Saggiorato Andrea
A.S.2023/2024

# PROGETTO DI ROBOTICA

## INDICE

Andrea Saggiorato A.S. 2022/2023

**Introduzione**

**Componenti**

- SSC-32U
- Arduino-Uno
- Servomotore

**Schema a blocchi**

**Realizzazione**

- Assemblaggio del Braccio Robot
- Taratura dei servomotori
- Programmi
- Immagini

**Calcoli matematici**

**Conclusioni**

**Bibliografia e sitografia**

## INTRODUZIONE

Fin da piccolo sono rimasto affascinato dai manipolatori, capaci di svolgere in modo autonomo gran parte delle azioni umane. Con gli anni ho compreso che oltre alla bellezza estetica, la robotica può anche essere una componente di notevole utilità all'interno delle industrie. Il mio progetto consiste nel creare un braccio robotico che vada ad automatizzare i processi di costruzione aziendale, e che

evitando l'impiego diretto di un essere umano garantisca contemporaneamente una maggiore sicurezza. Il progetto che ho idealizzato mi permetterà di effettuare delle costruzioni su un piano di lavoro, andando a prendere il materiale necessari negli appositi bancali. Per costruire, il manipolatore andrà a posizionarsi sopra al blocco da prendere, per poi chiudere l'end effector e andare a posizionare sul banco da lavoro il pezzo preso, nelle rispettive x e y, ripetendo il procedimento finché la costruzione non sarà completata. Nel caso ci sia la necessita di ripulire il piano di lavoro è possibile far effettuare il reset del piano di lavoro tramite il comando di clean. Questo progetto mi permette di mettere a frutto alcune delle competenze pratiche e teoriche maturate durante il triennio. Inizialmente il progetto posizionerà i blocchi dal punto di partenza in ordine casuale sugli scaffali. A questo punto potrò andare a posizionare a piacimento i blocchi sul piano di lavoro prendendoli dagli scaffali. Infine, andrò a comporre una scritta in modo semiautomatico. Questo è un banale esempio che ci aiuta però a comprendere come il progetto possa funzionare sia in modo automatico che manuale.

## COMPONENTI

Materiali/Componenti:

- Braccio robotico a 6 assi
- 6 servomotori
- Banco da lavoro
- Bancali di plastica
- Asse di legno
- Viti

- Trapano
- Contenitore per schede in plastica
- Cavi
- Ralla metallica
- Rialzo di plastica
- Interruttore alimentazione
- Rondelle
- Chiave inglese
- Pinza
- Dadi
- Bulloni
- Cacciavite
- Forbici
- Multimetro digitale
- Calibro
- Interrutore
- Riga da 50cm
- Goniometro
- Capicorda
- Scotch
- Guaina spiralata nera
- Fascette
- Stampante 3D Creality k1
- Filamento nero, rosso, rosa, bianco

Software/programmi pc

- Microsoft Excel
- Visual Studio
- Creality print

Schede elettroniche

- Arduino Uno
- SSC-32U

Alimentazione

- Alimentatore da 220V-6V
- Presa 10A con cavetteria mista

## SSC-32U

La scheda SSC-32U è un dispositivo, progettato per controllare fino a 32 servomotori, numerati da 0 a 31. Per collegare ogni servomotore, bisogna inserire i pin nei rispettivi slot sulla scheda. I pin sono disposti in un ordine specifico: verso l'esterno della scheda c'è la massa (GND), al centro si trova il pin dell'alimentazione, alimentato dai connettori VS1 o VS2, e verso l'interno della scheda si trova il pin per il segnale (PULSE). L'alimentazione della SSC-32U è suddivisa in due parti: una per il funzionamento della parte logica (VL) e l'altra per i servomotori. Questo significa che i servomotori numerati da 0 a 15 sono alimentati tramite il connettore VS1, mentre quelli numerati da 16 a 31 ricevono l'alimentazione dal connettore VS2. Questa separazione è fondamentale per garantire che i servomotori ricevano la potenza necessaria senza compromettere la stabilità della scheda stessa. Per collegare la SSC-32U al computer e programmarla, si utilizza un connettore USB, inviando delle stringhe di comando ben definite che seguono il formato #<ch>P<pw>S<spd>, dove # indica l'inizio del comando. Il parametro <ch> rappresenta il numero del servomotore a cui si desidera inviare l'impulso. Il parametro P<pw> specifica la durata dell'impulso, variabile tra 500 e 2500 microsecondi, permettendo così di posizionare il servomotore in qualsiasi angolo tra 0° e 180°. Infine, il parametro S<spd> definisce la velocità di esecuzione del movimento, espressa in microsecondi per

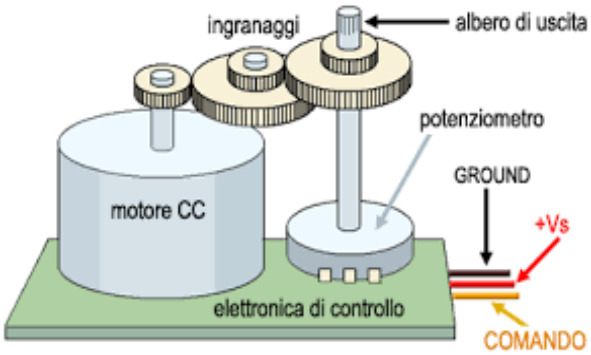
secondo. Grazie a questa struttura di comando, è possibile controllare con precisione i movimenti dei servomotori. Questo è particolarmente utile in applicazioni che richiedono un alto grado di coordinazione, come i bracci robotici che devono muoversi in modo sincronizzato per eseguire compiti complessi. un braccio robotico può utilizzare la SSC-32U per coordinare i movimenti di ogni giunto, permettendo di assemblare componenti delicati o manipolare oggetti con precisione. Nel mio progetto ho usato gli slot più bassi a partire dal pin 0 per far in modo di collegare simultaneamente tutti i servomotori. È importante premere il pulsante situato al centro della schedina finché i led non lampeggiano per avviare correttamente il suo funzionamento.

## ARDUINO

Arduino è una piattaforma elettronica programmabile, basata sul microcontrollore ATmega328P, che rende semplice e accessibile la creazione di progetti interattivi. La scheda è progettata con varie sezioni, ognuna con una funzione specifica: l'alimentazione, i pin analogici e i pin digitali. I pin analogici servono come ingressi per ricevere segnali variabili provenienti da diversi tipi di sensori, come trimmer, fotoresistenze e sensori a ultrasuoni. Questi segnali vengono convertiti in valori digitali grazie al convertitore analogico-digitale (A/D) di Arduino, che ha una risoluzione di 10 bit. In pratica, questo significa che il segnale analogico può essere rappresentato da 1024 livelli distinti, permettendo ad Arduino di leggere tensioni comprese tra 0 e 5 Volt con un'ottima precisione. I pin digitali, invece, possono essere utilizzati sia come ingressi sia come uscite, consentendo di ricevere o inviare segnali logici. Alcuni di questi pin digitali hanno la funzione di modulazione di larghezza di impulso (PWM), che permette di inviare o ricevere valori compresi tra 0 e 255. Questo rende i pin digitali molto versatili, permettendo loro di comportarsi in modo simile ai pin analogici in determinate applicazioni. La programmazione di Arduino Uno avviene tramite il software Arduino IDE, che consente di scrivere e caricare codice sulla scheda. Per caricare un programma, basta collegare Arduino al PC tramite un cavo USB e trasferire il codice. Inoltre, Arduino può funzionare in modo autonomo senza essere collegato al computer. Accanto al connettore USB, infatti, è presente un connettore di alimentazione che permette di alimentare la scheda utilizzando una fonte di alimentazione esterna, rendendo possibile l'utilizzo di Arduino in applicazioni portatili o remote. Nel mio progetto

attualmente non è ancora stato utilizzato, anche se ho inserito uno slot accanto all'SSC-32U, ma in futuro lo utilizzerò per collegare il braccio un'applicazione del cellulare creata appositamente e connessa tramite l'utilizzo di un modulo Bluetooth chiamato HC-05.

