Tecnológico de Monterrey

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

Reto Final

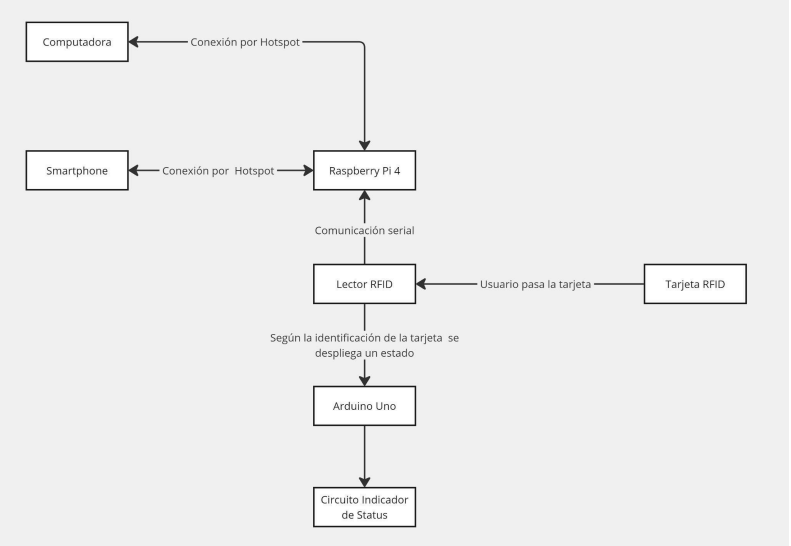
Como reto para la evidencia final, se optó por crear un sistema de acceso mediante un sistema RFID en el cual se pudiera tener control visual mediante una interfaz, de los usuarios que entran y salen, así como aquellos que no están registrados. Para poder desarrollar y controlar este proyecto, se utilizó una Raspberry Pi 4 debido a su bajo costo y tamaño compacto en comparación a otras computadoras. Consideramos que este proyecto es beneficioso para la comunidad, pues permite tener una mejor seguridad así como un control preciso de las personas que intentan acceder, así como aquellas que acceden y salen y a qué hora sucede esto. Tener esta información puede ser beneficioso en cualquier emergencia o prevención de delitos, crímenes

Materiales

Los materiales utilizados para este proyecto fueron los siguientes:

- Jumpers
- Arduino Uno (Circuito indicador de status - Lector RFID / Tarjeta RFID)
- Raspberry Pi 4
- Protoboard
- 4 leds (verde, rojo, amarillo y azul)
- Potenciómetro
- Display LCD 16x2
- Matriz de leds
- Lector RFID
- Tarjetas RFID

Diagrama de bloques



Funcionamiento

Decidimos implementar un Arduino Uno para programar el circuito responsable de mostrar la respuesta del Lector RFID, para esto usamos jumpers para realizar las conexiones del Arduino al protoboard, usamos 4 LED's de distintos colores para indicar el estado en que se encuentra la lectura de las tarjetas.

- Azul: Este representa el estado de espera, refleja que el circuito está encendido y que está esperando que se pasen tarjetas para leerlas.
- Verde: Indica al usuario que la tarjeta leída si es aceptada, es decir si está registrada.
- Rojo: Indica al usuario que la tarjeta leída no es aceptada, no está dada de alta.
- Amarillo: Le indica al usuario que es la primera vez en el día que se hace una lectura de la tarjeta que se acaba de pasar por el lector (independientemente de si es aceptada o rechazada).

También implementamos una matriz de LED's con el fin de que funcionara como un apoyo visual adicional, fue programado para prender los LED's exactos para generar dibujos que aparecen al momento en que se hacen los cambios de estado.

- Círculo: Esta forma se mantiene encendida mientras que el circuito esté en espera de la lectura de tarjeta (Se mantiene encendido a la par que el LED azul).
- Palomita: Esta forma se genera cada vez que una tarjeta escaneada es aceptada (Se enciende a la vez que se enciende el LED verde).
- Tache: Esta forma se genera cada vez que una tarjeta escaneada es rechazada (Se enciende a la vez que se enciende el LED rojo).

