**Desarrollo**

**Descripción general del sistema.**

Esta sección está enfocada en proporcionar una visión detallada sobre el desarrollo del proyecto. El sistema consta de 5 módulos que trabajan en conjunto para su buen funcionamiento. desde la creación del circuito capaz de conectarse con ayuda del Internet de las Cosas hasta las herramientas de software y hardware necesarias para su implementación y monitoreo en una aplicación web.

**Módulo recolector y transmisor de datos.**

El objetivo principal de este módulo consiste en la construcción del circuito que fungirá como el prototipo del sistema embebido. Este módulo se divide en dos partes, la primera parte presenta la forma en que se construyó un circuito con los diferentes sensores que componen el sistema y la segunda parte consiste en mostrar la codificación de un programa capaz de controlar todos sus componentes de manera simultánea.

**Primera Parte - Construcción del Circuito**

Para conectar diferentes elementos físicos (sensores, módulos, actuadores, etc.) y poder transferir datos a través de redes (por lo general, inalámbricas), se necesita una conexión a internet.

La conectividad inalámbrica a IoT mediante Wi-Fi o Bluetooth se ha hecho relativamente simple gracias a diferentes módulos y kits que existen en el mercado, por ejemplo, la ESP32, un módulo que presenta una solución de Wi-Fi/Bluetooth de forma eficiente y económica.

Como se ha mencionado anteriormente, para este proyecto se utiliza la ESP32 DEVKIT V1 de 30 Pines (ver figura #) que es una placa de desarrollo que permite controlar todo tipo de sensores, módulos y actuadores mediante WIFI y BLUETOOTH, además de ser económica, es eficiente por su bajo consumo le permite poder funcionar de manera ininterrumpida por lo que lo convierte en una excelente alternativa para proyectos que involucran IoT

.



Figura 0. ESP32 DEVKIT V1 de 30 Pines.

**Display LCD OLED Blanco 128×64 0.96**

El primer periférico que se conecta a la ESP32 es un **Display LCD OLED Blanco 128×64 0.96** (ver figura #) este es dispositivo electrónico tipo LED, que permite controlar cada píxel individualmente y mostrar tanto texto como gráficos. Además, por ser de tipo OLED no necesita de retroiluminación (Backlight) como los LCD, lo que hace que su consumo de energía sea mucho menor y aumenta su contraste.



Figura 0. Display Lcd OLED Blanco 128×64 0.96.

Esta Display nos sirve para indicar el estado en el que se encuentra el circuito, así como mensajes de lecturas satisfactorias que nos indicaran el resto de los componentes. Para hacer funcionar el Display OLED es necesario utilizar un microcontrolador (ESP32), su interfaz de comunicación es de tipo I2C y puede trabajar desde 3V a 5V gracias a su regulador de voltaje, por lo que la forma adecuada de conectarlo es la siguiente (ver figura #).

Imagen de la pantalla de un video juego

Descripción generada automáticamente con confianza media

Figura 0. Conexión ESP32 con Display Lcd OLED Blanco 128×64 0.96.

**Métodos de autenticación**

El sistema cuenta con 3 métodos de autenticación, Pad numérico y Huellas dactilares dirigido a profesores y personal de la institución, Tarjetas RFID dirigidas a los alumnos de la institución, es por ello por lo que necesitamos componentes que nos permitan realizar estas funcionalidades.

**Teclado Matricial 4×4 Numérico**

El segundo periférico que se conecta a la ESP32 es un **Teclado Matricial 4×4 Numérico** (ver figura #) este es dispositivo es un simple arreglo de botones conectados en filas y columnas, de modo que se pueden leer varios botones con el mínimo número de pines requeridos. Puede ser conectado a cualquier microcontrolador, sus aplicaciones van desde sistemas de seguridad, selección de menús o ingreso de datos en un sistema.

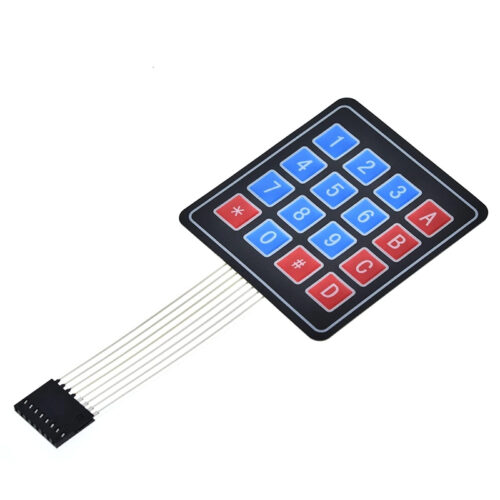


Figura 0. Teclado Matricial 4×4 Numérico.

Este teclado nos sirve para satisfacer el primer método de autenticación (Pad numérico), para los profesores y personal de la institución, además también sirve para que el circuito entre en un modo especial de detector RFID, modo registro de huellas dactilares y modo eliminación de huellas dactilares que serán explicados más adelante. La forma adecuada de conectarlo es la siguiente (ver figura #).

Imagen que contiene electrónica

Descripción generada automáticamente

Figura 0. Conexión ESP32 con Teclado Matricial 4×4 Numérico.

**Lector Sensor Huella Dactilar Digital As608**

Se consideraron varias opciones, pero muchas de las alternativas tienen elevados costos, por ello, las alternativas creadas por la empresa Adafruit Industries presentan una solución económica y asequible para muchos proyectos como el de nuestro sistema, sin embargo, se debe considerar que al implementar el sistema con este sensor la cantidad de huellas digitales que se pueden almacenar son limitadas.

El primer sensor que se conecta a la ESP32 es un Sensor de Huella As608 (ver figura #) este sensor biométrico, permite realizar sistemas de procesamiento digital de imágenes de manera interna con un DSP (Procesador de Señales Digitales). Además, incluye capacidades de comparación en base de datos y actualización de esta.

Una caja de cartón

Descripción generada automáticamente con confianza baja

Figura 0. Lector Sensor Huella Dactilar Digital As608.

Este sensor nos sirve para satisfacer el segundo método de autenticación (Huellas dactilares), para los profesores y personal de la institución, El dispositivo tiene la capacidad de almacenar hasta 162 huellas dactilares en su memoria FLASH interna. El sensor funciona con el protocolo serial, La forma adecuada de conectarlo es la siguiente (ver figura #).

Interfaz de usuario gráfica

Descripción generada automáticamente

Figura 0. Conexión ESP32 con Lector Sensor Huella Dactilar Digital As608.

**Modulo Lector RFID RC522**

El RFID RC522 (Identificador por radiofrecuencia) es un conjunto de tecnologías diseñadas para leer etiquetas (tags) a distancia de forma inalámbrica. Las etiquetas RFID están disponibles en una gran variedad de formatos, tales como pegatinas adheribles, tarjetas y llaveros.

Los RFID RC522 son ampliamente empleados, por ejemplo, en sistemas de alarma, cerraduras electrónicas, sistemas de pago, tarjetas personales, control de accesos recintos como gimnasios o piscinas, entre otras muchas aplicaciones.

El primer módulo que se conecta a la ESP32 es un Módulo Lector RFID RC522 (ver figura #).

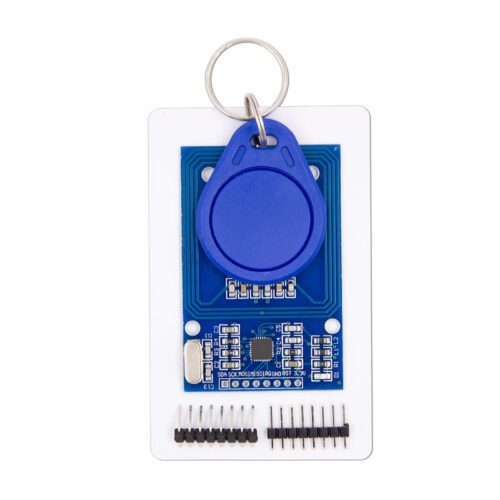


Figura 0. Modulo Lector RFID RC522.

