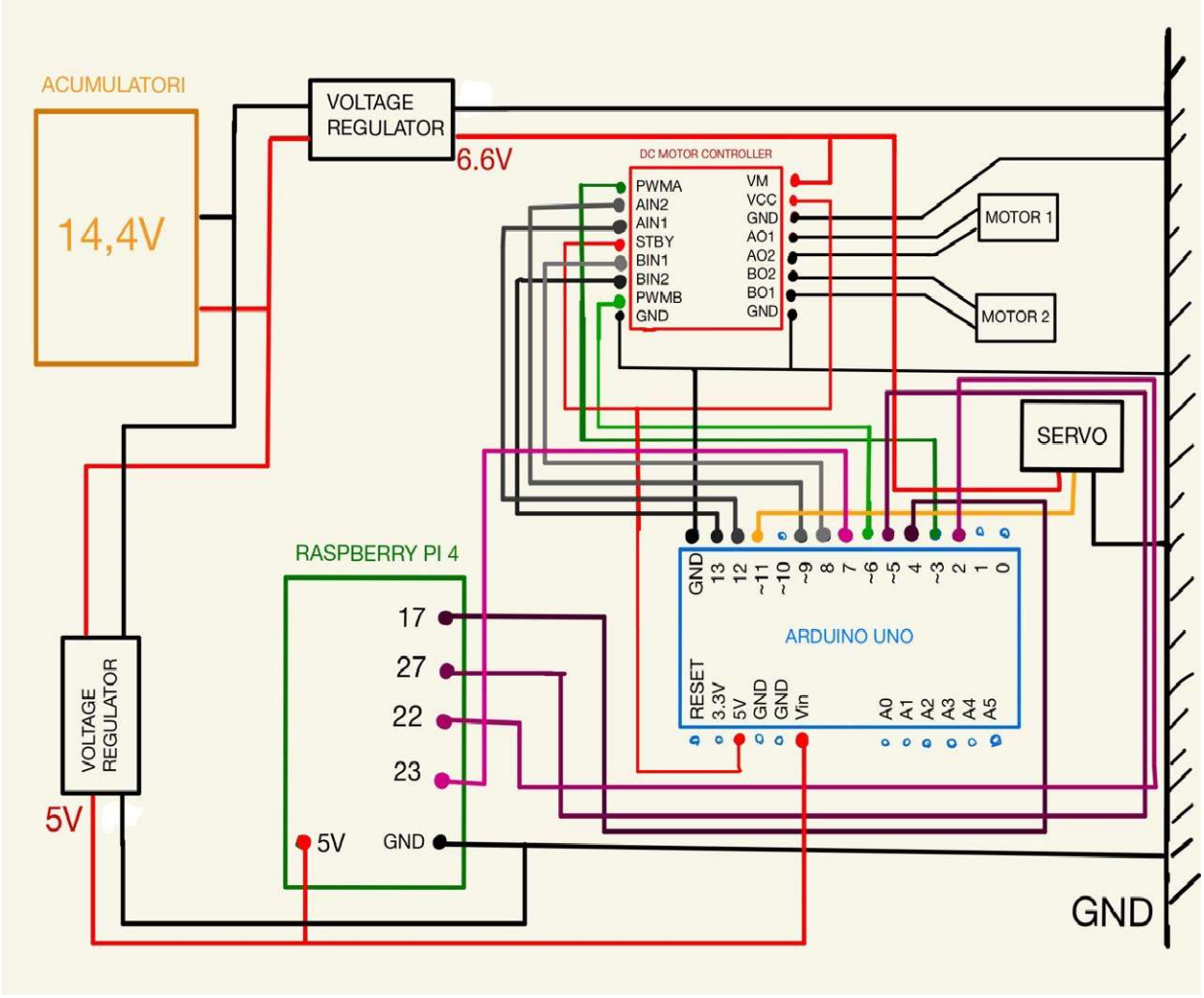
Mașină teleghidată WI-FI

-documentație-

Acest proiect este constituit de o mașină teleghidată prin wi-fi cu o conexiune socket care se stabilește între laptop-ul/desktop-ul care deține programul de control prin tastatură și Raspberry Pi-ul de pe mașină.

