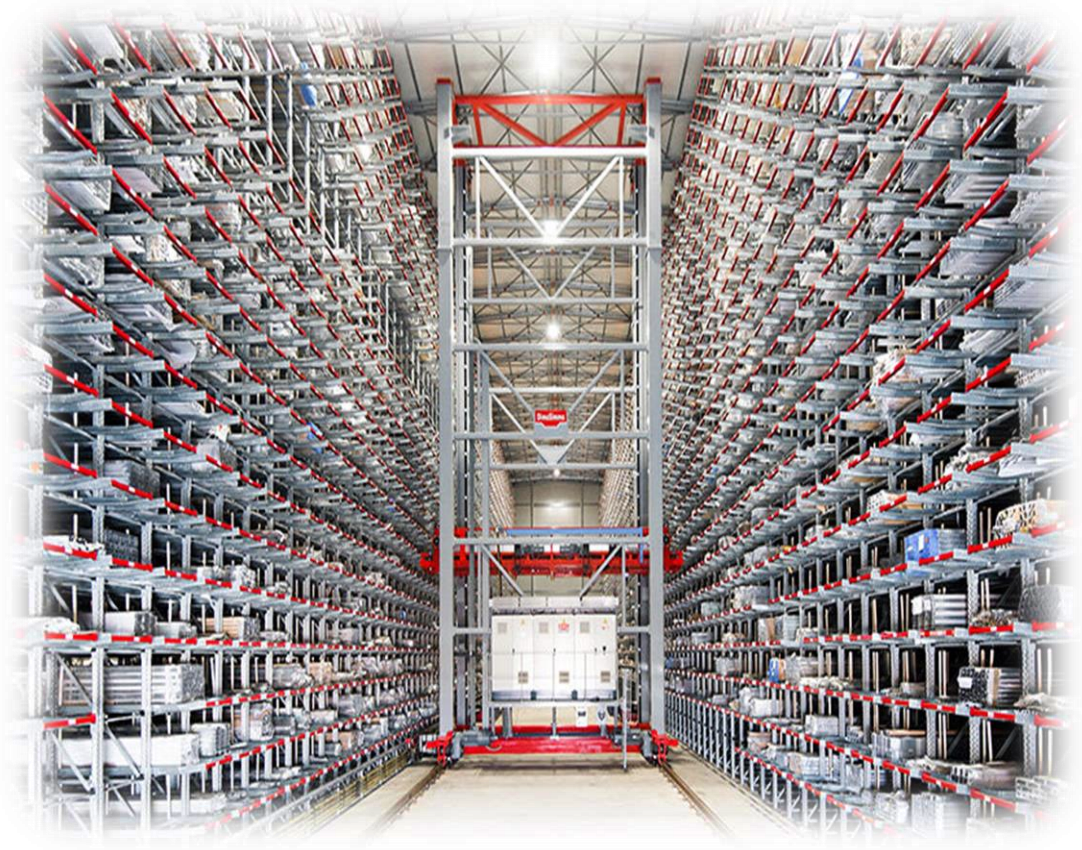
Paolo Modafferi

5A ROB

MAGAZZINO AUTOMATICO



INDICE

INTRODUZIONE

Una cosa che ritengo importante per un'azienda è possedere un magazzino ben organizzato. Da qui nasce la mia idea: creare un prototipo di magazzino automatico in miniatura.

Il mio progetto è di facile utilizzo e semplice da trasportare/spostare; costituito da una cassettiera porta-minuteria, che rappresenta il magazzino, da un braccio robot antropomorfo controllato tramite applicazione e da un pannello di controllo. L'app si può scaricare dal QR code posto a fianco del braccio; ciò significa che tutti possono usufruire del magazzino. Il collegamento tra l'applicazione e il braccio avviene tramite il Bluetooth e, quando una persona ha bisogno di un oggetto che si trova nel magazzino, con l'applicazione può selezionare il prodotto che vuole e il braccio apre l'apposito cassetto. Quando il cassetto è stato aperto si può prelevare il prodotto e al termine si chiude il cassetto.

Se però c'è bisogno di aggiungere prodotti al magazzino, l'applicazione ha una sua sezione fatta apposta per questo, però solo il manager del magazzino (in questo caso sarei io) può entrarci. Difatti l'app richiede una password per entrare nella sezione del manager. Per aggiungere prodotti, si deve posizionare l'oggetto da aggiungere nella zona segnata col nastro nero. In seguito il manager si collega al braccio e seleziona il prodotto che vuole aggiungere. Terminata la selezione, il braccio aprirà il cassetto, prende l'oggetto da aggiungere, lo mette dentro al cassetto e chiude.

Per evitare che più persone si collegano al braccio, il pannello di controllo avvisa con 2 led (uno rosso e uno verde) se il magazzino è al momento in utilizzo da qualcuno o no. Di preciso: se nessuno è collegato al braccio, il led verde è acceso e quello rosso è spento. Se una persona si collega al braccio, allora il led rosso si accende e quello verde si spegne.

COMPONENTI

Materiali/strumenti:

- Lynxmotion AL5D Braccio robot a 6 assi
- 6 servomotori
- Cassettiera porta-minuteria a 12 scomparti
- Batterie AA da 1.5 Volt
- Batterie AAA da 1.5 Volt
- Asse di legno
- Viti
- Trapano a batteria
- Breadboard
- Cavi
- HC-05 modulo Bluetooth
- 2 resistenze da 1KΩ
- Multimetro digitale
- Led rosso
- Led verde
- Interruttore
- Riga da 60 cm
- Jack di alimentazione
- Smartphone
- Goniometro
- Feltrini adesivi
- Scotch nero
- Anycubic Vyper stampante 3D
- Filamenti PLA rosso, nero, verde-blu, oro, argento

Software/programmi su PC:

- Arduino IDE
- Fusion 360
- Ultimaker Cura
- Microsoft Excel
- Visual Studio

Siti Web:

- MIT App Inventor

Schede elettroniche:

- Arduino Uno
- SSC-32U

Alimentazione:

- Batteria da 9 Volt
- Alimentatore da 6 Volt

HC-05

Il modulo Bluetooth HC-05 è un modulo che trasforma una porta UART (Universal Asynchronous Receiver-Transmitter, la seriale del PC) in una porta Bluetooth, generalmente con profilo SPP (Serial Port Profile); diventando così una seriale over Bluetooth.

Ha una portata media di 10 metri ed è utilizzato per far comunicare un microprocessore con uno smartphone. Per programmare questo modulo si utilizzano comandi AT, che permettono di impostare il nome del dispositivo, il codice di sicurezza, e molto altro.

Questo componente è dotato di 6 piedini:

- State: serve per verificare se il Bluetooth è connesso o no
- RXD: riceve i dati
- TXD: trasmette i dati
- EN: serve per attivare gli “AT Comands” (i comandi per la configurazione del dispositivo)
- VCC e GND: servono per l’alimentazione del modulo

Nel mio progetto, l’HC-05 viene utilizzato per ricevere dati dall’applicazione ed elaborarli su Arduino. Perciò ho utilizzato due seriali: una è quella che serve per mandare dati alla scheda SSC-32U, l’altra è quella del modulo Bluetooth. Per ottenere la seriale del Bluetooth si utilizza la libreria “Software Serial”.

ARDUINO UNO

Arduino è una scheda elettronica programmabile basata sul microcontrollore “ATmega328P”. È dotata di una sezione relativa all’alimentazione, una con i pin analogici e l’altra con i pin digitali.

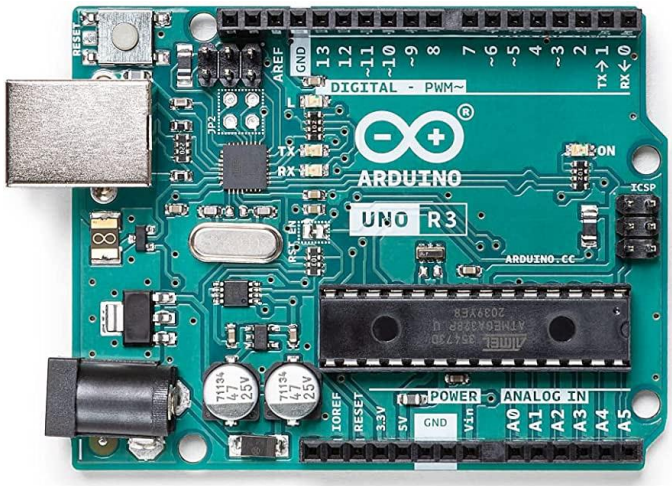
I pin analogici sono ingressi che permettono di ricevere segnali che provengono, per esempio, da sensori (trimmer, fotoresistenze, ultrasuoni, ecc.). Questa tensione viene convertita in un valore binario per farla leggere al microcontrollore. Il convertitore di Arduino lavora a 10 bit; ciò vuol dire che potrà esprimere 1023 valori diversi (1024 step). Inoltre Arduino può leggere valori di tensioni che stanno tra 0 e 5 Volt.

