**Proiect**

**Braț Robotic**

**Luci-Mihăiță Ștefănoiu**

**Profesor Îndrumător:**

**Dr. Ing. Mihnea Marin**

Cuprins

[1. Arduino. Prezentare generală 3](#_Toc169710569)

[2. Braț Robotic controlat cu ajutorul unui telefon mobil inteligent și a tehnologiei Arduino 5](#_Toc169710570)

[2.1. Prezentare generală 5](#_Toc169710571)

[2.2. Prezentarea Brațului Robotic Controlat de Smartphone 8](#_Toc169710572)

[2.3. Prezentarea codului sursă 14](#_Toc169710573)

[2.4. Concluzie 22](#_Toc169710574)

# Arduino. Prezentare generală

Ce este Arduino?

Arduino este o platformă electronică open-source bazată pe componenre hardware și software ușor de utilizat. Plăcile Arduino pot citi input-uri precum: lumina unui senzor senzor, apăsarea cu degetul pe un buton sau un mesaj pe Twitter - și le pot transforma în output-uri precum: aprinderea unui motor sau a unui LED, publicarea unui mesaj online. Prin programarea unei plăcuțe Arduino îi poți spune acesteia ce să facă prin trimiterea unui set de instrucțiuni către microcontroller-ul de pe placă. Pentru aceasta, se folosește limbajul de programare Arduino (bazat pe Wiring) și Software-ul Arduino (IDE), bazat pe Processing.

De-a lungul anilor, Arduino a fost componenta principală pentru mii de proiecte, de la obiecte de zi cu zi la instrumente științifice complexe. O comunitate globală de creatori - studenți, amatori, artiști, programatori și profesioniști - s-a adunat în jurul acestei platforme open-source, contribuțiile lor adunându-se într-o cantitate incredibilă de cunoștințe accesibile care pot fi de mare ajutor atât pentru începători, cât și pentru experți.

Arduino s-a născut la Ivrea Interaction Design Institute ca un instrument ușor pentru prototipare rapidă, destinat studenților fără experiență în electronică și programare. Imediat ce a ajuns la o comunitate mai largă, placa Arduino a început să se schimbe pentru a se putea adapta la noi nevoi și provocări, diferențiindu-și oferta de la plăci simple de 8 biți la produse pentru aplicații IoT (Internet of Things), wearable (smartwatch-uri, brățări fitness), imprimante 3D etc.

De ce Arduino?

Datorită experienței sale simple și accesibile de utilizare, Arduino a fost folosit în mii de proiecte și aplicații diferite. Software-ul Arduino este ușor de utilizat pentru începători, dar suficient de flexibil pentru utilizatorii avansați. Rulează pe Mac, Windows și Linux. Profesorii și studenții îl folosesc pentru a construi instrumente științifice cu costuri reduse, pentru a demonstra principii de chimie și fizică sau pentru a începe cu programarea și robotica. Designerii și arhitecții construiesc prototipuri interactive, muzicienii și artiștii îl folosesc pentru instalații și pentru a experimenta noi instrumente muzicale. Creatorii, desigur, îl folosesc pentru a construi multe dintre proiectele expuse. Arduino este un instrument cheie pentru a învăța lucruri noi. Oricine - copii, amatori, artiști, programatori - poate începe să experimenteze urmând instrucțiunile pas cu pas ale unui kit sau împărtășind idei online cu alți membri ai comunității Arduino.

Există mulți alte microcontrollere și platforme de microcontrollere disponibile pentru calcul fizic. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard și multe altele oferă funcționalități similare. Toate aceste instrumente iau detaliile complicate ale programării microcontrollerelor și le adaugă într-un pachet ușor de utilizat. Arduino simplifică, de asemenea, procesul de lucru cu microcontrollere, dar oferă câteva avantaje pentru profesori, studenți și amatori interesați față de alte sisteme:

* Cost redus - Plăcile Arduino sunt relativ ieftine în comparație cu alte platforme de microcontrollere. Cea mai ieftină versiune a modulului Arduino poate fi asamblată manual, iar chiar și modulele Arduino pre-asamblate costă mai puțin de 250 de lei.
* Compatibilitate între platforme - Software-ul Arduino (IDE) rulează pe sistemele de operare Windows, Macintosh OSX și Linux. Majoritatea sistemelor de microcontrollere sunt limitate la Windows.
* Mediu de programare simplu și clar - Software-ul Arduino (IDE) este ușor de utilizat pentru începători, dar suficient de flexibil pentru utilizatorii avansați. Pentru profesori, este convenabil bazat pe mediul de programare Processing, astfel încât studenții care învață să programeze în acel mediu sunt familiarizați cu modul în care funcționează Arduino IDE.
* Software open source și extensibil - Software-ul Arduino este publicat ca instrumente open source, disponibile pentru extindere de către programatori experimentați. Limbajul poate fi extins prin biblioteci C++, iar cei care doresc să înțeleagă detaliile tehnice pot face tranziția de la Arduino la limbajul de programare AVR C pe care se bazează. De asemenea, poți adăuga cod AVR-C direct în programele tale Arduino dacă dorești.
* Hardware open source și extensibil - Planurile plăcilor Arduino sunt publicate sub o licență Creative Commons, astfel încât designerii de circuite experimentați își pot crea propria versiune a modulului, extinzându-l și îmbunătățindu-l. Chiar și utilizatorii relativ neexperimentați pot construi versiunea pe breadboard a modulului pentru a înțelege cum funcționează și pentru a economisi bani.

