UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA

" PROGETTAZIONE E REALIZZAZIONE DI UNA MANO ROBOTICA CON ARDUINO "

Relatore: Prof. Meneghini Matteo

Laureanda: Fracaro Lucrezia

ANNO ACCADEMICO 2022 – 2023

Data di laurea 21 Novembre 2023

Abstract

Questa tesi si pone come obiettivo la progettazione e la realizzazione di una mano robotica. Il focus del progetto verrà dato al movimento delle dita, e quindi alla comunicazione tra servomotori e sensori di flessione.

L'elaborato seguirà una struttura progressiva, iniziando con una breve introduzione sull'utilizzo delle mani robotiche, seguita dall'enumerazione degli obiettivi del progetto e dalla descrizione dettagliata dei componenti impiegati.

Successivamente verrà illustrata la procedura di realizzazione della base e dei collegamenti, accompagnata dal listato del codice sorgente del programma. Nel codice saranno presenti dei commenti esplicativi per garantire una comprensione approfondita del funzionamento.

Nella fase conclusiva, sarà dimostrato il funzionamento del progetto, evidenziando eventuali problematiche riscontrate durante la fase di realizzazione e suggerendo possibili miglioramenti.

Questo approccio permetterà di seguire passo dopo passo il processo di realizzazione della mano robotica, facilitando la comprensione e offrendo spunti per eventuali sviluppi futuri.

Indice

1. Introduzione

Negli ultimi decenni, sono stati sviluppati numerosi prototipi di mano robotiche multi-dita con l'obiettivo di replicare la destrezza e le abilità della mano umana. Queste innovazioni trovano applicazione in vari settori, tra cui la produzione industriale, la chirurgia robotica, l'esplorazione spaziale, lo sviluppo di protesi cibernetiche e la ricerca in robotica umanoide. Ciò amplia significativamente le possibilità di interazione tra macchine e ambiente circostante.

Le prime mani robotiche erano spesso rudimentali e limitate nella loro capacità di manipolazione. Con il passare del tempo, si sono verificati miglioramenti significativi grazie allo sviluppo di nuovi materiali e all'introduzione di sensori più sofisticati, tanto da consentire alle mani robotiche di interagire in modo più sensibile con l'ambiente circostante.

Oggi, grazie all'evolversi dell'intelligenza artificiale e all'apprendimento automatico, le mani robotiche sono in grado di compiere una vasta gamma di movimenti e di adattarsi alle situazioni più complesse.

La sfida attuale è quella di sviluppare mani robotiche sempre più abili nell'imitare la precisione e la versatilità delle mani umane.

Il lavoro di tesi consiste nella progettazione e nello sviluppo di una mano robotica antropomorfa che sia in grado di simulare il movimento di una mano.

L'idea alla base di questo progetto è la realizzazione di un dispositivo in grado di consentire il controllo di un robot, in particolare attraverso l'implementazione di una mano robotica capace di replicare i movimenti della mano umana. Tale funzionalità è utile in scenari in cui la manipolazione di oggetti è richiesta in ambienti pericolosi o di difficile accesso, consentendo un controllo remoto sicuro ed efficace.

Per rispettare limiti di tempo e di budget, il focus è stato posto sul movimento delle dita. Questo costituisce un punto di partenza solido per un progetto più ampio e articolato.

Un possibile passo successivo potrebbe riguardare l'implementazione di una comunicazione wireless tra la mano robotica e quella umana. In alternativa, si potrebbe intraprendere uno studio più approfondito sul movimento delle dita, estendendo le funzionalità oltre la semplice apertura e chiusura, includendo anche movimenti laterali a destra e sinistra.

L'obiettivo di questo progetto è abilitare il controllo preciso del movimento delle dita robotiche in risposta a segnali derivanti dal movimento corrispondente delle dita umane.

Questi segnali vengono catturati e interpretati mediante l'uso di servomotori, consentendo così una replicazione accurata dei movimenti della mano umana attraverso la mano robotica.
La focalizzazione del progetto si è concentrata sull'implementazione di collegamenti efficaci tra i sensori e lo sviluppo di un codice che rendesse la mano operativa, piuttosto che sulla creazione di una mano funzionale specificamente dedicata alla presa di oggetti.
Nel caso in cui si fosse voluto realizzare una mano capace di compiere operazioni di presa degli oggetti, sarebbe stato necessario progettare una struttura più complessa che consentisse anche il movimento del palmo.
Per realizzare il progetto sono stati seguiti i seguenti passaggi:

- La ricerca di componenti adeguati e lo studio delle loro caratteristiche
- La progettazione e il montaggio del circuito
- La stesura del codice

2. Elenco dei componenti

2.1 Servo Motor SG90

Il servomotore è un dispositivo elettrico o elettronico utilizzato per controllare con precisione la posizione, la velocità e l'accelerazione di un oggetto o di un meccanismo.
Il servomotore SG90 è un tipo di servomotore che può effettuare rotazioni di 180°. Viene controllato dalla scheda Arduino Uno tramite una serie di impulsi, i quali servono ad indicare la posizione in cui il motore deve ruotare.
È composto da tre cavi:

- Il cavo di messa a terra, di color marrone (o nero), che deve essere collegato alla porta GND
- Il cavo di alimentazione, connesso alla porta dei 5V, indicato con il colore rosso
- Il cavo di segnale, tipicamente arancione, che viene connesso a uno dei pin digitali della scheda.

Per poterlo utilizzare è necessaria la libreria <Servo> , la quale permette alla scheda Arduino di controllare una molteplicità di servomotori.

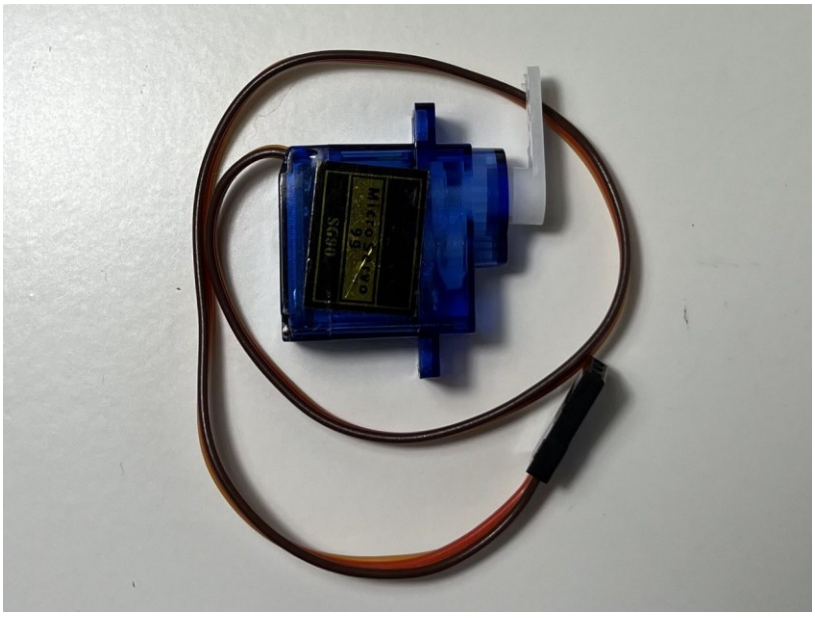


Fig. 1 Servomotore SG90

2.2 Sensore di flessione

Il sensore di flessione è un dispositivo progettato per rilevare e misurare la deformazione di un oggetto o superficie.

È composto da un sottile materiale sensibile alla flessione. Quando il sensore è soggetto ad una forza o una pressione, il materiale sensibile subisce una deformazione che si riflette in una variazione della sua resistenza elettrica.

Il funzionamento del sensore è quindi analogo a quello di una resistenza variabile.

Questo sensore è caratterizzato da una bassa tensione di funzionamento, che varia tra gli 0V e i 5V.

Quando il sensore è piatto presenta una resistenza di 25 kΩ . Durante la flessione la sua resistenza può variare tra i 45kΩ e i 125 kΩ, con una tolleranza di ±30%.

Fig. 2 Sensore di flessione

2.3 Scheda Arduino Uno

Arduino è una scheda programmabile dotata di un microcontrollore e di un ambiente di sviluppo software (IDE).

È dotata di un microcontrollore centrale e di alcuni componenti elettrici integrati che facilitano il collegamento con i dispositivi esterni.

La scheda Arduino Uno è basata su un microcontrollore ATmega328. Dispone di 14 pin digitali programmabili, che possono agire come ingressi e uscite. Alcuni di essi possiedono la capacità di essere utilizzati per funzioni dedicate, come la generazione di segnale PWM. Inoltre, sono presenti 6 pin d'ingresso dedicati all'acquisizione e all'elaborazione di segnali analogici. La scheda dispone anche di 5 pin di potenza, di cui due sono collegati a massa (GND), mentre i restanti 3 vengono utilizzati per l'alimentazione (3.3 V, 5V e uno per l'alimentazione esterna).

Il microcontrollore Atmega328 ha una memoria flash di 32KB, dove 5KB sono utilizzati per il "boot loader".

La scheda è inoltre caratterizzata da 2KB di SRAM e 1KB di EEPROM.