I pin digitali possono essere utilizzati sia come input, sia come output: quindi possono ricevere e spedire informazioni. Ci sono alcuni pin digitali che sono dotati della funzione PWM (Pulse Width Modulation). Questi permettono di spedire o ricevere valori da 0 a 255 e quindi rendere di fatto i pin digitali molto simili nelle potenzialità a quelli analogici.

La programmazione di Arduino Uno avviene tramite un software chiamato Arduino IDE il quale consente di programmare la scheda. Per caricare un codice, creato sul software, si collega Arduino al PC tramire un cavo USB e si carica il programma. Inoltre Arduino può essere utilizzato senza conza collegarlo al PC. Difatti, a fianco del connettore USB, c’è una boccola di alimentazione, che ci permette di alimentare Arduino senza necessità del collegamento col PC.

Nel mio progetto, Arduino fa da tramite tra il modulo Bluetooth e la scheda SSC-32U.Quando Arduino riceve un dato, ottenuto grazie all’HC-05, lo elabora e, in base al tipo di dato, manda un comando alla SSC-32U tramite la seriale.

Per alimentare Arduino utilizzo la batteria da 9V.

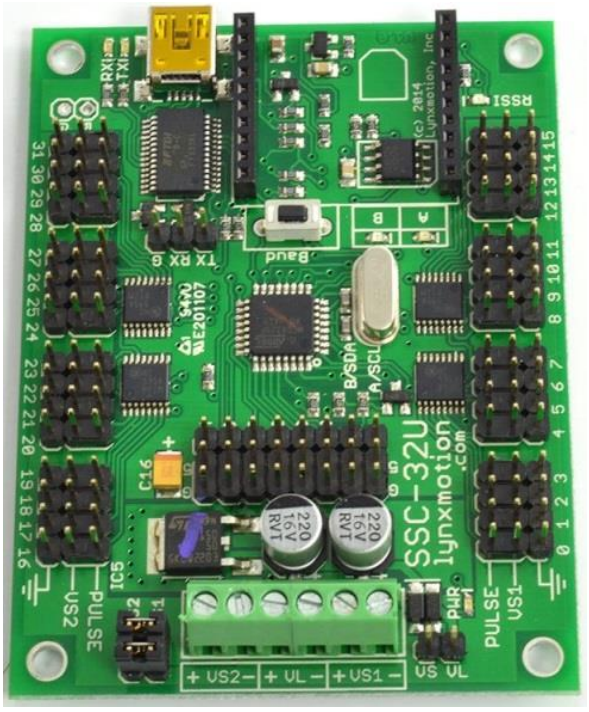


SSC-32U

La scheda SSC-32U è una scheda elettronica che permette di pilotare 32 servomotori, individuabili da un numero d'ordine variabile da 0 a 31. I connettori dei servomotori vanno opportunamente inseriti negli appositi slot: verso l'esterno della scheda troviamo la massa (GND), in mezzo si trova l'alimentazione (che proviene dai connettori VS1 o VS2) e verso l'interno della scheda si trova lo slot per il segnale (PULSE). L'alimentazione si suddivide tra quella necessaria per il funzionamento della parte logica (VL) e quella per i servomotori (VS1 per quelli da 0 a 15 e VS2 per quelli da 16 a 31). Per quanto riguarda la connessione col PC, la scheda dispone di un connettore mini USB. Per programmare la scheda si manda ad essa una stringa con il seguente formato:

#<ch>P<pw>S<spd>

- # rappresenta il carattere iniziale
- <ch> è il numero del motore al quale si manda un impulso
- P<pw> è la durata degli impulsi (che varia tra 500 µs e 2500 µs per portare un servomotore da 0° a 180°)
- S<spd> è la velocità di esecuzione misurata in µs per secondo.

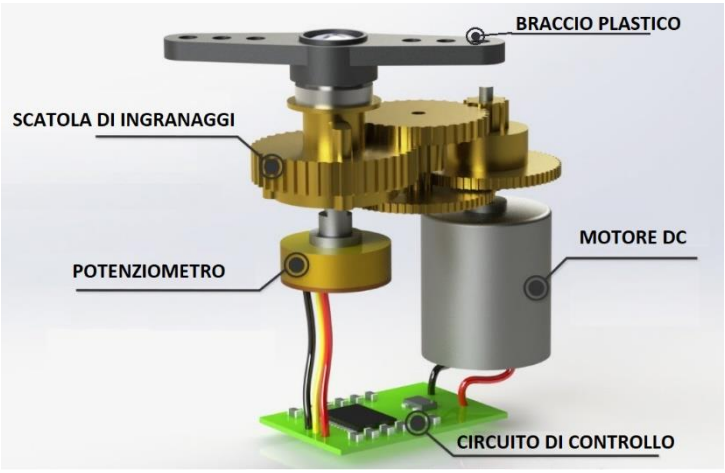


Nel mio progetto alimento la SSC-32U con l'alimentatore da 6V col connettore VS1 perché utilizzo gli slot che vanno da 0 a 5. Ogni slot viene usato per comandare i 6 servomotori del braccio.

La scheda deve ricevere la stringa attraverso Arduino e per farlo utilizzo gli slot RX e G. RX riceve la stringa, mentre G è la massa.

SERVOMOTORI

Un servomotore è un tipo di motore che si presenta come un piccolo contenitore di materiale plastico, da cui fuoriesce un perno in grado di ruotare in un angolo compreso tra 0° e 180°, mantenendo stabilmente la posizione raggiunta. La rotazione del perno avviene grazie ad un motore in corrente continua e un meccanismo di demoltiplica che consente di aumentare la coppia di fase di rotazione. La rotazione del motore viene effettuata grazie a un circuito di controllo, situato all'interno della struttura, in grado di rilevare l'angolo di rotazione raggiunto dal perno, tramite un potenziometro resistivo, e bloccare il motore sul punto desiderato.

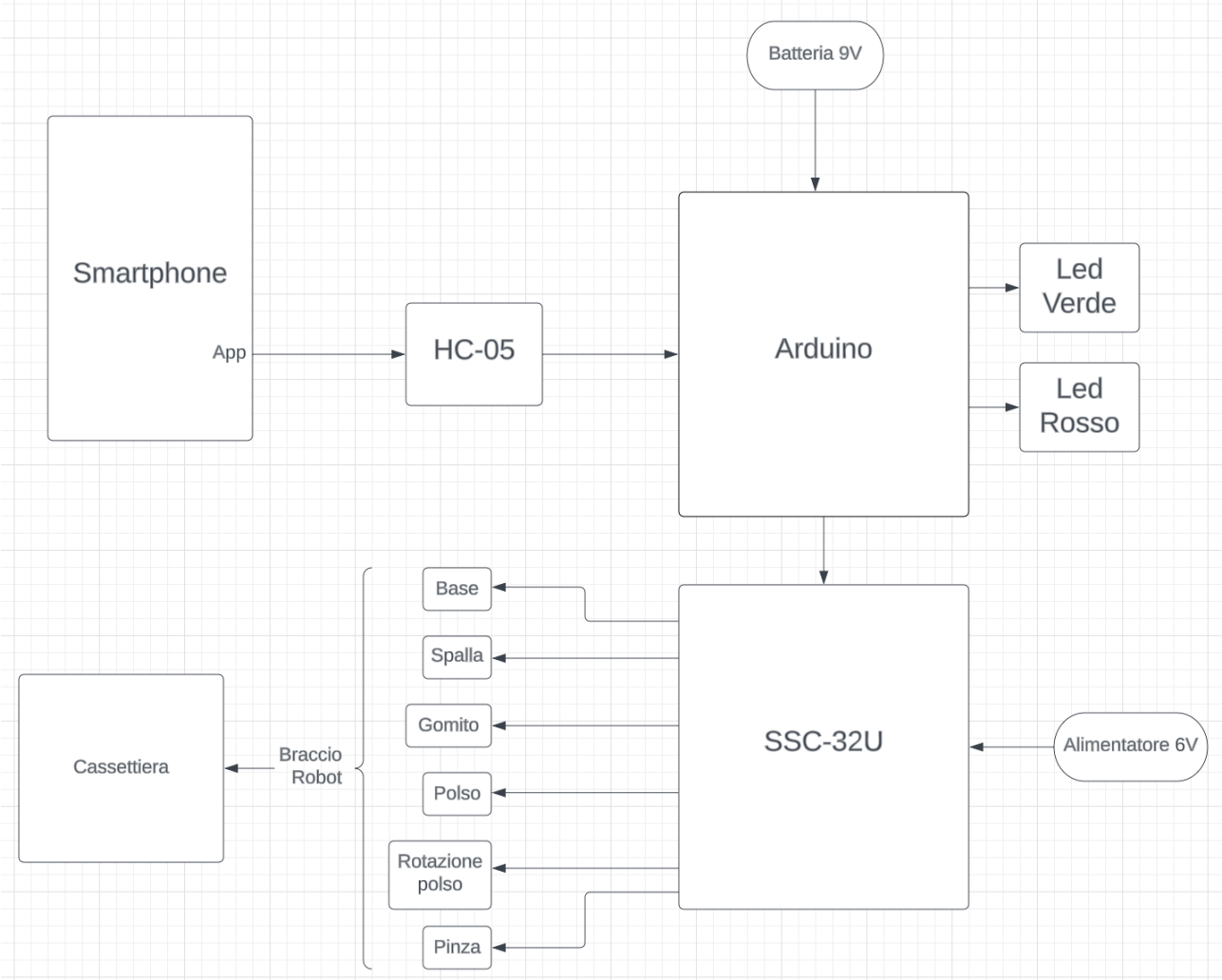


Per far girare il motore bisogna inviare un segnale digitale al circuito di controllo. Questo segnale è di tipo impulsivo, o PWM, e il suo valore varia tra 500 µs e 2500 µs. Gli impulsi rappresentano l'angolo di rotazione del perno.