# Braț Robotic controlat cu ajutorul unui telefon mobil inteligent și a tehnologiei Arduino

## Prezentare generală

**Introducere:**

Un braț robotic este un dispozitiv mecanic controlat de un sistem de comandă, adesea programabil, care poate efectua diverse sarcini automatizate. Aceste dispozitive sunt utilizate în diverse domenii, inclusiv în producție, asamblare, medicină și cercetare. Brațele robotice sunt proiectate să imite mișcările și funcționalitatea unui braț uman, având mai multe articulații și segmente care le permit să se miște în moduri complexe și precise.

#### Componente Principale:

1. **Baza**: Este punctul de fixare al brațului robotic și conține motorul principal sau alte mecanisme de rotație.
2. **Articulații (Joints)**: Permit mișcarea între segmentele brațului. Fiecare articulație oferă un grad de libertate (DOF), iar un braț robotic poate avea multiple DOF.

DOF înseamnă "Degrees of Freedom" (Grade de Libertate). Este un termen utilizat în robotică și alte domenii pentru a descrie numărul de moduri independente prin care un sistem poate să se miște sau să varieze.

**Detalii despre Gradele de Libertate (DOF):**

1. **Grade de Libertate Liniare**:
   * **Translație**: Mișcarea pe axele X, Y și Z. Fiecare direcție de mișcare reprezintă un grad de libertate.
     + **X**: Mișcare înainte și înapoi.
     + **Y**: Mișcare stânga și dreapta.
     + **Z**: Mișcare sus și jos.
2. **Grade de Libertate Rotative**:
   * **Rotire**: Mișcarea de rotație în jurul axelor X, Y și Z.
     + **Rotire în jurul axei X (Pitch)**: Înclinarea în față și în spate.
     + **Rotire în jurul axei Y (Yaw)**: Rotirea stânga și dreapta.
     + **Rotire în jurul axei Z (Roll)**: Rotirea în jurul propriei axe.

Un braț robotic cu 6 grade de libertate poate avea următoarele mișcări:

1. **Rotire la bază**: Permite brațului să se rotească pe axa verticală (Yaw).
2. **Mișcare în sus și în jos**: Articulația la bază permite brațului să se ridice și să coboare (Pitch).
3. **Îndoire la cot**: Permite segmentului mijlociu al brațului să se îndoaie și să se întindă (Pitch).
4. **Mișcarea sus-jos a încheieturii mâinii**: Permite segmentului final să ridice și să coboare.
5. **Rotire la încheietura mâinii**: Permite segmentului final să se rotească (Roll).
6. **Mișcare la prindere**: End-effectorul (mâna sau dispozitivul de prindere) poate să se deschidă și să se închidă.

Gradele de libertate sunt esențiale pentru a înțelege complexitatea și capacitățile unui sistem robotic. Cu cât un braț robotic are mai multe grade de libertate, cu atât poate efectua mișcări mai complexe și poate manipula obiecte într-un mod mai variat și precis.

1. **Segmentele (Links)**: Acestea sunt secțiunile rigide ale brațului care conectează articulațiile.
2. **Motoare și Actuatori**: Aceste componente care generează mișcarea articulațiilor. Pot fi motoare electrice, pneumatice sau hidraulice.
3. **Controller**: Este unitatea centrală care coordonează toate mișcările brațului robotic. Acesta interpretează comenzi și trimite semnale motoarelor și actuatoarelor.
4. **Senzori**: Oferă feedback controller-ului despre poziția, viteza și alte condiții ale brațului robotic, asigurând precizie și corectarea erorilor.
5. **End-Effector**: Este componenta de la capătul brațului robotic care interacționează cu mediul. Poate fi o gheară, o pensetă, un instrument de sudură, un dispozitiv de prindere etc.

#### Funcționalitate și Caracteristici:

* **Grade de Libertate (DOF)**: Numărul de direcții în care se poate mișca brațul. Cu cât mai multe DOF, cu atât mai complex și versatil este brațul.
* **Control**: Poate fi realizat prin diverse metode, inclusiv manual, prin telecomandă, sau automat, folosind programe predefinite.
* **Precizie și Repetabilitate**: Brațele robotice sunt proiectate pentru a efectua sarcini cu un nivel ridicat de precizie și pentru a putea repeta aceleași mișcări cu consistență.
* **Capacitatea de Încărcare**: Greutatea maximă pe care o poate manipula brațul fără a compromite performanța.

#### Aplicații:

* **Industrie**: Folosite în linii de asamblare pentru sudură, montaj, vopsire și manipulare de materiale.
* **Medicină**: Utilizate în chirurgia robotică pentru proceduri minim invazive, oferind o precizie crescută.
* **Cercetare**: Brațele robotice sunt esențiale în laboratoare pentru manipularea probelor și în experimentele automate.
* **Servicii**: În logistică pentru sortarea și manipularea coletelor, în restaurante pentru prepararea mâncării etc.

