Librepass

Informe técnico

Un trabajo presentado para la materia de
Proyectos y Diseño Electrónico

Krapp Ramiro

Instituto tecnológico San Bonifacio
Departamento de electrónica
30 de septiembre de 2022

Hecho en LaTeX
Versión Alpha 0.1

0. Índice

1. Introducción **2**
1.1. El proyecto y el Software Libre . . . 2
1.2. Licencia . . . 3

2. Diagrama en Bloques **4**

3. Diagrama Esquematico **4**

4. Partes del proyecto **5**
4.1. El protocolo de comunicación SPI . . . 5
4.1.1. Ventajas . . . 7
4.1.2. Desventajas . . . 8
4.2. DOIT ESP32 DevKit v1 . . . 9
4.2.1. Características Técnicas . . . 9
4.2.2. Pinout . . . 10
4.2.3. Usabilidad de pines . . . 11
4.3. Sistema RFID . . . 12
4.3.1. Pinout del dispositivo . . . 13
4.3.2. Mapeo de Memoria del tag RFID . . . 14
4.4. El Webserver asincrónico . . . 15
4.5. Framework Arduino . . . 16
4.5.1. Funciones . . . 16
4.5.2. Variables . . . 18
4.5.3. Estructura . . . 19

5. Codigo del programa **20**

6. Bitacoras Personales **22**
6.1. Krapp Ramiro . . . 22

El índice tiene hipervínculos incorporados! Toca en cada seccion y automaticamente tu lector de pdfs te llevara a esa página

1. Introducción

Librepass es un sistema FOSS(Free and Open Source) de seguridad para empresas. Al ser FOSS, está hosteado en un repositorio público en GitHub — `https://github.com/KrappRamiro/librepass`.

Fue desarrollado usando una placa de desarrollo DOIT ESP32 DevKit V1, conectado a un array de lectores RFID-RC552.

Estos lectores son capaces de leer un sistema de tarjetas y/o llaveros RFID con un código hexadecimal indentificador, el cual se asigna a cada empleado de la empresa, y sirve para identificar al empleado.

A nivel de hardware, hay 3 componentes involucrados:

1. El microcontrolador: un ESP32
2. EL PCD (Proximity Coupling Device): RFID-MFRC522
3. El PICC (Proximity Integrated Circuit Card): Una tarjeta o llavero usando la interfaz ISO 14443A

1.1. El proyecto y el Software Libre

En el desarrollo de este proyecto, se planteó usar la filosofía del software libre. Segun GNU [22]:

"«Software libre» es el software que respeta la libertad de los usuarios y la comunidad. A grandes rasgos, significa que los usuarios tienen la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software. Es decir, el «software libre» es una cuestión de libertad, no de precio. Para entender el concepto, piense en «libre» como en «libre expresión», no como en «barra libre». En inglés, a veces en lugar de «free software» decimos «libre software», empleando ese adjetivo francés o español, derivado de «libertad», para mostrar que no queremos decir que el software es gratuito.

Puede haber pagado dinero para obtener copias de un programa libre, o puede haber obtenido copias sin costo. Pero con independencia de cómo obtuvo sus copias, siempre tiene la libertad de copiar y modificar el software, incluso de vender copias.

(...)

Un programa es software libre si los usuarios tienen las cuatro libertades esenciales:

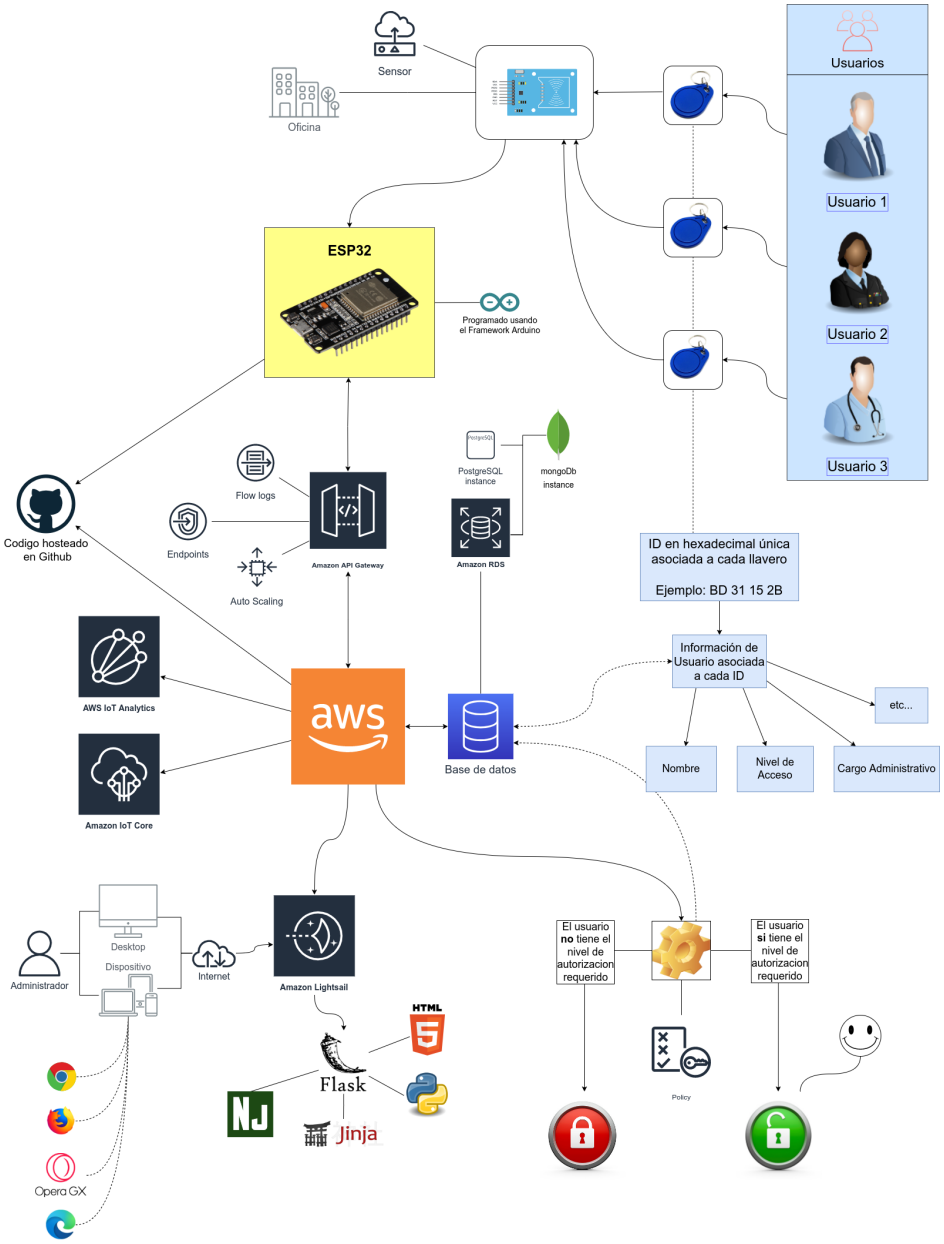
- La libertad de ejecutar el programa como se desee, con cualquier propósito (libertad 0).
- La libertad de estudiar cómo funciona el programa, y cambiarlo para que haga lo que se desee (libertad 1). El acceso al código fuente es una condición necesaria para ello.
- La libertad de redistribuir copias para ayudar a otros (libertad 2).
- La libertad de distribuir copias de sus versiones modificadas a terceros (libertad 3). Esto le permite ofrecer a toda la comunidad la oportunidad de beneficiarse de las modificaciones. El acceso al código fuente es una condición necesaria para ello.

