Evidencia Final Entrega Reto: Evidencia Reto Final

Diseño de sistemas en chips: TE2003B.601

Grupo 601

Profesor

Agustín Domínguez Oviedo
Pedro Daniel Ferrusca Monroy

Estudiantes

Alexa Jimena González Lucio - A01277701

Laura Helena Molina Jiménez - A01706282

Sebastián Castellanos Rodríguez - A01710226

13 de junio del 2024

Final Project

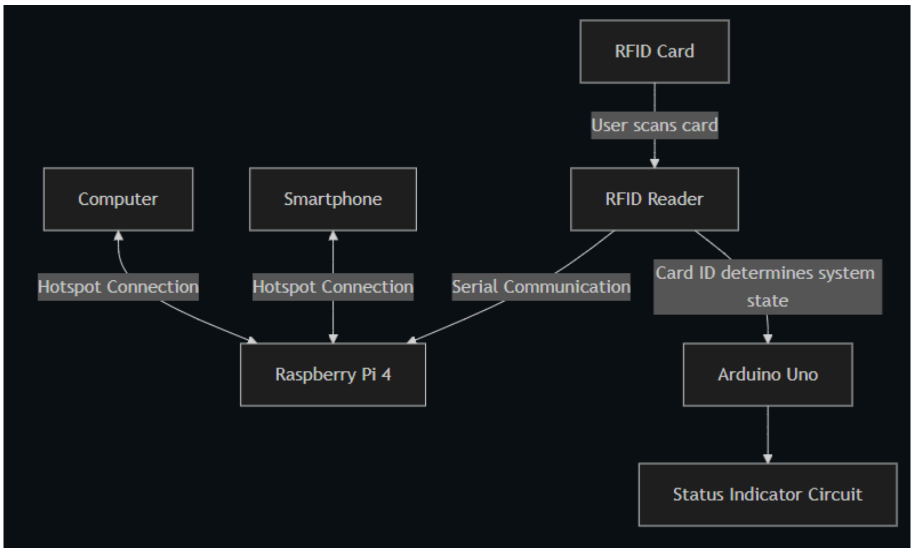
As a challenge for the final assessment, we chose to develop an RFID-based access control system that provides visual monitoring through an interface, allowing the identification of users who enter and exit, as well as those who are not registered. To develop and manage this project, a Raspberry Pi 4 was used due to its low cost and compact size compared to other computers.

We consider this project to be beneficial for the community, as it improves security and provides precise control over people attempting to access a location, as well as those who successfully enter or leave and the time at which these events occur. Having this information can be highly valuable in emergency situations or for crime prevention.

Materials

- The materials used for this project are listed below:
- Jumper wires
- Arduino Uno (status indicator circuit – RFID reader / RFID card)
- Raspberry Pi 4
- Breadboard
- 4 LEDs (green, red, yellow, and blue)
- Potentiometer
- 16x2 LCD display
- LED matrix
- RFID reader
- RFID cards

System Flow Diagram



System Operation

We decided to implement an Arduino Uno to program the circuit responsible for displaying the RFID reader's response. For this purpose, jumper wires were used to connect the Arduino to the breadboard. Additionally, four LEDs of different colors were used to indicate the current status of the card reading process.

Blue: Represents the waiting state, indicating that the circuit is powered on and waiting for RFID cards to be scanned.

Green: Indicates that the scanned card is accepted, meaning it is registered in the system.

Red: Indicates that the scanned card is not accepted, as it is not registered.

Yellow: Indicates that it is the first time during the day that the scanned card has been read (regardless of whether it is accepted or rejected).

We also implemented an LED matrix to provide additional visual feedback. The matrix was programmed to illuminate specific LEDs in order to display patterns corresponding to the system's state changes.

Circle: This pattern remains lit while the circuit is waiting for a card to be scanned (active simultaneously with the blue LED).

Check mark: This pattern is displayed whenever a scanned card is accepted (active simultaneously with the green LED).

Cross Symbol: This pattern is displayed whenever a scanned card is rejected (active simultaneously with the red LED).

Finally, we added a 16x2 LCD display to show messages indicating the current status of the RFID reader.

- Message ("Access Control"): Displayed while the system is waiting for a card to be scanned.
- Message ("Access Granted"): Displayed when a registered card is scanned.
- Message ("Access Denied"): Displayed when an unregistered card is scanned.