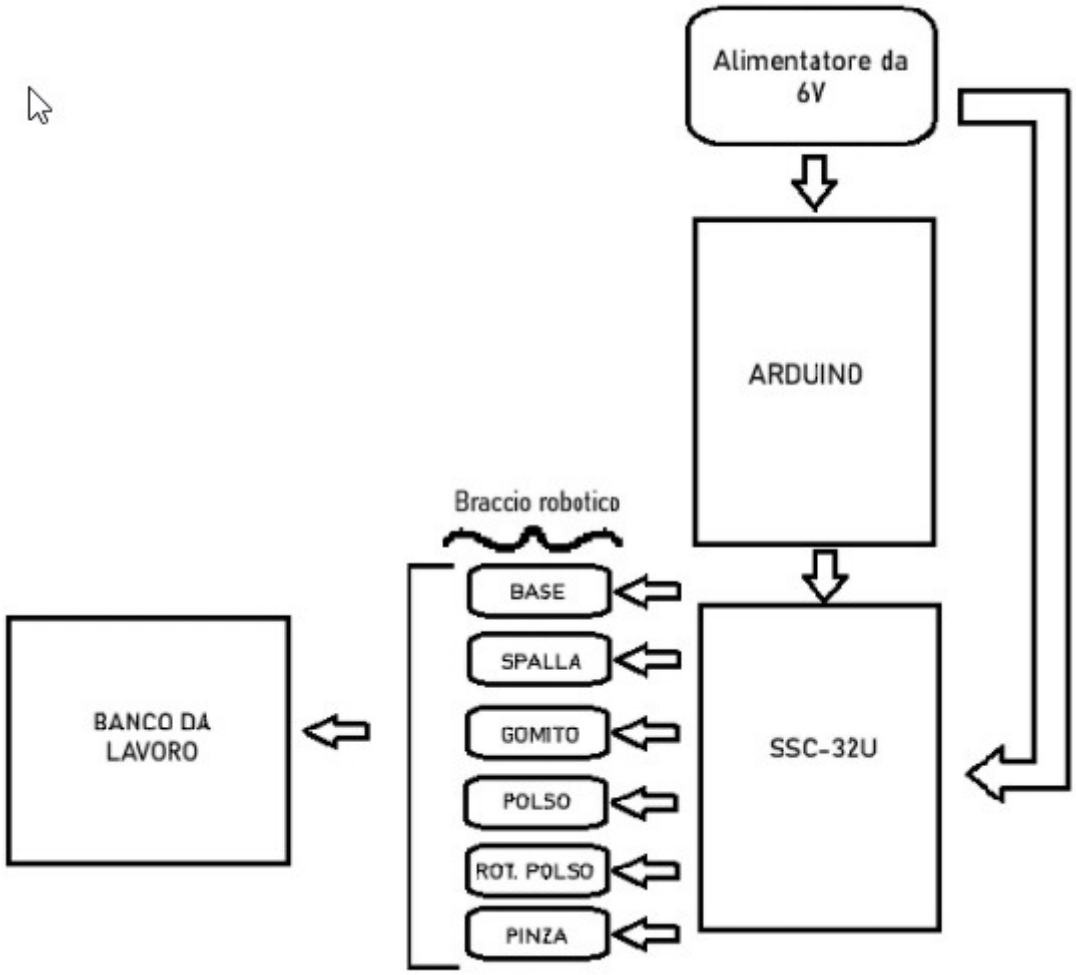
## SERVOMOTORI


I servomotori, dispositivi essenziali nell'ambito dell'automazione e della robotica sono caratterizzati da un involucro compatto che ospita un albero motore capace di compiere rotazioni precise all'interno di un angolo compreso tra 0° e 180°. Questo movimento è reso possibile da un motore a corrente continua, il cui output è opportunamente modulato da un sistema di demoltiplica che amplifica la coppia motrice e garantisce la stabilità del movimento. Al centro di ogni servomotore risiede un circuito di controllo, una componente elettronica intelligente che supervisiona costantemente l'angolo di rotazione dell'albero motore. Questo avviene attraverso l'uso di un potenziometro resistivo integrato, che fornisce al circuito di controllo un feedback in tempo reale sulla posizione del perno. Quando il perno raggiunge la posizione desiderata, il circuito di

controllo interviene regolando la tensione del motore per bloccare il movimento e garantire la stabilità della posizione. Per impartire comandi al servomotore, è necessario inviare un segnale digitale al circuito di controllo. Questo segnale, noto come segnale PWM (modulazione di larghezza di impulso), Ha una durata variabile tra 500 $\mu$s e 2500 $\mu$s, rappresentando l'ampiezza dell'angolo di rotazione desiderato. Modificando la durata di questo segnale, è possibile regolare con precisione la posizione del servomotore e controllare il suo movimento con grande accuratezza. Nonostante la loro reputazione di motori a bassa precisione, i servomotori sono ancora ampiamente utilizzati in una varietà di applicazioni, grazie alla loro affidabilità e semplicità di utilizzo. Tuttavia, in contesti che richiedono un controllo più preciso e dettagliato, come nel caso di sistemi robotici avanzati o applicazioni di automazione industriale, è preferibile utilizzare motori passo-passo. Questi ultimi offrono una maggiore precisione nel posizionamento e consentono di eseguire movimenti discreti e controllati con estrema precisione, rendendoli ideali per applicazioni che richiedono un controllo accurato del movimento. Nel mio progetto sono stati implementati per rappresentare gli assi di rotazione del braccio robotico, più dettagliatamente: la base, la spalla, il gomito, il polso, la pinza e la rotazione del polso (garantendo una maggiore libertà d'azione all'end-effector).

## SCHEMA A BLOCCHI




Riportato lo schema a blocchi nel quale viene rappresentato il collegamento generale tra i vari componenti del progetto.

## REALIZZAZIONE

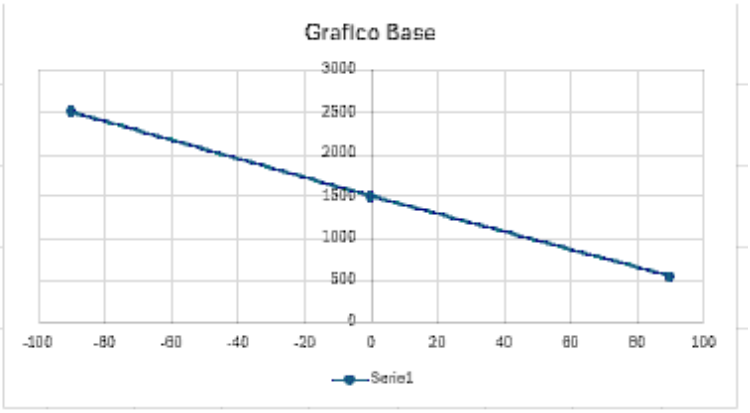
### ASSEMBLAGGIO

Inizialmente ho assemblato il braccio robotico della KIMISS con un pacchetto comprendente viti, bulloni, pezzi in alluminio ecc... Tutto alimentato con 6V e installato su un piano di legno.

### TARATURA

Ho tarato ciascun servomotore per poter trovare gli impulsi necessari a ottenere gli angoli desiderati (solitamente 0°,45°90°180°), per fare ciò ho utilizzato dei grafici Excel.

| 0 | BASE | |
|---|---|---|
| POSIZIONE | GRADI | POTENZA |
| SX | -90 | 2500 |
| RIPOSO | 0 | 1500 |
| DX | 90 | 550 |




Nell'immagine si può vedere il grafico dove in ascissa abbiamo gli angoli (espressi in gradi) e in ordinata abbiamo gli impulsi (espressi in $\mu s$).

Il risultato finale è una retta per ogni servomotore. A questo punto Excel è in grado di calcolare l'equazione della retta.

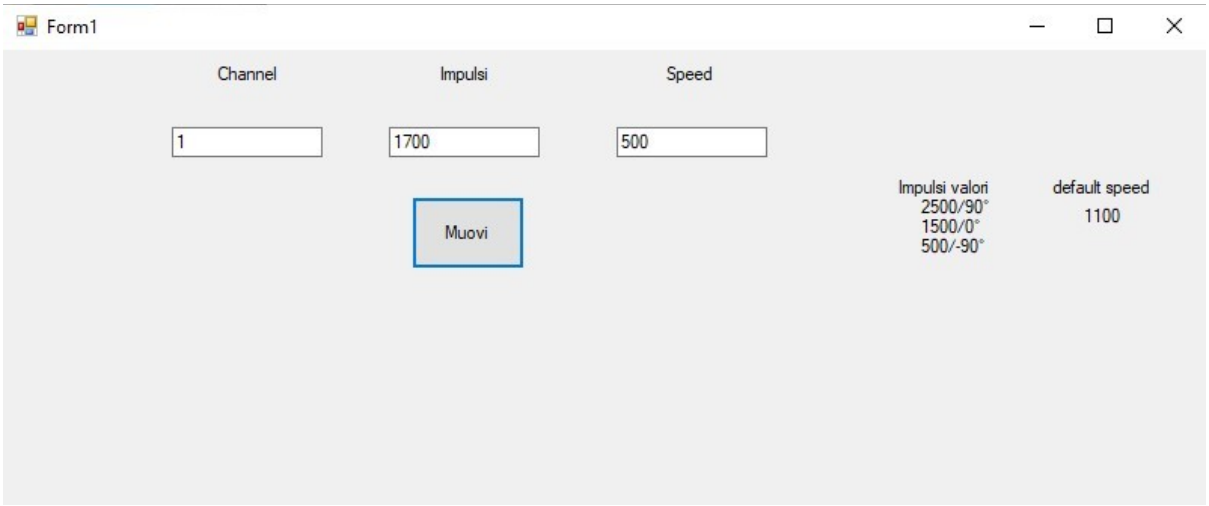
Impulsi = angolo * m + q

m è il coefficiente angolare della retta e q è l'ordinata di attraversamento dell'ordinata. Il valore di q è semplice da trovare, mentre per ricavare m si utilizza il calcolo del rapporto tra la differenza degli estremi in ordinata e la differenza tra gli estremi in ascissa:

m =$(P2Y-P1Y)$ /$(P2X-P1)$

PROGRAMMI

Successivamente ho sviluppato il programma di pilotaggio con C#, con un'interfaccia dotata di:




3 label: utilizzabile dal programmatore per dare maggiori informazioni all'utente che utilizza il programma.