Un programa es software libre si otorga a los usuarios todas estas libertades de manera adecuada. De lo contrario no es libre. Existen diversos esquemas de distribución que no son libres, y si bien podemos distinguirlos en base a cuánto les falta para llegar a ser libres, nosotros los consideramos contrarios a la ética a todos por igual."

1.2. Licencia

Se escogió usar la licencia MIT [23], la cual, en ingles, es la siguiente:

2. Diagrama en Bloques



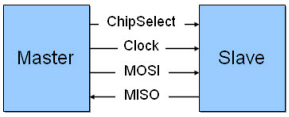
3. Diagrama Esquematico

Este es el diagrama esquematico del proyecto

4. Partes del proyecto

4.1. El protocolo de comunicación SPI

El protocolo Serial Peripheral Interface es un protocolo de comunicación creado por Motorola, anunciado en el año 1979. El mismo se divide en 4 lineas de comunicación, cada una con una función específica (por favor, ver figura 1) con:



- Una señal de clock llamada SCLK, enviada desde el bus master a todos los slaves. Todas las señales del protocolo van as er sínconas a esta señal de clock
- Una señal de selección de slave llamada SSn, usada para seleccionar con que slave se esta comunicando el master
- Una linea de datos desde master hacia slave, llamada MOSI (Master Out Slave In)
- Una linea de datos desde slave hacia master, llamada MISO (Master In Slave OUT)

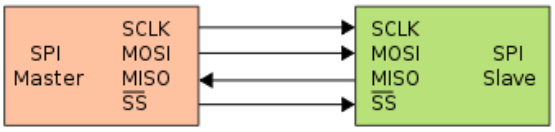


Figura 1: SPI master conectado a un único slave.

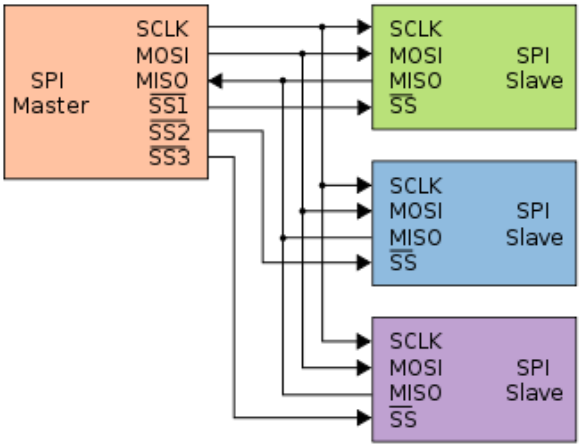


Figura 2: SPI master conectado a múltiples slaves.

no olvidarse de la daisy chained

SPI es un protoclo de comunicación single-master, esto significa que un dispositivo central (normalmente un microcontrolador) es el encargado de iniciar todas las comunicaciónes con los slaves.

Cuando el master SPI desea enviar o recibir información de un slave, selecciona el slave seteando en LOW la linea SS correspondiente, y activa la señal de clock a una frecuencia usable por el master y el slave. A partir de ese momento, el master envía la información por el canal MOSI mientras lee la información que hay en el canal MISO

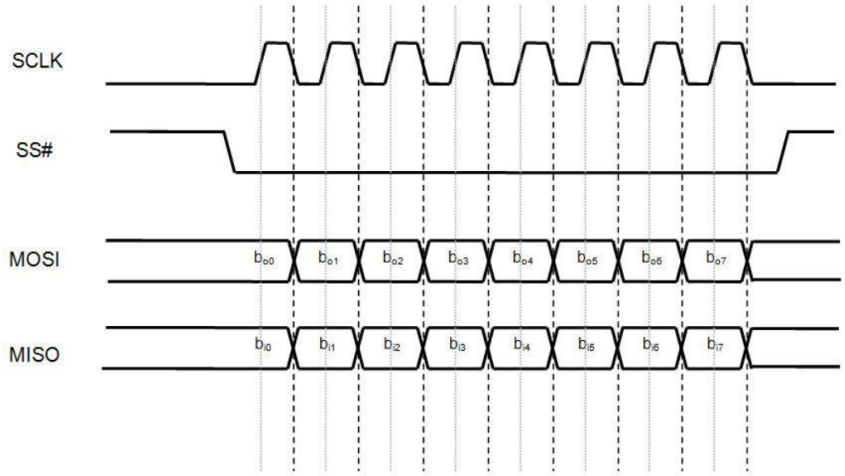


Figura 3: El timing de una comunicación SPI. En este ejemplo, La transmisión de datos por los canales MOSI y MISO es ejecutada por cada flanco descendente en la señal de clock en SCLK. En cambio, la lectura de datos es ejecutada por cada flanco ascendente. Esto se puede cambiar modificando el SPI mode

Como se menciona en la figura 3, hay 4 modos SPI, que van del 0 al 3. Los modos SPI definen en que flanco se activa la linea MOSI, MISO, y el estado (LOW o HIGH) de inactividad (idle) del canal SCLK. Cada modo esta definido por un par de parámetros llamados clock polarity (polaridad de clock) (CPOL), y clock phase (fase de clock) (CPHA)

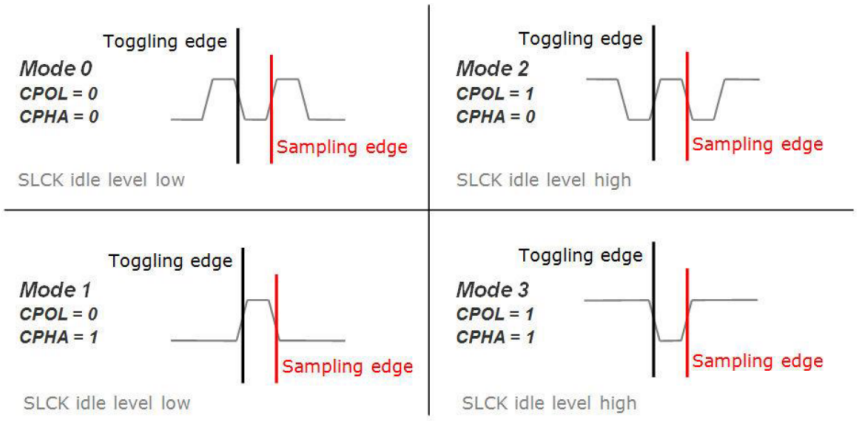


Figura 4: Los modos SPI son definidos con los parámetros CPOL (clock polarity) y CPHA (clock phase), que definen 3 parámetros: El flanco usado para envío de datos, el flanco usado para recepción de datos, y el estado de inactividad (idle) de SCLK

Una conexión SPI master/slave tiene que usar el mismo set de parámetros explicados en la figura 4 para poder efectuar una comunicación. Si de todas formas se desea que múltiples slaves tengan configuraciones distintas, el master deberá reconfigurarse cada vez que se desee comunicar con cada dispositivo.