Summary Table of Simultaneous Operation

Component	LED's	LED Matriz	16x2 LCD Display
Idle Mode	Blue LED ON	Circle pattern	"Access Control" message
Access Granted Mode	Green LED ON	Check mark pattern	"Access Granted" message
Access Denied Mode	Red LED ON	Cross pattern	"Access Denied" message
First Scan Detection Mode	Yellow LED ON	—	—

Now, the RFID reader is connected to the Raspberry Pi through a serial interface. In this way, the access interfaces register the users who scan their cards, the status of their access, and the time of access. Likewise, in order to enable serial communication between the RFID reader and the Raspberry Pi, a dedicated interface was designed and developed. This interface allows the user to select the device to be connected, configure the serial communication baud rate, and visualize the serial number of the card being scanned, ensuring that the communication has been correctly established.

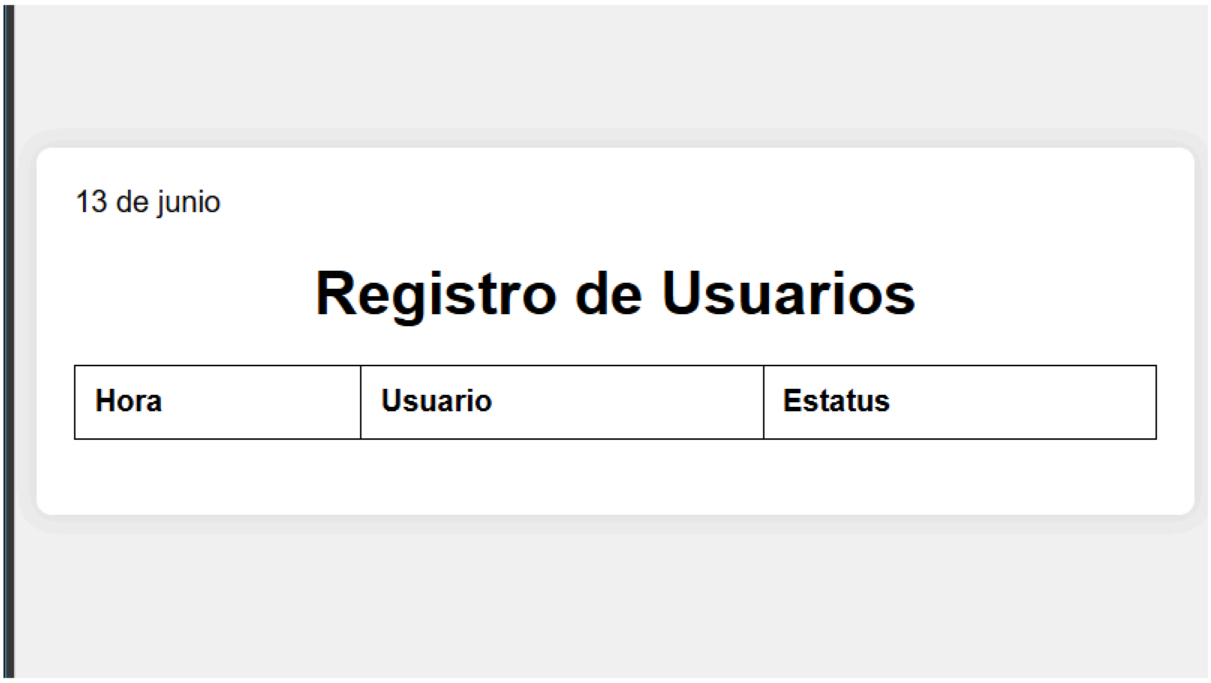
System Interfaces

Regarding the visual component of the access control system, three interfaces were developed using Python. The first one is the interface responsible for connecting to the RFID device that reads the cards. This interface receives the card code through serial communication and sends it to a web server, which can be accessed from a smartphone or a computer connected to the same local network using the Raspberry Pi's local IP address.

Additionally, a socket communication server interface was implemented using Python and Tkinter. To enable socket communication, a client was integrated into the web server. In other words, the web server also acts as a client that sends user access records to the socket server. It is worth mentioning that the web server was developed using FastAPI, with HTML used for the visual interface.

In this way, all communication specifications within the system were integrated in a synchronized and efficient manner.

