una memoria di programma, una memoria dati, ingressi e uscite e un orologio che gestisce l'esecuzione del programma. I modelli attuali di PIC sono identificati per i seguenti riferimenti : due cifre che denotano la famiglia del PIC

(10, 12, 16, 17, 18, 24 ...) ed una lettera che menziona il tipo di memoria di programma (C o F). La F indica che si tratta di una memoria flash e quindi cancellabile elettronicamente. La lettera C indica che la memoria può essere cancellata solo per esposizione ultravioletta. I caratteri che seguono il tipo di memoria denotano la versione del PIC.

1.2.2. Orologio al quarzo

Il microcontrollore ha un orologio interno, ma è possibile controllare la sua frequenza di lavoro esternamente. Si può citare in particolare l'oscillatore al quarzo, che ha il vantaggio di oscillare ad una frequenza stabile. Quando si applica una differenza di potenziale tra due elettrodi posti nel quarzo, è sottoposto ad un campo elettrico che provoca la sua deformazione dovuta all'effetto piezoelettrico. Quando questo campo viene tolto, il quarzo riprende la sua forma iniziale. Durante questa deformazione, il quarzo a sua volta genera un campo elettrico grazie di nuovo all'effetto piezoelettrico, creando così una differenza di potenziale tra i due Elettrodi. L'alternanza di questi due stati è stabilizzato su una delle due frequenze naturali del quarzo. La frequenza di risonanza dipende dalla forma e dimensioni del quarzo. Il ciclo di istruzione (che corrisponde ad una operazione) di un microcontrollore PIC dura quattro colpi orologio. Così, la frequenza dell'orologio interno del PIC è ottenuto dividendo per quattro quella del quarzo. Con un Quarzo di frequenza 20 MHz (quello che usiamo), il ciclo di istruzione è allora di 0,2 microsecondo.

1.2.3. Struttura e funzionamento del microcontrollore PIC16F877A

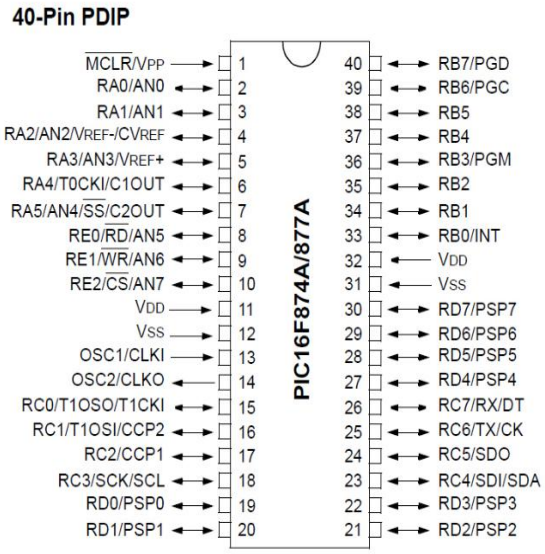


Figura 4 : Il microcontrollore PIC16F877A (estratto del documento tecnico).

Il PIC comunica con l'esterno attraverso connettori chiamati “pins”. Il linguaggio di comunicazione tra i diversi elementi del PIC e con l’esterno è il binario, quello vuol dire che esso invia sequenze di 0 e 1. Specificamente, 1 binario corrisponde ad una tensione di +5 V e 0 binario ad una tensione di 0 V. Come si può vedere dal grafico sopra, la maggior parte dei “pins” sono bidirezionali, in modo da poter scegliere di fare ingressi o uscite. Inoltre, notiamo che alcuni “pins” sono riservati ad applicazioni specifiche :

- Le porte 18 e 23 sono usate per la comunicazione I^2C,
- Le porte 25 e 26 per il collegamento seriale RS232,
- Le porte 11, 12, 31 e 32 per l’alimentazione in tensione del PIC ... ecc.

Altre informazioni sul PIC16F877A sono disponibili sul suo documento tecnico (“datasheet”) fornito dal costruttore o scaricabile gratuitamente dal sito di Microchip(http://www.microchip.com). La tabella seguente riassume le sue caratteristiche principali.

Parameter name	**Value**
Program memory type	Flash
Program memory (KB)	14
CPU speed (MIPS)	5
RAM (bytes)	368
Data EEPROM (bytes)	256
Digital communication periherals	1-A/E/USART,I-MSSP(SPI/I2C)
Capture/compare/PWM peripherals	2CCP
Timers	2x 8- bit , 1x16-bit
ADC (Analog to Digital Converter)	14 ch,10-bit
Comparators	2
Temperature range(°C)	- 40 to 125
Operatiing voltage range(V)	2 to 5.5
Pin count	40
Features	High sink/source current 25mA Self programming Parallel Slave port

2. Il braccio

2.1. Presentazione

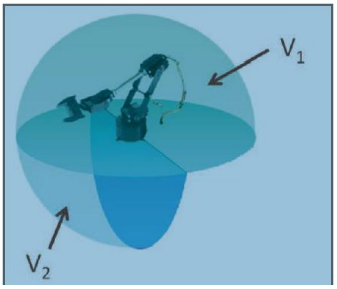
È un braccio costruito dalla società Lynxmotion e di modello AL5D. Esso a quattro gradi di libertà perché comporto da quattro servomotori (0,1,2,3 come mostrato nella figura 5) che permettono di posizionare la pinza in un punto desiderato. Il motore 4 permette la rotazione della pinza e il motore 5 assicura la sua apertura e chiusura. Per convenienza abbiamo scelto di dare dei numeri da 0 a 5 ai diversi attuatori . Gli attuatori da 0 a 4 consentono rotazioni tra due elementi diversi del braccio, invece l'attuatore 5 consente l'apertura e chiusura della pinza.

Figura 5 : Braccio Lynxmotion AL5D.

2.2. Volume di lavoro

Il volume di lavoro del braccio è lo spazio nel quale può afferrare un oggetto. Se supponiamo il braccio fisso, il suo volume di lavoro è stimato a $\frac{3}{4}$ di una sfera centrata nella sua base come lo mostra le figura seguente. V_1 è una semi-sfera e V_2 è un quarto di sfera.

Nota: il calcolo seguente est un'approssimazione del volume e non un valore esato.



Il volume totale vale $V = V_1 + V_2$

$V_1 = \frac{2}{3}\,\pi\, R_1^3$; $R_1 = 47mm$

Quindi $V_1 = 217446{,}4mm^3$

$V_2 = \frac{1}{3}\,\pi\, R_2^3$; $R_2 = 33{,}5mm$

Quindi $V_2 = 39369{,}8mm^3$

In fine $V = 256816{,}2mm^3$

Figura 6 : Volume di lavoro de braccio.

2.3. Funzionamento

Per assicurare il funzionamento corretto del braccio, è necessario controllare tutti i servomotori simultaneamente. Infatti, un servomotore svilluppa una coppia se solo se riceve un segnale di controllo. Per quello, se si vuole raggiungere un qualsiasi punto nel volume di lavoro, è importante tener sotto controllo tutti gli attuatori del braccio.

2.3.1. Principio del servomotore

Un servomotore è un attuatore asservito in posizione, cioè esso segue delle posizioni angolari rispetto al comando esterno ricevuto. La figura seguente mostra un servomotore e qualche esempio di segnale di controllo rappresentato su un periodo con l'angolo corrispondente.

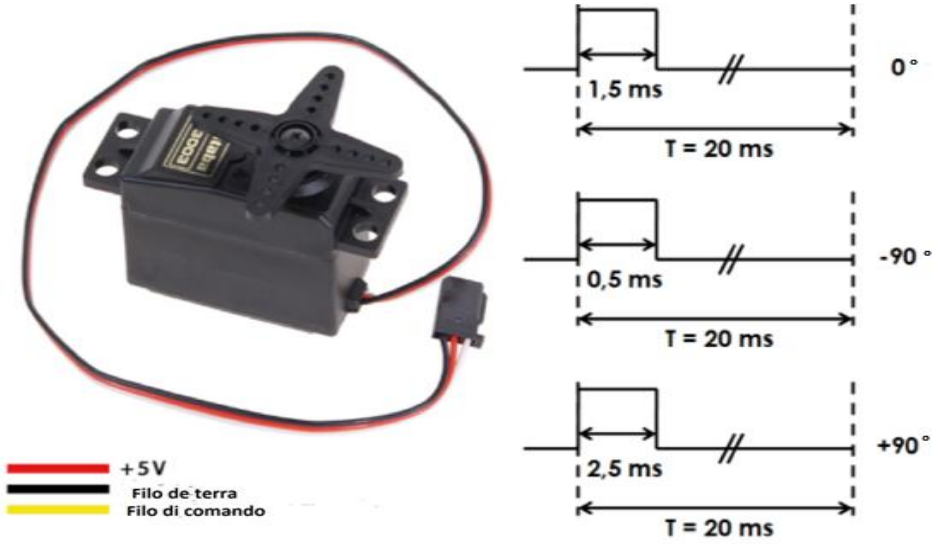


Figura 7 : Servomotore con il suo segnale di controllo.

Il segnale d'ingresso dei servomotori in questo caso è una finestra rettangolare di periodo 20 millisecondi. La posizione angolare che si vuole dare al servomotore è una funzione del rapporto ciclico, cioè il rapporto tra il tempo di messa a 1 (+5 V) durante un periodo e il periodo. Ogni servomotore a la sua funzione di commando.

Esempio: Per il servomottore 0 la relazione tra il tempo(in microsecondi) e l'angolo(in gradi) è la seguente: t = 1500 + (2000xangolo/180).

Il vantaggio è che il segnale di controllo è a corrente bassa. Infatti, l'alimentazione elettrica è fornita dal figlio rosso e nero, che permette il collegamento diretto del servomotore su un'uscita del PIC senza circuito di interfaccia.

2.3.2. La scheda elettronica del braccio

Gli attuatori del braccio sono controllati da una scheda elettronica con un microcontrollore PIC16F737. Questo PIC è simile a quello della scheda madre tranne che non ha la porta seriale RS232. Le Porte B5, B4, B3, B2, B1 e B0 del PIC16F737 sono collegati rispettivamente agli attuatori 5, 4, 3, 2, 1 e 0.

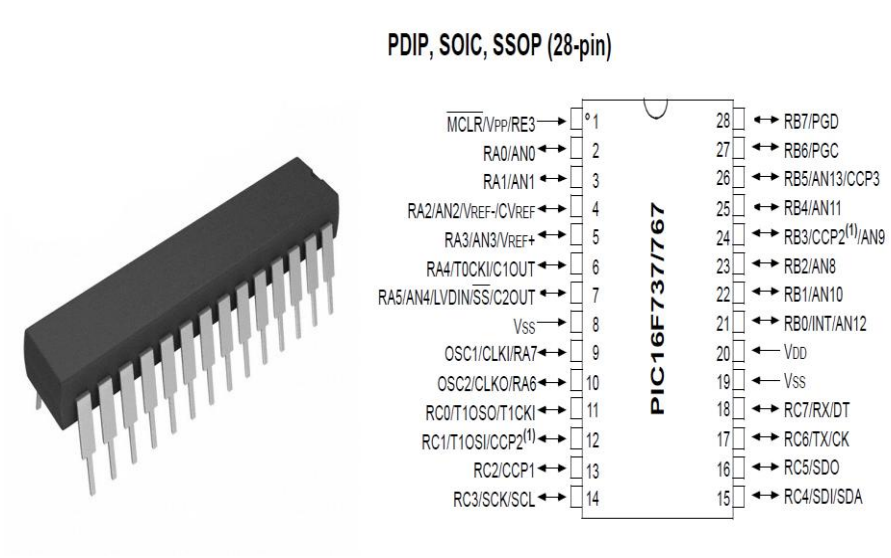


Figura 8 : Estratto del documento tecnico del PIC16F737.

Figura 9 : Scheda di controllo del braccio.

Come la scheda madre, questa è composta da :

- Il microcontrollore PIC16F737.
- Il quarzo.
- Il tasto di ripristina.
- Il connettore per l'alimentazione (+5V).
- Il connettore per la comunicazione I^2C.
- Il connettore RJ11 per il collegamento al computer.
- I connettori per I servomotori.
- 3 LED di verifica
- Resistori e condensatori.

Le caratteristiche del microcontrollore PIC16F737 sono consegnate nella seguente
tabella:

Parameter name	**Value**
Program memory type	Flash
Program memory (KB)	7
CPU speed (MIPS)	5
RAM (bytes)	368
Data EEPROM (bytes)	256
Digital communication periherals	1-A/E/USART,I-MSSP(SPI/I2C)
Capture/compare/PWM peripherals	3CCP
Timers	2x 8- bit , 1x16-bit
ADC (Analog to Digital converter)	11 ch,10-bit
Comparators	2
Temperature range(°C)	-40 to 125
Operatiing voltage range(V)	2 to 5.5
Pin count	28
Features	1PWM 10-bit Programmable Low Current LVD Multiple oscillator options

3. La fotocamera

3.1. Présentazione

La fotocamera usata nel robot è una **CMUCam3** sviluppata dall'università di Carnegie Mellon (Penssylvania). Essa è costituita da:

- un sensore CMOS collegato ad un microcontrollore LPC2106 del costruttore NPX.
- Un supporto avendo due servomotori che permettono la rotazione della fotocamera lungo due assi: orizzontale e verticale come mostrato nella Figura 10.

Nella nostra applicazione, Il ruolo della fotocamera è di inviare la posizione dell'oggetto (sotto forma di coordinate cartesiane nel sistema di coordinate della fotocamera) alla scheda madre.

Figura 10: Fotocamera montata sulla torretta.

4. La base mobile

4.1. Présentazione

Per spostarsi, il robot si serve di una base con due ruote motori e un sistema di palle montate agli angoli dalla base per assicurare l'equilibrio. Per questione di minimizzazione del peso, il materiale scelto per la base è l'alluminio per il fondo e gli angoli e la plastica per i bordi laterali. Le ruote hanno un diametro di 100mm e sono connesse a dei moto-riduttori (motori con un sistema di riduttori di movimento e un sensore di posizione). I motoriduttori sono alimentati a 12 V e Il loro controllo è assicurato dalla scheda elettronica MD25 a sua volta connessa alla scheda madre tramite un collegamento I^2C.

Figura 11 : Base mobile.

Figura 12 : Scheda MD25.