Tipo Microcontrollore	Atmel ATmega328
Tensione di lavoro	5Vdc
Tensione di alimentazione consigliata	da 7Vdc a 12Vdc
Pin digitali	14 configurabili come ingressi o uscite
Pin analogici	6 ingressi
Massima corrente per pin digitale	40mA massima
Memoria Flash	32KB
Memoria Sram	2KB
Memoria EEPROM	1KB
Velocità di clock del microcontrollore	16MHz

Fig. 3 Datasheet della scheda Arduino Uno

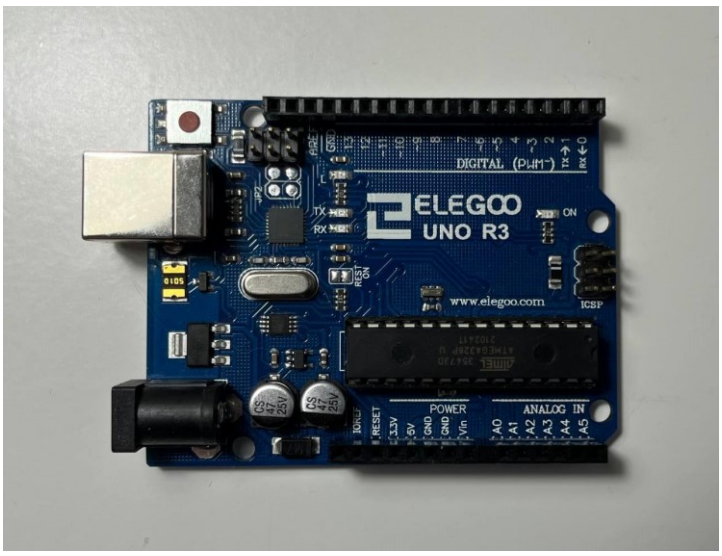


Fig. 4 Scheda Elegoo Uno, equivalente alla scheda Arduino Uno

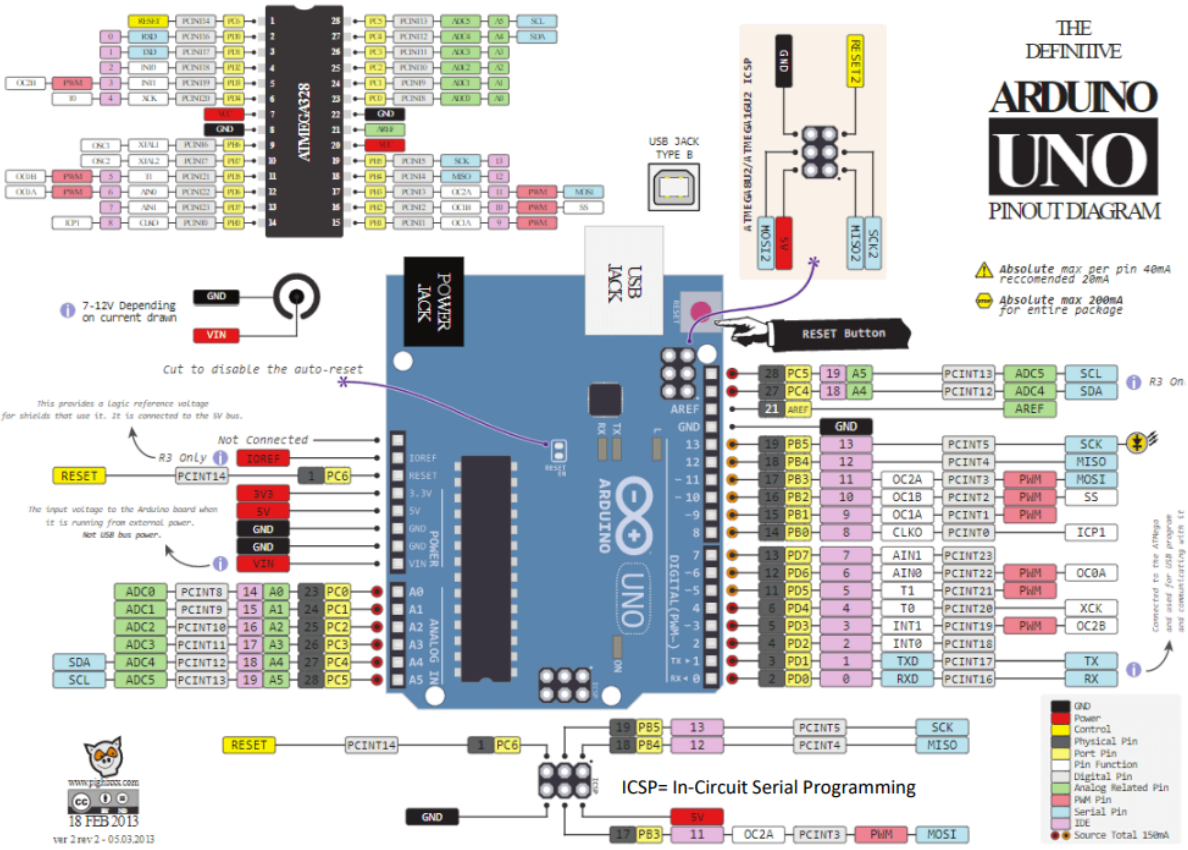


Fig. 5 Piedinatura della scheda

2.4 Breadboard

La breadboard è uno strumento utilizzato per la prototipazione di circuiti elettrici. La sua principale caratteristica è la facilità con cui consente di realizzare le connessioni senza la necessità di saldature, rendendola un dispositivo versatile e riutilizzabile.

La struttura della breadboard è costituita da una "zona del circuito", che comprende 63 colonne verticali nella parte superiore e 63 colonne verticali nella parte inferiore. In ogni colonna sono presenti cinque fori interconnessi tra loro attraverso delle barre metalliche. Questo permette di inserire componenti elettronici come resistenze, condensatori e semiconduttori nei fori, creando connessioni elettriche temporanee. Inoltre, è presente una "zona di alimentazione". Quest'area è composta da 4 "rotaie" orizzontali, due nella parte superiore e due nella parte inferiore. Queste "rotaie" permettono di fornire tensione elettrica ai componenti del circuito e di stabilire il collegamento a massa.

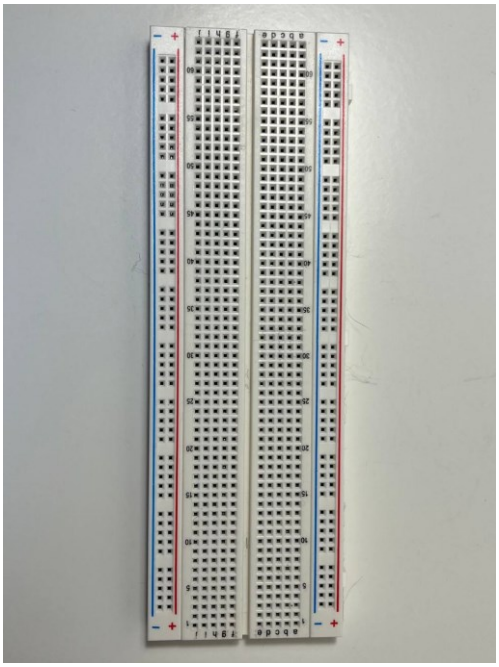


Fig. 6 Breadboard

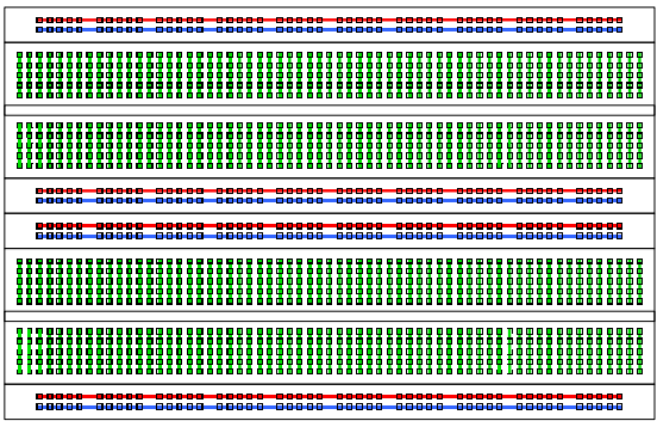


Fig. 7 Schema dei collegamenti interni alla breadboard

2.5 Cavi

Per collegare la breadboard con la scheda Arduino sono stati utilizzati dei cavi maschio-maschio. Questi cavi forniscono connessioni stabili e flessibili, consentendo un'installazione rapida e un facile reperimento dei collegamenti.
Per collegare i sensori di flessione con la breadboard sono stati utilizzati dei cavi classici successivamente stagnati ai sensori. Il rinforzo attraverso la stagnatura può migliorare la robustezza e la durabilità delle connessioni, proteggendo i cavi da eventuali danni meccanici o ambientali.

