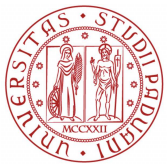


Università degli Studi di Padova

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Triennale in
Ingegneria Dell'Informazione

Braccio robotico a 3 assi per la scrittura su tastiera: progettazione e scrittura del codice

Relatore:
Prof. Matto Meneghini

Laureanda:
Giulia Pizzato
1218012

Anno Accademico 2021/2022
22 Settembre 2022

Ringraziamenti

Ringrazio la mia famiglia, che mi è stata vicino dal primo momento. Ringrazio il mio ragazzo, che mi ha spronato nei momenti di sconforto. Ringrazio i miei amici, senza i quali non sarei arrivata fin qui. E infine ringrazio me stessa, che sono riuscita a credere in me e non mollare fino alla fine.

Abstract

Questa tesi ha lo scopo di mostrare come si è ideato, progettato e scritto software per un braccio robotico a 3 assi. L'obbiettivo è permettere al braccio di riprodurre un testo scritto ricevuto in input. Dopo aver calcolato la mappatura delle coordinate di ogni lettera, verrà nella tesi descritta la gestione del movimento degli assi del robot. In seguito prima si descriverà il robot utilizzato con le sue caratteristiche hardware, e poi si parlerà del problema del movimento degli assi e della sua modellizzazione. Verrà quindi esposto l'effettivo codice, spiegato nelle sue parti più importanti, per concludere con le possibilità di miglioramento e sviluppo futuri.

Indice

Capitolo 1

Introduzione

I robot oggigiorno fanno parte integrante della nostra realtà. Grazie ad essi siamo riusciti a raggiungere risultati incredibili: robot come il "The da Vinci Surgical Robot" permettono operazioni chirurgiche di precisione [1], altri sono al centro di opere artistiche come nella famosa "Can't help myself" degli artisti Sun Yuan e Peng Yu [2], per non parlare dell'intero settore dell'automazione industriale.

Uno dei robot più noti, e quello che accomuna gli esempi sopra portati, è il braccio robotico.

Un braccio robotico è una tipologia di braccio meccanico programmabile e riprogrammabile. Salvo robot antropomorfi per scopi speciali, i bracci robotici hanno un numero di assi generalmente compreso tra i tre e i sei e possono rappresentare sia il sistema completo che essere parte di un sistema più ampio e complesso. Questo genere di braccio ricorda molto quello umano e spesso le sue parti prendono nomi come braccio superiore, gomito, avambraccio, polso e mano [3].

Il primo braccio meccanico sofisticato è stato ideato da Leonardo Da Vinci nel 1495, era dotato di 4 gradi di libertà e in grado di afferrare oggetti solamente muovendo tutti gli arti assieme. Era dotato di un controllore programmabile "a bordo" che provvedeva potenza e controllo sul braccio [4]. Da allora la tecnologia ha portato a passi sempre più ampi di innovazione, raggiungendo risultati come l'introduzione del primo robot industriale prodotto da Unimate nel 1961, robot che si è poi evoluto nel braccio PUMA, o come il primo braccio bionico indossato da Campbell Aird nel 1993 [4, 5].

Basati sul modello del braccio umano, questi robot sono dotati di grande flessibilità e mobilità e sono in grado di svolgere attività umane senza sosta e con grandissima precisione. Ma questi vantaggi vengono ad un costo: una grande

difficoltà analitica. Proprozionalmente al numero di giunti aumentano assieme la difficoltà nel controllo e la possibilità d'azione, un trade-off intrinseco alla tecnologia e all'innovazione.

In questa tesi si è deciso di sfruttare le capacità del braccio robotico ADA031-V4.0 dell'azienda Adeept [6] e scrivere software che renda il robot in grado di premere correttamente i tasti su una tastiera in base ai caratteri dati in input. Presenterò quindi in questo documento l'analisi matematico-strutturale di un braccio meccanico antropomorfo a 3 assi, tipologia in cui rientra il robot utilizzato, per poi esporre lo sviluppo del progetto.

Nello specifico la tesi si svilupperà come segue. Si inizierà dando le specifiche di progetto nel Capitolo 2, esponendo obiettivi, vincoli e assunti della tesi. Nel Capitolo 3, si andrà a presentare approfonditamente l'hardware usato e le sue limitazioni. I capitoli Capitolo 4 e Capitolo 5 verranno invece dedicati alla discussione della teoria dietro al funzionamento del codice e alla spiegazione approfondita dello stesso. La verifica del funzionamento e le conseguenti osservazioni possono essere trovate nel Capitolo 6. Infine i capitoli Capitolo 7 e Capitolo 8 verranno dedicati alle proposte di miglioramento per sviluppi futuri del progetto ed alla conclusione del documento. È possibile trovare il codice completo in Appendice A.

Capitolo 2

Specifiche di Progetto

In questa sezione verranno esposti gli specifici obiettivi, gli assunti e i vincoli del progetto.

2.1 Obiettivi

Questo progetto è stato intrapreso con lo scopo di sviluppare software in grado di permettere al braccio robotico di scrivere, attraverso una tastiera, una o più frasi date in input. Nello specifico il progetto deve essere in grado di:

- Ricevere in input da tastiera una stringa, interpretarla come una serie di caratteri separati e identificare la posizione di ciascun carattere sulla tastiera.
- Calcolare correttamente gli angoli che ciascun motore servo deve assumere affinché la posizione finale del robot prema il tasto.
- Rimanere in ascolto, accettare e processare in ordine FIFO tutti gli input dati da tastiera, anche se sono scritti mentre gli input precedenti stanno ancora venendo processati.
- Muovere i servo al fine di raggiungere la posizione desiderata senza che il robot si scontri in alcun modo con la tastiera o il piano su cui è poggiato.
- Muovere i servo ad un'andatura sufficientemente fluida da evitare il danneggiamento dell'hardware.
- Avvisare l'utente se qualsiasi dei caratteri ad input non è presente sulla tastiera o non è raggiungibile.

- Assumere una posizione neutra quando non sono presenti input da tastiera.
- Stampare sullo schermo OLED, connesso alla scheda elettronica, la lettera mentre sta venendo scritta.

2.2 Assunti

Gli assunti di progetto sono quelle caratteristiche che devono necessariamente essere presenti per il corretto funzionamento del codice.

Si assume che il robot:

- Lavori su una tastiera planare.
- Sia fissato sulla stessa superficie piana della tastiera, in una determinata posizione mostrata in figura 2.1.
- Lavori sulla tastiera di un computer del modello "MacBook Air (13 pollici, 2017)", mostrata in figura 2.1.
- Sia costantemente connesso, con l'apposito cavo, ad un computer su cui è caricato il software. Questo è necessario per ricevere l'input da tastiera.
- Lavori con una penna di lunghezza 55 mm posizionata saldamente sull'estremità ed a 90° rispetto il terzo link del robot.

2.3 Vincoli

I vincoli di progetto sono tutti quei dettagli o quelle caratteristiche intrinseche che limitano la qualità del progetto.

Il fattore più importante è sicuramente l'hardware a disposizione. I bracci meccanici hanno di loro natura delle limitazioni, come l'impossibilità di compiere più azioni contemporaneamente, che aggiunte alle limitazioni meccaniche dello specifico hardware hanno portato a dei chiari vincoli.

Questo è un progetto ideato per essere svolto in circa 75 ore (3 CFU) e quindi non confrontabile con le caratteristiche e raffinatezze che un lavoro di anni sicuramente presenterebbe.

A causa di questi fattori ci sono delle chiare limitazioni.

Il braccio:

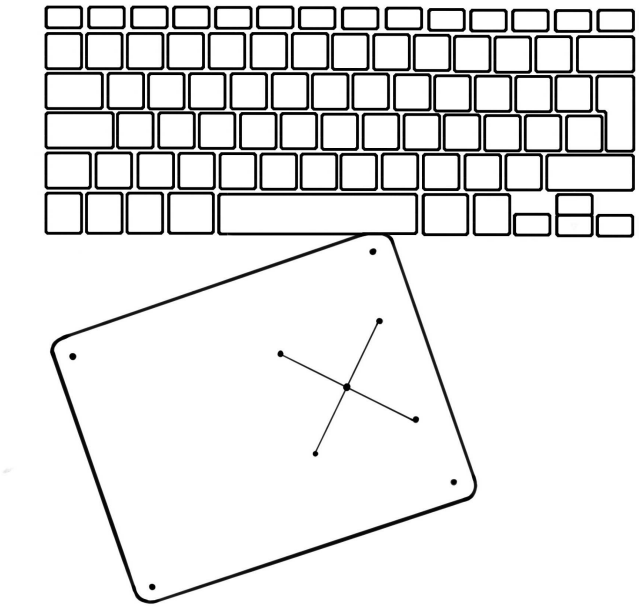


Figura 2.1: Posizione del robot sulla tastiera

- Non è in grado di raggiungere tutti i tasti della tastiera per motivazioni meccaniche (braccio di lunghezza insufficiente).
- Non può scrivere caratteri che necessitano di una combinazione di tasti. Avendo un solo end effector è di natura impossibilitato a pigiare più tasti contemporaneamente.
- Non può premere caratteri di controllo (come “esc”, “invio” o “canc”) in quanto sono tasti non leggibili dal codice come stringhe in input.
- Non può avere contemporaneamente massima velocità e fluidità nei movimenti. Si è dovuto scegliere un trade-off soddisfacente.
- Non può lavorare per lunghi periodi per il riscaldamento ed affaticamento dei motori.

Fatte queste precisazioni si andrà a presentare nel prossimo capitolo l’hardware utilizzato.

Capitolo 3

Hardware

Il braccio robotico utilizzato è il modello ADA031-V4.0 dell'azienda Adeept, commercialmente chiamato *"Adeept 5-DOF Robotic Arm Kit Compatible with Arduino IDE"*. Come visibile in figura 3.1, si tratta di una struttura a tre assi e tre giunti. Ogni giunto è dotato di un servomotore AD002 che, connesso alla "Adeept Arm Drive Board", permette il movimento e il controllo delle parti rigide. I motori servo di questo braccio hanno un range di movimento di 180°, nelle direzioni mostrate dalle frecce rosse, sempre in figura 3.1.

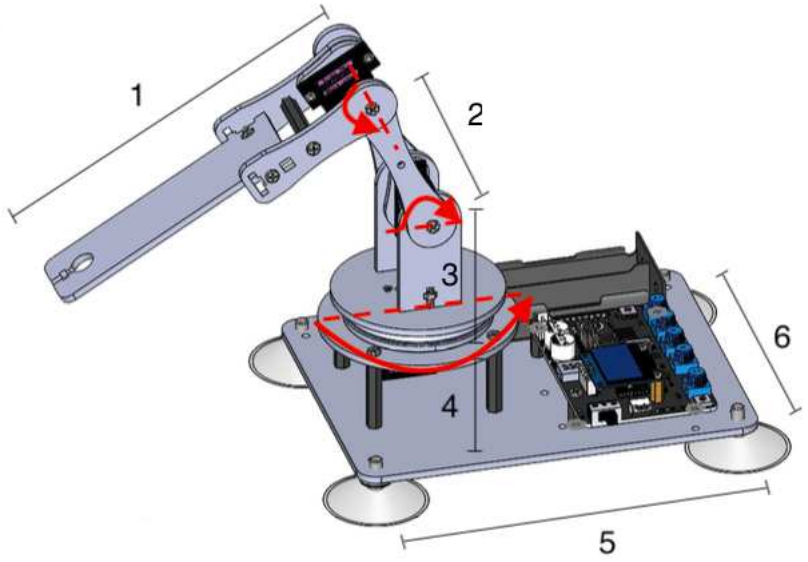


Figura 3.1: Dimensioni Robot Adeept

Tabella 3.1: Misurazioni fatte con strumentazione non specializzata, intervallo d'errore stimato sul mm

Pezzo	Dimensioni ± 1 mm	Numero in figura 3.1
Asse3	147	1
Asse2	65	2
Asse1	50	3
Rialzo dalla base	35	4
Base	282×115	5×6

La struttura è realizzata in plastica nera ed è posizionata su una pedana circolare, dotata di un cuscinetto e rialzata rispetto la base del robot. Sull'estremità finale del braccio è stato aggiunto un end-effector, creato appositamente per il progetto, che permette una maggiore precisione. Si tratta di un puntatore dotato di molla sulla punta, che permette una maggior pressione senza che si rischi lo sforzo dei motori 3.2.

Le articolazioni non sono dotate di sensori, eccetto i controllori della retroazione dei servomotori che hanno l'obiettivo di fermare la rotazione una volta raggiunto l'angolo desiderato. Sono detti di giunti attuati non sensorizzati.