#### Avantaje și Dezavantaje:

**Avantaje**:

* **Precizie și Repetabilitate**: Capabile să efectueze sarcini complexe cu un grad înalt de precizie și repetabilitate.
* **Eficiență**: Pot funcționa neîntrerupt, crescând productivitatea și eficiența operațională.
* **Versatilitate**: Pot fi programate și reconfigurate pentru a efectua o gamă largă de sarcini.

**Dezavantaje**:

* **Cost**: Investiția inițială pentru achiziționarea și implementarea unui braț robotic poate fi ridicată.
* **Complexitate**: Necesită expertiză pentru programare, operare și întreținere.
* **Limitări Fizice**: Pot avea restricții în ceea ce privește raza de acțiune și capacitatea de încărcare.

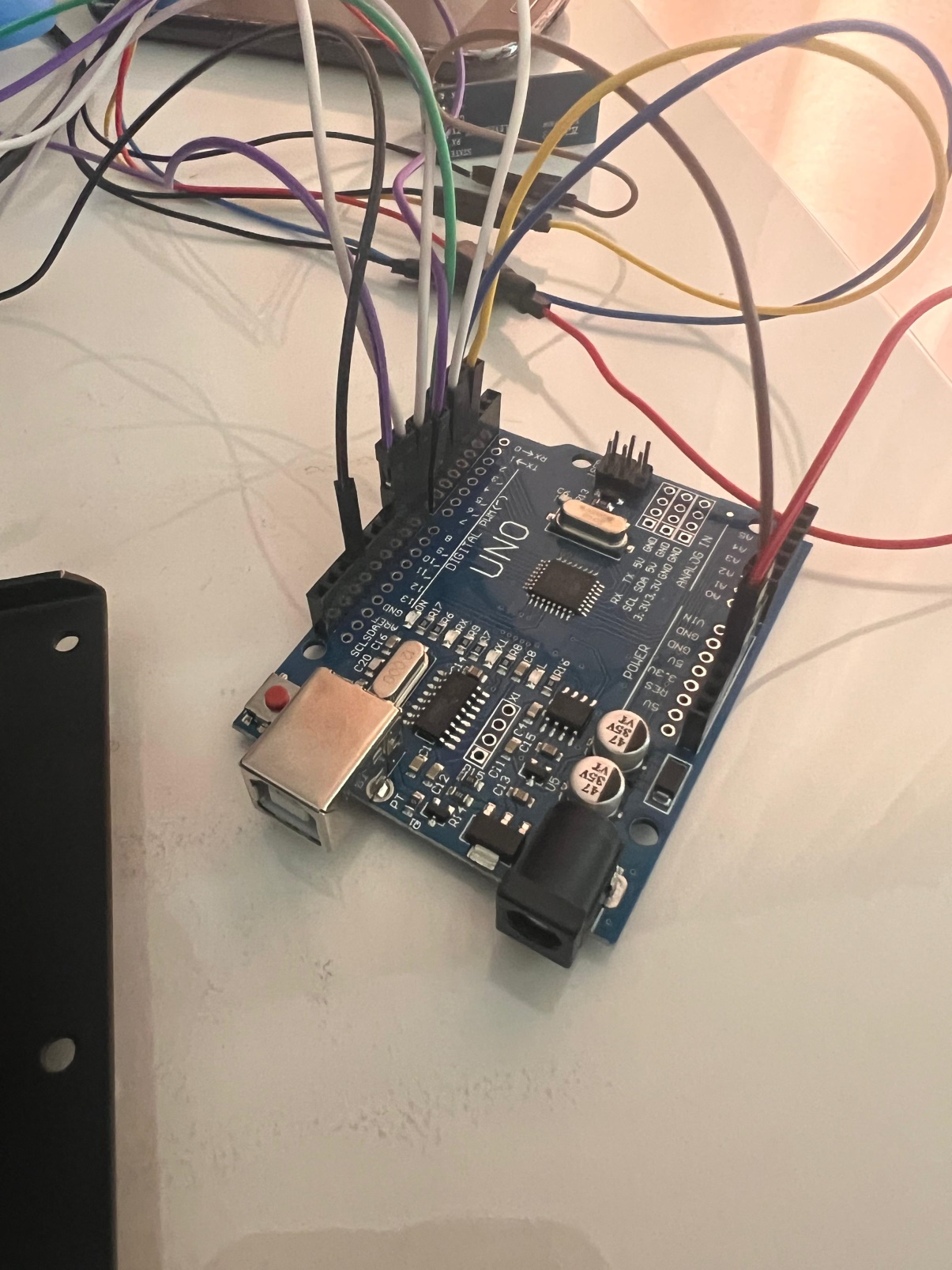
**Concluzie:**

Brațele robotice reprezintă un avans tehnologic semnificativ, aducând beneficii majore în diverse domenii prin automatizarea sarcinilor repetitive și precise. Cu o planificare și utilizare adecvată, acestea pot îmbunătăți considerabil eficiența și productivitatea operațiunilor.