Finalmente decidimos agregar un Display LCD 16x2 donde imprimimos mensajes que indican el estado del lector de tarjetas.

- Mensaje ("Control Acceso"): Este mensaje se muestra en el display siempre que el circuito se encuentre en espera de una lectura de tarjeta.
- Mensaje ("Acceso Concedido"): Este mensaje se muestra en el display cada vez que se escanea una tarjeta registrada.
- Mensaje ("Acceso Denegado"): Este mensaje se muestra en el display cada vez que se escanea una tarjeta que no está registrada.

Tabla de resumen de funcionamiento simultáneo

Componente	LED's	Matriz de LED's	Display LCD 16x2
Modo de espera	Encendido el azul	Figura de círculo	Mensaje "Control Acceso"
Modo de acceso concedido	Encendido el verde	Figura de palomita	Mensaje "Acceso Concedido"
Modo de acceso denegado	Encendido el rojo	Figura de tache	Mensaje "Acceso Denegado"
Modo detección de primer registro	Encendido el amarillo	—	—

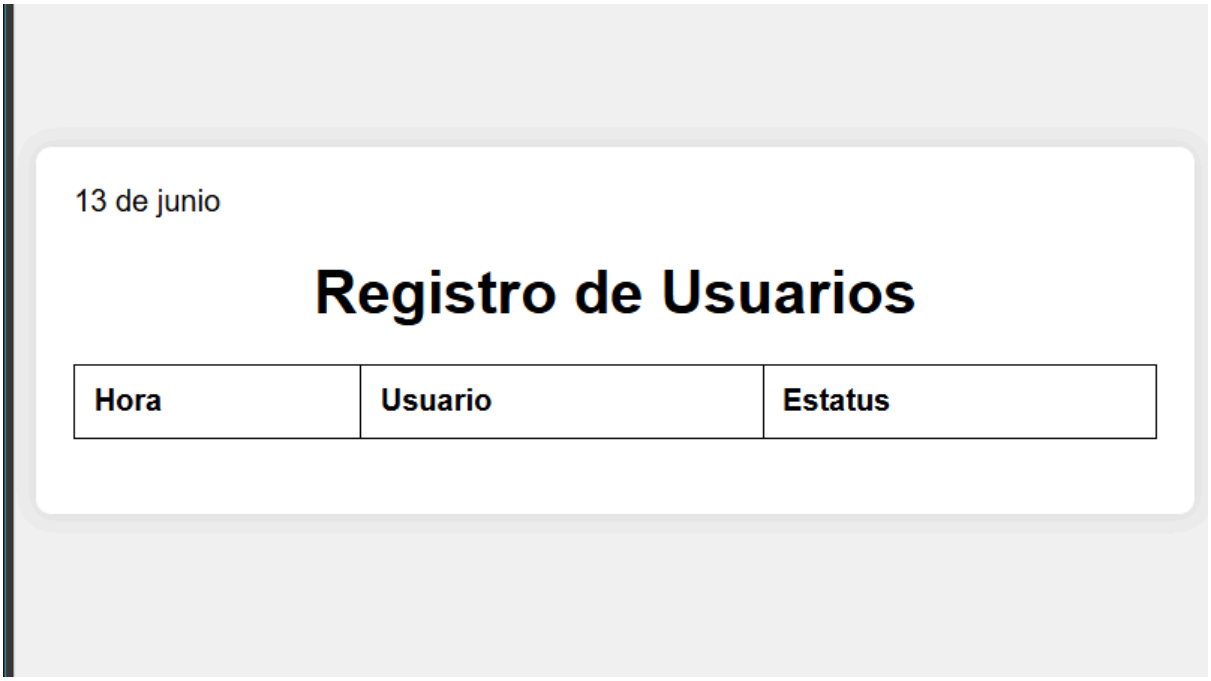
Ahora bien, el Lector RFID se encuentra conectado a la Raspberry de forma serial, de esta forma dentro de las interfaces de acceso se registran los usuarios que han pasado la tarjeta, el estatus de su acceso y la hora de acceso Asimismo, con el objetivo de poder conectar el Lector a la Raspberry de forma serial, se diseñó y creó una interfaz en la que se pueden seleccionar el dispositivo a conectar, la tasa de baudios de la comunicación serial y además, permite visualizar el número de serie de la tarjeta que está pasando por el lector, para verificar que se haya establecido de forma correcta la comunicación.