5. Il sistema di alimentazione

Certi elementi del robot richiedono una tensione d'alimentazione di 5 V (microcontrollori, servomotori, LED ...) e altri 12 V (moto-riduttori). Per quello abbiamo scelto di usare per il robot il sistema di alimentazione di un computer che genera direttamente due tensioni in uscita : 5 V e 12 V. Abbiamo scelto questa soluzione solo per motivo di mancanza di tempo per realizzare il proprio dispositivo di alimentazione.

Figura 13 : Dispositivo di alimentazione 5V/12V

CAPITOLO 2 : Le interconnessioni

Questo capitolo descrive il modo in cui i diversi elementi del robot comunicano tra di loro per assicurare il suo funzionamento. Due tipi di comunicazione sono trattati:

- Il collegamento seriale.
- Il collegamento I^2C.

1. Comunicazione tra fotocamera e scheda madre

1.1. Protocollo di comunicazione seriale

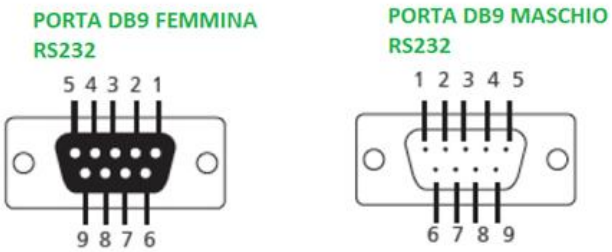
La comunicazione tra la fotocamera e la scheda madre è realizzata tramite un collegamento seriale RS232. In questa parte, si descrive il protocollo di comunicazione e si presenta un codice tipo per l'invio di un byte di dati dalla fotocamera verso la scheda madre.

Figura 14 : Comunicazione tra fotocamera – scheda madre

1.1.1. Il collegamento RS232

Nella comunicazione seriale, i “bits”(serie di 0 e 1) d’informazione sono trasmessi successivamente. Questa trasmissione si può fare ad intervalli regolari utilizzando un orologio (trasmissione sincrona), oppure ad intervalli casuali (trasmissione asincrona). Nel collegamento seriale RS232 usato in questo caso, la trasmissioni è asincrone.

La seguente figura descrive la porta seriale.



PIN	RS-232	
1	DCD	Detezione di un segnale nella linea
2	TxD	Trasmissione dei dati
3	RxD	Ricezione dei dati
4	DSR	Pronto per la ricezione dei dati
5	GND	Filo di terra
6	DTR	Dati pronti
7	CTS	Pronto per la trasmissione
8	RTS	Richiesta di trasmissione

Il piede 9 (RI=Ring Indicator), è un' indicatore di suoneria ma non serve nella trasmissione dei dati in questo caso.

Figura 15 : Porta seriale RS232.

Va inoltre osservato che la logica 0 e 1 usata in questa comunicazione è molto diversa da quella usata nel PIC (dove 0 corrisponde a 0 V e 1 a 5 V). Nella trasmissione seriale usata qui , il numero logico 1 è rappresenta tensioni comprese tra -25 V e - 3 V, e il 0 le tensioni comprese tra +3 V e +25 V. Ciò richiede un componente aggiuntivo, il convertitore MAX232 che converte il segnale inviato dal PIC in un ordine "comprensibile" dal protocollo seriale e viceversa.

1.1.2. Invio di un byte di dati

Il collegamento seriale rimane sempre a 1 finché non viene inviato un dato nelle linea.

Per inviare un byte di dati, si deve seguire i tre passi seguenti:

- Invio di un bit di avvio (0 logico) per iniziare la trasmissione.
- Invio del byte da trasmettere bit per bit a partire dal bit meno significativo.
- Invio di un bit di arresto (1 logico) per completare la trasmissione.

Collegando la fotocamera tramite la porta RS232 all'oscilloscopio, si può osservare il segnale mostrato nella figura seguente. Nell'esempio, la fotocamera invia il carattere 'b'.

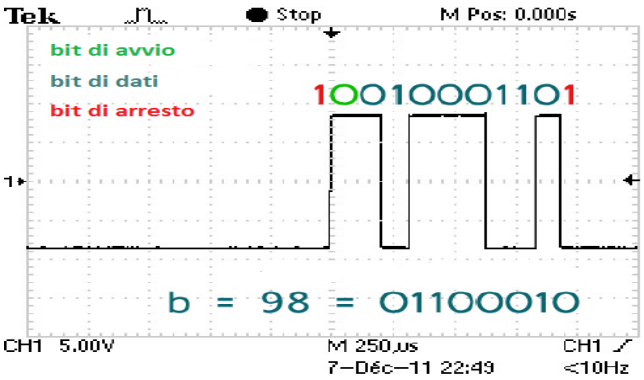


Figura 16 : Visualizzazione dell'invio del carattere "b" dalla fotocamera.

Nota : si vede nella figura che 1 corrisponde ad una tensione negativa e 0 ad una tensione positiva. Questo è la logica negativa.

Il codice seguente corrispondente all'invio di un byte(il carattere "b") tramite la RS232.

Codice impostato sulla scheda della fotocamera

```
// Dichiarazione delle biblioteche
#include <stdbool.h> // biblioteca dei booleani
#include <stdlib.h> // biblioteca di funzioni del linguaggio C
#include <ctype.h> // bibioteca dei tipi di variabili del linguaggio C
#include <math.h> // biblioteca delle funzioni matematiche
#include <cc3.h> // biblioteca delle funzioni della fotocamera
#include <cc3_ilp.h>//altra biblioteca delle funzioni della fotocamera
//Programma principale
int main (void){
uint32_t val;  // variabile usata per l'inizializzazione della fotocamera
cc3_camera_init (); // inizializzazione della fotocamera
//Inizializzazione della porta seriale
cc3_uart_init (0,CC3_UART_RATE_9600,CC3_UART_MODE_8N1,
CC3_UART_BINMODE_BINARY);
setbuf(stdout, NULL);
setbuf(stdin, NULL);
```

```
//Ciclo infinito per l'avvio de carattere 'b'
while (1){
cc3_led_set_state(2,true); // accensione della  LED  sinistra
cc3_timer_wait_ms (2000);// aspettare 2000 ms
putchar('b'); //invio del carattere  "b"
cc3_timer_wait_ms (1000); // aspettare 1000ms
 }
return 0;
```

La velocità di trasmissione del trasmettitore deve essere identica alla velocità di acquisizione del ricevitore. Si esprime le velocità di trasmissione in baud : 1baud = 1 bit / secondo. Ci sono diverse velocità standard: 9600, 4800,2400, 1200. . . bauds. Nel nostro caso, si usa una velocità classica di 9600 bauds.

Il codice seguente è quello impostato sulla scheda madre.

```
#include <16f877a.h>  // dichiarazione del PIC della scheda madre.
#fuses HS, NOWDT, LVP // definizione dei fusibili
#use delay (clock=20000000) // frequenza del PIC
//Definizione della porta seriale
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
//Programma principale
void main()
{
char d=0;  // variabile di memorizzazione dei dati ricevuti
d=getc(); // appena un dato è trasmesso, è subito memorizzato
}
```

L'istruzione getc() o getchar() può recuperare un byte in arrivo dal collegamento seriale. Questa funzione è presa in conto solo quando appaiono i dati sulla porta seriale. In altre parole, con questo codice, il PIC può aspettare per un tempo indeterminato finché un byte non è inviato prima di passare alla riga successiva. Per risolvere il problema, è più conveniente usare la funzione di "interruzione" specifica al collegamento seriale RS232 e così, quando appare un byte sulla porta seriale, l'esecuzione del programma corrente viene interrotta e quindi esegue questa funzione "Interruzione". Alla fine, il programma continua dove esecuzione è stata interrotta. Il codice seguente descrive la funzione di interruzione usata in questo caso.

Codice d'interruzione impostato sulla scheda madre

```
#include <16f877a.h> //  definizione del PIC
#fuses HS, NOWDT, LVP
#use delay (clock=20000000)
//Definizione della porta seriale
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7, bits=8)
char car=0;
//Définizion della funzione d' interruzione
#INT_RDA
RDA_isr(){
```

La parte seguente è da inserire nella scheda della fotocamera

```
car = getchar(); //ricezione del byte di dati
output_B(car); //Accensione della LED in corrispondenza al byte ricevuto
}
void main(){
char d=0;
//Inizio del protocollo d'interruzione
enable_interrupts(INT_RDA);
enable_interrupts(GLOBAL);
//Si mette qui il programma principale
}
```

Impostando questi codici sulle le due schede elettroniche (fotocamera, madre) si riesce a trasmettere dati dalla fotocamera verso la scheda madre. Questi dati nel nostro caso sono la posizione dell'oggetto da afferrare.
Nota: Il codice completo si trova nella parte Appendici A6

2. Comunicazione tra il braccio e la scheda madre

2.1. Protocollo di comunicazione I^2C

La comunicazione tra la scheda madre e quella del braccio è realizzata attraverso il collegamento I^2C. In questa parte è descritta la trasmissione dei dati tramite I^2C e un codice nel linguaggio C è proposto per la sua realizzazione.

2.1.1. Principio

Il collegamento I^2C permette la trasmissione di dati tra più microcontrollori PIC con una struttura di tipo “master-slave” (il “master” (scheda madre) invia ordini da seguire allo “slave” (scheda del braccio),la scheda “slave” non ha diritto di inviare ordini al “master”). Una connessione I^2C è fatta attraverso tre fili:

- Il filo SDA (Serial Data Line) con il quale i dati vengono trasferiti.
- Il filo SCL (Serial Clock Line) che trasmette il segnale dell’orologio imposto dal PIC della scheda madre.
- Il filo di terra.

Ogni PIC “slave” ha un indirizzo. Il PIC master essendo unico, non dispone di un indirizzo.

Il protocollo di comunicazione I^2C segue il seguente schema:

- Invio di un bit di inizio (START);
- Invio dell'indirizzo “slave”;
- Invio di un bit di lettura(READ) o scrittura(WRITE) ;
- In attesa o invio un bit di "ricevuta di ritorno"
- Invio o lettura di un byte di dati;
- In attesa o invio di un bit di "ricevuta di ritorno";
- Invio di un bit di stop per completare il protocollo.

La seguente figura descrive il cronografo del protocollo.

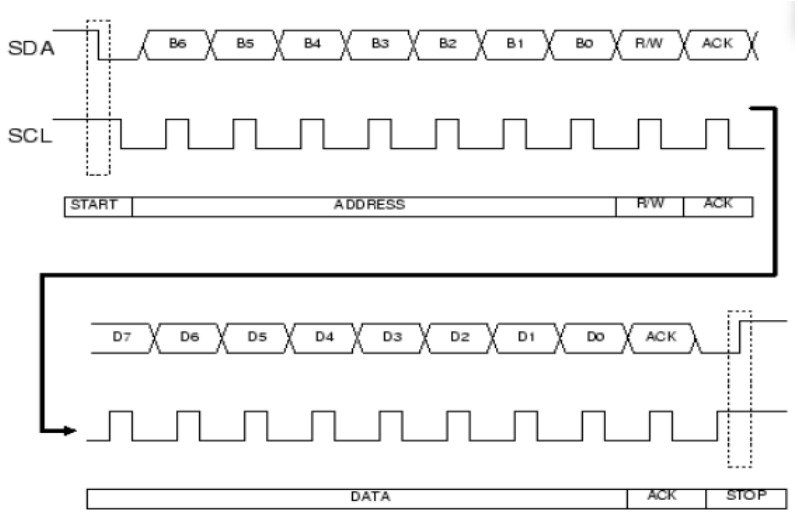


Figura 17 : Cronografo del protocollo I^2C.

Nota: Come si può osservare sulla **Figura 17**, ad ogni fronte di discesa di SDA, il SCL è a 1, è l'inizio del protocollo e ad ogni fronte di salita di SDA, il SCL è a 1, è la fine del protocollo.

Applicando il protocollo nel nostro caso abbiamo la seguente figura:

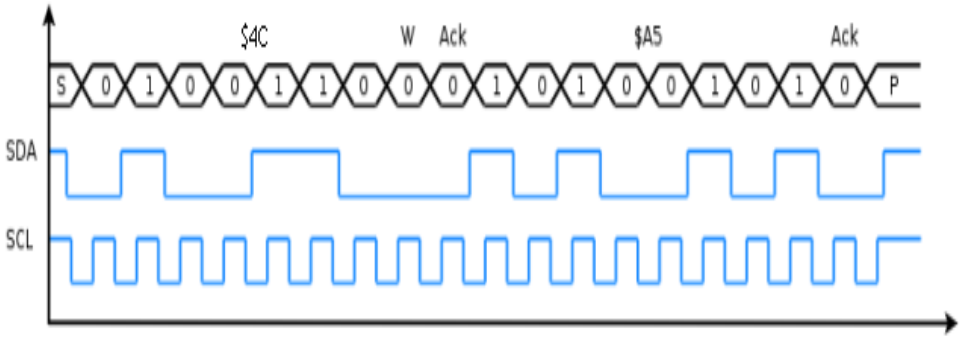


Figura 18 : Esempio di scrittura dei dati.

Interpretazione :

All'inizio della comunicazione,la scheda madre invia una (S) poi invia l'indirizzo della scheda del braccio(0x4C) e annuncia che gli sta inviando i dati (W); a questo punto la scheda madre aspetta la ricevuta di ritorno (Ack). Doppo aver ricevuto la risposta, invia i dati (0xA5) e aspetta la ricevuta di ritorno ,dopo la risposta dalla parte della scheda del braccio, invia (P) per terminare la comunicazione.

Per la lettura dei dati, il protocollo è abbastanza simile solo che l'invio dei dati (0xA5) si effettuerà dalla scheda del braccio verso la scheda madre.

2.1.2. Funzione d'interruzione

La funzione d'interruzione specifica al protocollo I^2C è un programma che permette di interrompe l'esecuzione del programma principale del braccio per dare la precedenza ai dati inviati nel collegamento I^2C. Dopo l'esecuzione di questo sotto programma, l'esecuzione del programma principale continua dove si è fermato per causa dell'interruzione. Questo è il modo più adeguato per trattare informazioni che arrivano in modo casuale nei "pins" del PIC e che richiedono una certa precedenza.

2.1.3. Realizzazione del protocollo I^2C nel linguaggio C

Nel codice seguente, "adresseCarte" e "data" si riferiscono a dei byte (possono essere definiti del tipo intero). Per esempio, prenderemo adresseCarte = 0x10. Si può osservare che la scheda del braccio non può rispondere direttamente alla scheda madre; infatti, è necessario che la scheda madre invia una richiesta alla scheda del braccio prima che quest'ultima sia in grado di rispondere. Il codice seguente mostra la trasmissione di un byte di dati dal "master" allo "slave" e la ricezione di questo byte dallo "slave".