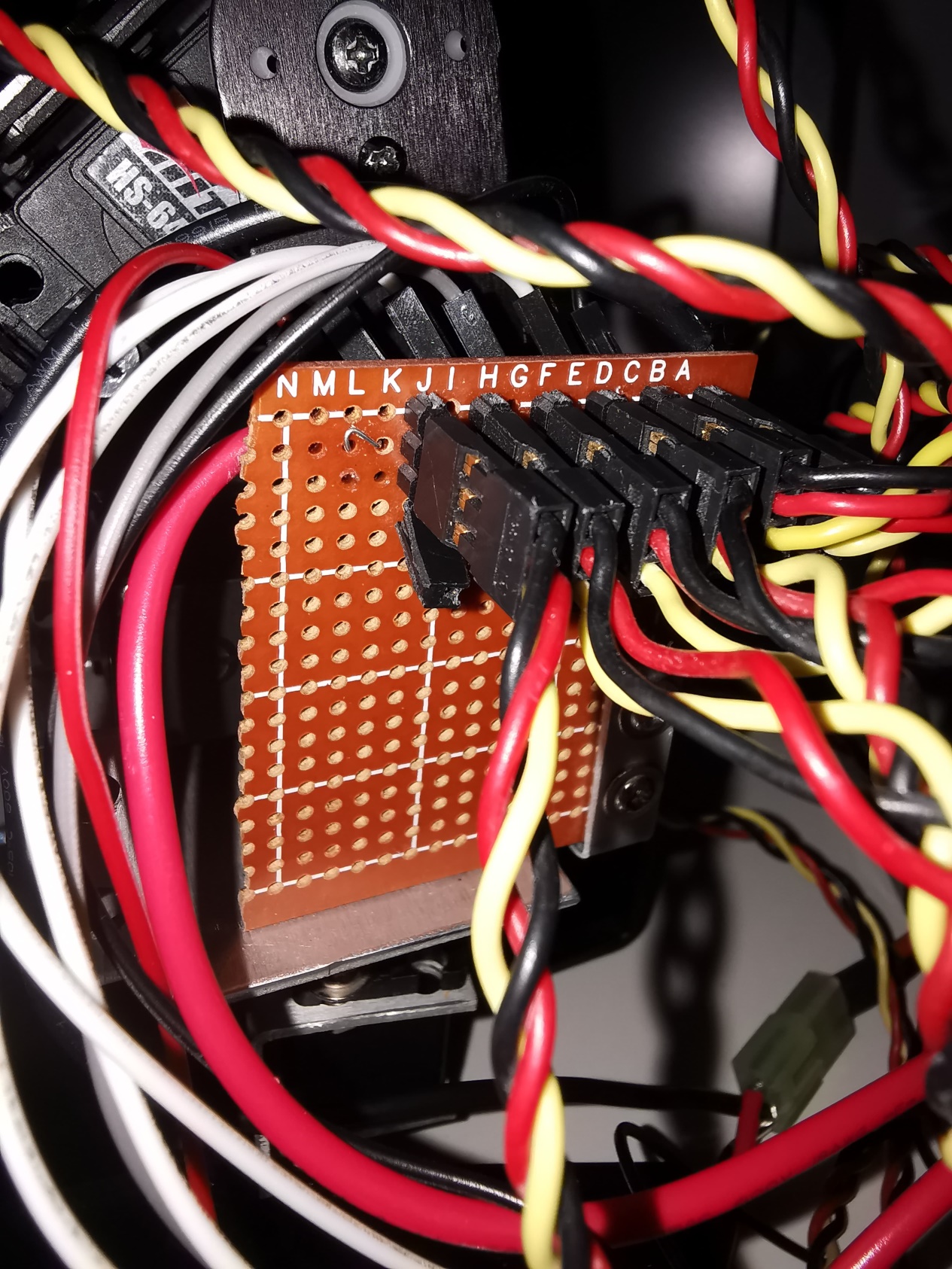
## Prezentarea Brațului Robotic Controlat de Smartphone

#### Componente Principale:

1. **Placa Arduino Uno**: Este centrul de control al brațului robotic, care primește comenzi de la aplicația de pe smartphone și controlează motoarele în funcție de aceste comenzi.



1. **Breadboard**: Utilizat pentru a conecta motoarele și alte componente la Arduino Uno într-un mod temporar și flexibil.



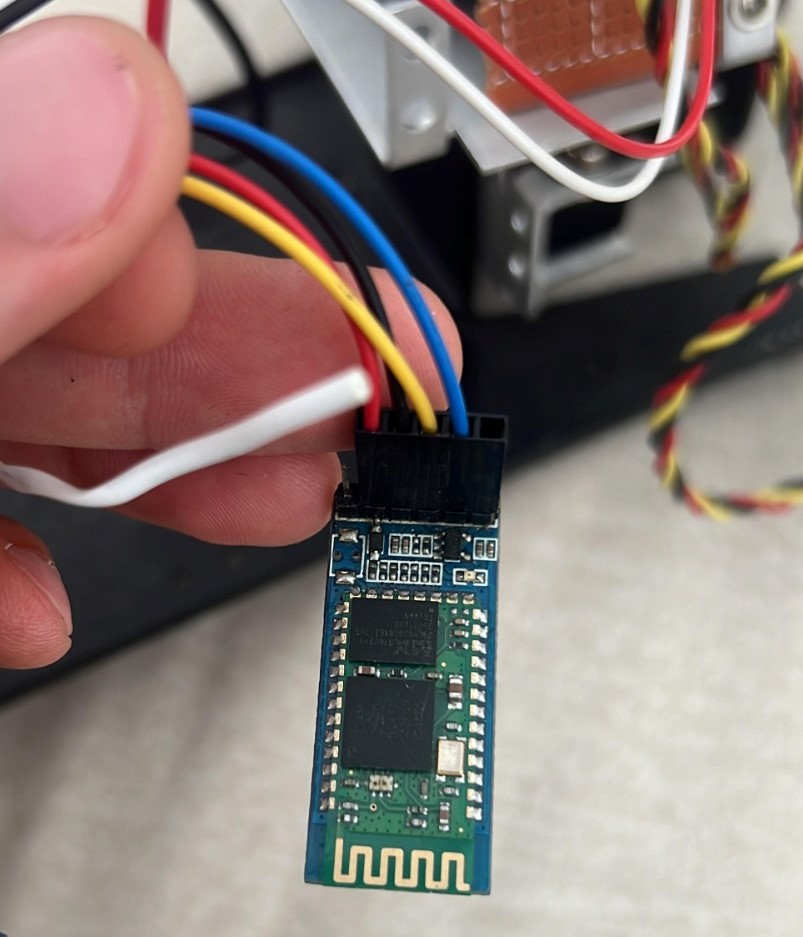
1. **Motoare**:
   * **6 servo-motoare HS-645MG**: Folosite pentru mișcările de bază ale brațului (talie, umăr, cot) și pentru mișcările încheieturii (rotire și înclinare), precum și pentru controlul ghearei.