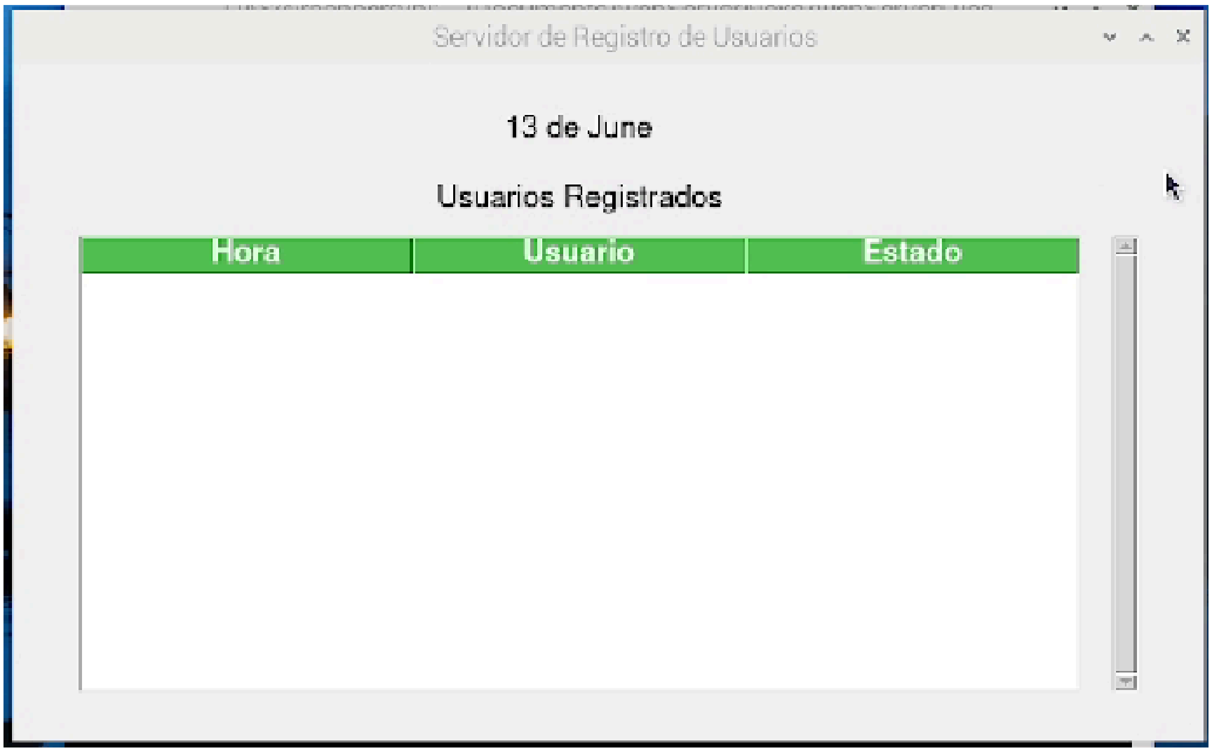
Web Server


13 de junio

Registro de Usuarios

Hora	Usuario	Estatus

Socket Server

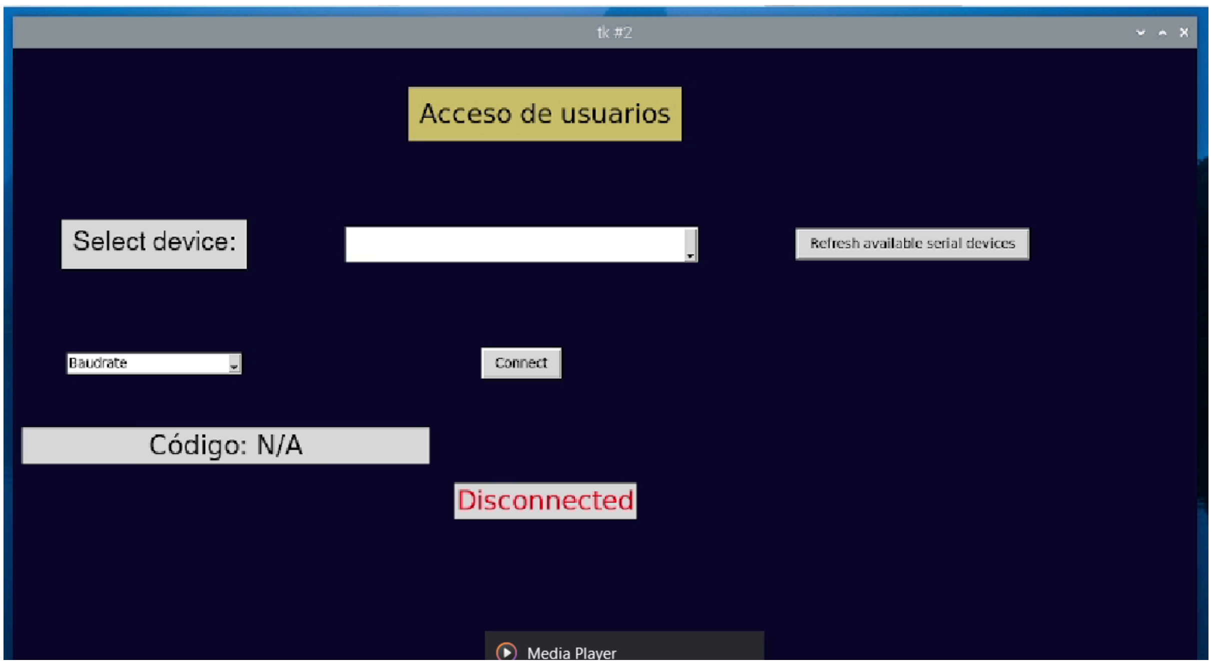


13 de June

Usuarios Registrados

Hora	Usuario	Estado

RFID Connection

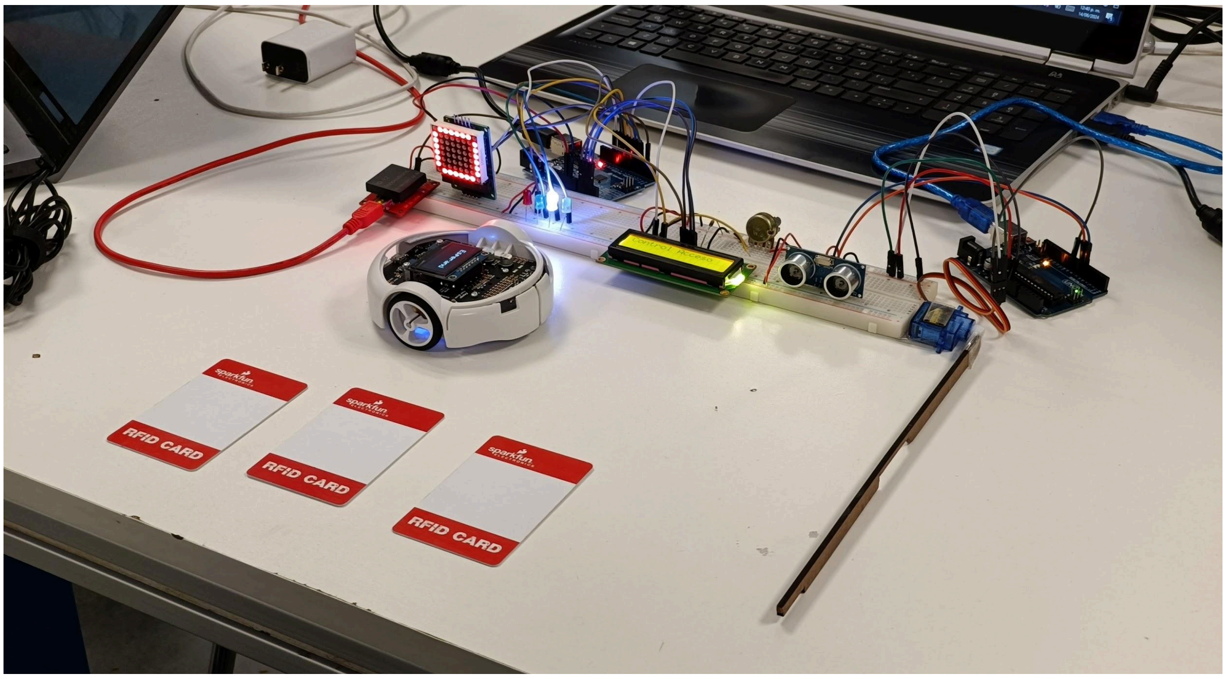


Acceso de usuarios

Código: N/A

Disconnected

Final Prototype



Arduino Code

- Activation of LEDs, LED Matrix, and LCD Display

```
#include <SoftwareSerial.h>
#include <LedControl.h>
#include <LiquidCrystal.h>

SoftwareSerial rSerial(2, 3);

const int greenLED = 4;
const int redLED = 5;
const int blueLED = 6;
const int yellowLED = 7;

const int DIN_PIN = 11;
const int CS_PIN = 10;
const int CLK_PIN = 13;

LedControl lc = LedControl(DIN_PIN, CLK_PIN, CS_PIN, 1);

const int rs = 8, en = 9, d4 = A0, d5 = A1, d6 = A2, d7 = A3;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

const int tagLen = 16;
const int idLen = 13;
const int kTags = 4;

char knownTags[kTags][idLen] = {
  // Store RFID known tags.
  "4F0088B20772",
  "444444444444",
  "555555555555",
  "7A005B0FF8D6"
};

char unknownTags[10][idLen];
int unknownTagCount = 0;

char newTag[idLen];

byte checkMark[8] = {B00010000, B00100000, B01000000, B00100000,
B00010000, B00001000, B00000100, B00000010};
```

```
byte circle[8] = {B01111110, B10000001, B10000001, B10000001,
B10000001, B10000001, B10000001, B01111110};

byte cross[8] = {B10000001, B01000010, B00100100, B00011000,
B00011000, B00100100, B01000010, B10000001};

void setup() {
  // Configures the serial communication, initializes the LED
module and the LCD display, sets the LED pins as outputs, and
executes a welcome sequence.
  Serial.begin(9600);
  rSerial.begin(9600);

  lc.shutdown(0, false);