Codice impostato sulla scheda madre.

```
#include <16f877a.h> // inclusione della libreria associata al PIC programmato
#fuses HS, NOWDT, LVP //definizione dei fusibili
#use delay (clock=20000000) // dichiarazione della frequenza dell'orologio esterno
//Definizione del protocollo I2C
#use I2C(master, SCL=PIN_C3, SDA=PIN_C4, stream=I2CM)
void main(){
        i2c_start(); //inizio della comunicazione
        i2c_write(adresseCarte); //invio dell'indirizzo della scheda del braccio
        i2c_write(data); //invio dei dati
        i2c_stop();} //fine della comunicazione
```

Codice impostato sulla scheda del braccio

```
#include <16f737.h> // inclusione della libreria associata al PIC programmato
#fuses HS, NOWDT // definizione dei fusibili
#use delay(clock=20000000) // dichiarazione della frequenza del clock esterno
//Definizione del protocollo I2C
#use i2c(SLAVE, sda=PIN_C4, scl=PIN_C3, address=0x10, FORCE_HW,
stream=I2CS)
int data=0; //variabile di recezione dei dati inviati tramite I2C
void main(){
// Inizio dell'interruzione per il protocollo I2C
enable_interrupts(GLOBAL);
enable_interrupts(INT_SSP);
// Inserire il codice del programma  principale }
#INT_SSP  //funzione interruzione
void i2c_interupt(){
        byte state;
        state = i2c_isr_state();
        if((state== 0 ) || (state == 0x80)){
                i2c_read(); //lettura dell'indirizzo del destinatario }
        if(state >= 0x80){
        i2c_write(1); }//si invia 1 se la scheda master richiede un'informazione
        else if(state > 0){
        data = i2c_read(); }}//lettura dei dati inviati
```

Impostando questi codici sulla scheda madre e quella del braccio, si riesce a trasmettere i dati dalla scheda madre verso la scheda del braccio. Questi dati nel nostro caso sono i segnali per controllare i sei servomotori del braccio.

2.1.4. Algoritmo di controllo dei sei servomotori del braccio

Come già detto precedentemente, per portare la pinza del braccio ad una posizione desiderata dello spazio, è necessario tener sotto controllo tutti i sei servomotori. Quando la fotocamera invia alla scheda madre le coordinate cartesiane (x, y, z) di un punto, essa le trasforma in un segnale che permetterà ai sei servomotori di adottare una posizione angolare e cosi portano il braccio al punto di coordinate (x,y ,z). Il segnale utilizzato nel controllo dei servomotori è del tipo PWM (pluse worth modulation); è una finestra rettangolare periodica con larghezza modulabile.

Un microcontrollore PIC può generare direttamente dei segnali PWM sui “pins” dedicati a questo effetto. Per il controllo del braccio, il PIC deve generare sei segnali PWM; purtroppo il PIC16F737 integrato nella scheda del braccio può fornire solo tre segnali PWM. Questo difetto ci obbliga a creare artificialmente dei segnali PWM con larghezza corrispondente alla posizione angolare che si vuole dare ai servomotori per portare la pinza del braccio al punto desiderato.

Per creare un segnale PWM, basta indicare il tempo per il quale il segnale è a 1 e conoscendo il periodo T della finestra rettangolare, si deduce il tempo per il quale il segnale è nullo.

Figura 19 : Comunicazione tra scheda braccio e scheda madre.

L’algoritmo seguente descrive la generazione dei segnali PWM per il controllo dei servomotori del braccio.

Main(x,y,z) **// Sulla scheda madre**

1) Invio delle coordinate cartesiane (x,y,z) di un punto dello spazio dalla fotocamera verso la scheda madre.

2) Calcolo dei 6 angoli corrispondenti alla posizione desiderata.

3) Invio degli angoli in forma di segnale PWM tramite il collegamento I^2C .

Main(t0,t1,t2,t3,t4,t5) // **Sulla scheda del braccio**

Inserire l'interruzione I^2C

1) t0,t1,t2…t5 sono tempi per i quali ogni segnale è a 1

2) Classifica dei tempi t0,t2…t5 in ordine crescente

Esempio : t0<t1<t2<t3 …

3) Calcolo delle durate relative (differenza tra due valori di tempo consecutivi) .

Esempio : nella classifica precedente, $\Delta 0=t0$, $\Delta 1 = t1-t0$ . . . $\Delta 5 =t5$.

4) Mettere a 1 i sei "pin" della scheda del braccio. Aspettare il tempo $\Delta 0$ poi mettere a zero il "pin" corrispondente al tempo t0

5) Aspettare il tempo $\Delta 1$ poi mettere a zero il "pin" corrispondente al tempo t1

6) Procedere della stessa maniera e alla fine, aspettare il tempo $T-\Delta 5$ per cominciare un nuovo ciclo.

Il diagramma seguente descrive schematicamente l'algoritmo con i tempi classificati come segue: (t0<t1<t2...<t5).

Per semplicità, la rappresentazione è limitata a tre segnali invece di sei.

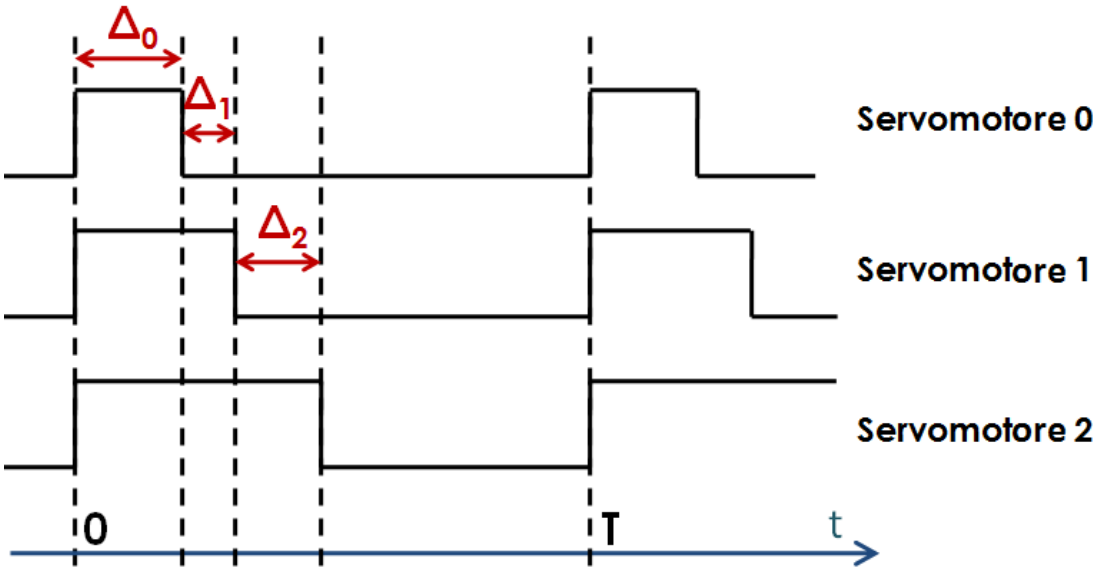


Figura 20 : Segnali PWM ordinati per il controllo dei servomotori.

3. Comunicazione tra fotocamera e base mobile

Il ruolo della base mobile è quello di portare il braccio abbastanza vicino all'oggetto da afferrare. Il principio è il seguente: la fotocamera identifica l'oggetto da afferrare, si centra su quest'ultimo e invia in base alla posizione dell'oggetto un ordine di spostamento della base mobile alla scheda madre (attraverso il collegamento seriale RS232); la scheda madre trasmette a sua volta quel ordine alla scheda di controllo dei motoriduttori (attraverso il collegamento I^2C). La figura seguente mostra la trasmissione.

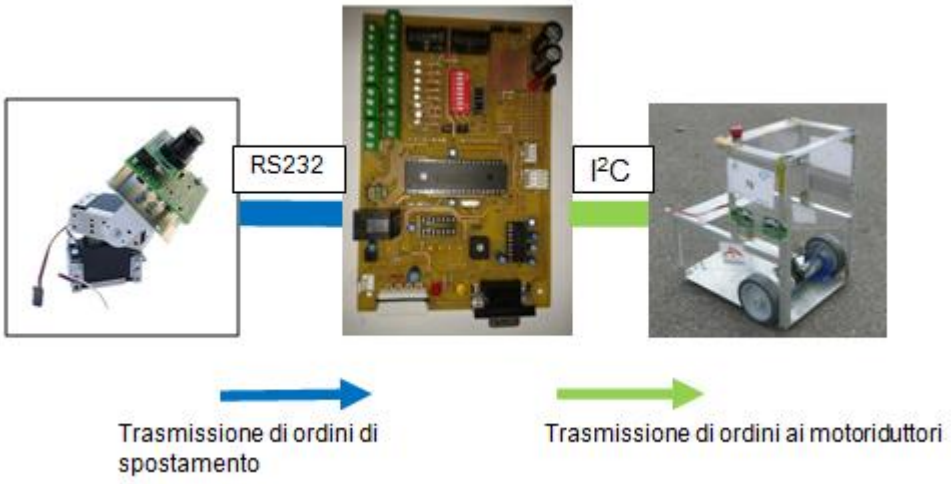


Figura 21 : Principio della comunicazione fotocamera-base rotolante.

3.1. algoritmo di approccio dell'oggetto da afferrare

La fotocamera, fissata sulla base rotolante la controlla indirettamente attraverso la schede madre. Per realizzare il controllo, si usa la posizione angolare orizzontale data dal servomotore rispetto ad un valore di riferimento. se l'angolo è troppo grande (rispettivamente troppo piccolo), la fotocamera invia un ordine scheda madre di girare a sinistra (rispettivamente a destra). Una volta che la base è sufficientemente centrata, deve avanzare o andare indietro in base alla distanza alla quale si desidera fermare la base rispetto all'oggetto; per quello, la dimensione apparente dell'oggetto sullo schermo (corrispondente all'acquisizione fatta dalla fotocamera al numero di pixel appartenenti all'oggetto) viene confrontato con un valore di riferimento corrispondente alla distanza desiderata. Se l'immagine sullo schermo è troppo piccola (rispettivamente grande), la telecamera invia un ordine alla scheda madre di andare avanti (rispettivamente indietro). Quando l'immagine apparente è sufficientemente vicino al valore di riferimento, la fotocamera invia l'ordine alla scheda madre di fermare i moto-riduttori. L'algoritmo completo è un ciclo infinito riassunto come segue: la fotocamera

rileva l'oggetto, ci si centra e poi invia le istruzioni alla scheda madre come spiegato sopra, poi il ciclo ricomincia dopo un periodo di attesa di 0, 12 secondi. La tabella seguente mostra i diversi ordini inviati dalla fotocamera alla scheda madre.

Ordini	**Carattere inviato dalla fotocamera alla scheda master**
A destra	d
A sinistra	g
Avanti	a
Indietro	r
Arresto	s
Bersaglio perduto	p

Il controllo dei moto-riduttori è realizzato dalla scheda madre usando gli ordini inviati dalla fotocamera. Per quello si deve agire sui registri presenti sulla scheda MD25 . Questa scheda presenta 17 registri di cui certi sono in solo lettura (non si può modificare il contenuto) quindi ci occuperemmo solo dei registri modificabili (0,1,15,16). Visto che non abbiamo bisogno di modificare l'accelerazione, il registro 14 non vera modificato. La tabella seguente è un estratto del documento tecnico (data Sheet) della scheda MD25 di controllo dei moto-riduttori.

Register	Name	Read/Write	Description
0	Speed 1	R/W	Motor1 speed (mode 0,1) or speed (mode 2,3)
1	Speed 2/Turn	R/W	Motor2 speed (mode 0,1) or turn (mode 2,3)
2	Enc1a	Read only	Encoder 1 position, 1st byte (highest), capture count when read
3	Enc1b	Read only	Encoder 1 position, 2nd byte
4	Enc1c	Read only	Encoder 1 position, 3rd byte
5	Enc1d	Read only	Encoder 1 position, 4th (lowest byte)
6	Enc2a	Read only	Encoder 2 position, 1st byte (highest), capture count when read
7	Enc2b	Read only	Encoder 2 position, 2nd byte
8	Enc2c	Read only	Encoder 2 position, 3rd byte
9	Enc2d	Read only	Encoder 2 position, 4th byte (lowest byte)
10	Battery Volts	Read only	The supply battery voltage
11	Motor 1 current	Read only	The current through motor 1
12	Motor 2 current	Read only	The current through motor 2
13	Software revision	Read only	Software Revision Number
14	Acceleration rate	R/W	Optional Acceleration register
15	Mode	R/W	Mode of operation
16	Command	R/W	Used for reset of encoder counts and module address changes

Figura 22 : Estratto del documento tecnico della scheda MD25.

3.1.1. Qualche osservazione sulla tabella

- Il registro “Mode”

Nel codice, è necessario innanzitutto scegliere il modo di operazione (registro15) perché deciderà della funzione di controllo della velocità dei motori . Ci sono quattro casi (0,1,2,3) divisi in due parti: se si sceglie il modo 0 o modo 1, il registro “Speed1” controllerà la direzione e la velocità del motore 1 e ” Speed2” controllerà la direzione e la velocità del motore 2. Ma se si sceglie il modo 2 o modo 3, il registro “Speed1” determinerà la direzione e la velocità di entrambi i motori, vale a dire che le due ruote girano alla stessa velocità e il registro “Speed2” diventa il modo

per girare a destra o a sinistra. Questo è il motivo per il quale è necessario selezionare il modo prima di tutto.

Il modo 0 è quello predefinito. Se il modo non è specificato nel codice, la scheda MD25 sceglierà automaticamente questo modo. In questo caso, la cifra 0 rappresenta la velocità massima, di rinculo, 255 la velocità massima in avanti e 128 corrisponde alla fermata. L'unica differenza tra il modo 0 e il modo 1 si trova sui numeri utilizzati; nel modo 1, -128 è la velocità massima di rinculo, 127 è la velocità massima in avanti e 0 corrisponde alla fermata. La differenza tra il modo 2 e il modo 3 è la stessa tra il modo 0 e il modo 1.

- Il registro “Speed1”

Come introdotto in precedenza, se si sceglie il modo 0 o il modo 1, si controllerà la velocità e la direzione del motore 1, se si sceglie il modo 2 o 3, si controllerà la velocità e la direzione dei due motori.

- Il registro “Speed2”

Nel modo 0 o 1, si controlla la velocità e la direzione del motore 2. Il registro consente anche usare il “turn mode” con il modo 2 o 3. In questo modo, ci sono due forme che dipendono dalla direzione.