I servomotori sono motori a bassa precisione, ma possono essere utilizzati per piccoli robot. Se si vuole utilizzare un motore più preciso bisogna utilizzare dei motori passo-passo.

Nel mio progetto li utilizzo per il braccio robot. Essi rappresentano gli assi del braccio: la base, la spalla, il gomito, il polso, la pinza e la rotazione del polso.

SCHEMA A BLOCCHI



Lo schema a blocchi riportato qui sopra rappresenta graficamente il magazzino automatico.

REALIZZAZIONE

1) Montaggio del braccio robot

Per prima cosa bisogna costruire il braccio antropomorfo. L'ho preso da una ditta americana (la Lynxmotion), che fornisce questi prodotti realizzati in alluminio. Il pacchetto comprende la struttura del braccio (con viti, bulloni, pezzi in alluminio, ecc.), 6 servomotori, la scheda SSC-32U, l'alimentatore da 6V e un interruttore. Tutto il sistema viene fissato su di un'asse di legno in modo da mantenere il braccio stabile.

Alla fine della costruzione del braccio, si passa alla taratura dei servomotori.

2) Taratura dei servomotori

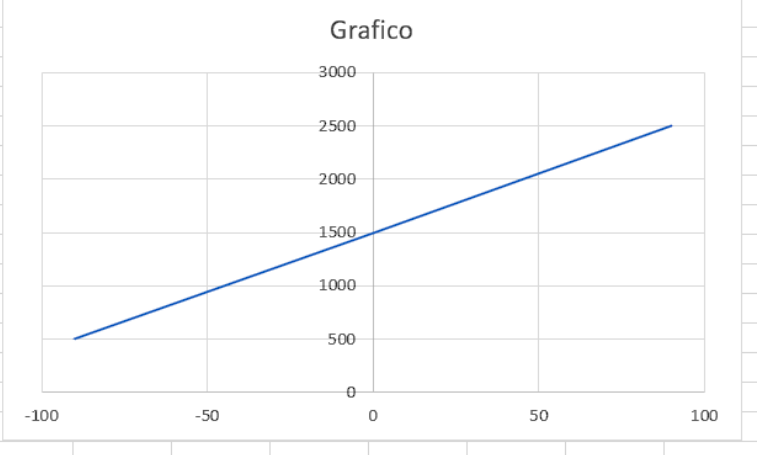
Con l'espressione "taratura" si intende la determinazione della corrispondenza tra angolo dei giunti (base, spalla, gomito, ecc.) e durata degli impulsi che bisogna fornire ai servomotori. Questa parte del progetto è estremamente importante; basta un piccolo errore di taratura di un singolo servo per compromettere la precisione di tutto il braccio. Lo scopo di questa procedura è quello di ricavare un'equazione che determina la durata degli impulsi da attribuire al servomotore, a seguito di un qualsiasi valore di angolo impostato.

Per eseguire la taratura si utilizza il software Visual Studio e un foglio di Microsoft Excel. Su Visual Studio creo un'interfaccia composta da 6 barre a scorrimento (le "trackBar") e 6 caselle di testo (le "textBox") per ogni servomotore. Con le trackBar imposto gli impulsi da mandare al servomotore; perciò, come valore minimo delle trackBar è 500 e come valore massimo 2500. Inoltre, aggiungo anche la comunicazione seriale (la "serialPort") modificando la velocità di trasmissione a 9600 bit al secondo e il numero di porta della SSC-32U (la COM).

Nel codice di Visual Studio scrivo che ogni volta che modifico il valore di ogni trackBar, sulla seriale mando la stringa (quella che legge la SSC-32U) con il numero del servomotore e gli impulsi impostati. Inoltre, leggerò il valore di impulsi che ho impostato nelle textBox. Grazie all'interfaccia, posso capire quanti impulsi devo mandare ad ogni servomotore per raggiungere l'angolo desiderato. Gli angoli da raggiungere si misurano col goniometro e sono diversi per i vari giunti, ma i valori sono i multipli di 45°. Qui sotto riporto l'interfaccia che ho progettato per la taratura:

Su Excel imposto delle tabelle per ogni servomotore, dove segno i valori degli angoli e i loro rispettivi impulsi (ricavati grazie a Visual Studio). In questo modo si può disegnare il grafico dove in ascissa abbiamo gli angoli (espressi in gradi) e in ordinata abbiamo gli impulsi (espressi in µs).

Gradi	Impulsi
-90	500
0	1500
90	2500



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:

$$m = \frac{P2Y - P1Y}{P2X - P1X}$$

3) Montaggio di tutta la struttura

Dopo aver tarato i servomotori, si passa alla costruzione del magazzino. Per realizzarlo ho utilizzato un'asse di legno e l'ho tagliata in più pezzi: il pezzo più grande serve per la base di tutto il progetto e quelli più piccoli li ho utilizzati per fissare il braccio, la cassettiera e il pannello di controllo. Tutti i pezzi più piccoli sono fissati a loro volta su quello più grande. Per tutti i fissaggi ho utilizzato delle viti e il trapano; inoltre ho incollato 4 feltrini sotto la base per non rischiare di rovinare i posti dove appoggio il progetto.

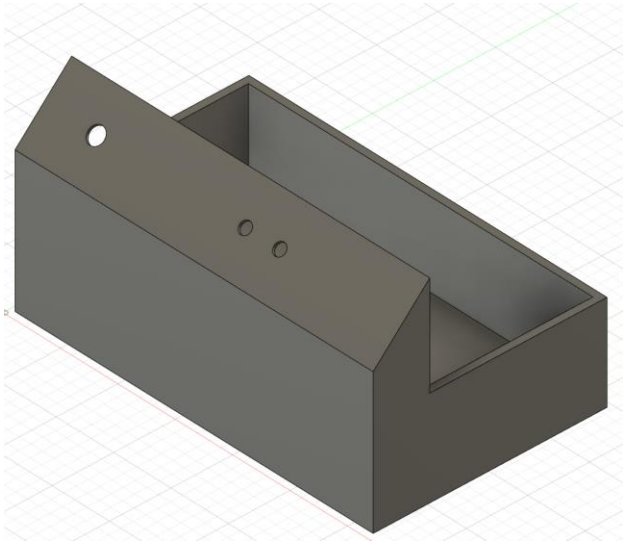
Il pannello di controllo serve per:

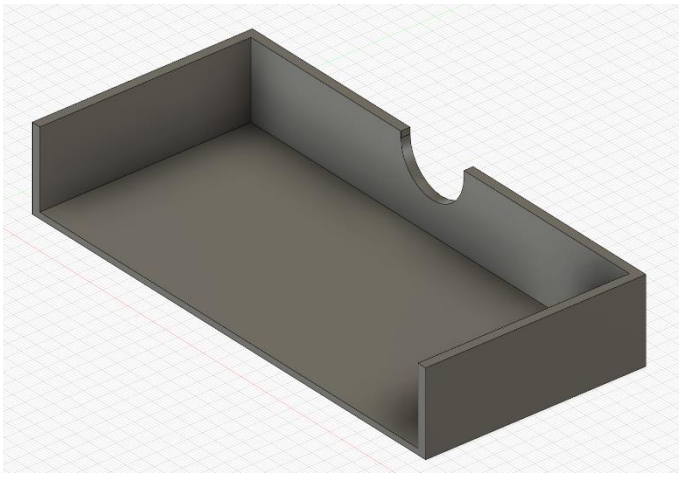
- Coprire il circuito elettrico
- Alimentare il braccio e Arduino
- Controllare se una persona è connessa al magazzino o no

Per costruirlo ho utilizzato la mia stampante 3D: inizialmente l'ho disegnato su Fusion 360 (un software CAD per la progettazione di modelli 3D), poi l'ho caricato su Ultimaker Cura per prepararlo alla stampa e alla fine la stampante 3D ha fatto il suo lavoro. Ho utilizzato un filamento PLA nero per stamparlo.