Interfaces del sistema

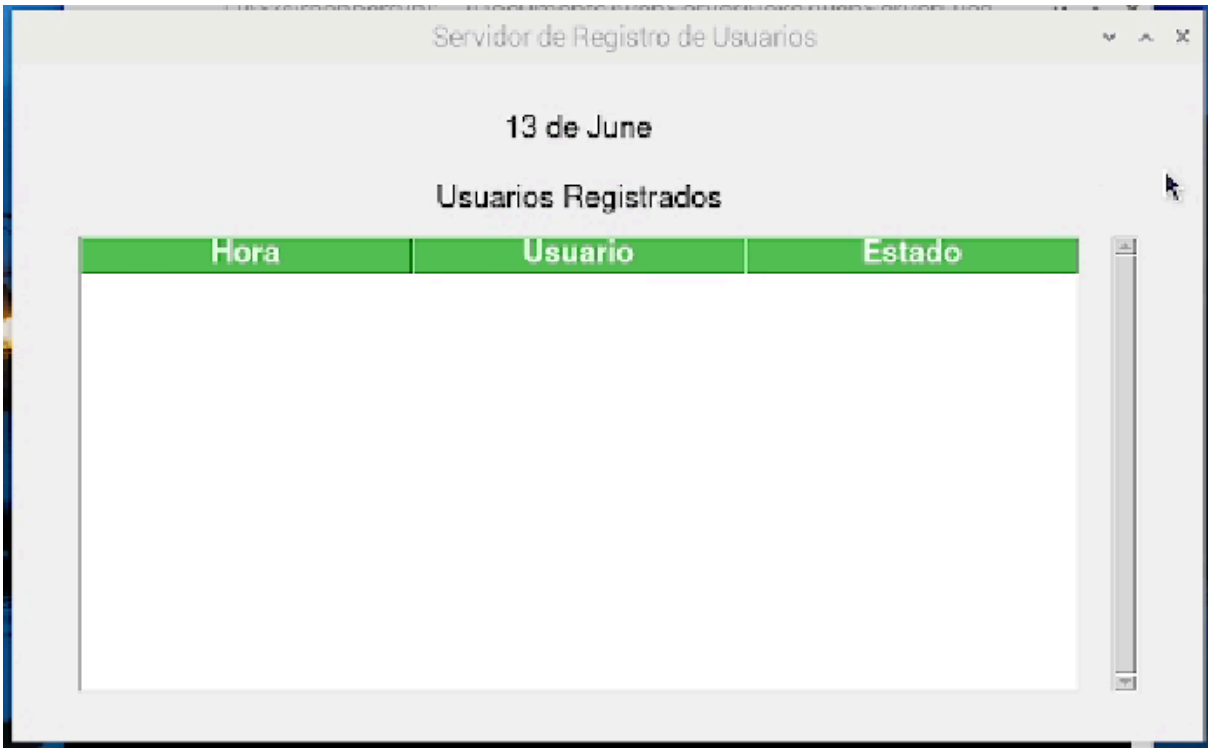
En cuanto a la parte visual del sistema de control de acceso se desarrollaron 3 interfaces con python. La primera es la interfaz de conexión al dispositivo rfdi que lee las tarjetas, esta interfaz recibe el código de la tarjeta de forma serial y lo envía al servidor web al cual se puede acceder desde un smartphone o una computadora conectada a la misma red local mediante la ip local de la raspberry pi. Por último, también se integró una interfaz de servidor de comunicación por sockets elaborada con python y tkinter. Para lograr la comunicación por sockets, se integró el cliente al servidor web. Es decir, el servidor web, es a la vez un cliente que envía los registros de los usuarios al servidor de socket. Cabe mencionar que el servidor web fue desarrollado con FastAPI y html para la parte visual.

