UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN INGEGNERIA ELETTRONICA

"Braccio robotico con controllo remoto tramite internet"

Relatore: Prof. / Dott. Meneghini Matteo

Laureando/a: De Boni Damiano

Correlatore: Prof,/Dott……

ANNO ACCADEMICO 2022 – 2023
Data di laurea 27/09/2023

INDICE

1. Introduzione 3

2. Descrizione componenti utilizzati 3
   2.1. Power supply 3
      2.1.1. Batterie agli ioni di litio (Li-Ion) 3
      2.1.2. Buck converter 4
   2.2. ESP32 6
   2.3. Arduino 7
      2.3.1. Motori DC e L298N 8
      2.3.2. Servo motori 9

3. Progetto 10
   3.1. Batteria 11
   3.2. ESP32-WROVER Board 11
   3.3. Arduino Nano 14
      3.3.1. Servo motori 15
      3.3.2. Motori DC 18
      3.3.3. Monitoraggio batteria 19

4. Verifica del funzionamento 20

5. Conclusioni 21

6. Codice 21
   6.1. ESP32-WROVER-MODULE 21
   6.2. Arduino Nano 36

7. Bibliografia e sitografia 45

8. Ringraziamenti 45

1. Introduzione

Il progetto consiste di un braccio robotico fatto con angolari in alluminio e azionato da Servo motori e posizionato su un piano con ruote che permette di renderlo mobile. L'alimentazione tramite batterie permette al braccio di non essere vincolato da cavi e un microcontrollore con videocamera connesso tramite WiFi permette di trasmettere in diretta la visuale del braccio e consente il controllo dei vari movimenti da remoto tramite internet su un qualsiasi dispositivo connesso quale computer o smartphone.

Gli obbiettivi per questo progetto sono: fare in modo che il braccio riesca a reggere il proprio peso senza crollare su sé stesso e che possa sollevare oggetti di piccole dimensioni, garantire che alla massima estensione i motori non cedano e che la base non si rovesci, permettere lo spostamento tramite ruote dell'intera struttura, ottenere una autonomia di almeno 30 minuti prima di dover ricaricare la batteria e che il tutto sia reattivo ai comandi mandati tramite internet.

2. Descrizione componenti utilizzati

2.1. Power supply

Per rendere il braccio libero di muoversi nell'ambiente senza doversi trascinare un cavo per l'alimentazione, utilizzo delle batterie ricaricabili agli ioni di litio (Li-Ion) con il relativo circuito di protezione e di ricarica. La tensione delle batterie viene poi abbassata da un buck converter impostato in modo da garantire una tensione in uscita di 5[V] e scelto in modo tale che possa erogare una corrente sufficiente per permettere agli altri componenti presenti di svolgere le loro funzioni.

2.1.1. Batterie agli ioni di litio (Li-Ion)

L'accumulatore agli ioni di litio è un tipo di batteria ricaricabile che utilizza la riduzione reversibile degli ioni di litio per immagazzinare energia. Queste batterie utilizzano un composto di litio sul catodo e grafite o titanio di litio sull'anodo. Possono essere costituite in una vasta gamma di forme e dimensioni, in modo da riempire gli spazzi disponibili nei dispositivi che li utilizzano. Sono anche più leggere degli equivalenti fabbricati con altri componenti chimici, questo perché gli ioni di litio hanno una densità di carica molto elevata, la più alta di tutti gli ioni che si sviluppano naturalmente. Questi accumulatori hanno inoltre un basso tasso di autoscarica, circa il 5% mensile rispetto all'oltre 30% mensile delle comuni batterie all'idruro metallico di nichel (NiMH) e al 20% mensile di quelle al nichel-cadmio, ma come altri tipi di batterie soffrono di una lenta perdita permanente di capacità.

La corrente massima che può essere prelevata in continuo dipende sia dalla capacità, sia dal tipo di carico. Per esempio, nei dispositivi dove sono richieste correnti elevate, le batterie agli ioni di litio anziché mostrare una graduale diminuzione della durata d'uso del dispositivo, possono smettere di funzionare bruscamente; al contrario i dispositivi che richiedono bassa potenza possono generalmente sfruttare l'intero ciclo di vita della batteria. Per evitare danni irreversibili un elemento agli ioni di litio non va mai scaricato sotto una certa tensione, di conseguenza tutti i sistemi che lo utilizzano sono equipaggiati con un circuito che spegne il dispositivo quando la batteria viene scaricata sotto la soglia predefinita.
Le batterie agli ioni di litio hanno una tensione nominale di circa 3,6[V] che rappresenta il valore medio tra la tensione a piena carica di 4,2[V] e quella oltre la quale non deve scendere di 3,0[V]. La carica si effettua a tensione costante con limitazione di corrente (processo di carica CC-CV, costant current-costant voltage), questo significa che la carica avviene a corrente costante fino al raggiungimento della tensione di 4,2[V] (per sicurezza, di solito è inferiore di alcune decine di mV a tale valore), dopodiché continua a tensione costante finché la corrente diventa quasi nulla. Il tempo di ricarica dipende dalla capacità ampere-ora della batteria e dalla corrente erogata dal caricabatterie, che in ogni caso non deve superare 1/10 della corrente di picco erogabile.

Per questo progetto ho deciso di utilizzare quattro batterie INR18/65 (tali numeri indicano le dimensioni in mm rispettivamente del diametro (18mm) e della lunghezza (65mm)) dalla capacità di 2000mAh in una configurazione 2S2P (due batterie in serie e due batterie in parallelo) per avere una batteria dalla tensione nominale di 8,4[V] e con una capacità di 4000mAh. Per garantire il corretto ciclo di carica/scarica della batteria utilizzo un BMS (Battery Management System) progettato per una configurazione 2S che mi garantisce che entrambe le celle in serie abbiano la stessa tensione e non ci sia una delle due che viene caricata/scaricata più dell'altra (possibile perché le celle presentano sempre delle piccole differenze per via del processo costruttivo), cosa che potrebbe portare a grandi correnti tra una cella e l'altra e di conseguenza alla loro distruzione. Per ricaricare il pacco batteria è presente un circuito che, ricevendo in ingresso una tensione di 5[V] tramite connettore USB di tipo C, fornisce in uscita una adeguata tensione per la ricarica delle batterie e che limita la corrente per la ricarica (scelta a 2A con la corrente consigliata per la ricarica di 1A*2 celle in parallelo), garantendo un corretto ciclo di carica e di conseguenza una maggiore longevità alle batterie.

2.1.2. Buck Converter

Il Buck Converter (detto anche Step-down converter) è un particolare tipo di convertitore di tensione DC-DC che converte una certa tensione in ingresso in una tensione stabile in uscita di

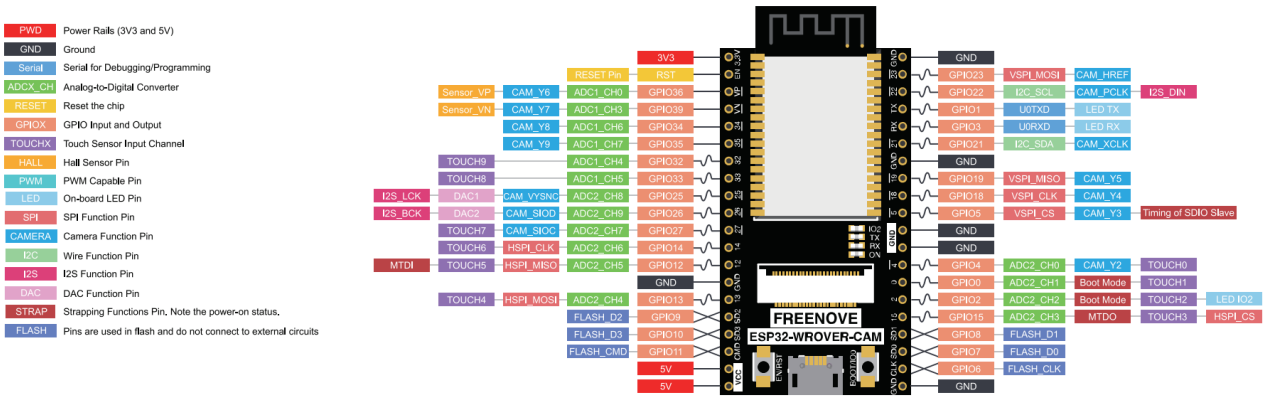
valore minore. Questo convertitore è di tipo switching ovvero è presente un interruttore (solitamente viene utilizzato un transistor MOS per via della sua bassa resistenza in conduzione che genera quindi basse perdite di potenza) che, mediante un circuito di pilotaggio, avendo una tensione in ingresso genera un'onda quadra in uscita il cui valor medio rappresenta la tensione in uscita desiderata data dalla formula V_{out}= Duty-Cycle*V_{in}. Quest'onda quadra oltre ad una componente continua corrispondente al valor medio, contiene anche una serie di armoniche dovute alla frequenza di switching del MOS che vanno filtrate per avere una tensione stabile in uscita. Viene quindi utilizzato un filtro passa basso del secondo ordine di tipo LC posto ad una frequenza minore rispetto a quella di switching (ad esempio si progetta il filtro per avere una pulsazione di risonanza ad una decade prima della F_s). Ciò permette di avere una oscillazione della tensione di uscita molto bassa e che può essere mitigata ulteriormente aumentando la frequenza di switching o utilizzando L e C maggiori anche se l'ultima è una scelta poco utilizzata perché la cosa porterebbe ad avere componenti più grandi a livello fisico e, soprattutto per quanto concerne L, il costo di tali componenti sarebbe più elevato e sarebbero presenti componenti parassite maggiori. Anche il primo metodo comporta degli svantaggi, maggiore sarà la F_s, maggiore sarà la potenza che si dissipa sul MOS durante la commutazione. Per mitigare tale effetto si usa un circuito di pilotaggio per il transistor che si occupa di caricare/scaricare il più velocemente possibile le capacità parassite, permettendo una commutazione più veloce e una minore potenza dissipata. Per permettere il circolo della corrente conservata nell'induttore quando il MOS è spento viene utilizzato un diodo di ricircolo che evita bruschi picchi di tensione in uscita dovuti all'induttore, ed anche il diodo deve essere veloce nel compiere le commutazioni.
Poiché si vuole che la tensione in uscita sia stabile indipendentemente dalle possibili variazioni della tensione in ingresso, si utilizza un sistema di feedback che monitora sia la tensione che la corrente in uscita. Un filtro PID crea un riferimento stabile e veloce rispetto ai cambiamenti della tensione che viene confrontato da un comparatore con un'onda a dente di sega, generando un segnale ad onda quadra che, mandato in ingresso al circuito di pilotaggio del MOS, controllerà il Duty-Cycle al quale commuta il transistor permettendo di avere in uscita una tensione stabile rispetto ai cambiamenti di quella in ingresso.