Schema de circuit simplificata:



Materiale folosite:

1x motor servo MG996

https://cleste.ro/motor-servo-mg996-13kg-360g.html

1x masinuta cu motoare incorporate

1x Arduino Uno R3

https://cleste.ro/arduino-uno-r3-atmega328p.html

1x Raspberry Pi 4 Model B 4GB

https://ardushop.ro/en/raspberry-pi/1734-raspberry-pi-4-model-b-4-gb.html

jumper wires

https://cleste.ro/10xfire-dupont-mama-tata-20cm.html?utm_medium=GoogleAds&utm_campaign=&utm_source=&gad_source=1&gclid=Cj0KCQiAnfmsBhDfARIsAM7MKi2NiUSdK_Tz2smRxaO_nUHRRSrPvY0uczeSPA3oFU2ZbpzLz18Bl1AaAmKREALw_wcB

module XL4015 si XL4016 regulatoare de voltaj step-down

https://www.optimusdigital.ro/ro/surse-coboratoare-reglabile/7334-modul-sursa-dc-dc-coboratoare-xl4015-cu-afiaj-5-a-i-tensiune-reglabila.html?gclid=Cj0KCQiAnfmsBhDfARIsAM7MKi21EG7H6dbpeyrSQZORyBm2Of1YoO4eCVWYJHKMDrNQukBhPfQznHMaAiJKEALw_wcB

https://www.optimusdigital.ro/ro/surse-coboratoare-reglabile/12643-modul-regulator-de-tensiune-xh-m404-dc-4-40v-8a.html?search_query=Regulator+tensiune&results=105

1x dc motor controller

https://cleste.ro/modul-diver-dual-pentru-motoare-tb6612fng.html?utm_medium=GoogleAds&utm_campaign=&utm_source=&gad_source=1&gclid=Cj0KCQiAnfmsBhDfARIsAM7MKi0gb9nSZvC6YHlkqvEWeUUrQQJG5XanYkqmwOQTIYCcvjjXARorH4EaAjuHEALw_wcB

1x breadboard mic

https://cleste.ro/breadboard-400-puncte.html

4x acumulatori Li-Ion 18650 SONY VTC6 3.6V 3120mAh

https://www.emag.ro/acumulator-li-ion-18650-sony-vtc6-3-6v-3120mah-fara-protectie-pentru-tigara-electronica-18650-3000/pd/D3NQXFBBM/

1x suport acumulatori

https://www.emag.ro/suport-acumulator-3-7v-18650-x-4-sloturi-s18650-4/pd/D8C49WBBM/

Principiul general de functionare:

Mașina utilizează atât un computer Raspberry Pi cat si un microcontroller Arduino Uno. Atât pe laptop cat si pe Raspberry Pi se găsește cod scris in limbajul de programare Java, astfel, partea software, împărțindu-se in 3 “bucăți”:

-partea de pe laptop:

Codul de pe laptop se ocupă cu ascultarea tastelor apăsate de la tastatură, prin intermediul unui KeyListener, implementat într-o fereastră de tip JFrame:

```
JFrame window = new JFrame();
window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
window.setResizable(false);
window.setTitle("Controller");

Display display = new Display(client);
window.add(display);

window.pack();

window.setLocationRelativeTo(null);
window.setVisible(true);

display.keyH.manageDataOut();
```

```
public Display(SocketClient client) {
    this.setPreferredSize(new Dimension(screenWidth, screenHeight));
    this.setBackground(Color.white);
    this.setDoubleBuffered(true);
    this.setFocusable(true);
    keyH = new KeyHandler(client);
    this.addKeyListener(keyH);
}
```

KeyListener-ul înțelege de fiecare dată când o tastă este “pressed” sau “released” si transmite mai departe char-ul asociat tastei apăsate către funcția manageDataOut():

```
    @Override
    public void keyPressed(KeyEvent e) {

        if(e.getKeyChar()=='w'){
            wKeyPressed = true;
        }

        if(e.getKeyChar()=='a'){
            aKeyPressed = true;
        }

        if(e.getKeyChar()=='s'){
            sKeyPressed = true;
        }

        if(e.getKeyChar()=='d'){
            dKeyPressed = true;
        }
```

```java
public void manageDataOut() throws IOException {
    if(wKeyPressed && !aKeyPressed && !dKeyPressed && !sKeyPressed){
        client.setInput(1);
        client.sendData();
        System.out.println("Mergi in fata!");
    }

    else if(!wKeyPressed && !aKeyPressed && !dKeyPressed && sKeyPressed){
        client.setInput(2);
        client.sendData();
        System.out.println("Mergi in spate!");
    }
```

Avem mai sus două exemple despre cum această funcție interpretează succesiunea de taste apăsate sau neapăsate si transmite mai departe către clasa SocketClient numărul corespunzător direcției de mers (exemplu: 0 - stai pe loc, 1 - mergi in fata, 2 - mergi in spate, etc..)