De esta forma se integraron todas las especificaciones de comunicación en el sistema de forma sincronizada y eficiente.

Servidor web

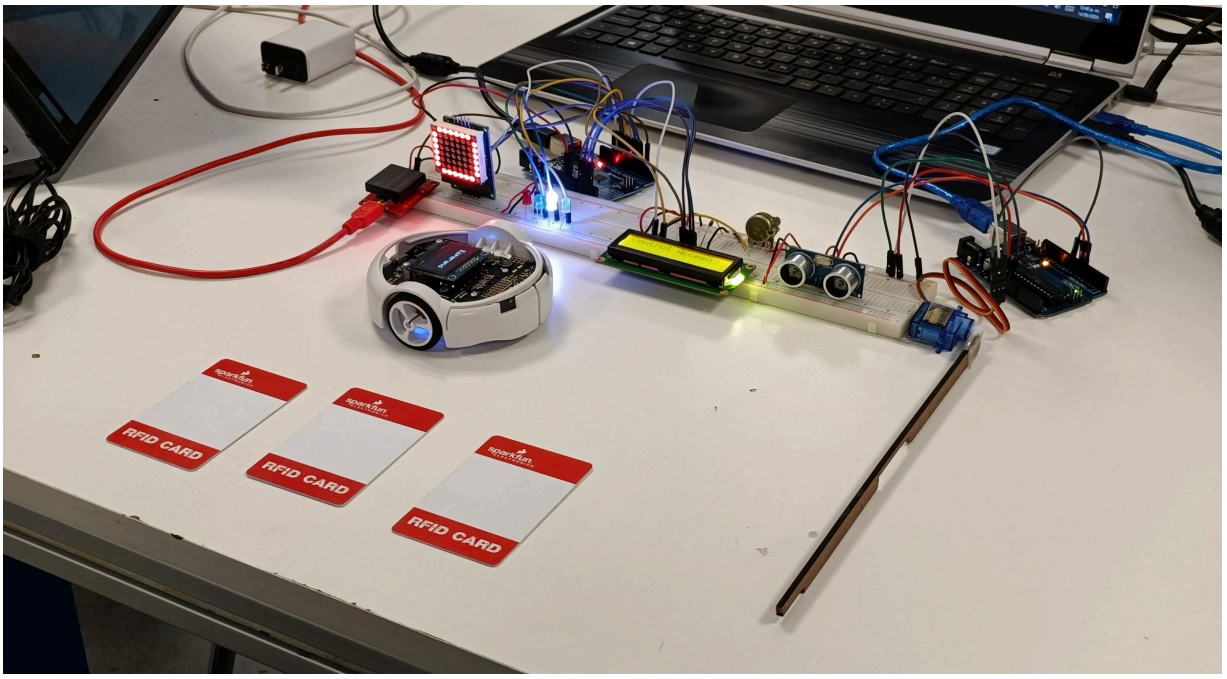


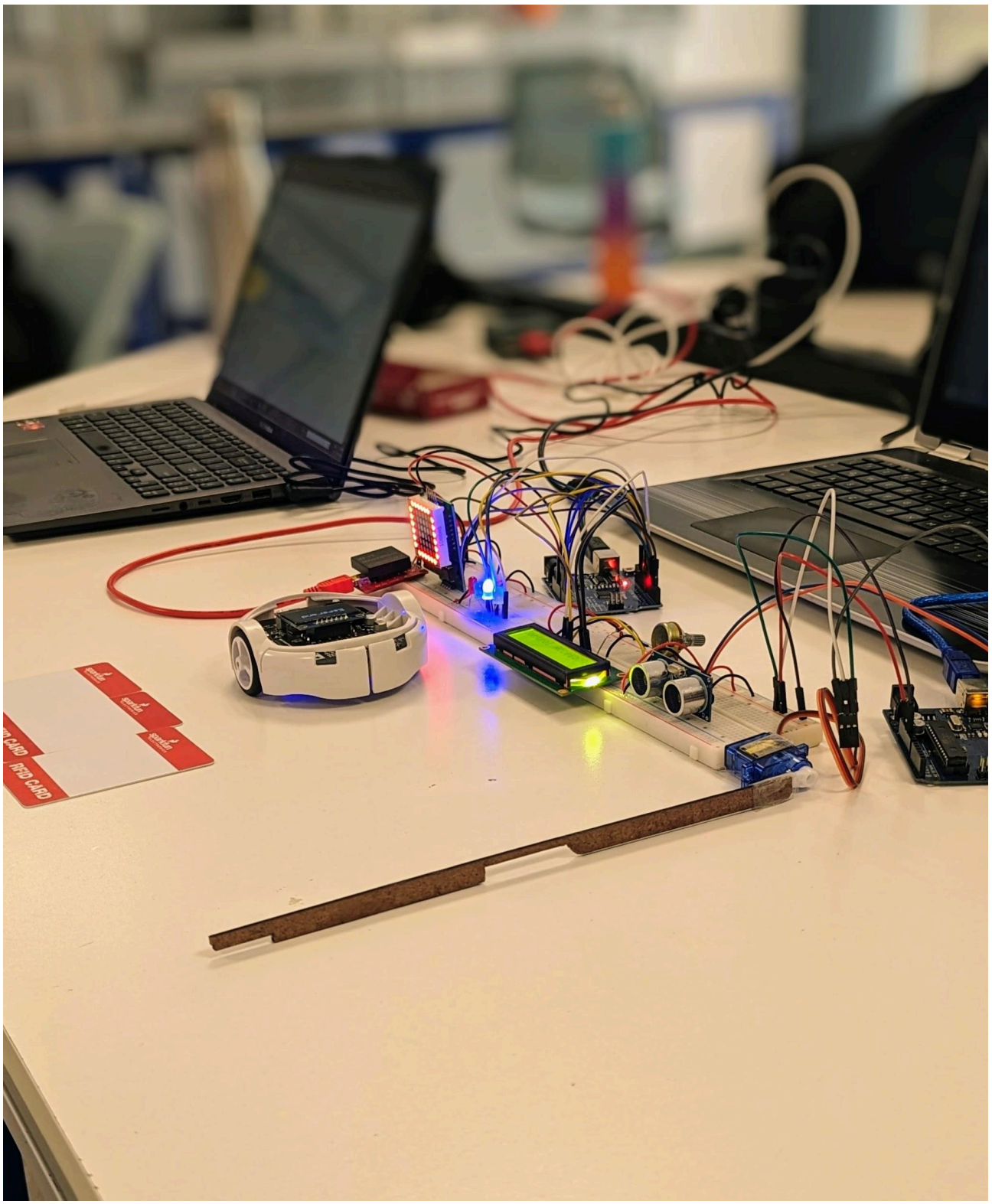
Servidor de socket



Conexión con el RFID

Circuito funcionando




RED CARD

Código Arduino

- Activación de Leds, matriz de leds y display LCD

```
#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] = {
  // Almacenan las etiquetas RFID conocidas.
  "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() {
  // Configura la comunicación serial, inicializa el módulo de
LED y la pantalla LCD, configura los pines de los LEDs como
salidas y ejecuta una secuencia de bienvenida.
  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() {
  // Se ejecuta continuamente y verifica si hay datos disponibles
en la comunicación serial. Si se detecta una etiqueta RFID, se
procesa con la función readRFID().
  if (rSerial.available() == tagLen) {
    digitalWrite(blueLED, LOW);
    readRFID();
    digitalWrite(blueLED, HIGH);
    lcd.clear();
    lcd.print("Control Acceso");
  }
}
```

```
void welcomeSequence() {
  // Realiza una secuencia de bienvenida encendiendo y apagando los LEDs en un orden específico.
  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() {
  // Lee la etiqueta RFID recibida, verifica si es conocida o desconocida y actúa en consecuencia encendiendo LEDs específicos y mostrando mensajes en la pantalla LCD.
  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[]) {
  // Verifica si una etiqueta es conocida.
  for (int i = 0; i < kTags; i++) {
    if (strncmp(tag, knownTags[i], idLen - 1) == 0) {
      return true;
    }
  }
  return false;
}
```