Per il progetto utilizzo un buck converter che accetta in ingresso una tensione compresa tra 6[V] e 40[V], con la possibilità di erogare in uscita una corrente massima di 20[A] ed avendo una corrente consigliata di 15[A], ampliamente sufficiente per alimentare i vari componenti. Sono inoltre presenti due potenziometri per regolare la tensione e la corrente massima in uscita e un interruttore che permette di disattivare il buck converter e di conseguenza spegnere il braccio.

2.2. ESP32

ESP32 è un microcontrollore sviluppato dalla Espressif System dai costi ridotti e una ottima capacità di calcolo in rapporto al prezzo. Una delle sue principali caratteristiche è difatti l'elaborazione parallela, che gli consente di gestire più processi contemporaneamente come la connessione WiFi, l'elaborazione di dati e il controllo di dispositivi esterni tutto allo stesso tempo. Un'altra sua caratteristica è la capacità di funzionare come dispositivo dual-mode, ovvero può essere utilizzato sia come un dispositivo autonomo che come un modulo slave in un sistema più grande, rendendolo particolarmente utile per sistemi di tipo IoT (Internet of Things). L'ESP32 è dotato di un modulo WiFi integrato (supporta le reti 2.4Ghz e 5GHz) e di un modulo Bluetooth (4.2 e 5.0) che lo rende molto versatile per la creazione di dispositivi connessi permettendogli di comunicare con un server o con un'applicazione mobile per effettuare controlli e monitoraggi da remoto. Possiede inoltre numerose interfacce integrate quali una serie di sensori, una porta seriale, un'interfaccia I2C, un'interfaccia SPI, un'interfaccia UART e un'interfaccia I2S. È inoltre possibile programmare la scheda sia tramite Micropython che tramite l'ambiente di sviluppo di Arduino.

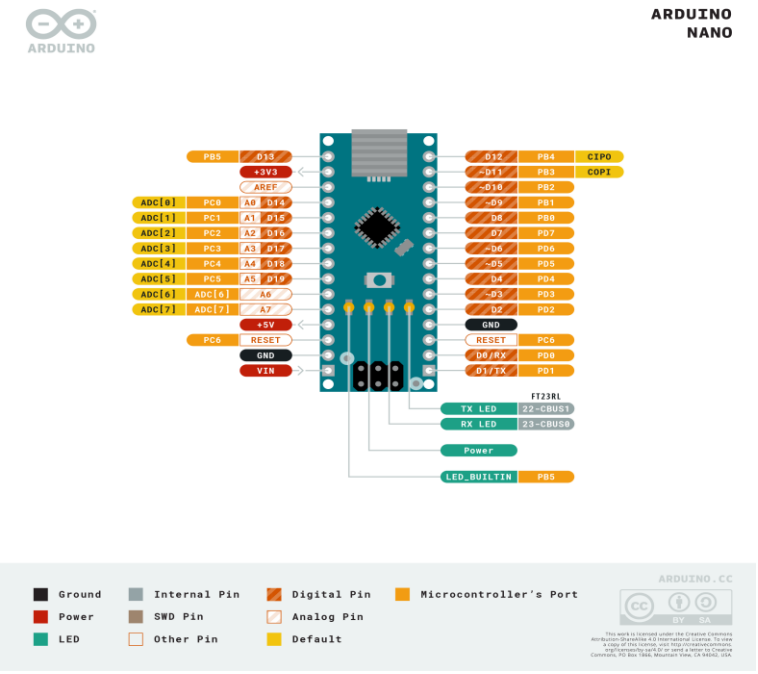
Per questo progetto utilizzo una scheda ESP32-WROVER CAM Board della Freenove funzionante con una alimentazione a 5[V] (che viene stabilizzata a 3,3[V] da un regolatore presente sulla scheda), con una PSRAM da 4[MB] e sulla quale è presente un connettore per utilizzare una videocamera OV2640 da 2 Megapixel. Sfruttando la connettività WiFi e la videocamera posso trasmettere le immagini tramite streaming video su un server dal quale posso configurare una serie di comandi che verranno ricevuti dall'ESP32. Un problema è che la maggior parte dei pins della scheda è utilizzato dalla videocamera il che mi porta a non avere abbastanza pins per tutte le periferiche. Verrà quindi utilizzato un Arduino Nano per gestire le periferiche e i comandi verranno trasmessi tra l'ESP32 e l'Arduino tramite comunicazione UART.



2.3. Arduino

Arduino è una piattaforma open-source basata su hardware e software facili da usare. Le schede Arduino sono capaci di leggere degli input (sensori, pulsanti ecc.) e trasformarli in degli output (accendere LED, azionare motori ecc.). Per dare istruzioni alla scheda si usa il linguaggio di programmazione di Arduino e il corrispettivo IDE. Grazie alla sua facilità di utilizzo, Arduino è facile da usare per i neofiti ma abbastanza flessibile per poter essere usato anche dagli esperti. I vantaggi che rendono Arduino un valido microcontrollore sono: il basso costo delle schede, la possibilità di programmarlo su ogni computer attraverso un ambiente di programmazione chiaro e semplice, inoltre, essendo un progetto open-source è possibile modificare ed ampliare sia il lato hardware che quello software (con, ad esempio, librerie aggiuntive) per renderlo adatto ad una vasta gamma di utilizzi.

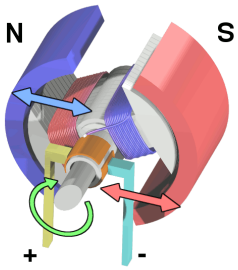
Per questo progetto utilizzo un Arduino Nano, una scheda basata sull'ATmega328, lo stesso dell'Arduino UNO ma che, a differenza di quest'ultimo, presenta dimensioni più compatte e due ingressi esclusivamente analogici in più. Funziona con una tensione di 5[V], ha una memoria flash di 32KB ed una SRAM di 2KB, un clock a 16MHz, 22 I/O pins con 8 pins con la funzione di ingresso analogico (ADC) e 6 con una uscita PWM. L'Arduino Nano ha a disposizione diversi protocolli di comunicazione per scambiare dati con un computer, un altro Arduino o altri microcontrollori. L'ATmega328 presenta una comunicazione seriale UART TTL (5[V]) presente sui pins 0 (RX) e 1 (TX) che utilizzerò per la comunicazione con l'ESP32. Supporta inoltre i protocolli I2C e SPI.