All'interno del pannello si trovano le due schede, l'HC-05, la breadboard e la batteria da 9V. All'esterno invece si trova l'interruttore, i due led rosso e verde e il jack per l'alimentatore da 6V. Il motivo per cui ho utilizzato la doppia alimentazione sta nel

fatto che le due schede (SSC-32U e Arduino) hanno bisogno di tensioni diverse per essere alimentate.

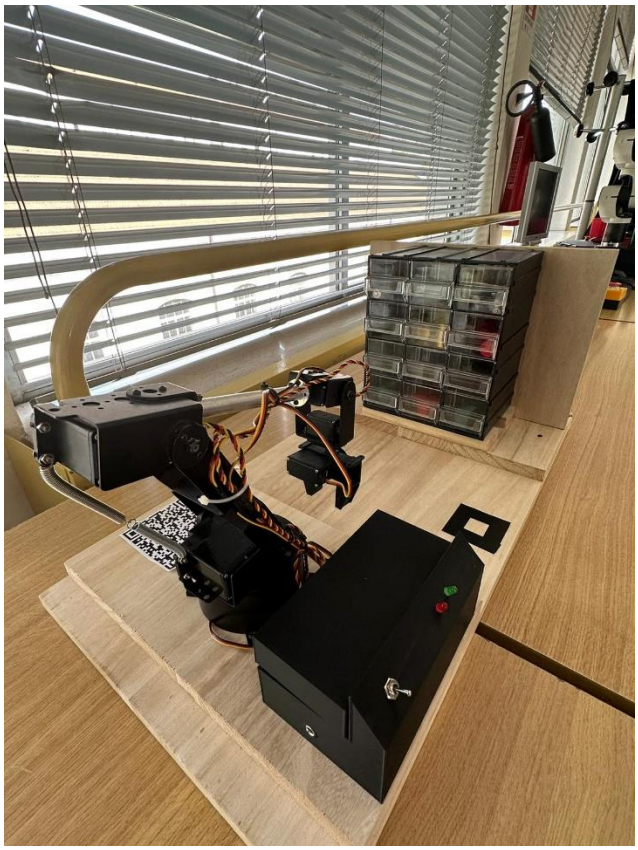


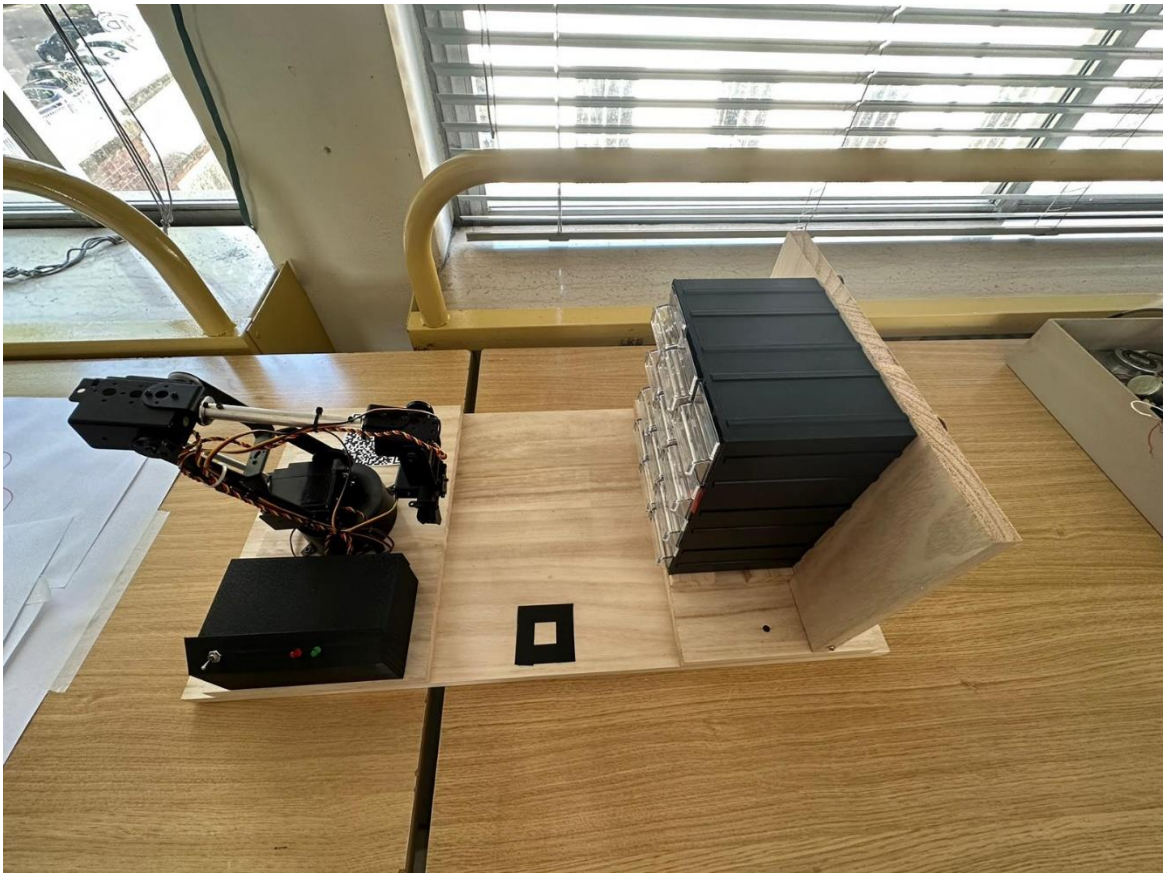


Ho usufruito della stampa 3D anche per costruire i prodotti del magazzino. Li ho fatti di diverse forme (cubi, cilindri e parallelepipedi) e di diversi colori (rosso, nero, oro, argento e verde-blu). Per riempire il magazzino ho anche utilizzato delle batterie di tipo AA e di tipo AAA.

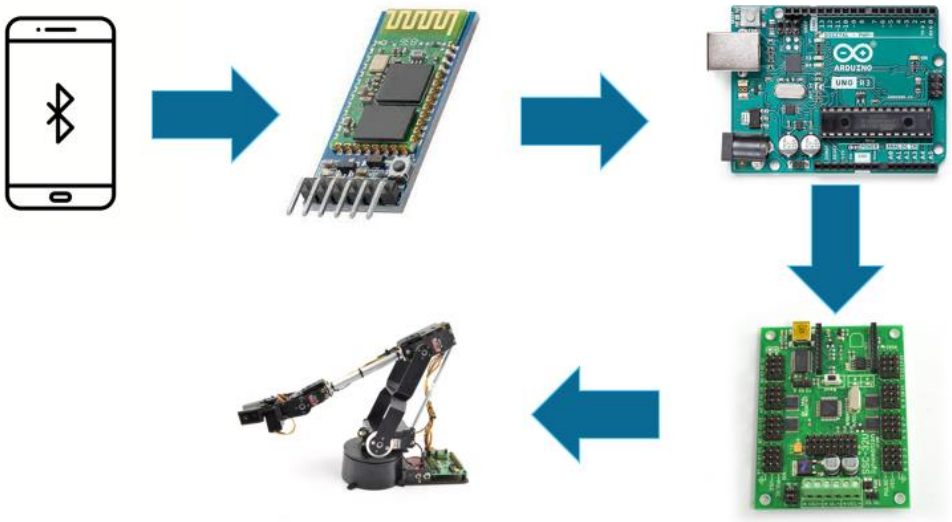
Per finire ho incollato 4 pezzi di scotch nero, per indicare la posizione dei prodotti che verranno aggiunti al magazzino, e il QR code per scaricare l'applicazione.

Ecco il risultato finale:





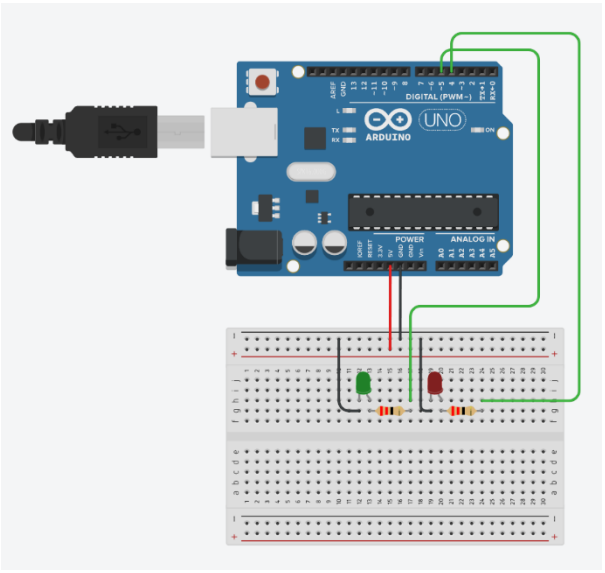
4) Circuito elettrico



Questa è una rappresentazione delle comunicazioni del mio progetto. Lo smartphone, attraverso l'applicazione, comunica con l'HC-05 tramite il Bluetooth (wireless). Arduino riceve i dati, li elabora e infine manda i dati alla SSC-32U. Quest'ultima elabora le stringhe, arrivate da Arduino, e fa muovere gli assi del braccio.