3 textbox: utilizzate per: specificare il motore utilizzato definire l'impulso da dare al servomotore per fargli compiere un'azione e la velocità con la quale il servomotore dovrà compiere l'azione.

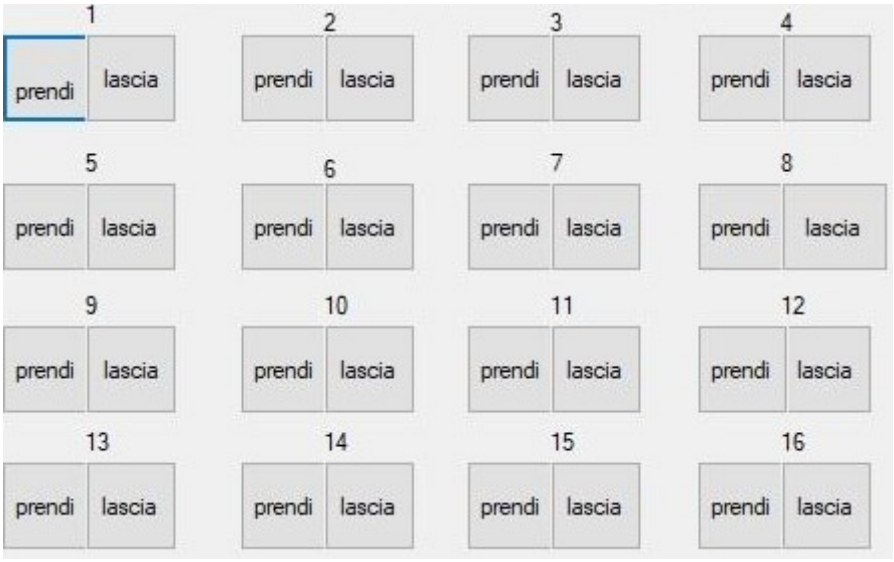
1 button: sarà il comando MUOVI che darà l'invio nella stringa dei dati inseriti precedentemente dall'utente e muoverà il motore.

[Codice]

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Manipolatore00
{
public partial class Form1 : Form
{
```

```
public Form1()

{

InitializeComponent();

}

private void cmdMuovi_Click(object sender, EventArgs e)

{

string comando;

serialPort1.Open();

//comando = "#" + txtChannel1.Text + "P" + txtImpulsi1.Text + "S" + txtSpeed1.Text + "#" +
txtChannel.Text + "P" + txtImpulsi.Text + "S" + txtSpeed.Text + "#" + txtChannel2.Text + "P" +
txtImpulsi2.Text + "S" + txtSpeed2.Text + "#" + txtChannel3.Text + "P" + txtImpulsi3.Text +
"S" + txtSpeed3.Text + "#" + txtChannel4.Text + "P" + txtImpulsi4.Text + "S" +
txtSpeed4.Text + "#" + txtChannel5.Text + "P" + txtImpulsi5.Text + "S" + txtSpeed5.Text +
Environment.NewLine;

comando = "#" + txtChannel1.Text + "P" + txtImpulsi1.Text + "S" + txtSpeed1.Text +
Environment.NewLine;

          serialPort1.Write(comando);

          serialPort1.Close();

      }

   }

}
```

A questo punto ho ricavato i valori di m e q dai grafici precedentemente creati con Excel e creato un altro programma per implementare l'uso dei buttons.




In questo modo sono andato ad ottimizzare il funzionamento del manipolatore riducendo al minimo le operazioni di trimmering, ossia la taratura per arrivare il prima possibile alla posizione desiderata. Una volta premuti i pulsanti, l'end effector andrà a posizionarsi precisamente sopra il target. A questo punto, partirà un comando che fa chiuderà la pinza, preleverà l'oggetto desiderato e lo porterà successivamente in una posizione da noi scelta. Il codice di questo programma verrà mostrato nella parte del codice finale dato che il programma in questione è stato integrato e ottimizzato nel codice che ho usato per usare il mio progetto.

PROGRAMMA FINALE

Per assicurarmi una corretta esecuzione ho dovuto misurare la distanza dei giunti, la distanza tra la base e la griglia di lavoro, l'altezza dell'end effector e infine ho inserito tutti i valori nel codice scritto su Visual Studio.

Una volta ricavati tutti i valori ho potuto scrivere il codice finale:

[Codice]

```
using System;

using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Threading;
using System.Security.AccessControl;
using System.Runtime.InteropServices;


//y(-4/15)
namespace Manipolatore00
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        //DICHIARO LE VARIABILI GLOBALI
        //valori identificazione singolo comando
        string cmdbase;
        string cmdspalla;
        string cmdgomito;
        string cmdpolso;
        string cmdpinza;
        string cmdrotazione;
        //valori di q per ogni motore(ricavato da grafici excel)
```

```
double qbase = 1500;//1500
double qspalla = 900;//550
double qgomito = 870;//500
double qpolso = 500;//500
//valori di m per ogni motore(ricavato da grafici excel)
double mbase = -10.8333;//-10.8333
double mspalla = 7.222222;//11.6666
double mgomito = 7.4;// 10.5555
double mpolso = 11.2;//11.2
//valore della distanza tra scacchiera e centro della base
double distbase = 8.5f;//8.4
// valore lunghezza dei link
double L1 = 21.0f;  //21.0
double L2 = 16.50f; // 16.50
// valore della distanza del targhet al centro della base
double d;
//variabili cordinate
double x;
double y;

//MOVIMENTO TASTI*********************************************
private void tasti()
{
    serialPort1.Open();
    //posizione_riposo();
    double tgfi;
    double firad;
    double figradi;
    double cosalfa;
```

```
        double arcocosalfa;
        double alfagradi;
        double cosbeta;
        double arcocosbeta;
        double betagradi;
        double cosgamma;
        double arcocosgamma;
        double gammagradi;
        double impalfa;
        double impgamma;
        double impbeta;
        double impfi;
        //*CALCOLO DELLA POSIIONE RISPETTO AL TARGET
        tgfi = x / (y + distbase);
        firad = Math.Atan(tgfi);
        figradi = ((180 * firad) / 3.14); //conversione in gradi
        impfi = mbase * figradi + qbase;
        //stringa di comando per la base
        cmdbase = "#" + "0" + "P" + Math.Round(impfi) + "S600" +        Environment.NewLine;
        //ricavare d con il teorema di Pitagora
        d = Math.Sqrt((x * x) + ((y + distbase) * (y + distbase)));
        //CALCOLO L ANGOLO ALFA PER RAGGIUNGERE I TARGET(polso)
        cosalfa = (-(L2 * L2) + (d * d) + (L1 * L1)) / (2 * d * L1);
        //*CALCOLO IL COSENO DI ALFA
        arcocosalfa = Math.Acos(cosalfa);// calcolo in radianti del cos alfa
        alfagradi = 90 + (((arcocosalfa * 180) / 3.14));//trasforma radianti in gradi
        //90 è l angolo da sommare per far piegare il polso e renderlo perpendicolare alla
scacchiera
        impalfa = mpolso * alfagradi + qpolso;
```

```
    cmdpolso = "#" + "3" + "P" + Math.Round(impalfa) + "S600" + Environment.NewLine;
    //CALCOLO L ANGOLO BETA PER RAGGIUNGERE I TARGET(gomito)
    cosbeta = (-(d * d) + (L1 * L1) + (L2 * L2)) / (2 * L1 * L2);
    arcocosbeta = Math.Acos(cosbeta);
    betagradi = (((arcocosbeta * 180) / 3.14));
    impbeta = mgomito * betagradi + qgomito;
    cmdgomito = "#" + "2" + "P" + Math.Round(impbeta) + "S300" + Environment.NewLine;
    //CALCOLO L ANGOLO GAMMA PER RAGGIUNGERE I TARGET(spalla)
    cosgamma = (-(L1 * L1) + (d * d) + (L2 * L2)) / (2 * d * L2);
    arcocosgamma = Math.Acos(cosgamma);
    gammagradi = (((arcocosgamma * 180) / 3.14));
    impgamma = mspalla * gammagradi + qspalla;
    cmdspalla = "#" + "1" + "P" + Math.Round(impgamma) + "S300" + Environment.NewLine;
     //IMPOSTAZIONE MOVIMENTI
    serialPort1.Write(cmdbase);
    Thread.Sleep(2000);
    serialPort1.Write(cmdgomito + cmdpolso + cmdspalla);
    Thread.Sleep(1000);
    serialPort1.Close();
}