Arduino Nano pinout

2.3.1. Motori DC e L289N

Un motore in corrente continua (DC) ha una parte che gira detta rotore o armatura composta da due avvolgimenti di filo, e una parte che genera un campo magnetico fisso (solitamente composta da due magneti permanenti) detta statore. Un interruttore rotante detto commutatore o collettore a spazzole inverte due volte ad ogni giro la direzione della corrente elettrica che percorre i due avvolgimenti generando un campo magnetico che entra ed esce dalle parti rotanti dell'armatura. Nascono quindi forze di attrazione e repulsione con i magneti permanenti che fanno ruotare il rotore, spostando i contatti che interagiscono con le spazzole (che sono fisse sullo statore) e portando ad un cambio nella direzione della corrente che fa si che il processo si ripeta. A seconda di come vengono posizionati i contatti positivo e negativo collegati alle spazzole, la direzione di rotazione del motore cambia. Richiedendo solo una tensione continua per il funzionamento questo tipo di motore è molto semplice da comandare ma presenta tuttavia alcuni svantaggi: le spazzole pongono un limite alla massima velocità di rotazione e, essendoci un contatto meccanico tra esse ed il rotore, sono soggette ad usura. Un altro problema è che gli avvolgimenti appesantiscono il rotore (aumenta il momento di inerzia): se il motore deve rispondere con rapidità e precisione (come avviene nelle automazioni industriali e nella robotica) il controllo diventa più complesso; per piccole potenze (da 1 a 2000W) e servocontrolli a volte si usano particolari tipi di motore con rotore con avvolgimenti a forma di bicchiere e privo del nucleo di ferro che portano ad avere una bassa inerzia e rendimento elettrico più elevato.



Rappresentazione dell'interno di un motore DC

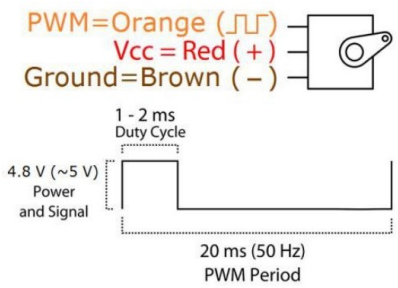
Per questo progetto utilizzo due motori DC con un sistema di ingranaggi che aumentano la coppia e spostano l'asse di rotazione permettendo di utilizzare una ruota senza occupare troppo spazio, i motori sono alimentati a 5[V]. Per fornire una corrente sufficiente al funzionamento (maggiore di quella erogata da un microcontrollore) e controllare la direzione di rotazione utilizzo un modulo basato sul L298N, che è composto da due ponti H (H-bridge), uno per motore e con 4 ingressi che stabiliscono la direzione di rotazione dei due motori in modo indipendente. È possibile dare in ingresso su due ulteriori pin un segnale PWM per regolare la velocità di rotazione del motore.

Motori DC con ruote

Modulo L298N

2.3.2. Servo motori

Il Servo motore è costituito da un motore DC, un circuito di controllo, degli ingranaggi che servono a ridurre la velocità del motore ed un potenziometro. La sua caratteristica è quella di effettuare rotazioni ben precise di oggetti o elementi. Il potenziometro presente al suo interno serve a stabilire in modo preciso il numero di gradi a cui far ruotare il Servo motore e mantenere tale rotazione ed insieme al motore ed al circuito di controllo formano un sistema di feedback ad anello chiuso. Sul Servo sono presenti tre cavi: due per l'alimentazione (solitamente compresa tra 4,8[V] e 6[V]) ed uno per il segnale che stabilisce l'angolo di rotazione. Il segnale è di tipo PWM con un periodo T=20[ms] e il cui T_{on} stabilisce l'angolo a cui si deve posizionare il Servo. Il range di movimento di un Servo è solitamente di 180° con 0° rappresentati da un impulso di 1[ms] e i 180° da un impulso di 2[ms].



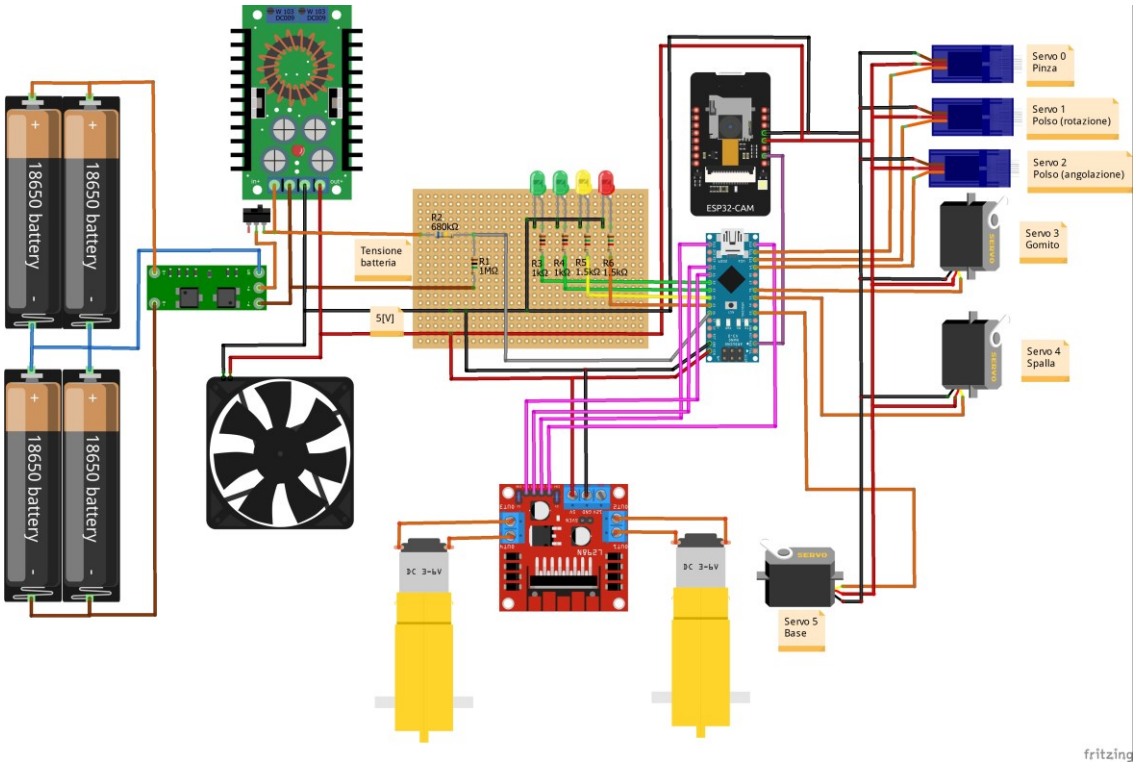
Collegamenti e segnale di controllo presi dal datasheet

Per questo progetto utilizzo due tipi di Servo motore: MG996R caratterizzato da dimensioni maggiori, ingranaggi in metallo ed una coppia di 9,4[Kg(f)*cm] a 4,8[V] ed SG90 dalle dimensioni ridotte, ingranaggi in plastica e coppia di 1,8[Kg(f)*cm] a 4,8[V].

MG996R

SG90

3. Progetto



Schema dei collegamenti del progetto (N.B. alcuni componenti non sono esattamente quelli utilizzati ma svolgono la stessa funzione ed hanno gli stessi collegamenti)

3.1. Batteria

Per alimentare il progetto utilizzo quattro batterie in configurazione 2S2P che mi danno una capacità di 4000[mAh] ed una tensione nominale di 8,4[V] per una energia complessiva nominale di 33,6[Wh]. Le batterie sono collegate ad un BMS che garantisce il corretto ciclo di scarica e carica delle batterie proteggendole da overcharge, overdischarge e cortocircuiti. Il positivo del pacco batteria si collega alla piazzola con la scritta 8,4[V] (o B+), il negativo a quella con la scritta 0[V] (o B-) e il collegamento intermedio tra le due batterie in serie va collegato alla piazzola 4,2[V] (o BM). L'uscita/ ingresso per la ricarica è presente sulle piazzole + e – (o P+ e P-) che rappresentano anche la polarità. La corrente di ricarica consigliata per queste batterie è di 1[A] e, dato che ho due batterie in parallelo, utilizzo una scheda che riceve in ingresso tramite un connettore usb di tipo C una tensione di 5[V] e fornisce in uscita una corrente massima di 2[A] e una tensione di 8,4[V] che va collegata sulle piazzole + e – del BMS.
Un partitore resistivo riceve in ingresso la tensione della batteria e la porta in un range da 3,4524[V] a 5[V] che può quindi essere letto da un ingresso analogico di Arduino per fornire una stima della carica restante. I valori sono stati scelti alti per non avere una eccessiva corrente di leakage evitando quindi che la batteria si scarichi eccessivamente quando il braccio non è attivo.

$$V_{MAX} = 5[V] = 8{,}4[V] \cdot \frac{R_1}{R_1+R_2} \Rightarrow \frac{R_2}{R_1} = 0{,}68 \Rightarrow R_1 = 1[M\Omega], R_2 = 680[K\Omega]$$

$$V_{min} = 5{,}8[V] \cdot \frac{R_1}{R_1 + R_2} = 3{,}4524[V]$$

$$I_{leak} = \frac{V_{Max_{bat}}}{R_1 + R_2} = 5[\mu A]$$

Un convertitore Buck riceve in ingresso la tensione della batteria e la porta stabile a 5[V] per poter alimentare i vari componenti del progetto. Un interruttore presente sul Buck permette di scollegare la batteria per spegnere il braccio.
Il tutto è posizionato all'interno di una scatola per progetti con anche il driver per i motori DC (L298N) e il driver per lo stepper motor e una ventola a 5[V] si occupa di creare un costante flusso d'aria per evitare il surriscaldamento dei componenti (in particolar modo quello del Buck).