```
bool isTagPreviouslyUnknown(char tag[]) {
  // Verifica si una etiqueta no es conocida.
  for (int i = 0; i < unknownTagCount; i++) {
    if (strncmp(tag, unknownTags[i], idLen - 1) == 0) {
      return true;
    }
  }
  return false;
}

void addUnknownTag(char tag[]) {
  // Agrega etiquetas desconocidas a una lista.
  if (unknownTagCount < 10) {
    strncpy(unknownTags[unknownTagCount], tag, idLen - 1);
    unknownTagCount++;
  }
}


void displayLEDs(int device, byte pattern[8]) {
  // Esta función se utiliza para mostrar un patrón en una matriz
de LEDs.
  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){
  // Esta es una función que envía un pulso ultrasónico y mide el
tiempo que tarda en volver, calculando así la distancia a un
objeto.
  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() {
  // Es la función de configuración que se ejecuta una vez al
iniciar el programa.
  pinMode(12, OUTPUT);
  digitalWrite(12, LOW);
  myservo.attach(9);
  myservo.write(270);
  Serial.begin(9600);
}

void loop() {
  // Es la función principal que se ejecuta repetidamente. Mide
la distancia con el sensor ultrasónico y la imprime en el monitor
serial.
  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);
}
```

Enlace al GitHub donde están los códigos de las interfaces:

https://github.com/alexaGonLuc16/linux-embebido-601-equipo4-testing/blob/main/webServerFiles/tarjetas.py

Código Pololu

```
#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 espera = false;

void setup() {

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

void loop() {
  //Revisar si el botón B está presionado
  if(buttonB.getSingleDebouncedPress()){

    display.clear();//borrar la pantalla