Figura 3.2: Puntatore

3.1 Servomotori

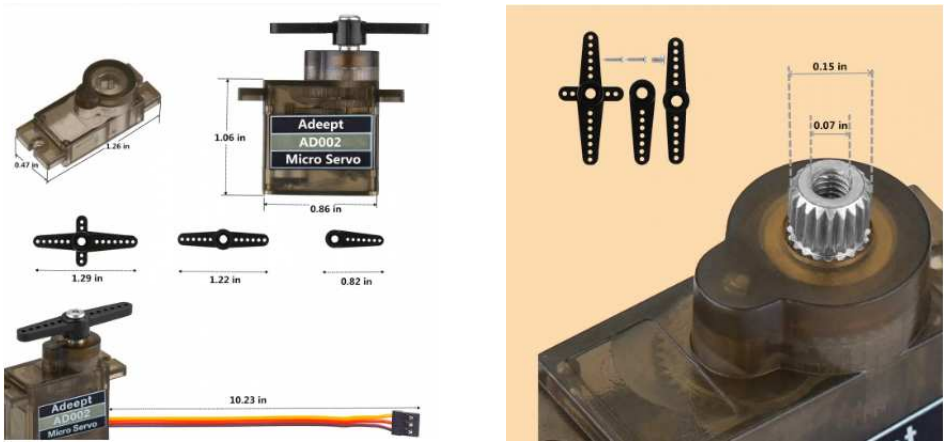
Il termine servomotore si riferisce ad un insieme di elementi che controlla il funzionamento dei componenti meccanici nel servosistema. Si tratta di dispositivi a corrente continua, dotati di un motore, un riduttore e una scheda elettrica adibita al controllo a catena chiusa. Si connettono alla drive board attraverso tre cavi

segnati con codice colore, uno per l'alimentazione, uno per la massa e uno per il segnale.

I servomotori sono di due tipologie, quelli standard e quelli a rotazione continua. La prima tipologia, che è anche quella con cui si è lavorato in questo progetto, permette una discreta precisione sull'angolo e quindi sulla posizione. La seconda invece permette una discreta precisione sulla velocità e quindi sull'accelerazione di rotazione. Sono controllati da impulsi inviati dal microcontrollore, come descritto in sezione 3.1.2. Il sistema servomotore include custodia, circuito stampato, motore, ingranaggi e rilevamento della posizione [9].

3.1.1 Dati

I motori servo presenti sul robot sono del modello Adeept Micro Servo AD002, di cui verranno ora esposti i parametri. Questi motori sono totalmente compatibili con i più comuni motori servo MG90S [11].



(a) (b)

Figura 3.3: Dimensioni servo AD002

Tabella 3.2: Dati

Dato	Valore	Unità di misura
Coppia di stallo	2	kg·cm
Angolo massimo	π	rad
Tensione nominale	4.8	V
Range di temperatura	0 - 55	°C
Velocità nominale	$\frac{0.11}{60}$ (4.8V)	$\frac{sec}{degress}$

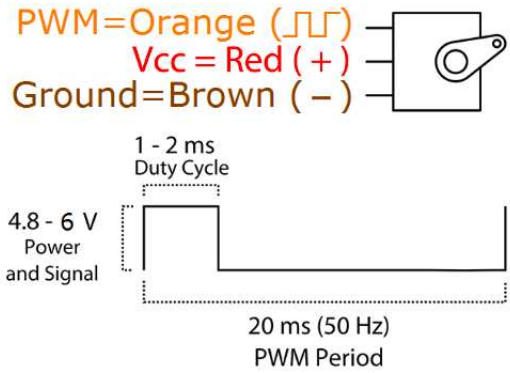


Figura 3.4: PWM dati

Tabella 3.3: filo

Colore filo	Funzione
Arancione	PWM (Segnale)
Rosso	Vcc (Polo positivo)
Marrone	Massa (Polo negativo)

Tabella 3.4: PWM

dato	valore	unità di misura
Tensione massima	4.8 - 6	V
Duty cycle	0.5-2.5	ms
Periodo PWM	20	ms

Altre caratteristiche sono gli ingranaggi del motore che sono in metallo e la compatibilità dei motori, che è con Arduino e Raspberry Pi.

3.1.2 Principio di funzionamento

Il servomeccanismo è un sistema di controllo automatico che consente alle quantità controllate dall'otuput di seguire cambiamenti arbitrari nel target di input in modo continuo nel tempo. Deve quindi essere presente una struttura che permette il feedback.

Il servomeccanismo è fondamentalmente guidato da un sistema di Pulse Width Modulation (PWM).

La PWM è una strategia utilizzata per il controllo del segnale in uscita al circuito quando si sta utilizzando un'output digitale. Difatti, al contrario di un

caso analogico, un segnale digitale può avere solo due valori: alto o basso. Se si necessità di un valore intermedio sembrerebbe impossibile ottenerlo. Si usa quindi la tecnica del PWM, che consiste nel inviare un'onda quadra e, facendo cambiare la durata del duty cycle (ovvero il rapporto tra la durata del segnale alto e la durata totale del segnale), cambiare il valore medio del segnale. Si possono ottenere così output digitali di valore "intermedio" e si possono quindi controllare sistemi che richiedono un range di valori in output, come i servomotori. In base a questo valore il servomeccanismo si sposta fino all'angolo corrispondente. Si possono vedere alcune corrispondenze in figura 3.5.

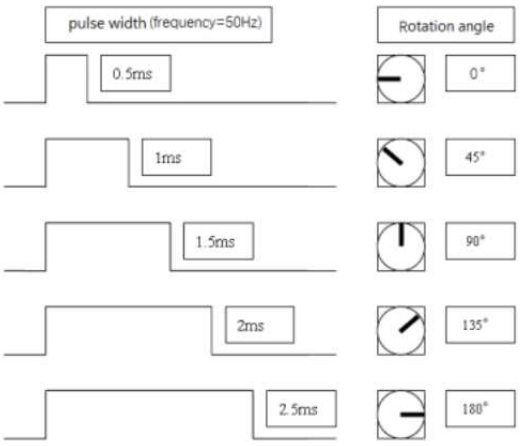


Figura 3.5: Esempi segnale PWM

Il segnale PWM, inviato da Adeept Arm Drive Board, viene quindi elaborato da un circuito integrato sul circuito stampato, al fine di calcolare il senso di rotazione del motore di azionamento, che viene quindi trasmesso attraverso un riduttore al braccio oscillante. Allo stesso tempo, il rilevatore di posizione restituisce un segnale di posizione per determinare se la posizione impostata è stata raggiunta o meno.

Il servomotore utilizza la retroazione negativa, ovvero è in grado di inviare impulsi corrispondenti alla posizione in cui si trova. Se l'impulso che torna al sistema non coincide con quello dato in input allora il servosistema mette autonomamente il motore in azione per tornare alla posizione corretta. In questo modo è possibile controllare con precisione la rotazione del motore [9].

Purtroppo i dati forniti riguardo i valori della PWM erano incoerenti, si è così deciso di andare a studiare la situazione in maniera più diretta. Con un oscilloscopio si è osservato il segnale inviato dal sistema ai motori, corrispondente

agli angoli 0 e 180. Del segnale si è guardato il duty cycle, la durata di un periodo della PWM e la tensione a cui veniva passato il segnale.

Figura 3.6: Segnale inviato ai servo per assumere un angolo di 0°

Come si può vedere in figura 3.6 la durata del periodo della PWM corrisponde a quella data, 20 ms. La tensione operativa è di 4.4 V circa. Il duty cycle è di 2.72%, ovvero il segnale alto dura 0.544 ms.

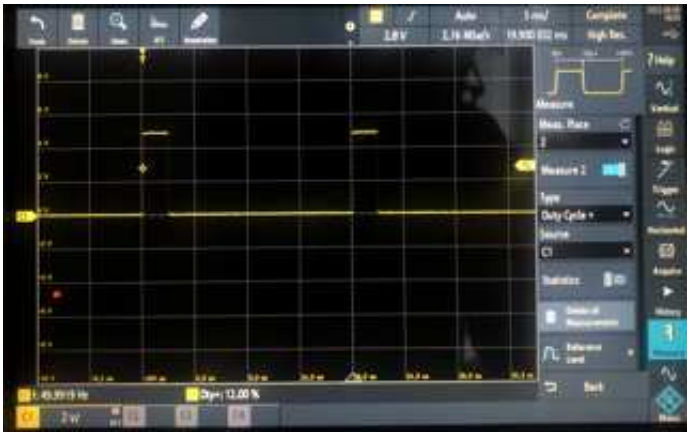


Figura 3.7: Segnale inviato ai servo per assumere un angolo di 180°

Al massimo angolo possibile, come visibile in figura 3.7, il duty cycle è 12%, quindi il segnale alto ha una durata di 2,4 ms circa.

3.2 Scheda Adeept

Il componente principale del braccio robotico, come scritto dall'azienda stessa, è l'Adeept Arm Drive Board. Citando direttamente il manuale (tradotto in italiano), sappiamo che: *"Simile alla scheda di sviluppo Arduino UNO, è una*

piattaforma di prototipazione elettronica open source di facile utilizzo, inclusa la parte hardware e la parte software (Arduino IDE). La scheda di sviluppo Adeept Arm Drive Board è composta principalmente da un microcontrollore (Micro Controller Unit, MCU), un'interfaccia di input/output universale, ecc. Può essere interpretata come una scheda madre per microcomputer." [9]

3.8.

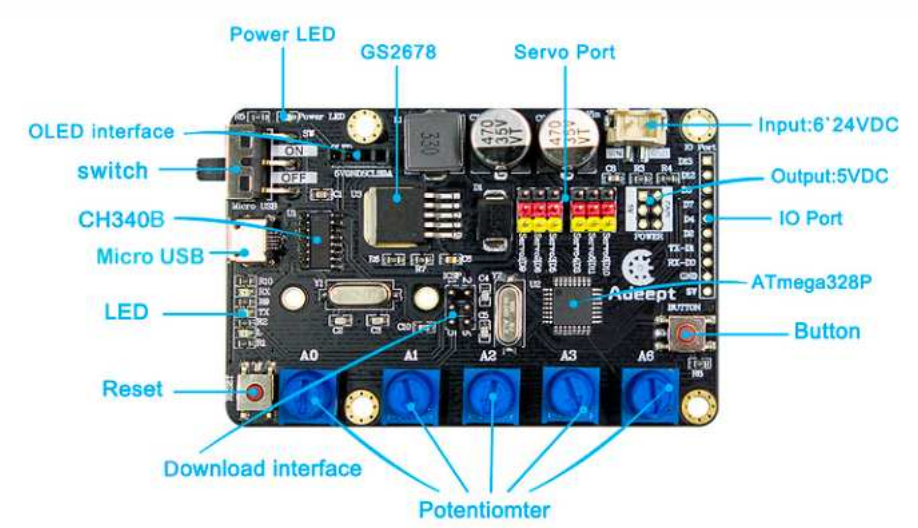


Figura 3.8: Adeept Arm Drive Board

Verranno ora descritte tutte le parti elencate sulla figura 3.8.

LED:

Led controllabile dal microprocessore per fornire informazioni supplementari.

Micro USB:

Questa Micro USB viene utilizzata per collegare la scheda al computer, permettendo la comunicazione seriale, il caricamento del programma e il monitoraggio seriale tra la scheda di sviluppo Adeept SmartHub e il computer. Può essere utilizzata anche come alimentazione a corrente continua a 5 V.

CH340B:

La famiglia di chip CH340 funge lo scopo di convertitori tra una porta USB ad una porta seriale (UART) o ad una porta stampante. Nella modalità UART, utilizzata nel progetto, CH340 provvede segnali MODEM standard, usati per estendere porte seriali dal computer o passare direttamente da un normale dispositivo seriale ad un bus USB [12].

Switch (interruttore):

Quando si utilizza V_{in} (6-24 V) come alimentatore esterno, questo interruttore è l'ON e OFF della scheda di sviluppo Adeept SmartHub.

OLED Interface:
Interfaccia pin dello schermo OLED.

Power Led:
Il LED di alimentazione viene utilizzato per indicare lo stato di alimentazione del sistema. Se è acceso, indica che il sistema è acceso e pronto per funzionare; altrimenti che il sistema è spento.

GS2678 :
Il GS2678 è un convertitore buck (step-down) DC/DC a PWM a frequenza fissa a 350 KHz, in grado di pilotare un carico di 5 A con alta efficienza, bassa ondulazione ed eccellente line e load regulation. Richiedendo un numero minimo di componenti esterni, il regolatore è semplice da usare e include la compensazione della frequenza interna e un oscillatore a frequenza fissa [13].

Servo:
Interfaccia pin dei Servo.

Input: 6-24 VDC:
È l'interfaccia pin per l'alimentazione esterna. L'alimentazione esterna per la scheda di sviluppo Adeept SmartHub richiesta è 6-24 V.

Output:5 VDC:
Output a 5 V a corrente continua.

IO Port:
Interfaccia pin Input Output.

ATmega328P:
Microcontrollore ad 8 bit [14].

Button:
Pulsante per vari utilizzi.

Potentiometer:
Sono presenti cinque potenziometri: A0, A1, A2, A3 e A6. Utilizzando l'apposito codice Arduino, fornito dall'azienda, è possibile utilizzare questi potenziometri per controllare il movimento del braccio robotico. Ogni potenziometro controlla un servomotore.