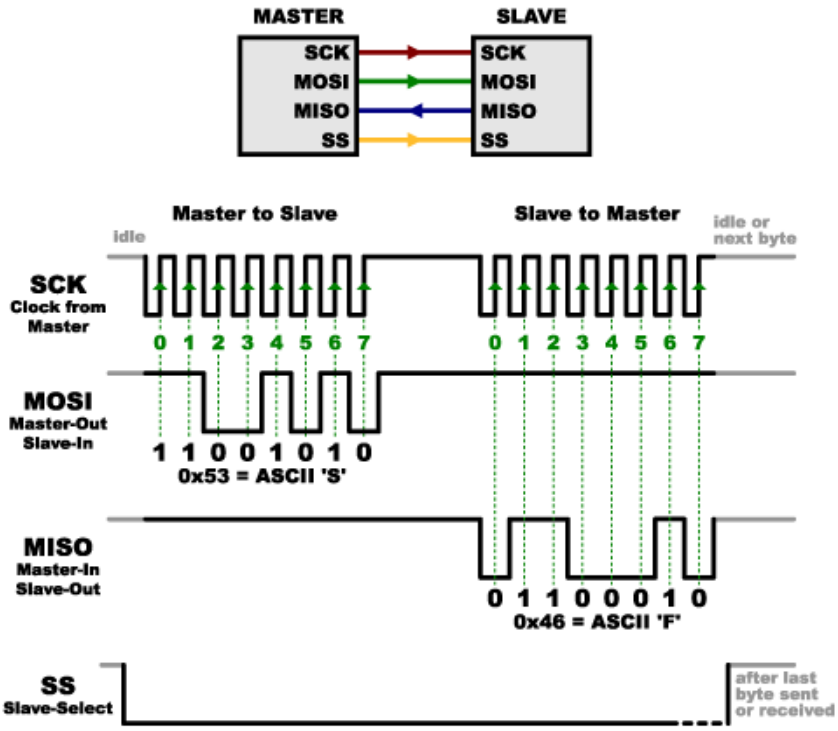


Figura 5: Grafico de comunicacion SPI

4.1.1. Ventajas

Segun Wikipedia[14]:

- Comunicación Full Duplex
- Mayor velocidad de transmisión que con I^2C o SMBus
- Protocolo flexible en que se puede tener un control absoluto sobre los bits transmitidos
- No está limitado a la transferencia de bloques de 8 bits
- Elección del tamaño de la trama de bits, de su significado y propósito
- Su implementación en hardware es extremadamente simple
- Consume menos energía que I^2C o que SMBus debido que posee menos circuitos (incluyendo las resistencias pull-up) y estos son más simples
- No es necesario arbitraje o mecanismo de respuesta ante fallos
- Los dispositivos clientes usan el reloj que envía el servidor, no necesitan por tanto su propio reloj
- No es obligatorio implementar un transceptor (emisor y receptor), un dispositivo conectado puede configurarse para que solo envíe, sólo reciba o ambas cosas a la vez
- Usa mucho menos terminales en cada chip/conector que una interfaz paralelo equivalente
- Como mucho una única señal específica para cada cliente (señal SS), las demás señales pueden ser compartidas

4.1.2. Desventajas

- Consume más pines de cada chip que I²C, incluso en la variante de 3 hilos
- El direccionamiento se hace mediante líneas específicas (señalización fuera de banda) a diferencia de lo que ocurre en I²C que se selecciona cada chip mediante una dirección de 7 bits que se envía por las mismas líneas del bus
- No hay control de flujo por hardware
- No hay señal de asentimiento. El servidor podría estar enviando información sin que estuviese conectado ningún cliente y no se daría cuenta de nada
- No permite fácilmente tener varios servidores conectados al bus
- Sólo funciona en las distancias cortas a diferencia de, por ejemplo, RS-232, RS-485, o Bus CAN

4.2. DOIT ESP32 DevKit v1

El kit de desarrollo DOIT ESP32 DevKit v1 es una de las placas de desarrollo creadas por DOIT. Esta basada en el microcontrolador ESP32, que en un mismo chip tiene soporte para WiFi, Bluetooth, Ethernet y Low-Power

4.2.1. Características Técnicas

- Microcontrolador: Tensilica 32-bit Single/Dual-core CPU Xtensa LX6
- Tensión de operación: 3.3V
- Tensión de alimentación: 7-12V
- Pines I/O digitales (DIO): 25
- Pines analógicos de Entrada (ADC): 6
- Pines analógicos de Salida (DAC): 2
- UARTs: 3
- SPIs: 2
- I2Cs: 3
- Memoria Flash: 4 MB
- SRAM: 520 KB
- Velocidad de clock: 240 Mhz
- Wi-Fi: IEEE 802.11 b/g/n/e/i, con las siguientes características:
  - Switch TR, Balun, LNA, Amplificador de potencia y antena integrados
  - Autenticación WEP, WPA/WPA2, con la opcion de tambien acceder a redes abiertas.

