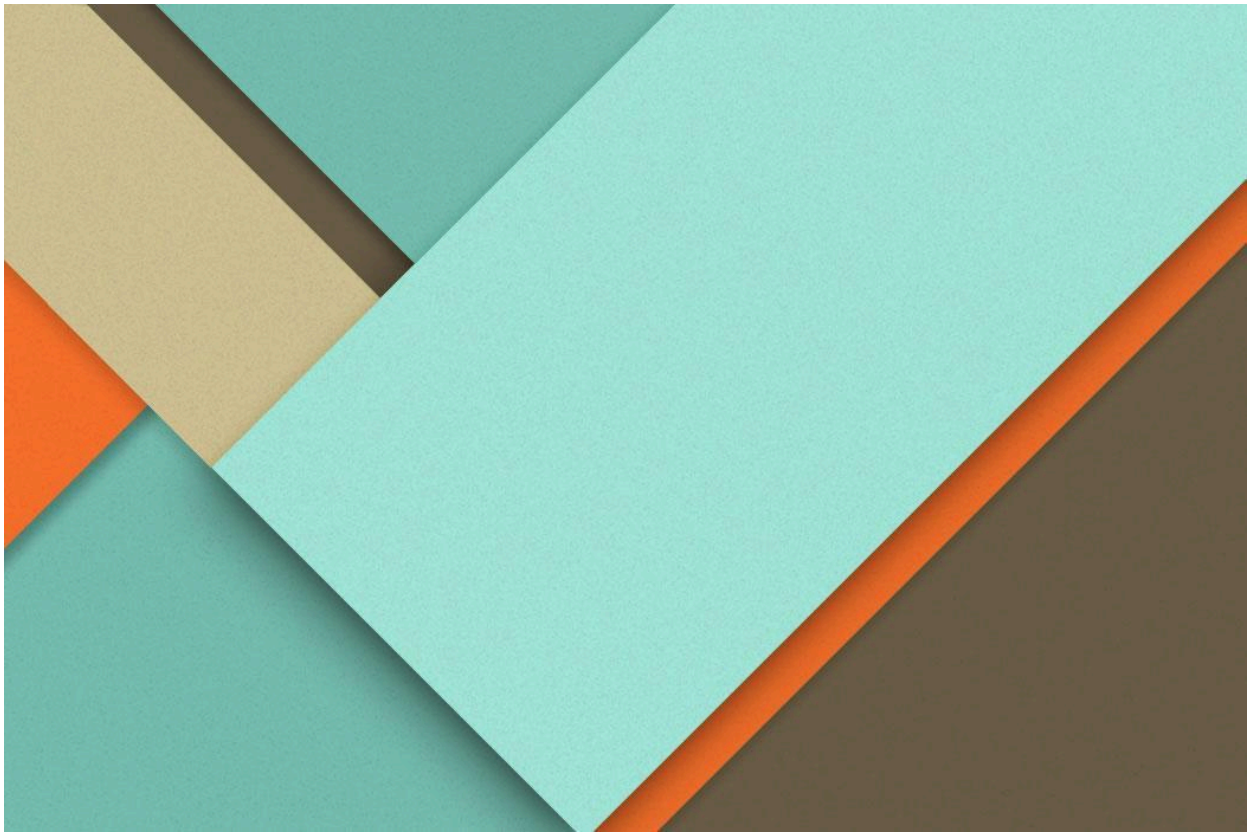


COLOR - CASCADE

24/05/2024

Matteo Amoroso
I.T.I Omar Novara
5° A Rob
A.S. 2023/2024

INDICE:

INTRODUZIONE:

Uno smistatore a colori, come mai?

Ebbene si come progetto di finale ho pensato di realizzare uno smistatore di colori, quest'idea è nata dalla voglia di automatizzare ciò che per me è sempre stato noioso fare, sistemare gli oggetti una volta utilizzati. Un esempio che lascia intuire velocemente il motivo di questo progetto è: quando dopo aver giocato, realizzato o costruito qualcosa con la lego, dovevo rimettere a posto i pezzi rimanenti, i quali finivano inesorabilmente tutti in un'unica scatola e mai suddivisi. Bene proprio per questo disagio è nato questo smistatore a colori, il quale mi permette di suddividere i pezzi della lego in modo automatico, ma non solo, anche tutti gli altri oggetti che hanno differenze di colori.

Il progetto quindi è composto da un braccio robotico stampato interamente in 3D che grazie a dei servomotori possiede ben quattro gradi di libertà nello spazio, più una finta intercambiabile facilmente per una maggior possibilità nella presa di diversi tipi di oggetti. La seconda parte invece è l'effettivo sistema che svolge la suddivisione a colori. Questo sistema prende a campione l'oggetto posto in esame dal braccio, ne rivela il colore e lo suddivide in una scatola differente per ogni colore disponibile.

Nel corso del progetto però mi è sorta una domanda, perchè limitarsi a una struttura che smista i pezzi della lego; ed è qui che mi è venuto in mente che questo stesso sistema potrebbe essere utilizzato nella compravendita online, mi spiego: Il cliente dal suo smartphone, attraverso un'applicazione desktop (download disponibile collegandosi al sito: mycolorcascade.altervista.org) apre uno store, nel quale è possibile selezionare diversi prodotti di colore appunto differente. Il cliente una volta aggiunto questi articoli manda l'ordine.

Questo ordine viene ricevuto dal sistema, il quale prepara un "pacco" pronto per la spedizione con all'interno gli articoli con i differenti colori. Ovviamente immaginarlo con i colori potrebbe non avere senso, ma lo stesso sensore a colori potrebbe essere sostituito con un lettore di codici a barre o QR. Così da rendere più universale il sistema di smistamento.