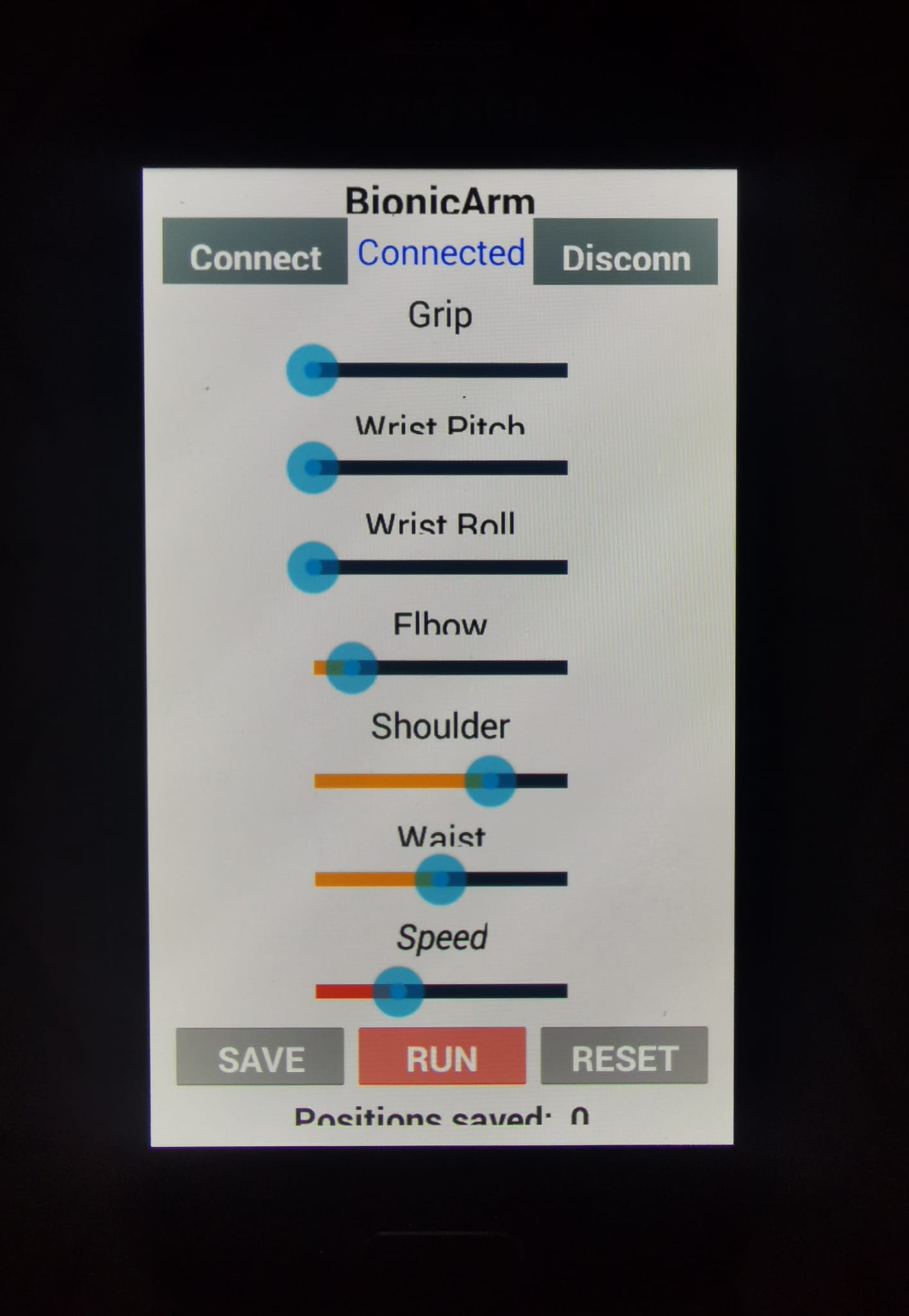
1. **Baterie**: Sursa de alimentare pentru brațul robotic și Arduino Uno, pentru a-l face independent de o sursă de curent fixă.