Download interface:
Interfaccia di programmazione del microcontrollore.

Reset:
Pulsante per il riavvio della scheda di sviluppo di Adeept SmartHub.

Capitolo 4

Basi Teoriche

Al fine di comprendere il funzionamento del software è necessario presentare una parte di teoria di robotica e alcune tecniche di analisi della dinamica di un robot. Verrà poi spiegato come sono state ricavate le formule utilizzate nel codice.

4.1 Definizione e caratteristiche di un braccio robotico

Un braccio meccanico, anche detto manipolatore industriale, è una struttura meccanica formata da corpi (supposti rigidi), chiamati *link* o membri, e da articolazioni, dette *joints* o giunti, che assieme vanno a formare una catena cinematica. Un'estremità della serie è connessa ad una base d'appoggio, che può essere mobile o fissata, mentre l'altra è libera e a volte dotata di un sistema per l'interazione con il mondo, detta end-effector [8].

Un elemento caratterizzante della struttura di un robot sono i suoi giunti. Essi possono essere di varie tipologie, ma sono due quelle più note: prismatiche e rotazionali. Prismatiche significa che sono in grado di muovere i loro segmenti linearmente lungo i tre assi x, y e z; corrispondenti ai movimenti laterale, longitudinale e verticale. Le rotazionali invece sono in grado di ruotare i segmenti attorno uno o più assi [10]. In base alle articolazioni di cui un robot è formato esso è in grado di raggiungere uno specifico spazio d'azione. L'insieme dei punti di questo spazio viene definito spazio di lavoro e in base alla geometria di questo spazio il braccio viene inserito in una categoria. Distinguiamo tra attuatori:

- Cartesiani (tre giunti prismatici)
- Cilindrici (due giunti prismatici e uno rotoidale)

- Sferici (un giunto prismatico e due rotoidali)
- Selective Compliance Assembly Robot Arm (SCARA) (due giunti rotoidali e uno prismico)
- Antropomorfi (tre o più giunti rotoidali)

Il robot Adeept è un braccio robotico con geometria antropomorfa.

La geometria antropomorfa è realizzata da 3 giunti di tipo rotazionale, dove l'asse di rotazione del primo è ortogonale agli assi di rotazione dei secondi due, che sono invece paralleli tra loro. Dalla somiglianza col braccio umano, la seconda articolazione è detta "spalla" e la terza è detta "gomito". La struttura antropomorfa è la più agile tra le geometrie, ma perde la corrispondenza tra i giunti e lo spazio cartesiano, rendendo più complesso lo studio. Il suo spazio di lavoro è approssimativamente una sfera, visibile in figura 4.1, ed ha quindi un grande volume rispetto all'ingombro del robot stesso [8].

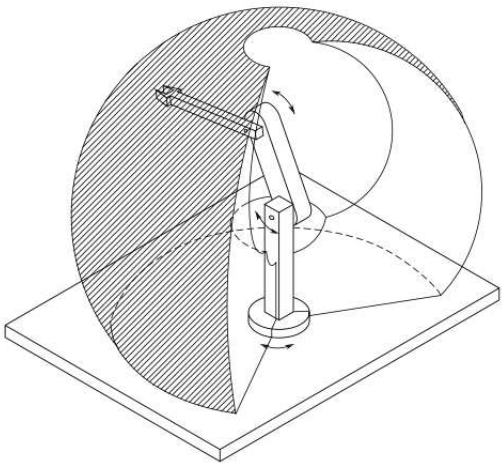


Figura 4.1: Spazio di lavoro di un robot antropomorfo, figura presa da [8]

4.2 Controllo di un braccio robotico

≪La Cinematica è la scienza che ha per oggetto lo studio del moto indipendente delle forze che lo producono. [...] Noi ci proponiamo di studiare le leggi del moto delle macchine dal punto di vista cinematico. Sotto questo aspetto la macchina può considerarsi come un complesso di corpi, combinati in guisa, che quando uno di essi vien messo in

movimento, tutti gli altri acquistano determinati movimenti allo scopo di ottenere un dato lavoro od altro effetto» (Tessari, 1885)

[7].

4.2.1 Cinematica Diretta

La cinematica diretta è lo studio della posizione del braccio rispetto agli angoli assunti dai giunti. Con "posizione del braccio" si intende la posizione dell'utensile (end-effector) rispetto ad un sistema di coordinate fissate alla base del braccio. La tecnica più famosa per il calcolo della cinematica diretta è la procedura di Denavit-Hartenberg. Questa tecnica permette di ricavare una mappa della cinematica diretta, ovvero una serie di matrici di rototraslazione che moltiplicate tra di loro danno la posizione dell'end-effector rispetto alle coordinate base.

4.2.2 Cinematica Inversa

La cinematica inversa consiste nell'ottenere, data una posizione nello spazio dell'end-effector, gli angoli che devono assumere i vari motori perché la punta si trovi in quella esatta posizione. Purtroppo non esiste alcuna tecnica di tipo generale che applicata sistematicamente dia una soluzione, anzi si può addirittura ottenere che la soluzione non è unica o persino non esiste. Se non esistono soluzioni si dice che il punto si trova all'esterno dello spazio di lavoro. Le tecniche risolutive sono essenzialmente due [8]:

- una ad approccio *algebrico*. Ovvero il manipolare le equazioni cinematiche dirette fino ad ottenere delle equazioni invertibili.
- una ad approccio *geometrico*. Ovvero si basta sullo studio della struttura fisica del manipolatore.

Nel caso di questo progetto si è usato un approccio geometrico.

4.3 Controllo del braccio robotico ADA031-V4.0

Per ottenere la mappatura degli angoli corrispondenti alle coordinate di ogni tasto, si è cercato prima di tutto di semplificare il più possibile l'analisi della cinematica inversa. Si è immediatamente osservato che il primo servomotore, quello posizionato sulla pedana rialzata, era studiabile indipendentemente dagli

altri due. Si è quindi prima considerato il piano su cui giace la tastiera e per ogni tasto si sono considerate le coordinate polari. L'angolo delle coordinate polari, preso rispetto ad un asse parallelo alla lunghezza della tastiera, corrisponde con l'angolo da associare al primo servo. Se ne può vedere un esempio in figura 4.2.

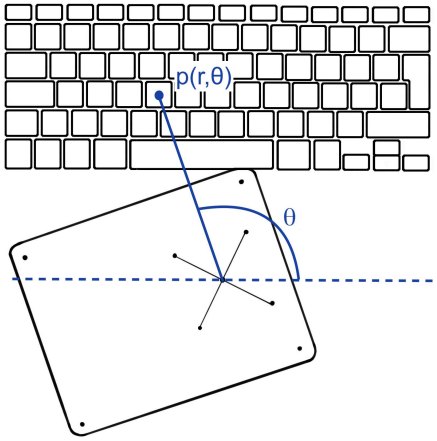


Figura 4.2: Esempio di angolo associato al primo servo per un tasto

Per i secondi due servo invece si è andato a considerare il piano, perpendicolare a quello sopra considerato, dove giacciono i secondi due assi del robot. Su questo piano si è andato a nominare gli angoli e le lunghezze di nostro interesse come visibile in figura 4.3.

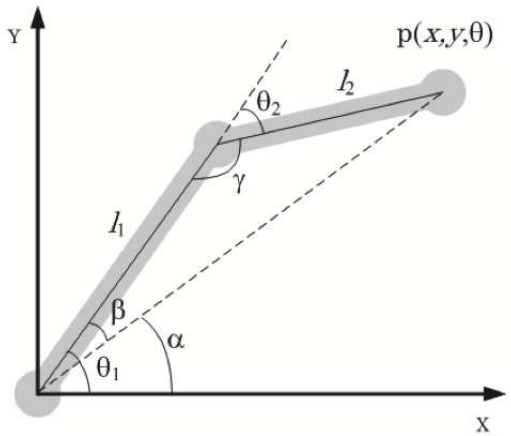


Figura 4.3: Analisi geometrica dei secondi due assi del braccio, figura dal libro [15]

Con questo approccio geometrico è possibile usare il teorema del coseno. Si ottiene quindi che

$$(x^2 + y^2) = l_1^2 + l_2^2 - 2l_1 l_2 cos(180 - \theta_2) \tag{4.1}$$

Osservando che $cos(180 - \theta_2) = -cos(\theta_2)$ allora l'equazione (4.1) diventa

$$(x^2 + y^2) = l_1^2 + l_2^2 + 2l_1 l_2 cos(\theta_2) \tag{4.2}$$

E risolvendo l'equazione per θ_2 si ottiene

$$\theta_2 = \arccos\left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2}\right) \tag{4.3}$$

Tornando a guardare la figura 4.3, vediamo che per il teorema dei seni vale

$$\frac{\sin(\beta)}{l_2} = \frac{\sin(\gamma)}{\sqrt{x^2 + y^2}} \tag{4.4}$$

$$\alpha = \arctan\left(\frac{y}{x}\right) \tag{4.5}$$

Osservando che $\sin(\gamma) = \sin(180 - \theta_2) = \sin(\theta_2)$, si può sostituire $\sin(\gamma)$ con $\sin(\theta_2)$ nell'equazione (4.4), ottenendo così

$$\beta = \arcsin\left(\frac{l_1 \sin(\theta_2)}{\sqrt{x^2 + y^2}}\right) \tag{4.6}$$

E infine, ricordando che $\theta_1 = \beta + \alpha$, si può ottenere θ_1 come

$$\theta_1 = \arcsin\left(\frac{l_1 \sin(\theta_2)}{\sqrt{x^2 + y^2}}\right) + \arctan\left(\frac{y}{x}\right) \tag{4.7}$$

Con questo metodo, descritto nel libro Robot Arms [15], si sono riuscite ad ottenere le equazioni che permettono la cinematica inversa. Ma non si è tenuto conto dell'end-effector.

Per sistemare questo dettaglio si è ipotizzato che il puntatore sia esattamente a 90° rispetto l'asse l_2 ed esattamente sulla punta dell'asse.

Se questo è vero si può considerare il triangolo formato dal link l_2 e dall'end-effector. Presa l'ipotenusa del triangolo, in figura 4.4 chiamata l_4, essa può essere considerata come un fittizio terzo link. Utilizzando l_4 al posto di l_2 nelle equazioni precedenti, l'angolo ottenuto dall'equazione (4.3) sarà la somma degli angoli θ_2 e ξ.

La lunghezza di l_4 per il teorema di Pitagora sarà

$$l_4 = \sqrt{l_2^2 + l_3^2} \tag{4.8}$$

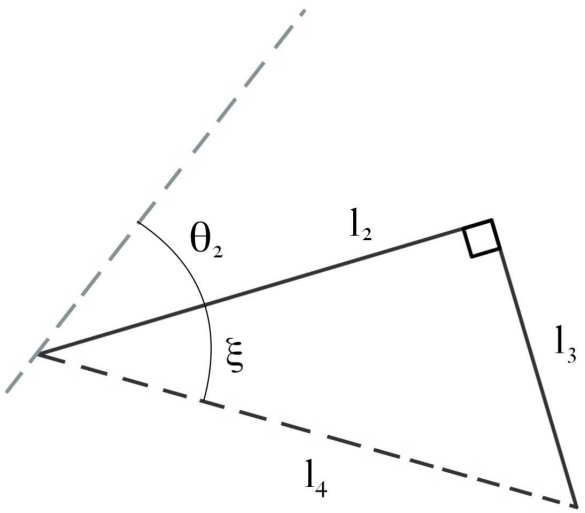


Figura 4.4: Terzo asse e puntatore

E quindi, grazie alla trigonometria, si può ottenere ξ:

$$\xi = \arccos\left(\frac{l_2}{l_4}\right) \tag{4.9}$$

Utilizzando le equazioni (4.3) e (4.7) sugli assi l_1 ed l_4 otterremo gli angoli θ_1 e il fittizio θ_2, dal quale si ricava il vero angolo togliendo lo ξ calcolato nell'equazione (4.9).

Si ricordi infine che l_1 ed l_2 sono stati misurati e sono visibili in tabella 3.2. L'ascissa e l'ordinata si ottengono rispettivamente come il raggio della coordinata polare del tasto e l'altezza dei tasti rispetto il secondo giunto. Si osservi che, lavorando con una tastiera planare, la y è una costante. Per la precisione è una costante negativa, visto che il piano d'appoggio della tastiera è inferiore a quello dove giace la spalla.

Capitolo 5

Codice a blocchi

Come si può vedere nella figura 5.1 il codice è stato sviluppato in due linguaggi separati, la prima parte di calcolo su Matlab e la seconda esecutiva su Arduino. Questa scelta è stata fatta considerando le caratteristiche di velocità computativa dei due ambienti di lavoro. Arduino, lavorando su un microcontrollore di minor potenza rispetto a Matlab, è più lento e ha una memoria molto più limitata. Il codice Matlab viene infatti eseguito una sola volta su un elaboratore, mentre è il codice Arduino che viene caricato sulla scheda elettronica del braccio robotico, su cui continua ad essere eseguito in loop fintanto che il robot è alimentato.