  lc.setIntensity(0, 8);
  lc.clearDisplay(0);

  lcd.begin(16, 2);
  lcd.print("Control de acceso");

  pinMode(greenLED, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(blueLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);

  welcomeSequence();

  digitalWrite(blueLED, HIGH);
  displayLEDs(0, circle);
}

void loop() {
  // Runs continuously and checks whether data is available on
the serial communication. If an RFID tag is detected, it is
processed using the readRFID() function.
  if (rSerial.available() == tagLen) {
    digitalWrite(blueLED, LOW);
    readRFID();
    digitalWrite(blueLED, HIGH);
    lcd.clear();
    lcd.print("Control Acceso");
  }
}
```

```
void welcomeSequence() {
  // Performs a welcome sequence by turning the LEDs on and off
in a specific order.
  digitalWrite(greenLED, HIGH);
  delay(500);
  digitalWrite(greenLED, LOW);
  delay(500);
  digitalWrite(blueLED, HIGH);
  delay(500);
  digitalWrite(blueLED, LOW);
  delay(500);
  digitalWrite(yellowLED, HIGH);
  delay(500);
  digitalWrite(yellowLED, LOW);
  delay(500);
  digitalWrite(redLED, HIGH);
  delay(500);
  digitalWrite(redLED, LOW);
  delay(500);
}

void readRFID() {
  // Reads the received RFID tag, checks whether it is recognized
or unrecognized, and acts accordingly by activating specific LEDs
and displaying messages on the LCD screen.
  int i = 0;
  while (rSerial.available()) {
    int byteRead = rSerial.read();
    if (byteRead != 2 && byteRead != 13 && byteRead != 10 &&
byteRead != 3 && i < idLen - 1) {
      newTag[i++] = byteRead;
    }
  }
  newTag[i] = '\0';

  if (checkKnownTag(newTag)) {
    Serial.println("Acceso concedido");
    Serial.print("Tarjeta aprobada: ");
    Serial.println(newTag);