Quando la base va avanti, si ha:

Speed 1 = speed - turn

Speed 2 = speed + turn

Dove Speed1 è la velocità attuale del motore 1, Speed2 è la velocità attuale del motore 2, speed è la velocità che si è data ai due motori (inizialmente nel registro 0) e turn è la velocità data (nel registro attuale) per ottenere la rotazione.

Quando la base va indietro, si ha

Speed 1 = speed + turn; Speed 2 = speed – turn.

3.1.2. Esempio per la funzione Avanti()

Prima di eseguire il programma principale, è necessario inizializzare la scheda elettronica dei motori, scegliendo il modo di operazione. Qui si sceglie il modo 2 cosi per controllare entrambi i motori e per poter girare. Questo codice descrive l'inizializzazione della scheda MD25.

```
i2c_start(); // inizio del protocollo
i2c_write(I2CM;MD25); //indirizzo della scheda
i2c_write(I2CM; 15); // registro 15 : scelta del modo
i2c_write(I2CM; 2); // modo 2 scelto
i2c_stop(); // fine del protocollo
```

Dopo l'inizializzazione, si può allora inviare degli ordini sotto forma di funzioni di tipo: avanti(),indietro(), giri() ... Questo è l'esempio della funzione avanti()

```
i2c_start();
i2c_write(I2CM;MD25);
i2c_write(I2CM; 0); // registro 0 per il controllo della velocità
i2c_write(I2CM; 200); // si sceglie una velocità  200 .Nel modo 2, si va avanti
i2c_stop();
```

CAPITOLO 3: Strategia per afferrare un oggetto

Questo capitolo descrive le tecniche implementate per controllare la fotocamera, la base mobile e il braccio allo scopo di afferrare un dato oggetto.

1. Inseguimento di un oggetto da afferrare

La fotocamera è in grado di inseguire un oggetto identificando il suo movimento o colore. È questa seconda funzione che sarà dettagliata nel seguito.

1.1. Coordinate in pixel di un oggetto

Un'immagine è in realtà una griglia di punti colorati, chiamati “pixel”. Questa è una tabella in cui ogni punto è identificato dalle sue coordinate (x, y). Così, il punto (0,0) è il punto in alto sinistra dell'immagine, mentre il punto (s_x, s_y) è in basso desta. s_x e s_y sono la larghezza e l'altezza del quadro intero e la risoluzione dell’immagine è s_xxs_y .

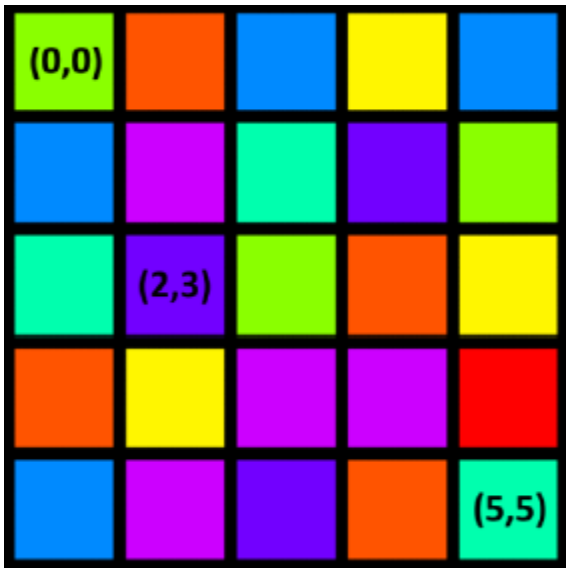
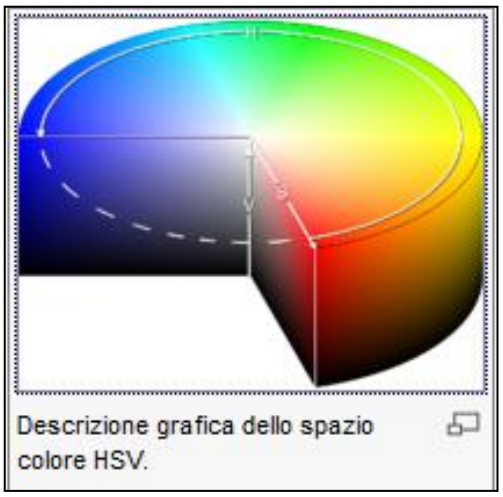


Figura 23 : Rappresentazione di un’immagine 5x5.

1.2. Spazio H S V (Hue-Saturation –Value)

HSB è un modello di colore percettivo perché è molto vicino alla percezione fisiologica del colore dall'occhio umano. In questo sistema i colori sono caratterizzati in tre dimensioni :

- La tinta (H) : corrisponde alla percezione del colore ed è misurata su una scala circolare (cerchio di cromaticità di Newton) per un angolo da 0 ° a 360 °.
- La saturazione (S): misura il grado di purezza di un colore, cioè la quantità di grigio aggiunta al colore . È rappresentata dal raggio di una sezione circolare del cono e varia da 1 (colore puro o saturo) a 0 (corrispondente livello di grigio).
- La luminosità (V): è il grado di schiarimento o di oscuramento del colore. Essa è definita come una scala lineare da 0 (nero) a 1 (bianco) attraverso tutti i livelli di grigio. La figura seguente mostra lo spazio HSV.



Descrizione grafica dello spazio colore HSV.

Figura 24 : Spazio di colore HSV.

1.3. Rivelazione di un oggetto

I passi per la rilevazione dell'oggetto sono:

- selezionare i pixel appartenenti all'oggetto da inseguire: un “pixel” è selezionato se il suo colore è sufficientemente vicino al colore dell'oggetto da inseguire, la rappresentazione dei colori viene effettuata nello spazio HSV descritto sopra.

- calcolare il baricentro dei “pixel” selezionati per trovare le coordinate (x, y) dell'oggetto nel piano dell’immagine come mostrato nella figura seguente .

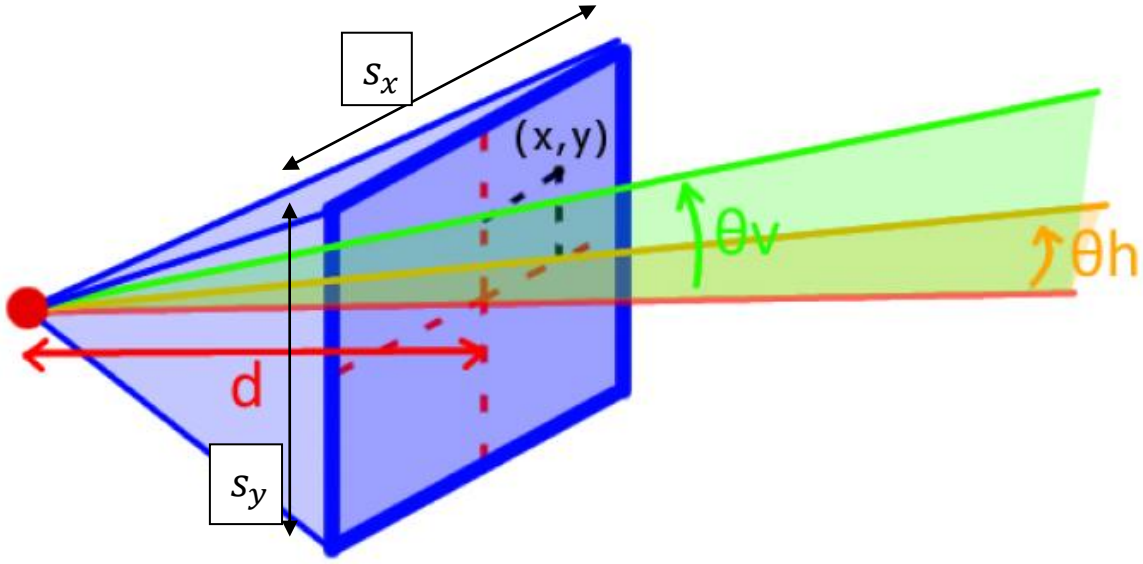


Figura 25 : Piano immagine.

Nota: d è la distanza dalla fotocamera al piano immagine

θ_v è la posizione angolare sulla verticale di uno dei servomotori che controllano la posizione della fotocamera

θ_h è la posizione angolare dell'altro servomotore sull'orizzontale

1.4. Orientamento della fotocamera sull'oggetto

I passi per l'inseguimento dell'oggetto sono:

- calcolare gli angoli θ_h e θ_v in funzione della posizione in pixel (x, y) dell'oggetto
- Inviare i comandi ai due servomotori per posizionare la fotocamera in modo che l'oggetto appaia al centro del campo.

1.4.1. Calcolo di θ_h e θ_v

Sia s_x , s_y i valori definendo la risoluzione in pixel della fotocamera cioè le dimensioni del quadro che delimita il piano immagine. Ci interessiamo a due angoli particolari (orizzontale α e verticale β) della fotocamera come mostrato nella seguente figura. Quei angoli sono forniti dal costruttore della fotocamera oppure calcolati usando una interpolazione.

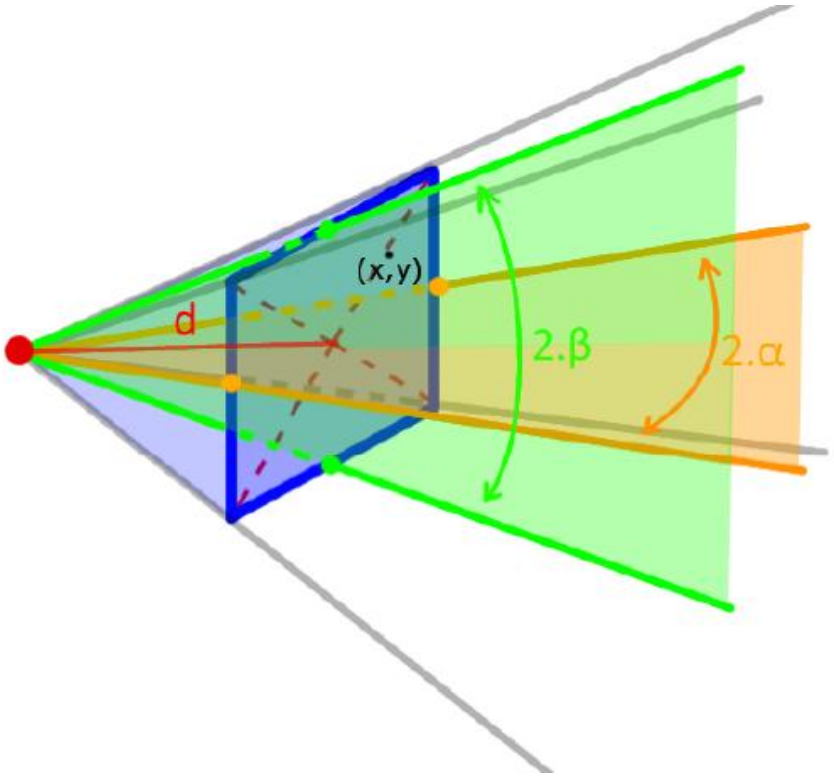


Figura 26 : angoli di campo della fotocamera.

Da questa rappresentazione si ha direttamente:

$$\tan(\beta) = \frac{s_y}{2d} \quad \text{e} \quad \tan(\alpha) = \frac{s_x}{2d} \quad \text{quindi} \quad \tan(\alpha) = \frac{s_x}{s_y}\tan(\beta) \qquad (1)$$

Si nota che basta conoscere uno dei due angoli per dedurre l'altro.

Dalla figura 20, si ha:

$$\tan(\theta_h) = \frac{\frac{s_x}{2} - x}{d} \quad \text{e} \quad \tan(\theta_v) = \frac{\frac{s_y}{2} - y}{d} \qquad (2)$$

da (1) si deduce $d = \frac{s_y}{2\tan(\beta)} = \frac{s_x}{2\tan(\alpha)}$ (3)

Usando (2) e (3) si ha finalmente:

$$\tan(\theta_h) = \frac{2.d_x.\tan(\alpha)}{s_x} \quad \text{con} \quad d_x = \frac{s_x}{2} - x.$$

$$\tan(\theta_v) = \frac{2.d_y.\tan(\beta)}{s_y} \quad \text{con} \quad d_y = \frac{s_y}{2} - y.$$

d_x e d_y sono le coordinate relative dell'oggetto nel piano immagine.

Da queste due ultime formule si nota che posizionando l'oggetto a dei valori conosciuti d'angoli θ_h e θ_v e riportando le coordinate in pixel nel piano immagine, si calcolano gli angoli di campo α e β per interpolazione lineare tracciando la tangente dell'angolo in funzione della posizione in pixel.

1.5. Coordinate dell'oggetto

Per afferrare un oggetto, è necessario avere le coordinate della sua posizione relativa ad un sistema di riferimento legato alla base del braccio. Finora, abbiamo visto che la fotocamera è in grado di rilevare un oggetto nel piano dell'immagine. Si chiede allora di dedurre le coordinate cartesiane dell'oggetto nel riferimento associato la base della fotocamera (che si trova a qualche traslazione della base del braccio)

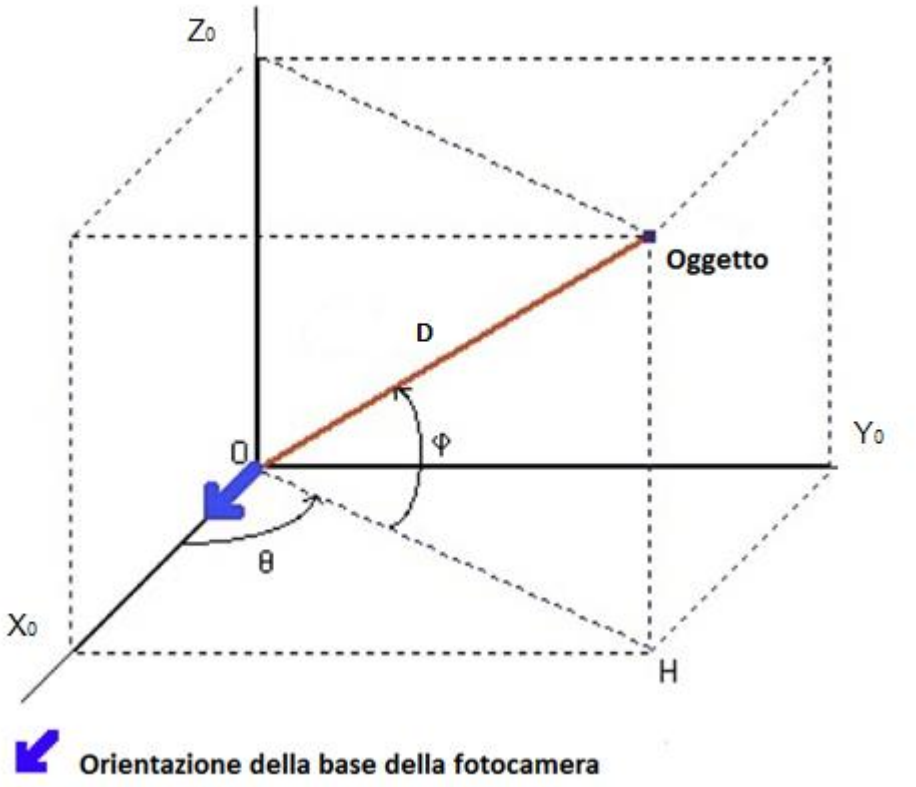


Figura 27 : Coordinate sferiche dell'oggetto.