4.2.2. Pinout

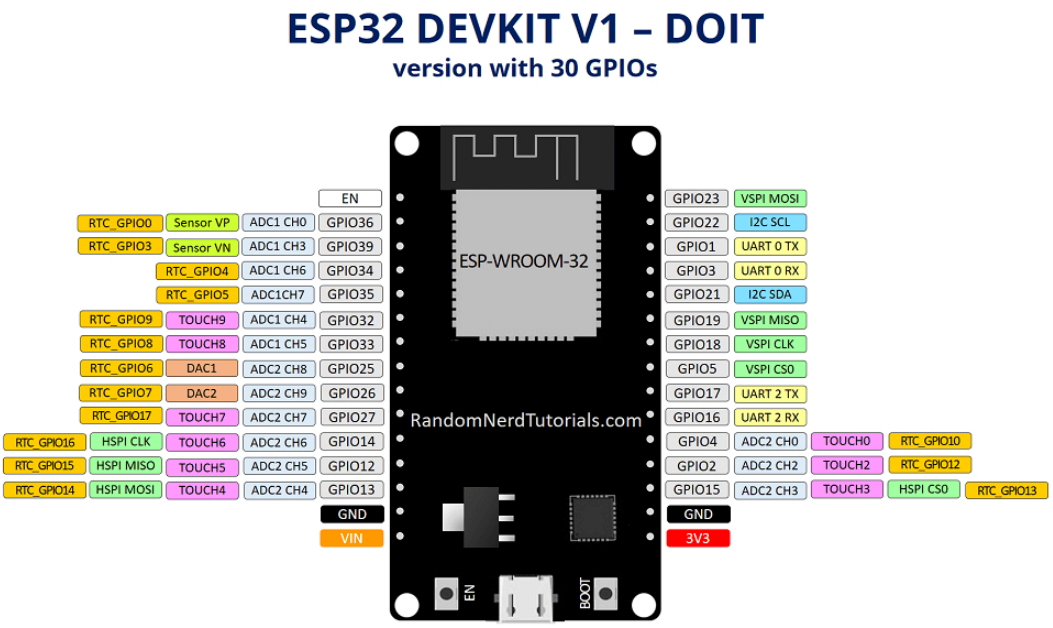


Figura 6: Pinout de la placa de desarrollo DOIT DevKit V1 ESP32 de 30 pines

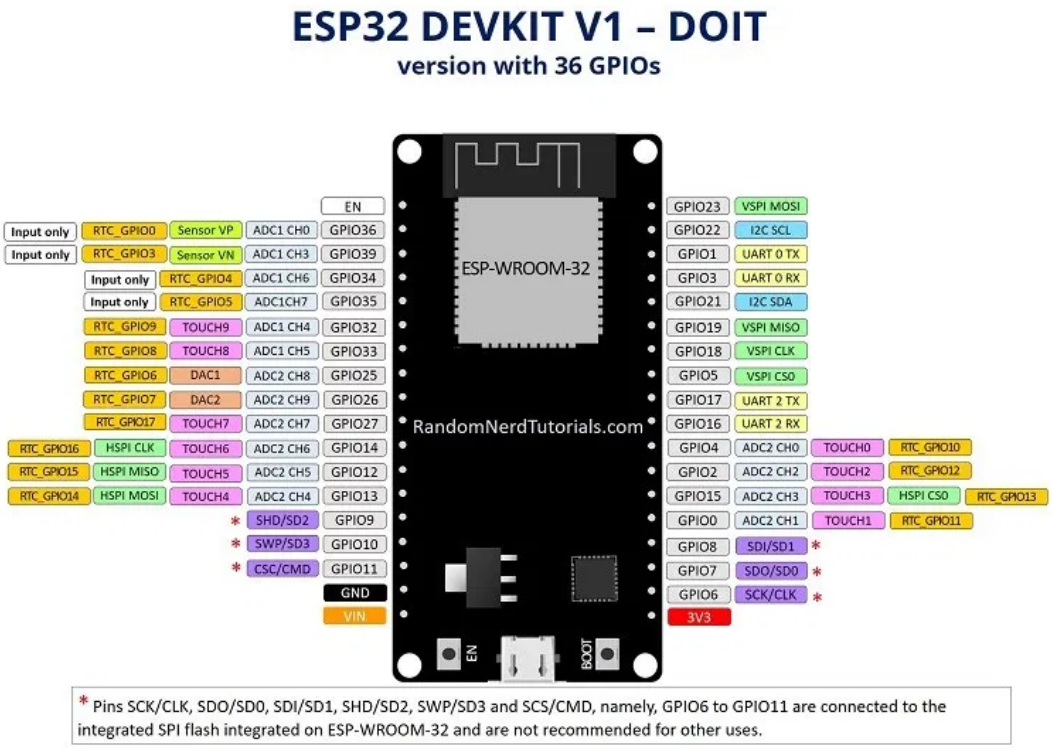


Figura 7: Pinout de la placa de desarrollo DOIT DevKit V1 ESP32 de 36 pines, cabe recordar que los pines GPIO 6-11 estan reservados al sistema SPI integrado, y su uso no es recomendado

4.2.3. Usabilidad de pines

El ESP32 cuenta con una multitud de pines, pero no todos pueden ser usados libremente, esto es explicado en la tabla 1 que muestra que pines pueden ser utilizados y cuales no, dependiendo de las cirscuntancias.

A la hora de ser usados en el codigo c++ del framework Arduino, simplemente se refieren por el numero

GPIO	Input	Output	Notes
0	pulled up	OK	Hace output de señal PWM al arranque
1	TX pin	OK	debug output al arranque
2	OK	OK	Conectado al LED_ONBOARD
3	OK	RX pin	En estado HIGH al arranque
4	OK	OK	
5	OK	OK	Hace output de señal PWM al arranque
6	x	x	Conectado al flash SPI integrado
7	x	x	Conectado al flash SPI integrado
8	x	x	Conectado al flash SPI integrado
9	x	x	Conectado al flash SPI integrado
10	x	x	Conectado al flash SPI integrado
11	x	x	Conectado al flash SPI integrado
12	OK	OK	El arranque falla si está pulleado en HIGH
13	OK	OK	
14	OK	OK	Hace output de señal PWM al arranque
15	OK	OK	Hace output de señal PWM al arranque
16	OK	OK	
17	OK	OK	
18	OK	OK	
19	OK	OK	
21	OK	OK	
22	OK	OK	
23	OK	OK	
25	OK	OK	
26	OK	OK	
27	OK	OK	
32	OK	OK	
33	OK	OK	
34	OK	x	Solamente de input
35	OK	x	Solamente de input
36	OK	x	Solamente de input
39	OK	x	Solamente de input

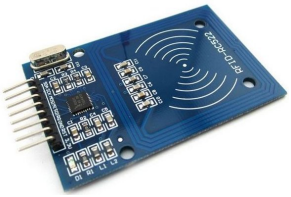
Cuadro 1: Una tabla con las funciones de cada pin de la placa de desarrollo DOIT DevKit v1 ESP32

4.3. Sistema RFID

Segun Wikipedia[13]:

"RFID o identificación por radiofrecuencia (del inglés Radio Frequency Identification) es un sistema de almacenamiento y recuperación de datos remotos que usa dispositivos denominados etiquetas, tarjetas o transpondedores RFID.

El propósito fundamental de la tecnología RFID es transmitir la identidad de un objeto (similar a un número de serie único) mediante ondas de radio. Las tecnologías RFID se agrupan dentro de las denominadas Auto ID (automatic identification, o identificación automática).



Las etiquetas RFID (RFID tag en inglés) son unos dispositivos pequeños, similares a una pegatina, que pueden ser adheridas o incorporadas a un producto, un animal o una persona. Contienen antenas para permitirles recibir y responder a peticiones por radiofrecuencia desde un emisor-receptor RFID. Las etiquetas pasivas no necesitan alimentación eléctrica interna, mientras que las activas sí lo requieren.