Este módulo nos sirve para satisfacer el método de autenticación (Tarjetas RFID), para los alumnos de la institución. El módulo lector RFID-RC522 RF utiliza 3.3V como voltaje de alimentación y se controla a través del protocolo SPI. La forma adecuada de conectarlo es la siguiente (ver figura #).

Imagen que contiene electrónica, circuito

Descripción generada automáticamente

Figura 0. Conexión ESP32 con Modulo Lector RFID RC522.

**Mecanismo de apertura**

El sistema cuenta con un indicador de apertura/cierre de la puerta además de una cerradura magnética para el cierre de la puerta, es por ello por lo que necesitamos componentes que nos permitan realizar estas funcionalidades.

**Sensor Magnético MC-38**

El mecanismo del sensor MC-38 es normalmente cerrado (NC), por lo cual, manda un 1 lógico cuando ambas partes del sensor están en contacto y 0 cuando están separadas, El segundo sensor que se conecta a la ESP32 es Sensor Magnético MC-38 (ver figura #)



Figura 0. Sensor Magnético MC-38.

Este sensor nos sirve para saber el estado de la puerta (Abierta o Cerrada) cada que existan un cambio. La forma adecuada de conectarlo es la siguiente (ver figura #).

Imagen de la pantalla de un computador

Descripción generada automáticamente con confianza baja

Figura 0. Conexión ESP32 con Sensor Magnético MC-38.

**Cerradura Eléctrica 12v**

La Cerradura Eléctrica Solenoide (ver figura #) es un electroimán que consta de una bobina de alambre de cobre con una armadura (un lingote de metal) en el medio. Cuando se energiza la bobina, el centro de metal se mueve en la bobina. Esto hace que el solenoide sea capaz de tirar de un extremo. Aunque esta cerradura no es la adecuada para una puerta su funcionamiento y bajo costo nos servirá para la simulación del prototipo.



Figura 0. Cerradura Eléctrica 12v.

Para su funcionamiento es necesario un voltaje de 12 V, La ESP32 no puede suministrar la energía necesaria para la cerradura, es por ellos que se usará una fuente externa, además de un módulo relevador de 5V (ver figura #) que nos permitirá controlar dispositivos que trabajan máximo a 250 V/ 15 A por medio de una señal de control de 5V.



Figura 0. Módulo Relevador 5V.

Este módulo relevador 5V funciona como un interruptor que tan solo hay que aplicar un voltaje alto o bajo en la bobina para abrir o cerrar mecánicamente el electroimán para controlar circuitos eléctricos de manera independientes. La forma adecuada de conectarlo es la siguiente (ver figura #).

Diagrama

Descripción generada automáticamente

Figura 0. Conexión ESP32 con Cerradura Eléctrica 12V y Modulo Relevador 5V.

**Medición de la temperatura**

El sistema tiene la capacidad de encender o apagar la medición de la temperatura de forma opcional, para tomar la lectura corporal de los usuarios en los recintos que sean necesarios, esto sirve para tener un mejor control de aforos en tiempos pandémicos, se sugiere activarla conforme el semáforo epidemiológico, restringiendo el acceso a las personas con una temperatura corporal superior al promedio, es decir, superior a 37.5 °C. es por ello por lo que necesitamos componentes que nos permitan realizar estas funcionalidades.

**Sensor de Temperatura Infrarrojo MLX90614**

El sensor de temperatura infrarrojo MLX90614, permite medir temperatura de un objeto a distancia (sin contacto), el cual está diseñado para ser sensible a la radiación infrarroja emitida por un objeto a distancia, puede obtener lecturas desde los -70°C hasta 380°C, con una precisión de 0.5°C. La salida del sensor es lineal y se compensa de acuerdo con las variaciones de la temperatura ambiente. El tercer sensor que se conecta a la ESP32 es un sensor de Temperatura Infrarrojo MLX90614 (ver figura #).



Figura 0. Sensor de Temperatura Infrarrojo GY-906 MLX90614.

Este sensor nos servirá para tomar la lectura corporal de los usuarios. Su interfaz de comunicación es de tipo I2C y puede trabajar desde 3V a 5V gracias a su regulador de voltaje, por lo que la forma adecuada de conectarlo es la siguiente (ver figura #).

Imagen que contiene electrónica, circuito

Descripción generada automáticamente

Figura 0. Conexión ESP32 con Sensor de Temperatura Infrarrojo GY-906 MLX90614.

**Sensor Ultrasónico HC-SR04**

El sensor ultrasónico HC-SR04 sirve para medir distancias por medio de sus dos transductores: un micrófono y altavoz. Genera pulsos de alta frecuencia (no perceptible por el ser humano) que rebota en los objetos cercanos y es reflejado hacia el sensor, que es captado por un micrófono. El cuarto que se conecta a la ESP32 es un Sensor Ultrasónico HC-SR04 (ver figura #)



Figura 0. Sensor Ultrasónico HC-SR04.

Usualmente son usados en robótica para la detección de obstáculos, para determinar las posiciones, etc. Este sensor nos sirve para detectar la presencia de un usuario frente al Sensor de Temperatura Infrarrojo, que nos ayudara a tomar una mejor lectura de temperatura corporal. La

forma adecuada de conectarlo es la siguiente (ver figura #).

Imagen que contiene electrónica, circuito

Descripción generada automáticamente

Figura 0. Conexión ESP32 con Sensor Ultrasónico HC-SR04.

**Circuito Propuesto (Prototipo).**

El ensamblaje del circuito en conjunto con los sensores, módulos y componentes mencionados con anterioridad (ver figura #), otorga la posibilidad de que cada uno de los sensores que conformar al circuito funcionen de forma independiente uno del otro.

El circuito permite a los usuarios identificarse con alguno de los métodos de autenticación (dependiendo del usuario), posteriormente se realiza la lectura de temperatura corporal (en caso de estar habilitada), permitiendo al sistema tomar la decisión de la apertura de un laboratorio o aula equipada considerando su aforo.

Diagrama, Esquemático

Descripción generada automáticamente

Figura 0. Circuito propuesto, prototipo.

El prototipo ensamblado se puedo observar en la siguiente fotografía (ver figura # y #).

Un grupo de folletos sobre una mesa de madera

Descripción generada automáticamente con confianza baja

Figura 0. Vista lateral del Circuito propuesto, prototipo.

Imagen que contiene circuito, medidor

Descripción generada automáticamente

Figura 0. Vista superior del Circuito propuesto, prototipo.

**Segunda Parte - Programa** **recolector y transmisor de datos**

En el siguiente diagrama de flujo, se muestra el algoritmo utilizado para la programación del sistema embebido del circuito propuesto.

Diagrama

Descripción generada automáticamente

Figura 0. Diagrama de flujo del Módulo recolector y transmisor de datos.

En esta sección de código (ver tabla #) se especifican las bibliotecas requeridas para el funcionamiento del prototipo, también se establece la configuración de los GPIOS y se inicializan los objetos necesarios utilizados por los diferentes sensores, módulos y componentes que utiliza el circuito