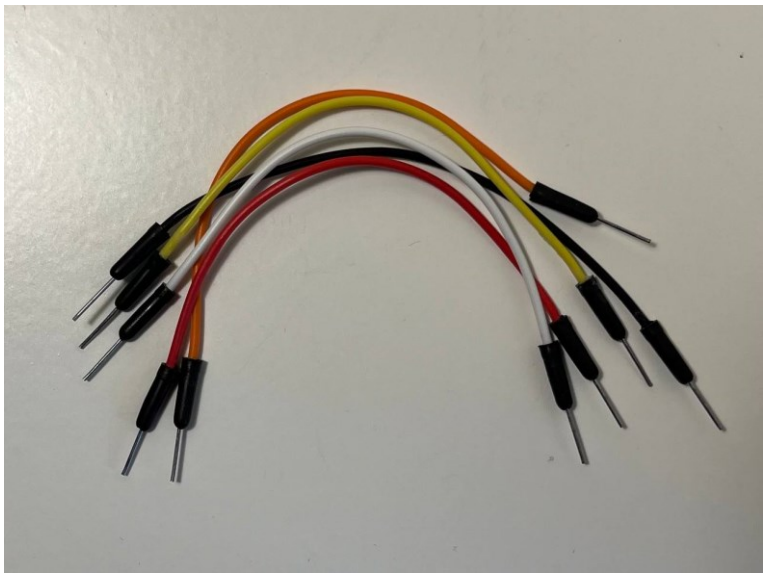


Fig. 8 Cavi maschio-maschio

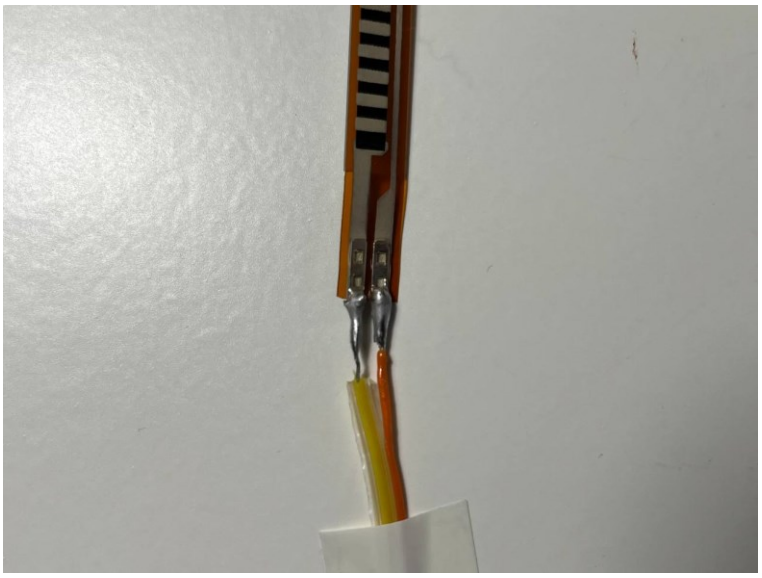


Fig. 9 Cavi saldati con il sensore di flessione

2.6 Resistenze

La resistenza è un componente passivo che limita il flusso di corrente elettrica in un circuito. La sua principale funzione è quella di opporsi al passaggio della corrente, generando una caduta di tensione proporzionale alla corrente che l'attraversa. È misurata in ohm (Ω).
Le resistenze sono utilizzate per controllare la corrente di un circuito, regolare e dividere la tensione e per altre funzioni importanti per la progettazione elettronica.
Sono caratterizzate da un codice a colori per rappresentare il loro valore. Le prime due bande colorate rappresentano le prime due cifre del valore della resistenza, la terza banda indica il fattore di moltiplicazione a cui elevare il valore ottenuto dalle prime due bande e la quarta banda rappresenta la tolleranza della resistenza.

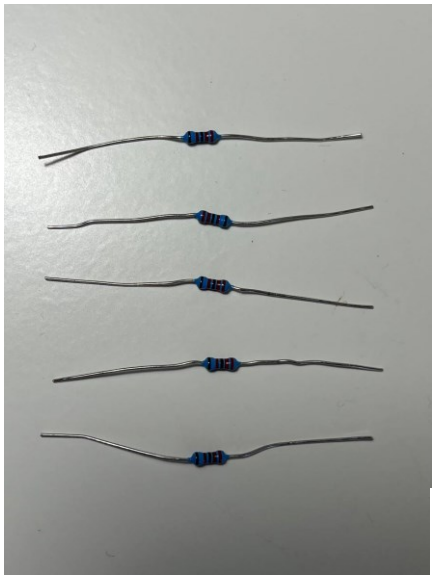
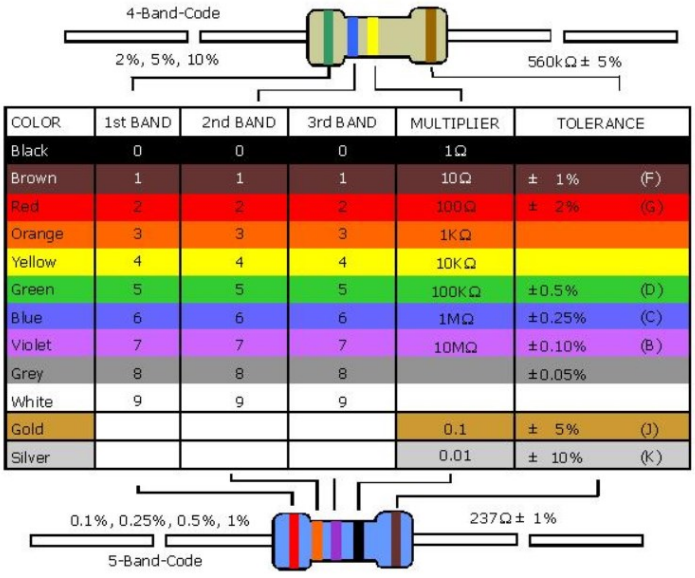


Fig. 10 Resistenze utilizzate



COLOR	1st BAND	2nd BAND	3rd BAND	MULTIPLIER	TOLERANCE	
Black	0	0	0	1Ω		
Brown	1	1	1	10Ω	± 1%	(F)
Red	2	2	2	100Ω	± 2%	(G)
Orange	3	3	3	1KΩ		
Yellow	4	4	4	10KΩ		
Green	5	5	5	100KΩ	±0.5%	(D)
Blue	6	6	6	1MΩ	±0.25%	(C)
Violet	7	7	7	10MΩ	±0.10%	(B)
Grey	8	8	8		±0.05%	
White	9	9	9			
Gold				0.1	± 5%	(J)
Silver				0.01	± 10%	(K)

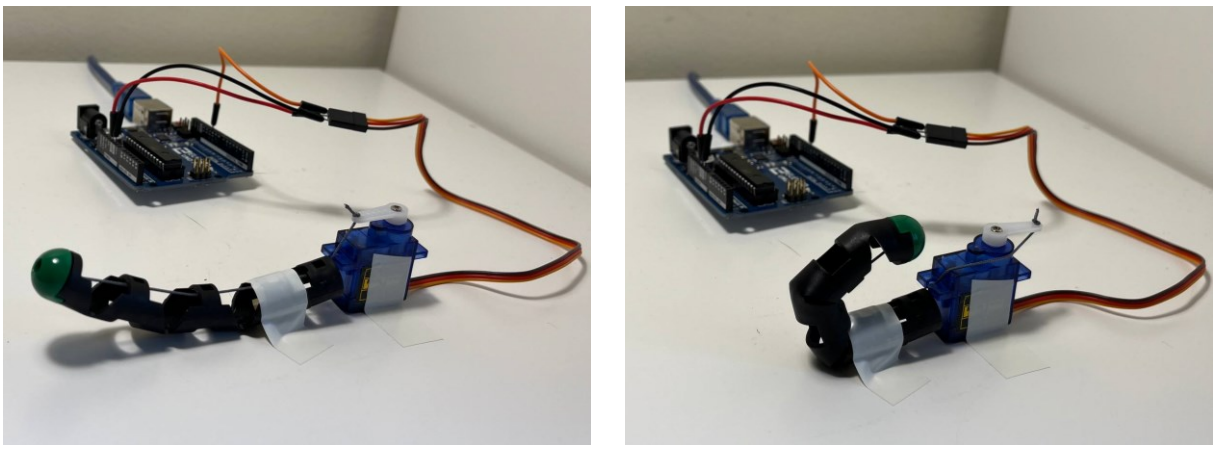
Fig. 11 Codice colori

3. Funzionamento

Per la realizzazione della mano robotica, si è cominciato creando una base. In questo caso la base è costituita dal palmo della mano e da una struttura per mantenerlo sollevato. Ciò è dovuto al fatto che le uniche parti mobili della mano robotica sono le dita.
Per la creazione del palmo è stato impiegato del cartone, in quanto è un materiale semplice da tagliare ma allo stesso tempo abbastanza resistente da riuscire a sostenere le dita e i servomotori.
Dopo aver delineato e ritagliato la sagoma del palmo sul cartone sono stati creati dei fori.

Fig. 12 Foto della base realizzata

Le dita sono fatte di plastica morbida, il che li consente di piegarsi e di ritornare facilmente alla forma originale.
Per permettere il movimento delle dita, viene impiegato un filo che connette la punta del dito al servomotore. Quindi quando il servomotore viene azionato, si genera una tensione sul filo, la quale permette al dito di piegarsi.
Si può osservare che in questo caso è sufficiente un servomotore in grado di effettuare rotazioni fino a 180°, poiché in questo intervallo si raggiuge la massima tensione del filo, consentendo così la massima piegatura del dito.