3.2. ESP32-WROVER Board

L'ESP32-WROVER dispone di una videocamera e, grazie alla sua connettività WiFi, può trasmettere in streaming tramite internet il video. Dalla stessa pagina web è possibile impartire i comandi che verranno poi inviati all'Arduino tramite UART e che indicano le varie azioni che

devono eseguire i motori. I comandi vengono inviati come numero intero, quindi esiste una partizione dei numeri per indicare il motore che deve compiere una azione e il valore ad essa associato.

Comando (intervallo)	Istruzione
001%099	Rotazione Servo 0
100%199	Rotazione Servo 1
200%299	Rotazione Servo 2
300%399	Rotazione Servo 3
400%499	Rotazione Servo 4
500%699	Rotazione Servo 5
700%704	Movimento motori DC

I comandi sono fatti in modo che la prima cifra indichi qual è il motore destinatario, mentre le successive due forniscono l'istruzione che verrà adattata da Arduino.

Per poter fornire i comandi vengono utilizzati sei slider in cui ciascuno rappresenta la rotazione di un determinato motore, e cinque pulsanti che controllano il movimento dettato dai motori DC. Tutto ciò e realizzato nella struttura della pagina HTML che verrà visualizzata dal dispositivo che si userà per il controllo (computer, smartphone ecc.).

```
<body>
  <h1>ESP32-CAM Robot</h1>
  <img src="" id="photo" >
  <p>
    <input type="range" onchange="updateServo0(this)" id="servo0" min="1" max="99" value="servoVal" step="1" class="slider">
  </p>
  <p>
    <input type="range" onchange="updateServo1(this)" id="servo1" min="100" max="199" value="servoVal" step="1" class="slider">
  </p>
  <p>
    <input type="range" onchange="updateServo2(this)" id="servo2" min="200" max="299" value="servoVal" step="1" class="slider">
  </p>
  <p>
    <input type="range" onchange="updateServo3(this)" id="servo3" min="300" max="399" value="servoVal" step="1" class="slider">
  </p>
  <p>
    <input type="range" onchange="updateServo4(this)" id="servo4" min="400" max="499" value="servoVal" step="1" class="slider">
  </p>
  <p>
    <input type="range" onchange="updateServo5(this)" id="servo5" min="500" max="699" value="servoVal" step="1" class="slider">
  </p>
```

Corpo della pagina con indicati la trasmissione video e i 6 slider con rispettivi valori

```
<table>
  <tr><td colspan="3" align="center">
  <button class="button" onmousedown="updateMovement('701');" ontouchstart="updateMovement('701');" onmouseup="updateMovement('700');" ontouchend="updateMovement('700');">Forward</button></td></tr>
  <tr><td align="center">
  <button class="button" onmousedown="updateMovement('704');" ontouchstart="updateMovement('704');" onmouseup="updateMovement('700');" ontouchend="updateMovement('700');">Left</button></td>
  <td align="center">
  <button class="button" onmousedown="updateMovement('700');" ontouchstart="updateMovement('700');">Stop</button></td>
  <td align="center">
  <button class="button" onmousedown="updateMovement('703');" ontouchstart="updateMovement('703');" onmouseup="updateMovement('700');" ontouchend="updateMovement('700');">Right</button></td></tr>
  <tr><td colspan="3" align="center">
  <button class="button" onmousedown="updateMovement('702');" ontouchstart="updateMovement('702');" onmouseup="updateMovement('700');" ontouchend="updateMovement('700');">Backward</button></td></tr>
</table>
```

Questi comandi generano una richiesta che viene successivamente inviata all'ESP32. Le richieste per i Servo sono ottenute tramite Id della variabile, qui è riportata solo la prima ma non ci sono altre differenze oltre all'Id con le altre.

```
function updateServo0(servoVal) {
  var servoVal= document.getElementById("servo0").value;
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/action?var=servo&val=" + servoVal, true);
  xhr.send();
}
```

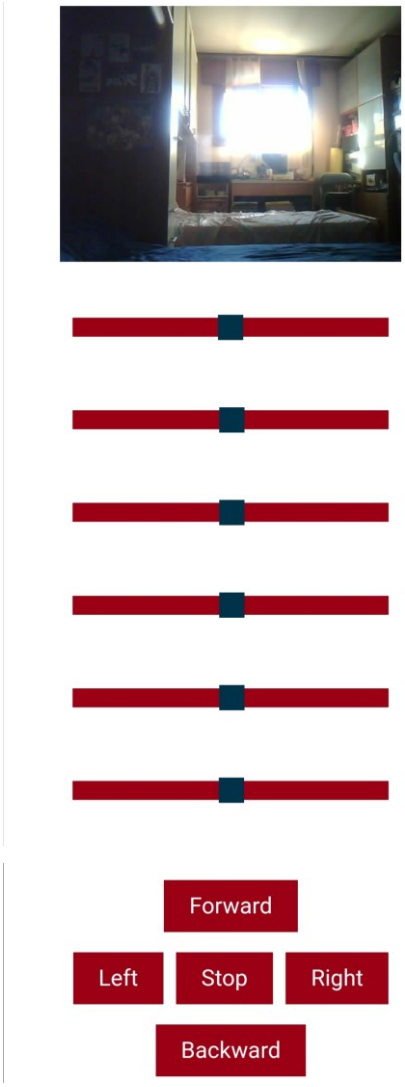
```
function updateMovement(x) {
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/action?var=mov&val=" + x, true);
  xhr.send();
}
```

Nella sezione cmd_handler() del codice l'ESP riceve la richiesta dalla pagina e la invia tramite Serial.println() all'Arduino che gestirà le varie periferiche.

```
if(!strcmp(variable, "mov")) {
  Serial.println(val);
}
else if(!strcmp(variable, "servo")) {
  Serial.println(val);
}
else {
  res = -1;
}
```

La gestione dello streaming video è stata presa dagli sketch di esempio della scheda.

La pagina su internet risulterà quindi in questo modo:



3.3. Arduino Nano

Il compito dell'Arduino Nano è quello di ricevere i comandi dall'ESP32 tramite comunicazione UART ed elaborarli per eseguire il corretto movimento dei motori.

Pins Arduino	Collegamenti
0 (RX)	ESP32 (TX)
1 (TX)	
2	
3 (PWM)	Servo 5
4	
5 (PWM)	Servo 4
6 (PWM)	Servo 3
7	

8	
9 (PWM)	Servo 2
10 (PWM)	Servo 1
11 (PWM)	Servo 0
12	IN1 (motore DC)
13	IN2 (motore DC)
A0	IN3 (motore DC)
A1	IN4 (motore DC)
A2	LED Verde
A3	LED Verde
A4	LED Giallo
A5	LED Rosso
A6	Partitore Batteria
A7	

Per distinguere l'azione a cui il comando inviato dall'ESP32 si riferisce viene modificata una variabile 'sel' per selezionare il motore che si vuole comandare.

```
void loop() {
  if(Serial.available()){

    //read the serial data
    val=Serial.parseInt();
   if(val!=0){
    //Serial.print("I recived: ");
    //Serial.println(val);

    sel=byte(val/100);
    //Serial.print("Selection: ");
    //Serial.println(sel);
```

Poiché i comandi presentano la prima cifra come selettore, dividendo il comando per cento e prendendo solo la parte intera con un cast a byte, riesco a isolare la prima cifra e a sapere il motore da selezionare.