|  |
| --- |
| // Giovanni Olmos Salmones - 2172002785  // Gabriel Hurtado Aviles - 2172000781  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Codigo principal "Sistema de control de aforos en instituciones educativas basada en el internet de las cosas"  Objetivo General :  Desarrollar un sistema modular capaz de gestionar y almacenar la información de  múltiples usuarios a través de Internet de las Cosas como un prototipo para el control de  aforos en instituciones educativas.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  #include <WiFi.h> // Conexion Wifi.  #include <PubSubClient.h> // Conexion MQTT.  #include <ArduinoJson.h> // Manipulacion de JSON.  const char\* ssid = "GosCoop23"; // SSID de la red a conectar la cerradura.  const char\* password = "Olmedo21720027"; // Password de la red a conectar la cerradura.  //const char\* mqtt\_server = "189.211.115.183"; // IP donde esta instalado el servidor MQTT.  const char\* mqtt\_server = "192.168.100.203"; // Servidor MQTT Personal;  const String Lab\_Aula = "Laboratorio de Sistemas Digitales"; // Laboratorio o aula equipada donde se encuentra la cerradura.  WiFiClient espClient;  PubSubClient client(espClient);  const int ledPin = 15; // LED Pin (led para la señal de apertura de la chapa).  #include <Keypad.h> // Libreria teclado.  #include <Adafruit\_GFX.h> // Liberia pantalla .  #include <Adafruit\_SSD1306.h> // Pantalla OLED 128x64.  #include <SPI.h> //Sensor RFID-RC522.  #include <MFRC522.h>  #include <HardwareSerial.h> // Sensor de huellas dactilares - Para Registrar.  #include <SoftwareSerial.h> //  uint8\_t id; // Variable para registrar huellas.  #include <Adafruit\_Fingerprint.h> // Sensor de huellas dactilares.  #define mySerial Serial2 // Puertdo TX0 y RX0 donde esta conectado.  Adafruit\_Fingerprint finger = Adafruit\_Fingerprint(&mySerial); // Creamos el objeto del sensor de huellas.  #include <Wire.h> //Sensor de temperatura.  #include <Adafruit\_MLX90614.h>  const int trigPin = 5; // Ultrasonico DHT11.  const int echoPin = 35; // pines uilizados por el sensor ultrasonico.  // Definimos la velocidad del sonido en cm / uS para el sensor ultrasonico.  #define SOUND\_SPEED 0.034  #define CM\_TO\_INCH 0.393701  long duration;  int distanceCm;  int distanceInch;  //Variables Extra  int Espera1 = 500; //Espera en el loop.  int Dist; //Distancia del ultrasonico.  int DistMin = 12; //Distancia minima para detectar al sujeto (mm).  int Presente = 0; //Si hay alguien frente al Termometro.  int Espera = 3000; //Tiempo de espera para verificar sujeto.  unsigned long Tiempo = 0; //Tiempo que lleva detectado para Millis.  int Ahora = 0; //Millis en el momento que se inicia.  //Temperatura  float TempObj; //Temperatura del sujeto (Usuario).  float TempMax = 37.50; //Temperatura maxima permitida.  //Inicializamos sensor de temperatura.  Adafruit\_MLX90614 mlx = Adafruit\_MLX90614();  //Definimos la pantalla OLED 128x64.  #define SCREEN\_WIDTH 128 // OLED display ancho, en pixeles.  #define SCREEN\_HEIGHT 64 // OLED display alto, en pixelss.  Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, -1); // Creamos un objeto a la pantalla oled  //Definmos el sensor RFID  #define RST\_PIN 2 // Pin RST.  #define SS\_PIN 4 // Pin SDA.  MFRC522 mfrc522(SS\_PIN, RST\_PIN); // Creamos un objeto al sensor MFRC522.  //Teclado  const byte filas = 4; //Numero de filas del teclado.  const byte columnas = 4; //Numero de columnas del teclado.  //Defino una matriz 4x4 con la posicion de las filas y columnas.  char matriz[filas][columnas] = {  { '1', '2', '3', 'A'},  { '4', '5', '6', 'B'},  { '7', '8', '9', 'C'},  { '\*', '0', '#', 'D'},  };  byte pinesFilas[filas] = {13, 12, 14, 27}; //Pines donde van conectadas las filas del teclado.  byte pinesColumnas[columnas] = {26, 25, 33, 32}; //Pines donde van conectadas las columnas del teclado.  //Inicializo el teclado con el numero de filas, columnas, los pines del Arduino utilizados y la matriz.  Keypad teclado = Keypad( makeKeymap(matriz), pinesFilas, pinesColumnas, filas, columnas);  char num1[10] = ""; // Aqui se guardara la contrasena tecleada.  int y = 0; // Iterador de la cadena tecleada.  // Sensor de apertura de la puerta  const int pinDoor = 34; // pin de entrada de la señal del electroiman.  int EstadoV = 0; // Estados del sensor.  int EstadoN = 0;  String doorState ; // Estado de la puerta.  const int boton = 1; // Botón asignado en el pin 1.  int anterior; // Guardamos el estado anterior.  int estado; // Estado del botón.  unsigned long temporizador; // Temporizador del boton.  unsigned long tiemporebote = 50; //  // Logotipo  static const uint8\_t image[648] = {  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x01, 0xe0, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x47, 0xc8, 0x80, 0x00, 0x00,  0x00, 0x00, 0x00, 0x18, 0x72, 0x60, 0x00, 0x00,  0x00, 0x00, 0x0f, 0xff, 0x5d, 0x90, 0x00, 0x00,  0x00, 0x00, 0x00, 0x18, 0xd4, 0x48, 0x00, 0x00,  0x00, 0x00, 0x0f, 0x38, 0xb1, 0xa4, 0x00, 0x00,  0x00, 0x00, 0x30, 0x71, 0xaa, 0xd2, 0x00, 0x00,  0x00, 0x00, 0x07, 0xf1, 0x64, 0x69, 0x00, 0x00,  0x00, 0x00, 0x10, 0x86, 0x5d, 0x34, 0x80, 0x00,  0x00, 0x00, 0x48, 0x00, 0xd6, 0x92, 0x00, 0x00,  0x00, 0x01, 0xb6, 0x0e, 0xa2, 0x59, 0x00, 0x00,  0x00, 0x00, 0xd1, 0x05, 0xab, 0x64, 0x80, 0x00,  0x00, 0x00, 0xac, 0x7c, 0x6d, 0xb3, 0x40, 0x00,  0x00, 0x01, 0x49, 0x01, 0x16, 0xd9, 0xc0, 0x00,  0x00, 0x02, 0x94, 0x38, 0x59, 0x64, 0x80, 0x00,  0x00, 0x05, 0x29, 0x83, 0x2c, 0xb2, 0x40, 0x00,  0x00, 0x01, 0x52, 0x00, 0x96, 0xd9, 0x20, 0x00,  0x00, 0x02, 0x84, 0x7c, 0x5b, 0x64, 0xa0, 0x00,  0x00, 0x04, 0xe9, 0xc3, 0x2d, 0xb3, 0x00, 0x00,  0x00, 0x09, 0xcb, 0x01, 0x92, 0x19, 0xc0, 0x00,  0x00, 0x01, 0x5c, 0x7c, 0x4b, 0x45, 0x60, 0x00,  0x00, 0x02, 0x95, 0x83, 0x25, 0x25, 0x80, 0x00,  0x00, 0x04, 0xbd, 0x38, 0xb4, 0xb2, 0xc0, 0x00,  0x00, 0x01, 0xba, 0x64, 0x52, 0x5b, 0x60, 0x00,  0x00, 0x0e, 0x56, 0x93, 0x49, 0x20, 0x80, 0x00,  0x00, 0x12, 0xf5, 0x45, 0x24, 0xb4, 0x40, 0x00,  0x00, 0x06, 0xa5, 0x12, 0x92, 0xda, 0x20, 0x00,  0x00, 0x09, 0x69, 0x85, 0x49, 0x4b, 0x00, 0x00,  0x00, 0x0a, 0x49, 0xb4, 0xa9, 0x24, 0x80, 0x00,  0x00, 0x02, 0xda, 0xaa, 0x14, 0x92, 0xc0, 0x00,  0x00, 0x04, 0x12, 0xa5, 0x52, 0x49, 0x60, 0x00,  0x00, 0x09, 0x3a, 0x90, 0xa9, 0x24, 0xc0, 0x00,  0x00, 0x02, 0x2a, 0x4a, 0x94, 0x92, 0x00, 0x00,  0x00, 0x06, 0x5a, 0x49, 0x4a, 0x49, 0x80, 0x00,  0x00, 0x0c, 0x94, 0x24, 0xa8, 0x24, 0x60, 0x00,  0x00, 0x01, 0xa4, 0x13, 0x95, 0x13, 0x00, 0x00,  0x00, 0x03, 0x28, 0x4c, 0xca, 0xcc, 0x40, 0x00,  0x00, 0x06, 0x48, 0x23, 0x35, 0xa3, 0x80, 0x00,  0x00, 0x00, 0x90, 0x98, 0x8b, 0x58, 0x00, 0x00,  0x00, 0x01, 0x23, 0x76, 0x66, 0x33, 0x00, 0x00,  0x00, 0x02, 0x48, 0x19, 0x99, 0xe0, 0x00, 0x00,  0x00, 0x00, 0x31, 0xa6, 0x67, 0x1e, 0x00, 0x00,  0x00, 0x00, 0x4e, 0x3d, 0x99, 0xe0, 0x80, 0x00,  0x00, 0x01, 0x30, 0x07, 0xe7, 0x1c, 0x00, 0x00,  0x00, 0x00, 0x47, 0xfc, 0xe1, 0xc0, 0x00, 0x00,  0x00, 0x00, 0x7c, 0x06, 0x1e, 0x00, 0x00, 0x00,  0x00, 0x01, 0x87, 0x38, 0xe3, 0x88, 0x00, 0x00,  0x00, 0x00, 0x3e, 0x7e, 0x6c, 0x0c, 0x00, 0x00,  0x00, 0x00, 0x02, 0x00, 0x78, 0xd0, 0x00, 0x00,  0x00, 0x00, 0x1e, 0x3f, 0x19, 0xf8, 0x00, 0x00,  0x00, 0x00, 0x30, 0x18, 0xff, 0x80, 0x00, 0x00,  0x00, 0x00, 0x0e, 0x3e, 0x00, 0x10, 0x00, 0x00,  0x00, 0x00, 0x1f, 0x03, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x03, 0xc3, 0xe0, 0x00, 0x00,  0x00, 0x00, 0x03, 0xf9, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x01, 0xff, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x3c, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00  }; |