(a) Servomotore ruotato a 0° e dito non piegato **(b)** Servomotore ruotato a 180° e dito piegato

Fig. 13 Foto del collegamento tra il dito e il servomotore

Per permettere il funzionamento dei servomotori è fondamentale collegarli alla scheda Arduino, e ovviamente fornire loro l'alimentazione necessaria.

Come menzionato in precedenza, il servomotore è composto da tre cavi, uno per l'alimentazione (rosso), uno per la massa (marrone) e uno per il segnale (arancione). Si procede collegando il cavo rosso al pin di alimentazione (5V) , il cavo marrone al GND e il cavo arancione ad uno dei pin digitali.

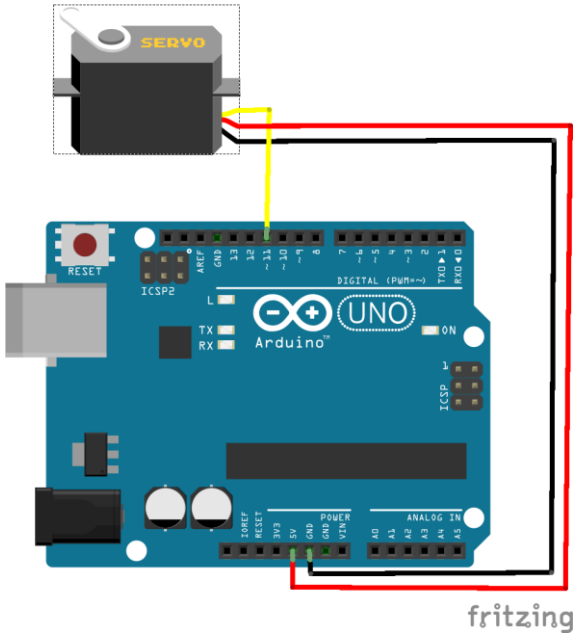


Fig. 14 Schema del collegamento tra la scheda Arduino e il servomotore

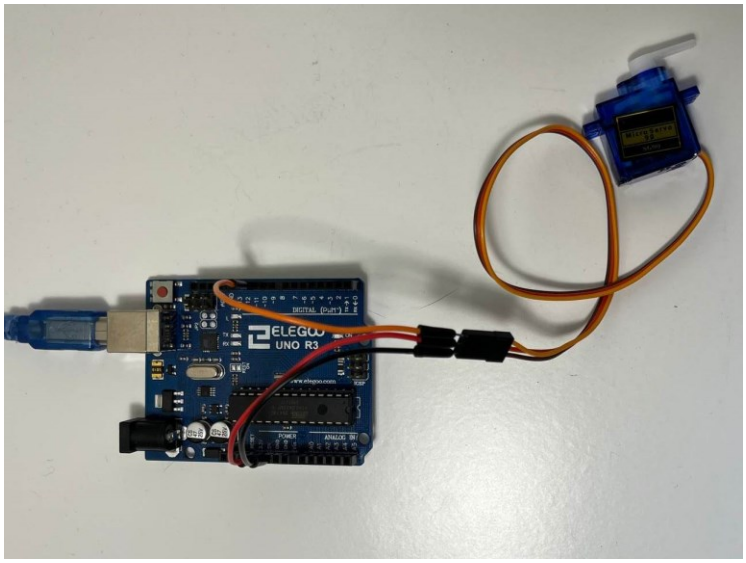


Fig. 15 Foto del collegamento tra la scheda Elegoo e il servomotore

Dovendo utilizzare cinque servomotori, si fa uso della breadboard. I pin di alimentazione e di massa della scheda Arduino vengono collegati attraverso dei cavi maschio-maschio ad una delle "rotaie" della breadboard. Questo consente l'alimentazione dell'intera "rotaia" e di alimentare contemporaneamente più servomotori.

Successivamente, i cavi di alimentazione e di massa dei servomotori vengono collegati alla breadboard, mentre i cavi per il segnale sono collegati ai pin digitali 5, 6, 9, 10, 11.

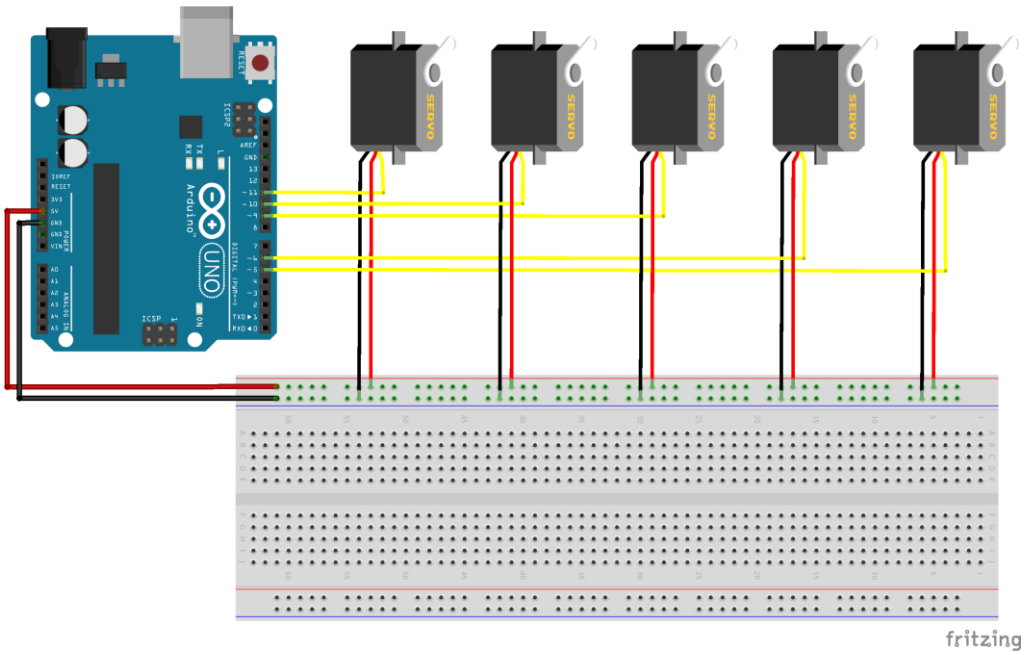


Fig. 16 Schema del collegamento tra la scheda Arduino e i cinque servomotori

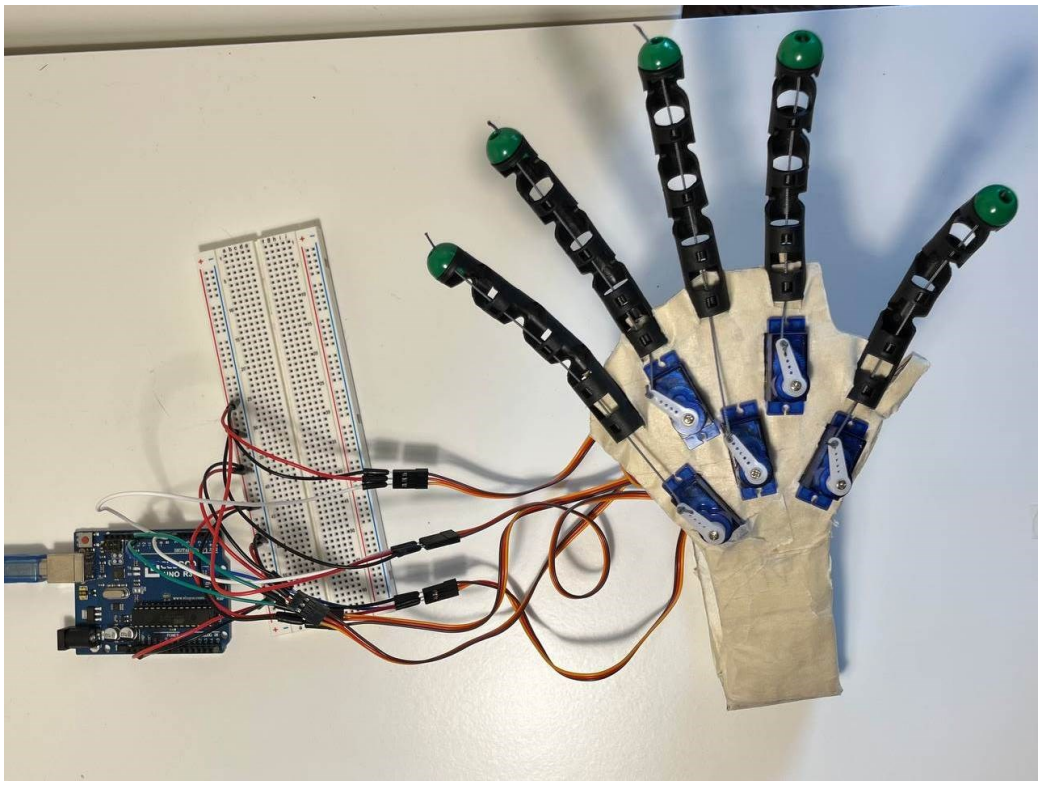
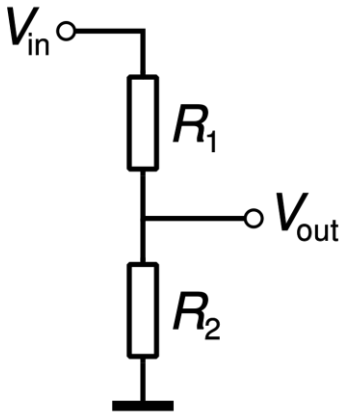


Fig. 17 Foto del collegamento tra la scheda Elegoo e i cinque servomotori

Per collegare un sensore di flessione alla scheda Arduino, bisogna considerare che, come si è menzionato in precedenza, questo sensore modifica la sua resistenza in base a quanto viene piegato. Per rilevarla correttamente è necessario realizzare un partitore di tensione.



$$V_{out} = V_{in} \frac{R_2}{R_1 + R_2}$$

Fig. 18 Partitore di tensione

Considerando la resistenza variabile R_1 come il sensore di flessione e la resistenza R_2 come la resistenza da inserire, è possibile calcolare il valore di quest'ultima.