Prima di tutto, si calcolano le coordinate sferiche dell'oggetto (D, θ, ϕ). Si può notare che con la localizzazione dell'oggetto nel piano immagine, si ha θ = θ_v e ϕ = θ_h. Manca la distanza della fotocamera all'oggetto. Per calcolare D, una calibrazione viene fatta all'avvio della fotocamera e si riporta il numero A_0 di "pixels" dell'oggetto posto a distanza D_0 . Si deduce allora la distanza D ad ogni momento ed in ogni punto usando la relazione empirica seguente:

$D = D_0\sqrt{\frac{A_0}{A}}$ A è il numero istantaneo di "pixels " dell'immagine.

Per trovare le coordinate cartesiane dell'oggetto nel riferimento relativo alla fotocamera, basta usare le formule di passaggio dal sistema coordinate sferiche a quello di coordinate cartesiane. Si ha allora:

$X_0 = Dcos(\theta)\cos(\varphi)$; $Y_0 = Dsin(\theta)\cos(\varphi)$; $Z_0 = Dsin(\varphi)$.

Queste sono le coordinate che la fotocamera invia alla scheda madre per permettere al braccio di afferrare l'oggetto. Le coordinate nel riferimento relativo alla base del braccio si deducono da queste per traslazione visto che la fotocamera e il braccio sono posizionati sulla base mobile a distanza fissa.

2. Controllo del braccio per afferrare l'oggetto

Per controllare il braccio, si deve calcolare i sei angoli corrispondenti ai sei servomotori. Visto la complessità di gestire tutti i gradi di libertà del braccio, abbiamo scelto di fissare certi angoli e anche scomporre il movimento globale in movimenti elementari. Infatti, grazie alla base mobile, il robot si avvicinerà sufficientemente vicino all'oggetto. Il braccio sarà sempre ad una distanza fissa dall'oggetto prima di afferrarlo. Ci basiamo su questa configurazione particolare per risolvere il problema.

2.1. Parametrizzazione del problema

Le dimensioni del braccio sono:

- $h = 8{,}9\ cm$
- $l_1 = 14{,}7\ cm$
- $l_2 = 18{,}7\ cm$
- $l_3 = 10\ cm$
- La pinza può aprirsi fino a $3{,}2\ cm$

I parametri di localizzazione dell'oggetto sono:

- Z : l'altezza della pinza
- R : il raggio vettore dal servomotore 1 al servomotore 3
- a_i : l'angolo corrispondente al servomotore numero i .

Gli angoli fissati sono:

- $a_4 = a_5 = 0°$; questo per avere la pinza sufficientemente aperta e le dita posizionati l'uno in faccia all'altro orizzontalmente. Quando la posizione desiderata è raggiunta, si potrà modulare l'angolo a_5 in base alle dimensioni dell'oggetto.
- Per avere la pinza sempre parallelo al pavimento, si fissa a_3 tale che $a_1 + a_2 + a_3 = 90°$.

La posizione dell'oggetto è data dall'altezza Z_0 misurato dal pavimento. Il raggio vettore R_0 è misurato dal servomotore1.

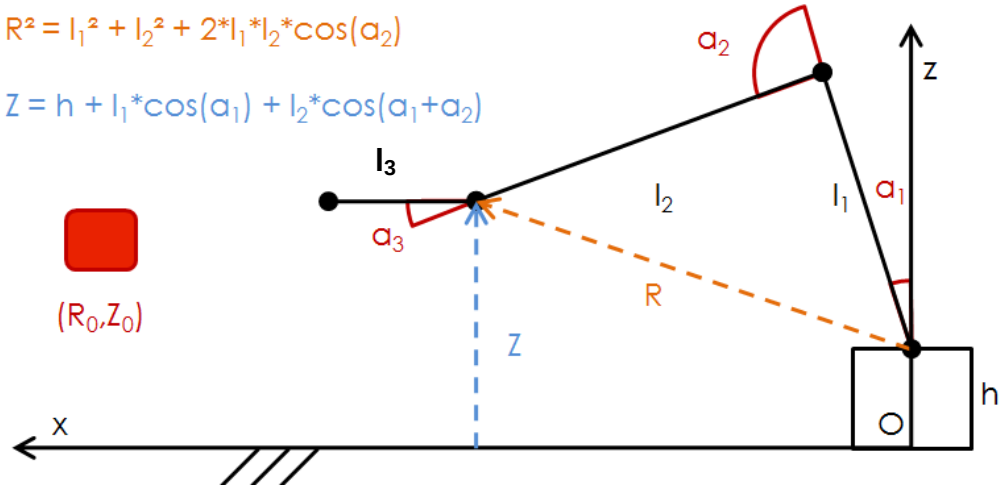


Figura 28 : Parametrizzazione del braccio.

Con questa parametrizzazione, ci rimangono da calcolare gli angoli a_0 , a_1 , a_2 . La posizione iniziale del braccio è definita con i valori seguenti : $a_0 = 0°$, $a_1 = 0°$, $a_2 = 90°$.

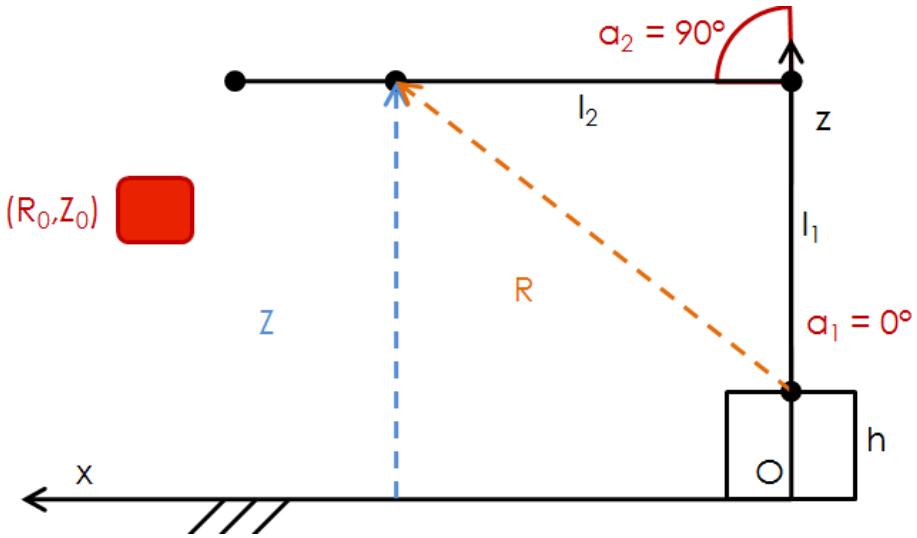


Figura 29 : Posizione iniziale del braccio.

2.2. Calcolo dell'angolo a_0

Per posizionare la pinza nello stesso piano dell'oggetto e tutto il braccio, è necessario agire sul servomotore numero 0. Modificare a_0 corrisponde ad una rotazione della base del braccio attorno all'asse z fino ad avere $Y = Y_0$ dove Y è le seconda coordinata (ordinata) della pinza nel riferimento relativa alla base del braccio. La fotocamera e il braccio sono posizionati alla distanza x_0 seguendo le ascisse , h_0 seguendo l'asse z e hanno la stessa ordinata .

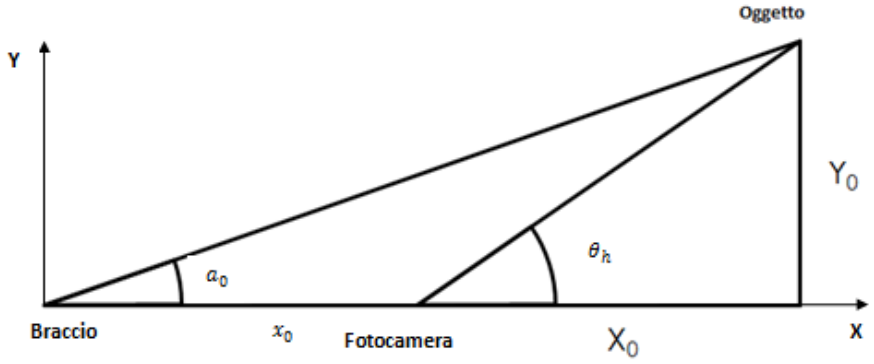


Figura 30 : Primo passo per afferrare : controllo dell' angolo a0.

L'ordinata della pinza è datta dalla seguente formula : Y = $(x_0 + X_0)\tan(a_0)$

2.3. Calcolo del l'angolo a_2

Partendo dalla nuova posizione , si modifica a_2 fino ad avere $R = R_0$ con $R_0 = \sqrt{X_0^2 + (Z_0 - h)^2}$. Cosi, la pinza si trova sul cerchio di raggio R e centrato sul servomotore 1.

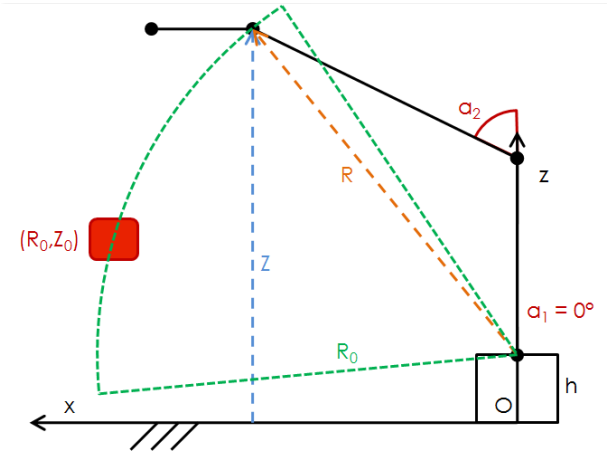


Figura 31 : Secondo passo per afferrare : controllo dell'angolo a2.

2.4. Calcolo dell'angolo a_1

Partendo da a_2, si modifica a_1 spostando la pinza sul cerchio precedente fino ad ottenere $Z = Z_0$.

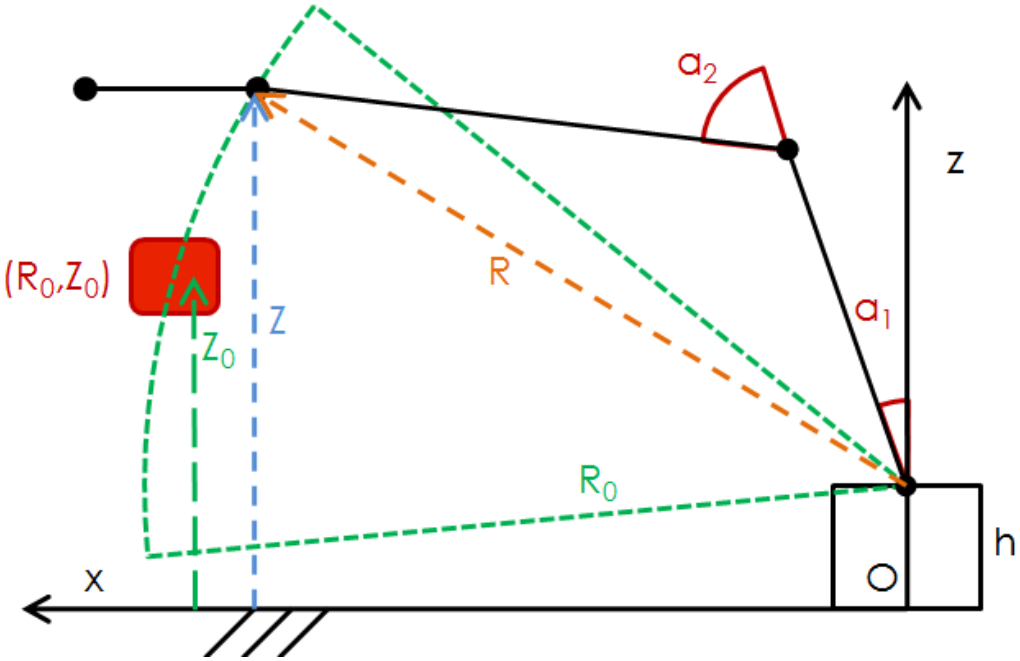
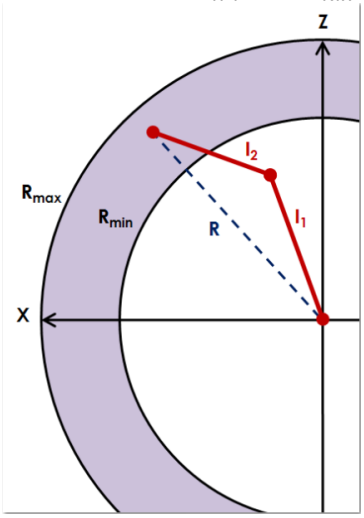


Figura 32 : Terzo passo per afferrare : controllo dell'angolo a1.

Nota: nell'algoritmo non si è tenuto conto della lunghezza l_3 della pinza quindi, se impostiamo il programma cosi, l'oggetto si vera colpito dalla pinza nell'ultima fase di afferramento. Per risolvere il problema, si aggiunge il valore l_3 all'ascisse dell'oggetto par calcolare la distanza alla quale si fermerà la base mobile.

2.5. Zona di afferramento

La posizione della fotocamera sulla base mobile è sfavorevole a certi suoi movimenti per quello la zona di afferramento viene limitata al volume compreso tra le due semi - sfere di raggi R_{min} e R_{max} come mostrato nella figura seguente.



$R_{max} = l_1 + l_2$

R_{min} dipende dagli angoli a_1 e a_2

Figura 33 : Zona di afferramento.