1. **Chip-ul (Modulul)** Bluetooth HC-05: Este utilizat pentru a face posibilă conexiunea wilreless dintre Smartphone și microcontroller-ul Arduino



1. **Smartphone**: Este utilizat pentru a controla brațul robotic prin intermediul unei aplicații speciale Arduino, care permite utilizatorului să trimită comenzi pentru mișcare și acțiuni.
2. **Aplicația Arduino**: Instalată pe smartphone, această aplicație permite utilizatorului să interacționeze cu brațul robotic prin intermediul Bluetooth, trimițând semnale de control către Arduino.



#### Funcționare:

* **Control prin Aplicație**: Utilizatorul deschide aplicația Arduino pe smartphone și stabilește conexiunea cu Arduino Uno prin Bluetooth.
* **Interacțiunea cu Brațul Robotic**: Utilizatorul poate utiliza interfața aplicației pentru a controla mișcările brațului robotic. De exemplu, există butoane sau glisoare pentru a controla poziția și mișcarea fiecărui motor în parte.
* **Execuția Comenzilor**: Arduino Uno primește comenzi de la aplicație și convertește aceste comenzi în semnale pentru motoarele respective. De exemplu, pentru a ridica brațul sau pentru a deschide și închide gheara.
* **Alimentare și Portabilitate**: Datorită bateriei, brațul robotic poate fi utilizat în diverse locații fără a depinde de o sursă de alimentare fixă.

#### Aplicații și Utilizări:

* **Educație și Învățare**: Este ideal pentru a învăța despre principiile robotică și programare Arduino.
* **Demonstrații și Proiecte de Hobby**: Poate fi utilizat în expoziții și evenimente pentru a demonstra capacitățile tehnologice.
* **Experimente și Prototipuri**: Permite utilizatorilor să experimenteze cu diferite configurări și să dezvolte prototipuri de aplicații robotice mai avansate.

## Prezentarea codului sursă

* + 1. *Includerea librăriilor SoftwareSerial pentru comunicarea serială a modulului Bluetooth, precum și librăriilor pentru servo-motoare.*

Librăriile sunt incluse în Arduino IDE. Definirea celor șase servo-motoare, a modulului Bluetooth HC-05 și a câtorva variabile pentru stocarea poziției curente și anterioare a servo-motoarelor, precum și a array-urilor destinate stocării pozițiilor sau pașilor pentru modul automat.

#include <SoftwareSerial.h>

#include <Servo.h>

Servo servo01;

Servo servo02;

Servo servo03;

Servo servo04;

Servo servo05;

Servo servo06;

SoftwareSerial Bluetooth(3, 4); // Arduino(RX, TX) - HC-05 Bluetooth (TX, RX)

int servo1Pos, servo2Pos, servo3Pos, servo4Pos, servo5Pos, servo6Pos; // current position

int servo1PPos, servo2PPos, servo3PPos, servo4PPos, servo5PPos, servo6PPos; // previous position

int servo01SP[50], servo02SP[50], servo03SP[50], servo04SP[50], servo05SP[50], servo06SP[50]; // for storing positions/steps

int speedDelay = 20;

int index = 0;

String dataIn = "";

* + 1. *Funcția setup()*

În secțiunea de configurare (setup), are loc inițializarea servo-motoarele și a modulului Bluetooth, cât și setarea brațului robotic în poziția sa inițială. Acest lucru se realizeaz**ă** folosind funcția predefini**tă** write(), care mută servo-motorul într-o poziție între 0 și 180 de grade.

void setup() {

  servo01.attach(5);

  servo02.attach(6);

  servo03.attach(7);

  servo04.attach(8);

  servo05.attach(9);

  servo06.attach(10);

  Bluetooth.begin(9600); // Default baud rate of the Bluetooth module

  Bluetooth.setTimeout(1);

  delay(20);

  // Robot arm initial position

  servo1PPos = 90;

  servo01.write(servo1PPos);

  servo2PPos = 100;

  servo02.write(servo2PPos);

  servo3PPos = 95;

  servo03.write(servo3PPos);

  servo4PPos = 10;

  servo04.write(servo4PPos);

  servo5PPos = 25;

  servo05.write(servo5PPos);

  servo6PPos = 10;

  servo06.write(servo6PPos);

}

* + 1. *Funcția loop()*

În secțiunea loop, folosind funcția Bluetooth.available(), se verifică constant dacă există date de intrare de la smartphone. În caz afirmativ, folosind funcția readString(), se citesc datele ca un șir de caractere și se stochează în variabila dataIn. În funcție de datele primite, brațul robotic va ști ce acțiuni să întreprindă. Fiecare buton de tipul cursor (slider) acționează câte un servo-motoraș al brațului. Astfel funcția loop(), în funcție de input-urile primite de la smartphone (telefon), va evalua ce cursor (slider) a fost modificat și va mișca servo-motorașul brațului robotic specific cursorului (slider-ului) în funcție de acesta.