Vediamo il circuito elettrico di Arduino: viene alimentata la scheda attraverso batteria da 9V, in modo da evitare il collegamento col computer. Ho utilizzato l'alimentazione da 5V, il GND e alcuni dei pin digitali.

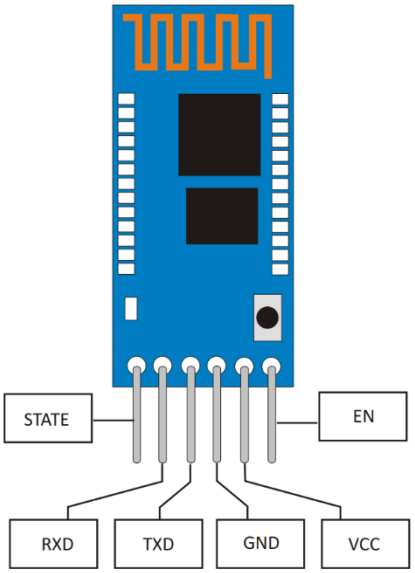
- Pin 3: collegato alla RX della SSC-32U per trasmettere i dati
- Pin 4: controlla il led rosso
- Pin 5: controlla il led verde
- Pin 10: collegato al TXD dell'HC-05 per ricevere i dati.
- Pin 11: collegato all'RXD dell'HC-05 per trasmettere i dati.
- 5V: alimentano il modulo Bluetooth
- GND: massa dei led, dell'HC-05 e della SSC-32U



Per quanto riguarda il modulo bluetooth il collegamenti sono i seguenti:

- RXD: collegato al pin 11 di Arduino
- TXD: collegato al pin 10 di Arduino
- GND: collegato alla massa di Arduino
- VCC: collegato a 5V di Arduino

I pin più esterni (STATE e EN) non mi sono serviti.



Infine, per la SSC-32U, ho utilizzato la VS1 per alimentare i 6 servomotori. L'alimentazione è collegata all'interruttore in serie al Jack per l'alimentazione da 6V. In questo modo posso attivare o disattivare il braccio. I pin che ho utilizzato per i servomotori sono:

- 0 per la base
- 1 per la spalla
- 2 per il gomito
- 3 per il polso
- 4 per la pinza
- 5 per la rotazione del polso

Inoltre ho utilizzato i pin RX e G per ricevere i dati da Arduino. RX è collegato al pin 3 e G alla massa.