Tabla 0. Primera parte del Código.

La función Fun\_Ver\_RFID() permite al circuito entrar en modo de detección o identificación por radiofrecuencia para obtener la identidad de un objeto gracias a etiquetas que pueden ser integradas en tarjetas, llaveros, etc.

|  |
| --- |
| void Fun\_Ver\_RFID(){  display.clearDisplay();  display.setTextColor(WHITE);  display.setCursor(0,0);  display.print("Detector");  display.setCursor(0,20);  display.print("RFID");  display.display();    while(true){  char key = teclado.getKey(); // Se lee el caracter tecleado por el usuario.  if(key){  if(key == '#'){  display.clearDisplay();  display.setTextSize(2);  display.setTextColor(WHITE);  display.setCursor(0,0);  display.print("Saliendo");  display.setCursor(0,20);  display.print("del modo");  display.setCursor(0,40);  display.print("detector");  display.display();  delay(2000);  return ;  }  }  if(mfrc522.PICC\_IsNewCardPresent()) {  unsigned long uid = getID();  if(uid != -1){  display.clearDisplay();  display.setTextSize(1);  display.setTextColor(WHITE);  display.setCursor(0, 0);  display.print("Tarjeta detectada : ");  display.setTextSize(2);  display.setCursor(0, 30);  display.print(uid);  display.display();  }  }  }  } |

Tabla 0. Fun\_Ver\_RFID() del código

La función Fun\_Huella\_Registrar() despliega un menú que permite capturar una nueva huella dactilar, una vez escrito el numero de huella a capturar (1 al 161), tenemos la opción de registrar la huella, de corregir el numero de huella o de salir de este menú.

|  |
| --- |
| void Fun\_Huella\_Registrar() {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.print("Registro");  display.setCursor(0,20);  display.print("de Huellas");  display.setCursor(0,40);  display.print("dactilares");  display.display();  delay(1000);    char numHuella[3] = "" ;  int x = 0 ;  while(true){  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(1);  display.setCursor(0,0);  display.print("Escriba el # huella");  display.setCursor(0,10);  display.print("(del 1 al 127)");  display.setCursor(0,20);  display.print("Oprime A - Ir a registrar");  display.setCursor(0,30);  display.print("Oprime B - limpiar #");  display.setCursor(0,40);  display.print("Oprime # - Salir");  display.display();    char key = teclado.getKey(); // Se lee el caracter tecleado por el usuario.  if(key){  if(key == '#'){  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.print("Saliendo");  display.setCursor(0,20);  display.print("del modo");  display.setCursor(0,40);  display.print("registro");  display.display();  delay(2000);  return ;  } else if (key == 'A') {  String valueS = String(numHuella);  int value = valueS.toInt(); // Convertimos la cadena de string a int  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.print("# Huella a");  display.setCursor(0,20);  display.print("Registar :");  display.setCursor(0,40);  display.print(value);  display.display();  delay(2000);  if( value >= 1 && value <= 127){  id = value;  Fun\_Enroll();  x = 0; // Reinicimaos el iterador.  numHuella[0] = ' ' ;  numHuella[1] = ' ' ;  numHuella[2] = ' ' ;  } else {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.print("Numero");  display.setCursor(0,20);  display.print("incorrecto...");  display.setCursor(0,40);  display.print("... :C");  display.display();  delay(1000);  x = 0; // Reinicimaos el iterador.  numHuella[0] = ' ' ;  numHuella[1] = ' ' ;  numHuella[2] = ' ' ;  }  } else if (key == 'B') {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.print("limpiando");  display.setCursor(0,20);  display.print("...");  display.display();  delay(1000);  x = 0; // Reinicimaos el iterador.  numHuella[0] = ' ' ;  numHuella[1] = ' ' ;  numHuella[2] = ' ' ;  } else {  numHuella[x] = key;  x++;  }  }  }  }  void Fun\_Enroll() {  while(true){  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(1);  display.setCursor(0,0);  display.print("Huella a registrar : ");display.print(id);  display.setCursor(0,20);  display.print("Oprime A - Registrar ");  display.setCursor(0,30);  display.print("Oprime B - Cambiar #");  display.setCursor(0,40);  display.print("Oprime # - Salir ");  display.display();    char key = teclado.getKey(); // Se lee el caracter tecleado por el usuario.  if(key){  if(key == '#'){  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.print("Saliendo");  display.setCursor(0,20);  display.print("del modo");  display.setCursor(0,40);  display.print("registro");  display.display();  delay(2000);  return ;  } else if (key == 'A'){  getFingerprintEnroll() ;  } else if (key == 'B') {  Fun\_Huella\_Registrar() ;  }  }  }  } |

Tabla 0. Fun\_Huella\_Registrar() del código.

La función getFingerprintEnroll() prepara al circuito para entrar en un modo que permite guardar la información biométrica al posicionar una huella dactilar sobre el sensor para el reconocimiento de personal. El dato biométrico en forma de huella es capturado, almacenado, comparado e interpretado en función de la capacidad disponible en el sensor.