COMPONENTI:

- **MANIPOLATORE**:

1. Arduino UNO R3 - Elegoo x1
2. Modulo PCA9685 x1
3. Servomotori DM996 x3
4. Servomotore 8120MG x1
5. Servomotore SG90 x1
6. Jumper Wires x5
7. Ingranaggi di plastica x4
8. Elastico x1
9. Cavi 28 AWG
10. Guaine termorestringenti
11. Stampe in 3D del manipolatore
12. Interruttore ON/OFF
13. Alimentatore da 9V - 1A

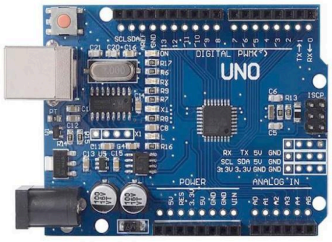
- **SMISTATORE**:

14. Arduino Nano x1
15. Shield Arduino Nano I/O x1
16. Sensore TCS3200 GY-31 x1
17. Servomotore DM996 x1
18. Servomotore SG90 x1
19. Stampe in 3D dello smistatore
20. Jumper Wires
21. Batteria da 9V ricaricabile x1

SPECIFICHE DEI COMPONENTI:

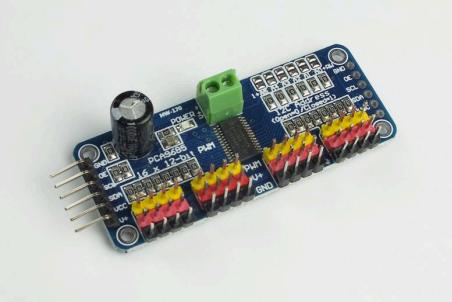
Di seguito verranno esposte in breve le specifiche dei componenti utilizzati per la realizzazione del progetto con le relative immagine:

Arduino Uno R3 - Elegoo:



è una scheda microcontrollore basata sull'ATmega328P. Dispone di 14 pin di ingresso/uscita digitali, 6 utilizzabili come uscite PWM, 6 ingressi analogici, una connessione USB, un jack di alimentazione, e un pulsante di reset. Contiene già tutto il necessario per supportare il microcontrollore; basta collegarlo a un computer con un cavo USB o alimentarlo con un adattatore CA-CC.

Modulo PCA9685:



è un modulo che utilizza l'interfaccia I2C, compatto e versatile per il controllo di fino a 16 servomotori o LED. Sia il pin Vcc che i pin di segnale possono gestire una tensione compresa tra 3 e 5 V, mentre la tensione massima per il pin V+ è 6 V. La risoluzione dei segnali PWM è impressionante a 12 bit.

Servomotore DM996:

è caratterizzato da ottime performance di potenza; caratteristica fondamentale è la Rotazione Continua che permette di sfruttare le potenzialità del servo in applicazioni robotiche. Questo servo dispone di ingranaggi in metallo e di due cuscinetti a sfera che ne assicurano una lunga durata. Tipo di motore: DC, tensione operativa: 4.8V - 6.6V DC.

Servomotore 8120MG:

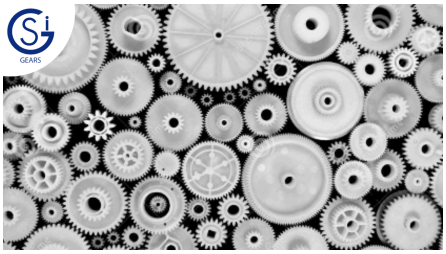
è un servomotore digitale molto potente con ingranaggi in metallo impermeabile Td-8120Mg con coppia elevata da 20Kg, angolo 180°.

Servomotore SG90:

è un micro servo analogico 9g è caratterizzato da dimensioni molto ridotte, pur conservando ottime performance di potenza; dispone di ingranaggi in materiale plastico. Tensione operativa: 4.8V DC, albero: 21T.

Jumper Wires:

sono uno dei componenti più importanti permettono di creare collegamenti momentanei (evitano la saldatura) tra schede microcontrollori e moduli o breadboard.



Ingranaggi in plastica:

sono piccoli ingranaggi bianchi in plastica, utilizzati per trasportare la rotazione del “gomito” all’end-effector del manipolatore.

Cavi 28AWG e Guaine Termorestringenti:

questi due componenti sono stati utilizzati per permettere la realizzazione di “prolunghe” dei cavi già presenti nei servomotori. Le caratteristiche dei cavi da 28 AWG è che sono dello stesso spessore dei cavi dei servomotori mentre le guaine termorestringenti mi servivano per isolare la saldatura.

Interruttore ON/OFF:

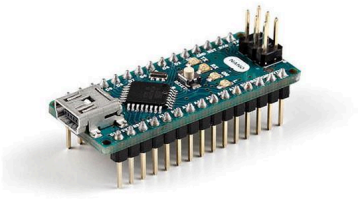
ha la funzione di aprire o chiudere in modo stabile un circuito elettrico. Il pulsante permette quindi di aprire o chiudere in modo momentaneo un determinato circuito elettrico. Ho utilizzato un interruttore bipolare che presenta le posizioni di aperto e chiuso.

Alimentatore 9V - 1A:

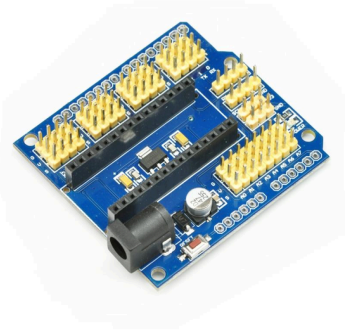
semplice alimentatore da muro che permette la trasformazione di 100-240V , 50-60Hz alternata in una tensione di 9V - 1A di uscita. Utile per alimentare il mio manipolatore senza dover cambiare o ricaricare pile.

Arduino Nano:

è una scheda piccola, completa e compatibile basata sui ATmega328 (Arduino Nano 3.x). Ha più o meno le stesse funzionalità dell’Arduino Uno, ma in una confezione diversa. Manca solo il jack di alimentazione CC e funziona con un cavo USB Mini-B.