Una de las ventajas del uso de radiofrecuencia (en lugar, por ejemplo, de infrarrojos) es que no se requiere visión directa entre emisor y receptor"

(a) Un llavero RFID

(b) Una tarjeta RFID

Figura 8: Distintos tags RFID

4.3.1. Pinout del dispositivo

pin SDA — Este pin se utiliza de forma distinta dependiendo del protocolo de comunicación utilizado.

- En I2C, se usa como el pin SDA.
- En UART, se usa como pin RX.
- En SPI, se usa como el pin SS

pin SCK — El pin SCK se usa para mantener el sincronísmo con una señal de reloj

pin MOSI — El pin MOSI sirve para hacer una transmisión Master Out - Slave In

pin MISO — El pin MISO sirve para hacer una transmisión Master In - Slave Out

pin IRQ — Se usa para las interrupciones

GND — Sirve para mantener la referencia con Masa

RST — Este pin sirve para resetear o desactivar el circuito integrado

VCC — Pin de alimentación **3.3v**

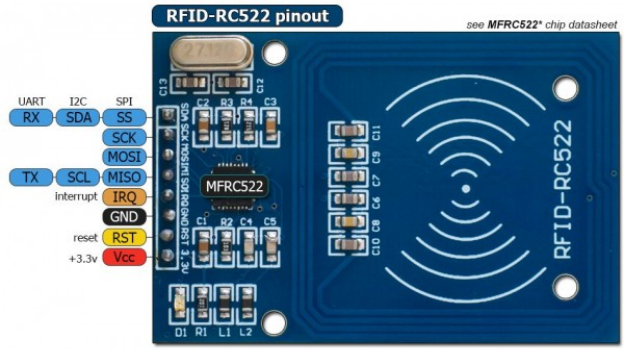


Figura 9: El pinout del lector RFID-RC552. Se puede notar como este dispositivo está adaptado para funcionar con 3 protocolos distintos, comunicación por UART, comunicación por I2C y comunicacion por SPI

4.3.2. Mapeo de Memoria del tag RFID

La identificación se realiza con unos llaveros o unas tarjetas, que tienen este mapeo de memoria:

Tenemos 1k de memoria adentro de este chip, la cual esta organizada de la siguiente manera: Hay 16 sectores de 4 bloques, y cada bloque contiene 16 bytes.

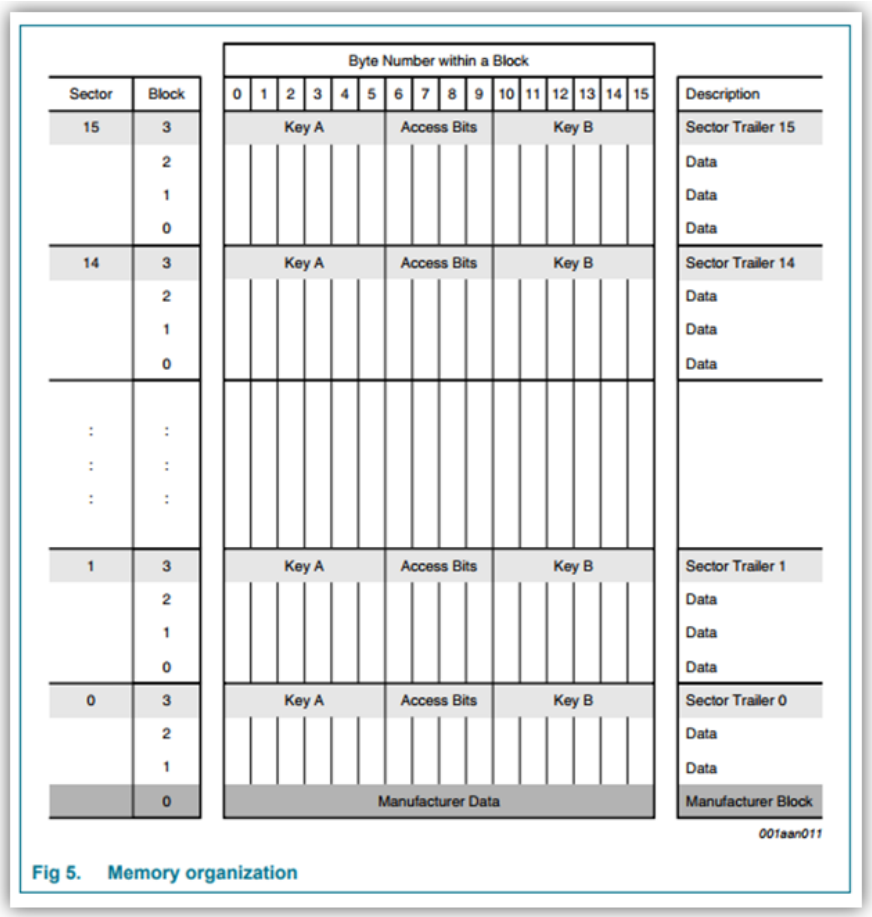


Fig 5. Memory organization

4.4. El Webserver asincrónico

Para este proyecto, se usó un sistema de servidor web integrado en el DOIT DevKit V1 ESP32

meter captura de la pagina web

meter pedazo de codigo de los server.on, el processor y el javascript

4.5. Framework Arduino

Para este proyecto, se uso el framework Arduino, que provee una amplia variedad de clases, metodos y funciones útiles para el desarrollo en sistemas embebidos. Segun la referencia de Arduino[10], estas son las cosas que el framework provee:

4.5.1. Funciones

Para controlar la placa y realizar cómputos

Digital I/O

- digitalRead()
- digitalWrite()
- pinMode()

Analog I/O

- analogRead()
- analogReference()
- analogWrite()

Zero, Due & MKR Family

- analogReadResolution()
- analogWriteResolution()

Advanced I/O

- noTone()
- pulseIn()
- pulseInLong()
- shiftIn()
- shiftOut()
- tone()

Time

- delay()
- delayMicroseconds()
- micros()
- millis()

Math

- abs()
- constrain()
- map()
- max()
- min()
- pow()
- sq()
- sqrt()

Trigonometry

- cos()
- sin()
- tan()

Characters

- isAlpha()
- isAlphaNumeric()
- isAscii()
- isControl()
- isDigit()
- isGraph()
- isHexadecimalDigit()
- isLowerCase()
- isPrintable()
- isPunct()
- isSpace()
- isUpperCase()
- isWhitespace()

Random Numbers

- random()
- randomSeed()

Bits and Bytes

- bit()
- bitClear()
- bitRead()
- bitSet()
- bitWrite()
- highByte()
- lowByte()

External Interrupts

- attachInterrupt()
- detachInterrupt()

Interrupts

- interrupts()
- noInterrupts()

Communication

- Serial
- SPI
- Stream
- Wire

USB

- Keyboard
- Mouse

4.5.2. Variables

Tipos de Datos y Constantes definidos en el framework

Constants

- HIGH | LOW
- INPUT | OUTPUT | INPUT_PULLUP
- LED_BUILTIN
- true | false
- Floating Point Constants
- Integer Constants

Conversion

- (unsigned int)
- (unsigned long)
- byte()
- char()
- float()
- int()
- long()
- word()

Data Types

- array
- bool
- boolean
- byte
- char
- double
- float
- int
- long
- short
- size_t
- string
- String()
- unsigned char
- unsigned int
- unsigned long
- void
- word

Variable Scope & Qualifiers

- const
- scope
- static
- volatile

Utilities

- PROGMEM
- sizeof()

4.5.3. Estructura

Los elementos del codigo Arduino (C++).