|  |
| --- |
| uint8\_t getFingerprintEnroll() {  int p = -1;  while (p != FINGERPRINT\_OK) {  p = finger.getImage();  switch (p) {  case FINGERPRINT\_OK:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Imagen tomada");  display.display();  break;  case FINGERPRINT\_NOFINGER:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("...");  display.display();  break;  case FINGERPRINT\_PACKETRECIEVEERR:  Serial.println("Communication error");  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Error de comunicacion");  display.display();  break;  case FINGERPRINT\_IMAGEFAIL:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Error de imagen");  display.display();  break;  default:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Error desconocido");  display.display();  break;  }  }    // OK success!    p = finger.image2Tz(1);  switch (p) {  case FINGERPRINT\_OK:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Imagen convertida");  display.display();  break;  case FINGERPRINT\_IMAGEMESS:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Imagen demasiado desordenada");  display.display();  return p;  case FINGERPRINT\_PACKETRECIEVEERR:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Error de comunicacion");  display.display();  return p;  case FINGERPRINT\_FEATUREFAIL:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("No se pudieron encontrar las características");  display.display();  delay(1000);  case FINGERPRINT\_INVALIDIMAGE:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("No se pudieron encontrar las características");  display.display();  return p;  default:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Error desconocido");  display.display();  return p;  }    display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Remueve el dedo");  display.display();  delay(2000);  p = 0;  while (p != FINGERPRINT\_NOFINGER) {  p = finger.getImage();  }  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("# de Huella :" + id);  display.display();  p = -1;  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Vuelva a colocar el mismo dedo");  display.display();  delay(1000);  while (p != FINGERPRINT\_OK) {  p = finger.getImage();  switch (p) {  case FINGERPRINT\_OK:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Imagen tomada");  display.display();  break;  case FINGERPRINT\_NOFINGER:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("...");  display.display();  break;  case FINGERPRINT\_PACKETRECIEVEERR:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Imagen tomada");  display.display();  break;  case FINGERPRINT\_IMAGEFAIL:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Error de imagen");  display.display();  break;  default:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Error desconocido");  display.display();  break;  }  }    // OK success!    p = finger.image2Tz(2);  switch (p) {  case FINGERPRINT\_OK:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Imagen convertidad");  display.display();  break;  case FINGERPRINT\_IMAGEMESS:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Imagen demasiado desordenado");  display.display();  return p;  case FINGERPRINT\_PACKETRECIEVEERR:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Error de comunicacion");  display.display();  return p;  case FINGERPRINT\_FEATUREFAIL:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("No se pudieron encontrar las características");  display.display();  return p;  case FINGERPRINT\_INVALIDIMAGE:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("No se pudieron encontrar las características");  display.display();  return p;  default:  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Error desconocido");  display.display();  return p;  }    // OK converted!  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Creando modelo para # " + id);  display.display();      p = finger.createModel();  if (p == FINGERPRINT\_OK) {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Huellas combinadas");  display.display();    } else if (p == FINGERPRINT\_PACKETRECIEVEERR) {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Error de comunicacion");  display.display();    return p;  } else if (p == FINGERPRINT\_ENROLLMISMATCH) {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Las huellas dactilares no coinciden");  display.display();  delay(1000);    return p;  } else {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Error desconocido");  display.display();    return p;  }    display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("# ID : " + id);  display.display();    p = finger.storeModel(id);  if (p == FINGERPRINT\_OK) {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Almacenda!");  display.display();  delay(3000);    return p;  } else if (p == FINGERPRINT\_PACKETRECIEVEERR) {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Error de comunicacion");  display.display();    return p;  } else if (p == FINGERPRINT\_BADLOCATION) {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("No se pudo almacenar en esa ubicación");  display.display();    return p;  } else if (p == FINGERPRINT\_FLASHERR) {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Error al escribir en flash");  display.display();    return p;  } else {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Error desconocido");  display.display();  return p;  }  } |

Tabla 0. getFingerprintEnroll() del código.

La función Fun\_Huella\_Borrar() despliega un menú que permite borrar una nueva huella dactilar, una vez escrito el numero de huella a borrar (1 al 161), tenemos la opción de borrar la huella , de corregir el numero de huella o de salir de este menú.

|  |
| --- |
| void Fun\_Huella\_Borrar() {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.print("Borrar");  display.setCursor(0,20);  display.print("Huellas");  display.setCursor(0,40);  display.print("dactilares");  display.display();  delay(1000);    char numHuella[3] = "" ;  int x = 0 ;  while(true){  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(1);  display.setCursor(0,0);  display.print("Escriba el # huella");  display.setCursor(0,10);  display.print("(del 1 al 127)");  display.setCursor(0,20);  display.print("Oprime A - Borrar");  display.setCursor(0,30);  display.print("Oprime B - limpiar #");  display.setCursor(0,40);  display.print("Oprime # - Salir");  display.display();    char key = teclado.getKey(); // Se lee el caracter tecleado por el usuario.  if(key){  if(key == '#'){  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.print("Saliendo");  display.setCursor(0,20);  display.print("del modo");  display.setCursor(0,40);  display.print("borrar");  display.display();  delay(2000);  return ;  } else if (key == 'A') {  String valueS = String(numHuella);  int value = valueS.toInt(); // Convertimos la cadena de string a int  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.print("# Huella a");  display.setCursor(0,20);  display.print("Borrar :");  display.setCursor(0,40);  display.print(value);  display.display();  delay(2000);  if( value >= 1 && value <= 127){  id = value;  deleteFingerprint(id);  x = 0; // Reinicimaos el iterador.  numHuella[0] = ' ' ;  numHuella[1] = ' ' ;  numHuella[2] = ' ' ;  } else {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.print("Numero");  display.setCursor(0,20);  display.print("incorrecto...");  display.setCursor(0,40);  display.print("... :C");  display.display();  delay(1000);  x = 0; // Reinicimaos el iterador.  numHuella[0] = ' ' ;  numHuella[1] = ' ' ;  numHuella[2] = ' ' ;  }  } else if (key == 'B') {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.print("limpiando");  display.setCursor(0,20);  display.print("...");  display.display();  delay(1000);  x = 0; // Reinicimaos el iterador.  numHuella[0] = ' ' ;  numHuella[1] = ' ' ;  numHuella[2] = ' ' ;  } else {  numHuella[x] = key;  x++;  }  }  }  } |

Tabla 0. Fun\_Huella\_Borrar() del código.

La función deleteFingerprint() prepara al circuito para entrar en un modo que permite borrar la información biométrica al escribir un numero de huella dactilar. El dato biométrico en forma de huella es borrado, en función de la capacidad disponible en el sensor.

|  |
| --- |
| uint8\_t deleteFingerprint (uint8\_t id) {  uint8\_t p = -1;  p = finger.deleteModel(id);  if (p == FINGERPRINT\_OK) {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.print("Borrada!");  display.display();  delay(2000);  } else if (p == FINGERPRINT\_PACKETRECIEVEERR) {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.print("Error de comunicacion");  display.display();  delay(1000);  } else if (p == FINGERPRINT\_BADLOCATION) {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.print("No se");  display.setCursor(0,20);  display.print("puede");  display.setCursor(0,40);  display.print("borra");  display.display();  delay(1000);  } else if (p == FINGERPRINT\_FLASHERR) {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,0);  display.print("Error almemoria flash");  display.display();  delay(1000);  }  return p;  } |

Tabla 0. deleteFingerprint(uint8\_t id) del código.

La función deleteFingerprint() prepara al circuito para que reciba una lectura de una huella dactilar El dato biométrico en forma de huella es capturada, para posteriormente enviado a través de un topic MQTT hacia un servidor Node-Red el cual se encargar de hacer la comparación con la Base de Datos donde se almacena el número de huella que le corresponde a cada usuario.

|  |
| --- |
| uint8\_t getFingerprintID() {  uint8\_t p = finger.getImage();  switch (p) {  case FINGERPRINT\_OK:  //Serial.println("Image taken");  break;  case FINGERPRINT\_NOFINGER:  //Serial.println("No finger detected");  return p;  case FINGERPRINT\_PACKETRECIEVEERR:  //Serial.println("Communication error");  return p;  case FINGERPRINT\_IMAGEFAIL:  //Serial.println("Imaging error");  return p;  default:  //Serial.println("Unknown error");  return p;  }  // OK success!  // Convertimos la imagen de la huella presentada codigo para compararla.  p = finger.image2Tz();  switch (p) {  case FINGERPRINT\_OK:  //Serial.println("Image converted");  break;  case FINGERPRINT\_IMAGEMESS:  //Serial.println("Image too messy");  return p;  case FINGERPRINT\_PACKETRECIEVEERR:  //Serial.println("Communication error");  return p;  case FINGERPRINT\_FEATUREFAIL:  //Serial.println("Could not find fingerprint features");  return p;  case FINGERPRINT\_INVALIDIMAGE:  //Serial.println("Could not find fingerprint features");  return p;  default:  // Serial.println("Unknown error");  return p;  }  // OK converted!  p = finger.fingerSearch();  if (p == FINGERPRINT\_OK) {  //Serial.println("Found a print match!");  } else if (p == FINGERPRINT\_PACKETRECIEVEERR) {  //Serial.println("Communication error");  return p;  } else if (p == FINGERPRINT\_NOTFOUND) {  //Serial.println("Did not find a match");    //Si la huella presentada no conincide con ninguna huella.  display.clearDisplay();  display.setCursor(0,0);  display.setTextColor(WHITE);  display.print("Huella no registrada");  display.display();  delay(3000);  return p;    } else {  //Serial.println("Unknown error");  return p;  }    //Si la huella presentada conincide con alguna huella resgitrada.  char dactilarString[12];  dtostrf(finger.fingerID, 1, 0, dactilarString); //Almacenamos el id de la huella detectada    StaticJsonDocument<256> doc; //Creamos un documento JSON, donde enviaremos la informacion del id de la huella para comprobar si esta registrada.  doc["dactilar"] = dactilarString; //Solo mandamos el id de identifiacion de la huella.  char out[128];  int b = serializeJson(doc, out); //Sereliarizamos el JSON para poderlo mandar por mqtt.    client.publish("Gabanni/UAMazc/compararHuella/Profesor/0", out); //Publicamos en MQTT.  return finger.fingerID;  }  int getFingerprintIDez() {  uint8\_t p = finger.getImage();  if (p != FINGERPRINT\_OK) return -1;    p = finger.image2Tz();  if (p != FINGERPRINT\_OK) return -1;    p = finger.fingerFastSearch();  if (p != FINGERPRINT\_OK) return -1;    return finger.fingerID;  } |

Tabla 0. getFingerprintID()del código.