2.6. Riassunto della strategia per afferrare un oggetto

- La fotocamera rileva l'oggetto e calcola le sue coordinate cartesiane (X_0, Y_0, Z_0). Queste coordinate sono poi inviate alla scheda madre tramite il collegamento seriale RS232 assieme all'ordine di spostamento dalla base mobile.
- La scheda madre trasferisce l'ordine di spostamento della base mobile tramite un collegamento I^2C e poi calcola a partire delle coordinate dell'oggetto i sei angoli di controllo per i servomotori del braccio. Dopo questo calcolo, gli ordini sono inviati alla scheda del braccio tramite un altro collegamento I^2C .
- La base mobile si sposta alla posizione desiderata e Il braccio si mette in azione per afferrare l'oggetto.

Conclusione

L'obiettivo di questo progetto di ricerca era la realizzazione di un robot mobile che può identificare un dato oggetto e andare ad afferrarlo. Il robot è costituito di una base mobile sulla quale sono montati una fotocamera autonoma e un braccio robotico che sono stati oggetto di progetti precedenti di studio; mancava di realizzare un sistema di comunicazione tra i diversi elementi per soddisfare i requisiti delle specifiche. La soluzione adottata è quella di utilizzare una scheda elettronica denominata scheda madre che gestisce le trasmissione di ordini inviati dalla scheda dalla fotocamera e elabora il segnale di controllo della base mobile e del braccio.

Il valore aggiunto di questo progetto di ricerca può essere riassunto attraverso le linee seguenti:

- La produzione di un programma funzionale per il controllo del braccio con un movimento rallentato;
- La realizzazione di un programma funzionale per recuperare le coordinate dell'oggetto da afferrare e per convertirli in comandi di controllo adeguate per il braccio;
- La redazione di una relazione tecnica che riassume il funzionamento di ogni parte del robot ed i processi di comunicazione implementati.

Oltre alle prime barriere per controllare i vari strumenti, le principali difficoltà incontrate si trovano nella realizzazione dei diversi programmi. A volte nonostante una compilazione corretta, il codice non produce le azioni previste. Si deve quindi rileggere passo dopo passo tutto il codice per individuare il problema. Questa

operazione può essere lunga sopratutto quando si tratta di un programma utilizzato nei progetti precedenti e non è sempre possibile trovare le persone che hanno partecipato al suo sviluppo dall'inizio.

Il robot completo e la presente relazione rappresentano il risultato del nostro lavoro.

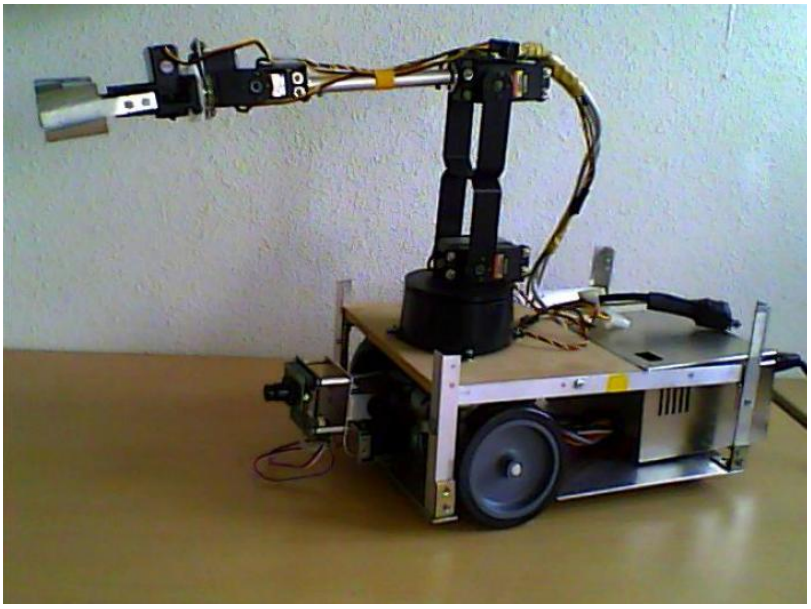


Figura 34 : Fotografia del robot.

❖ proposte per eventuali miglioramenti.

Le soluzioni per soddisfare le specifiche tecniche hanno dimostrato la loro efficacia, però è sempre possibile apportare miglioramenti per ottimizzare quello che è stato fatto. In particolare, potrebbe essere interessante di:

- utilizzare microcontrollori che possono generare contemporaneamente al meno sei segnali PWM . Ad esempio una **scheda Arduino**
- aggiungere altri sensori alla fotocamera per misurare la distanza tra l'oggetto da afferrare e il robot. Questo permetterebbe un guadagno di precisione nell'afferramento.
- Mettere un sensore di forza sulle dita delle pinza del braccio per evitare danni sull'oggetto afferrato. O meglio ancora, sostituire la pinza con un meccanismo di tipo "ventosa" indifferente alla forma dell'oggetto da afferrare. Mettere dei sensori sul braccio per evitare colpi su l'oggetto da afferrare e anche altri oggetti nella zona di afferramento.
- Sostituire la fotocamera con un dispositivo di tipo **Kinect** che tracce molto precisamente lo "scheletro" di una persona. Il braccio robotico potrebbe quindi essere controllato direttamente tracciando il movimento di un braccio umano.

Bibliografia

[1] BRIKAT Zoubaïr ELFAID Youssef GUILLOT Blandine SARCHER Aurélie WILLERVAL Alexandre. Rapport du projet d'étude n°10, 2010. (Ecole Centrale Lyon)

[2] GHANEM Léa DURONIO Romain ZHANG Zhao PIERRE Alexandre BOCQUELET Florent. Rapport du projet d'étude n°105, 2010. (Ecole Centrale Lyon)

[3] GUO Fenfei YE Fanbing BRAUD Jérémy MIETKA Colin QUIRET Samuel. Rapport du projet d'étude n°97, 2011. (Ecole Centrale Lyon)

[4]Honda. Asimo, the world's most advanced humanoid robot. http://asimo.honda.com/gallery.aspx

[5] Lynxmotion. Lynxmotion robot kits. http://www.lynxmotion.com/c-130-al5d.aspx

[6] Microchip. Microchip technology inc. http://www.microchip.com/ , http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en026561 http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010219

[7] Edgewall Software. Cmucam : open source programmable embedded color vision sensors. http://www.cmucam.org/

[8] Porta seriale RS232, http://it.wikipedia.org/wiki/EIA_RS-232

[9] Spazio dei colori TSV http://it.wikipedia.org/wiki/Hue Saturation_Brightness

Appendici

A1. Partecipanti al progetto

A1.1. Organizzazione delle responsabilità

Studenti avendo partecipato al progetto e le loro responsabilità.




Matthieu GRARD	Johanna HAUMANN	Philémon HENRY	Parfait KEUMEJIO
Capo del progetto	**Documentazione**	**Meccanica**	**Bilancio**
Comunicazione	**Linguaggio C**	**Fotocamera**	**Braccio robotico**

Professori avendo partecipato al progetto e le loro responsabilità

Tutori scientifici:

- Alexandre saidi , Dipartimento di Matematica e informatica
- Fabien Mieyeville, Dipartimento di Elettronica, Elettrotecnica, Automatica

Consulente in comunicazione:

- Carole MAYER , Dipartimento di comunicazione con le aziende

Consulente nella gestione del progetto:

- Bertrand HOUX, Dipartimento di meccanica-meccatronica

A2. Programmazione di un microcontrollore PIC

A2.1. Il programma MPLAB

A2.1.1. Installazione

Il programma MPLAB è sviluppato dalla società Microchip per la realizzazione di programmi, compilazione e trasferimento del codice al PIC tramite un cavo (USB,lato computer e RJ11 lato scheda elettronica).

Per le schede elettroniche che abbiamo a disposizione, per programmarle, deve essere installato sul computer:

- Il programma MPLAB.
- Il compilatore CSS (pcdideupd.exe poi pcwhdupd.exe).
- CCS plug-in per MPLAB.
- Il driver per l'utilità di trasmissione del codice

Tutti questi programmi sono scaricabili gratuitamente dal sito:
http://www.microchip.com/

Figura 35 : Utilità di trasmissione del codice dal computer al PIC.

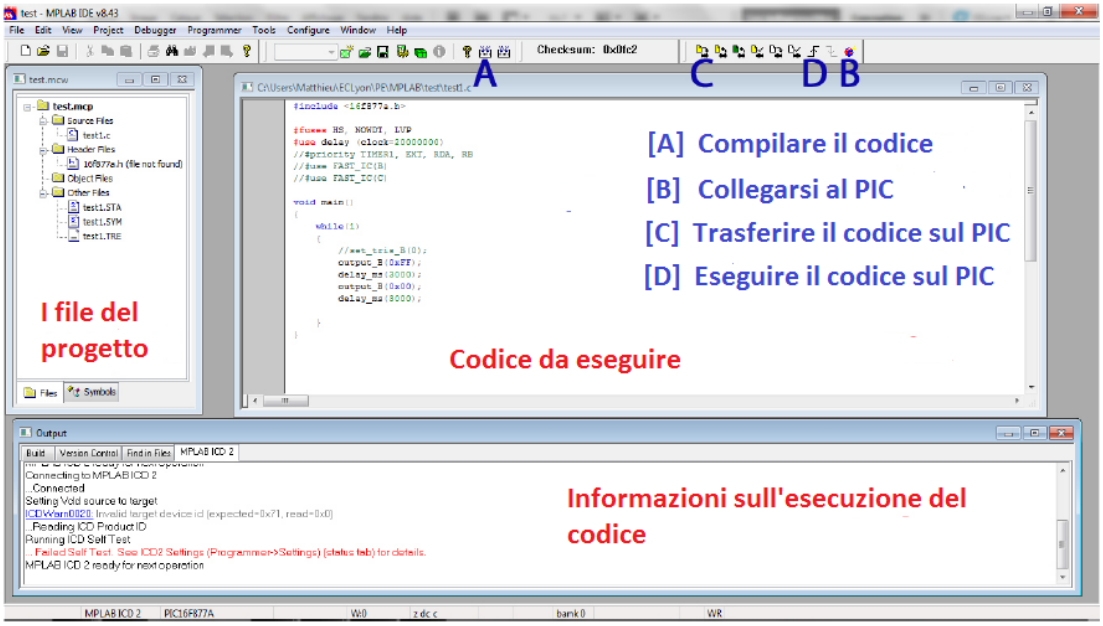


Figura 36 : Interfaccia MPLAB.

A2.2. Le istruzioni in linguaggio C per microcontrollori PIC

I programmi (codici) nella presente relazione sono scritti nel linguaggio C. Di seguito sono elencati le principali istruzioni da sapere per programmare un PIC. Esempio del PIC16F877A.

```
//Dichiarazione delle biblioteche relative al PIC16F877A
#include <16f877a.h>
//Definizione dei fusibili
#fuses HS, NOWDT, LVP
//Frequenza dell'orologio
#use delay (clock=20000000)
//Programma principale
void main()
{
// Ciclo
while(condizione) {azione;}
//Condizione semplice
if(condizione){azione;} else {azione;}
//Definizione degli ingressi e uscite
//0 per uscita, 1 per ingresso
// Con la seguente si mettono tutti i "pins"  della  porta  X  in uscita
set_tris_X(0);
//Invio di un byte, 0xFF (esadecimale) = 11111111 (binario)
output_B(0xFF); //Messa a +5V di tutti i "pins" della porta B
output_bit(pin_B7,1); //Messa a +5V del "pin" B7
//Ricezione
input(pin_B6); //restituisce 0 se il "pin"  B6 è a 0V, 1 se il "pin" è a +5V
input_X(); //restituisce un byte composto dei valori di ogni "pin" dalla porta X
//Aspettare 3000 millisecondi
delay_ms (3000);
//Aspettare 3000 microsecondi
delay_us (3000);
}
```

A2.3. Le funzioni trigonometriche

L'uso delle funzioni trigonometriche (soprattutto coseno, seno e tangente) è possibile includendo la libreria math.h. L'argomento della funzione deve essere espresso in radianti fra $-\frac{\pi}{2}$ e $\frac{\pi}{2}$. Inoltre, se esempio si vuole calcolare il coseno di un angolo x superiore a 90 gradi, si può calcolare - sin(x - $\frac{\pi}{2}$)

A3. Programmazione della scheda della fotocamera

A3.1. Installazione

I programmi necessari alla programmazione della scheda della fotocamera sono:

- Il blocco note o Notepad per scrivere il file nel linguaggio
- Cygwin per compilare il file C in un file hex leggibile dalla la scheda dalla fotocamera
- Utilità flash Philips per inviare il file hex sulla scheda dalla fotocamera

Le ultime due applicazioni possono essere scaricate gratuitamente dal sito http://www.cmucam.org/

A3.2. Programmazione

- Scrivere il codice e salvarlo con estensione .c.
- Copiare questa cartella e la cartella MAKEFILE in una nuova cartella e poi riporla in cygwin/cc3/projects .Accedere a questa cartella usando Cygwin (con il comando cd). Quindi digitare il comando “make” per compilare.

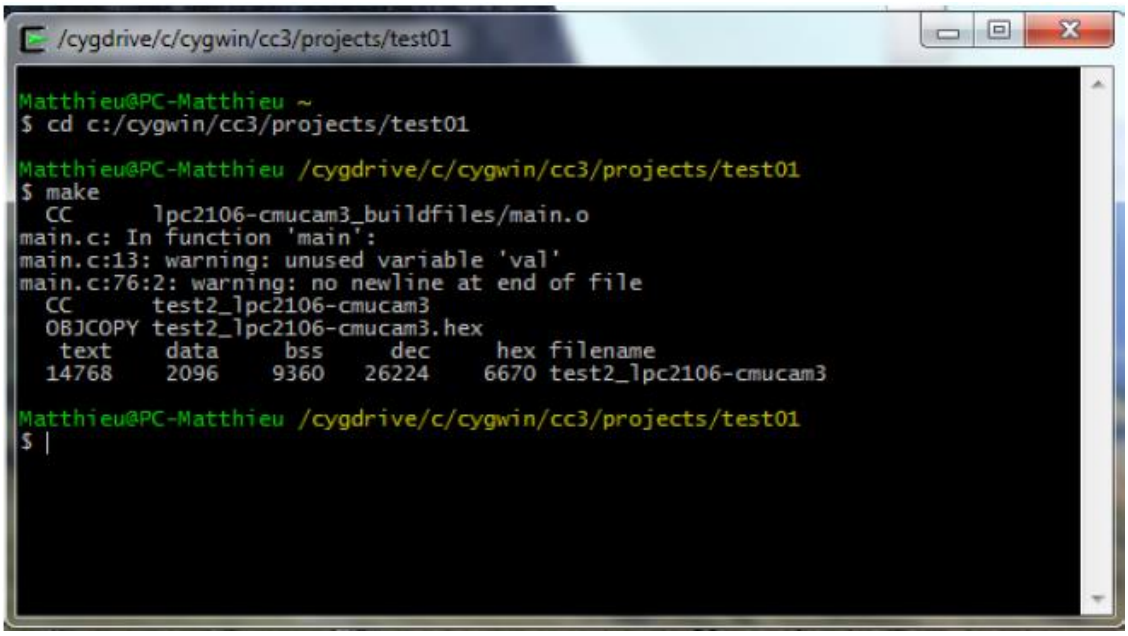


Figura 37 : interfaccia cygwin.