În mod similar, cu ajutorul acestor declarațiilor de tip if și al buclelor for are loc mutarea servo-motoarele în următoarea lor poziție. În final, dacă apăsăm butonul RESET, vom șterge toate datele din array-uri și le vom seta cu 0 (zero), resetând totodată și indexul acestora la 0 (zero) pentru a putea reprograma brațul robotic cu noi mișcări.

void loop() {

  // Check for incoming data

  if (Bluetooth.available() > 0) {

    dataIn = Bluetooth.readString();  // Read the data as string

    // If "Waist" slider has changed value - Move Servo 1 to position

    if (dataIn.startsWith("s1")) {

      String dataInS = dataIn.substring(2, dataIn.length()); // Extract only the number. E.g. from "s1120" to "120"

      servo1Pos = dataInS.toInt();  // Convert the string into integer

      // We use for loops so we can control the speed of the servo

      // If previous position is bigger then current position

      if (servo1PPos > servo1Pos) {

        for ( int j = servo1PPos; j >= servo1Pos; j--) {   // Run servo down

          servo01.write(j);

          delay(20);    // defines the speed at which the servo rotates

        }

      }

      // If previous position is smaller then current position

      if (servo1PPos < servo1Pos) {

        for ( int j = servo1PPos; j <= servo1Pos; j++) {   // Run servo up

          servo01.write(j);

          delay(20);

        }

      }

      servo1PPos = servo1Pos;   // set current position as previous position

    }

    // Move Servo 2

    if (dataIn.startsWith("s2")) {

      String dataInS = dataIn.substring(2, dataIn.length());

      servo2Pos = dataInS.toInt();

      if (servo2PPos > servo2Pos) {

        for ( int j = servo2PPos; j >= servo2Pos; j--) {

          servo02.write(j);

          delay(50);

        }

      }

      if (servo2PPos < servo2Pos) {

        for ( int j = servo2PPos; j <= servo2Pos; j++) {

          servo02.write(j);

          delay(50);

        }

      }

      servo2PPos = servo2Pos;

    }

    // Move Servo 3

    if (dataIn.startsWith("s3")) {

      String dataInS = dataIn.substring(2, dataIn.length());

      servo3Pos = dataInS.toInt();

      if (servo3PPos > servo3Pos) {

        for ( int j = servo3PPos; j >= servo3Pos; j--) {

          servo03.write(j);

          delay(30);

        }

      }

      if (servo3PPos < servo3Pos) {

        for ( int j = servo3PPos; j <= servo3Pos; j++) {

          servo03.write(j);

          delay(30);

        }

      }

      servo3PPos = servo3Pos;

    }

    // Move Servo 4

    if (dataIn.startsWith("s4")) {

      String dataInS = dataIn.substring(2, dataIn.length());

      servo4Pos = dataInS.toInt();

      if (servo4PPos > servo4Pos) {

        for ( int j = servo4PPos; j >= servo4Pos; j--) {

          servo04.write(j);

          delay(30);

        }

      }

      if (servo4PPos < servo4Pos) {

        for ( int j = servo4PPos; j <= servo4Pos; j++) {

          servo04.write(j);

          delay(30);

        }

      }

      servo4PPos = servo4Pos;

    }

    // Move Servo 5

    if (dataIn.startsWith("s5")) {

      String dataInS = dataIn.substring(2, dataIn.length());

      servo5Pos = dataInS.toInt();

      if (servo5PPos > servo5Pos) {

        for ( int j = servo5PPos; j >= servo5Pos; j--) {

          servo05.write(j);

          delay(30);

        }

      }

      if (servo5PPos < servo5Pos) {

        for ( int j = servo5PPos; j <= servo5Pos; j++) {

          servo05.write(j);

          delay(30);

        }

      }

      servo5PPos = servo5Pos;

    }

    // Move Servo 6

    if (dataIn.startsWith("s6")) {

      String dataInS = dataIn.substring(2, dataIn.length());

      servo6Pos = dataInS.toInt();

      if (servo6PPos > servo6Pos) {

        for ( int j = servo6PPos; j >= servo6Pos; j--) {

          servo06.write(j);

          delay(30);

        }

      }

      if (servo6PPos < servo6Pos) {

        for ( int j = servo6PPos; j <= servo6Pos; j++) {

          servo06.write(j);

          delay(30);

        }

      }

      servo6PPos = servo6Pos;

    }

    // If button "SAVE" is pressed

    if (dataIn.startsWith("SAVE")) {

      servo01SP[index] = servo1PPos;  // save position into the array

      servo02SP[index] = servo2PPos;

      servo03SP[index] = servo3PPos;

      servo04SP[index] = servo4PPos;

      servo05SP[index] = servo5PPos;

      servo06SP[index] = servo6PPos;

      index++;                        // Increase the array index

    }

    // If button "RUN" is pressed

    if (dataIn.startsWith("RUN")) {

      runservo();  // Automatic mode - run the saved steps

    }

    // If button "RESET" is pressed

    if ( dataIn == "RESET") {

      memset(servo01SP, 0, sizeof(servo01SP)); // Clear the array data to 0

      memset(servo02SP, 0, sizeof(servo02SP));

      memset(servo03SP, 0, sizeof(servo03SP));

      memset(servo04SP, 0, sizeof(servo04SP));

      memset(servo05SP, 0, sizeof(servo05SP));

      memset(servo06SP, 0, sizeof(servo06SP));

      index = 0;  // Index to 0

    }

  }

}

* + 1. *Funcția runservo()*

Apăsarea butonului RUN va declanșa funcția runservo(), care rulează pașii stocați. Așadar, aici se vor rula toți pașii stocați prin intermediul butonului SAVE al aplicației în mod repetat până când apăsăm butonul RESET. Astfel, folosind bucla for, se parcurg toate pozițiile stocate în array-uri și în același timp se verifică dacă avem date de intrare de la smartphone. Aceste date pot fi înregistrate cu ajutorul butonului RUN/PAUSE, care pune pe pauză robotul, iar dacă este apăsat din nou, continuă mișcările automate. De asemenea, prin schimbarea poziției cursorului SPEED, noua valoare se poate utiliza pentru a schimba timpul de întârziere între fiecare iterație a buclelor for de mai jos, care controlează viteza servo-motoarelor.