La función Fun\_RFID() prepara al circuito para que reciba una lectura del lector RFID el numero RFID en formato decimal es capturado, para posteriormente enviado a través de un topic MQTT hacia un servidor Node-Red el cual se encargar de hacer la comparación con la Base de Datos donde se almacena el número RFID que le corresponde a cada usuario.

|  |
| --- |
| void Fun\_RFID() {  unsigned long uid = getID(); // Numero de identificacion en formato decimal.  Serial.println("ESTOY LEYENDO LA TARJETA");  if(uid != -1){ // Comprobamos que la lectura de la tarjeta sea correcta  char uidString[12]; // Variable que se usara para almacenar el id de la tarjeta RFID.  dtostrf(uid, 1, 0, uidString); // Convertimos la cadena de unsigned long a string    StaticJsonDocument<256> doc; // Creamos un documento JSON, donde enviaremos la infromacion de la tarjeta RFID para comprobar si esta registrada.  doc["RFID"] = uidString; // Solo mandamos el id de identifiacion de la tarjeta.    char out[128];  int b = serializeJson(doc, out); //Sereliarizamos el JSON para poderlo mandar por mqtt.      client.publish("Gabanni/UAMazc/compararRFID/Alumno/0", out); //Publicamos en MQTT.  }  } |

Tabla 0. Fun\_RFID() del código.

La función Fun\_teclado() prepara al circuito para que reciba una lectura del Pad numerico el en formato decimal, es capturado, para posteriormente enviado a través de un topic MQTT hacia un servidor Node-Red el cual se encargar de hacer la comparación con la Base de Datos donde se almacena el número (clave) que le corresponde a cada usuario.

|  |
| --- |
| void Fun\_teclado(char x) {  num1[y] = x; // Char x es el primer caracter de la contrasena tecleada.  y++; // Iterador de la cadena tecleada.  while(true){ // Ciclo infinito mmientras el usuario no deje de teclear.  char key = teclado.getKey(); // Se lee el caracter tecleado por el usuario.  if(key){  if(key == '#'){ // Si el usuario teclea "#", quiere decir que ya ah acabado de poner su contrasena.  Serial.println(key);  String codigo = String(num1); //En num1 se guarda el codigo tecleado por el usuario, lo convertimos a string.  /////// Verificar Tarjetas - RFID////////  if(codigo == "1903152172"){  Fun\_Ver\_RFID(); // Funcion que nos ayuda a detectar el numero RFID de las tarjetas.  y = 0; // Reinicimaos el iterador.  limpiar(); // Limpiamos la cadena para el proximo usuario.  return ;  } else if (codigo == "1903152173") {  Fun\_Huella\_Registrar(); // Funcion que nos ayuda a registrar un huella dactilar.  y = 0; // Reinicimaos el iterador.  limpiar(); // Limpiamos la cadena para el proximo usuario.  return ;  } else if (codigo == "1903152174") {  Fun\_Huella\_Borrar(); // Funcion que nos ayuda a borrar un huella dactilar.  y = 0; // Reinicimaos el iterador.  limpiar(); // Limpiamos la cadena para el proximo usuario.  return ;  } else {  StaticJsonDocument<256> doc; //Creamos un documento JSON, donde enviaremos el codigo tecleado para comprobar si esta registrado.  doc["codigo"] = codigo; //Solo mandamos el id del codigo tecleado.  char out[128];  int b = serializeJson(doc, out); //Sereliarizamos el JSON para poderlo mandar por mqtt.    client.publish("Gabanni/UAMazc/compararteclado/Profesor/0", out); //Publicamos en MQTT.    y = 0; // Reinicimaos el iterador.  limpiar(); // Limpiamos la cadena para el proximo usuario.  return ;  }  /////////////////////////////////////////    } else if (y <= 9) {  num1[y] = key;  y++;  }  }  }  } |

Tabla 0. Fun\_RFID() del código

La función EstadoPuerta() recibe la señal del sensor magnético, este nos indica el estado de la puerta (Abierta/Cerrada ) para posteriormente enviado a través de un topic MQTT hacia un servidor Node-Red el cual se encargar de almacenar la información en la Base de Datos.

|  |
| --- |
| void EstadoPuerta() {  EstadoV = EstadoN ;  if(EstadoV == 0){  doorState = "abierto";  } else{  doorState = "cerrado";  }  StaticJsonDocument<256> doc; //Creamos un documento JSON, donde enviaremos la informacion del id de la huella para comprobar si esta registrada.  doc["estado"] = doorState; //Solo mandamos el eatado de la puerta.  char out[128];  int b = serializeJson(doc, out); //Sereliarizamos el JSON para poderlo mandar por mqtt.  client.publish("Gabanni/UAMazc/EstadoPuerta/0", out); //Publicamos  } |

Tabla 0. EstadoPuerta() del código.

La función getID() recibe un numero RFID en formato Hexademical, y nos lo devuelve en formato decimal.

|  |
| --- |
| unsigned long getID(){  if ( ! mfrc522.PICC\_ReadCardSerial()) {  return -1;  }  unsigned long hex\_num;  hex\_num = mfrc522.uid.uidByte[0] << 24;  hex\_num += mfrc522.uid.uidByte[1] << 16;  hex\_num += mfrc522.uid.uidByte[2] << 8;  hex\_num += mfrc522.uid.uidByte[3];  mfrc522.PICC\_HaltA();  return hex\_num;  } |

Tabla 0. getID() del código.

La función limpiar() limpia la entrada de la variables num1, esta variables es utilizada por los usuarios que teclean su clave en el pad numérico, cada vez que se acaba de utilizar se limpia para almacenar una nueva clave tecleada.

|  |
| --- |
| void limpiar() {  for (int i = 0; i <= 9; ++i){  num1[i] = ' ';  }  } |

Tabla 0. limpiar() del código.

La función setup\_wifi() es la que nos ayuda a establecer una conexión Wifi en el microcontrolador (ESP32), mientras que la función reconnect() es la encargada de restablecer la reconexión en caso de pedida, además es la encargada se suscribirse a los topics de MQTT que entra en modo de escucha gracias a la función callback, que son las encargadas de realizar múltiples funciones en el código.