Shield Arduino Nano I/O:



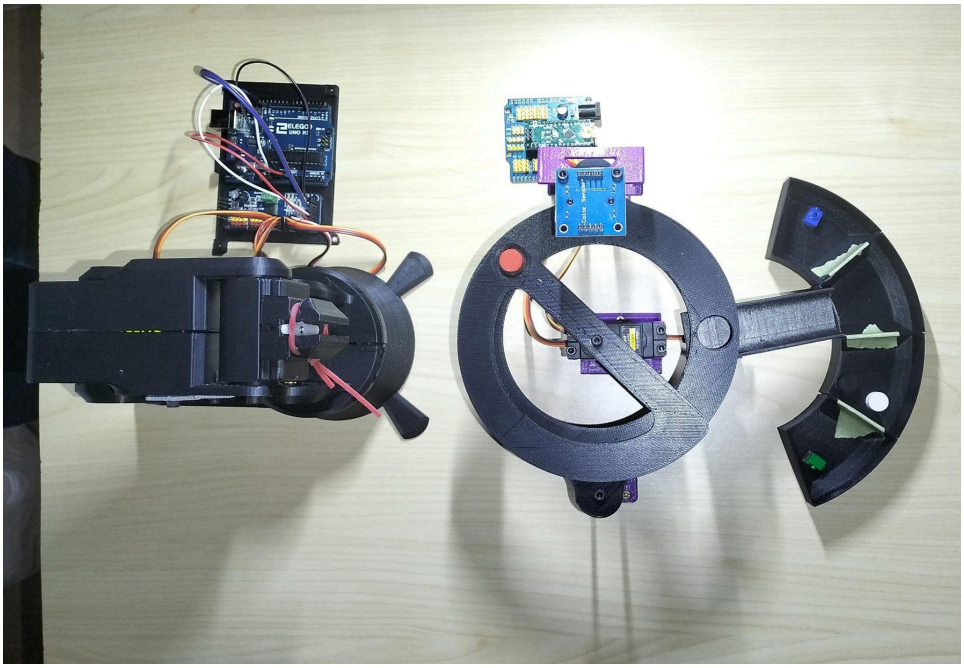
è appositamente progettata per facilitare una connessione tra Arduino Nano e molti altri dispositivi. In essenza, espande il controller Nano per collegare tali dispositivi in modo semplice. Dispone di 14 pin I/O, 8 pin analogici, 6 pin PW, 1 jack per l'alimentazione.

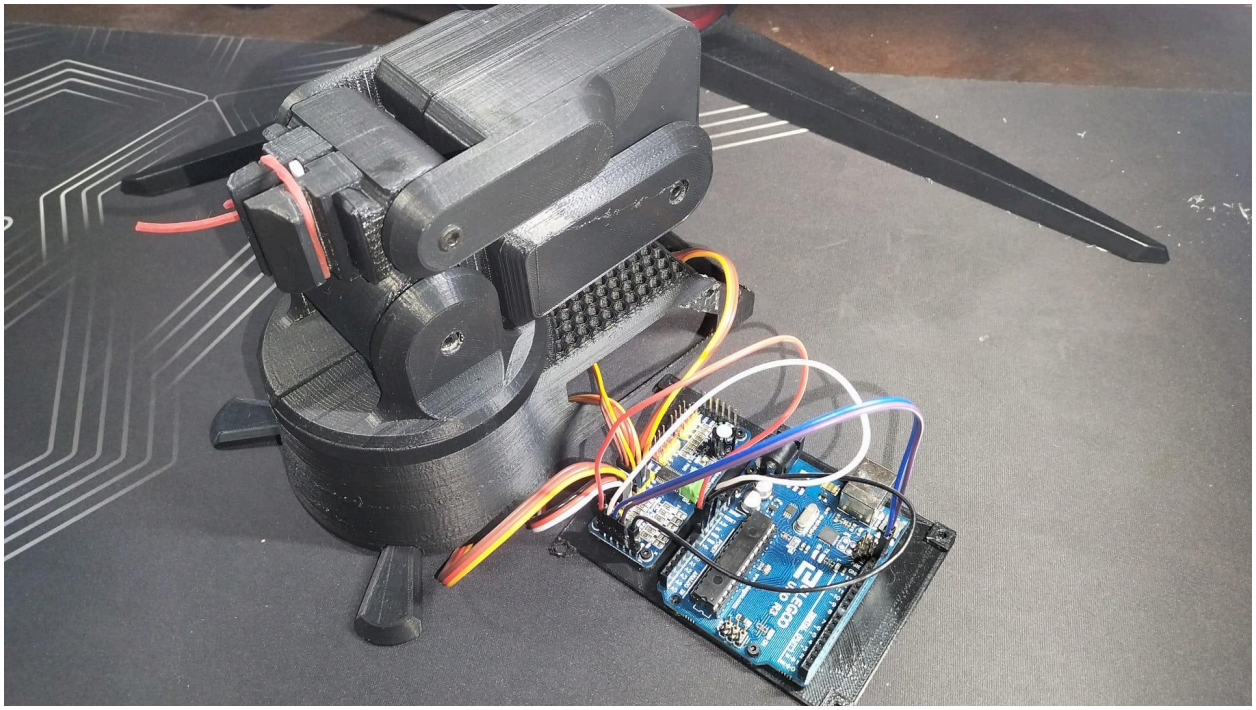
Sensore TCS3200 GY-31:

è un sensore di colore, rilevatore di colori grazie al chip TAOS TCS3200; è in grado di rilevare e misurare la luce incidente su di esso, è composto da una serie di fotodiodi posti in matrice 8x8 per un totale di 64 sensori, 16 sensori sono filtrati per rilevare la luce di colore rosso, 16 quella blu, 16 quella verde e gli ultime 16 non sono filtrati. Tensione di alimentazione: 2.7 - 5.5 Vcc.