3.3.1. Servo motori

All'inizio del codice, prima del void setup(), vengono creati grazie alla libreria <Servo.h> sei oggetti di tipo Servo, uno per ogni servo del progetto. Vengono inoltre definite le variabili per il filtro IIR che renderà più fluido il movimento dei Servo.

```
//servo
  Servo servo0; //create 6 servo object
  Servo servo1;
  Servo servo2;
  Servo servo3;
  Servo servo4;
  Servo servo5;

  //variable for the IIR filter
  float finalVal;
  float preVal0=700, preVal1=700, preVal2=700, preVal3=700, preVal4=700, preVal5=700;
  float smoothVal0, smoothVal1, smoothVal2, smoothVal3, smoothVal4, smoothVal5;
```

All'interno del setup viene specificato il pin PWM al quale sono collegati i segnali dei Servo. Per ottenere un range di movimento di circa 180°, il segnale di controllo deve andare da un T_{on} di 0,5[ms] a 2,5[ms] cercando di non utilizzare valori troppo vicini ad essi. Tramite la funzione writeMicroseconds() imposto la posizione iniziale dei servo.

```
//servo
  //connect the servo to pin x (PWM)
  //mg996R
  servo5.attach(3);
  servo4.attach(5);
  servo3.attach(6);
  //sg90
  servo2.attach(9);
  servo1.attach(10);
  servo0.attach(11);

  //set the initial servo position
  servo0.writeMicroseconds(preVal0);
  servo1.writeMicroseconds(preVal1);
  servo2.writeMicroseconds(preVal2);
  servo3.writeMicroseconds(preVal3);
  servo4.writeMicroseconds(preVal4);
  servo5.writeMicroseconds(preVal5);
```

All'interno del loop, dopo che la variabile 'sel' è stata aggiornata, se è minore o uguale a quattro indica che il comando è destinato ad un Servo (il servo 5 alla base ha un comando a parte) e il valore che indica la sua rotazione è ottenuto prendendo il resto dopo una divisione per cento, ottenendo quindi la seconda e terza cifra del comando. Il valore viene poi mappato per avere il valore in microsecondi che rappresenta l'angolo di rotazione che deve raggiungere il Servo che viene selezionato tramite una funzione switch(). I Servo raggiungerebbero l'angolo finale il più velocemente possibile ma ciò darebbe vita ad un movimento molto meccanico oltre che a forze che farebbero scuotere la struttura e potrebbero portare alla rottura dei Servo. All'interno di un ciclo while() viene quindi applicato un filtraggio tramite filtro IIR al valore in microsecondi che fa quindi eseguire ai Servo un movimento più fluido e controllato che decelera verso la posizione di arrivo e

non crea instabilità meccanica sulla struttura. Per non restare troppi cicli sulla parte finale del filtraggio il ciclo si blocca 25 numeri prima che rende il movimento molto preciso rispetto alla richiesta e non richiede un tempo eccessivo per il suo raggiungimento. Le variabili vengono salvate per poter essere utilizzate per il successivo movimento del Servo.

```
//servo selection
if(sel <= 4){

  val=val%100;

  //convert the value to microseconds for the servo
  finalVal= map(val, 0, 99, 700, 2000);
  //Serial.print("final value: ");
  //Serial.println(finalVal);

  switch(sel){
    case 0:
      //create a smooth movment for the servo using a IIR filter
      while((smoothVal0 < (finalVal-25)) || (smoothVal0 > (finalVal+25))){

        //IIR filter y(k)= (x(k) * b) + (y(k-1) * (1-b))
        smoothVal0 = (finalVal * 0.05) + (preVal0 * 0.95);
        preVal0 = smoothVal0; //y(k-1)

        servo0.writeMicroseconds(smoothVal0);
        delay(5);
      }
      //Serial.print("servo 0 smoothed value: ");
      //Serial.println(smoothVal0);
      //Serial.println(" ");
    break;
```

```
//servo5 selection
else if(sel==5 || sel==6){
  //convert the value
  finalVal=map(val, 500, 699, 700, 2000);

  //create a smooth movment for the servo using a IIR filter
  while((smoothVal5 < (finalVal-25)) || (smoothVal5 > (finalVal+25))){

    //IIR filter y(k)= (x(k) * b) + (y(k-1) * (1-b))
    smoothVal5 = (finalVal * 0.05) + (preVal5 * 0.95);
    preVal5 = smoothVal5; //y(k-1)

    servo5.writeMicroseconds(smoothVal5);
    delay(5);
  }
  //Serial.print("servo 5 smoothed value: ");
  //Serial.println(smoothVal5);
  //Serial.println(" ");

}
```

Il servo numero 5 ha un range maggiore per garantire maggiore precisione del movimento alla base che risulterebbe grossolano per via del movimento delle ruote

3.3.2. Motori DC

Per i motori DC vengono definiti i pins corrispondenti agli ingressi dell'L298N con, per semplicità di scrittura per le istruzioni successive, la divisione tra pins per il motore A e quelli per il motore B.

```
//DC motor
  //motor A
  #define IN1 12
  #define IN2 13
  //motor B
  #define IN3 A0
  #define IN4 A1
```

All'interno del setup vengono definiti i pins come uscite e portandoli tutti a 0[V] ci si assicura che i motori siano inizialmente spenti.

```
//DC motor
  //initialize the motor pin
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
```

All'interno del loop, dopo essere stati selezionati, il valore che specifica il movimento dei due motori è ottenuto tramite il resto della divisione per cento e, per ognuno dei cinque possibili valori, una combinazione delle direzioni di rotazione dei due motori determina il movimento finale con il "destra" e "sinistra" determinati da una rotazione sul posto dell'intera stuttura.

```
//DC motor selection
else if(sel==7){
  val=val%100;
  //stop

  if(val==0){
    //motor A
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    //motor B
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    //Serial.println("Stop");
  }

  //avanti
  else if(val==1){
    //motor A
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    //motor B
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    //Serial.println("Avanti");
  }
```

```
//indietro
else if(val==2){
  //motor A
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  //motor B
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  //Serial.println("Indietro");
}

//destra
else if(val==3){
  //motor A
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  //motior B
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  //Serial.println("Destra");
}

//sinistra
else if(val==4){
  //motor A
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  //motor B
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  //Serial.println("Sinistra");
}
```

3.3.3. Monitoraggio batteria

Per verificare il livello di carica della batteria utilizzo l'ADC sul pin A6 di Arduino e visualizzo lo stato della batteria mediante quattro LED (2 verdi, 1 giallo, 1 rosso) che rappresentano approssimativamente le fasce di carica dal 100% al 75%, dal 74% al 50%, dal 50% al 25% e dal 25% allo 0% con lo 0% rappresentato dallo spegnimento di tutti i LED. Vengono quindi definiti i pin dell'ADC e dei LED che verranno salvati all'interno di un array. Vengono anche definite delle variabili di supporto per i cicli.

```
//battery
  #define bat A6 //ADC
  //led pin indicating the charge level
  #define LED75 A2
  #define LED50 A3
  #define LED25 A4
  #define LED00 A5
  byte LED[4]={LED00, LED25, LED50, LED75};

  byte i, j;
```

All'interno del setup i LED vengono tutti accesi ed eventualmente spendi partendo da quello che rappresenta la carica massima fino al raggiungimento della corretta visualizzazione dello stato di carica. Ciò avviene mappando su una variabile j il valore in tensione della batteria (707=5,8[V]=2,9[V]*2 che è il valore di tensione sotto il quale non bisognerebbe scendere). Il valore viene mappato tra 0 e 5 per garantire la visualizzazione della fascia di carica dal 100% al 75% tramite il LED che altrimenti sarebbe acceso solo quando la carica è esattamente 100% e non fornirebbe ulteriori informazioni.

```
//battery
  //turn on the LED
  for(i=0; i<4; i++){
    pinMode(LED[i], OUTPUT);
    digitalWrite(LED[i], HIGH);
  }
  //turn off the LED to undicate the charge level
  j= map(analogRead(bat), 707, 1023, 0, 5);
  for(i=3; i>=j; i--){
    digitalWrite(LED[i], LOW);
  }
```

All'interno del loop viene effettuato un aggiornamento del livello della batteria ogni minuto tramite la funzione millis() e una variabile che memorizza il valore dell'ultimo controllo. Vengono quindi spenti gli eventuali LED per visualizzare il livello di carica corretto.

```
//battery level indicator (check the voltage every minute)
if(millis()-preMillis>=60000){
  preMillis=millis();
  j= map(analogRead(bat), 707, 1023, 0, 5);
  //turn off the LED to indicate the cherge level
  for(i=3; i>=j; i--){
    digitalWrite(LED[i], LOW);
  }
}
```

4. Verifica del funzionamento

Una volta attivato l'interruttore dell'alimentazione il braccio si porta nella posizione iniziale e i LED visualizzano correttamente il livello di carica della batteria. La ventola si accende correttamente ma è molto rumorosa, è tuttavia possibile scollegarla in caso di necessità. Dopo circa venti secondi l'ESP si collega correttamente alla rete ed è possibile accedere alla pagina web per inviare i comandi. La trasmissione video e quella dei comandi sono molto dipendenti dalla qualità della connessione internet e di conseguenza, quando la connessione è scarsa o il braccio è troppo lontano dalla fonte, la trasmissione dei dati video è a scatti o del tutto assente mentre i comandi dei Servo presentano un ritardo ma vengono comunque eseguiti correttamente. I comandi per le ruote risentono maggiormente della mancanza di segnale: dopo aver inviato il comando cominciano ad eseguirlo in ritardo e la funzione di "stop", dettata dal rilascio del pulsante, a volte viene inviata con un ritardo tale che rende quasi imprevedibile sapere dove si fermerà.