|  |
| --- |
| void setup\_wifi() {  delay(10);  // Empezamos conectándonos a una red WiFi.  Serial.print("Connecting to ");  Serial.println(ssid);  //Mensaje en pantalla.  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 0);  display.print("Conectando...");  display.setTextSize(2);  display.display();    WiFi.begin(ssid, password);    while (WiFi.status() != WL\_CONNECTED) {  delay(500);  Serial.print(".");  }    Serial.println("");  Serial.println("WiFi connected");  Serial.println("IP address: ");  Serial.println(WiFi.localIP());  //Mensaje en pantalla.  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 0);  display.print("WiFi");  display.setCursor(0, 20);  display.print("Conectado!");  display.display();  }  void reconnect() {  // Bucle hasta que estemos reconectados.  while (!client.connected()) {  Serial.print("Intentando conexión MQTT...");  // Intento de conexión  if (client.connect("ESP32Client0")) {  Serial.println("Connected");  // Subscribe a topics MQTT.  client.subscribe("Gabanni/UAMazc/Ocupado/0");  client.subscribe("Gabanni/UAMazc/Bloqueo/0");  client.subscribe("Gabanni/UAMazc/Apertura/0");  client.subscribe("Gabanni/UAMazc/LecturaTemperatura/Alumno/0");  client.subscribe("Gabanni/UAMazc/LecturaTemperatura/Profesor/0");  client.subscribe("Gabanni/UAMazc/SinAcceso/0");  client.subscribe("Gabanni/UAMazc/Despedida/0");  client.subscribe("Gabanni/UAMazc/AforoLleno/0");  client.subscribe("Gabanni/UAMazc/TemperaturaAlta/0");  client.subscribe("Gabanni/UAMazc/DeshabilitadoTemperatura/Alumno/0");  client.subscribe("Gabanni/UAMazc/DeshabilitadoTemperatura/Profesor/0");    //Ubicacion de la cerradura  StaticJsonDocument<256> doc; // Creamos un documento JSON, donde enviaremos la infromacion de la tarjeta RFID para comprobar si esta registrada.  doc["ubicacion"] = Lab\_Aula; // Laboratorio o aula equipada donde se encuentra la cerradura  char out[128];  int b = serializeJson(doc, out); //Sereliarizamos el JSON para poderlo mandar por mqtt.  client.publish("Gabanni/UAMazc/Ubicacion/0", out); //Publicamos en MQTT.  } else {  Serial.print("failed, rc=");  Serial.print(client.state());  Serial.println(" try again in 5 seconds");  // Espere 5 segundos antes de volver a intentar.  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 0);  display.print("Error del");  display.setCursor(0, 20);  display.print("servidor");  display.setTextSize(1);  display.setCursor(0, 40);  display.print(" Reconectando ... ");  display.display();  delay(5000);  }  }  }  void callback(char\* topic, byte\* message, unsigned int length) {  Serial.print("Mensaje sobre el tema: "); Serial.print(topic); // Mostramamos el tema del que estamos recibiendo.  Serial.print("Message: "); // Mostramamos el mensaje que estamos recibiendo.  String messageTemp;  // En esta funcion estamos leyendo caracter por caracter del mensaje entero recibido.  for (int i = 0; i < length; i++) {  Serial.print((char)message[i]);  messageTemp += (char)message[i];  }  StaticJsonDocument <256> doc; //Creamos un documento JSON, donde almacenaremos el mensaje que recibimos en formato JSON.  deserializeJson(doc,message); //Se deserializa el el mensaje recibido.  //Posibles campos recibidos en el JSON.  String nombre = doc["nombre"];  String id = doc["id"];  String RFID = doc["RFID"];  String codigo = doc["codigo"];  // Funcion para mandar senal de apertura de la cerradura (Cuando se presenta RFID,codigo o huella dactilar).  if (String(topic) == "Gabanni/UAMazc/Apertura/0") {  if(messageTemp == "true"){ // Se desbloquea la cerradura solo por 5 segundos y posterioemente se vuelve a bloquear.  digitalWrite(ledPin, HIGH); // Se desbloquea la cerradura solo por 5 segundos y posterioemente se vuelve a bloquear.  delay(5000);  digitalWrite(ledPin, LOW);  } else if(messageTemp == "false"){ // Se desbloquea la cerradura.  digitalWrite(ledPin, LOW);  }  }  // Funcion para mandar senal de apertura de la cerradura.  if (String(topic) == "Gabanni/UAMazc/Bloqueo/0") {  if(messageTemp == "true"){ // Se desbloquea la cerradura.  digitalWrite(ledPin, HIGH);  } else if(messageTemp == "false"){ // Se bloquea la cerradura.  digitalWrite(ledPin, LOW);  }  }    // Funcion para indicar al usuario que el metdo de acceso fue incorrecto.  if (String(topic) == "Gabanni/UAMazc/SinAcceso/0") {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 0);  display.print("Acceso");  display.setCursor(0, 20);  display.print("denegado");  display.display();  delay(2000);  }  // Funcion para indicar al usuario que el metdo de acceso fue incorrecto.  if (String(topic) == "Gabanni/UAMazc/Ocupado/0") {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 0);  display.print("Lab.");  display.setCursor(0, 20);  display.print("Ocupado");  display.display();  delay(2000);  }  if (String(topic) == "Gabanni/UAMazc/AforoLleno/0") {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 0);  display.print("Aforo");  display.setCursor(0, 20);  display.print("lleno");  display.display();  delay(2000);  }  if (String(topic) == "Gabanni/UAMazc/TemperaturaAlta/0") {  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 0);  display.print("Temp. ");  display.setCursor(0, 20);  display.print("corporal");  display.setCursor(0, 40);  display.print("alta");  display.display();  delay(2000);  }  // Funcion para indicar al usuario que el metdo de acceso fue incorrecto.  if (String(topic) == "Gabanni/UAMazc/Despedida/0") {  // Mesnaje de identifacion.  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 0);  display.print("Adios");  display.setCursor(0, 20);  display.print(nombre);  display.display();  delay(2000);  }  if (String(topic) == "Gabanni/UAMazc/DeshabilitadoTemperatura/Alumno/0") {  // Mesnaje de identifacion.  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 0);  display.print("Hola");  display.setCursor(0, 20);  display.print(nombre);  display.display();  delay(2000);  StaticJsonDocument<256> doc; // Creamos un documento JSON, donde enviaremos la informacion de la temperatura y el usuario.  doc["matricula"] = id;  doc["nombre"] = nombre;  char out[128];  int b = serializeJson(doc, out); //Sereliarizamos el JSON para poderlo mandar por mqtt.  client.publish("Gabanni/UAMazc/InfoDeshabilitadoTemperatura/Alumno/0", out); //Publicamos en MQTT.  }  if (String(topic) == "Gabanni/UAMazc/DeshabilitadoTemperatura/Profesor/0") {  // Mesnaje de identifacion.  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 0);  display.print("Hola");  display.setCursor(0, 20);  display.print(nombre);  display.display();  delay(2000);  StaticJsonDocument<256> doc; // Creamos un documento JSON, donde enviaremos la informacion de la temperatura y el usuario.  doc["noEmpleado"] = id;  doc["nombre"] = nombre;  char out[128];  int b = serializeJson(doc, out); //Sereliarizamos el JSON para poderlo mandar por mqtt.  client.publish("Gabanni/UAMazc/InfoDeshabilitadoTemperatura/Profesor/0", out); //Publicamos en MQTT.  }  // Funcion para la lectura de temperatura coorporal para alumnos.  if (String(topic) == "Gabanni/UAMazc/LecturaTemperatura/Alumno/0") {  // Mesnaje de identifacion.  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 0);  display.print("Hola");  display.setCursor(0, 20);  display.print(nombre);  display.display();  delay(2000);  bool i = true;  while(i == true){ // Bucle para la lectura de la temperatura coorporal.    // Borra el trigPin  digitalWrite(trigPin, LOW);  delayMicroseconds(2);  // Establece el trigPin en estado ALTO durante 10 microsegundos  digitalWrite(trigPin, HIGH);  delayMicroseconds(10);  digitalWrite(trigPin, LOW);    // Lee el echoPin, devuelve el tiempo de viaje de la onda de sonido en microsegundos  duration = pulseIn(echoPin, HIGH);    // Calcular la distancia  int Dist = duration \* SOUND\_SPEED/2;    //No hay nadie  if(Dist>DistMin){  Presente=0;  Tiempo = millis();  }    //Llego alguien, tomemos el tiempo  if(Dist<=DistMin && Presente==0){  Presente=1;  Tiempo = millis();  }    if(Presente == 1){  if(millis()-Tiempo>Espera){ //Se completo el tiempo  Presente = 2;  }  }    TempObj=mlx.readObjectTempC(); // Temperatura del usuario.  TempObj = TempObj + 7.05; // 9.45 Ajuste a la temperatura.  switch(Presente){  case 0: // No hay nadie  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 30);  display.print("Buscando..");  display.display();  break;    case 1: // Llego alguien  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 30);  display.print("Leyendo...");  display.display();  break;    case 2: // Se Completo el tiempo se muestra la temperatura al usuario.  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(1);  display.setCursor(0,0);  display.print("Temperatura corporal:");  display.setTextSize(2);  display.setCursor(0,30);  display.print(TempObj);  display.print(" ");  display.setTextSize(1);  display.cp437(true);  display.write(167);  display.setTextSize(2);  display.print("C");  display.display();  delay(2000);  i = false;    StaticJsonDocument<256> doc; // Creamos un documento JSON, donde enviaremos la informacion de la temperatura y el usuario.  doc["matricula"] = id;  doc["nombre"] = nombre;  doc["temperatura"] = TempObj;  char out[128];  int b = serializeJson(doc, out); //Sereliarizamos el JSON para poderlo mandar por mqtt.  client.publish("Gabanni/UAMazc/InfoTemperatura/Alumno/0", out); //Publicamos en MQTT.  break;  }  }  }  // Funcion para la lectura de temperatura coorporal para alumnos.  if (String(topic) == "Gabanni/UAMazc/LecturaTemperatura/Profesor/0") {  // Mesnaje de identifacion.  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 0);  display.print("Hola");  display.setCursor(0, 20);  display.print(nombre);  display.display();  delay(2000);  bool i = true;  while(i == true){ // Bucle para la lectura de la temperatura coorporal.    // Borra el trigPin  digitalWrite(trigPin, LOW);  delayMicroseconds(2);  // Establece el trigPin en estado ALTO durante 10 microsegundos  digitalWrite(trigPin, HIGH);  delayMicroseconds(10);  digitalWrite(trigPin, LOW);    // Lee el echoPin, devuelve el tiempo de viaje de la onda de sonido en microsegundos  duration = pulseIn(echoPin, HIGH);    // Calcular la distancia  int Dist = duration \* SOUND\_SPEED/2;    //No hay nadie  if(Dist>DistMin){  Presente=0;  Tiempo = millis();  }    //Llego alguien, tomemos el tiempo  if(Dist<=DistMin && Presente==0){  Presente=1;  Tiempo = millis();  }    if(Presente == 1){  if(millis()-Tiempo>Espera){ //Se completo el tiempo  Presente = 2;  }  }    TempObj=mlx.readObjectTempC(); // Temperatura del usuario.  TempObj = TempObj + 7.05; // 9.45 Ajuste a la temperatura.  switch(Presente){  case 0: // No hay nadie  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 30);  display.print("Buscando..");  display.display();  break;    case 1: // Llego alguien  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 30);  display.print("Leyendo...");  display.display();  break;    case 2: // Se Completo el tiempo se muestra la temperatura al usuario.  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(1);  display.setCursor(0,0);  display.print("Temperatura corporal:");  display.setTextSize(2);  display.setCursor(0,30);  display.print(TempObj);  display.print(" ");  display.setTextSize(1);  display.cp437(true);  display.write(167);  display.setTextSize(2);  display.print("C");  display.display();  delay(2000);  i = false;    StaticJsonDocument<256> doc; // Creamos un documento JSON, donde enviaremos la informacion de la temperatura y el usuario.  doc["noEmpleado"] = id;  doc["nombre"] = nombre;  doc["temperatura"] = TempObj;  char out[128];  int b = serializeJson(doc, out); //Sereliarizamos el JSON para poderlo mandar por mqtt.  client.publish("Gabanni/UAMazc/InfoTemperatura/Profesor/0", out); //Publicamos en MQTT.  break;  }  }  }  } |