Între Raspberry Pi si Laptop se realizează o conexiune Socket, care stă stabilă pe tot parcursul rulării programului:

```java
public class SocketClient {

    1 usage
    String hostname="192.168.4.1";
    1 usage
    int port = 6666;
    2 usages
    int controlInput=0;
    2 usages
    DataOutputStream out;
    2 usages
    Socket socket;

    1 usage
    public void establishConnection(){
        try {
            socket = new Socket(hostname, port);
        }
        catch(Exception e){
            System.out.println(e);
        }

    }
```

Astfel, în Main există un loop while(true) care in permanență preia date de la KeyListener și transmite mai departe peste socket către mașină datele primite corespunzând direcției de mers:

```
display.keyH.manageDataOut();

while(true){
    display.keyH.manageDataOut();
    Thread.sleep( millis: 50);
}
```

-partea de pe Raspberry PI

Daca pe laptop am avut partea de Client a conexiunii Socket, pe Raspberry, vom avea partea de Server. Raspberry PI funcționează sub sistemul de operare Raspbian, care este un Debian adaptat la nevoile acestui dispozitiv. Pentru ca proiectul să poată fi prezentat fără a fi nevoie de o conexiune externa la vreo rețea WI-FI, a trebuit să trec Raspberry-ul in modul router (sau hotspot), pentru a putea fi detectat de device-urile din împrejurime:

Codul scris in Java, l-am arhivat într-un fișier executabil de tip .jar, iar acesta este accesat automat la start-up (când raspberry-ul pornește) printr-un service (myscript.service) localizat in directorul /etc/systemd/system/. Acesta accesează automat myScript.sh, care conține comanda bash pentru rularea programului: „sudo -i java -jar /home/pi/Desktop/Proiect/Program.jar”.

In funcția main(), se inițializează conexiunea socket si se pregătesc pinii 17,27,23 si 22 pe care ii vom folosi pentru trimiterea datelor către Arduino.

```
SocketSrv socketServer = new SocketSrv();
socketServer.init();

GPIOControl gpioControl = new GPIOControl();
gpioControl.setupPins();
```

Apoi, urmează un loop while() în care în permanență primim input de la laptop, dar nu trimitem către arduino date, decât atunci când input-ul se schimbă în vreun fel. Astfel, dacă, de exemplu, raspberry primește de 10 ori comanda „mergi în față”, el nu va trimite la arduino decât o dată ”mergi în față” si va mai trimite din nou când această comandă se schimbă

```
while (true) {
    log.info("Waiting for connection...");
    socketServer.establishConnection();
    log.info("Established connection, reading data...");

    while (!socketServer.isClosed()) {
        socketServer.receive();

        if (currentDirection != socketServer.direction) {
            currentDirection = socketServer.direction;
            gpioControl.setInput(socketServer.direction);
            gpioControl.managePinOutput();
        }

    }

    log.info("connection closed...");
}
```

```
public void establishConnection() {

    try {
        s = socketServer.accept();//establishes connection
        log.info("Established new connection with {} port {}", s.getInetAddress(), s.getLocalPort());
        dataIn = new DataInputStream(s.getInputStream());
        //socketServer.close();
    } catch (Exception e) {
        log.error("Error in establishConnection()", e);
    }
}

1 usage
public void receive() {
    try {
        direction = dataIn.read();
    } catch (Exception e) {
        log.error("Error in receive", e);
    }
}
```

```
public void managePinOutput() {
    if (input == 1) {

        log.info("fata");
        pin17.high();
        pin27.low();
        pin22.low();
        pin23.low();
    }

    if (input == 2) {
        log.info("spate");
        pin17.low();
        pin27.low();
        pin22.high();
        pin23.low();

    }

    if (input == 3) {
        log.info("fata-stanga");
        pin17.high();
        pin27.high();
        pin22.low();
        pin23.low();

    }

    if (input == 4) {
        log.info("fata-dreapta");
        pin17.high();
        pin27.low();
        pin22.low();
        pin23.high();
```

Setăm pinii de pe raspberry în diferite configurații care reprezintă direcțiile. Aceste configurații vor fi „traduse” mai apoi pe arduino si interpretate.