Sul codice `keyscoordinates.m` avviene l'effettivo calcolo degli angoli corrispondenti a ciascun servomotore, che vengono poi stampati a video tramite la funzione `printAngles()`. Questa fornisce gli angoli in un formato facile da trasferire al codice `Robot_.ino`, nello specifico alla funzione `writeLetter()`.

La prima azione fatta dal codice è una mappatura cartesiana dei tasti, in relazione al piano su cui giacciono. Bisogna quindi specificare che è stata fatta in corrispondenza di un set di assi, x e y, paralleli rispettivamente alla lunghezza e alla larghezza della tastiera, con origine nel centro della base circolare del braccio.

Bisogna inoltre specificare che la rotazione della tastiera ha lo scopo di far corrispondere la posizione assunta dal primo motore all'angolo zero con l'asse delle ascisse. Il come si è scelto dove posizionare esattamente il robot e il calcolo della rotazione della base (visibile nelle figure 2.1 e 4.2) sono spiegati più approfonditamente nel Capitolo 6.

I blocchi verranno spiegati più specificatamente nelle prossime due sezioni.

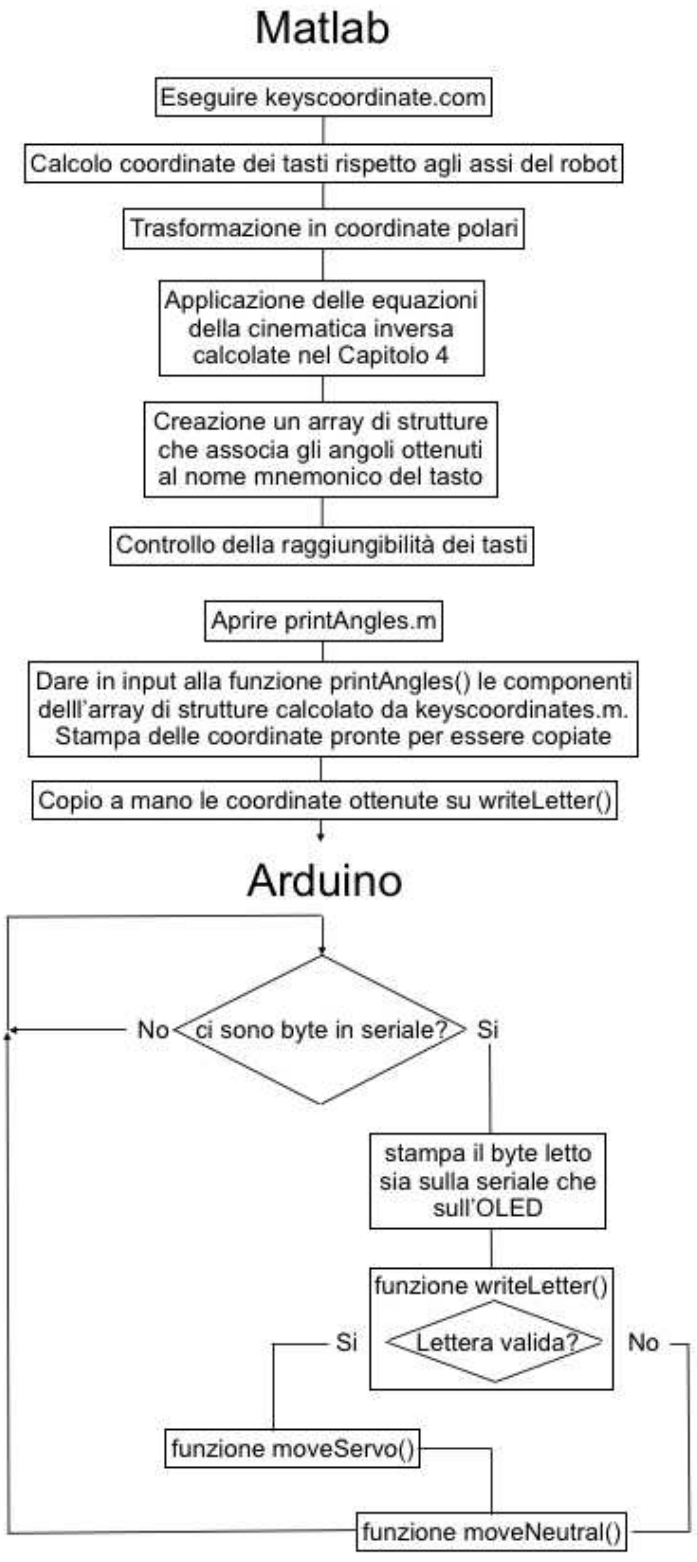


Figura 5.1: Schema a blocchi del codice

5.1 Matlab

L'obiettivo di questa parte di codice è ottenere l'esatto valore degli angoli associati ad ogni tasto, stamparli in un formato comodo per il trasferimento al codice Arduino e individuare se sono presenti tasti non raggiungibili dal robottino sulla tastiera. Il primo obiettivo è il più complesso, per raggiungerlo i passi sono molteplici, principalmente divisi in: fare una mappatura cartesiana dei tasti, trasformare la mappatura da cartesiana a polare, e applicare le formule trovate nel Capitolo 4.

5.1.1 Mappatura cartesiana dei tasti

```
%keys[riga,ennesimo tasto della riga, 1=x 2=y]
cartesian = zeros(6,14,2);

%dimensioni dei 7 tipi di tasti lunghezzaXaltezza 1: prima riga e
    frecce; 2: lettere e numeri; 3: tab; 4: maiusc; 5:cmd; 6:
   shift lungo; 7: spazio
dimkeys_larghezza = [16 ; 15.5 ; 25 ; 30 ; 20 ; 39 ; 89];
dimkeys_lunghezza = [9; 15.5]; %prima riga; righe dalla due alla
   5 (la sesta, che ha un altro valore,    trattata separatamente
   )
dimSpace = 3.8;
```

Listing 5.1: definizione struttura e dimensioni tasti

Per prima cosa sono state prese le misure di tutti i tasti della tastiera. Essi si presentano in 5 diverse dimensioni di larghezza, come si può vedere nello schema 5.2, mentre solo 3 in altezza: la prima riga, l'ultima e il gruppo di quelle in mezzo.

Individuate le tipologie di tasto, per fare l'effettivo calcolo serviva una struttura di supporto. Si è creata una matrice tridimensionale $6\times14\times2$ rappresentante la tastiera. Come in una scacchiera, i primi due indici fanno da coordinate per la posizione del tasto, partendo a contare dal più in alto a sinistra. Il terzo indice serve per indicare se ci si riferisce alla prima coordinata (l'ascissa) o la seconda (l'ordinata). Per esempio i valori associati carattere "e" saranno quelli in posizione e(x)=(3,4,1) ed e(y)=(3,4,2), in quanto è il quarto carattere della terza riga. Nelle coordinate a cui non corrispondeva nessun tasto si è dato il valore Not a Number (NaN). Per esempio la riga 6 ha solo 11 tasti, quindi i valori della matrice (6,12,*) = (6,13,*) = (6,14,*) contengono il valore "NaN".

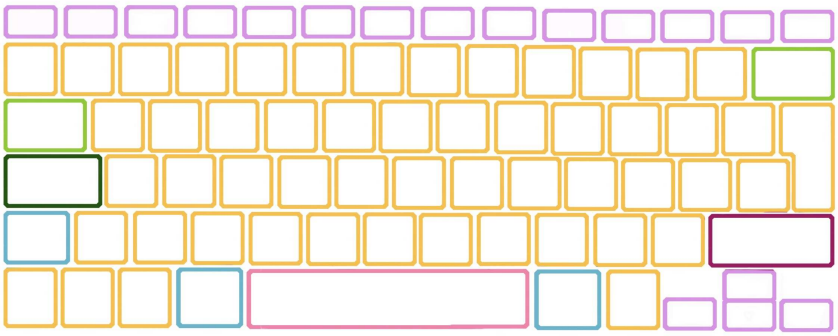


Figura 5.2: Tasti colorati per larghezza

Il passo successivo è l'effettivo calcolo delle ascisse e delle ordinate dei centri dei tasti. È stato trattato dove possibile con dei cicli automatizzati o altrimenti con dei calcoli specifici per le sezioni più complesse. Le ordinate sono risultate più automatizzabili delle ascisse, si è prima misurata l'altezza della prima riga, ovvero l'ordinata massima `ymax`, e si è poi andato a diminuire il valore ad ogni ciclo della quantità necessaria ad arrivare al centro del tasto successivo.

Allo stesso modo si è studiata l'ascissa, partendo a calcolare dall'estremo sinistro della tastiera e andando ad aggiungere allo stesso modo. Scegliendo questa partenza è stato poi necessario imporre un offset a tutte le ascisse, per spostare lo zero sull'asse x inizialmente definito.

È possibile vedere le coordinate ottenute nel grafico 5.3.

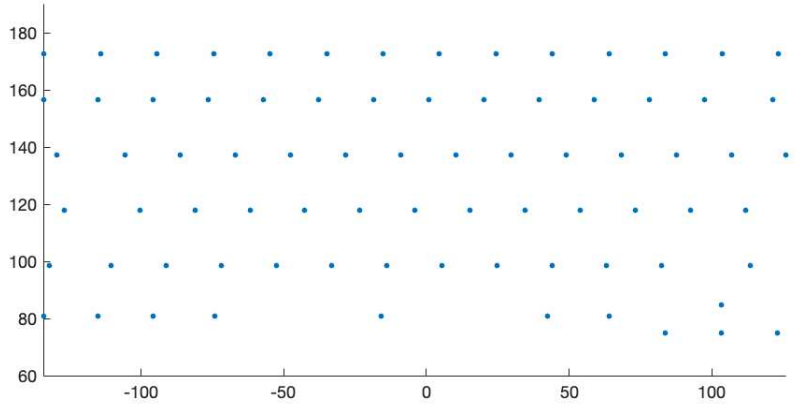


Figura 5.3: Grafico delle coordinate cartesiane dei tasti

5.1.2 Trasformazione da coordinate cartesiane a coordinate polari

```
%% trasformazione in coordinate polari

%polar = [riga tasto, colonna tasto, 1=raggio 2=theta in gradi]
polar = zeros(6,14,2);

%da trigonometria so che raggio    sqrt(x^2 + y^2)
x_squared = cartesian(:,:,1).^2;
y_squared = cartesian(:,:,2).^2;
for j = 1:14
    for i = 1:6
        if(cartesian(i,j,1) == -150)
            polar(i,j,1) = -1;
        else
        polar(i,j,1) = sqrt(x_squared(i,j)+y_squared(i,j));
        end
    end
end

%da trigonometria so che angolo    90 se x=0; arctan(y/x) se x&y
    >0;
%arctan(y/x)+pi se x<0 (atan2d fa gi  la suddivisione)
angoloOffset = 8.4;
for j = 1:14
    for i = 1:6
        polar(i,j,2) = atan2d(cartesian(i,j,2),cartesian(i,j,1))-
    angoloOffset;
    end
end
```

Listing 5.2: trasformazione coordinate cartesiane in polari

Per il calcolo delle coordinate polari si sono applicate le formule per la trasformazione da coordinate cartesiane (x, y) a polari (r, θ).

Il raggio, che si ricordi verrà poi usato come coordinata x nelle formule della cinematica inversa, è ottenuto come:

$$r = \sqrt{x^2 + y^2} \tag{5.1}$$

Per quanto riguarda l'angolo si è sfruttata la funzione `atan2d`, che tiene già conto delle varie casistiche dell'arcotangente nei quattro quadranti. Ad ogni

angolo è stato infine tolto un offset, dettaglio che verrà meglio spiegato nel Capitolo 6.

Il risultato è stato salvato nella matrice `polar`.

5.1.3 Calcolo degli angoli tramite la Cinematica Inversa

```
%lunghezza link tra giunto1 e giunto2
s1 = 65;
%lunghezza link finale
t = 147;
%lunghezza penna
l = 55;
%lunghezza  e angolo rispetto a t del vettore t+l
s2 = sqrt(t^2 + l^2);
theta = atand(l/t);

%altezza asse da terra
h = -85;

%angoli finali servos: angles = [servo1, servo2, servo3]
angles = zeros(6,14,3);
alpha = zeros(6,14);
beta = zeros(6,14);
frac = zeros(6,14);
sinn = zeros(6,14);
for i = 1:6
    for j = 1:14
        angles(i,j,1) = polar(i,j,2);
        angles(i,j,3) = acosd((polar(i,j,1)^2+ h^2 -s1^2-s2^2)
   /(2*s1*s2));
        sinn(i,j) = sind(angles(i,j,3));
        frac(i,j) = (s2*sinn(i,j))/(sqrt(polar(i,j,1)^2+ h^2));
        beta(i,j) = asind(frac(i,j));
        alpha(i,j) = atand (h/polar(i,j,1));
        angles(i,j,2) = abs(beta(i,j)) - abs(alpha(i,j));
    end
end

for i = 1:6
    for j = 1:14
        angles(i,j,3) = angles(i,j,3)- theta;
    end
end
```

Listing 5.3: applicazione delle formule della cinematica inversa

Qui sono state applicate le formule dimostrate nel Capitolo 4. I valori ottenuti sono salvati nella matrice 6×14×3 `angles`.