    displayLEDs(0, checkMark);
    digitalWrite(greenLED, HIGH);
    lcd.clear();
    lcd.print("Concedido");
    if (!isTagPreviouslyUnknown(newTag)) {
      Serial.println("Primera lectura del día.");
      addUnknownTag(newTag);
      digitalWrite(yellowLED, HIGH);
    }
    delay(500);
    digitalWrite(greenLED, LOW);
    digitalWrite(yellowLED, LOW);
  } else {
    Serial.print("Tarjeta no registrada: ");
    Serial.println(newTag);
    displayLEDs(0, cross);
    digitalWrite(redLED, HIGH);
    lcd.clear();
    lcd.print("  Denegado");
    if (!isTagPreviouslyUnknown(newTag)) {
      Serial.println("Primera lectura del día.");
      addUnknownTag(newTag);
      digitalWrite(yellowLED, HIGH);
    }
    delay(500);
    digitalWrite(redLED, LOW);
    digitalWrite(yellowLED, LOW);
  }


  for (int i = 0; i < idLen; i++) {
    newTag[i] = 0;
  }

  displayLEDs(0, circle);
}

bool checkKnownTag(char tag[]) {
  // Verifies if a tag is known
  for (int i = 0; i < kTags; i++) {
    if (strncmp(tag, knownTags[i], idLen - 1) == 0) {
      return true;
    }
  }
  return false;
}
```

```
bool isTagPreviouslyUnknown(char tag[]) {
  // Verifies if a tag is not known
  for (int i = 0; i < unknownTagCount; i++) {
    if (strncmp(tag, unknownTags[i], idLen - 1) == 0) {
      return true;
    }
  }
  return false;
}

void addUnknownTag(char tag[]) {
  // Adds unknown tags to the list
  if (unknownTagCount < 10) {
    strncpy(unknownTags[unknownTagCount], tag, idLen - 1);
    unknownTagCount++;
  }
}


void displayLEDs(int device, byte pattern[8]) {
  // This function is used to display a pattern on an LED matrix.
  for (int row = 0; row < 8; row++) {
    lc.setRow(device, row, pattern[row]);
  }
}
```