Dai valori forniti nel datasheet del sensore di flessione, risulta necessario selezionare una resistenza con un valore compreso tra 10 kΩ e 100 kΩ.

Infatti, scegliendo una resistenza compresa in questo intervallo, si otterrà un guadagno inferiore ad uno, per qualsiasi valore che può assumere la resistenza variabile.

Sulla base delle resistenze disponibili, è stata selezionata una resistenza con un valore di 22 kΩ, la quale rispetta i parametri necessari.

Dopo aver scelto il valore della resistenza è possibile connettere il sensore di flessione alla breadboard e successivamente alla scheda Arduino.

I due piedini del sensore, così come quelli di una resistenza, non presentano una polarità. Di conseguenza, non è rilevante quale dei due piedini viene collegato all'alimentazione e quale a massa.

Pertanto, un piedino viene collegato direttamente al pin di alimentazione, mentre l'altro piedino è collegato alla resistenza. Quest'ultima, a sua volta, è collegata a massa da un lato ed al pin analogico dall'altro.

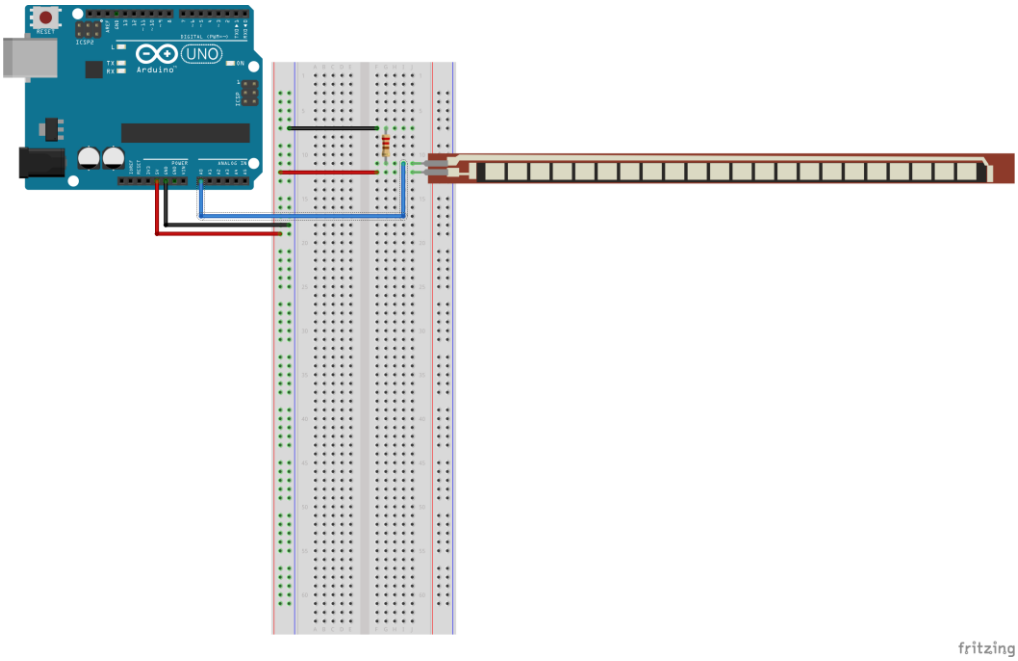


Fig. 19 Schema del collegamento tra la scheda Arduino e il sensore di flessione

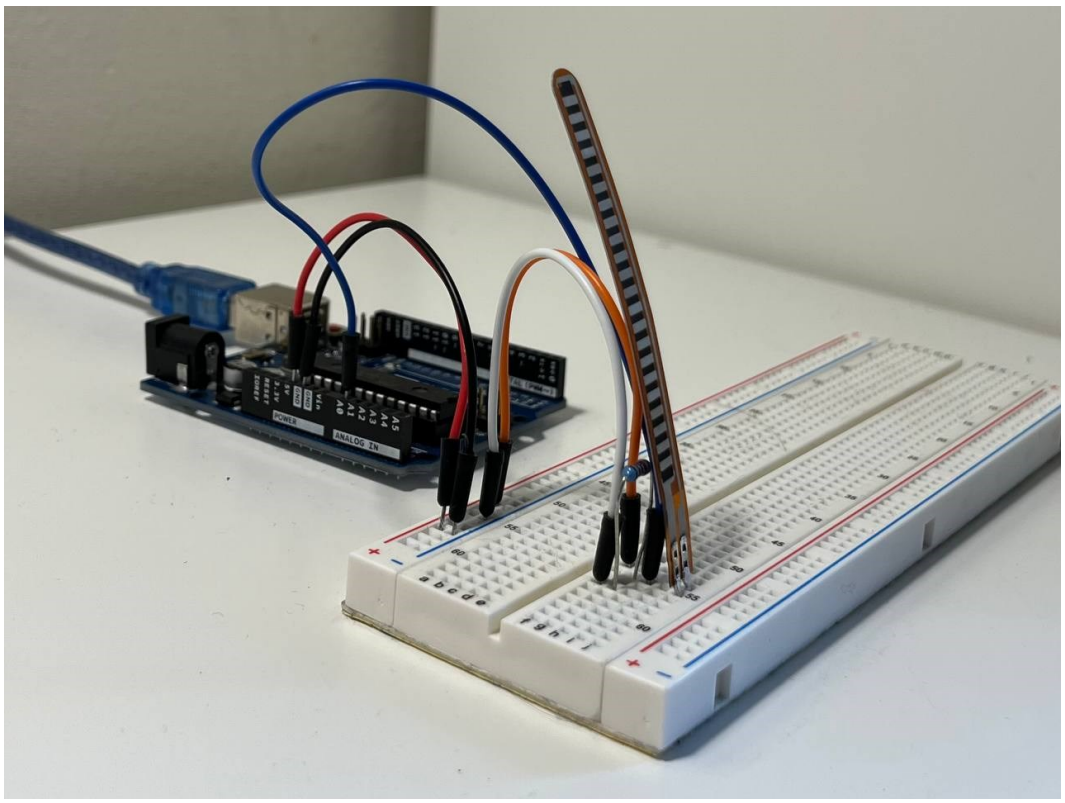


Fig. 20 Foto del collegamento tra la scheda Elegoo e il sensore di flessione

Analogamente con quanto affermato per i servomotori, è necessario collegare cinque sensori di flessione. Per collegare i sensori alla breadboard, i piedini sono stati saldati a dei cavi, e l'estremità opposta dei cavi è stata inserita nella breadboard.

La lunghezza dei cavi è stata scelta in modo tale da consentire il movimento della mano senza il rischio che si stacchino dalla breadboard.

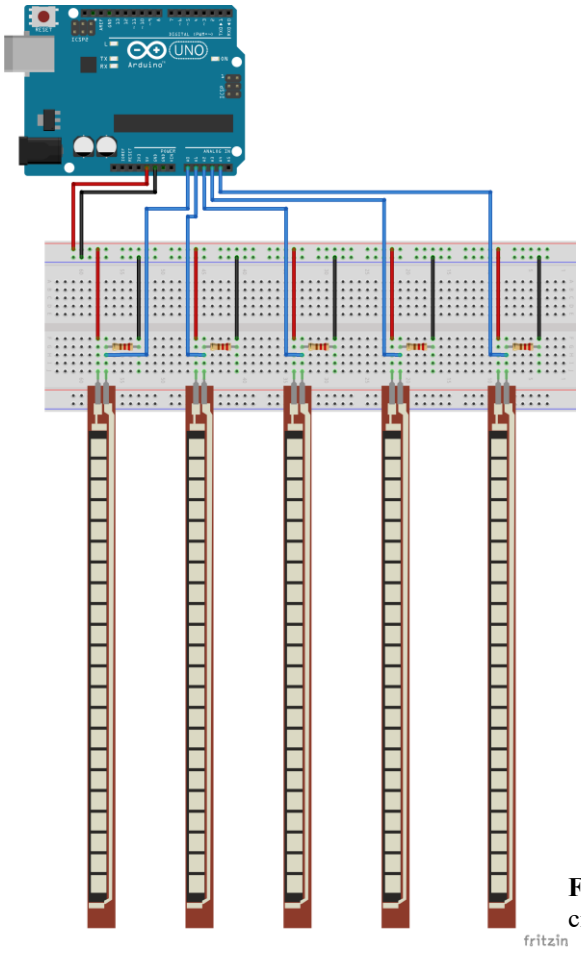


Fig. 21 Schema del collegamento tra la scheda Arduino e i cinque sensori di flessione