5.1.4 Controllo della raggiungibilità dei tasti

```
for i = 1:6
    for j = 1:14
        if angles(i,j,2) < 0
            fprintf('l''angolo 2: %d %d risulta negativo \n',i,j)
    ;
        end
        if imag(angles(i,j,2))~=0
            fprintf('l''angolo 2: %d %d risulta complesso \n',i,j
    );
        end
        if angles(i,j,3) < 0
            fprintf('l''angolo 3: %d %d risulta negativo \n',i,j)
    ;
        end
        if imag(angles(i,j,3))~=0
            fprintf('l''angolo 3: %d %d risulta complesso \n',i,j
    );
        end
    end
end
```

Listing 5.4: controllo dei tasti non raggiungibili dal robot

Alcuni tasti non sono fisicamente raggiungibili dal braccio, a causa delle sue limitazioni. Si è quindi creata questa funzione per osservare, anche in base al puntatore usato, quando gli angoli calcolati dalle equazioni della sezione precedente risultano complessi o negativi. Questa funzione è necessaria in quanto la funzione `printAngles()` stampa gli angoli complessi come reali e potrebbe essere causa di confusione. Nel caso specifico di questo progetto il ciclo da in output:

```
l'angolo 3: 1 1 risulta negativo
l'angolo 3: 1 1 risulta complesso
l'angolo 2: 1 2 risulta negativo
l'angolo 3: 1 2 risulta negativo
l'angolo 3: 1 2 risulta complesso
l'angolo 2: 1 13 risulta negativo
l'angolo 2: 1 14 risulta negativo
l'angolo 3: 1 14 risulta negativo
```

```
l'angolo 3: 1 14 risulta complesso
l'angolo 2: 2 1 risulta negativo
l'angolo 3: 2 1 risulta negativo
l'angolo 3: 2 1 risulta complesso
l'angolo 2: 2 14 risulta negativo
```

Ciò significa che i tasti "esc", "F1", "F12", "On/Off" e "\" non sono fisicamente raggiungibili in questo progetto.

5.1.5 Funzione printAngles()

```
function printAngles(x)
    fprintf('\n');
    for k = 1:3
        kk=k-1;
        fprintf('finalAngle[%d]= %4.2f; \n', kk, x(k));
    end
    fprintf('\n');
end
```

Listing 5.5: funzione printAngles

Questa funzione permette di ottenere facilmente i valori degli angoli corrispondenti ad un tasto, senza dover contare quali siano le coordinate corrispondenti sulla matrice `angles`. È stata prima creata una struttura dati che associa un valore mnemonico ad ogni tris di angoli (visibile nel codice `keyscoordinates.m` in appendice). Dato in input il valore mnemonico alla funzione essa ritorna gli angoli come una stringa. Per esempio, dato in input alla funzione `alphabet.e`, l'output risultante è:

```
finalAngle[0]= 107.59;
finalAngle[1]= 37.88;
finalAngle[2]= 67.18;
```

5.2 Arduino

L'obiettivo di questa parte di codice è leggere i caratteri che arrivano sulla seriale, associarli alle corrette coordinate e muovere il braccio di conseguenza. Questo è ottenuto grazie alle funzioni `writeLetter()`, `moveNeutral()` e `moveServo()`, che sono state scritte appositamente per questo programma e verranno esposte nel dettaglio nei prossimi paragrafi.

5.2.1 Ciclo loop

Nel ciclo loop si può leggere

```
1  if (Serial.available() > 0) {
2    byte ch = Serial.read(); //legge il byte in entrata
3    display.fillScreen(BLACK); //metto l'OLED tutto nero
4    Serial.print("I received: ");
5    Serial.println((char)ch); //alla fine delle lettere legge
     sempre anche un "Line feed" (ascii 10)
6    display.setTextSize(5);
7    display.print((char)ch);
8    //Inizia a mostrare
9    display.display();
10   //movimento del robot
11   writeLetter(ch);
12   delay(1000);
13   moveNeutral();
14   delay(1000);
15 }
```

Listing 5.6: Loop codice Arduino "Robot_"

Questa condizione è il cuore del ciclo loop. Attende che sia disponibile un valore dalla seriale e, appena se ne presenta uno, lo salva nella variabile `ch`. Una volta ricevuto e salvato il carattere, esso viene stampato sia in seriale che sullo schermo OLED. La riga 4 ha lo scopo di aggiornare lo schermo OLED, in modo da non sovrapporre le lettere precedenti con quelle nuove.

Vengono poi utilizzate le funzioni `writeLetter()` e, dopo 1000 ms, `moveNeutral()`. La prima, in base al carattere in input, associa i corretti angoli obbiettivo ai servo e, appoggiandosi alla funzione `moveServo()`, porta il robot a premere il tasto. La seconda invece sposta il robot in una posizione neutra. In questo modo ad ogni ciclo l'ordine delle azioni è: controllo della presenza o meno di byte in input da seriale, salvataggio dell'input come `byte ch`, azionamento del robot che va a premere il tasto corrispondente a `ch` e infine riposizionamento del robot ad una posizione neutra, in attesa del carattere successivo da scrivere.

5.2.2 Funzione writeLetter()

```
1 void writeLetter(char ch){
2
3   //lettere
4
```

```
5   if (ch == 'a'){
6       finalAngle[0]= 122.04;
7       finalAngle[1]= 33.20;
8       finalAngle[2]= 62.89;
9   }
10  else if (ch == 'b'){
11      finalAngle[0]= 89.66;
12      finalAngle[1]= 38.40;
13      finalAngle[2]= 104.57;
14  }
```

[...]

```
1   else if (ch == '.'){
2     finalAngle[0]= 48.91;
3     finalAngle[1]= 52.41;
4     finalAngle[2]= 92.33;
5   }
6   else if (ch == '-'){
7     finalAngle[0]= 41.65;
8     finalAngle[1]= 46.92;
9     finalAngle[2]= 83.98;
10  }
11  else{ //nel caso sia stato premuto un tasto non adatto o per il
      valore "newLine a fine frase" metto in posizione neutrale
12    finalAngle[0]= 90;
13    finalAngle[1]= 90;
14    finalAngle[2]= 45;
15    Serial.print("il tasto ");
16    Serial.print((char)ch);
17    Serial.println(" non risulta raggiungibile");
18  }
19  int i = 0;
20  moveServo(i, finalAngle[i]);
21  i = 2;
22  moveServo(i, finalAngle[i]);
23  i = 1;
24  moveServo(i, finalAngle[i]);
25 }
```

Listing 5.7: codice funzione writeLetter() Arduino "Robot_"

Qui in base al `byte ch` vengono selezionati gli angoli corretti, calcolati precedentemente su Matlab. Non c'è una struttura di comunicazione diretta tra i codici nei due linguaggi, quindi l'inserimento dei valori avviene manualmente. Creare

un canale di comunicazione diretta potrebbe essere un interessante miglioramento futuro, come evidenziato nel capitolo Capitolo 7.

Se il carattere non è raggiungibile perché viola uno degli assunti, allora il codice entra nell'`else` a riga 11. Questo stampa in seriale l'avviso *"il tasto __ non risulta raggiungibile"* e seleziona gli angoli per la posizione neutrale.

Infine il codice utilizza la funzione `moveServo()` per l'effettivo movimento all'angolo desiderato dei servo. Si osservi che i motori non vengono azionati in ordine crescente, ma il terzo (il gomito) viene mosso prima del secondo (la spalla). La sequenza di attivazione dei servomotori ha un'ordine preciso per evitare che l'end-effector vada a colpire le superfici esterne al target interessato. Difatti la traiettoria che il puntatore percorre per passare dalla posizione di risposo a quella finale dipende dall'ordine in cui vengono azionati i servo, vista l'impossibilità di muoverli tutti contemporaneamente. Si è empiricamente dimostrato che, azionando prima il gomito della spalla, il rischio che la traiettoria di spostamento vada a colpire la tastiera, o comunque luoghi indesiderati, diventa nullo.

5.2.3 Funzione moveServo()

```
1  void moveServo(int num,double finalAngle){
2    int offet_degrees = 4;
3    boolean done = false;
4    while(!done){
5      if (angle[num] < finalAngle){
6        angle[num]=angle[num]+offet_degrees;
7        if (angle[num] > posMax){ //if di controllo che non si esca
       dai bound inizialente posti
8          angle[num] = posMax;
9          done = true;
10       }
11       else if(angle[num] >= finalAngle){ //sono arrivata alla
       fine?
12         angle[num] = finalAngle;
13         done = true;
14       }
15       servo[num].write(angle[num]);
16     }
17     else if (angle[num] > finalAngle){
18       angle[num]=angle[num]-offet_degrees;
19       if (angle[num] < posMin){ //if di controllo che non si esca
       dai bound inizialente posti
20         angle[num] = posMin;
```

```
21          done=true;
22        }
23        else if(angle[num] <= finalAngle){ //sono arrivata alla
    fine?
24          angle[num] = finalAngle;
25          done = true;
26        }
27        servo[num].write(angle[num]);
28      }
29      else{
30        done = true;
31      }
32      delay(70);
33    }
34  }
```

Listing 5.8: codice funzione moveServo() Arduino "Robot_"

Questa funzione ha l'unico scopo di muovere in maniera più fluida i motori. Esiste già una funzione, parte della libreria Servo, che muove il servo alla posizione finale. Si tratta della funzione `write()`. In un servo standard essa fa ruotare il servo fino alla posizione angolare voluta. In un servo a rotazione continua invece essa setta la velocità di rotazione del motore (0 indica che ruota a velocità massima in una direzione, 180 che gira a velocità massima nell'altra direzione, 90 che è fermo). In questo progetto ricordiamo che si usano servo standard. La funzione `write()` però porta a degli spostamenti molto bruschi e rischia di rovinare gli stessi servo. Si è così sviluppata questa funzione `moveServo()`.

La funzione `moveServo()` richiede in input un `int num`, che è il servo che si desidera muovere e un `double finalAngle`, che è l'angolo a cui si desidera portare il servo.

Questa funzione per prima cosa, grazie ad una serie di condizioni, individua se l'angolo a cui si trova attualmente il servo è minore, maggiore o corrisponde all'angolo obiettivo. Una volta individuata la casistica la struttura è la stessa ma simmetrica rispetto ai segni. Si modifica di 4 gradi l'angolo attuale e, prima di muovere effettivamente il motore, si va ad osservare dove ha portato questa modifica. Se l'angolo nuovo è maggiore o uguale all'angolo obiettivo allora `done` viene impostato su `true`, e il servo va all'angolo obiettivo con uno spostamento compreso tra gli 0 e i 4 gradi. Altrimenti il servomotore fa una rotazione di 4 gradi e il loop ricomincia aumentando di altri 4 gradi.

Le condizioni `if angle[num] > posMax` e `if angle[num] < posMin` servo-

no in caso ci siano degli angoli massimi oltre i quali non si vuole che i servo vadano. È una caratteristica utile nel caso ci siano impedimenti fisici attorno alla tastiera come un soffitto da non colpire per esempio. In questo specifico progetto non c'era necessità quindi sono stati settati come `posMin = 0` e `posMax = 180`.

Infine si attendono 70 ms per rendere il movimento più lento e quindi meno oneroso sui motori.

5.2.4 Funzione moveNeutral()

```
1 void moveNeutral(){ //angolo1 = 90, angolo2 = 45, angolo0 = non
    cambiato
2   //secondo servo
3   moveServo(1,90);
4   delay(100);
5   //primo servo
6   moveServo(2,45);
7 }
```

Listing 5.9: codice funzione moveNeutral() Arduino "Robot_"

L'ultima funzione che ho creato è stata `moveNeutral()`, che ha l'obiettivo di riportare il braccio ad una posizione "neutrale" tra una lettera e l'altra. Il passare sempre per una posizione di riposo tra una lettera e l'altra evita molte casistiche di scontro tra il puntatore e la tastiera nel movimento, come il caso del muovere per primo il primo servo, dove si va a trascinare l'end-effector sulla tastiera. Questa funzione si basa su `moveServo()` e portando gli assi sempre alla stessa posizione non richiede nessun parametro in entrata.

Capitolo 6

Verifica funzionamento

Questo capitolo è dedicato alle calibrazioni fatte prima di scrivere il codice per il suo corretto funzionamento ed a tutti gli aggiustamenti applicati durante la fase di test per migliorarne la precisione.

Per fare queste misurazioni è stata inizialmente preparata una copia planare della tastiera dell'elaboratore "MacBook Air (13 pollici, 2017)", tastiera su cui sono stati fatti tutti i test. Questa copia ha permesso di muovere il robot senza il rischio di danno e di poter modificare il codice durante i test, vista l'impossibilità di usare la tastiera reale mentre il robot vi è sopra.