//POSIZIONE DI RIPOSO
private void posizione_riposo2()
{
    serialPort1.Open();
    int speed = 500;
    cmdbase = "#" + "0" + "P" + "1500" + "S" + speed + Environment.NewLine;
    cmdspalla = "#" + "1" + "P" + "1550" + "S" + speed + Environment.NewLine;
    cmdgomito = "#" + "2" + "P" + "1570" + "S" + speed + Environment.NewLine;
```

```
        cmdpolso = "#" + "3" + "P" + "1400" + "S" + speed + Environment.NewLine;
        // cmdpinza = "#" + "5" + "P" + "750" + "S" + speed + Environment.NewLine;
        cmdrotazione = "#" + "4" + "P" + "1400" + "S" + speed + Environment.NewLine;
        serialPort1.Write(cmdspalla + cmdpolso + cmdgomito);
        Thread.Sleep(1000);
        serialPort1.Write(cmdbase + cmdrotazione);
        Thread.Sleep(1000);
        serialPort1.Close();
    }
    // prova ritorno spalla indietro
    private void posizione_ripososp()
    {
        serialPort1.Open();
        int speed = 500;
        cmdbase = "#" + "0" + "P" + "1500" + "S" + speed + Environment.NewLine;
        cmdspalla = "#" + "1" + "P" + "1750" + "S" + speed + Environment.NewLine;
        cmdgomito = "#" + "2" + "P" + "1570" + "S" + speed + Environment.NewLine;
        cmdpolso = "#" + "3" + "P" + "1000" + "S" + "1500" + Environment.NewLine;
        cmdrotazione = "#" + "4" + "P" + "1400" + "S" + speed + Environment.NewLine;
        serialPort1.Write(cmdspalla + cmdpolso + cmdgomito);
        Thread.Sleep(800);
        cmdspalla = "#" + "1" + "P" + "1550" + "S" + speed + Environment.NewLine;
        cmdpolso = "#" + "3" + "P" + "1400" + "S" + speed + Environment.NewLine;
        serialPort1.Write(cmdspalla+ cmdpolso);
        Thread.Sleep(1000);
        serialPort1.Write(cmdbase + cmdrotazione);
        serialPort1.Close();
    }
    private void posizione_riposo()
```

```
{
    serialPort1.Open();
    int speed = 500;
    cmdbase = "#" + "0" + "P" + "1500" + "S" + speed + Environment.NewLine;
    cmdspalla = "#" + "1" + "P" + "1550" + "S" + speed + Environment.NewLine;
    cmdgomito = "#" + "2" + "P" + "1570" + "S" + speed + Environment.NewLine;
    cmdpolso = "#" + "3" + "P" + "1400" + "S" + speed + Environment.NewLine;
    cmdrotazione = "#" + "4" + "P" + "1400" + "S" + speed + Environment.NewLine;
    serialPort1.Write( cmdspalla + cmdrotazione);
    Thread.Sleep(400);
    serialPort1.Write(cmdpolso+ cmdgomito);
    Thread.Sleep(1000);
    serialPort1.Write(cmdbase );
    serialPort1.Close();
}
private void posizione_riposo3()
{
    serialPort1.Open();
    int speed = 400;
   cmdspalla = "#" + "1" + "P" + "1550" + "S" + speed + Environment.NewLine;
   cmdgomito = "#" + "2" + "P" + "1570" + "S" + speed + Environment.NewLine;
   cmdpolso = "#" + "3" + "P" + "1400" + "S" + speed + Environment.NewLine;
   cmdpinza = "#" + "5" + "P" + "1450" + "S" + speed + Environment.NewLine;
   cmdrotazione = "#" + "4" + "P" + "1400" + "S" + speed + Environment.NewLine;
   serialPort1.Write(cmdspalla );
   serialPort1.Write(cmdpolso + cmdgomito);
    Thread.Sleep(1000);
    serialPort1.Write( cmdrotazione );
    Thread.Sleep(1000);
```

```
        serialPort1.Close();
    }
    //riposo intermedio
    private void posizione_intermedio()
    {
        serialPort1.Open();

        cmdspalla = "#" + "1" + "P" + "1300" + "S" + "700" + Environment.NewLine;
        cmdgomito = "#" + "2" + "P" + "1705" + "S" + "600" + Environment.NewLine;
        cmdpolso = "#" + "3" + "P" + "2000" + "S" + "600" + Environment.NewLine;
        serialPort1.Write(cmdspalla);
       Thread.Sleep(200);

        serialPort1.Write(cmdpolso + cmdgomito);
        cmdgomito = "#" + "2" + "P" + "1700" + "S" + "600" + Environment.NewLine;
        cmdpolso = "#" + "3" + "P" + "1800" + "S" + "600" + Environment.NewLine;
        serialPort1.Write(cmdspalla);
        serialPort1.Write(cmdpolso);
        serialPort1.Close();
    }
    private void cmdRiposo_Click(object sender, EventArgs e)
    {
        posizione_riposo2();
    }
    private void posizione_riposo1()
    {
        serialPort1.Open();
        int speed = 400;
        cmdbase = "#" + "0" + "P" + "1500" + "S" + speed + Environment.NewLine;
```

```
        cmdspalla = "#" + "1" + "P" + "1550" + "S" + speed + Environment.NewLine;
        cmdgomito = "#" + "2" + "P" + "1570" + "S" + speed + Environment.NewLine;
        cmdpolso = "#" + "3" + "P" + "1400" + "S" + speed + Environment.NewLine;
        cmdpinza = "#" + "5" + "P" + "1450" + "S" + speed + Environment.NewLine;
        cmdrotazione = "#" + "4" + "P" + "1400" + "S" + speed + Environment.NewLine;
        serialPort1.Write(cmdspalla);
        serialPort1.Write(cmdpolso + cmdgomito);
        Thread.Sleep(1000);
        serialPort1.Write(cmdrotazione + cmdbase);
        serialPort1.Close();
    }
    //  POSIZIONE BANCALI
    // posizione bancale 3
    private void POSBAN1()
    {
        serialPort1.Open();
        cmdbase = "#" + "0" + "P" + "2100" + "S" + "600" + Environment.NewLine;
        cmdspalla = "#" + "1" + "P" + "1060" + "S" + "200" + Environment.NewLine;
        cmdgomito = "#" + "2" + "P" + "1965" + "S" + "300" + Environment.NewLine;
        cmdpolso = "#" + "3" + "P" + "1600" + "S" +  "400" + Environment.NewLine;
        serialPort1.Write(cmdbase + cmdspalla + cmdgomito );
        Thread.Sleep(1000);
        serialPort1.Write(cmdpolso );
        serialPort1.Close();
    }
    // posizione bancale 2
    private void POSBAN2()
    {
        serialPort1.Open();
```

```
    cmdbase = "#" + "0" + "P" + "1860" + "S" + "600" + Environment.NewLine;
    cmdspalla = "#" + "1" + "P" + "1060" + "S" + "200" + Environment.NewLine;
    cmdgomito = "#" + "2" + "P" + "1980" + "S" + "300" + Environment.NewLine;
    cmdpolso = "#" + "3" + "P" + "1600" + "S" + "400" + Environment.NewLine;
    serialPort1.Write(cmdbase + cmdspalla + cmdgomito  );
    Thread.Sleep(1000);
    serialPort1.Write(cmdpolso);
    serialPort1.Close();

}
// posizione bancale 3
private void POSBAN3()
{
    serialPort1.Open();
    cmdbase = "#" + "0" + "P" + "1600" + "S" + "600" + Environment.NewLine;
    cmdspalla = "#" + "1" + "P" + "1060" + "S" + "200" + Environment.NewLine;
    cmdgomito = "#" + "2" + "P" + "1980" + "S" + "300" + Environment.NewLine;
    cmdpolso = "#" + "3" + "P" + "1600" + "S" + "400" + Environment.NewLine;
    serialPort1.Write(cmdbase + cmdspalla + cmdgomito );
    Thread.Sleep(1000);
    serialPort1.Write(cmdpolso);
    serialPort1.Close();
}
// posizione bancale 4
private void POSBAN4()
{
    serialPort1.Open();
    cmdbase = "#" + "0" + "P" + "1330" + "S" + "600" + Environment.NewLine;
    cmdspalla = "#" + "1" + "P" + "1060" + "S" + "200" + Environment.NewLine;
```