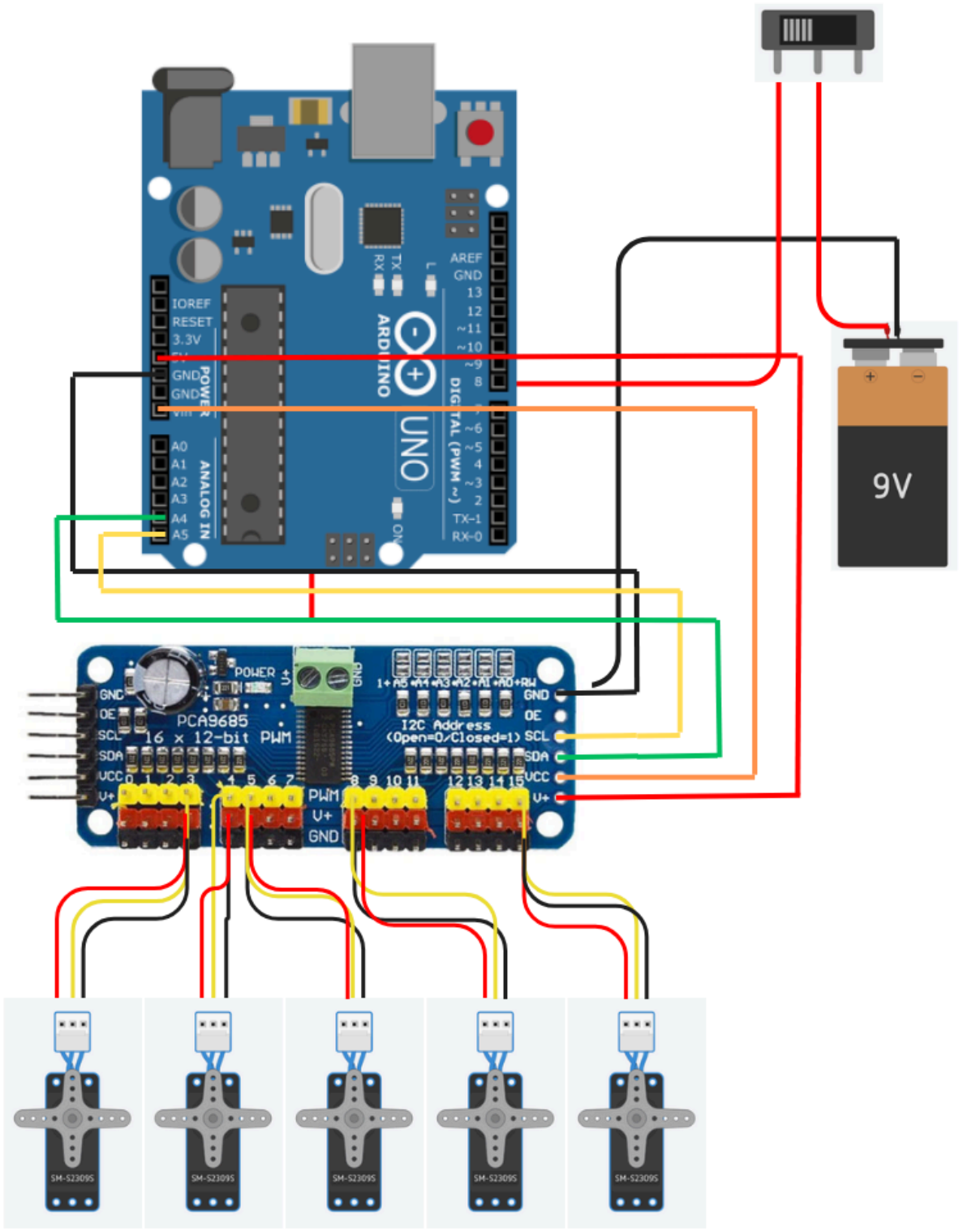
Stampe in 3D del progetto:

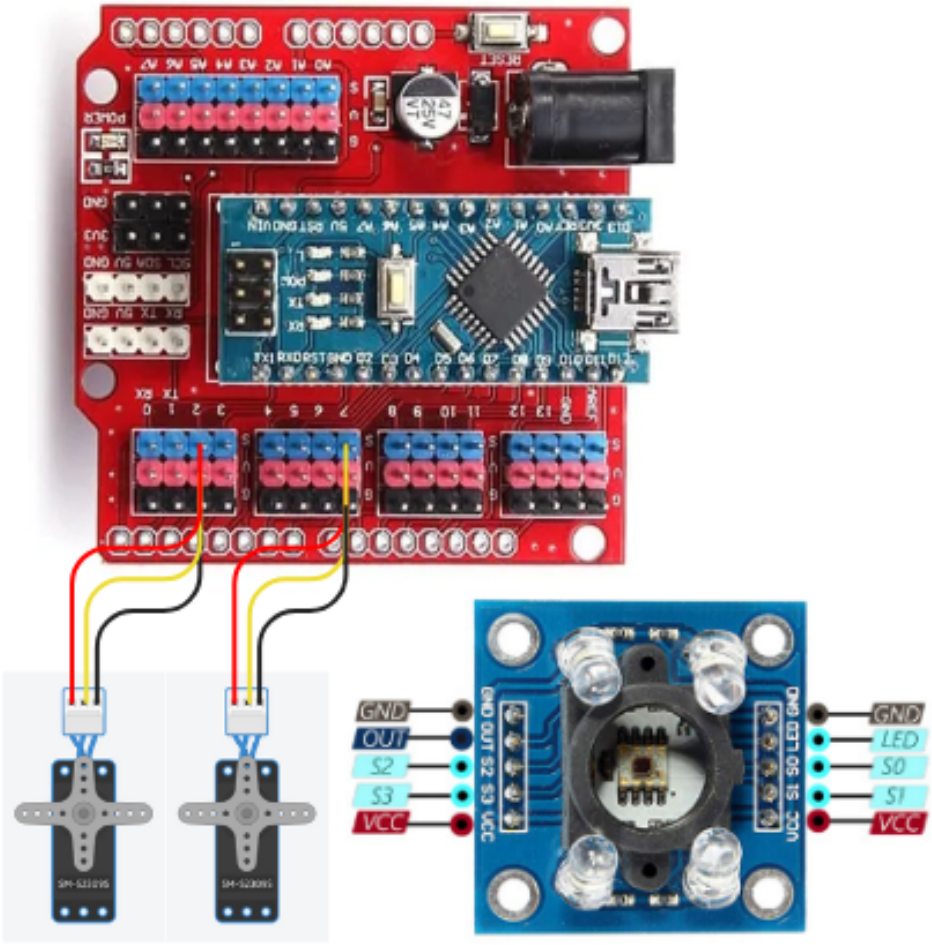
le stampe in 3D del progetto sono state realizzate utilizzando una stampante delta della FLSUN il modello Q5; il materiale del filamento è PLA (acido polilattico) , di colore nero e viola.






DM
DM996


UNO
ARDUINO
9V
PCA9685
16 x 12-bit PWM
I2C Address
(Open=0/Closed=1)
SM-S2309S


GND
OUT
S2
S3
VCC
GND
LED
S0
S1
VCC

TAPPE DI REALIZZAZIONE:

I. Stampa 3D di tutti i componenti:

Il primo step è stato la realizzazione e la stampa delle parti in 3D che compongono il manipolatore e lo smistatore. Ho scelto come materiale di utilizzare un filamento PLA (Acido polilattico) di colore nero e viola. La stampante utilizzata è la FLSUN Q5, una delta che permette movimenti armonici e circolari a differenza di quelle cartesiane.