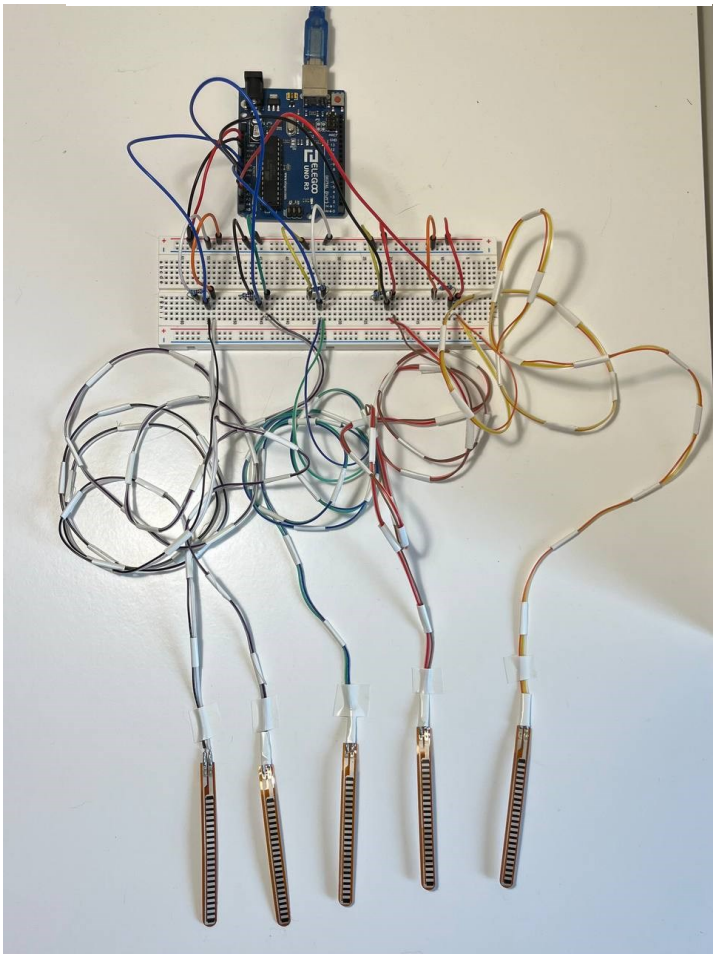


Fig. 22 Foto del collegamento tra la scheda Elegoo e i cinque sensori di flessione

Per poter controllare i servomotori muovendo la mano, è necessario vincolare i sensori a quest'ultima.

Per realizzare ciò, i sensori sono stati fissati ad un guanto con del nastro isolante. Questa procedura assicura una buona aderenza al guanto e riduce il rischio di spostamenti.

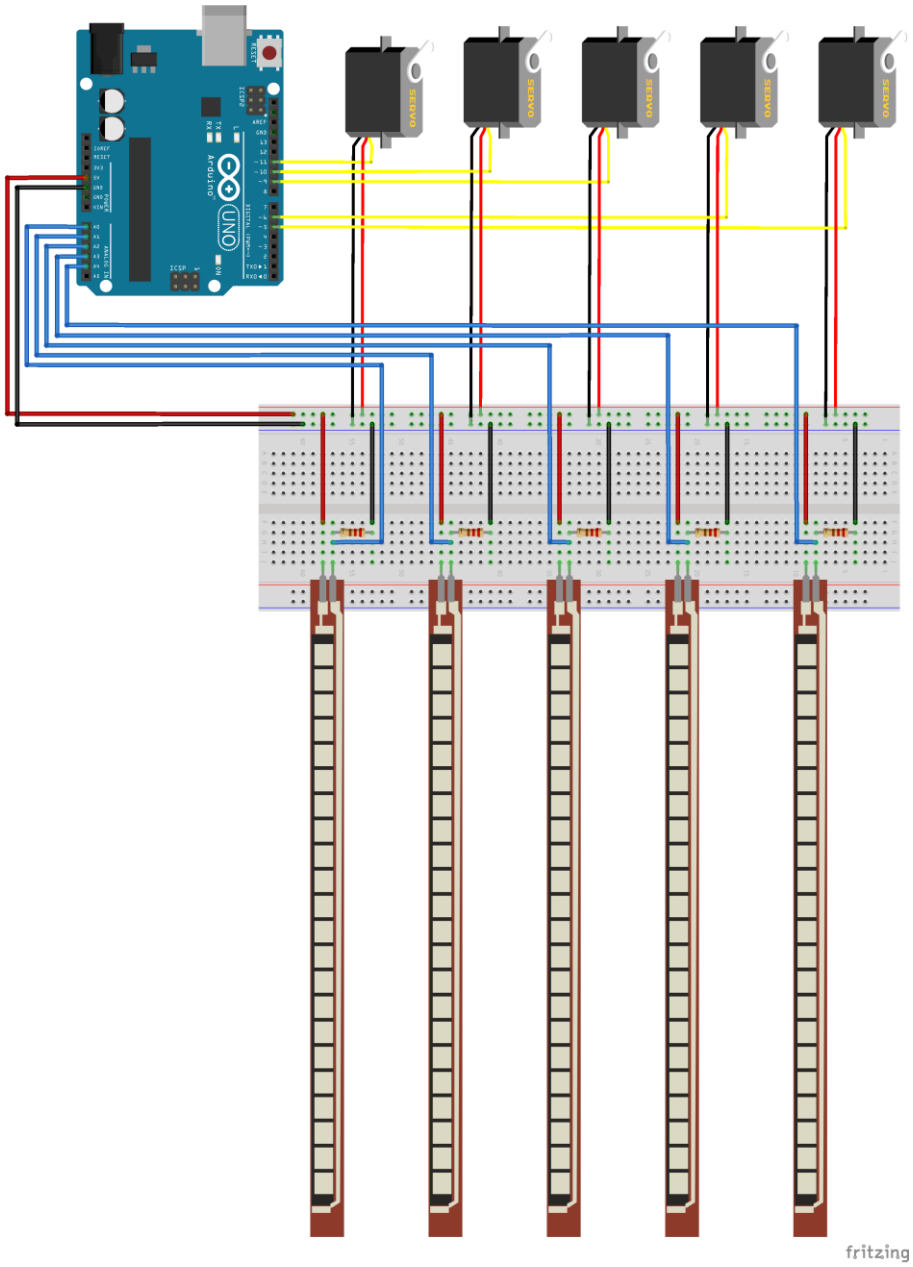


Fig. 23 Schema del circuito completo

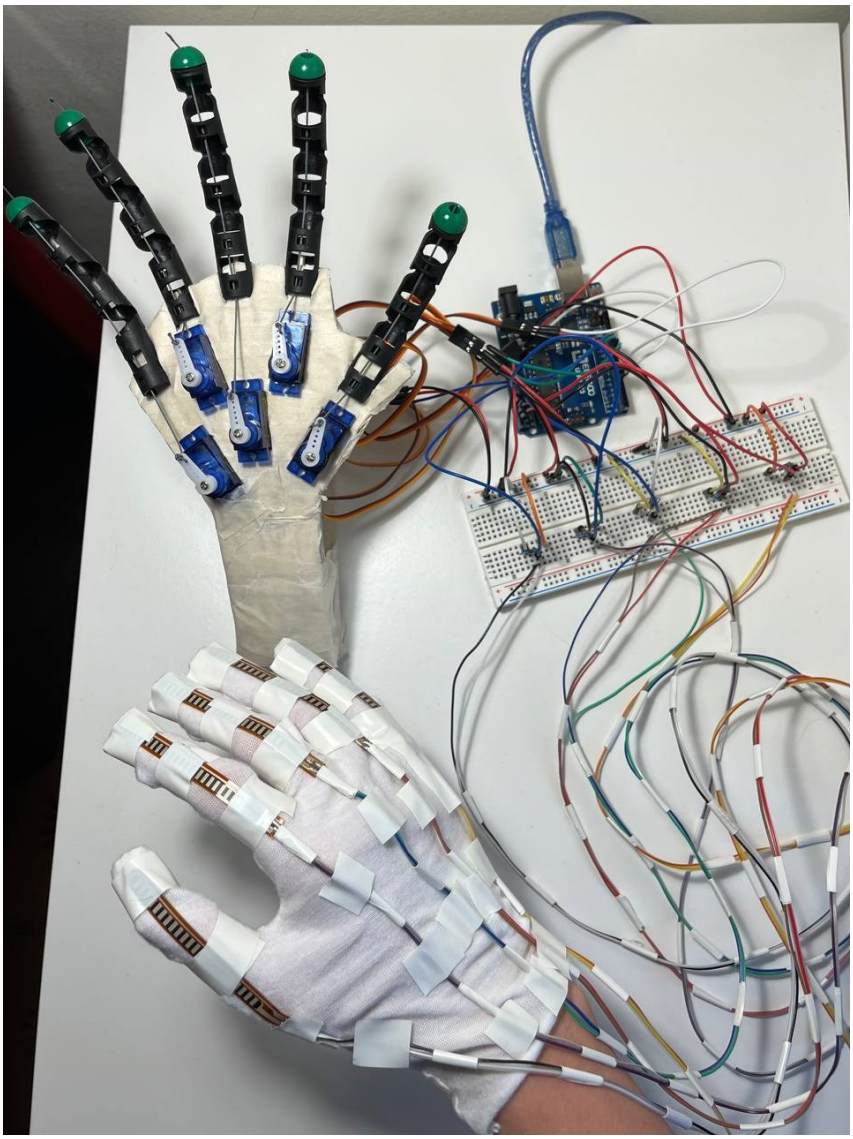


Fig. 24 Foto del circuito completo

4. Codice

Il linguaggio di programmazione usato in Arduino è una versione semplificata del linguaggio C.