La posizione cartesiana del braccio rispetto alla tastiera è stata scelta come prima cosa. Il braccio è stato collocato con l'obiettivo di permettere all'end-effector di raggiungere più tasti possibile, raggiungendo il maggior range d'azione disponibile. La rotazione della base è stata calcolata solo in seguito.

Inizialmente, dopo il montaggio del robot, si sono fatte delle prove di funzionamento della struttura e, al momento dei test sui motori, si è notato che la posizione presa all'angolo 0° non era quella desiderata per nessuno dei tre. Nel caso del gomito e della spalla si è deciso di risolvere il problema meccanicamente. Si è data istruzione al servomotore di posizionarsi all'angolo 0 e si è staccato il corrispettivo asse che è poi stato riattaccato nella direzione desiderata. Questo spostamento meccanico implica un posizionamento non perfettamente corretto, possibile motivo di imprecisioni nella posizione finale. Il primo servo era invece montato in maniera più complessa, ragion per cui si è scelto di ruotare la base per sopperire alla differenza piuttosto che smontare la struttura.

Al momento dei test è sorto che la rotazione scelta per la base non era perfetta. Un offset di errore sull'angolo del primo servo era ancora presente. Piuttosto di fare lunghe modifiche del codice si è preferito fare un aggiustamento all'angolo

togliendo, come visibile nella sezione 5.1.2 del Capitolo 5, l'errore a tutti gli angoli ottenuti dal primo servo.

Alla verifica del funzionamento del codice è stato osservato quanto la lunghezza degli assi fosse una variabile impattante nella precisione dell'end-effector. Ad una variazione di pochi millimetri dei dati inseriti la posizione finale era sensibilmente cambiata. Non erano a disposizione strumenti di misura di precisione, quindi l'errore di misura di circa 1 mm è causa di possibili imprecisioni. Le misure finali sono visibili nella tabella 3.1 del Capitolo 3. È sempre questo il motivo per cui, alla creazione del puntatore si è deciso di farlo di lunghezza ridotta, per evitare che possibili imprecisioni nella perpendicolarità tra esso e l'asse 3 venissero aumentate proporzionalmente alla lunghezza della penna.

Alcuni ultimi perfezionamenti al codice, svolti durante i test di funzionamento, sono stati: il numero di gradi di variazione nella funzione `moveServo()`, e quindi la velocità di spostamento del braccio; il posizionamento delle scritte sullo schermo OLED; si è persino scoperto un limite del codice, ovvero che i tasti "à","è","ì","ò" ed "ù" non sono raggiungibili, argomento trattato meglio nel capitolo Capitolo 7.

Capitolo 7

Miglioramenti futuri

Le possibili migliorie ed espansioni di questa tesi sono molteplici. Possono essere racchiuse in tre categorie: potenziamento dell'hardware, espansione del software, aggiunta di sistemi esterni.

Hardware Dotato di un hardware di maggiore qualità molte delle limitazioni del progetto sarebbero superate. Allungando gli assi il volume dello spazio di lavoro aumenterebbe e si potrebbe raggiungere l'intera tastiera. Utilizzando dei motori più resistenti si potrebbe tenere attivo il robot per lunghi periodi, senza temere il surriscaldamento o danneggiamento degli stessi, e insieme ottenere un movimento più fluido, elegante, preciso e silenzioso. Disponendo invece di un dispositivo a più bracci, o comunque di un sistema robotico più avanzato, si potrebbe risolvere la problematica dell'uso di combinazioni di tasti.

Software Dal punto di vista software una prima miglioria dovrebbe essere un metodo per il trasferimento dati tra i codici, in quanto l'inserimento manuale degli angoli nel codice esecutivo è un'operazione tediosa. Questo trasferimento d'informazioni potrebbe essere risolto attraverso il controllo di Arduino direttamente tramite Matlab. Sarebbe interessante risolvere la questione delle lettere accentate, rendendo completo l'uso della tastiera. Difatti i tasti "à","è","ì","ò" ed "ù" non sono contenuti nel codice ASCII standard quindi, al contrario del resto dei tasti che sono rappresentati da 1 byte, questi sono indicati da 2. Di conseguenza non vengono letti correttamente dal codice e non possono essere usati. Questo è un dettaglio risolvibile scrivendo la vocale seguita da un apostrofo, ma si tratta sicuramente di un possibile miglioramento futuro. Infine un'implementazione desiderabile sarebbe un'interfaccia orientata al cliente, su cui esso possa inserire

l'input, al posto dell'attuale uso del monitor seriale di Arduino. Basandosi sulla stessa idea si potrebbe utilizzare al riguardo anche lo schermo OLED, rendendo l'uso più coinvolgente per l'utente.

Sistemi Esterni Non è solo perfezionando ciò che è già presente che si può migliorare il progetto. L'aggiunta di sensori potrebbe essere una componente in grado di dare una svolta alla precisione del braccio: sensori di pressione per evitare il rischio che il tasto venga premuto troppo e quindi rovinato; telecamere per migliorare, attraverso programmi per la *computer vision*, la precisione nel centrare il tasto. La presenza di telecamere abbatterebbe anche la barriera del doversi limitare ad una tastiera singola e del dover essere saldamente posto in una precisa posizione. Ovviamente il codice è già attualmente estendibile ad altre tastiere, ma la parte di calibrazione cartesiana dei tasti va completamente ricalcolata. Anche l'annessione di un modulo bluetooth potrebbe rivelarsi utile. Il robot attualmente necessita di essere costantemente connesso con cavo al compilatore da dove arriva la sorgente input, ma con un modulo bluetooth sulla scheda questo limite si potrebbe superare. Il robot necessita comunque di un'alimentazione di qualche tipologia, ma si può risolvere connettendo delle batterie all'interfaccia pin per l'alimentazione esterna della scheda.

Capitolo 8

Conclusioni

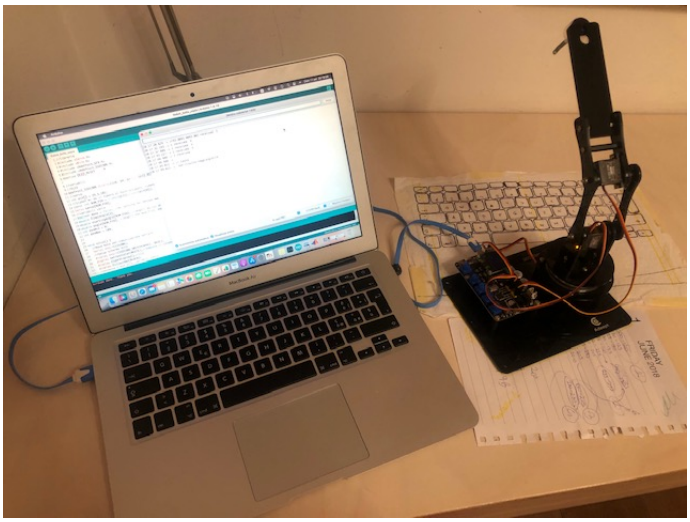


Figura 8.1: Foto del sistema completo per la funzione del robot

Il progetto tratta un braccio robotico a 3 assi, programmato al fine di battere a tastiera una frase ricevuta in input. Per ottenere software funzionante le fasi di apprendimento sono state diverse.

Prima è servito uno studio approfondito riguardo l'hardware a disposizione, per capire cosa fosse fisicamente possibile e come fosse controllabile il robot. Nel Capitolo 3 sono esposti i risultati di questa ricerca: prima le misure fisiche del robot, poi le caratteristiche ed il funzionamento dei motori, per finire con la descrizione della scheda elettronica.

Nel frattempo è stata affrontata la ricerca riguardo l'aspetto teorico del movimento di un braccio robotico. Si è andato a trovare la terminologia associata ai bracci robotici, le loro caratteristiche e classificazioni. Si è andata a studiare la

cinematica diretta e a cercare quella inversa, la seconda necessaria allo sviluppo del progetto e le cui equazioni sono esposte al Capitolo 4.

Queste conoscenze hanno portato allo sviluppo software finale. Sviluppato in parte su Matlab e in parte su Arduino, esso copre il processo dal calcolo delle coordinate dei tasti al movimento dei motori. Precedentemente all'accensione del braccio, un codice calcola il valore degli angoli a cui ciascun motore servo deve trovarsi, affinché il braccio prema il tasto. Dopo l'accensione, una stringa arriva da seriale sulla scheda Adeept Arm Drive Board, dove sul microcontrollore è stato precedentemente caricato il programma `Robot_.ino`, e viene processata byte per byte. Ogni byte corrisponde ad una lettera, che viene premuta dal braccio, se raggiungibile. Su Matlab è avvenuta la parte di calcolo per gli angoli, mentre sul codice Arduino si è andati a fare il controllo del movimento del braccio. Il codice è spiegato nelle sue sezioni più essenziali al Capitolo 5, ed incluso nella sua totalità in Appendice A.

Il progetto presenta ancora delle forti limitazioni che potrebbero essere superate attraverso migliorie hardware, software o dall'aggiunta di sensori. Queste sono state elencate nel Capitolo 7, con il quale si è concluso questo progetto e, quindi, questa tesi.