5) Programma di Arduino

```
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10, 11);  //seriale HC-05
SoftwareSerial Braccio(2, 3);     //seriale SSC-32U

 //valori m di ogni motore
 double mBase = 9.6556;
 double mSpalla = 8.9889;
 double mGomito = -9.4991;
 double mPolso = 9.9556;

 //valori q di ogni motore
 double qBase = 1279.8;
 double qSpalla = 1379.5;
 double qGomito = 2457.6;
 double qPolso = -266.67;


 //stringhe da mandare alla seriale
 String Base;
 String Spalla;
 String Gomito;
 String Polso;
 String Pinza;
 String rot;

 double L1 = 18.6;
 double L2 = 14.7;
 double distBase = 12.5;
 double d;
 double r;
 double h;
 double gamma1Rad;
 double gamma1Gra;
 double gammaTot;
 double alfaTot;
 double alfa2 = 0;
 double tgFi;
 double fiRad;
 double fiGra;
 double cosBeta;
 double arcocosBeta;
```

```
 double betaGra;
 double cosGamma;
 double arcocosGamma;
 double gammaGra;
 double impAlfa;
 double impGamma;
 double impBeta;
 double impFi;
 double x;
 double y;

 char c;

void setup()
{
 Braccio.begin(9600);       //trasmissione seriale per la SSC-32U
 BTSerial.begin(9600);      //trasmissione seriale per l'HC-05

 pinMode(4, OUTPUT);        //dichiaro i pin dei led in OUTPUT
 pinMode(5, OUTPUT);
 digitalWrite(5, HIGH);     //atrtivo il led verde
}

void loop()
{
   if (BTSerial.available())        //lettura del dato ricevuto
 {
   c = BTSerial.read();
 }

  if(c=='H')                //cicli if
 {
   digitalWrite(4, HIGH);
   digitalWrite(5, LOW);
 }

 if(c=='L')
 {
   digitalWrite(4, LOW);
   digitalWrite(5, HIGH);
 }

 if(c=='A')
 {
```

```
  seq1();
}

if(c=='B')
{
  seq2();
}

if(c=='C')
{
  seq3();
}

 if(c=='D')
{
  seq4();
}

  if(c=='E')
{
  seq5();
}

  if(c=='F')
{
  seq6();
}

  if(c=='G')
{
  seq7();
}

  if(c=='I')
{
  seq8();
}

  if(c=='M')
{
  seq9();
}

   if(c=='N')
```

```
{
  seq10();
}

   if(c=='P')
{
  seq11();
}
  if(c=='J')
{
  seq1();
  prendi();
  delay(1000);
  rot0();
  polso90();
  AdueH();
  delay(1000);
  avanti2();
  delay(1000);
  pinzaA();
  delay(1000);
  indietro2();
  delay(1000);
  Adue();
  delay(1000);
  avanti();
  indietro2();
  start();
  polso0();
  rot0();
}

  if(c=='K')
{
  seq2();
  prendi();
  delay(1000);
  rot0();
  polso90();
  BtreH();
  delay(1000);
  avanti2();
  delay(1000);
  pinzaA();
```

```
  delay(1000);
  indietro2();
  delay(1000);
  Btre();
  delay(1000);
  avanti();
  indietro2();
  start();
  polso0();
  rot0();
}

 if(c=='Q')
{
  seq3();
  prendi();
  delay(1000);
  rot0();
  polso90();
  DdueH();
  delay(1000);
  avanti2();
  delay(1000);
  pinzaA();
  delay(1000);
  indietro2();
  delay(1000);
  Ddue();
  delay(1000);
  avanti();
  indietro2();
  start();
  polso0();
  rot0();
}

 if(c=='R')
{
  seq4();
  prendi();
  delay(1000);
  rot0();
  polso90();
  BunoH();
```

```
  delay(1000);
  avanti2();
  delay(1000);
  pinzaA();
  delay(1000);
  indietro2();
  delay(1000);
  Buno();
  delay(1000);
  avanti();
  indietro2();
  start();
  polso0();
  rot0();
}

 if(c=='S')
{
  seq5();
  prendi();
  delay(1000);
  rot0();
  polso90();
  AunoH();
  delay(1000);
  avanti2();
  delay(1000);
  pinzaA();
  delay(1000);
  indietro2();
  delay(1000);
  Auno();
  delay(1000);
  avanti();
  indietro2();
  start();
  polso0();
  rot0();
}

 if(c=='T')
{
  seq6();
  prendi();
```

```
  delay(1000);
  rot0();
  polso90();
  AtreH();
  delay(1000);
  avanti2();
  delay(1000);
  pinzaA();
  delay(1000);
  indietro2();
  delay(1000);
  Atre();
  delay(1000);
  avanti();
  indietro2();
  start();
  polso0();
  rot0();
}

 if(c=='U')
{
  seq7();
  prendi();
  delay(1000);
  rot0();
  polso90();
  BdueH();
  delay(1000);
  avanti2();
  delay(1000);
  pinzaA();
  delay(1000);
  indietro2();
  delay(1000);
  Bdue();
  delay(1000);
  avanti();
  indietro2();
  start();
  polso0();
  rot0();
}
```

```
 if(c=='V')
{
  seq8();
  prendi();
  delay(1000);
  rot0();
  polso90();
  CdueH();
  delay(1000);
  avanti2();
  delay(1000);
  pinzaA();
  delay(1000);
  indietro2();
  delay(1000);
  Cdue();
  delay(1000);
  avanti();
  indietro2();
  start();
  polso0();
  rot0();
}

 if(c=='W')
{
  seq9();
  prendi();
  delay(1000);
  rot0();
  polso90();
  CtreH();
  delay(1000);
  avanti2();
  delay(1000);
  pinzaA();
  delay(1000);
  indietro2();
  delay(1000);
  Ctre();
  delay(1000);
  avanti();
  indietro2();
  start();
```

```
  polso0();
  rot0();
 }

 if(c=='X')
 {
  seq10();
  prendi();
  delay(1000);
  rot0();
  polso90();
  DunoH();
  delay(1000);
  avanti2();
  delay(1000);
  pinzaA();
  delay(1000);
  indietro2();
  delay(1000);
  Duno();
  delay(1000);
  avanti();
  indietro2();
  start();
  polso0();
  rot0();
 }

 if(c=='Y')
 {
  seq11();
  prendi();
  delay(1000);
  rot0();
  polso90();
  DtreH();
  delay(1000);
  avanti2();
  delay(1000);
  pinzaA();
  delay(1000);
  indietro2();
  delay(1000);
  Dtre();
```

```
   delay(1000);
   avanti();
   indietro2();
   start();
   polso0();
   rot0();
  }
}

void seq1()       //A2
{
 start();
 delay(1000);
 polso90();
 Adue();
 pinzaA();
 rot90();
 delay(1000);
 avanti();
 pinzaC();
 delay(1000);
 indietro();
 pinzaA();
 delay(1000);
 indietro2();
 delay(1000);
 start();
 polso0();
 rot0();
}

void seq2()    //B3
{
  start();
 delay(1000);
 polso90();
 Btre();
 pinzaA();
 rot90();
 delay(1000);
 avanti();
 pinzaC();
 delay(1000);
 indietro();
```

```
 pinzaA();
 delay(1000);
 indietro2();
 delay(1000);
 start();
 polso0();
 rot0();
}

void seq3()   //D2
{
 start();
 delay(1000);
 polso90();
 Ddue();
 pinzaA();
 rot90();
 delay(1000);
 avanti();
 pinzaC();
 delay(1000);
 indietro();
 pinzaA();
 delay(1000);
 indietro2();
 delay(1000);
 start();
 polso0();
 rot0();
}

void seq4()    //B1
{
 start();
 delay(1000);
 polso90();
 Buno();
 pinzaA();
 rot90();
 delay(1000);
 avanti();
 pinzaC();
 delay(1000);
 indietro();
```

```
 pinzaA();
 delay(1000);
 indietro2();
 delay(1000);
 start();
 polso0();
 rot0();

}

void seq5()    //A1
{
 start();
 delay(1000);
 polso90();
 Auno();
 pinzaA();
 rot90();
 delay(1000);
 avanti();
 pinzaC();
 delay(1000);
 indietro();
 pinzaA();
 delay(1000);
 indietro2();
 delay(1000);
 start();
 polso0();
 rot0();
}

void seq6()       //A3
{
 start();
 delay(1000);
 polso90();
 Atre();
 pinzaA();
 rot90();
 delay(1000);
 avanti();
 pinzaC();
 delay(1000);
```

```
 indietro();
 pinzaA();
 delay(1000);
 indietro2();
 delay(1000);
 start();
 polso0();
 rot0();
}

void seq7()       //B2
{
 start();
 delay(1000);
 polso90();
 Bdue();
 pinzaA();
 rot90();
 delay(1000);
 avanti();
 pinzaC();
 delay(1000);
 indietro();
 pinzaA();
 delay(1000);
 indietro2();
 delay(1000);
 start();
 polso0();
 rot0();
}

void seq8()       //C2
{
 start();
 delay(1000);
 polso90();
 Cdue();
 pinzaA();
 rot90();
 delay(1000);
 avanti();
 pinzaC();
 delay(1000);
```

```
 indietro();
 pinzaA();
 delay(1000);
 indietro2();
 delay(1000);
 start();
 polso0();
 rot0();
}

void seq9()      //C3
{
 start();
 delay(1000);
 polso90();
 Ctre();
 pinzaA();
 rot90();
 delay(1000);
 avanti();
 pinzaC();
 delay(1000);
 indietro();
 pinzaA();
 delay(1000);
 indietro2();
 delay(1000);
 start();
 polso0();
 rot0();
}

void seq10()     //D1
{
 start();
 delay(1000);
 polso90();
 Duno();
 pinzaA();
 rot90();
 delay(1000);
 avanti();
 pinzaC();
 delay(1000);
```

```
 indietro();
 pinzaA();
 delay(1000);
 indietro2();
 delay(1000);
 start();
 polso0();
  rot0();
}

void seq11()     //D3
{
 start();
 delay(1000);
 polso90();
 Dtre();
 pinzaA();
 rot90();
 delay(1000);
 avanti();
 pinzaC();
 delay(1000);
 indietro();
 pinzaA();
 delay(1000);
 indietro2();
 delay(1000);
 start();
 polso0();
 rot0();
}

void prendi()    //movimenti per prendere il prodotto da aggiungere al magazzino
{
 addProd();
 pinzaA();
 rot45();
 delay(1000);
 down();
 pinzaC();
 delay(1000);
 up();
 start();
}
```

```
//calcoli degli impulsi
void calcoli()
    {
      r = sqrt((x * x) + ((y + distBase) * (y + distBase)));
      d = sqrt((r * r) + (h * h));

      gamma1Rad = atan(h / r);
      gamma1Gra = (gamma1Rad * 57.3);

      //calcolo dell'angolo gamma
      cosGamma = (-(L1 * L1) + (d * d) + (L2 * L2)) / (2 * d * L2);
      arcocosGamma = acos(cosGamma);
      gammaGra = (arcocosGamma * 57.3);

      //calcolo dell'angolo beta
      cosBeta = (-(d * d) + (L1 * L1) + (L2 * L2)) / (2 * L1 * L2);
      arcocosBeta = acos(cosBeta);
      betaGra = (arcocosBeta * 57.3);

      gammaTot = gammaGra + gamma1Gra;
      alfaTot = alfa2 + 267 - gammaTot - betaGra;

      //calcolo della base
      tgFi = x / (y + distBase);
      fiRad = atan(tgFi);
      fiGra = (57.3 * fiRad);
      impFi = mBase * fiGra + qBase;
      Base = "#0P" + String(impFi, 0) + "S500";

      impAlfa = mPolso * alfaTot + qPolso;
      Polso = "#3P" + String(impAlfa, 0) + "S500";

      impBeta = mGomito * betaGra + qGomito;
      Gomito = "#2P" + String(impBeta,0) + "S500";

      impGamma = mSpalla * gammaTot + qSpalla;
      Spalla = "#1P" + String(impGamma,0) + "S500";
    }

     void comandi1()              //funzione per mandare i dati alla SSC-32U
    {
      String comando;
```

```
    comando = Base + Spalla + Gomito + Polso + rot;

    Braccio.println(comando);
    delay(1000);

}

 void pinzaA()        //pinza aperta
{
    Pinza = "#4P750T500";
    Braccio.println(Pinza);
}

void pinzaC()        //pinza chiusa
{
    Pinza = "#4P1800T500";
    Braccio.println(Pinza);
}

void indietro()      //apre il cassetto
{
  y = 4;

  calcoli();
  comandi1();
}

void indietro2()     //allontanamento dal cassetto aperto
{
   y = -1;

  calcoli();
  comandi1();
}

void addProd()    //coordinate del prodotto da aggiungere
{
   x = 14;
   y = 9;
   h = 14;

  calcoli();
  comandi1();
}
```

```
//coordinate dei cassetti

void Auno()
{
  x = -4;
  y = 3;
  h = 16;
  rot = "#5P650T500";

  calcoli();
  comandi1();
}

void AunoH()
{
  x = -4;
  y = 0;
  h = 25;


  calcoli();
  comandi1();
}

void Adue()
{
  x = 0;
  y = 2;
  h = 16;
  rot = "#5P650T500";

  calcoli();
  comandi1();
}

void AdueH()
{
  x = 0;
  y = -1;
  h = 25;

  calcoli();
  comandi1();
```

```
}

void Atre()
{
  x = 4;
  y = 2;
  h = 16;
  rot = "#5P650T500";

  calcoli();
  comandi1();
}

void AtreH()
{
  x = 4;
  y = -1;
  h = 25;

  calcoli();
  comandi1();
}

 void Buno()
{
  x = -4;
  y = 2;
  h = 9;
   rot = "#5P650T500";

  calcoli();
  comandi1();
}

  void BunoH()
{
  x = -4;
  y = 2;
  h = 17;

  calcoli();
  comandi1();
}
```

```
void Bdue()
{
  x = 0.5;
  y = 2;
  h = 9;
   rot = "#5P650T500";

  calcoli();
  comandi1();
}

void BdueH()
{
  x = 0.5;
  y = 2;
  h = 17;

  calcoli();
  comandi1();
}

void Btre()
{
  x = 4;
  y = 2;
  h = 9;
  rot = "#5P650T500";

  calcoli();
  comandi1();
}

void BtreH()
{
  x = 4;
  y = 2;
  h = 17;

  calcoli();
  comandi1();
}

void Cdue()
{
```

```
  x = 0.5;
  y = 2;
  h = 3;
  rot = "#5P650T500";

  calcoli();
  comandi1();
}


void CdueH()
{
  x = 0.5;
  y = 2;
  h = 11;


  calcoli();
  comandi1();
}

void Ctre()
{
  x = 4.5;
  y = 2;
  h = 3;
  rot = "#5P650T500";

  calcoli();
  comandi1();
}

void CtreH()
{
  x = 4.5;
  y = 2;
  h = 11;


  calcoli();
  comandi1();
}

void Duno()
```

```
{
  x = -3;
  y = 2;
  h = -3;
  rot = "#5P650T500";

  calcoli();
  comandi1();
}

 void DunoH()
{
  x = -3;
  y = 2;
  h = 4;

  calcoli();
  comandi1();
}


void Ddue()
{
  x = 1;
  y = 2;
  h = -3;
   rot = "#5P650T500";

  calcoli();
  comandi1();
}

void DdueH()
{
  x = 1;
  y = 2;
  h = 4;

  calcoli();
  comandi1();
}

 void Dtre()
{
```

```
  x = 4.5;
  y = 2;
  h = -3;
  rot = "#5P650T500";

  calcoli();
  comandi1();
}

 void DtreH()
{
  x = 4.5;
  y = 2;
  h = 4;


  calcoli();
  comandi1();
}

void polso0()       //polso a 0°
{
  alfa2 = 5;

  calcoli();
  comandi1();
}

void rot0()        //rotazione del polso a 0°
{
  rot = "#5P1500T500";
   calcoli();
  comandi1();

}

void rot45()        //rotazione del polso a 45° (utilizzato per afferrare gli oggetti da aggiungere)
{
  rot = "#5P1800T500";
   calcoli();
  comandi1();

}
```

```
 void rot90()    //rotazione del polso a 90°
{
 rot = "#5P650T500";

}

void avanti()     //avvicina il braccio ai cassetti e li chiude
{
 y = 11;

 calcoli();
 comandi1();
}

void avanti2()    //avvicinamento ai cassetti per rilasciare il prodotto
{
 y = 8;

 calcoli();
 comandi1();
}

 void polso90()     //polso a 90°
{
 alfa2 = 90;
 calcoli();
 comandi1();
}

 void up()       //alzamento del braccio
{
   h = 10;

   calcoli();
   comandi1();

}

void down()     //abbassamento del braccio
{
   h = 3;

   calcoli();
   comandi1();
```

```
}

void start()     //posizione iniziale e finale
{
   x = 0;
   y = -1;

   h = 10;

   calcoli();
   comandi1();

}
```