II. Montaggio del manipolatore:

Supposto di aver finito la stampa in 3D e di aver acquistato tutti i componenti, sono passato al montaggio del manipolatore, mettendo assieme le parti stampate con i componenti elettrici acquistati. Sono partito dal manipolatore in quanto il suo montaggio era sicuramente più complesso; ho iniziato con montare l'end-effector e man mano mi spostavo alla base. Infine ho inserito la scheda Arduino Uno R3 e il modulo PCA9685, seguendo lo schema elettrico (sopra riportato) ho connesso tutti i cavi e tutti i collegamenti.

III. Taratura dei servomotori del manipolatore:

Grazie ai collegamenti fatti nel punto (2), e attraverso un programma di arduino e visual studio (C#), mi è stato possibile tarare i miei servomotori. Quando si parla di taratura, parlo di trovare gli impulsi che servono ai servomotori per permettergli di effettuare la posizione massima e minima interessata. Come per la base che mi interessava una posizione massima di 90°, -90° e la posizione di partenza 0°, ogni servomotore ha i propri angoli di apertura e chiusura.

IV. I primi movimenti:

Dopo aver effettuato la taratura, ho iniziato a far muovere il manipolatore utilizzando come software il programma di arduino, nella sezione listati verrà inserito il codice con le spiegazioni pezzo per pezzo...

I principali movimenti che ho cercato di comporre con il manipolatore sono il movimento per la posizione di cattura, per la posizione di posizionamento del cubo sullo smistatore e il ritorno in posizione di "taratura".

V. Montaggio dello smistatore:

Il montaggio dello smistatore è stato sicuramente più veloce e meno articolato rispetto a quello del manipolatore, ma per poterlo montare ho dovuto recuperare un asse di legno, e ritagliarla delle misure da me desiderate. Dopo di che ho preso le parti stampate in 3D (principalmente i sostegni viola) e li ho avvitati, dopo aver creato i buchi, sull'asse di legno. Infine ho inserito nei luoghi appositi i componenti come i servomotori o il sensore di colore.

VI. Taratura dei servomotori dello smistatore:

Anche in questa parte di progetto come nel manipolatore, troviamo dei servomotori, i quali vanno tarati nello stesso modo del primo ...

VII. Calibrazione del sensore di colori:

Per la calibrazione del sensore di colori, ho dovuto mettere sotto esame (del sensore), i miei 3 cubi colorati, per un po' di minuti ciascuno, e segnarmi il valore massimo e minimo dei valori RGB di tutti e tre. Questi valori massimi e minimi di RGB (RED, GREEN, BLU), mi permettono di creare le condizioni "if" per permettere al sensore di riconoscere che cubo riceve in esame.

VIII. I primi movimenti:

Eseguito il montaggio ed eseguita la taratura / calibrazione di tutti i componenti, ho realizzato i primi movimenti, permettendo allo smistatore di leggere il colore di alcuni Cubi stampati in 3D con colori di PLA differenti tra loro.

IX. Creazione e aggiunta di un app e sito web:

Dopo aver completato tutta la realizzazione hardware del progetto che corrisponde quindi alla realizzazione del manipolatore e dello smistatore di colori, e dopo aver realizzato i vari programmi per il loro normale funzionamento. Ho deciso di realizzare un applicazione per computer in estensione .exe utilizzando Visual Studio; questa applicazione mi ha permesso di realizzare il concetto di "e-commerce" o meglio è un applicazione per simulare un acquisto online. Inoltre ho pensato anche a come rendere "pubblica" la diffusione dell'app e quale buon modo se non realizzare un sito in html e css, per il download del file .exe.

LISTATI/PROGRAMMI:

Listato 1) Taratura dei servomotori sia del manipolatore che dello smistatore:

- 1) Variabili + Inizializzazione.
- In questa parte del programma dichiaro tutte le variabili di cui avrò bisogno e le ho inizializzate nel setup.

```
#include<Servo.h>

Servo feeder; //Dichiaro due variabili per i
Servo scivolo;
byte potfeeder = A0; //Utilizzo due potenzio
byte potscivolo = A2;

void setup() {
  feeder.attach (9); //Dichiaro a quale PIN
  scivolo.attach(11);
  pinMode(A0, INPUT); //Dichiaro che i poten
  pinMode(A2, INPUT);
  Serial.begin(9600); //Inizializzo la Seria
}
```

- 2) Programma loop.
- In quest'altra parte ho prelevato il valore dei potenziometri e gli ho mappati per poi inviarli ai servomotori.

```
void loop() {
  int posfeeder = analogRead(potfeeder); //Salvo la lett
  int posscivolo = analogRead(potscivolo);
  int mapfeeder = map(posfeeder, 0, 1023, 0, 180); //Map
  int mapscivolo = map(posscivolo, 0, 1023, 0, 180); //M

  feeder.write(mapfeeder); //Invio ai miei servomotori q
  scivolo.write(mapscivolo);

  Serial.print("La posizione del Feeder è: "); //Metto a
  Serial.println(mapfeeder);
  Serial.print("La posizione dello scivolo è: ");
  Serial.println(mapscivolo);
  delay(200);
}
```

Listato 2) Calibrazione sensore di colori → TCS3200

- 1) Variabili + Inizializzazioni.
- In questa parte del programma dichiaro tutte le variabili di cui avrò bisogno e le ho inizializzate nel setup.

```
#define S0 2 //Dichiaro tutti i Pin utilizzare dal se
#define S1 3
#define S2 4
#define S3 5
#define sensorOut 6

int redMin = 79; //Dichiaro tutti i valori massimi e
int redMax = 224; //tutti i valori massimi e minimi d
int greenMin = 89;
int greenMax = 262;
int blueMin = 79;
int blueMax = 232;

int redPW = 0; //Dichiaro variabili per contenre gli
int greenPW = 0;
int bluePW = 0;

void setup(){
  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);
  pinMode(sensorOut, INPUT);
  digitalWrite(S0,HIGH); //Setto la frequenza al 20%
  digitalWrite(S1,LOW);
  Serial.begin(9600);
}
```