- Aprire l’utilità Philips, selezionare il file hex e cliccare su Upload dopo aver verificato i parametri dell’utilità corrispondono alla porta seriale del computer.

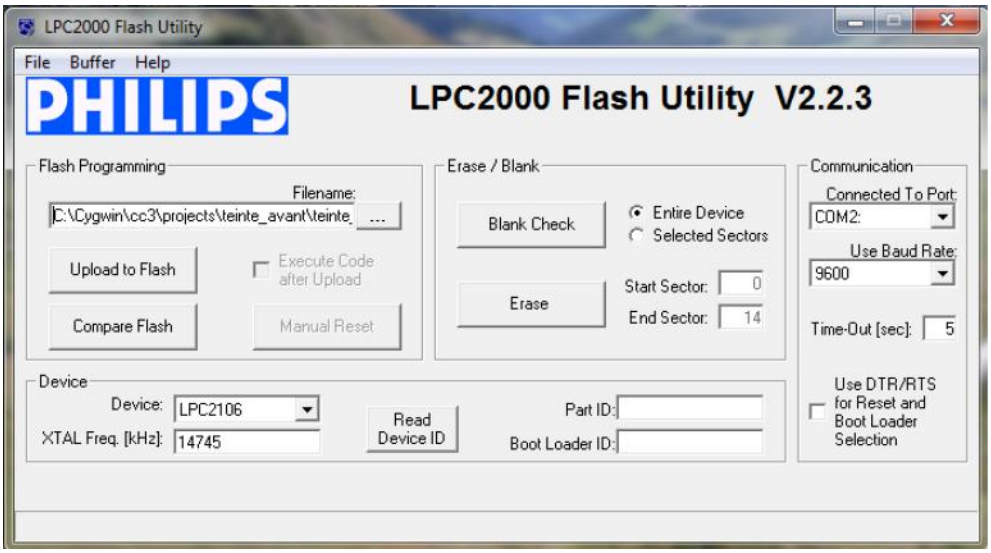


Figura 38 : Interfaccia dell'utilità Philips

A 4. Servomotori

Le figure seguenti mostrano il funzionamento dei vari attuatori del braccio. La quantità t è il tempo di messa a 1 durante un periodo del segnale di controllo e corrisponde all'angolo desiderato. Il servomotore 5 che permette di controllore le dita della pinza è un pò particolare : un angolo di controllo negativo apre la pinza e un angolo di controllo positivo la chiude.

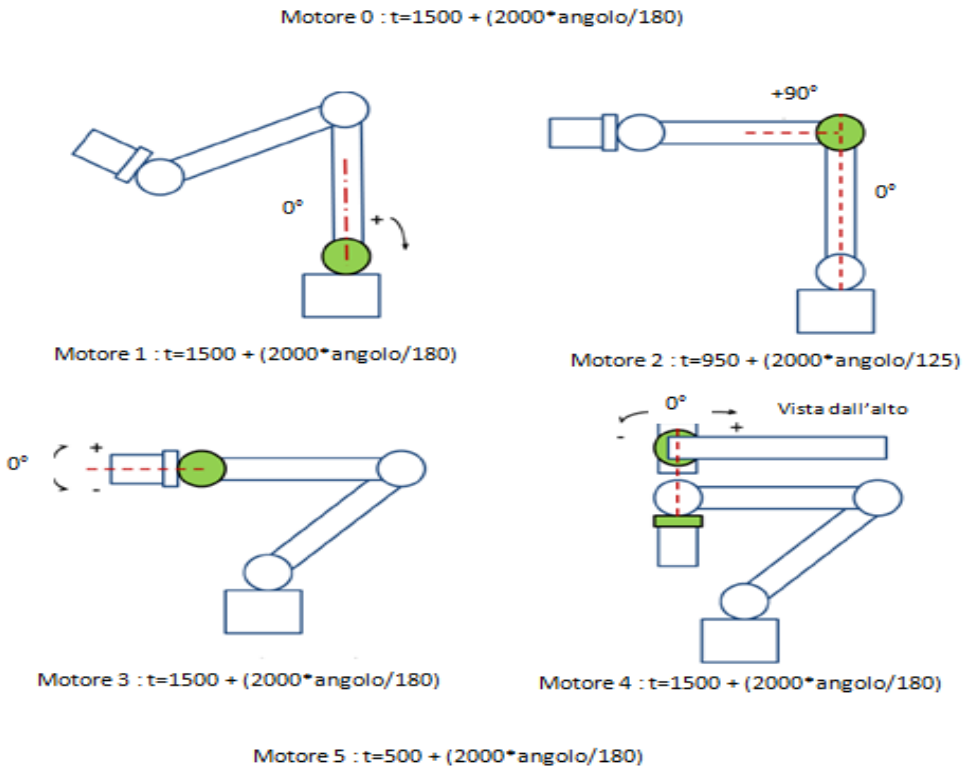


Figura 39 : Rotazione dei servomotori.

A5. Montaggio generale

Di seguito, lo schema del montaggio generale del robot. Per la comunicazione seriale, è necessario che il “pin” 2 della porta seriale sulla fotocamera (che corrisponde alla trasmissione Tx) sia collegato al “pin” 3 della seriale la scheda madre (corrispondente alla ricezione Rx).

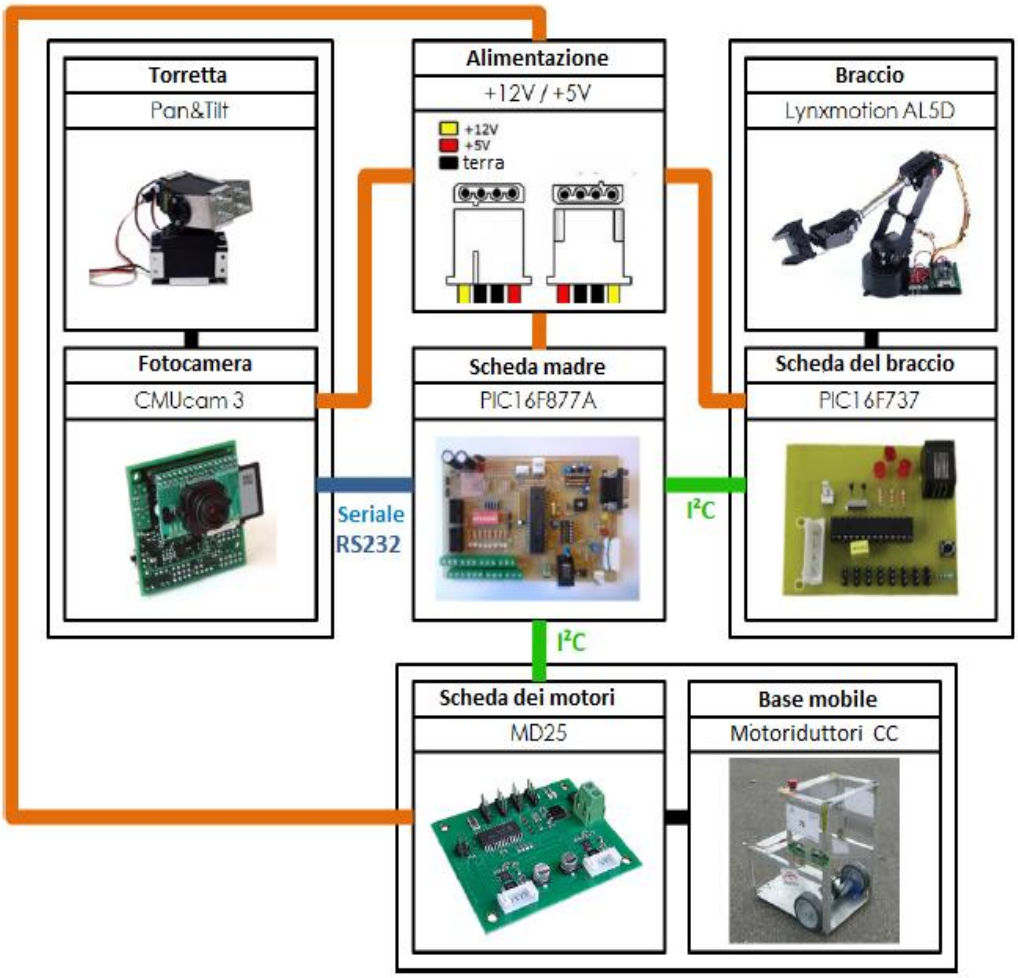


Figura 40 : Montaggio generale del robot.

A6. Codici

A6.1. Scheda della fotocamera

Solo la parte del codice della fotocamera che abbiamo scritto è qui presentato qui sotto, il codice sorgente completo della fotocamera essendo molto lungo. Il codice sorgente iniziale completo può essere consultato riportandosi alla relazione del progetto di studio sulla fotocamera

```
//Dopo che la fotocamera si  è centrata sull'oggetto,si calcolano le coordinate
//cartesiane (X0, Y0, Z0 ) a partire delle coordinate sferiche dell'oggetto
// (distance,angleh,anglev)
distance = sqrt(NBRE_PIXEL_INT);
distance = distance/sqrt(n_pixels_int);
distance = distance*30.0;
xcoord = cos(angleh*pii/180);
xcoord = xcoord*cos(anglev*pii/180);
xcoord = distance*xcoord;
ycoord = sin(angleh*pii/180);
ycoord = ycoord*cos(anglev*pii/180);
ycoord = distance*ycoord;
zcoord = sin(anglev*pii/180);
zcoord= zcoord*distance + 9.35;
xint=(int)xcoord;
yint=(int)ycoord;
zint=(int)zcoord;
//Invio delle coordinate sferiche tramite il collegamento seriale RS232
putchar(xint);
putchar(yint);
putchar(zint);
```

A6.2. Scheda madre per il controllo del braccio

```
#include <16f877a.h>
//Fusibili
#fuses HS, NOWDT, NOPROTECT
//Orologio del PIC
#use delay(clock=20000000)
//Definizione del collegamento I2C
#use I2C(master, scl=PIN_C3, sda=PIN_C4,stream=I2CM)
//Definizione della porta seriale
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7, bits=8)
//Inclusione delle biblioteche
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
//Definizione delle funzioni
void envoiAngles(int,int,int,int,int,int);
void initAngles();
void calculAngle0();
void calculAngle1();
void calculAngle2();
//-------------------------------------------------------
//Definizione delle variabili
//-------------------------------------------------------
signed long buffer[3]; //Per la recezione delle coordinate
signed long angles[6]; //Per lo stoccaggio dei sei angoli
int i=0; //Indica la coordinata ricevuta
signed long Y0=0,Z0=0; //Per lo stoccaggio della posizione desiderata
long R0=0,X0=0;
float l1=14.7,l2=18.7; //Dimensioni del braccio
int nouvellePosition=1; //Indica se una nuova coordinata è stata ricevuta
```

```
//-------------------------------------------------------
//Interruzione  sulla porta seriale
//-------------------------------------------------------
#INT_RDA
RDA_isr()
{
buffer[i] = getchar();
if (i == 2){
i=0;
nouvellePosition = 1;}
else i++;}
//-------------------------------------------------------
//Programma principale
//-------------------------------------------------------
void main(){
//Coordinate iniziali
buffer[0] = 24;
buffer[1] = -5;
buffer[2] = 11;
//Avvio del processo d'interruzione
enable_interrupts(INT_RDA);
enable_interrupts(GLOBAL);
initAngles();
envoiAngles(angles[0],angles[1],angles[2],angles[3],angles[4],angles[5]);
//delay_ms(10000);
while(1){
X0 = buffer[0];
Y0 = buffer[1];
Z0 = buffer[2];
if(nouvellePosition==1){
```

```
//Calcolo degli angoli 0,  2, 1
calculAngle0();
calculAngle2();
calculAngle1();
//Adaptazione per invio degli angoli
if (angles[3]<0) angles[3] = -angles[3];
else angles[3] = -angles[3] + 200;
if (angles[1]>=0) angles[1] = -angles[1]+200;
if (angles[5]<0) angles[5] = angles[5] + 200;
if (angles[0]<0) angles[0] = angles[0] + 200;
envoiAngles(angles[0],angles[1],angles[2],angles[3],angles[4],angles[5]);
delay_ms(20000);
//Afferramento dell'oggetto
envoiAngles(angles[0],angles[1],angles[2],angles[3],angles[4],109);
delay_ms(10000);
//Riposizionamento
initAngles();
envoiAngles(angles[0],angles[1],angles[2],angles[3],angles[4],angles[5]);
delay_ms(10000);
//Afferramento completato
nouvellePosition = 0;}}}
//Inizializzazione degli angloli
void initAngles(){
angles[0] = 0;
angles[1] = 0;
angles[2] = 90;
angles[3] = 90 - angles[2] - angles[1];
angles[4] = 0;
angles[5] = 0;}
```

```
// Calcolo angolo 2
void calculAngle2(){
int k=0;
float m = 0;
float m1=0,m2=0;
float R0reel=0,Rreel=0;
long R0=0,R=0;
m = l1*l2;
m1 = l1*l1;
m2 = l2*l2;
//Cacolo di R0
//R0reel = Z0-h;
R0reel = Z0;
R0reel = R0reel*R0reel;
R0reel = R0reel + X0*X0;
R0reel = sqrt(R0reel);
R0 = (long) R0reel;
//Calcolo di R iniziale
Rreel = m1 + m2 + 2*m;
Rreel = sqrt(Rreel);
R = (long) Rreel;
for(k=0;k<18;k++){
if (R==R0) break;
angles[2] = angles[2]-5;
//Calcolo di R
Rreel = m*cos(angles[2]*pi/180);
Rreel = m1 + m2 + 2*Rreel;
Rreel = sqrt(Rreel);
R = (long) Rreel;}
angles[3] = 90 - angles[2] - angles[1];
k=0;
}
//Fine di calculAngle2()
```

```
//---------------------------------------------------------
// Calolo angolo 1
//---------------------------------------------------------
void calculAngle1(){
int k=0;
float a=0,b=0,Zreel=0;
signed long Z=0;
float h=7.0;
//Calcolo di Z iniziale
a = l1*cos(angles[1]*(pi/180))+h;
b = -l2*cos((angles[1]+angles[2])*(pi/180));
Zreel = a + b;
Z = (long) Zreel;
for(k=0;k<90;k++){
if (Z==Z0) break;
//Incrementazione dell'angolo 1
if (Z>Z0) angles[1] = angles[1] + 1;
else angles[1] = angles[1] - 1;
a = l1*cos(angles[1]*(pi/180)) + h;
if ((angles[1]+angles[2])>90){
b = -l2*sin((angles[1]+angles[2]-90)*(pi/180));
Zreel = a + b;
Z = (long) Zreel ;
}
els{
b = l2*cos((angles[1]+angles[2])*(pi/180));
Zreel = a + b;
Z = (long) Zreel ;
}
}//Fine del ciclo  for
//Calcolo dell'angolo 3
angles[3] = 90 - angles[2] - angles[1];
}
//Fine di  calculAngle1()
```

```
//-------------------------------------------------------
// Calcolo dell'angolo  0
//-------------------------------------------------------
void calculAngle0()
{
int k=0;
signed long Y=0;
long d=5;
for(k=0;k<30;k++)
{
if (Y==Y0) break;
if (Y0<0) angles[0] = angles[0]-2;
else angles[0] = angles[0]+2;
//Calcolo di Y
Y = (X0+d)*tan(angles[0]*(pi/180));
}
if (angles[0]==-10) output_B(0xFF);
}
//Fine di  calculAngle0()
//-------------------------------------------------------
//Invio degli angoli alla scheda del braccio
//-------------------------------------------------------
void envoiAngles(int a0, int a1, int a2, int a3, int a4, int a5)
{
i2c_start(); //inizio della trasmissione
i2c_write(0x10); //indirizzo della scheda del braccio
i2c_write(a0); //invio dei dati
i2c_write(a1);
i2c_write(a2);
i2c_write(a3);
i2c_write(a4);
i2c_write(a5);
i2c_stop(); //fine della comunicazione
}
```