- Sensor ultrasónico y servo motor para simular funcionamiento de plumilla.

```
#include <Servo.h>

Servo myservo;
int cm = 0;

long readUltrasonicDistance(int triggerPin, int echoPin){
  // This is a function that sends an ultrasonic pulse and
measures the time it takes to return, thereby calculating the
distance to an object.
  pinMode(triggerPin, OUTPUT);
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);
  return pulseIn(echoPin, HIGH);
}

void setup() {
  // It is the setup function that runs once when the program
starts.
  pinMode(12, OUTPUT);
  digitalWrite(12, LOW);
  myservo.attach(9);
  myservo.write(270);
  Serial.begin(9600);
}

void loop() {
  // It is the main function that runs repeatedly. It measures
the distance with the ultrasonic sensor and prints it to the
serial monitor.
  cm = 0.01723 * readUltrasonicDistance(6, 7);
  Serial.print("Distancia medida: ");
  Serial.print(cm);
  Serial.println(" cm");

  if (cm < 10) {
    Serial.println("Objeto detectado dentro de 10 cm. Levantando
pluma.");

    for (int pos = 270; pos >= 90; pos -= 1) {
      myservo.write(pos);
      delay(30);
    }
    delay(4000);


    for (int pos = 90; pos <= 270; pos += 1) {
      myservo.write(pos);
      delay(30);
    }
    delay(500);
  } else {
    Serial.println("No se detecta objeto dentro de 10 cm.");
  }
  delay(500);
}
```

Pololu Robot code

```
#include <Pololu3piPlus32U4.h>

using namespace Pololu3piPlus32U4;

OLED display;
Buzzer buzzer;
ButtonA buttonA;
ButtonB buttonB;
ButtonC buttonC;
LineSensors lineSensors;
BumpSensors bumpSensors;
Motors motors;
Encoders encoders;

bool wait_signal = false;

void setup() {

  display.setLayout8x2();
  display.noAutoDisplay();
}

void loop() {
  //Check if B button is pressed
  if(buttonB.getSingleDebouncedPress()){

    display.clear();//Clean screen
    motors.setSpeeds(50,50);
    delay(1200);
    wait_signal = true;

  }else{
    //Receive message
    if(wait_signal){
      display.gotoXY(0, 0);
```

```
        display.print("Esperando");
        motors.setSpeeds(0,0);
        display.display();//Show message in display
        wait_signal = false;

      }
    }

}
```

Conclusions

The development of this final project represented the culmination of the knowledge acquired throughout the semester, allowing us to integrate concepts from embedded Linux, microcontrollers, Arduino programming, communication protocols, and software development into a single functional system. By choosing an RFID-based access control system, we were able to design a solution with a clear real-world application, combining hardware and software components such as the Raspberry Pi 4, Arduino Uno, visual indicators (LEDs, LED matrix, and LCD display), and multiple interfaces. The project challenged us to transform raw sensor data into meaningful visual feedback and accessible records, reinforcing our understanding of how different technologies interact within an integrated system.

Additionally, this project strengthened our teamwork, communication, and problem-solving skills. Coordinating multiple deliverables within a limited timeframe required effective task distribution based on individual strengths, as well as constant collaboration. From experimenting with electronic components and synchronizing their behavior, to developing Python-based interfaces using Tkinter, FastAPI, and socket communication, each stage contributed to our technical growth. Despite the challenges and workload, this experience was highly rewarding and helped clarify our interests within the field, particularly in software development, embedded systems, and microcontrollers. Overall, this project has strong potential for future expansion and served as a valuable learning experience that will be highly beneficial in our professional development.

Demo Video

🎬 SistemaControlAcceso
https://drive.google.com/file/d/1fr2G8CZ9CwhPZrF2a_m_3ezAevOk_d_a/view