```
    cmdgomito = "#" + "2" + "P" + "1980" + "S" + "300" + Environment.NewLine;
    cmdpolso = "#" + "3" + "P" + "1600" + "S" + "400" + Environment.NewLine;
    serialPort1.Write(cmdbase + cmdspalla + cmdgomito );
    Thread.Sleep(1000);
    serialPort1.Write(cmdpolso);
    serialPort1.Close();
}
// posizione bancale 5
private void POSBAN5()
{
    serialPort1.Open();
    cmdbase = "#" + "0" + "P" + "1100" + "S" + "600" + Environment.NewLine;
    cmdspalla = "#" + "1" + "P" + "1060" + "S" + "200" + Environment.NewLine;
    cmdgomito = "#" + "2" + "P" + "1980" + "S" + "300" + Environment.NewLine;
    cmdpolso = "#" + "3" + "P" + "1600" + "S" + "400" + Environment.NewLine;
    serialPort1.Write(cmdbase + cmdspalla + cmdgomito );
    Thread.Sleep(1000);
    serialPort1.Write(cmdpolso);
    serialPort1.Close();

}
// posizione bancale 6
private void POSBAN6()
{
    serialPort1.Open();
    cmdbase = "#" + "0" + "P" + "890" + "S" + "600" + Environment.NewLine;
    cmdspalla = "#" + "1" + "P" + "1060" + "S" + "200" + Environment.NewLine;
    cmdgomito = "#" + "2" + "P" + "1970" + "S" + "300" + Environment.NewLine;
    cmdpolso = "#" + "3" + "P" + "1600" + "S" + "400" + Environment.NewLine;
```

```
        serialPort1.Write(cmdbase + cmdspalla + cmdgomito );
        Thread.Sleep(1000);
        serialPort1.Write(cmdpolso);
        serialPort1.Close();

    }
    // posizione A1
    private void posizione_A1()
    {
        serialPort1.Open();
        cmdrotazione = "#" + "4" + "P" + "1550" + "S" + "600" + Environment.NewLine;
        serialPort1.Write(cmdrotazione);
        serialPort1.Close();
        x = -6.5;
        y = 18;
        tasti();
    }
    // posizione A2
    private void posizione_A2()
    {
        serialPort1.Open();
        cmdrotazione = "#" + "4" + "P" + "1450" + "S" + "600" + Environment.NewLine;
        serialPort1.Write(cmdrotazione);
        serialPort1.Close();
        x = -1.35;
        y = 17;
        tasti();
    }
    // posizione A3
```

```
private void posizione_A3()
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1315" + "S" + "600" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    x = 4.6;
    y = 16;
    tasti();
}
// posizione A4
private void posizione_A4()
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1200" + "S" + "600" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    x = 9;
    y = 16.5;
    tasti();
}
// posizione B1
private void posizione_B1()
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1600" + "S" + "600" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    x = -7;
```

```
        y = 12.5;
        tasti();
    }
    // posizione B2
    private void posizione_B2()
    {
        serialPort1.Open();
        cmdrotazione = "#" + "4" + "P" + "1470" + "S" + "600" + Environment.NewLine;
        serialPort1.Write(cmdrotazione);
        serialPort1.Close();
        x = -1.45;
        y = 12;
        tasti();
    }
    //
    // posizione B3
    //
    private void posizione_B3()
    {
        serialPort1.Open();
        cmdrotazione = "#" + "4" + "P" + "1300" + "S" + "600" + Environment.NewLine;
        serialPort1.Write(cmdrotazione);
        serialPort1.Close();
        x = 4;
        y = 12;
        tasti();
    }
    //
    // posizione B4
```

```
//
private void posizione_B4()
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1200" + "S" + "600" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    x = 8.5;
    y = 11.25;
    tasti();
}
//
// posizione C1
//
private void posizione_C1()
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1650" + "S" + "600" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    x = -7;
    y = 7.7;
    tasti();
}
// posizione C2
private void posizione_C2()
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1475" + "S" + "600" + Environment.NewLine;
```

```
        serialPort1.Write(cmdrotazione);
        serialPort1.Close();
        x = -2.1;
        y = 7.8;
        tasti();
    }
    // posizione C3
    private void posizione_C3()
    {
        serialPort1.Open();
        cmdrotazione = "#" + "4" + "P" + "1300" + "S" + "600" + Environment.NewLine;
        serialPort1.Write(cmdrotazione);
        serialPort1.Close();
        x = 3.6;
        y = 7.5;
        tasti();

    }
    // posizione C4
    private void posizione_C4()
    {
        serialPort1.Open();
        cmdrotazione = "#" + "4" + "P" + "1100" + "S" + "600" + Environment.NewLine;
        serialPort1.Write(cmdrotazione);
        serialPort1.Close();
        x = 8.50;
        y = 7;
        tasti();
    }
```

```
// posizione D1
private void posizione_D1()
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1750" + "S" + "600" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    x = -7;
    y = 3;
    tasti();
}
// posizione D2
private void posizione_D2()
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1550" + "S" + "600" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    x = -2.3;
    y = 2.8;
    tasti();

}
// posizione D3
private void posizione_D3()
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1250" + "S" + "600" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
```

```
        serialPort1.Close();
        x = 3;
        y = 2.5;
        tasti();

    }
    // posizione D4
    private void posizione_D4()
    {
        serialPort1.Open();
        cmdrotazione = "#" + "4" + "P" + "1015" + "S" + "600" + Environment.NewLine;
        serialPort1.Write(cmdrotazione);
        serialPort1.Close();
        x = 8;
        y = 2;
        tasti();
    }
    // posizine PINZA APERTA (pAP)
    private void pAP()
    {
        serialPort1.Open();
        cmdpinza = "#" + "5" + "P" + "700" + "S" + "600" + Environment.NewLine;
        serialPort1.Write(cmdpinza);
        Thread.Sleep(1500);
        serialPort1.Close();
    }
    // posizine PINZA CHIUSA (pCH)
    private void pCH()
    {
```

```
        serialPort1.Open();
        cmdpinza = "#" + "5" + "P" + "1200" + "S" + "600" + Environment.NewLine;
        serialPort1.Write(cmdpinza);
        Thread.Sleep(1500);
        serialPort1.Close();
    }
    //MOVIMENTI TASTIERA APERTURA CHIUSA
    //MOVIMENTI A1 PRENDI (TA1P)
    private void TA1P()
    {
        pAP();
        posizione_A1 ();
        // Thread.Sleep(1000);
        Thread.Sleep(1000);
        pCH();
    }
    //MOVIMENTI A1 LASCIA (TA1L)
    private void TA1L()
    {
        posizione_A1();
        Thread.Sleep(1000);
        pAP();
    }
    //MOVIMENTI A2 PRENDI (TA2P)
    private void TA2P()
    {
        pAP();
        posizione_A2();
        Thread.Sleep(1000);
```

```
        pCH();
    }
    //MOVIMENTI A2 LASCIA (TA2L)
    private void TA2L()
    {
        posizione_A2();
        Thread.Sleep(1000);
        pAP();
    }
    //MOVIMENTI A3 PRENDI (TA3P)
    private void TA3P()
    {
        pAP();
        posizione_A3();
        Thread.Sleep(1000);
        pCH();
    }
    //MOVIMENTI A3 LASCIA (TA3L)
    private void TA3L()
    {
        posizione_A3();
        Thread.Sleep(1000);
        pAP();
    }
    //MOVIMENTI A4 PRENDI (TA4P)
    private void TA4P()
    {
        pAP();
        posizione_A4();
```

```
        Thread.Sleep(1000);
        pCH();
    }
    //
    //MOVIMENTI A4 LASCIA (TA4L)
    private void TA4L()
    {
        posizione_A4();
        Thread.Sleep(1000);
        pAP();
    }
    //MOVIMENTI B1 PRENDI (TB1P)
    private void TB1P()
    {
        pAP();
        posizione_B1();
        Thread.Sleep(1000);
        pCH();
    }
    //MOVIMENTI B1 LASCIA (TB1L)
    private void TB1L()
    {
        posizione_B1();
        Thread.Sleep(1000);
        pAP();
    }
    //MOVIMENTI B2 PRENDI (TB2P)
    private void TB2P()
    {
```

```
        pAP();
        posizione_B2();
        Thread.Sleep(1000);
        pCH();
    }
    //MOVIMENTI B2 LASCIA (TB2L)
    private void TB2L()
    {
        posizione_B2();
        Thread.Sleep(1000);
        pAP();
    }
    //MOVIMENTI B3 PRENDI (TB3P)
    private void TB3P()
    {
        pAP();
        posizione_B3();
        Thread.Sleep(1000);
        pCH();
    }
    //MOVIMENTI B3 LASCIA (TB3L)
    private void TB3L()
    {
        posizione_B3();
        Thread.Sleep(1000);
        pAP();
    }
    //MOVIMENTI B4 PRENDI (TB4P)
    private void TB4P()
```