Questo è tutto il programma di Arduino: l'applicazione manda una lettera (mandata dall'applicazione) e, quando Arduino la riceve, in base alla tipologia di carattere ci sono vari comandi. Ogni lettera rappresenta una sequenza di comandi, la quale è formata da sotto funzioni di diversi comandi. Si possono differenziale le sotto funzioni in diversi gruppi:

- Calcoli degli impulsi
- Invio degli impulsi
- Coordinate dei cassetti (per esempio "Adue")
- Coordinate di posizioni sopra i cassetti (per esempio "AdueH")
- Spostamenti in avanti, indietro, in basso e in alto
- Posizione iniziale
- Apertura e chiusura della pinza
- Rotazione del polso

Alla fine, la SSC-32U riceve un'unica stringa formata dalle stringhe di ogni servomotore. La struttura è la seguente: **#0P…S…#1P…#2…#3…#4…#5…**

Ho disposto le coordinate (misurate con la riga) in base alla tabella seguente:

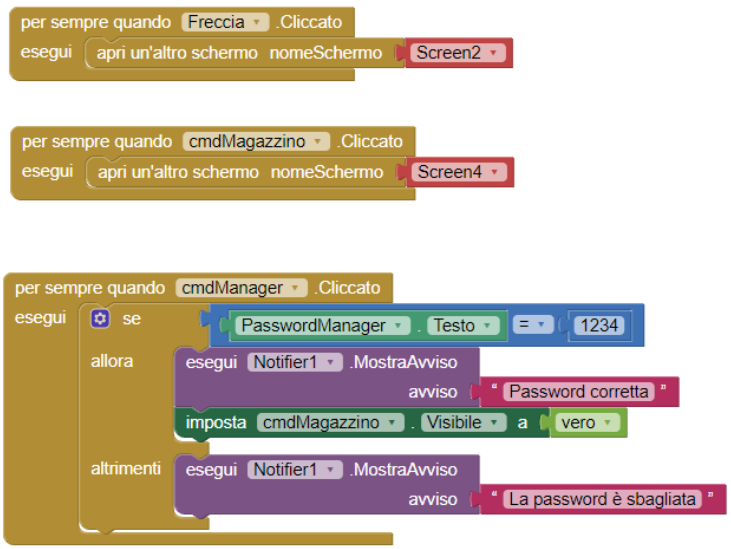
A1	A2	A3
B1	B2	A3
C1	C2	C3
D1	D2	D3

6) Creazione dell'applicazione

Per la creazione dell'applicazione ho utilizzato MIT App Inventor: un sito web per creare applicazioni per Android gratuitamente. Se si vuole avere l'app nel proprio dispositivo, basta scansionare il QR code: questo porta al drive per poi scaricarla.

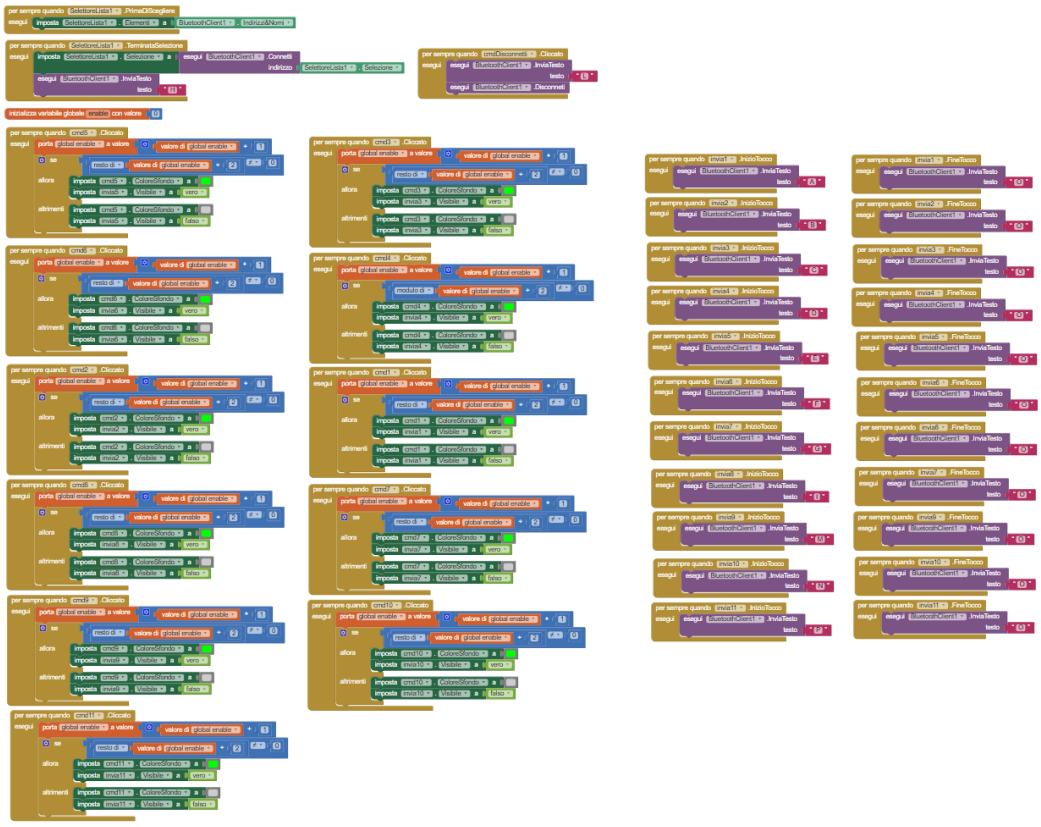
L'applicazione è divisa su 3 schermi: la schermata iniziale, la schermata per il cliente e la schermata per il manager

Schermata iniziale



Appena si accede nell'applicazione si trova questa schermata divisa in due parti: se si è un cliente bisogna cliccare sulla freccia per andare nella schermata successiva; mentre se ad entrare è il manager (perché deve aggiungere prodotti nel magazzino) allora deve inserire una password per entrare nella sua schermata.

Schermata del cliente



In questa seconda schermata il cliente chiede i prodotti che vuole: per prima cosa si connette al Bluetooth del magazzino e seleziona i prodotti che gli servono. Ogni volta che clicca su un pulsante di un prodotto, questo si colora di verde e appare un tasto per inviare la richiesta. Questi pulsanti mandato un dato diverso ad Arduino (una lettera). Al termine si deve disconnettere dal Bluetooth.

Schermata del manager

In questa schermata il manager aggiunge i prodotti al magazzino. Come la schermata del cliente, si deve connettere al Bluetooth del magazzino e selezionare il prodotto. Dopo la selezione clicca il tasto sotto per inviare il dato al Arduino. Al termine si deve sempre disconnettere al Bluetooth.