Appendice A

Codice

A.1 Matlab

```
clc;
clear all;

%keys[riga,ennesimo tasto della riga, 1=x 2=y]
cartesian = zeros(6,14,2);

%dimensioni dei 7 tipi di tasti lunghezzaXaltezza 1: prima riga e
    frecce; 2: lettere e numeri; 3: tab; 4: maiusc; 5:cmd; 6:
   shift lungo; 7: spazio
dimkeys_larghezza = [16 ; 15.5 ; 25 ; 30 ; 20 ; 39 ; 89];
dimkeys_lunghezza = [9; 15.5]; %prima riga; righe dalla due alla
   5 (la sesta, che ha un altro valore,    trattata separatamente
   )
dimSpace = 3.8;

%% ordinate
%prima riga
ymax = 172.5; %tasti pi  alti
for j = 1:14
    cartesian(1,j,2) = ymax;
end
%righe 2-5
for j = 1:14
    %alla seconda riga togliamo mezzo tasto del tipo1, uno spazio
   , mezzo tasto del tipo 2
    cartesian(2,j,2)=cartesian(1,1,2) - (dimkeys_lunghezza(1)/2 +
    dimSpace + dimkeys_lunghezza(2)/2);
    for i=3:5
```

```
23          %alle altre righe tolgo sempre 2 mezzi tasti del tipo 2 e
        uno spazio
24          cartesian(i,j,2) = cartesian(i-1,1,2) - (
       dimkeys_lunghezza(2) + dimSpace);
25      end
26  end
27  %ultima riga
28  for j = 1:14
29      cartesian(6,j,2) = 81;
30  end
31  % 4 frecce hanno altezze speciali
32  cartesian(6,8,2) = 75; %<-
33  cartesian(6,11,2) = 75; %->
34  cartesian(6,10,2) = 75; %freccia giu
35  cartesian(6,9,2) = 85; %freccia su
36
37  %% ascisse
38  %parto con x=0 sul lato esterno dei tasti pi   a sinistra (
       metter  poi un offset)
39  %la prima colonna ha tutti tasti di tipo diverso, quindi
       inserisco a mano
40  cartesian(1,1,1)=dimkeys_larghezza(1)/2;
41  cartesian(2,1,1)=dimkeys_larghezza(2)/2;
42  cartesian(3,1,1)=dimkeys_larghezza(3)/2;
43  cartesian(4,1,1)=dimkeys_larghezza(4)/2;
44  cartesian(5,1,1)=dimkeys_larghezza(5)/2;
45  cartesian(6,1,1)=dimkeys_larghezza(2)/2;
46
47  %prima riga
48  for j = 2:14
49      %il centro del tasto successivo    mezzo tasto1 + spazio+
       mezzo tasto1
50      cartesian(1,j,1) = cartesian(1,j-1,1) + (dimkeys_larghezza(1)
        + dimSpace);
51  end
52  %seconda riga
53  for j = 2:13
54      cartesian(2,j,1) = cartesian(2,j-1) + dimkeys_larghezza(2) +
       dimSpace;
55  end
56  cartesian(2,14,1) = cartesian(2,13,1) + dimkeys_larghezza(2)/2 +
       dimSpace + dimkeys_larghezza(3)/2;
57  %terza riga
58  cartesian(3,2,1) = cartesian(3,1,1) + dimkeys_larghezza(3)/2 +
```

```
    dimSpace + dimkeys_larghezza(2)/2;
for j = 3:14
    cartesian(3,j,1) = cartesian(3,j-1) + dimkeys_larghezza(2) +
    dimSpace;
end
%quarta riga
cartesian(4,2,1) = cartesian(4,1,1) + dimkeys_larghezza(4)/2 +
    dimSpace + dimkeys_larghezza(2)/2;
%Ignoro l'ultima casella perch  l'invio lo considero solo nella
    riga sopra
for j = 3:13
    cartesian(4,j,1) = cartesian(4,j-1) + dimkeys_larghezza(2) +
    dimSpace;
end
%quinta riga (ultima casella 0 perch  la 5 riga ha solo 13 tasti
    )
cartesian(5,2,1) = cartesian(5,1,1) + dimkeys_larghezza(5)/2 +
    dimSpace + dimkeys_larghezza(2)/2;
for j = 3:12
    cartesian(5,j,1) = cartesian(5,j-1) + dimkeys_larghezza(2) +
    dimSpace;
end
cartesian(5,13,1) = cartesian(5,12,1) + dimkeys_larghezza(2)/2 +
    dimSpace + dimkeys_larghezza(6)/2;
%sesta riga (ha solo 11 tasti)
for j = 2:3
    cartesian(6,j,1) = cartesian(6,j-1) + dimkeys_larghezza(2) +
    dimSpace;
end
cartesian(6,4,1) = cartesian(6,3,1) + dimkeys_larghezza(2)/2 +
    dimSpace + dimkeys_larghezza(5)/2;
cartesian(6,5,1) = cartesian(6,4,1) + dimkeys_larghezza(5)/2 +
    dimSpace + dimkeys_larghezza(7)/2;
cartesian(6,6,1) = cartesian(6,5,1) + dimkeys_larghezza(7)/2 +
    dimSpace + dimkeys_larghezza(5)/2;
cartesian(6,7,1) = cartesian(6,6,1) + dimkeys_larghezza(5)/2 +
    dimSpace + dimkeys_larghezza(2)/2;
%frecce
cartesian(6,8,1) = cartesian(6,7,1) + dimkeys_larghezza(2)/2 +
    dimSpace + dimkeys_larghezza(1)/2;
cartesian(6,9,1) = cartesian(6,8,1) + dimkeys_larghezza(1) +
    dimSpace;
cartesian(6,10,1) = cartesian(6,9,1);
cartesian(6,11,1) = cartesian(6,10,1) + dimkeys_larghezza(1) +
```

```
    dimSpace;

%sposto lo zero delle coordinate x da sinistra al centro (metto l'offset) e
%imposto come NaN i tasti senza coordinate (ovvero le coordinate che non corrispondono a nessun tasto)
offset = 142;
for j = 1:14
    for i = 1:6
        if cartesian(i,j,1)==0
            cartesian(i,j,1) = nan;
            cartesian(i,j,2) = nan;
        else
            cartesian(i,j,1) = cartesian(i,j,1) - offset;
        end
    end
end

%plot
scatter(reshape(cartesian(:,:,1),[],1),reshape(cartesian(:,:,2),[],1), 'fill');
axis('equal');
ylim([60,190]);

for j = 1 : 14
    for i = 1:6
        line([0, cartesian(i,j,1)], [0, cartesian(i,j,2)]);
    end
end

%% trasformazione in coordinate polari

%polar = [riga tasto, colonna tasto, 1=raggio 2=theta in gradi]
polar = zeros(6,14,2);

%da trigonometria so che raggio    sqrt(x^2 + y^2)
x_squared = cartesian(:,:,1).^2;
y_squared = cartesian(:,:,2).^2;
for j = 1:14
    for i = 1:6
        if(cartesian(i,j,1) == -150)
            polar(i,j,1) = -1;
        else
        polar(i,j,1) = sqrt(x_squared(i,j)+y_squared(i,j));
```

```
127         end
128     end
129 end
130 
131 %da trigonometria so che angolo    90 se x=0; arctan(y/x) se x&y
    >0;
132 %arctan(y/x)+pi se x<0 (atan2d fa gi  la suddivisione)
133 angoloOffset = 8.4;
134 for j = 1:14
135     for i = 1:6
136         polar(i,j,2) = atan2d(cartesian(i,j,2),cartesian(i,j,1))-
    angoloOffset;
137     end
138 end
139 
140 %% cinematica inversa
141 
142 %lunghezza link tra giunto1 e giunto2
143 s1 = 65;
144 %lunghezza link finale
145 t = 147;
146 %lunghezza penna
147 l = 55;
148 %lunghezza  e angolo rispetto a t del vettore t+l
149 s2 = sqrt(t^2 + l^2);
150 theta = atand(l/t);
151 
152 %altezza asse da terra
153 h = -85;
154 
155 %angoli finali servos: angles = [servo1, servo2, servo3]
156 angles = zeros(6,14,3);
157 alpha = zeros(6,14);
158 beta = zeros(6,14);
159 frac = zeros(6,14);
160 sinn = zeros(6,14);
161 for i = 1:6
162     for j = 1:14
163         angles(i,j,1) = polar(i,j,2);
164         angles(i,j,3) = acosd((polar(i,j,1)^2+ h^2 -s1^2-s2^2)
    /(2*s1*s2));
165         sinn(i,j) = sind(angles(i,j,3));
166         frac(i,j) = (s2*sinn(i,j))/(sqrt(polar(i,j,1)^2+ h^2));
167         beta(i,j) = asind(frac(i,j));
```

```
168         alpha(i,j) = atand (h/polar(i,j,1));
169         angles(i,j,2) = abs(beta(i,j)) - abs(alpha(i,j));
170     end
171 end
172
173 for i = 1:6
174     for j = 1:14
175         angles(i,j,3) = angles(i,j,3)- theta;
176     end
177 end
178
179
180 %% struttura dati
181
182 %prima riga
183 alphabet = struct;
184 alphabet.esc = squeeze(angles(1,2,:));
185 alphabet.sSun = squeeze(angles(1,2,:));
186 alphabet.bSun = squeeze(angles(1,3,:));
187 alphabet.rectangles = squeeze(angles(1,4,:));
188 alphabet.squares = squeeze(angles(1,5,:));
189 alphabet.sLight = squeeze(angles(1,6,:));
190 alphabet.bLight = squeeze(angles(1,7,:));
191 alphabet.backward = squeeze(angles(1,8,:));
192 alphabet.startStop = squeeze(angles(1,9,:));
193 alphabet.forward = squeeze(angles(1,10,:));
194 alphabet.mute = squeeze(angles(1,11,:));
195 alphabet.sVolume = squeeze(angles(1,12,:));
196 alphabet.bVolume = squeeze(angles(1,13,:));
197 alphabet.onOff = squeeze(angles(1,14,:));
198 %seconda riga
199 alphabet.backslash = squeeze(angles(2,1,:));
200 alphabet.one = squeeze(angles(2,2,:));
201 alphabet.two = squeeze(angles(2,3,:));
202 alphabet.three = squeeze(angles(2,4,:));
203 alphabet.four = squeeze(angles(2,5,:));
204 alphabet.five = squeeze(angles(2,6,:));
205 alphabet.six = squeeze(angles(2,7,:));
206 alphabet.seven = squeeze(angles(2,8,:));
207 alphabet.eight = squeeze(angles(2,9,:));
208 alphabet.nine = squeeze(angles(2,10,:));
209 alphabet.zero = squeeze(angles(2,11,:));
210 alphabet.questionMark = squeeze(angles(2,12,:));
211 alphabet.pow = squeeze(angles(2,13,:));
```

```
212 alphabet.canc = squeeze(angles(2,14,:));
213 %terza riga
214 alphabet.tab = squeeze(angles(3,1,:));
215 alphabet.q = squeeze(angles(3,2,:));
216 alphabet.w = squeeze(angles(3,3,:));
217 alphabet.e = squeeze(angles(3,4,:));
218 alphabet.r = squeeze(angles(3,5,:));
219 alphabet.t = squeeze(angles(3,6,:));
220 alphabet.y = squeeze(angles(3,7,:));
221 alphabet.u = squeeze(angles(3,8,:));
222 alphabet.i = squeeze(angles(3,9,:));
223 alphabet.o = squeeze(angles(3,10,:));
224 alphabet.p = squeeze(angles(3,11,:));
225 alphabet.lbrachet = squeeze(angles(3,12,:));
226 alphabet.plus = squeeze(angles(3,13,:));
227 alphabet.enter = squeeze(angles(3,14,:));
228 %quarta riga
229 alphabet.maiusc = squeeze(angles(4,1,:));
230 alphabet.a = squeeze(angles(4,2,:));
231 alphabet.s = squeeze(angles(4,3,:));
232 alphabet.d = squeeze(angles(4,4,:));
233 alphabet.f = squeeze(angles(4,5,:));
234 alphabet.g = squeeze(angles(4,6,:));
235 alphabet.h = squeeze(angles(4,7,:));
236 alphabet.j = squeeze(angles(4,8,:));
237 alphabet.k = squeeze(angles(4,9,:));
238 alphabet.l = squeeze(angles(4,10,:));
239 alphabet.at = squeeze(angles(4,11,:));
240 alphabet.hashtag = squeeze(angles(4,12,:));
241 alphabet.uAccentata = squeeze(angles(4,13,:));
242 %quinta riga
243 alphabet.shift1 = squeeze(angles(5,1,:));
244 alphabet.greater = squeeze(angles(5,2,:));
245 alphabet.z = squeeze(angles(5,3,:));
246 alphabet.x = squeeze(angles(5,4,:));
247 alphabet.c = squeeze(angles(5,5,:));
248 alphabet.v = squeeze(angles(5,6,:));
249 alphabet.b = squeeze(angles(5,7,:));
250 alphabet.n = squeeze(angles(5,8,:));
251 alphabet.m = squeeze(angles(5,9,:));
252 alphabet.comma = squeeze(angles(5,10,:));
253 alphabet.dot = squeeze(angles(5,11,:));
254 alphabet.dash = squeeze(angles(5,12,:));
255 alphabet.shift2 = squeeze(angles(5,13,:));
```

```
%sesta riga
alphabet.fn = squeeze(angles(6,1,:));
alphabet.ctrl = squeeze(angles(6,2,:));
alphabet.alt1 = squeeze(angles(6,3,:));
alphabet.cmd1 = squeeze(angles(6,4,:));
alphabet.space = squeeze(angles(6,5,:));
alphabet.cmd2 = squeeze(angles(6,6,:));
alphabet.alt2 = squeeze(angles(6,7,:));
alphabet.leftArrow = squeeze(angles(6,8,:));
alphabet.upArrow = squeeze(angles(6,9,:));
alphabet.downArrow = squeeze(angles(6,10,:));
alphabet.rightArrow = squeeze(angles(6,11,:));

%% check dei tasti

for i = 1:6
    for j = 1:14
        if angles(i,j,2) < 0
            fprintf('l''angolo 2: %d %d risulta negativo \n',i,j)
    ;
        end
        if imag(angles(i,j,2))~=0
            fprintf('l''angolo 2: %d %d risulta complesso \n',i,j
    );
        end
        if angles(i,j,3) < 0
            fprintf('l''angolo 3: %d %d risulta negativo \n',i,j)
    ;
        end
        if imag(angles(i,j,3))~=0
            fprintf('l''angolo 3: %d %d risulta complesso \n',i,j
    );
        end
    end
end
```

```
function printAngles(x)
    fprintf('\n');
    for k = 1:3
        kk=k-1;
        fprintf('finalAngle[%d]= %4.2f; \n', kk, x(k));
    end
    fprintf('\n');
end
```