```
{
    pAP();
    posizione_B4();
    Thread.Sleep(1000);
    pCH();
}
//MOVIMENTI B4 LASCIA (TB4L)
private void TB4L()
{
    posizione_B4();
    Thread.Sleep(1000);
    pAP();
}
//MOVIMENTI C1 PRENDI (TC1P)
private void TC1P()
{
    pAP();
    posizione_C1();
    Thread.Sleep(1000);
    pCH();
}
//MOVIMENTI C1 LASCIA (TC1L)
private void TC1L()
{
    posizione_C1();
    Thread.Sleep(1000);
    pAP();
}
//MOVIMENTI C2 PRENDI (TC2P)
```

```
private void TC2P()
{
    pAP();
    posizione_C2();
    Thread.Sleep(1000);
    pCH();
}
//MOVIMENTI C2 LASCIA (TC2L)
private void TC2L()
{
    posizione_C2();
    Thread.Sleep(1000);
    pAP();
}
//MOVIMENTI C3 PRENDI (TC3P)
private void TC3P()
{
    pAP();
    posizione_C3();
    Thread.Sleep(1000);
    pCH();
}
//MOVIMENTI C3 LASCIA (TC3L)
private void TC3L()
{
    posizione_C3();
    Thread.Sleep(1000);
    pAP();
}
```

```
//MOVIMENTI C4 PRENDI (TC4P)
private void TC4P()
{
    pAP();
    posizione_C4();
    Thread.Sleep(1000);
    pCH();
}
//MOVIMENTI C4 LASCIA (TC4L)
private void TC4L()
{
    posizione_C4();
    Thread.Sleep(1000);
    pAP();
}
//MOVIMENTI D1 PRENDI (TD1P)
private void TD1P()
{
    pAP();
    posizione_D1();
    Thread.Sleep(1000);
    pCH();
}
//MOVIMENTI D1 LASCIA (TD1L)
private void TD1L()
{
    posizione_D1();
    Thread.Sleep(1000);
    pAP();
```

```
    }
    //MOVIMENTI D2 PRENDI (TD2P)
    private void TD2P()
    {
        pAP();
        posizione_D2();
        Thread.Sleep(1000);
        pCH();
    }
    //MOVIMENTI D2 LASCIA (TD2L)
    private void TD2L()
    {
        posizione_D2();
        Thread.Sleep(1000);
        pAP();
    }
    //MOVIMENTI D3 PRENDI (TD3P)
    private void TD3P()
    {
        pAP();
        posizione_D3();
        Thread.Sleep(1000);
        pCH();
    }
    //MOVIMENTI D3 LASCIA (TD3L)
    private void TD3L()
    {
        posizione_D3();
        Thread.Sleep(1000);
```

```
        pAP();
    }
    //MOVIMENTI D4 PRENDI (TD4P)
    private void TD4P()
    {
        pAP();
        posizione_D4();
        Thread.Sleep(1000);
        pCH();
    }
    //MOVIMENTI D4 LASCIA (TD4L)
    private void TD4L()
    {
        posizione_D4();
        Thread.Sleep(1000);
        pAP();
    }
   // QUANDO PREMI IL PULSANTE
    // pulsante A1 Prendi
    private void cmdA1_Click(object sender, EventArgs e)
    {
        TA1P();
        Thread.Sleep(800);
        posizione_riposo();
    }
    //  PULSANTE A1 lascia
    private void cmdA1L_Click(object sender, EventArgs e)
    {
        TA1L();
```

```
        Thread.Sleep(800);
        posizione_riposo();
    }
    // PULSANTE A2 prendi
    private void cmdA2P_Click(object sender, EventArgs e)
    {
        TA2P();
        Thread.Sleep(800);
        posizione_riposo();
    }
    //PLSANTE A2 lascia
    private void cmdA2L_Click(object sender, EventArgs e)
    {
        TA2L();
        Thread.Sleep(800);
        posizione_riposo();
    }
    //PULSANTE A3 lascia
    private void cmdA3L_Click(object sender, EventArgs e)
    {
        TA3L();
        Thread.Sleep(800);
        posizione_riposo();
    }
    // PULSANTE A3 Prendi
    private void cmdA3P_Click(object sender, EventArgs e)
    {
        TA3P();
        Thread.Sleep(800);
```

```
        posizione_riposo();
    }
    // PULSANTE A4 lascia
    private void cmdA4L_Click(object sender, EventArgs e)
    {
        TA4L();
        Thread.Sleep(800);
        posizione_riposo();
    }
    // PULSANTE A4 prendi
    private void cmdA4P_Click(object sender, EventArgs e)
    {
        TA4P();
        Thread.Sleep(800);
        posizione_riposo();
    }
    //PULSANTE B1 lascia
    private void cmdB1L_Click(object sender, EventArgs e)
    {
        TB1L();
        Thread.Sleep(800);
        posizione_riposo();
    }
    //PULSANTE B1 prendi
    private void cmdB1P_Click(object sender, EventArgs e)
    {
        TB1P();
        Thread.Sleep(800);
        posizione_riposo();
```

```
}
// PULSANTE B2 prendi
private void cmdB2P_Click(object sender, EventArgs e)
{
    TB2P();
    Thread.Sleep(800);
    posizione_riposo();
}
//PULSANTE B2 Lascia
private void cmdB2L_Click(object sender, EventArgs e)
{
    TB2L();
    Thread.Sleep(800);
    posizione_riposo();
}
// PULSANTE B3 Prendi
private void cmdB3P_Click(object sender, EventArgs e)
{
    TB3P();
    Thread.Sleep(800);
    posizione_riposo();
}
// PULSANTE B3 lascia
private void cmdB3L_Click(object sender, EventArgs e)
{
    TB3L();
    Thread.Sleep(800);
    posizione_riposo();
}
```

```
// PULSANTE B4 Prendi
private void cmdB4P_Click(object sender, EventArgs e)
{
    TB4P();
    Thread.Sleep(800);
    posizione_riposo();
}
// PUSANTE B4 lascia
private void cmdB4L_Click(object sender, EventArgs e)
{
    TB4L();
    Thread.Sleep(800);
    posizione_riposo();
}
//PULSANTE C1 lascia
private void cmdC1L_Click(object sender, EventArgs e)
{
    TC1L();
    Thread.Sleep(800);
    posizione_ripososp();
}
//PULSANTE C1 prendi
private void cmdC1P_Click(object sender, EventArgs e)
{
    TC1P();
    Thread.Sleep(800);
    posizione_ripososp();
}
// PULSANTE C2 prendi
```

```
private void cmdC2P_Click(object sender, EventArgs e)
{
    TC2P();
    Thread.Sleep(800);
    posizione_ripososp();
}
//PULSANTE C2 Lascia
private void cmdC2L_Click(object sender, EventArgs e)
{
    TC2L();
    Thread.Sleep(800);
    posizione_ripososp();
}
// PULSANTE C3 Prendi
private void cmdC3P_Click(object sender, EventArgs e)
{
    TC3P();
    Thread.Sleep(800);
    posizione_ripososp();
}
// PULSANTE C3 lascia
private void cmdC3L_Click(object sender, EventArgs e)
{
    TC3L();
    Thread.Sleep(800);
    posizione_ripososp();
}
// PULSANTE C4 Prendi
private void cmdC4P_Click(object sender, EventArgs e)
```

```
{
    TC4P();
    Thread.Sleep(800);
    posizione_ripososp();
}
// PUSANTE C4 lascia
private void cmdC4L_Click(object sender, EventArgs e)
{
    TC4L();
    Thread.Sleep(800);
    posizione_ripososp();
}
//PULSANTE D1 lascia
private void cmdD1L_Click(object sender, EventArgs e)
{
    TD1L();
    Thread.Sleep(800);
    posizione_ripososp();
}
//PULSANTE D1 prendi
private void cmdD1P_Click(object sender, EventArgs e)
{
    TD1P();
    Thread.Sleep(800);
    posizione_ripososp();
}
// PULSANTE D2 prendi
private void cmdD2P_Click(object sender, EventArgs e)
{
```

```
        TD2P();
        Thread.Sleep(800);
        posizione_ripososp();
    }
    //PULSANTE D2 Lascia
    private void cmdD2L_Click(object sender, EventArgs e)
    {
        TD2L();
        Thread.Sleep(800);
        posizione_ripososp();
    }
    // PULSANTE D3 Prendi
    private void cmdD3P_Click(object sender, EventArgs e)
    {
        TD3P();
        Thread.Sleep(800);
        posizione_ripososp();
    }
    // PULSANTE D3 lascia
    private void cmdD3L_Click(object sender, EventArgs e)
    {
        TD3L();
        Thread.Sleep(800);
        posizione_ripososp();
    }
    // PULSANTE D4 Prendi
    private void cmdD4P_Click(object sender, EventArgs e)
    {
        TD4P();
```

```
        Thread.Sleep(800);
        posizione_ripososp();
    }
    // PUSANTE D4 lascia
    private void cmdD4L_Click(object sender, EventArgs e)
    {
        TD4L();
        Thread.Sleep(800);
        posizione_ripososp();
    }
    // PULSANTE VAI a coordiata
    private void button1_Click(object sender, EventArgs e)
    {
        serialPort1.Close();
        x = double.Parse(txtX.Text);
        y = double.Parse(txtY.Text);
        tasti();