Sketch

- loop()
- setup()

Control Structure

- break
- continue
- do...while
- else
- for
- goto
- if
- return
- switch...case
- while

Further Syntax

- #define (define)
- #include (include)
- /* */ (block comment)
- // (single line comment)
- ; (semicolon)
- (curly braces)

Arithmetic Operators

- % (remainder)
- * (multiplication)
- + (addition)
- - (subtraction)
- / (division)
- = (assignment operator)

Comparison Operators

- != (not equal to)
- <(less than)
- <= (less than or equal to)
- == (equal to)
- >(greater than)
- >= (greater than or equal to)

Boolean Operators

- ! (logical not)
- && (logical and)
- || (logical or)

Pointer Access Operators

- & (reference operator)
- * (dereference operator)

Bitwise Operators

- & (bitwise and)
- < < (bitshift left)
- > > (bitshift right)
- ^ (bitwise xor)
- | (bitwise or)
- ~ (bitwise not)

Compound Operators

- %= (compound remainder)
- &= (compound bitwise and)
- *= (compound multiplication)
- ++ (increment)
- += (compound addition)
- -- (decrement)
- - = (compound subtraction)
- /= (compound division)
- ^ = (compound bitwise xor)
- |= (compound bitwise or)

5. Codigo del programa

El codigo de programa fue escrito usando Visual Studio Code, usando la extensión de platformIO con el framework de Arduino.

Para el versionado del código, se uso Git https://git-scm.com/, un programa FOSS estandar en la industria. Para una mejor organización, se dividió en tres branches principales:

1. Una branch main, con las versiones estables del código.
2. Una branch dev, con las versiones de desarrollo.
3. Una branch dev-tema-a-desarrollar, (reemplazando tema-a-desarrollar por el tema que se desarrolla, se usaba una de estas y despues se mergeaba con dev)

Codigo principal

```
#include "ArduinoJson.h"
#include "HTTPClient.h"
#include "krapp_utils.h"
#define SS_PIN 5
#define RST_PIN 21
#define SIZE_BUFFER 18 // Este es el tamaño del buffer con el que voy a estar trabajando.
// Por que es 18? Porque son 16 bytes de los datos del tag, y 2 bytes de checksum
#define MAX_SIZE_BLOCK 16
#define greenPin 12
#define redPin 32

//------------------ INICIO DE Configuracion de conexion a internet ----------------
const char* ssid = "Krapp"; // Nombre de la red
const char* password = "laputamadre"; // Contraseña de la red
// String for storing server response
String response = "";
// JSON document
DynamicJsonDocument doc(2048);
// ----------------- FIN DE Configuracion de conexion a internet ------------------

//  ---------------- INICIO DE Variables del MFRC552 ---------------------
// key es una variable que se va a usar a lo largo de todo el codigo
MFRC522::MIFARE_Key key;
// Status es el codigo de estado de autenticacion
MFRC522::StatusCode status;
// Defino los pines que van al modulo RC552
MFRC522 mfrc522(SS_PIN, RST_PIN);
// ----------------- FIN DE Variables del MFRC552 ---------------------

void setup()
{
   Serial.begin(9600);
   SPI.begin(); // Inicio el bus SPI
   Log.begin(LOG_LEVEL_NOTICE, &Serial); // Inicio del sistema de logging

   // Prendo el led de la placa cuando inicia el sistema
   pinMode(LED_BUILTIN, OUTPUT);
   digitalWrite(LED_BUILTIN, HIGH);
   delay(1000);
   digitalWrite(LED_BUILTIN, LOW);

   // Me conecto a internet mediante Wi-Fi
   WiFi.begin(ssid, password);
   while (WiFi.status() != WL_CONNECTED) {
      delay(1000);
      Serial.println("Connecting to WiFi..");
   }
   // Imprimo la IP local del ESP32 (192.168.x.x)
   Serial.println(WiFi.localIP());

```

```
51    // Inicio el MFRC552
52    mfrc522.PCD_Init();
53    // Le pido al usuario que acerque el tag RFID
54    Serial.println(F("Acerca tu tarjeta RFID\n"));
55 }
56
57 void loop()
58 {
59
60    HTTPClient http; // Inicio el cliente http
61    String request = "http://192.168.50.62:5000/api/let_employee_pass/AC_35_71_3A/3";
62
63    // Start the request
64    http.begin(request);
65    // Use HTTP GET request
66    http.GET();
67    // Response from server
68    response = http.getString();
69    // Parse JSON, read error if any
70    DeserializationError error = deserializeJson(doc, response);
71    if (error) {
72       Serial.print(F("deserializeJson() failed: "));
73       Serial.println(error.f_str());
74       return;
75    }
76    // Print parsed value on Serial Monitor
77    Serial.println(doc["mensaje"].as<char*>());
78    // Close connection
79    http.end();
80    // Wait two seconds for next joke
81    delay(2000);
82    // ------- INICIO DEL LECTOR RFID --------------
83    // Se espera a que se acerque un tag
84    if (!mfrc522.PICC_IsNewCardPresent()) {
85       return;
86    }
87    // Se espera a que se lean los datos
88    if (!mfrc522.PICC_ReadCardSerial()) {
89       return;
90    }
91    // Descomentar solamente si se quiere Dumpear toda la info acerca de la tarjeta leida, ojo que llama
      ↪  automaticamente a PICC_HaltA()
92    // mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
93
94    // Le dice al PICC que se vaya a un estado de STOP cuando esta activo (o sea, lo haltea)
95    mfrc522.PICC_HaltA();
96
97    // Esto "para" la encriptación del PCD (proximity coupling device).
98    // Tiene que ser llamado si o si despues de la comunicacion con una
99    // autenticación exitosa, en otro caso no se va a poder iniciar otra comunicación.
100   mfrc522.PCD_StopCrypto1();
101   // ------- FIN DEL LECTOR RFID --------------
102 }
```

6. Bitacoras Personales

6.1. Krapp Ramiro

6.1.0. 24/03/2022

- Comence creando un repositorio en github para subir todos los cambios del proyecto
- Cree un codigo en C++, para definir un sistema de clases. La idea es hacer una clase Tren, para que sirva de blueprint para todos los trenes, y una clase Persona, para que sea padre de otras dos clases, Maquinista y Pasajero. Al pasajero le voy a asignar una sube, y al maquinista le voy a asignar un salario y un seniority.