- 2) Programma loop.
- In questa parte del programma ho solo assegnato alle variabili prima dichiarate, il risultato di una funzione esterna per la rilevazione degli impulsi dei vari colori.

```
void loop(){
  redPW = getRedPW(); //Imposto che ogni varibiale
  delay(200);
  greenPW = getGreenPW();
  delay(200);
  bluePW = getBluePW();
  delay(200);

  Serial.print("Red = "); //Pongo a schermo il risu
  Serial.print(redPW);
  Serial.print(" - Green = ");
  Serial.print(greenPW);
  Serial.print(" - Blue = ");
  Serial.println(bluePW);
  delay(500);
}
```

- 3) Funzioni esterne.
- In questa parte del programma invece ho realizzato delle funzioni esterne, che vengono richiamate nella parte di Loop.

```
//Funzioni esterne per prelevare gli impulsi rossi, blu e verdi
int getRedPW() {
  digitalWrite(S2,LOW);
  digitalWrite(S3,LOW);
  int PW;
  PW = pulseIn(sensorOut, LOW);
  return PW;
}
int getGreenPW() {
  digitalWrite(S2,HIGH);
  digitalWrite(S3,HIGH);
  int PW;
  PW = pulseIn(sensorOut, LOW);
  return PW;
}
int getBluePW() {
  digitalWrite(S2,LOW);
  digitalWrite(S3,HIGH);
  int PW;
  PW = pulseIn(sensorOut, LOW);
  return PW;
}
```

Listato 4) Programma per il funzionamento dello smistatore:

```
#include<Servo.h>
#define S0 2
#define S1 3
#define S2 4
#define S3 5
#define Out 6

int freq = 0;
byte posizione = 0;
Servo feeder;
Servo scivolo;
byte posizionepartenza = 190;
byte posizionesensore = 95;
byte posizonefinale = 0;
byte scatola_UTENTE = 0;
byte scatola_ROSSO = 25;
byte scatola_VERDE = 60;
byte scatola_BLU= 94;
int redMin = 79;
int redMax = 224;
int greenMin = 89;
int greenMax = 262;
int blueMin = 79;
int blueMax = 232;
int redPW = 0;
int greenPW = 0;
int bluePW = 0;
String input = "R2B1V1";
String cubirossi;
String cubiblu;
String cubiverdi;
int numR;
int numB;
int numV;
int stato = 0;
```

```
void setup() {
  pinMode (S0, OUTPUT);
  pinMode (S1, OUTPUT);
  pinMode (S2, OUTPUT);
  pinMode (S3, OUTPUT);
  pinMode (Out, INPUT);

  digitalWrite(S0, HIGH);
  digitalWrite(S1, LOW);

  feeder.attach (9);
  scivolo.attach(11);

  Serial.begin(9600);
}

void loop() {

  if(Serial.available()>0)
  {
    while(stato == 0){
    input = Serial.readString();
    cubirossi = input.substring(1,2);
    cubiblu = input.substring(3,4);
    cubiverdi = input.substring(5,6);

    numR = cubirossi.toInt();
    numB = cubiblu.toInt();
    numV = cubiverdi.toInt();

    Serial.print(numR);
    Serial.print(", ");
    Serial.print(numB);
    Serial.print(", ");
    Serial.println(numV);
    stato = 1;
    }
  }
  delay(4000); //Un grande delay che perme
```

- In questa parte del programma vengono scelte le posizioni dello scivolo e ci si assicura il movimento del "feeder".

```
77  for(int i = posizionepartenza; i > posizionesensore; i--){
78    feeder.write(i);
79    delay(30);
80  }
81  delay(1000);
82  //Viene avviata la riconosciutare del colore, e di conseguenza viene decisa l
83  posizione = letturaColore();
84  switch (posizione){
85    case 1:
86      scivolo.write(scatola_ROSSO);
87      break;
88    case 2:
89      scivolo.write(scatola_VERDE);
90      break;
91    case 3:
92      scivolo.write(scatola_BLU);
93      break;
94    case 4:
95      scivolo.write(scatola_UTENTE);
96      break;
97  }
98  delay(1000);
99  //Porta in posizione finale lo smistatore per permettere al Cubo di cadere:
100 for(int i = posizionesensore; i > posizonefinale; i--){
101   feeder.write(i);
102   delay (30);
103 }
104 delay(1000);
105 //Porta in posizine di partenza lo smistatore:
106 for(int i = posizonefinale; i < posizionepartenza; i++){
107   feeder.write(i);
108   delay(5);
109 }
110 posizione = 0;
111 delay(2000);
```

```
113   if(numR == 0 && numB == 0 && numV == 0)
114   {
115     stato = 0;
116   }
117 }
118
119 //Funzione per la lettura del colore
120 int letturaColore(){
121   redPW = getRedPW();
122   delay(200);
123   greenPW = getGreenPW();
124   delay(200);
125   bluePW = getBluePW();
126   delay(200);
127   Serial.print("Red = ");
128   Serial.print(redPW);
129   Serial.print(" - Green = ");
130   Serial.print(greenPW);
131   Serial.print(" - Blue = ");
132   Serial.println(bluePW);
133   delay(500);
134   //PARAMETRI PER L'UTENTE//
135   if(numR != 0){
136     if (redPW<=250 && redPW>=200) {
137     posizione = 1; // UTENTE ROSSO
138     Serial.println("UTENTE ROSSO");
139     numR = numR - 1;
140     Serial.print(numR);
141     }
142   }
```

- In questa parte si notano i parametri scelti per decidere quale posizione e quando, lo scivolo deve assumere.

```
143   //PARAMETRI PER IL ROSSO//
144   else if (redPW<=250 && redPW>=200) {
145     posizione = 2; // ROSSO
146     Serial.println("ROSSO");
147   }
148   else if(numB != 0){
149     if (redPW<=380 && redPW>=300 && bluePW<=200 && bluePW>=100) {
150     posizione = 1; // UTENTE BLU
151     Serial.println("UTENTE BLU");
152     numB = numB - 1;
153     Serial.print(numB);
154     }
155   }
156   //PARAMETRI PER IL BLU//
157   else if (redPW<=380 && redPW>=300 && bluePW<=200 && bluePW>=100) {
158     posizione = 4; // BLU
159     Serial.println("BLU");
160   }
161   else if(numV != 0){
162     if (redPW<=500 && redPW>=400 && bluePW<=400 && bluePW>=250) {
163     posizione = 1; // UTENTE VERDE
164     Serial.println("UTENTE VERDE");
165     numV = numV - 1;
166     Serial.print(numV);
167     }
168   }
169   //PARAMETRI PER IL VERDE//
170   else if (redPW<=500 && redPW>=400 && bluePW<=400 && bluePW>=250) {
171     posizione = 3; // VERDE
172     Serial.println("VERDE");
173   }
174   return posizione;
175 }
```