    }
    // PULSANTE ROTAZIONE PINZA

    private void comRot_Click(object sender, EventArgs e)
    {
        serialPort1.Open();
        cmdrotazione = "#" + "4" + "P" + (txtRot.Text) + "S" + "600" + Environment.NewLine;
        serialPort1.Write(cmdrotazione);
        Thread.Sleep(2000);
        serialPort1.Close();
    }
```

```
//prendi bancale 1 predi
private void cmdScaffale1_Click(object sender, EventArgs e)
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "400" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    pAP();
    POSBAN1();
    Thread.Sleep(2000);
    pCH();
    Thread.Sleep(800);
    posizione_intermedio();
    posizione_riposo();
}
//LASCIA BANCALE1
private void button2_Click(object sender, EventArgs e)
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "400" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    POSBAN1();
    Thread.Sleep(2000);
    pAP();
    Thread.Sleep(800);
    posizione_intermedio();
    posizione_riposo();
}
```

```
//LASCIA BANCALE2
private void cmdScaffale2L_Click(object sender, EventArgs e)
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "400" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    POSBAN2();
    Thread.Sleep(2000);
    pAP();
    Thread.Sleep(800);
    posizione_intermedio();
    posizione_riposo();
}
//prendi bancale 2
private void cmdScaffale2_Click_1(object sender, EventArgs e)

{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "400" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    pAP();
    POSBAN2();
    Thread.Sleep(2000);
    pCH();
    Thread.Sleep(800);
    posizione_intermedio();
    posizione_riposo();
```

```
}
//prendi bancale 3 predi
private void cmdScaffale3P_Click(object sender, EventArgs e)
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "400" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    pAP();
    POSBAN3 ();
    Thread.Sleep(2000);
    pCH();
    Thread.Sleep(800);
    posizione_intermedio();
    posizione_riposo();
}
//LASCIA BANCALE 3
private void cmdScaffale3L_Click(object sender, EventArgs e)
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "400" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    POSBAN3();
    Thread.Sleep(2000);
    pAP();
    Thread.Sleep(800);
    posizione_intermedio();
    posizione_riposo();
```

```
    }
    // LASCIA BANCALE 4
    private void cmdScaffale4L_Click(object sender, EventArgs e)
    {
        serialPort1.Open();
        cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "400" + Environment.NewLine;
        serialPort1.Write(cmdrotazione);
        serialPort1.Close();
        POSBAN4();
        Thread.Sleep(2000);
        pAP(); Thread.Sleep(800);
        posizione_intermedio();
        posizione_riposo();
    }
    //PRENDE BANCALE 4
    private void cmdScaffale4_Click(object sender, EventArgs e)
    {
        serialPort1.Open();
        cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "400" + Environment.NewLine;
        serialPort1.Write(cmdrotazione);
        serialPort1.Close();
        pAP();
        POSBAN4();
        Thread.Sleep(2000);
        pCH();
        Thread.Sleep(800);
        posizione_intermedio();
        posizione_riposo();
    }
```

```
// PRENDI BANCALE 5
private void cmdScaffale5_Click(object sender, EventArgs e)
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "400" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    pAP();
    POSBAN5();
    Thread.Sleep(2000);
    pCH();
    Thread.Sleep(800);
    posizione_intermedio();
    posizione_riposo();
}
//LASCIA BANCALE 5
private void cmdScaffale5L_Click(object sender, EventArgs e)
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "400" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close(); POSBAN5();
    Thread.Sleep(2000);
    pAP();
    Thread.Sleep(800);
    posizione_intermedio();
    posizione_riposo();
}
//PRENDE BANCALE 6
```

```
private void cmdScaffale6_Click(object sender, EventArgs e)
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "400" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    pAP();
    POSBAN6();
    Thread.Sleep(2000);
    pCH();
    Thread.Sleep(800);
    posizione_intermedio();
    posizione_riposo();
}
//LASCA BANCALE 6
private void cmdScaffale6L_Click(object sender, EventArgs e)
{
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "400" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    POSBAN6();
    Thread.Sleep(2000);
    pAP();
    Thread.Sleep(800);
    posizione_intermedio();
    posizione_riposo();
}
// Posizione oggetto base
```

```
private void posizione_oggetto_base()
{
    x = -33;
    y = 4;
    tasti();
}
//  PRENDI OGGETTO BASE MESSO
private void prendi_oggetto_base()
{
    pAP();
    posizione_oggetto_base();
    Thread.Sleep(2000);
    pCH();
    Thread.Sleep(800);
    posizione_riposo2();
    Thread.Sleep(300);
}
//  PRENDI OGGETTO BASE MESSO 00
private void prendi_oggetto_base00()
{
    pAP();
    posizione_oggetto_base();
    Thread.Sleep(1000);
    pCH();
    Thread.Sleep(800);
}
// prendi oggetto base girato
private void prendi_oggetto_base_girato()
{
```

```
        pAP();
        serialPort1.Open();
        cmdrotazione = "#" + "4" + "P" + "500" + "S" + "1500" + Environment.NewLine;
        serialPort1.Write(cmdrotazione);
        serialPort1.Close();
        x = -33;
        y = 4.30;
        tasti();
        Thread.Sleep(500);
        pCH();
        Thread.Sleep(300);
    }
    // tasto oggetto base
    private void button8_Click(object sender, EventArgs e)
    {
        prendi_oggetto_base();
    }
    // tasto apertura pinza
    private void Pinzaperta_Click(object sender, EventArgs e)
    {
        pAP();
    }

    //tsto chiusura pinza
    private void pinzachiusa_Click(object sender, EventArgs e)
    {
        pCH();
    }
    // posiziona tutto
```

```
private void posiziona_tutto()
{
    posizione_riposo2();
    prendi_oggetto_base00();
    posizione_riposo3();
    POSBAN1 ();
    Thread.Sleep(3100);
    pAP(); Thread.Sleep(800);
    posizione_riposo3();
    prendi_oggetto_base00();
    posizione_riposo3();
    POSBAN2 ();
    Thread.Sleep(3100);
    pAP(); Thread.Sleep(200);
    posizione_riposo3();
    prendi_oggetto_base00();
    posizione_riposo3(); Thread.Sleep(800);
    POSBAN3();
    Thread.Sleep(3100);
    pAP(); Thread.Sleep(400);
    posizione_riposo1(); Thread.Sleep(400);
    prendi_oggetto_base00();
    posizione_riposo3(); Thread.Sleep(900);
    POSBAN4();
    Thread.Sleep(3100);
    pAP(); Thread.Sleep(200);
    posizione_riposo(); Thread.Sleep(400);
    prendi_oggetto_base00();
    posizione_riposo1(); Thread.Sleep(1000);
```

```
    POSBAN5();
    Thread.Sleep(3100);
    pAP(); Thread.Sleep(200);
    posizione_riposo1(); Thread.Sleep(400);
    prendi_oggetto_base00();
    posizione_riposo1(); Thread.Sleep(1200);
    POSBAN6();
    Thread.Sleep(3100);
    pAP(); Thread.Sleep(200);
    posizione_riposo1();
}
// pulsante posiziona
private void cmdPosiziona_tutto_Click(object sender, EventArgs e)
{
    posiziona_tutto();
}
// TASTO DI PARTNZA
private void partenza_Click(object sender, EventArgs e)
{
        serialPort1.Open();
        cmdbase = "#" + "0" + "P" + "1500" + "S" + "500" + Environment.NewLine;
        cmdspalla = "#" + "1" + "P" + "2500" + "S" + "500" + Environment.NewLine;
        cmdgomito = "#" + "2" + "P" + "1100" + "S" + "500" + Environment.NewLine;
        cmdpolso = "#" + "3" + "P" + "1200" + "S" + "500" + Environment.NewLine;
        cmdrotazione = "#" + "4" + "P" + "1450" + "S" + "500" + Environment.NewLine;
        cmdpinza = "#" + "5" + "P" + "1100" + "S"+"500" + Environment.NewLine;
        serialPort1.Write(cmdbase + cmdspalla + cmdpinza + cmdrotazione);
        serialPort1.Write(cmdgomito + cmdpolso);
        serialPort1.Close();
```

```
}
//TASTO GRAZIE**************************************************
private void button9_Click(object sender, EventArgs e)
{
    //        ** Grazie b 1 **
    prendi_oggetto_base_girato();Thread.Sleep(200);
    posizione_intermedio(); posizione_riposo();
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "2200" + "S" + "2000" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    Thread.Sleep(300);
    POSBAN1();
    Thread.Sleep(1000);
    pAP();
    Thread.Sleep(300);
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "2000" + Environment.NewLine;
    serialPort1.Write(cmdrotazione);
    serialPort1.Close();
    Thread.Sleep(100);
    posizione_intermedio();
    Thread.Sleep(200);
    //        *Grazie b2*
    prendi_oggetto_base_girato();
    Thread.Sleep(300);
    posizione_intermedio(); posizione_riposo();
    serialPort1.Open();
    cmdrotazione = "#" + "4" + "P" + "1850" + "S" + "2000" + Environment.NewLine;
```