6.1.0. 25/03/2022

- Pienso implementar la sube con un sistema usando RFID
  https://randomnerdtutorials.com/security-access-using-mfrc522-rfid-reader-with-arduino/
- La idea seria armar un sistema en el que cada usuario pueda tener un llavero RFID, y que asigne ese llavero RFID con una cuenta. Tambien necesito comprar los lectores para RFID. En total, tengo pensado comprar 2 lectores y 4 llaveros RFID. Por qué 2 lectores? Estaba pensando en asignar cada uno a una estación distinta. Por qué 4 llaveros? Estaba pensando en asignar cada uno a un pasajero distinto.
- Encontre que para en LaTeX dejar de tener problema con las url yendose fuera pantalla, puedo usar el paquete url con la opcion [hyphens], lo unico es que hay que cargar este paquete antes de hyperref. Esto es porque por defecto el paquete hyperref ya carga al paquete url https://tex.stackexchange.com/questions/544671/option-clash-for-package-url-urlstyle

6.1.0. 26/03/2022

- Encontre mucha documentacion del ESP32 y de proyectos con el RFID, la principal es esta:
- https://arduinogetstarted.com/tutorials/arduino-rfid-nfc
- https://olddocs.zerynth.com/latest/official/board.zerynth.doit_esp32/docs/index.html
- https://testzdoc.zerynth.com/reference/boards/doit_esp32/docs/
- https://randomnerdtutorials.com/esp32-pinout-reference-gpios/
- https://randomnerdtutorials.com/getting-started-with-esp32/
- Voy a usar el grafico de randomnerdtutorials, del link de getting-started..., el que incluye que pines son GPIO, me va a servir un montón. Para cuando quiera programar, solamente tengo que recordar que lo mejor es usar los GPIO del 13 al 33, y que mi DOIT ESP32 DevKit V1 es la version de 30 pines
- Decidi seguir el tutorial de este link https://www.instructables.com/ESP32-With-RFID-Access-Control/
- Hice andar el codigo durante un tiempo, grabe que funcionaba incluso, pero de repente dejo de funcionar, solamente me da un error: PCD_Authenticate() failed: Timeout in communication.
- Creo que se por qué dejó de funcionar, me parece que cortocircuité algo con el la parte de metal del llavero, me parece haber cortocircuitado los pines del lector RFID-RC552

6.1.0. 27/03/2022

- Hice una branch nueva en git para trabajar exclusivamente en el informe, la llamé update_informe. Aproveché para eliminar la sección Base de Datos, que me había quedado ahí de un copypaste de un proyecto anterior.

6.1.0. 28/03/2022

- Cometi un error haciendo un stash en git y elimine parte del trabajo que hice en el informe :'(
- Encontre este codigo que me puede servir

  https://esp32io.com/tutorials/esp32-rfid-nfc
- Tambien encontre la documentacion de la libreria para los RFID que usa el protocolo de comunicacion MFRC, pero como usa SPI no se como meter varios RFID en paralelo sin usar RFID, lo tendria que investigar https://www.arduino.cc/reference/en/libraries/mfrc522/

6.1.0. 29/03/2022

- Investigando info para hacer el informe y saber más sobre SPI, encontre esto:
  - https://www.arduino.cc/en/reference/SPI
  - https://arduinogetstarted.com/faq/how-to-connect-multiple-spi-sensors-devices-with-arduino (Especialmente útil para conectar multiples dispositivos)
  - https://arduinogetstarted.com/tutorials/arduino-rfid-nfc
  - https://www.exostivlabs.com/files/documents/Introduction-to-I2C-and-SPI-Protocols.pdf?/article/aa-00255/22/introduction-to-spi-and-ic-protocols.html
  - https://www.corelis.com/education/tutorials/spi-tutorial/
  - https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/
- Creo que para lo que quiero hacer me sirve la conexion daisy-chained del protocolo SPI
- Para las bibliografias, voy a usar esto https://latex-tutorial.com/tutorials/bibtex/
- también este tutorial sirve https://www.overleaf.com/learn/latex/Bibliography_management_with_biblatex
- Estuve trabajando en el informe, hice gran parte de la sección del SPI

6.1.0. 30/03/2022

- Hoy estoy trabajando en la documentacion del sistema RFID, mientras espero que lleguen los componentes que compré. Estoy haciendo la documentación porque me sirve para estudiar y ya entrar a armar cosas con más conocimiento.
- Cambie de proyecto, voy a hacer un sistema de seguridad para empresas. Lo hice porque no me coordinaba con los contenidos de la materia Empleo Local y Desarrollo Productivo.

  La voy a llamar Librepass

6.1.0. 31/03/2022

- Hoy estuve haciendo pruebas, logre hacer andar el lector usando la tarjeta, incluso pareciera como si la tarjeta leyera más rápido. Estuve usando el código que saqué de instructables. Yo pensé que había quemado el lector, pero me parece que lo que dejó de andar es el tag RFID, desconozco exactamente el por qué, sospecho que dejó de funcionar cuando quise leer/escribir en otros bloques que no sean el 1. Desconozco también el por qué esto arruinaría el tag.
- Encontre una nueva version de la libreria MFRC522, que soporta I2C https://github.com/OSSLibraries/Arduino_MFRC522v2
- Esa libreria no me detecta mi lector, probe con el ejemplo CheckFirmware y DumpInfo, y segun CheckFirmware, no todos los hardwares son soportados
- Probe con la version original de la libreria, y tampoco me detecta mi lector

6.1.0. 01/04/2022

- Creo que se por que no funcionaba, la version nueva de la libreria funciona con ciertos sensores, y no estoy del todo seguro de por qué.

  Ademas, los ejemplos que habia en la documentacion de la nueva no funcionaban en ningun caso, no estoy seguro de por qué.

  Estoy pensando de mantener el pinout de la version que me funciona, y modificar el pinout del codigo de la version v2, pero no creo que me sirva para demasiado, si total la version original ya me funciona para todo lo que quiero hacer.

  Mucho más que leer la UID no necesito para este sistema. Por qué no guardo la info de usuario en la propia tarjeta? Porque es una idea estúpida, es muy facil clonar cualquiera de estas tarjetas, entonces lo que tengo que hacer es un sistema de usuarios y contraseñas. Deberia de ver si hay alguna forma de no guardar la contraseña en plaintext, seguro que hay alguna forma, siempre hay una forma.
- Estaba teniendo un problema con la declaración de la clase Empleado, por alguna razón todos los setters que seteaban Strings no funcionaban. La razón? Me confundí al hacer un copypaste, y llame a todos los setters de la misma forma, void setName() ...
- Al final voy a usar SPI, no pude hacer andar la libreria para que funcione con I2C. Igual al final es mejor, llego a poder experimentar con SPI. En cierta forma me conviene, porque SPI es rapidisimo, y la realidad es que nadie quere estar mas de 1 segundo apoyando la tarjeta para que se la lea.
- Del informe me queda pendiente hacer la seccion del ESP32 de partes-proyecto, hablar un poco de las ventajas y desventajas de SPI, y unas cosas más.