- L'ultima parte del programma riguarda le stesse funzioni esterne che vengono utilizzate per il programma di taratura del sensore di colore...

```
176 // Funzione per leggere gli impulsi Rossi
177 int getRedPW(){
178   digitalWrite(S2,LOW);
179   digitalWrite(S3,LOW);
180   int PW;
181   PW = pulseIn(Out, LOW);
182   return PW;
183 }
184 
185 // Funzione per leggere gli impulsi Verdi
186 int getGreenPW(){
187   digitalWrite(S2,HIGH);
188   digitalWrite(S3,HIGH);
189   int PW;
190   PW = pulseIn(Out, LOW);
191   return PW;
192 }
193 
194 // Funzione per leggere gli impulsi Blu
195 int getBluePW(){
196   digitalWrite(S2,LOW);
197   digitalWrite(S3,HIGH);
198   int PW;
199   PW = pulseIn(Out, LOW);
200   return PW;
201 }
```

Listato 6) Programma per il funzionamento del manipolatore:

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>

Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

int spalla = 13;
int gomito = 6;
int pinza = 10;
int polso = 2;
int base = 7;

int stato = 0;
void setup() {
  Serial.begin(9600);
  pwm.begin();
  pwm.setPWMFreq(60);
}

void loop() {
  while(stato == 0){
    for(int i=520; i>=380; i--){
      pwm.setPWM(pinza,0,i);
      delay(5);
    }
    delay(2000);
    for(int i=415; i>=160; i--){
      pwm.setPWM(base,0,i);
      delay(5);
    }
    delay(2000);
    for(int i=205; i<=300; i++){
      pwm.setPWM(spalla,0,i);
      delay(5);
```

```
35      delay(2000);
36      for(int i=550; i>=450; i--){
37        pwm.setPWM(gomito,0,i);
38        delay(5);
39      }
40      delay(2000);
41      for(int i=300; i<=355; i++){
42        pwm.setPWM(spalla,0,i);
43        delay(5);
44      }
45      delay(1000);
46      for(int i=355; i<=433; i++){
47        pwm.setPWM(spalla,0,i);
48        delay(5);
49      }
50      delay(2000);
51      for(int i=380; i<=520; i++){
52        pwm.setPWM(pinza,0,i);
53        delay(5);
54      }
55      delay(2000);
56      for(int i=433; i>=350; i--){
57        pwm.setPWM(spalla,0,i);
58        delay(5);
59      }
60      delay(2000);
61      for(int i=450; i>=300; i--){
62        pwm.setPWM(gomito,0,i);
63        delay(5);
64      }
65      delay(2000);
66      for(int i=160; i<=440; i++){
67        pwm.setPWM(base,0,i);
68        delay(5);
69      }
```

```
    }
    delay(2000);
    for(int i=520; i>=380; i--){
      pwm.setPWM(pinza,0,i);
      delay(5);
    }
    delay(2000);
    taratura();
    stato = 1;
  }
}

void taratura(){
  for(int i=440; i>=415; i--){
    pwm.setPWM(base,0,i);
    delay(5);
  }
  delay(1000);
  for(int i=350; i>=205; i--){
    pwm.setPWM(spalla,0,i);
    delay(5);
  }
  delay(1000);
  for(int i=300; i<=550; i++){
    pwm.setPWM(gomito,0,i);
    delay(5);
  }
  delay(1000);
  for(int i=380; i<=520; i++){
    pwm.setPWM(pinza,0,i);
    delay(5);
  }
  delay(1000);
}
```

Listato 7) Applicazione e shop design

- 1) Parte grafica / Design:
- In questa parte ho progettato l'interfaccia grafica dell'applicazione:

- 2) Programma di funzionamento dell'app:

```
int CubiRossi = 0;
int CubiBlu = 0;
int CubiVerdi = 0;

1 riferimento
private void btn_PiuRosso_Click(object sender, EventArgs e)
{
    if(CubiRossi<3)
    {
        CubiRossi = CubiRossi + 1;
        numRossi.Text = CubiRossi.ToString();
        numRossi1.Text = numRossi.Text;
        numTot.Text = (CubiRossi + CubiBlu + CubiVerdi).ToString();
    }
    else { }
}

1 riferimento
private void btn_MenoRosso_Click(object sender, EventArgs e)
{
    if (CubiRossi > 0)
    {
        CubiRossi = CubiRossi - 1;
        numRossi.Text = CubiRossi.ToString();
        numRossi1.Text = numRossi.Text;
        numTot.Text = (CubiRossi + CubiBlu + CubiVerdi).ToString();
    }
    else { }
}

private void btn_PiuBlu_Click(object sender, EventArgs e)
{
    if (CubiBlu < 3)
    {
        CubiBlu = CubiBlu + 1;
        numBlu.Text = CubiBlu.ToString();
        numBlu1.Text = numBlu.Text;
        numTot.Text = (CubiRossi + CubiBlu + CubiVerdi).ToString();
    }
    else { }
}


private void btn_MenoBlu_Click(object sender, EventArgs e)
{
    if (CubiBlu > 0)
    {
        CubiBlu = CubiBlu - 1;
        numBlu.Text = CubiBlu.ToString();
        numBlu1.Text = numBlu.Text;
        numTot.Text = (CubiRossi + CubiBlu + CubiVerdi).ToString();
    }
    else { }
}
```

```
private void btn_PiuVerde_Click(object sender, EventArgs e)
{
    if (CubiVerdi < 3)
    {
        CubiVerdi = CubiVerdi + 1;
        numVerdi.Text = CubiVerdi.ToString();
        numVerdi1.Text = numVerdi.Text;
        numTot.Text = (CubiRossi + CubiBlu + CubiVerdi).ToString();
    }
    else { }
}