-partea de pe Arduino

Pe Arduino setam pinii 4,5,2,7 să primească INPUT. Astfel, de câte ori, Raspberry are potențialul la pinii respectivi de 3.3V, vom putea detecta acest lucru:

```
#include <Servo.h>

const int pinW=4;
const int pinA=5;
const int pinS=2;
const int pinD=7;

void setup() {
  //Pinii care primesc input de la Raspberry
  pinMode(4, INPUT);
  pinMode(5, INPUT);
  pinMode(2, INPUT);
  pinMode(7, INPUT);
```

Pentru fiecare situație în care pinii au un voltaj mai mare decât 0, controller-ul interpretează fie în a porni motoarele într-o anume direcție, fie a le porni si a dirija servo-motorul la un unghi prestabilit pentru cârmirea la stânga/dreapta:

```
const int neutral = 90; //unghiul pentru mers in fata
const int left = 45 ;  //roti orientate la stanga
const int right = 135; //roti orientate la dreapta

const int M1front = 12;
const int M1rear = 9;
const int M2front = 8;
const int M2rear = 13;
```

```
void loop() {

  if(digitalRead(pinW)==HIGH){ //daca apas pe w, masina merge in fata
    if(frontSpeed< maxSpeed){
      digitalWrite(M1front, HIGH); //ambele motoare merg in fata
      digitalWrite(M2front, HIGH);
      digitalWrite(M1rear, LOW);
      digitalWrite(M2rear, LOW);

      frontSpeed += accelerationStep; //crestem viteza
      analogWrite(frontMotorPin, frontSpeed); //setam noua viteza la motor
      analogWrite(rearMotorPin, frontSpeed);
      delay(100); //lasam putin timp ca acceleratia sa aiba timp sa se observe
    }
  }
```

```
if(digitalRead(pinA)==HIGH){
  if(angle!=left){
    angle=left;
    direction.write(left);

  }
}

if(digitalRead(pinD)==HIGH){
  if(angle!=right){
    angle=right;
    direction.write(right);
  }
}

if(digitalRead(pinA)==LOW && digitalRead(pinD)==LOW){
  if(angle!=neutral){
    angle=neutral;
    direction.write(neutral);

  }
}
```

Când mașina stă, deci nicio tastă (W, A, S sau D) nu este apăsată, motoarele încep să decelereze cu un pas (30) de la viteza curenta pana la oprire.

```
if(digitalRead(pinW) == LOW && digitalRead(pinS) == LOW){
  digitalWrite(M1front, LOW); //ambele motoare OFF
  digitalWrite(M2front, LOW);
  digitalWrite(M1rear, LOW);
  digitalWrite(M2rear, LOW);
  if(rearSpeed>0){
    if(rearSpeed - deccelerationStep >=0)
      rearSpeed -=  deccelerationStep;
    else
      rearSpeed -= 15;

  }
  if(frontSpeed>0){
    if(frontSpeed - deccelerationStep >=0)
      frontSpeed -=  deccelerationStep;
    else
      frontSpeed -= 15;

  }
}
```

<u>Scenariu practic:</u>

Această mașină este, în principiu, mai mult un proiect cu scop didactic, în al cărui proces de realizare, am învățat:

-elemente de rețelistică

-să utilizez anumite clase utile în Java

-desfășurarea în Linux Command Line

-să depistez probleme, fie ele legate de cod sau de partea hardware, utilizând multimetrul sau leduri

-să citesc datasheet-uri

-să am grijă la parametrii de funcționare ai pieselor pe care le folosesc

Luând în considerare toate cele menționate mai sus, mașina, în urma unor mici perfecționări pe partea de siguranță și aspect, ar putea servi drept o foarte performantă și foarte scumpă jucărie pentru copii.

Vă mulțumesc!