    motors.setSpeeds(50,50);
    delay(1200);
    espera = true;
```

```
  }else{
    //Escribimos el mensaje
    if(espera){
      display.gotoXY(0, 0);
      display.print("Esperando");
      motors.setSpeeds(0,0);
      display.display();//desplegar los mensaje en el display
      espera = false;

    }
  }

}
```

Conclusiones

Sebastián: La realización de este proyecto final fue sin duda alguna muy interesante y entretenida, fue la culminación de todos los conocimientos adquiridos a lo largo del semestre, nosotros decidimos buscar una implementación que nos permitiera utilizar los aprendizajes de cada uno de los módulos del bloque y al ver los sensores que tenía el profe el primero que nos llamó la atención fue el sensor RFID, aprendimos sobre su funcionamiento y pensamos en una manera de hacer visual los registros que se realizaban, al mismo tiempo que se desarrolló una interfaz y un registro donde también se pudieran ver estas lecturas.

Por parte del circuito, inicialmente se realizó un prototipo utilizando el ATmega328P, pero se decidió programar el funcionamiento del circuito únicamente con un Arduino, para la implementación física se programó la reacción de leds de distintos colores, el encendido y apagado de una matriz de leds para formar figuras y el desplegado de mensajes en un display LCD, todo esto con la finalidad de mostrar de manera visual cuando un acceso era concedido, cuando era denegado o cuando el circuito se encontraba en espera de una lectura; debo admitir que esta fue la parte más divertida para mí ya que pude experimentar con el uso de componentes electrónicos que no había usado antes y era fascinante ver como la activación de cada uno de ellos se podía sincronizar.