Tabla 0. setup\_wifi() , reconnect() y callback() del código.

La función setup() se incluye la declaración de variables, se trata de la primera función que se ejecuta en el programa. Esta función se ejecuta una única vez y es empleada para configurar e inicializar variables, componentes, pines, etc.

|  |
| --- |
| void setup() {  Serial.begin(115200); //Se inicia el monitor serie a 115200 baudios.    //Se inicia la pantalla  display.begin(SSD1306\_SWITCHCAPVCC, 0x3C);    setup\_wifi(); // Se inicia la conexion wifi.  client.setServer(mqtt\_server, 1883); // Se inicia comunicacion con el servidor MQTT.  client.setCallback(callback); // Se inicia la funcion callback.  pinMode(pinDoor, INPUT); // Se inicializa el sensor de puerta  pinMode(ledPin, OUTPUT); // Se inicializa el LED Pin (led para la senal de apertura de la chapa).  pinMode(2, OUTPUT);  digitalWrite(2,LOW);  //Iniciamos sensor RFID  SPI.begin(); // Iniciamos el bus SPI.  mfrc522.PCD\_Init(); // Iniciamos el objeto MFRC522.  delay(4); // Le damos tiempo para que se inicie.  mfrc522.PCD\_DumpVersionToSerial(); //Opcional, mostramos detalles de las tarjetas MFRC522.  //Iniciamos sensor de huellas  finger.begin(57600); //Establecemos la velocidad de datos para el puerto serie del sensor  delay(5);    //Iniciando el sensor ultrasonico  pinMode(trigPin, OUTPUT);  pinMode(echoPin, INPUT);  //Se inicia el sensor Temperatura  mlx.begin(0x5A);  //Mesnaje de bienvenida  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 0);  display.print("Bienvenido");  display.setCursor(0, 20);  display.print(" GABANNI ");  display.display();  delay(2000);  // Dibujamos el logo.  display.clearDisplay();  display.drawBitmap(32, 0, image, 64, 81, 1);  display.display();  delay(2000);  pinMode(boton,INPUT\_PULLUP);  estado = HIGH;  anterior = HIGH;  } |

Tabla 0. setup() del código.

La función loop() contiene el código que se ejecutara continuamente(lectura de entradas, activación de salidas, etc.), Esta función de ejecuta justo después de la función setup(), su principal característica es que se ejecutara infinitamente, por lo que, aquí se escribe toda la lógica de programación , que forma parte de nuestro programa.

|  |
| --- |
| void loop() {  if (!client.connected()) { // Reconectado en caso de perdida de senal Wifi.  reconnect();  }    client.loop();  // Si el estado es igual a lo leido, la entrada no ha cambiado lo que  // significa que no hemos apretado el botón (ni lo hemos soltado); asi que  // tenemos que parar el temporizador.  if ( estado==digitalRead(boton) ) {  temporizador = 0;  }  // Si el valor distinto significa que hemos pulsado/soltado el botón. Ahora  // tendremos que comprobar el estado del temporizador, si vale 0, significa que  // no hemos guardado el tiempo en el que sa ha producido el cambio, así que  // hemos de guardarlo.  else  if ( temporizador == 0 ) {  // El temporizador no está iniciado, así que hay que guardar  // el valor de millis en él.  temporizador = millis();  }  else  // El temporizador está iniciado, hemos de comprobar si el  // el tiempo que deseamos de rebote ha pasado.  if ( millis()-temporizador > tiemporebote ) {  // Si el tiempo ha pasado significa que el estado es lo contrario  // de lo que había, asi pues, lo cambiamos.  estado = !estado;  }    // Ya hemos leido el botón, podemos trabajar con él.  if ( anterior==HIGH && estado==LOW ){  digitalWrite(ledPin, HIGH);  delay(5000);  digitalWrite(ledPin, LOW);  }    // Recuerda que hay que guardar el estado anterior.  anterior = estado;  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0,20);  display.print("Esperando");  display.setCursor(0,40);  display.print(" .... ");  display.display();  //Si el usuario oprime el teclado, le hablamos a la funcion Fun\_teclado()  char key = teclado.getKey();  if (key){  display.clearDisplay();  display.setTextColor(WHITE);  display.setTextSize(2);  display.setCursor(0, 20);  display.print("Escribe tu");  display.setCursor(0, 40);  display.print("clave");  display.display();  Fun\_teclado(key);  delay(2000);  }  // Si el usuario presenta su dedo en el sensor de huellas datilares, le hablamos a la funcion getFingerprintID()  getFingerprintID();    // Si el usuario presenta una tarjeta en el sensor RFID, le hablamos a la funcion Fun\_RFID()  if(mfrc522.PICC\_IsNewCardPresent()) {  Fun\_RFID();  }  EstadoN = digitalRead(pinDoor);  if(EstadoV != EstadoN){  EstadoPuerta() ;  }  } |

Tabla 0. loop() del código.