I Servo riescono a sorreggere tranquillamente il braccio senza carico e il braccio è riuscito a sollevare un oggetto di 60[g] senza particolari difficoltà. Anche nelle posizioni "estreme" quali il braccio esteso quanto più possibile al di fuori della base, non si sono verificati problemi anche se è udibile che i Servo stanno faticando a mantenere tale posizione. Il ridotto angolo di rotazione del Servo "spalla" impedisce alla pinza di raggiungere il suolo. Il filtro IIR applicato al movimento dei Servo è molto poco percettibile ad occhio una volta assemblato il braccio e ciò e dovuto allo sforzo dei Servo per andare nella posizione assegnata sollevando tutto ciò che si trova verso la pinza, che non presenta invece questo problema e permette di apprezzare ad occhio nudo il movimento più controllato. L'azione del filtro è tuttavia presente, senza i movimenti sarebbero molto più rapidi e darebbero vita ad oscillazioni nella struttura molto più accentuate rispetto ad adesso.

I motori DC non hanno difficoltà a far muovere la struttura ma per via delle due ruote multidirezionali, posizionate per accompagnare il movimento e sorreggere parte del peso per non farlo gravare tutto sugli assi delle ruote, dopo una rotazione a destra o sinistra, il movimento avanti o indietro subirà una piccola deviazione di traiettoria dovuta al riposizionamento delle ruote multidirezionali.

La corrente massima assorbita dai Servo MG996R in posizione di stallo è di 2,5[A], quella dei Servo Sg90 è di 500[mA] e la corrente massima passante sui motori DC è di 1[A] l'uno. Considerando queste come le casistiche peggiori ed escludendo le correnti assorbite da Arduino, ESP32, LED e L298N (tutte nell'ordine di decine di mA) si ottiene un consumo di potenza di circa 55[W] e, avendo una batteria con una energia nominale di 33,6[Wh] si può stimare una durata della batteria superiore ai 36 minuti che la fa rientrare a pieno nelle specifiche desiderate.

5. Conclusioni

Il progetto è funzionante, anche se presenta alcuni problemi come un leggero ritardo nella ricezione dei comandi e una trasmissione delle immagini non molto stabile e reattiva ma ciò è dovuto alla connessione ad internet ad alla capacità dell'ESP che resta comunque un ottimo microcontrollore capace di gestire i dati nonostante il basso costo.
Per avere un braccio più grande sarebbero necessari motori più potenti che porterebbero il costo ad aumentare notevolmente e richiederebbero anche una batteria più grande per garantire un tempo di funzionamento non risicato.
Si può concludere che il braccio svolge un ottimo lavoro e, a livello didattico, comprende numerosi modi di utilizzare un microcontrollore quale Arduino e ESP32 integrandoli anche con una rete internet dimostrando le infinite possibilità che questi microcontrollori possono offrire.

6. Codice

ESP32 Wrover Module

```
/*********
  Rui Santos
  Complete instructions at https://RandomNerdTutorials.com/esp32-cam-projects-ebook/

  Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files.
  The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
*********/
/*
Modified and upgraded by Damiano De Boni
*/
#include "esp_camera.h"
```

```
#include <WiFi.h>
#include "esp_timer.h"
#include "img_converters.h"
#include "Arduino.h"
#include "fb_gfx.h"
#include "soc/soc.h"            // disable brownout problems
#include "soc/rtc_cntl_reg.h"   // disable brownout problems
#include "esp_http_server.h"

// Replace with your network credentials
const char* ssid = "Microonde telefonico ";
const char* password = "61scroccone";

#define PART_BOUNDARY "123456789000000000000987654321"

//#define CAMERA_MODEL_AI_THINKER
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WITHOUT_PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM_B
#define CAMERA_MODEL_WROVER_KIT

#if defined(CAMERA_MODEL_WROVER_KIT)
  #define PWDN_GPIO_NUM    -1
  #define RESET_GPIO_NUM   -1
  #define XCLK_GPIO_NUM    21
  #define SIOD_GPIO_NUM    26
  #define SIOC_GPIO_NUM    27

  #define Y9_GPIO_NUM      35
  #define Y8_GPIO_NUM      34
  #define Y7_GPIO_NUM      39
  #define Y6_GPIO_NUM      36
  #define Y5_GPIO_NUM      19
  #define Y4_GPIO_NUM      18
```

```
  #define Y3_GPIO_NUM       5
  #define Y2_GPIO_NUM       4
  #define VSYNC_GPIO_NUM   25
  #define HREF_GPIO_NUM    23
  #define PCLK_GPIO_NUM    22

#elif defined(CAMERA_MODEL_M5STACK_PSRAM)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     25
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       32
  #define VSYNC_GPIO_NUM    22
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_M5STACK_WITHOUT_PSRAM)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     25
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
```

```
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       17
  #define VSYNC_GPIO_NUM    22
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21

#elif defined(CAMERA_MODEL_AI_THINKER)
  #define PWDN_GPIO_NUM     32
  #define RESET_GPIO_NUM    -1
  #define XCLK_GPIO_NUM      0
  #define SIOD_GPIO_NUM     26
  #define SIOC_GPIO_NUM     27

  #define Y9_GPIO_NUM       35
  #define Y8_GPIO_NUM       34
  #define Y7_GPIO_NUM       39
  #define Y6_GPIO_NUM       36
  #define Y5_GPIO_NUM       21
  #define Y4_GPIO_NUM       19
  #define Y3_GPIO_NUM       18
  #define Y2_GPIO_NUM        5
  #define VSYNC_GPIO_NUM    25
  #define HREF_GPIO_NUM     23
  #define PCLK_GPIO_NUM     22

#elif defined(CAMERA_MODEL_M5STACK_PSRAM_B)
  #define PWDN_GPIO_NUM     -1
  #define RESET_GPIO_NUM    15
  #define XCLK_GPIO_NUM     27
  #define SIOD_GPIO_NUM     22
```

```
  #define SIOC_GPIO_NUM     23

  #define Y9_GPIO_NUM       19
  #define Y8_GPIO_NUM       36
  #define Y7_GPIO_NUM       18
  #define Y6_GPIO_NUM       39
  #define Y5_GPIO_NUM        5
  #define Y4_GPIO_NUM       34
  #define Y3_GPIO_NUM       35
  #define Y2_GPIO_NUM       32
  #define VSYNC_GPIO_NUM    25
  #define HREF_GPIO_NUM     26
  #define PCLK_GPIO_NUM     21

#else
  #error "Camera model not selected"
#endif

static const char* _STREAM_CONTENT_TYPE = "multipart/x-mixed-replace;boundary="
PART_BOUNDARY;
static const char* _STREAM_BOUNDARY = "\r\n--" PART_BOUNDARY "\r\n";
static const char* _STREAM_PART = "Content-Type: image/jpeg\r\nContent-Length:
%u\r\n\r\n";

httpd_handle_t camera_httpd = NULL;
httpd_handle_t stream_httpd = NULL;

String servoVal= "0";
const char* PARAM_INPUT = "value";

static const char PROGMEM INDEX_HTML[] = R"rawliteral(
<html>
  <head>
    <title>ESP32-CAM Robot</title>
```

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
  body { font-family: Arial; text-align: center; margin:0px auto; padding-top: 30px;}
  table { margin-left: auto; margin-right: auto; }
  td { padding: 8 px; }
  p {font-size: 1.9rem;}
  .button {
    background-color: #9b0014;
    border: none;
    color: white;
    padding: 10px 20px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 18px;
    margin: 6px 3px;
    cursor: pointer;
    -webkit-touch-callout: none;
    -webkit-user-select: none;
    -khtml-user-select: none;
    -moz-user-select: none;
    -ms-user-select: none;
    user-select: none;
    -webkit-tap-highlight-color: rgba(0,0,0,0);
  }

  .slider {
    -webkit-appearance: none;
    margin: 14px;
    width: 250px;
    height: 15px;
    background: #9b0014;
    outline: none;
    -webkit-transition: .2s;
```

```
      transition: opacity .2s;
    }
    .slider::-webkit-slider-thumb {
      -webkit-appearance: none;
      appearance: none;
      width: 20px; height: 20px;
      background: #003249;
      cursor: pointer;
    }
    .slider::-moz-range-thumb {
      width: 20px; height: 20px;
      background: #003249;
      cursor: pointer;
    }

    img {  width: auto ;
      max-width: 75% ;
      height: auto ;
    }
  </style>
 </head>
 <body>
  <h1>ESP32-CAM Robot</h1>
  <img src="" id="photo" >
  <p>
    <input type="range" onchange="updateServo0(this)" id="servo0" min="1" max="99"
value="servoVal" step="1" class="slider">
  </p>
  <p>
    <input type="range" onchange="updateServo1(this)" id="servo1" min="100" max="199"
value="servoVal" step="1" class="slider">
  </p>
  <p>
    <input type="range" onchange="updateServo2(this)" id="servo2" min="200" max="299"
```