```
serialPort1.Write(cmdrotazione);
serialPort1.Close();
Thread.Sleep(800);
POSBAN2();
Thread.Sleep(1000);
pAP();
Thread.Sleep(200);
serialPort1.Open();
cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "2000" + Environment.NewLine;
serialPort1.Write(cmdrotazione);
serialPort1.Close();
Thread.Sleep(200);
posizione_intermedio();
Thread.Sleep(200);
//      ** Grazie b3 ***
prendi_oggetto_base_girato();
Thread.Sleep(300);
posizione_intermedio();posizione_riposo();
serialPort1.Open();
cmdrotazione = "#" + "4" + "P" + "1550" + "S" + "2000" + Environment.NewLine;
serialPort1.Write(cmdrotazione);
serialPort1.Close();
Thread.Sleep(800);
POSBAN3();
Thread.Sleep(1300);
pAP();
Thread.Sleep(200);
serialPort1.Open();
cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "2000" + Environment.NewLine;
```

```
serialPort1.Write(cmdrotazione);
serialPort1.Close();
Thread.Sleep(200);
posizione_intermedio();
Thread.Sleep(200);
//**** Grazie b4
prendi_oggetto_base_girato();
Thread.Sleep(300);
posizione_intermedio(); posizione_riposo();
serialPort1.Open();
cmdrotazione = "#" + "4" + "P" + "1300" + "S" + "2000" + Environment.NewLine;
serialPort1.Write(cmdrotazione);
serialPort1.Close();
Thread.Sleep(800);
POSBAN4();
Thread.Sleep(1300);
pAP();
Thread.Sleep(200);


serialPort1.Open();
cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "2000" + Environment.NewLine;
serialPort1.Write(cmdrotazione);
serialPort1.Close();
Thread.Sleep(200);
posizione_intermedio();
Thread.Sleep(200);
//        *Grazie b5*
prendi_oggetto_base_girato();
Thread.Sleep(300);
```

```
posizione_intermedio(); posizione_riposo();
serialPort1.Open();
cmdrotazione = "#" + "4" + "P" + "1100" + "S" + "2000" + Environment.NewLine;
serialPort1.Write(cmdrotazione);
serialPort1.Close();
Thread.Sleep(800);
POSBAN5();
Thread.Sleep(1300);
pAP();
Thread.Sleep(200);
serialPort1.Open();
cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "2000" + Environment.NewLine;
serialPort1.Write(cmdrotazione);
serialPort1.Close();
Thread.Sleep(200);
posizione_intermedio();
Thread.Sleep(200);
//          *Grazie b6*
prendi_oggetto_base_girato();
Thread.Sleep(300);
posizione_intermedio();
Thread.Sleep(300);
posizione_intermedio(); posizione_riposo();
serialPort1.Open();
cmdrotazione = "#" + "4" + "P" + "800" + "S" + "2000" + Environment.NewLine;
serialPort1.Write(cmdrotazione);
serialPort1.Close();
Thread.Sleep(800);
POSBAN6();
```

```
            Thread.Sleep(1300);
            pAP();
            Thread.Sleep(200);
            serialPort1.Open();
            cmdrotazione = "#" + "4" + "P" + "1400" + "S" + "2000" + Environment.NewLine;
            serialPort1.Write(cmdrotazione);
            serialPort1.Close();
            Thread.Sleep(200);
            posizione_intermedio(); posizione_riposo();
            Thread.Sleep(200);
        }
    }
}
```

## IMMAGINI

Braccio dall'alto:

Braccio da davanti:

CALCOLI MATEMATICI

Nel codice finale di Arduino, il mio obiettivo era quello di scrivere delle coordinate di ogni cassetto (x, y e altezza dalla base di legno) e il braccio automaticamente calcola gli impulsi esatti da mandare ad ogni servomotore. Questo processo viene definito come "cinematica inversa": Invece di calcolare la posizione dell'end effector dati gli angoli delle giunture (come accade nella cinematica diretta), si calcolano gli angoli delle giunture necessari per posizionare la pinza in una posizione e orientamento specifici nello spazio. La cinematica diretta è stata utilizzata per impostare la posizione di riposo, mentre quella indiretta è stata utile nell'implementazione dei bancali e della griglia.

TEOREMA DI CARNOT

Per realizzare il progetto è stato molto importante il teorema di Carnot che afferma: in un triangolo, il quadrato di un lato è uguale alla somma dei quadrati degli altri due lati a cui si sottrae il loro doppio prodotto moltiplicato per il

coseno dell'angolo tra loro compreso (e opposto al primo).

$ac2$ = $ab2$ + $bc2$– 2 ∗ ab ∗ bc ∗ cos(β)

Il teorema è stato utile per comprendere l'altezza con la quale il braccio robotico andava a prendere gli oggetti, oltre che per ricavare vari valori.

TEOREMA DI PITAGORA

Il teorema afferma che in tutti i triangoli rettangoli, il quadrato dell'ipotenusa (il lato più lungo di un triangolo) è equivalente alla somma dei quadrati dei cateti.

$c2$ = $a2$ + $b2$

Per ricavare l'ipotenusa si esegue la radice quadrata su entrambi i membri.

$c$ = √ ($a2$ + $b2$)

Una volta prese le misure di tutti i componenti del braccio robotico possiamo ricavare gli impulsi da dare ai servomotori. Inizialmente si forniscono delle coordinate x e y e un'altezza h.

BASE

Riconoscendo il triangolo rettangolo che si crea nella prima immagine, la tangente dell'angolo $\varphi$ vale:

tan $\varphi$ =$x$/ ($y$ + $Db$)

Da cui si può ricavare l'angolo in radianti con la funzione arcotangente

$\varphi\ (in\ radianti)$ = arctan($tan\varphi$)

Per convertire l'angolo da radianti in gradi si fa una proporzione

180 : $\pi$ = $\varphi\ (in\ gradi)$ : $\varphi\ (in\ radianti)$

Adesso si possono calcolare gli impulsi con l'equazione della retta

$impBase$ = $mBase * \varphi\ (in\ gradi)$ + $qBase$

Per calcolare gli alti impulsi bisogna ricavare i valori di r e d grazie al teorema di Pitagora

$r$ = √ ($x$2 + ($y$ + $Db$)2)

$d$ = √ ($r$2 + $h$2)

SPALLA

Per ricavare l'angolo, bisogna calcolare sia $\gamma$ che $\gamma$1 per poi sommarli. Per $\gamma$ si utilizza la formula inversa del teorema ci Carnot

cos($\gamma$) = ($L$22 + $d$2 − $L$12) / (2 * $d$ * L2)

Da cui si può ricavare l'angolo in radianti con la funzione arcocoseno

$\gamma\ (in\ radianti)$ = arccos[cos($\gamma$)]

Convertendo l'angolo da radianti a gradi (moltiplicando il valore per 57.3) si ricava $\gamma$. A questo punto ricaviamo anche $\gamma$1

con la funzione arcotangente e convertendo in gradi $\gamma$1

($in\ gradi$) = [arctan(tan $\gamma$1)] * 57.3

Infine, sommiamo i due angoli e si calcolano gli impulsi con l'equazione della retta

$impSpalla = mSpalla * (\gamma + \gamma 1) + qSpalla$

GOMITO

Per ricavare l'angolo $\beta$, si utilizza la formula inversa del teorema di Carnot

$\cos(\beta) = (L22 + L12 - d2) / (2 * L1 * L2)$

Di seguito si applica la funzione arcocoseno e si converte da radianti a gradi

$\beta\ (in\ gradi) = \arccos[\cos(\beta)] * 53.7$

Infine, si ricava l'equazione della retta per calcolare gli impulsi del gomito

$impGomito = mGomito * \beta\ (in\ gradi) + qGomito$

POLSO

Per ricavare l'ultimo angolo ($\alpha$) basta fare

$\alpha\ (in\ gradi) = 270 - \beta - (\gamma + \gamma 1)$

Infine, si calcolano gli impulsi sempre con l'equazione della retta

$impPolso = mPolso * \alpha\ (in\ gradi) + qPsolo$

Per i giunti della pinza e della rotazione del polso non servono calcoli. Per la pinza ho dato due valori di impulsi fissi: uno per l'apertura della pinza e l'altro per la chiusura. Per la rotazione del polso mi sono aiutato col goniometro e impostato un valore di impulsi a 0°, 45° e 90°.

SITOGRAFIA

Libro: “L@borobotica volume B"

Sito: "https://www.laborobotica.com/"