I programmi, dopo essere scritti dal PC utilizzando l'IDE di Arduino, vengono trasmessi alla scheda via USB.

Il codice sorgente di un programma per Arduino è chiamato sketch.

Ogni sketch è diviso in due funzioni:

- Setup, dove si configura la scheda
- Loop, che contiene il programma vero e proprio

La funzione di setup viene eseguita una sola volta dopo aver acceso la scheda, mentre la funzione di loop viene ripetuta ciclicamente.

Il codice sviluppato per l'utilizzo della mano robotica mira a:

- Inizializzare i dispositivi, stabilendo una connessione sia tra la scheda Arduino e i servomotori sia tra la scheda Arduino e i sensori di flessione
- Acquisire i dati provenienti dai sensori di flessione
- Convertire i valori di flessione rilevati dai sensori in gradi, che successivamente verranno utilizzati per la rotazione del servomotore.
- Azionare il servomotore e consentirgli di ruotare secondo i gradi calcolati precedentemente.

Il codice completo è riportato in seguito:

```
1    #include <Servo.h>
2
3    Servo thumb;
4    Servo index;
5    Servo middle;
6    Servo ring;
7    Servo pinkie;
8
```

```
9   int flexThumbPin = A0;
10  int flexIndexPin = A1;
11  int flexMiddlePin = A2;
12  int flexRingPin = A3;
13  int flexPinkiePin = A4;
14
15  void setup() {
16    Serial.begin(9600);
17
18    thumb.attach(11);
19    index.attach(10);
20    middle.attach(9);
21    ring.attach(6);
22    pinkie.attach(5);
23
24  }
25
26  void loop() {
27
28    int thumbRead = analogRead(flexThumbPin);
29    //Serial.println("Pollice: " + String(thumbRead));
30
31    int indexRead = analogRead(flexIndexPin);
32    //Serial.println("Indice: " + String(indexRead));
33
34    int middleRead = analogRead(flexMiddlePin);
35    //Serial.println("Medio: " + String(middleRead));
36
37    int ringRead = analogRead(flexRingPin);
38    //Serial.println("Anulare: " + String(ringRead));
39
40    int pinkieRead = analogRead(flexPinkiePin);
41    //Serial.println("Mignolo: " + String(pinkieRead));
42
```

```
43      int thumbPos = map(thumbRead, 450, 240, 0, 180);
44      int indexPos = map(indexRead, 510, 250, 0, 180);
45      int middlePos = map(middleRead, 500, 240, 0, 180);
46      int ringPos = map(ringRead, 500, 200, 0, 180);
47      int pinkiePos = map(pinkieRead, 500, 200, 0, 180);
48
49      thumb.write(thumbPos);
50      //Serial.println("P: " + String(thumbPos));
51
52      index.write(indexPos);
53     //Serial.println("I: " + String(indexPos));
54
55      middle.write(middlePos);
56      //Serial.println("M: " + String(middlePos));
57
58      ring.write(ringPos);
59      //Serial.println("A: " + String(ringPos));
60
61      pinkie.write(pinkiePos);
62      //Serial.println("Mg: " + String(pinkiePos));
63
64      delay(250);
65    }
```

A partire da questo punto, verrà eseguita un'analisi approfondita del codice.

```
1     #include <Servo.h>
```

Questo comando incorpora la libreria `<Servo.h>`, la quale permette di controllare i servomotori. La libreria supporta fino ad un massimo di 12 motori utilizzando un solo timer. È importante notare che l'utilizzo di questa libreria disabilita la funzione `analogWrite()`, sui pin 9 e 10, indipendentemente dal fatto che su questi sia presente un servomotore o meno. Lo scopo di questo comando è di attivare i cinque servomotori attraverso comandi successivi.

```
3     Servo thumb;
4     Servo index;
5     Servo middle;
6     Servo ring;
7     Servo pinkie;
```

Il comando `Servo` viene utilizzato per nominare un servomotore. Questa operazione è ripetuta cinque volte, associando ognuna di queste iterazioni ad un diverso dito e al rispettivo servomotore.

```
9     int flexThumbPin = A0;
10    int flexIndexPin = A1;
11    int flexMiddlePin = A2;
12    int flexRingPin = A3;
13    int flexPinkiePin = A4;
```

Con questo comando ogni sensore di flessione viene associato ad un pin analogico.

All'interno della funzione di set up si trovano i seguenti comandi:

```
16      Serial.begin(9600);
```

Questa istruzione viene utilizzata per l'inizializzazione della comunicazione seriale sulla scheda.
La comunicazione seriale è un metodo di trasmissione dei dati, tra dispositivi digitali, nella quale i bit sono trasferiti lungo un solo canale di trasmissione, uno di seguito all'altro. Ciò può avvenire in diverse modalità, come UART, SPI, I2C e altri. Questi protocolli sono normalmente utilizzati per collegare dispositivi elettronici (come microcontrollori, sensori, …) in modo che possano scambiare informazioni in modo efficiente ed affidabile.
Questa comunicazione, a differenza della comunicazione parallela che richiede che più bit vengano trasmessi contemporaneamente su più fili differenti, è particolarmente utile quando sono presenti limitazioni di spazio o di risorse.
Con la seguente istruzione è possibile impostare la comunicazione seriale, definendo la velocità della comunicazione in bits per secondo, o baud.

La velocità di trasmissione viene impostata a 9600 bit al secondo, in quanto questa velocità deve corrispondere a quella impostata nella console seriale dell'ambiente Arduino per garantire una corretta comunicazione tra la scheda Arduino e il computer.

```
18      thumb.attach(11);
19      index.attach(10);
20      middle.attach(9);
21      ring.attach(6);
22      pinkie.attach(5);
```

Questo comando appartiene alla libreria `<Servo.h>`. Viene utilizzato per collegare il servomotore ad un pin digitale. Per il controllo dei servomotori, sono stati selezionati i pin analogici dotati di funzionalità PWM (Modulazione di Larghezza di Impulso).
In Arduino Uno, i pin PWM sono i pin 3, 5, 6, 9, 10, 11 e sono contrassegnati con il simbolo "~".

All'interno della funzione loop sono presenti i seguenti comandi:

```
28      int thumbRead = analogRead(flexThumbPin);
29      //Serial.println("Pollice: " + String(thumbRead));
30
31      int indexRead = analogRead(flexIndexPin);
32      //Serial.println("Indice: " + String(indexRead));
33
34      int middleRead = analogRead(flexMiddlePin);
35      //Serial.println("Medio: " + String(middleRead));
36
37      int ringRead = analogRead(flexRingPin);
38      //Serial.println("Anulare: " + String(ringRead));
39
40      int pinkieRead = analogRead(flexPinkiePin);
41      //Serial.println("Mignolo: " + String(pinkieRead));
```

Dopo aver inizializzato una variabile intera, le viene assegnato il valore dato dal comando `analogRead()`.
Tale comando legge il valore dal pin analogico specificato.
La scheda Arduino Uno è dotata di un convertitore analogico-digitale a 6 canali, con risoluzione a 10 bit. Questo significa che mappa le tensioni di ingresso comprese tra 0 e 5 V su valori interi compresi tra 0 e 1023.