```
value="servoVal" step="1" class="slider">
  </p>
  <p>
    <input type="range" onchange="updateServo3(this)" id="servo3" min="300" max="399"
value="servoVal" step="1" class="slider">
  </p>
  <p>
    <input type="range" onchange="updateServo4(this)" id="servo4" min="400" max="499"
value="servoVal" step="1" class="slider">
  </p>
  <p>
    <input type="range" onchange="updateServo5(this)" id="servo5" min="500" max="699"
value="servoVal" step="1" class="slider">
  </p>
  <table>
    <tr><td colspan="3" align="center">
    <button class="button" onmousedown="updateMovement('701');"
ontouchstart="updateMovement('701');" onmouseup="updateMovement('700');"
ontouchend="updateMovement('700');">Forward</button></td></tr>
    <tr><td align="center">
    <button class="button" onmousedown="updateMovement('704');"
ontouchstart="updateMovement('704');" onmouseup="updateMovement('700');"
ontouchend="updateMovement('700');">Left</button></td>
    <td align="center">
    <button class="button" onmousedown="updateMovement('700');"
ontouchstart="updateMovement('700');">Stop</button></td>
    <td align="center">
    <button class="button" onmousedown="updateMovement('703');"
ontouchstart="updateMovement('703');" onmouseup="updateMovement('700');"
ontouchend="updateMovement('700');">Right</button></td></tr>
    <tr><td colspan="3" align="center">
    <button class="button" onmousedown="updateMovement('702');"
ontouchstart="updateMovement('702');" onmouseup="updateMovement('700');"
ontouchend="updateMovement('700');">Backward</button></td></tr>
```

```
</table>
<script>
function updateMovement(x) {
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/action?var=mov&val=" + x, true);
  xhr.send();
}
function updateServo0(servoVal) {
  var servoVal= document.getElementById("servo0").value;
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/action?var=servo&val=" + servoVal, true);
  xhr.send();
}
function updateServo1(servoVal) {
  var servoVal= document.getElementById("servo1").value;
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/action?var=servo&val=" + servoVal, true);
  xhr.send();
}
function updateServo2(servoVal) {
  var servoVal= document.getElementById("servo2").value;
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/action?var=servo&val=" + servoVal, true);
  xhr.send();
}
function updateServo3(servoVal) {
  var servoVal= document.getElementById("servo3").value;
  var xhr = new XMLHttpRequest();
  xhr.open("GET", "/action?var=servo&val=" + servoVal, true);
  xhr.send();
}
function updateServo4(servoVal) {
  var servoVal= document.getElementById("servo4").value;
  var xhr = new XMLHttpRequest();
```

```
    xhr.open("GET", "/action?var=servo&val=" + servoVal, true);
    xhr.send();
   }
   function updateServo5(servoVal) {
    var servoVal= document.getElementById("servo5").value;
    var xhr = new XMLHttpRequest();
    xhr.open("GET", "/action?var=servo&val=" + servoVal, true);
    xhr.send();
   }
  window.onload = document.getElementById("photo").src = window.location.href.slice(0, -1) +
":81/stream";
 </script>
 </body>
</html>
)rawliteral";

static esp_err_t index_handler(httpd_req_t *req){
 httpd_resp_set_type(req, "text/html");
 return httpd_resp_send(req, (const char *)INDEX_HTML, strlen(INDEX_HTML));
}

static esp_err_t stream_handler(httpd_req_t *req){
 camera_fb_t * fb = NULL;
 esp_err_t res = ESP_OK;
 size_t _jpg_buf_len = 0;
 uint8_t * _jpg_buf = NULL;
 char * part_buf[64];

 res = httpd_resp_set_type(req, _STREAM_CONTENT_TYPE);
 if(res != ESP_OK){
  return res;
 }

 while(true){
```

```
    fb = esp_camera_fb_get();
    if (!fb) {
      //Serial.println("Camera capture failed");
      res = ESP_FAIL;
    } else {
      if(fb->width > 400){
        if(fb->format != PIXFORMAT_JPEG){
          bool jpeg_converted = frame2jpg(fb, 80, &_jpg_buf, &_jpg_buf_len);
          esp_camera_fb_return(fb);
          fb = NULL;
          if(!jpeg_converted){
            //Serial.println("JPEG compression failed");
            res = ESP_FAIL;
          }
        } else {
          _jpg_buf_len = fb->len;
          _jpg_buf = fb->buf;
        }
      }
    }
    if(res == ESP_OK){
      size_t hlen = snprintf((char *)part_buf, 64, _STREAM_PART, _jpg_buf_len);
      res = httpd_resp_send_chunk(req, (const char *)part_buf, hlen);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len);
    }
    if(res == ESP_OK){
      res = httpd_resp_send_chunk(req, _STREAM_BOUNDARY,
strlen(_STREAM_BOUNDARY));
    }
    if(fb){
      esp_camera_fb_return(fb);
      fb = NULL;
```

```
      _jpg_buf = NULL;
    } else if(_jpg_buf){
      free(_jpg_buf);
      _jpg_buf = NULL;
    }
    if(res != ESP_OK){
      break;
    }
    //Serial.printf("MJPG: %uB\n",(uint32_t)(_jpg_buf_len));
  }
  return res;
}

static esp_err_t cmd_handler(httpd_req_t *req){
  char*  buf;
  size_t buf_len;
  char variable[32] = {0,};
  char value[32] = {0,};

  buf_len = httpd_req_get_url_query_len(req) + 1;
  if (buf_len > 1) {
    buf = (char*)malloc(buf_len);
    if(!buf){
      httpd_resp_send_500(req);
      return ESP_FAIL;
    }
    if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK) {
      if (httpd_query_key_value(buf, "var", variable, sizeof(variable)) == ESP_OK &&
          httpd_query_key_value(buf, "val", value, sizeof(value)) == ESP_OK) {
      } else {
        free(buf);
        httpd_resp_send_404(req);
        return ESP_FAIL;
      }
```

```
    } else {
      free(buf);
      httpd_resp_send_404(req);
      return ESP_FAIL;
    }
    free(buf);
  } else {
    httpd_resp_send_404(req);
    return ESP_FAIL;
  }

  int val= atoi(value);
  sensor_t * s = esp_camera_sensor_get();
  int res = 0;

  if(!strcmp(variable, "mov")) {
    Serial.println(val);
  }
  else if(!strcmp(variable, "servo")) {
    Serial.println(val);
  }
  else {
    res = -1;
  }

  if(res){
    return httpd_resp_send_500(req);
  }

  httpd_resp_set_hdr(req, "Access-Control-Allow-Origin", "*");
  return httpd_resp_send(req, NULL, 0);
}

void startCameraServer(){
```

```
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  config.server_port = 80;
  httpd_uri_t index_uri = {
    .uri       = "/",
    .method    = HTTP_GET,
    .handler   = index_handler,
    .user_ctx  = NULL
  };

  httpd_uri_t cmd_uri = {
    .uri       = "/action",
    .method    = HTTP_GET,
    .handler   = cmd_handler,
    .user_ctx  = NULL
  };
  httpd_uri_t stream_uri = {
    .uri       = "/stream",
    .method    = HTTP_GET,
    .handler   = stream_handler,
    .user_ctx  = NULL
  };
  if (httpd_start(&camera_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(camera_httpd, &index_uri);
    httpd_register_uri_handler(camera_httpd, &cmd_uri);
  }
  config.server_port += 1;
  config.ctrl_port += 1;
  if (httpd_start(&stream_httpd, &config) == ESP_OK) {
    httpd_register_uri_handler(stream_httpd, &stream_uri);
  }
}

void setup() {
  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0); //disable brownout detector
```

```
Serial.begin(115200);
Serial.setDebugOutput(false);

camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;

if(psramFound()){
  config.frame_size = FRAMESIZE_VGA;
  config.jpeg_quality = 10;
  config.fb_count = 2;
} else {
  config.frame_size = FRAMESIZE_SVGA;
  config.jpeg_quality = 12;
  config.fb_count = 1;
```

```
  }

  // Camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    //Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }
  // Wi-Fi connection
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    //Serial.print(".");
  }
  //Serial.println("");
  //Serial.println("WiFi connected");

  //Serial.print("Camera Stream Ready! Go to: http:");
  //Serial.println(WiFi.localIP());

  // Start streaming web server
  startCameraServer();
}

void loop() {

}
```

Arduino Nano

```
/*
Code made by Damiano De Boni
*/
#include <Servo.h>
```

```
int val=0;
byte sel=0;
long preMillis=0;

//servo
  Servo servo0; //create 6 servo object
  Servo servo1;
  Servo servo2;
  Servo servo3;
  Servo servo4;
  Servo servo5;

  //variable for the IIR filter
  float finalVal;
  float preVal0=700, preVal1=700, preVal2=700, preVal3=700, preVal4=700, preVal5=700;
  float smoothVal0, smoothVal1, smoothVal2, smoothVal3, smoothVal4, smoothVal5;

//DC motor
  //motor A
  #define IN1 12
  #define IN2 13
  //motor B
  #define IN3 A0
  #define IN4 A1

//battery
  #define bat A6 //ADC
  //led pin indicating the charge level
  #define LED75 A2
  #define LED50 A3
  #define LED25 A4
  #define LED00 A5
  byte LED[4]={LED00, LED25, LED50, LED75};
```