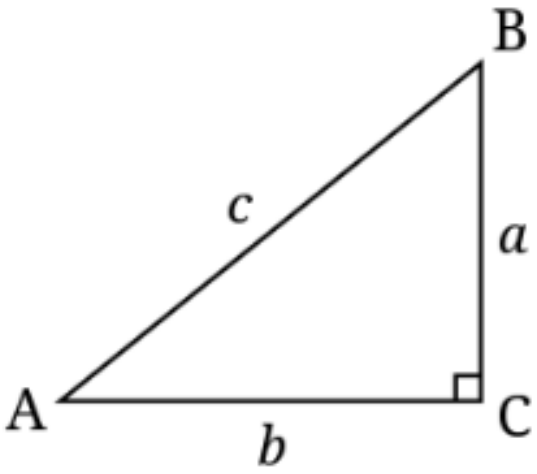
L'applicazione, come già detto all'inizio, è di facile utilizzo perché ogni schermata è dotata di istruzioni.

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”: consiste nel determinare le coordinate di un punto, corrispondenti ad una data posizione e ad un dato orientamento del manipolatore. Per raggiungere questo obiettivo mi ha aiutato la matematica. Ripassiamo un po' di goniometria e trigonometria….

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.



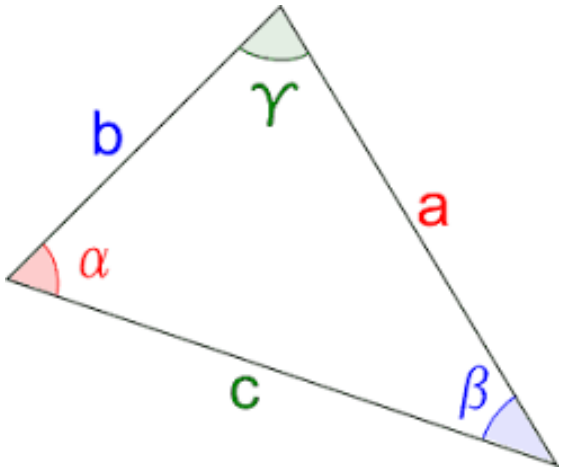
$$c^2 = a^2 + b^2$$

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

$$c = \sqrt{a^2 + b^2}$$

Teorema di Carnot (o teorema del coseno):

Il teorema afferma che 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).

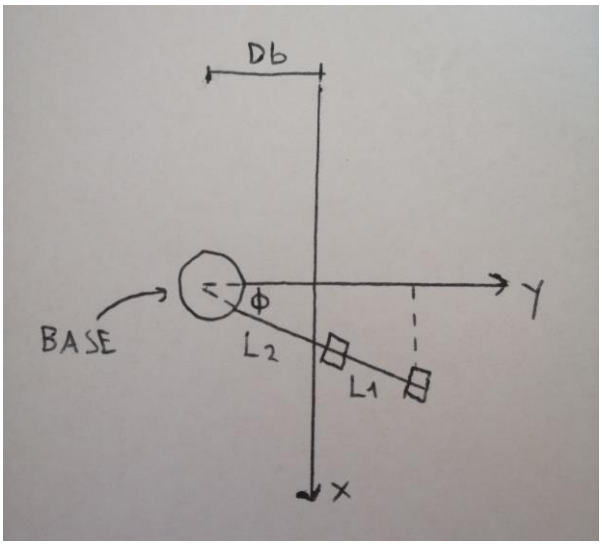


$$ac^2 = ab^2 + bc^2 - 2 * \text{ab} * \text{bc} * \cos(\beta)$$

Questi teoremi ci torneranno utili per i calcoli degli impulsi da fornire ai servomotori per muoversi nelle coordinate date.

Grazie ad Excel mi sono ricavato una parte del calcolo degli impulsi (m e q), invece con la matematica mi posso ricavare l'angolo di rotazione partendo dalle coordinate.

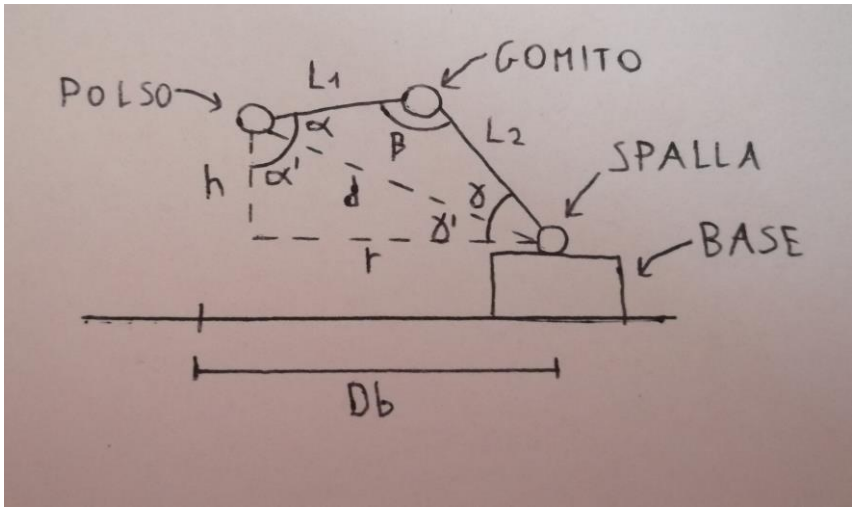
Vediamo quali sono le basi trigonometriche del braccio robot:



Nell'immagine di sopra viene rappresentato dall'alto il braccio e dei valori:

- Db è la distanza tra il giunto di base e l'asse x (nel mio progetto ho scelto una distanza di 12 cm)
- L2 è il link tra la spalla e il gomito, mentre L1 è il link tra il gomito e il polso
- X e y sono gli assi cartesiani
- φ è l'angolo che il braccio forma distanziandosi dall'asse y

Vediamo ora il braccio visto lateralmente:



Come si può vedere, vengono rappresentati 2 triangoli perché dobbiamo anche impostare l'altezza dei cassetti.

- H è l'altezza e rappresenta la distanza tra il polso e la base di legno
- r è il cateto del triangolo più in basso che si crea
- d è l'ipotenusa di entrambi i lati e rappresenta anche la distanza tra il polso e la base
- α, β e γ sono gli angoli che si creano per i vari giunti (α è l'angolo della base, β è quello del gomito e γ è quello del polso). I due triangoli dividono in due parti gli angoli totali.

Ora che abbiamo tutti i dati che ci servono, 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 φ vale

$$\tan\varphi = \frac{x}{y + Db}$$

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

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

Da cui si ricava

$$\varphi(in\ gradi) = \frac{180 *\ \varphi(in\ radianti)}{\pi}$$

Si può approssimare il calcolo come

$$\varphi(in\ gradi) = 57.3 *\ \varphi(in\ radianti)$$

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 (seconda immagine):

$$r = \sqrt{x^2 + (y + Db)^2}$$

$$d = \sqrt{r^2 + h^2}$$

Spalla

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

$$\cos(\gamma) = \frac{L2^2 + d^2 - L1^2}{2 * d * L2}$$

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 γ. A questo punto ricaviamo anche γ^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 β, si utilizza la formula inversa del teorema di Carnot

$$\cos(\beta) = \frac{L2^2 + L1^2 - d^2}{2 * L1 * L2}$$

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 (α) 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°, a 45° e a 90°:

- 0° è la posizione standard
- 90° per aprire i cassetti e chiuderli
- 45° per prendere il prodotto che deve essere aggiunto al magazzino

CONCLUSIONI

Il magazzino automatico è stato realizzato senza grossi problemi; alcuni piccoli ne ho trovati, ma sono sempre riuscito a risolverli. Mi ritengo molto soddisfatto di quello che ho fatto perché il risultato finale ha rappresentato le mie aspettative, una piccola idea è diventata un progetto di alto livello. Inoltre sono riuscito a cavarmela da solo e il magazzino è stato apprezzato da parenti, professori, studenti e amici.

Questo potrebbe essere uno spunto per progetti futuri e per chi volesse costruire un magazzino ben organizzato.

In particolare devo ringraziare i miei professori e i miei genitori: i professori perché è grazie ai loro insegnamenti che sono riuscito a realizzarlo. I miei genitori, invece, mi hanno sempre appoggiato e aiutato per qualsiasi cosa.

Ma non solo: mi ha dato anche una mano il libro “L@borobotica volume B”. È un libro scritto da due professori di robotica (Lorenzo Arco e Giuseppe Peretti), contenente moltissime informazioni utili per progettare e costruire robot. Consiglio altamente di procurarsene una copia.

Con il mio progetto ho concluso bene il mio percorso scolastico. In alcuni momenti è stato difficile, ma avrò sempre bellissimi ricordi di questi 5 anni passati all’ITI Omar di Novara.

BIBLIOGRAFIA E SITOGRAFIA

Bibliografia:

“L@borobotica volume B, manipolatori e IoT”

Sitografia:

https://www.lucidchart.com/

http://ai2.appinventor.mit.edu/