Il comando `Serial.println()` permette di stampare i valori acquisiti nella riga precedente.
In questo modo è possibile osservare per quali valori il sensore di flessione risulterà piegato.
Nel caso del sensore di flessione applicato al pollice, la variabile `thumbRead` assume il valore di 450 quando il sensore non è piegato, mentre raggiunge il valore di 250 quando è completamente piegato.
È possibile osservare che il range dei valori acquisiti varia tra un sensore e l'altro, in quanto i sensori non sono identici a causa di diversi fattori di produzione.

```
43      int thumbPos = map(thumbRead, 450, 250, 0, 180);
44      int indexPos = map(indexRead, 470, 220, 0, 180);
45      int middlePos = map(middleRead, 430, 190, 0, 180);
46      int ringPos = map(ringRead, 500, 280, 0, 180);
47      int pinkiePos = map(pinkieRead, 450, 250, 0, 180);
```

Con questo comando è possibile rimappare un numero da un range ad un altro.
Il valore acquisito dal sensore di flessione sarà trasformato o rimappato nel range dei gradi corrispondenti alla capacità di rotazione del servomotore, ovvero tra 0 e 180 gradi.

```
49      thumb.write(thumbPos);
50      //Serial.println("P: " + String(thumbPos));
51
52      index.write(indexPos);
53     //Serial.println("I: " + String(indexPos));
54
55      middle.write(middlePos);
56      //Serial.println("M: " + String(middlePos));
57
```

```
58      ring.write(ringPos);
59      //Serial.println("A: " + String(ringPos));
60
61      pinkie.write(pinkiePos);
62      //Serial.println("Mg: " + String(pinkiePos));
```

Il comando `.write()` fa parte della libreria `<Servo.h>`. Con questo comando è possibile regolare l'angolo del servomotore.
Il valore fornito come argomento rappresenta l'angolo in gradi, e per questo, prima dell'utilizzo di questo comando, i valori acquisiti dal sensore di flessione sono stati rimappati in gradi.
I valori precedentemente mappati sono stati stampati utilizzando il comando `Serial.println()`. Tale procedura serve come verifica per garantire il corretto funzionamento del codice, permettendo di visualizzare e controllare gli angoli corrispondenti ai valori acquisiti dal sensore di flessione.

```
64      delay(250);
```

Per concludere, viene inserito un comando di delay, il quale mette in pausa l'esecuzione del programma per un periodo di tempo specificato in millisecondi.
Questa istruzione viene utilizzata per introdurre una pausa o un ritardo nel codice per permettere alla scheda di acquisire nuovi valori.
È stato scelto un intervallo di 250 millisecondi per consentire una sufficiente fase di acquisizione, bilanciando l'obiettivo di ottenere un adeguato tempo di acquisizione senza introdurre eccessivi ritardi.

5. Conclusione

All’accensione di Arduino, tenendo la mano aperta, non è visibile alcun cambiamento. Tuttavia, piegando le dita, è possibile constatare il corretto funzionamento del progetto, sia piegando un solo dito che più contemporaneamente.

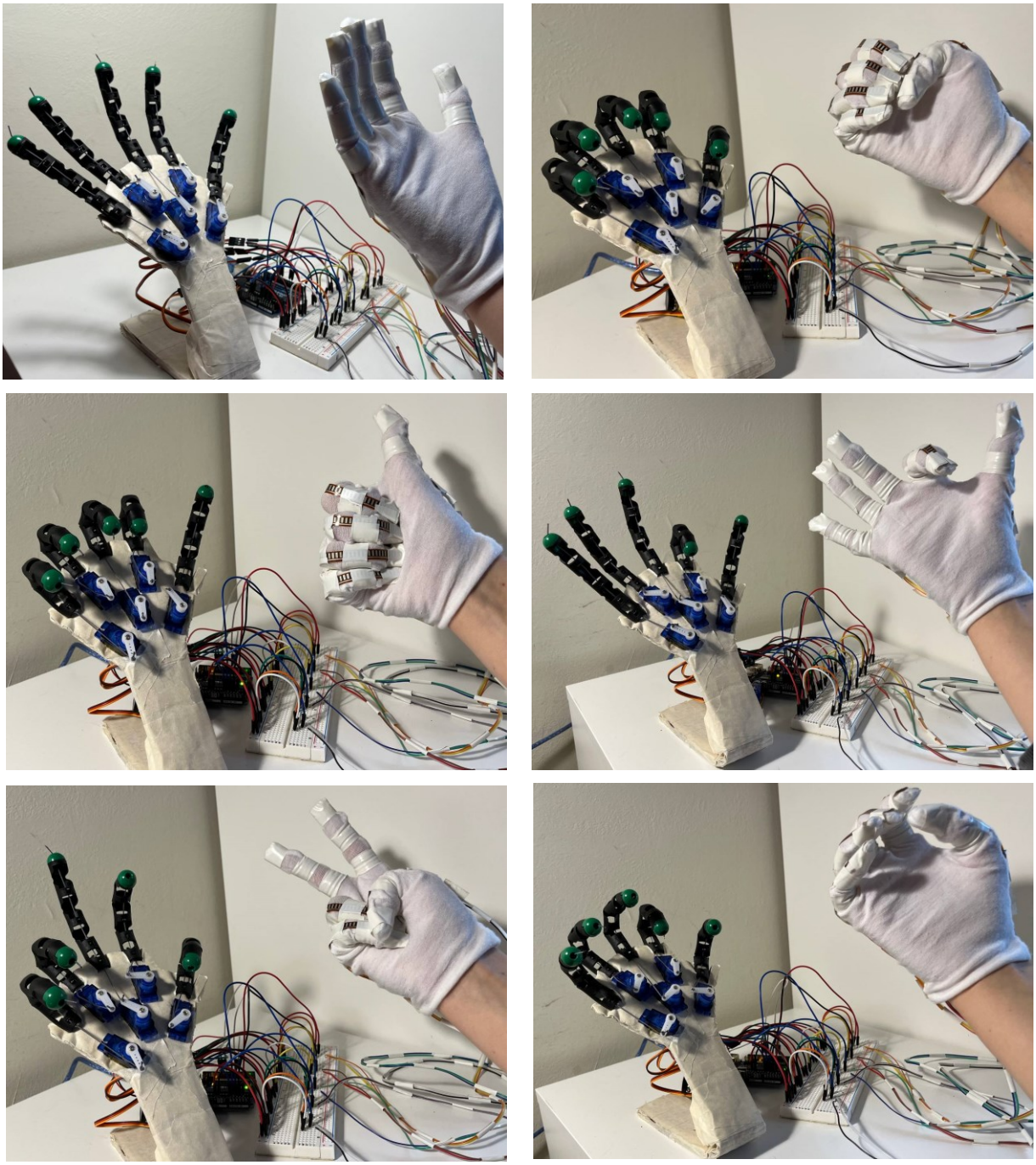


Fig. 25 Piegamento delle dita in base al movimento della mano

Nonostante ciò, ci sono sicuramente margini per dei miglioramenti:

- È presente un evidente ritardo, attribuibile sia al codice, dovuto al comando di delay, che a una non idealità dei cavi.
- La mancanza di precisione è causata non solo dal ritardo, ma anche da collegamenti instabili.

  Può succedere alcune volte che un dito robotico non segua il movimento del dito a causa di connessioni poco stabili sulla breadboard. Se il cavo non è ben posizionato, si verifica una mancanza di contatto. Dato che i cavi sono collegati ad una mano in movimento, è facile che uno di essi si sposti.

  Saldando i cavi al posto di utilizzare la breadboard si eliminerebbe il problema.

  Un altro fattore che influisce sulla precisione è l'aderenza del guanto alla mano. Essendo in cotone, non aderisce completamente alla mano, quindi piegando il dito, il sensore di flessione non si piega sempre completamente.

  Per migliore ciò, servirebbe un guanto di un materiale più robusto o collegare i servomotori direttamente al dito.

  È stata anche esplorata l'opzione di posizionare i sensori all'interno del dito anziché sul dorso della mano, tuttavia, questa scelta rendeva più difficile il movimento, compromettendo il risultato finale.

  La precisione del movimento è influenzata dalla qualità dei servomotori. Un servomotore di migliore qualità garantirebbe un movimento più preciso e ridurrebbe il rumore durante il funzionamento. Questo miglioramento della qualità potrebbe aumentare il costo complessivo del prodotto. Pertanto, per lo scopo di questo progetto, l'adozione di un servomotore di qualità superiore potrebbe risultare in un rapporto qualità-prezzo poco conveniente.
- Come ultimo miglioramento, si potrebbe utilizzare un materiale diverso dal cartone per la realizzazione la base. Questo miglioramento sarebbe principalmente estetico.

  Sicuramente una base in un materiale più resistente, come l'alluminio, garantirebbe una maggiore resistenza e una maggiore stabilità. Tuttavia, va notato che in questo caso la stabilità non è un requisito particolarmente importante, poiché la base in cartone non ha causato problemi.

Bibliografia

Arduino. (s.d.). *Arduino*. Tratto da Sito web di documentazione di Arduino: https://www.arduino.cc/reference/it/language/functions/analog-io/analogread/

Arduino Facile. (s.d.). Tratto da Come Utilizzare il Monitor Seriale per Determinare se un Pulsante Funziona: https://www.arduinofacile.it/2020/11/06/come-utilizzare-il-monitor-seriale-per-determinare-se-un-pulsante-funziona/#:~:text=mediante%20l'istruzione%3A-,Serial.,bits%20per%20second%20(baud).

Arduino. (s.d.). *Sito di documentazione Arduino*. Tratto da Arduino: https://www.arduino.cc/reference/en/libraries/servo/

Francello, S. (s.d.). *Servo e Arduino*. Tratto da Progetti Arduino: https://www.progettiarduino.com/5-servo-e-arduino.html

Leonardo, C. (s.d.). *leonardo canducci*. Tratto da Il linguaggio di programmazione di Arduino: https://www.leonardocanducci.org/wiki/sta/arduino_2

Logica programmabile. (s.d.). Tratto da Caratteristiche Arduino Uno: https://logicaprogrammabile.it/caratteristiche-arduino-uno/

Matteo, M. (s.d.). Slide del corso di Laboratorio di microelettronica.

Salvatore, F. (s.d.). *Progetto Arduino*. Tratto da Arduino come funziona: https://www.progettiarduino.com/cosa-egrave-arduino-come-funziona-arduino.html#google_vignette