A.2 Arduino

```
//librerie
#include <Servo.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define OLED_RESET     4

//variabili
//OLED
Adafruit_SSD1306 display(128, 64, &Wire, OLED_RESET);
//servo
int pin[] = {9,5,10};
//sizeof mi da il numero di byte occupato. numero di byte
   occupato dall'array diviso numero di byte per un elemento mi
   da il num di elementi
const int NUM_PIN = sizeof(pin)/sizeof(pin[0]);
Servo servo[NUM_PIN];
//variabili varie
double finalAngle[3];
double startingAngle[NUM_PIN]; //angoli da cui partono i giunti
double angle[NUM_PIN]; //angoli che modifico, parte uguale a
   startingAngle
int posMin = 0;
int posMax = 180;


void setup() {
  //Baud Rate per comunicazione seriale
  Serial.begin(115200);
  //Setup schermo OLED
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.setTextColor(WHITE);//Imposta il colore del font
  display.clearDisplay();//cls
  display.fillScreen(BLACK);
  //associo i pin alle variabili servo
  for (int i = 0; i < NUM_PIN; ++i) {
    servo[i].attach(pin[i]);
    startingAngle[i] = servo[i].read();
    Serial.print(startingAngle[i]);
    angle[i] = startingAngle[i];
  }
}
```

```
40
41
42 void loop() {
43   //OLED
44   display.setTextSize(2); //Imposto la dimensione del font
45   display.setCursor(55,10); //Imposto dove vengono mostrate le
      lettere sullo schermo OLED
46
47   if (Serial.available() > 0) {
48     byte ch = Serial.read(); //legge il byte in entrata
49     display.fillScreen(BLACK); //metto l'OLED tutto nero
50     Serial.print("I received: ");
51     Serial.println((char)ch); //alla fine delle lettere legge
     sempre anche un "Line feed" (ascii 10)
52     display.setTextSize(5);
53     display.print((char)ch);
54     //Inizia a mostrare
55     display.display();
56     //movimento del robot
57     writeLetter(ch);
58     delay(1000);
59     moveNeutral();
60     delay(1000);
61   }
62 }
63
64 void moveNeutral(){ //angolo1 = 90, angolo2 = 45, angolo0 = non
     cambiato
65   //secondo servo
66   moveServo(1,90);
67   delay(100);
68   //primo servo
69   moveServo(2,45);
70 }
71
72 void moveServo(int num,double finalAngle){
73   int offet_degrees = 4;
74   boolean done = false;
75   while(!done){
76     if (angle[num] < finalAngle){
77       angle[num]=angle[num]+offet_degrees;
78       if (angle[num] > posMax){ //if di controllo che non si esca
     dai bound inizialente posti
79         angle[num] = posMax;
```

```
80          done = true;
81        }
82        else if(angle[num] >= finalAngle){ //sono arrivata alla
    fine?
83          angle[num] = finalAngle;
84          done = true;
85        }
86        servo[num].write(angle[num]);
87      }
88      else if (angle[num] > finalAngle){
89        angle[num]=angle[num]-offet_degrees;
90        if (angle[num] < posMin){ //if di controllo che non si esca
     dai bound inizialente posti
91          angle[num] = posMin;
92          done=true;
93        }
94        else if(angle[num] <= finalAngle){ //sono arrivata alla
    fine?
95          angle[num] = finalAngle;
96          done = true;
97        }
98        servo[num].write(angle[num]);
99      }
100     else{
101       done = true;
102     }
103     delay(70);
104   }
105 }
106
107
108 void writeLetter(char ch){
109
110   //lettere
111
112   if (ch == 'a'){
113       finalAngle[0]= 122.04;
114       finalAngle[1]= 33.20;
115       finalAngle[2]= 62.89;
116   }
117   else if (ch == 'b'){
118       finalAngle[0]= 89.66;
119       finalAngle[1]= 38.40;
120       finalAngle[2]= 104.57;
```

```
121   }
122   else if (ch == 'c'){
123       finalAngle[0]= 109.67;
124       finalAngle[1]= 50.50;
125       finalAngle[2]= 96.16;
126   }
127   else if (ch == 'd'){
128       finalAngle[0]= 109.29;
129       finalAngle[1]= 44.69;
130       finalAngle[2]= 80.56;
131   }
132   else if (ch == 'e'){
133       finalAngle[0]= 107.59;
134       finalAngle[1]= 34.48;
135       finalAngle[2]= 64.86;
136   }
137   else if (ch == 'f'){
138       finalAngle[0]= 101.45;
139       finalAngle[1]= 48.51;
140       finalAngle[2]= 86.40;
141   }
142   else if (ch == 'g'){
143       finalAngle[0]= 92.76;
144       finalAngle[1]= 50.98;
145       finalAngle[2]= 90.16;
146   }
147   else if (ch == 'h'){
148       finalAngle[0]= 83.52;
149       finalAngle[1]= 52.03;
150       finalAngle[2]= 91.75;
151   }
152   else if (ch == 'i'){
153       finalAngle[0]= 69.40;
154       finalAngle[1]= 41.03;
155       finalAngle[2]= 74.96;
156   }
157   else if (ch == 'j'){
158       finalAngle[0]= 74.18;
159       finalAngle[1]= 51.59;
160       finalAngle[2]= 91.08;
161   }
162   else if (ch == 'k'){
163       finalAngle[0]= 65.22;
164       finalAngle[1]= 49.69;
```

```
165         finalAngle[2]= 88.20;
166     }
167     else if (ch == 'l'){
168         finalAngle[0]= 57.00;
169         finalAngle[1]= 46.41;
170         finalAngle[2]= 83.19;
171     }
172     else if (ch == 'm'){
173         finalAngle[0]= 67.56;
174         finalAngle[1]= 40.50;
175         finalAngle[2]= 103.16;
176     }
177     else if (ch == 'n'){
178         finalAngle[0]= 78.49;
179         finalAngle[1]= 37.53;
180         finalAngle[2]= 105.14;
181     }
182     else if (ch == 'o'){
183         finalAngle[0]= 61.96;
184         finalAngle[1]= 38.26;
185         finalAngle[2]= 70.69;
186     }
187     else if (ch == 'p'){
188         finalAngle[0]= 55.14;
189         finalAngle[1]= 34.13;
190         finalAngle[2]= 64.33;
191     }
192     else if (ch == 'q'){
193         finalAngle[0]= 119.16;
194         finalAngle[1]= 21.90;
195         finalAngle[2]= 45.50;
196     }
197     else if (ch == 'r'){
198         finalAngle[0]= 100.72;
199         finalAngle[1]= 38.51;
200         finalAngle[2]= 71.07;
201     }
202     else if (ch == 's'){
203         finalAngle[0]= 116.15;
204         finalAngle[1]= 39.60;
205         finalAngle[2]= 72.75;
206     }
207     else if (ch == 't'){
208         finalAngle[0]= 93.24;
```

```
209       finalAngle[1]= 41.18;
210       finalAngle[2]= 75.18;
211   }
212   else if (ch == 'u'){
213       finalAngle[0]= 77.28;
214       finalAngle[1]= 42.46;
215       finalAngle[2]= 77.14;
216   }
217   else if (ch == 'v'){
218       finalAngle[0]= 100.24;
219       finalAngle[1]= 42.95;
220       finalAngle[2]= 101.50;
221   }
222   else if (ch == 'w'){
223       finalAngle[0]= 113.73;
224       finalAngle[1]= 29.03;
225       finalAngle[2]= 56.46;
226   }
227   else if (ch == 'x'){
228       finalAngle[0]= 117.69;
229       finalAngle[1]= 50.11;
230       finalAngle[2]= 88.83;
231   }
232   else if (ch == 'y'){
233       finalAngle[0]= 85.33;
234       finalAngle[1]= 42.50;
235       finalAngle[2]= 77.21;
236   }
237   else if (ch == 'z'){
238       finalAngle[0]= 124.37;
239       finalAngle[1]= 44.12;
240       finalAngle[2]= 79.69;
241   }
242
243   //numeri
244
245   else if (ch == '1'){
246     finalAngle[0]= 117.91;
247     finalAngle[1]= 3.55;
248     finalAngle[2]= 17.59;
249   }
250   else if (ch == '2'){
251     finalAngle[0]= 113.04;
252     finalAngle[1]= 14.05;
```

```
253     finalAngle[2]= 33.49;
254   }
255   else if (ch == '3'){
256     finalAngle[0]= 107.61;
257     finalAngle[1]= 21.14;
258     finalAngle[2]= 44.34;
259   }
260   else if (ch == '4'){
261     finalAngle[0]= 101.63;
262     finalAngle[1]= 26.20;
263     finalAngle[2]= 52.10;
264   }
265   else if (ch == '5'){
266     finalAngle[0]= 95.17;
267     finalAngle[1]= 29.65;
268     finalAngle[2]= 57.41;
269   }
270   else if (ch == '6'){
271     finalAngle[0]= 88.33;
272     finalAngle[1]= 31.66;
273     finalAngle[2]= 60.51;
274   }
275   else if (ch == '7'){
276     finalAngle[0]= 81.29;
277     finalAngle[1]= 32.28;
278     finalAngle[2]= 61.48;
279   }
280   else if (ch == '8'){
281     finalAngle[0]= 74.26;
282     finalAngle[1]= 31.54;
283     finalAngle[2]= 60.33;
284   }
285   else if (ch == '9'){
286     finalAngle[0]= 67.45;
287     finalAngle[1]= 32.72;
288     finalAngle[2]= 59.40;
289   }
290   else if (ch == '0'){
291     finalAngle[0]= 67.45;
292     finalAngle[1]= 29.40;
293     finalAngle[2]= 57.04;
294   }
295
296   //caratteri speciali
```

```
297
298   else if (ch == ' '){
299     finalAngle[0]= 92.64;
300     finalAngle[1]= 19.71;
301     finalAngle[2]= 116.08;
302   }
303   else if (ch == 92){ // backslash
304     finalAngle[0]= 90; //vado in posizione neutra
305     finalAngle[1]= 90;
306     finalAngle[2]= 45;
307     Serial.println("can't reach \\");
308   }
309   else if (ch == 39){ // '
310     finalAngle[0]= 55.09;
311     finalAngle[1]= 20.61;
312     finalAngle[2]= 43.52;
313   }
314   else if (ch == '+'){
315     finalAngle[0]= 43.68;
316     finalAngle[1]= 21.30;
317     finalAngle[2]= 44.58;
318   }
319   else if (ch == '<'){
320     finalAngle[0]= 129.86;
321     finalAngle[1]= 36.97;
322     finalAngle[2]= 68.71;
323   }
324   else if (ch == ','){
325     finalAngle[0]= 57.56;
326     finalAngle[1]= 46.82;
327     finalAngle[2]= 98.80;
328   }
329   else if (ch == '.'){
330     finalAngle[0]= 48.91;
331     finalAngle[1]= 52.41;
332     finalAngle[2]= 92.33;
333   }
334   else if (ch == '-'){
335     finalAngle[0]= 41.65;
336     finalAngle[1]= 46.92;
337     finalAngle[2]= 83.98;
338   }
339   else{ //nel caso sia stato premuto un tasto non adatto o per il
        valore "newLine a fine frase" metto in posizione neutrale
```

```
340     finalAngle[0]= 90;
341     finalAngle[1]= 90;
342     finalAngle[2]= 45;
343     Serial.print("il tasto ");
344     Serial.print((char)ch);
345     Serial.println(" non risulta raggiungibile");
346   }
347   int i = 0;
348   moveServo(i, finalAngle[i]);
349   i = 2;
350   moveServo(i, finalAngle[i]);
351   i = 1;
352   moveServo(i, finalAngle[i]);
353 }
```

Bibliografia

[1] About Da Vinci Systems, Da Vinci Surgical System — Robotic Technology, Copyright: © 2022 Intuitive Surgical, davincisurgery.com, https://www.davincisurgery.com/da-vinci-systems/about-da-vinci-systems, ultima consultazione: 30/8/2022

[2] Sun Yuan and Peng Yu Can't Help Myself, Can't Help Myself — The Guggenheim Museums and Foundation, Copyright: © 2022 THE SOLOMON R. GUGGENHEIM FOUNDATION, guggenheim.org, https://www.guggenheim.org/artwork/34812, ultima consultazione: 30/8/2022

[3] Michael Newland, Studio, sviluppo e realizzazione di un mini-robot dimostrativo, 2017-2018, M00006 – Progetto di diploma, Dipartimento tecnologie innovative, Scuola universitaria professionale della Svizzera italiana, https://tesi.supsi.ch/2499/1/MichaelNelwand_Studio%2C%20sviluppo%20e%20realizzazione%20di%20un%20mini-robot%20dimostrativo_Rapporto.pdf

[4] Moran ME. Evolution of robotic arms. J Robot Surg. 2007;1(2):103-11. doi: 10.1007/s11701-006-0002-x. Epub 2007 May 1. PMID: 25484945; PMCID: PMC4247431.

[5] EMAS bionic arm, Copyright: © National Museums Scotland, Scottish Charity SC011130, nms.ac.uk, https://www.nms.ac.uk/explore-our-collections/collection-search-results/prosthetic-arm/691231, ultima consultazione: 30/8/2022.

[6] Adeept 5-DOF Robotic Arm Kit Compatible with Arduino IDE, Programmable DIY Coding STEM Educational 5 Axis Robot Arm with OLED Display Processing Code, Copyright: © 2014-2022 Adeept.com, adeept.com, https://www.adeept.com/adeept-arduino-compatible-diy-5-dof-robotic-arm-kit-for-arduino-uno-r3-steam-robot-arm-kit-with-arduino-and-processing-code_p0118.html, ultima consultazione: 30/8/2022.

[7] Domenico Tessari, *La cinematica applicata alle macchine ad uso delle scuole d'applicazione per gli ingegneri degli ingegneri, e costruttori meccanici*, luogo: Ermanno Loescher, 1890, https://quod.lib.umich.edu/u/umhistmath/ACA1491.0001.001/1?rgn=full+text;view=pdf

[8] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, Giuseppe Oriolo, *Robotics: Modelling, Planning and Control*, Springer, 2009, http://people.disim.univaq.it/~costanzo.manes/EDU_stuff/Robotics_Modelling,%20Planning%20and%20Control_Sciavicco_extract.pdf.

[9] Link to Adeept Code and Tutorials of the ADA031-V4.0 robotic arm, https://www.adeept.com/learn/detail-64.html.

[10] Michela di Rocchi, Cinzia Ferrari, *I Mech: English for Mechanical Technology*, Hoepli, 2019.

[11] MG90S datasheet, Unclassified Manufacturer, https://pdf1.alldatasheet.com/datasheet-pdf/view/1132104/ETC2/MG90S.html.

[12] CH340 datasheet, WCH, upload time: 2022-05-24, http://www.wch-ic.com/downloads/CH340DS1_PDF.html

[13] GS2678 datasheet, GSM, https://datasheetspdf.com/pdf-file/699027/SHENZHENGSMELECTAONIC/GS2678/1

[14] ATmega328P datasheet, Atmel, 2015, http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf.

[15] Satoru Goto, *ROBOT ARMS*, InTech, 2011, https://dl.icdst.org/pdfs/files3/36ee3042bc5837b2c458b15d5732ab00.pdf.