// Automatic mode custom function - run the saved steps

void runservo() {

  while (dataIn != "RESET") {   // Run the steps over and over again until "RESET" button is pressed

    for (int i = 0; i <= index - 2; i++) {  // Run through all steps(index)

      if (Bluetooth.available() > 0) {      // Check for incomding data

        dataIn = Bluetooth.readString();

        if ( dataIn == "PAUSE") {           // If button "PAUSE" is pressed

          while (dataIn != "RUN") {         // Wait until "RUN" is pressed again

            if (Bluetooth.available() > 0) {

              dataIn = Bluetooth.readString();

              if ( dataIn == "RESET") {

                break;

              }

            }

          }

        }

        // If speed slider is changed

        if (dataIn.startsWith("ss")) {

          String dataInS = dataIn.substring(2, dataIn.length());

          speedDelay = dataInS.toInt(); // Change servo speed (delay time)

        }

      }

      // Servo 1

      if (servo01SP[i] == servo01SP[i + 1]) {

      }

      if (servo01SP[i] > servo01SP[i + 1]) {

        for ( int j = servo01SP[i]; j >= servo01SP[i + 1]; j--) {

          servo01.write(j);

          delay(speedDelay);

        }

      }

      if (servo01SP[i] < servo01SP[i + 1]) {

        for ( int j = servo01SP[i]; j <= servo01SP[i + 1]; j++) {

          servo01.write(j);

          delay(speedDelay);

        }

      }

      // Servo 2

      if (servo02SP[i] == servo02SP[i + 1]) {

      }

      if (servo02SP[i] > servo02SP[i + 1]) {

        for ( int j = servo02SP[i]; j >= servo02SP[i + 1]; j--) {

          servo02.write(j);

          delay(speedDelay);

        }

      }

      if (servo02SP[i] < servo02SP[i + 1]) {

        for ( int j = servo02SP[i]; j <= servo02SP[i + 1]; j++) {

          servo02.write(j);

          delay(speedDelay);

        }

      }

      // Servo 3

      if (servo03SP[i] == servo03SP[i + 1]) {

      }

      if (servo03SP[i] > servo03SP[i + 1]) {

        for ( int j = servo03SP[i]; j >= servo03SP[i + 1]; j--) {

          servo03.write(j);

          delay(speedDelay);

        }

      }

      if (servo03SP[i] < servo03SP[i + 1]) {

        for ( int j = servo03SP[i]; j <= servo03SP[i + 1]; j++) {

          servo03.write(j);

          delay(speedDelay);

        }

      }

      // Servo 4

      if (servo04SP[i] == servo04SP[i + 1]) {

      }

      if (servo04SP[i] > servo04SP[i + 1]) {

        for ( int j = servo04SP[i]; j >= servo04SP[i + 1]; j--) {

          servo04.write(j);

          delay(speedDelay);

        }

      }

      if (servo04SP[i] < servo04SP[i + 1]) {

        for ( int j = servo04SP[i]; j <= servo04SP[i + 1]; j++) {

          servo04.write(j);

          delay(speedDelay);

        }

      }

      // Servo 5

      if (servo05SP[i] == servo05SP[i + 1]) {

      }

      if (servo05SP[i] > servo05SP[i + 1]) {

        for ( int j = servo05SP[i]; j >= servo05SP[i + 1]; j--) {

          servo05.write(j);

          delay(speedDelay);

        }

      }

      if (servo05SP[i] < servo05SP[i + 1]) {

        for ( int j = servo05SP[i]; j <= servo05SP[i + 1]; j++) {

          servo05.write(j);

          delay(speedDelay);

        }

      }

      // Servo 6

      if (servo06SP[i] == servo06SP[i + 1]) {

      }

      if (servo06SP[i] > servo06SP[i + 1]) {

        for ( int j = servo06SP[i]; j >= servo06SP[i + 1]; j--) {

          servo06.write(j);

          delay(speedDelay);

        }

      }

      if (servo06SP[i] < servo06SP[i + 1]) {

        for ( int j = servo06SP[i]; j <= servo06SP[i + 1]; j++) {

          servo06.write(j);

          delay(speedDelay);

        }

      }

    }

  }

}

## Concluzie

Brațul robotic controlat de smartphone reprezintă o introducere accesibilă în lumea robotică și a programării Arduino. Prin intermediul componentelor menționate și a aplicației dedicate, utilizatorii pot învăța și experimenta cu principii de mecatronică și automatizare, demonstrând potențialul și versatilitatea platformei Arduino în domeniul roboticelor.