```
  byte i, j;

void setup() {
  //initialize the serial comunication
  Serial.begin(115200);
//servo
  //connect the servo to pin x (PWM)
  //mg996R
  servo5.attach(3);
  servo4.attach(5);
  servo3.attach(6);
  //sg90
  servo2.attach(9);
  servo1.attach(10);
  servo0.attach(11);

  //set the initial servo position
  servo0.writeMicroseconds(preVal0);
  servo1.writeMicroseconds(preVal1);
  servo2.writeMicroseconds(preVal2);
  servo3.writeMicroseconds(preVal3);
  servo4.writeMicroseconds(preVal4);
  servo5.writeMicroseconds(preVal5);



//DC motor
  //initialize the motor pin
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
```

```
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);

//battery
  //turn on the LED
  for(i=0; i<4; i++){
    pinMode(LED[i], OUTPUT);
    digitalWrite(LED[i], HIGH);
  }
  //turn off the LED to undicate the charge level
  j= map(analogRead(bat), 707, 1023, 0, 5);
  for(i=3; i>=j; i--){
    digitalWrite(LED[i], LOW);
  }

}

void loop() {
  if(Serial.available()){

    //read the serial data
    val=Serial.parseInt();
   if(val!=0){
    //Serial.print("I recived: ");
    //Serial.println(val);

    sel=byte(val/100);
    //Serial.print("Selection: ");
    //Serial.println(sel);

    //servo selection
    if(sel <= 4){

      val=val%100;
```

```
//convert the value to microseconds for the servo
finalVal= map(val, 0, 99, 700, 2000);
//Serial.print("final value: ");
//Serial.println(finalVal);

switch(sel){
  case 0:
    //create a smooth movment for the servo using a IIR filter
    while((smoothVal0 < (finalVal-25)) || (smoothVal0 > (finalVal+25))){

      //IIR filter y(k)= (x(k) * b) + (y(k-1) * (1-b))
      smoothVal0 = (finalVal * 0.05) + (preVal0 * 0.95);
      preVal0 = smoothVal0; //y(k-1)

      servo0.writeMicroseconds(smoothVal0);
      delay(5);
    }
    //Serial.print("servo 0 smoothed value: ");
    //Serial.println(smoothVal0);
    //Serial.println(" ");
  break;

  case 1:
    //create a smooth movment for the servo using a IIR filter
    while((smoothVal1 < (finalVal-25)) || (smoothVal1 > (finalVal+25))){

      //IIR filter y(k)= (x(k) * b) + (y(k-1) * (1-b))
      smoothVal1 = (finalVal * 0.05) + (preVal1 * 0.95);
      preVal1 = smoothVal1; //y(k-1)

      servo1.writeMicroseconds(smoothVal1);
      delay(5);
    }
```

```
  //Serial.print("servo 1 smoothed value: ");
  //Serial.println(smoothVal1);
  //Serial.println(" ");
break;

case 2:
  //create a smooth movment for the servo using a IIR filter
  while((smoothVal2 < (finalVal-25)) || (smoothVal2 > (finalVal+25))){

    //IIR filter y(k)= (x(k) * b) + (y(k-1) * (1-b))
    smoothVal2 = (finalVal * 0.05) + (preVal2 * 0.95);
    preVal2 = smoothVal2; //y(k-1)

    servo2.writeMicroseconds(smoothVal2);
    delay(5);
  }
  //Serial.print("servo 2 smoothed value: ");
  //Serial.println(smoothVal2);
  //Serial.println(" ");
break;

case 3:
  //create a smooth movment for the servo using a IIR filter
  while((smoothVal3 < (finalVal-25)) || (smoothVal3 > (finalVal+25))){

    //IIR filter y(k)= (x(k) * b) + (y(k-1) * (1-b))
    smoothVal3 = (finalVal * 0.05) + (preVal3 * 0.95);
    preVal3 = smoothVal3; //y(k-1)

    servo3.writeMicroseconds(smoothVal3);
    delay(5);
  }
  //Serial.print("servo 3 smoothed value: ");
  //Serial.println(smoothVal3);
```

```
      //Serial.println(" ");
    break;

    case 4:
     //create a smooth movment for the servo using a IIR filter
     while((smoothVal4 < (finalVal-25)) || (smoothVal4 > (finalVal+25))){

      //IIR filter y(k)= (x(k) * b) + (y(k-1) * (1-b))
      smoothVal4 = (finalVal * 0.05) + (preVal4 * 0.95);
      preVal4 = smoothVal4; //y(k-1)

      servo4.writeMicroseconds(smoothVal4);
      delay(5);
     }
     //Serial.print("servo 4 smoothed value: ");
     //Serial.println(smoothVal4);
     //Serial.println(" ");
    break;
  }
}

//servo5 selection
else if(sel==5 || sel==6){
  //convert the value
  finalVal=map(val, 500, 699, 700, 2000);

  //create a smooth movment for the servo using a IIR filter
  while((smoothVal5 < (finalVal-25)) || (smoothVal5 > (finalVal+25))){

    //IIR filter y(k)= (x(k) * b) + (y(k-1) * (1-b))
    smoothVal5 = (finalVal * 0.05) + (preVal5 * 0.95);
    preVal5 = smoothVal5; //y(k-1)

    servo5.writeMicroseconds(smoothVal5);
```

```
    delay(5);
  }
  //Serial.print("servo 5 smoothed value: ");
  //Serial.println(smoothVal5);
  //Serial.println(" ");

}

//DC motor selection
else if(sel==7){
  val=val%100;
  //stop

  if(val==0){
    //motor A
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
    //motor B
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, LOW);
    //Serial.println("Stop");
  }

  //avanti
  else if(val==1){
    //motor A
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    //motor B
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    //Serial.println("Avanti");
  }
```

```
    //indietro
    else if(val==2){
      //motor A
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      //motor B
      digitalWrite(IN3, HIGH);
      digitalWrite(IN4, LOW);
      //Serial.println("Indietro");
    }

    //destra
    else if(val==3){
      //motor A
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, HIGH);
      //motior B
      digitalWrite(IN3, HIGH);
      digitalWrite(IN4, LOW);
      //Serial.println("Destra");
    }

    //sinistra
    else if(val==4){
      //motor A
      digitalWrite(IN1, HIGH);
      digitalWrite(IN2, LOW);
      //motor B
      digitalWrite(IN3, LOW);
      digitalWrite(IN4, HIGH);
      //Serial.println("Sinistra");
    }
  }
}
```

```
  }
  //battery level indicator (check the voltage every minute)
  if(millis()-preMillis>=60000){
    preMillis=millis();
    j= map(analogRead(bat), 707, 1023, 0, 5);
    //turn off the LED to indicate the cherge level
    for(i=3; i>=j; i--){
      digitalWrite(LED[i], LOW);
    }
  }

}
```

7. Bibliografia e sitografia

https://it.wikipedia.org/wiki/Accumulatore_agli_ioni_di_litio

https://win.adrirobot.it/servotester/il_servomotore.htm

https://randomnerdtutorials.com/esp32-cam-car-robot-web-server/

https://it.wikipedia.org/wiki/Motore_passo-passo

https://www.maffucci.it/2020/10/28/controllare-uno-stepper-28byj-48-con-uln2003-mediante-arduino/

https://it.wikipedia.org/wiki/Macchina_in_corrente_continua

https://www.arduino.cc/en/Guide/Introduction#why-arduino

https://store.arduino.cc/products/arduino-nano

https://www.moreware.org/wp/blog/2023/01/15/che-cosa-e-esp32-e-differenze-con-arduino-secondo-chatgpt/

8. Ringraziamenti

Non ho la minima idea di come dovrei scrivere dei ringraziamenti su una tesi; quindi, li farò molto brevi e poi vi ringrazio di persona sperando che nessuno li legga mai da qui. Come prima persona devo ringraziare mia mamma che mi ha sempre supportato e sopportato in tutti questi anni (soprattutto con la mia fantastica ansia durante i periodi degli esami), lo sai che se sono arrivato fin qui è anche grazie a te. Ringrazio poi tutti gli zii, zie, cugini che mi sono sempre stati accanto e

aiutato, in particolare lo zio Giorgio che mi ha aiutato a tagliare e forare angolari in alluminio per questo progetto e mi ha dato ottimi consigli per tutta la parte meccanica di cui non sono esperto. Ringrazio anche i miei colleghi di studio Leonardo, Paolo e Federico che mi hanno aiutato a passare molti esami senza impazzire per farlo. E ultimi per mancanza di fiducia nelle mie creazioni e in me in generale, ringrazio i miei amici, Leonardo, Eleonora, Matteo, Alice, Alberto, Valentina, Alessia e Ilghis che non hanno assolutamente mai dubitato che ciò che costruisco possa prendere fuoco ed esplodere da un momento all'altro, vero Leonardo? Se il prossimo regalo che ti faccio emette uno strano ticchettio non badarci troppo. Ok ora ho davvero finito le idee su cosa scrivere; un grazie a tutti.