A6.3. Scheda del braccio

```
#include <16f737.h>
#DEVICE ICD=TRUE
#fuses HS, NOWDT
#use delay(clock=20000000)
//Definizione del protocollo i2c
#use i2c(SLAVE, sda=PIN_C4, scl=PIN_C3, address=0x10, FORCE_HW,
stream=I2CS)
#include <stdio.h>
#include <stdlib.h>
//---------------------------------------------------------------------------
//Definizione delle funzioni
//---------------------------------------------------------------------------
void i2c(); //Protocollo di comunicazione
void classement(); //Mette i tempi in ordine crescente
void output0(int); //Mette a 0 la porta corrispondente al servomotore di cui il
numero è indicato come argomento.
void initialisation(); //inizializza il vettore tab1 contenente  i tempi e i numeri dei
servomotori  corrispondenti
void actualisation(); //Aggiorna il vettore  tab1
void commande(); //Invia i segnale ai servomotori
void comparaison(); //incrementa o decrementa  i tempi attivi
//---------------------------------------------------------------------------
//Definizione di variabili
//---------------------------------------------------------------------------
signed long tab1[3][6]; //tab1[0] : tempi attivi ;tab1[1] : numeri dei servomotori
//tab1[2] : tempi desiderati
signed long tab2[7]; //tempi relativi di messa a  1
int send[0x08]; //variabile di invio e recezione di dati dal collegamento I2C
signed long data[6],data1[6]; //stoccaggio dei dati ricevuti
signed long buffer=0;
int k=0; //Indica il numero dell'angolo da ricevere
```

```
//-------------------------------------------------------------
//Funzione d' interruzione  per  ricuperare gli angoli desiderati
//-------------------------------------------------------------
#INT_SSP
void i2c(){
byte state;
state = i2c_isr_state();
if((state== 0 ) || (state == 0x80)){
i2c_read();
}
if(state >= 0x80){
i2c_write(send[state - 0x80]);
}
else if(state > 0){
data[k] = i2c_read(); //Si ricupera l'angolo k
if (data[k]>=110)data[k]=data[k]-200;
if (k==5)
{
k=0;
actualisation();
}
else k++;
} }
//-------------------------------------------------------
//Programma principale
//-------------------------------------------------------
void main(){
//Posizione iniziale desiderata
data[0]=0;
data[1]=-25;
data[2]=60;
data[3]=90;
data[4]=0;
data[5]=90;
```

```
//Posizione  attuale iniziale
data1[0]=0;
data1[1]=0;
data1[2]=90;
data1[3]=0;
data1[4]=0;
data1[5]=0;

initialisation();
classement();
//Avvio del processo d'interruzione
enable_interrupts(GLOBAL);
enable_interrupts(INT_SSP);
while(1)
{
comparaison();
classement();
commande();
}
}
//-----------------------------------------------------
//Funzione che mette i tempi in ordine crescente
//-----------------------------------------------------
void classement()
{
int i,j;
long memo[3];
//Ordinamento dei tempi nel vettore  tab1
for(i=0;i<6;i++)
{
for(j=i+1;j<6;j++)
{
```

```
if(tab1[0][i]>tab1[0][j]){
memo[0] = tab1[0][i];
memo[1] = tab1[1][i];
memo[2] = tab1[2][i];
tab1[0][i] = tab1[0][j];
tab1[1][i] = tab1[1][j];
tab1[2][i] = tab1[2][j];
tab1[0][j] = memo[0];
tab1[1][j] = memo[1];
tab1[2][j] = memo[2];
}}}
//Calcolo dei tempi relativi nel vettore  tab2
tab2[0] = tab1[0][0];
for (i=1; i<6; i++){
tab2[i] = tab1[0][i]-tab1[0][i-1];
}
tab2[6] = 20000-tab1[0][5];
}
//-----------------------------------------------------
//Mettere a 0 il serovomotore di cui il numero è  indicato come  argomento
// della funzione
//-----------------------------------------------------
void output0(int m){ //m è il numero del servomotore
if(m==0)
output_bit(PIN_B0,0);
else if(m==1)
output_bit(PIN_B1,0);
else if(m==2)
output_bit(PIN_B2,0);
else if(m==3)
output_bit(PIN_B3,0);
else if(m==4)
output_bit(PIN_B4,0);
else
output_bit(PIN_B5,0);
}
```

```
//---------------------------------------------------------
//Funzione che aggiorna  le colonne di  tab1
//---------------------------------------------------------
void actualisation(){
int i;
for (i=0;i<6;i++){
if (tab1[1][i]==0) tab1[2][i]=1500+data[0]*11;
else if (tab1[1][i]==1) tab1[2][i]=1500+data[1]*11;
else if (tab1[1][i]==2) tab1[2][i]=950+data[2]*9;
else if (tab1[1][i]==3) tab1[2][i]=1500+data[3]*10;
else if (tab1[1][i]==4) tab1[2][i]=1500+data[4]*11;
else tab1[2][i]=500+data[5]*11;
}}
//---------------------------------------------------------
//Funzione d'inizializzazione di tab1
//---------------------------------------------------------
void initialisation(){
int i;
//Tempi corrispondenti alla posizione desiderata
tab1[2][0]=1500+data[0]*11;
tab1[2][1]=1500+data[1]*11;
tab1[2][2]=950+data[2]*9;
tab1[2][3]=1500+data[3]*10;
tab1[2][4]=1500+data[4]*11;
tab1[2][5]=500+data[5]*11;
//Numeri dei servomotori
for(i=0; i<6; i++) tab1[1][i]=i;
//Tempi corrispondenti alla posizione iniziale
tab1[0][0]=1500+data1[1]*11;
tab1[0][1]=1500+data1[1]*11;
tab1[0][2]=950+data1[2]*9;
tab1[0][3]=1500+data1[3]*10;
tab1[0][4]=1500+data1[4]*11;
tab1[0][5]=500+data1[5]*11;
}
```

```
//-------------------------------------------------------
// La funzione seguente invia i sei segnali ai servomotori (a partire da tempi attuali)
//-------------------------------------------------------
void commande()
{
output_B(0b00111111); //messa a 1 di tutti servomotori
// Aspettare il primo tempo e mettere a 0 la parta del servomotore corrispondente e
cosi via
delay_us(tab2[0]);
output0(tab1[1][0]);
delay_us(tab2[1]);
output0(tab1[1][1]);
delay_us(tab2[2]);
output0(tab1[1][2]);
delay_us(tab2[3]);
output0(tab1[1][3]);
delay_us(tab2[4]);
output0(tab1[1][4]);
delay_us(tab2[5]);
output0(tab1[1][5]);
delay_us(tab2[6]);
}
//-------------------------------------------------------
//La funzione seguente confronta i tempi attuali e quelli desiderati poi incrementa o
decrementa i tempi attuali in conseguenza
//-------------------------------------------------------
void comparaison()
{
int i;
for(i=0 ; i<6 ; i++ )
{
if((tab1[2][i]-tab1[0][i])>0) tab1[0][i] = tab1[0][i] + 1;
else if((tab1[0][i]-tab1[2][i])>0) tab1[0][i] = tab1[0][i] - 1;
}}
```

A6.4. Scheda madre per il controllo della base mobile

Questo codice è stato sviluppato da un progetto precedente e l'abbiamo provato per verificare il funzionamento del materiale.

```
#include <16f877a.h>
//Fusibili
#fuses HS, NOWDT, LVP
//Orologio del PIC
#use delay(clock=20000000)
//Collegamento RS232
#use rs232(baud=115200, xmit=PIN_C6, rcv=PIN_C7)
//Collegamento I2C
#use I2C(master, SCL=PIN_C3, SDA=PIN_C4)
#define   MD25   0xB0   // indirizzo della scheda MD25
void setup_md25(void);
void drive(unsigned char speed);
void drive_turn(unsigned char turn);
void avancer();
void reculer();
void turnright();
void turnleft();
void main()   {
char d=0;
setup_md25();
 //Inizializzazione della porta b
set_tris_b(0x00);
output_B(0x00);
  while(1)    {
d = getc(); // ricezione dell'ordine dalla fotocamera
switch (d) // analisi dell'ordine
{case 'a' : //avanti
   avancer();
break;
case 'r' : //indietro
   reculer();
break;
```

```
case 'd' : //girare a destra
   turnright();
break;
case 'g' : // girare a sinistra
   turnleft();
break;
case 's' : // fermata
   drive(128);
   drive_turn(128);
break;
case 'p' : // bersaglio perso
   drive(128);
   drive_turn(128);
break;}       }      }
void setup_md25(void) //inizializzazione della comunicazione con  MD25
{
  i2c_start();
   i2c_write(MD25);
   i2c_write(15);   // registro per la scelta del modo
   i2c_write(2);  // modo 2 : velicità : 0 .. 128 .. 255
   i2c_stop();
   //Messa a zero all'inizio
    i2c_start();
   i2c_write(MD25);
   i2c_write(0);
   i2c_write(128);
   i2c_stop();
      i2c_start();
   i2c_write(MD25);
   i2c_write(1);
   i2c_write(128);
   i2c_stop();
}
```

```
void drive(unsigned char speed){
  i2c_start();
   i2c_write(MD25);
   i2c_write(0);    // registro 0
   i2c_write(speed);
   i2c_stop();
}
void drive_turn(unsigned char turn){
   i2c_start();
   i2c_write(MD25);
   i2c_write(1);   // registro 1
   i2c_write(turn);
   i2c_stop();
}
void avancer(){
   drive(128+50);
   drive_turn(128);
}
void reculer(){
   drive(128-50);
   drive_turn(128);
}
void turnright(){
   drive(128+40);
   drive_turn(128+10); // il valore di drive_ turn non è alto perché il robot deve girare
abbastanza lentemente per mantenersi centrato sull'oggetto.
}
void turnleft(){
   drive(128+40);
   drive_turn(128-10);
}
```

Riassunto del progetto

Nella nostra società odierna dove il numero dei vecchi è sempre crescente, la robotica si presenta come una soluzione d'aiuto a quelle persone e in generale come una soluzione per tutte le persone a mobilità ridotta. Questo progetto mira a sviluppare un robot in grado di identificare un dato oggetto, di avvicinarsi ed afferrarlo. È un progetto di ricerca in continuazione di precedenti progetti di ricerca su un braccio robotico controllato in posizione, una fotocamera che può identificare un dato oggetto di un unico colore e stimare la distanza che glielo separa e una base mobile guidata dalla fotocamera.

Attraverso questo progetto, le tre parti del robot (base mobile, fotocamera e il braccio robotico) sono stati assemblati e riprogrammati per soddisfare le specifiche. La descrizione precisa del ogni elemento del robot, gli algoritmi ed i dettagli della loro attuazione (compreso il codice) sono presentati nella presente relazione. Alla fine di questo progetto di ricerca, la comunicazione tra il braccio e la fotocamera è stata fatta: il braccio è capace di afferrare un oggetto lentamente a partire da informazioni che riceve dalla scheda madre. È stato verificato il funzionamento delle comunicazione tra scheda madre e la base mobile ma al momento della scrittura di questa relazione,non è stata realizzata la comunicazione tra fotocamera e base mobile. In fine, sarebbe interessante estendere l'afferramento per qualsiasi oggetto e aggiungere altri sensori per facilitare l'approccio del robot all'oggetto e anche evitare gli ostacoli nel suo ambiente di lavoro .

Abstract

In the context of an aging society, robots are to be a support in our daily life. Guarding this idea, this project elaborates a robot which can identify a preset object, approach and collect it.

The project is based on earlier works regarding an arm moving according to given positions, a camera which can identify a colored object, estimate its distance and command a wheeled support to keep a certain distance to this object.

Throughout this project, the three components, wheeled support, camera and arm are combined and where it is necessary, programmed anew to achieve the wished performance. Functional studies of the elements and the algorithms as well the implementation details (e.g. source codes) are presented in this report and its appendix. Regarding the results of the project, the communication between the camera and the arm do function as predicted but the wheeled support could not be fully

included into the communication within the time. It is therefore recommended to continue the started development and improve, for example, the flexibility of the robot regarding the object´s properties. Adding another sensor to the camera for estimating a more precise position of the object can also be an objective for further projects. It is also recommended to add sonsors to the arm to avoid obstacles in its working environement.