1 riferimento
private void btn_MenoVerde_Click(object sender, EventArgs e)
{
    if (CubiVerdi > 0)
    {
        CubiVerdi = CubiVerdi - 1;
        numVerdi.Text = CubiVerdi.ToString();
        numVerdi1.Text = numVerdi.Text;
        numTot.Text = (CubiRossi + CubiBlu + CubiVerdi).ToString();
    }
    else { }
}

private void bin_Rossi_Click(object sender, EventArgs e)
{
    CubiRossi = 0;
    numRossi1.Text = "0";
    numRossi.Text = "0";
    numTot.Text = (CubiRossi + CubiBlu + CubiVerdi).ToString();
}

1 riferimento
private void bin_Blu_Click(object sender, EventArgs e)
{
    CubiBlu = 0;
    numBlu1.Text = "0";
    numBlu.Text = "0";
    numTot.Text = (CubiRossi + CubiBlu + CubiVerdi).ToString();
}

1 riferimento
private void bin_Verdi_Click(object sender, EventArgs e)
{
    CubiVerdi = 0;
    numVerdi1.Text = "0";
    numVerdi.Text = "0";
    numTot.Text = (CubiRossi + CubiBlu + CubiVerdi).ToString();
}
```

```
private void bin_Tot_Click(object sender, EventArgs e)
{
    CubiRossi = 0;
    numRossi1.Text = "0";
    numRossi.Text = "0";
    CubiBlu = 0;
    numBlu1.Text = "0";
    numBlu.Text = "0";
    CubiVerdi = 0;
    numVerdi1.Text = "0";
    numVerdi.Text = "0";
    numTot.Text = (CubiRossi + CubiBlu + CubiVerdi).ToString();
}

1 riferimento
private void cmd_Shop_Click(object sender, EventArgs e)
{
    string comando;
    comando = "R" + CubiRossi.ToString() + "B" + CubiBlu.ToString() + "V" + CubiVerdi.ToString() + Environment.NewLine;
    serialPort1.Open();
    serialPort1.Write(comando);
    serialPort1.Close();
}
```

Listato 8) WebPage → link: mycolorcascade.altervista.org

- 1) L'HTML (struttura del sito):

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Download My App</title>
    <style>···
    </style>
</head>
<body>
    <h1>COLORCASCADE</h1>
    <a href="COLOR-CASCADE.zip"><button type="button">
        <i>
            <svg viewBox="0 0 24 24" fill="none" xmlns="http://www.w3.org/2000/svg">
                <path d="M12 2C12.5523 2 13 2.44772 13 3V13.5858L15.2929 11.2929C15.6834
                <path d="M4 15C4.55228 15 5 15.4477 5 16V18H19V16C19 15.4477 19.4477 15
            </svg>
        </i>
    </button></a>
    <aside>Click to download</aside>
</body>
</html>
```

- 2) CSS (estetica del sito):

```
7       <style>
8   >   * {…
10      }
11      body,html {
12      overflow: hidden;
13      }
14      body {
15      position: relative;
16      display: flex;
17      flex-direction: column;
18      align-items: center;
19      justify-content: center;
20      width: 100vw;
21      height: 100vh;
22      margin: 0;
23      background: black;
24      color: #ccc;
25      }
26      h1 {
27      position: absolute;
28      top: 5%;
29      left: 50%;
30      transform: translateX(-50%);
31      font: normal normal 900 54pt / 2cap "Architects Daughter", cursive;
32      color: #ccc;
33      margin: 0;
34      z-index: 1;
35      transition: all 0.3s ease;
36      text-shadow: 0 0 10px rgba(255, 59, 242, 0.7),
37                   0 0 20px rgba(255, 59, 242, 0.5),
38                   0 0 30px rgba(255, 59, 242, 0.3);
39      }
```

```
40      h1::after {
41      content: '';
42      position: absolute;
43      left: 0;
44      bottom: -10px;
45      width: 100%;
46      height: 4px;
47      background: linear-gradient(
48          90deg,
49          #ff0000,
50          #ff7f00,
51          #ffff00,
52          #00ff00,
53          #0000ff,
54          #4b0082,
55          #8b00ff,
56          #ff0000,
57          #ff7f00,
58          #ffff00,
59          #00ff00,
60          #0000ff,
61          #4b0082,
62          #8b00ff
63      );
64      background-size: 200% 100%;
65      animation: gradient 5s infinite linear;
66      box-shadow: 0 0 10px rgba(255, 59, 242, 0.7),
67                  0 0 20px rgba(255, 59, 242, 0.5),
68                  0 0 30px rgba(255, 59, 242, 0.3);
69      border-radius: 2px;
70      }
71  >   @keyframes gradient {···
78      }
79  >   button {···
113     }
114 >   button:hover {···
122     }
123 >   button:active {···
126     }
127 >   button::before {···
136     }
137 >   button:hover::before {···
142     }
143 >   button:active::before {···
147     }
148 >   button::after {···
159     }
```

```
171 >   button:active::after {···
175     }
176 >   button > svg {···
186     }
187 >   button i {···
194     }
195 >   button i svg {···
201     }
202 >   button:hover i svg {···
204     }
205     @keyframes shake {
206         0% {
207             transform: rotate(0deg);
208         }
209         25% {
210             transform: rotate(15deg);
211         }
212         75% {
213             transform: rotate(-15deg);
214         }
215     }
216 >   button span {···
232     }
233 >   button:hover span {···
236     }
237 >   button span p {···
243     }
244 >   button span svg {···
250     }
251 >   aside {···
257     }
258 >   button:hover ~ aside {···
261     }
262     @media (max-height: 900px) {
263         h1{
264             font: normal normal 900 26pt / 2cap "Architects Daughter", cursive;
265         }
266     }
267     @media (max-height: 270px) {
268         h1{
269             font: normal normal 900 14pt / 2cap "Architects Daughter", cursive;
270         }
271         aside {
272             display: none;
273         }
274     }
```

Conclusione:

Tra le problematiche riscontrate durante lo svolgimento di questo progetto troviamo sicuramente problematiche a livello hardware, infatti le parti stampate in 3D non assicurano una grande resistenza a vantaggio però della loro leggerezza.

Inoltre troviamo la problematica della qualità dei componenti, i servomotori utilizzati sono spesso imprecisi; Ma nonostante piccole inconvenienze il progetto è giunto al termine.

Ho inoltre deciso di apportare una piccola modifica alla stampa della spalla per renderla più stabile. Ho abbellito il tutto come sono riuscito utilizzando piccole stampe funzionali che si possono trovare ai bordi dell'asse di legno.