Finalmente tras concluir con el circuito conectado al lector RFID realizamos un segundo circuito que serviría como un complemento del proyecto para expandir nuestra propuesta, este circuito cuenta con un sensor ultrasónico y un servo motor al que le pegamos un palo de madera, su propósito es que usando el robot Pololu y una programación pequeña con la cual avanza hacia adelante y se detiene simulamos una plumilla de acceso, el sensor detecta al Pololu y el servo hace que el palo se levante por 5 segundos antes de descender de nuevo.

Lo que más disfrute de este entregable final es la libertad que tuvimos para elegir que realizar, el único límite fue nuestra imaginación y creatividad, lo único que me hubiera gustado que fuera distinto fue el tiempo que tuvimos para trabajarlo pero fuera de eso fue muy genial darle poco a poco forma a este entregable que aún cuenta con un gran potencial de expansión; sin duda alguna disfrute mucho este bloque y aprendí bastantes cosas interesantes que me serán de gran utilidad para el futuro.

Helena: Haber desarrollado este proyecto me dio la oportunidad de poder integrar todos los conocimientos que obtuve en esta materia a lo largo de todo el semestre. Desde linux, el uso de distintos frameworks y herramientas, hasta la programación de microcontroladores y arduino. En este caso logramos crear un sistema de control de acceso mediante el uso de distintos componentes tales como la Raspberry Pi 4 y el Arduino UNO. Considero que el desarrollo de este reto fue bastante interesante y divertido de hacer, ya que nos mostró un ejemplo útil en la vida cotidiana de cómo se pueden utilizar las herramientas y programas que nos enseñaron a lo largo del bloque. Asimismo, fue retadora la organización, ya que habían varias entregas dentro del plazo en el que se desarrolló este sistema, para ello fue crucial la comunicación y el dividirnos el trabajo según nuestras fortalezas, de tal forma que todos en equipo pudiéramos conseguir terminar con una buena calidad este proyecto.

Por otro lado, si bien no se programó directamente el microcontrolador ATMEGA328P, me quedó con todos los conocimientos y programas realizados en esta clase para futuras aplicaciones y proyectos que vayamos realizando a lo largo de nuestra carrera, pues el conocer y saber cómo funciona este microcontrolador nos permite realizar múltiples funciones de distintas complejidades que se pueden adecuar al proyecto, reto o tarea que se nos esté solicitando resolver.

Alexa: Realizar este proyecto me permitió aplicar los conocimientos del módulo de linux embebido, del robot 3pi y en menor medida del módulo de microcontroladores. Me agradó integrar todos los aprendizajes adquiridos hasta ahora en el proyecto final, ya que me ayudó demasiado a reforzarlos y mejorarlos. Asimismo tuve la oportunidad de trabajar con mis compañeros de equipo y conocerlos mucho mejor, lo cual también me pareció muy importante para fortalecer mis habilidades de trabajo en equipo.

La parte en la que estuve más involucrada fue en el desarrollo del software a través de python, Tkinter, Fast API y linux embebido. Esto me resultó muy enriquecedor para la aplicación de los protocolos de comunicación mediante socket y las formas en las que se puede lograr que varios dispositivos puedan acceder a cierta información.

Cabe mencionar que realizar un proyecto como este fue algo difícil en este bloque, pues hubo demasiadas entregas a lo largo del semestre, pero a pesar de todo siento que fue una unidad de formación muy útil que define gran parte de la carrera y que me ayudó a darme cuenta que me gusta mucho trabajar en la parte de microcontroladores y en general todas las áreas que

tengan mayor contacto con software. Esto fue muy importante para mi porque al inicio del semestre aún me encontraba insegura acerca de seguir cursando esta carrera.

Link del video del circuito funcionando

- SistemaControlAcceso