  Pero ahora me voy a poner a programar, quiero diseccionar el codigo que saqué de un tutorial de instructables.
- En muchas partes se habla del PICC, significa Proximity Integrated Circuit Card, es el chip que esta adentro del tag RFID.
- Tambien, PCD significa proximity coupling device.
- Voy a eliminar la funcion writingData(), no me sirve para nada, ya que me voy a manejar todo por el sistema microcontrolado
- Me puse a curiosear (de vuelta) con la version v2 de la libreria MFRC522v2, (que necesita que incluyas wire) y lo hice andar. Nomas tuve que copiar el pinout que aparece en este github https://github.com/OSSLibraries/Arduino_MFRC522v2, pero con RST en el pin 22, y puse SPI SS (o sea, SDA) en el pin 21. Deberia ver que pasa si lo conecto como aparece en el github, pero por ahora, **El dumpinfo está andando**.

- Ahi lo probé con el pinout como el github, y anda jajajaja, que locuras de la vida, es la primera vez que un circuito que armo a las 20:55 funciona, por lo general es al revés, me voy a dormir con los circuitos que no funcionan.
- Bueno, creo que lo decidí, me voy a pasar a la version v2 de la libreria, es más nueva y tiene soporte, no como la otra que está abandonada y solamente acepta pull requests para corregir typos.
- La razon por la que antes no andaba es muy sencilla, creo que solamente conectado un pin, el de SS. O sea, cuando lo armé me parecía muy raro que en código solamente especificara un solo pin, pero como estaba muy quemado pensé que capaz solamente usaba un pin. No tuvo ningun sentido la verdad, son cosas de estar muy quemado de la cabeza

6.1.0. 02/04/2022

- Me puse a testear los codigos de ejemplo del MFRC522v2, parece que el CheckFirmware anda bien ahora que lo conecté bien
- Este issue de github me puede guiar a usar multiples lectores RFID
- La carpeta doc de la libreria MFRC522 me tira documentación muy util
- https://github.com/OSSLibraries/Arduino_MFRC522v2/tree/master/doc
- De todas formas, voy a seguir trabajando con la librería original, por lo menos por ahora
- Me puse a leer el codigo fuente de la libreria original, y creo que entiendo por qué el autor la abandonadó, es un caos, escasean los comentarios y no hay documentación del propósito de cada función.
- Implemente una función para leer el UID, creo que no necesito mucho más.
- Encontre una libreria para poder hacer logging, en vez de Serialprintear a lo imbecil. https://github.com/thijse/Arduino-Log/
- Acabo de hacer una estupidez. Quise sujetar el ESP32 en el protoboard, para que no esté dando vueltas en el aire, con todos los cables dupont. Como no podia meterlo porque tenía cables en un lado, se me ocurrió la maravillosa idea de sujetarlo usando el carril de VCC del protoboard.

  En el momento, me dije a mí mismo " No va a pasar nada, si no tengo nada conectado en el carril de VCC".

  Momentos despues, se apagó el LED del ESP32, y me dí cuenta de mí error, acababa de cortocircuitar los primeros pines de la fila izquierda, acababa de cortocircuitar VIN y GND, quemando así el ESP32.

  Es un sábado a las 8 de la noche, y aora me quedé sin que programar el domingo. Encima voy a tener que comprar uno nuevo, y estan como 1300 pesos.

  Tambien tengo la opcion de ver que es lo que se quemó, pudo haber sido el regulador de tensión AMS 1117.
- Me puse a buscar en internet, y no soy el primer imbecil que cortocircuitó esos dos pines. Parece que hay una solución, y es alimentarlo de forma externa desde VIN, porque lo que deja de funcionar cuando haces lo que yo hice es la alimentación desde cable USB.
- Lo hice andar con alimentación externa

6.1.0. 04/04/2022

- Eventualmente voy a tener que usar FreeRTOS, para usar multiples tasks https://randomnerdtutorials.com/esp32-dual-core-arduino-ide/
- Anduve probando de hacer la lista de empleados con un vector en vez de un array, pero se ve más complejo de usar. Eso no es un problema, pero preferiría dejarlo para el final. Tambien buscando cosas, encontre esta libreria https://github.com/janelia-arduino/Vector, tendría que comparar que ventajas tiene con respecto a https://github.com/mike-matera/ArduinoSTL?utm_source=platformio&utm_medium=piohome, que es un port de la C++ standard library.
- Tuve problemas creando un array de la clase Empleado, me tiraba errores con el constructor, y lo solucioné haciendo sobrecarga de constructores, hice un constructor con parametros, y otro sín parámetros. Tambien para mantener una cuenta de los empleados, hice una variable static en la clase Empleado, llamada cuentaEmpleados, que se modifica en los constructores y destructores.
- Estuve trabajando hasta las 9:30PM en una placa preperforada para no tener que estar con el protoboard, que tiene una calidad deplorable, y tener todo bien organizado. adjuntar imagen

6.1.0. 07/04/2022

- Agregue la licencia al informe, ahora me voy a poner a trabajar en la documentacion del ESP32
- Hice una branch nueva en git, para poder trabajar en esta librería. https://github.com/esphome/ESPAsyncWebServer. La idea sería usar esta librería para poder hacer la configuración

6.1.0. 08/04/2022

- No me anda el constructor, no llegan los parametros. Para mi tiene que ver con el hecho de que estoy teniendo un constructor vacío para que ande el array, voy a probar a laburar sin el array a ver que es lo que pasa.
- Sino lo que se me ocurrió es abandonar el constructor y pasarme a usar los setters directamente
- Me acabo de fijar, me parece que lo que no anda no es el constructor, sino la libreria de logging, no me aparecen las variables
- Ya se por qué no andaba, me olvidé de leer la documentación. Log.info() funciona como printf, hay que declarar el formato de la misma forma que printf lo hace. Entonces, me habia olvidado de los %s. Tambien, dentro del Log.info(çosas a printearÇR), CR significa "\n"
- Hice un merge de la branch dev-webserver hacia la branch dev

6.1.0. 16/04/2022

- Hace un tiempo que no trabajaba en el proyecto y ya me estaba poniendo nervioso, asi que hoy me puse a trabajar en el tema webserver. Para ello, me guié de estas páginas:
  - https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-web-server-arduino-ide/
  - https://www.w3schools.com/XML/ajax_xmlhttprequest_response.asp
  - https://techtutorialsx.com/2018/07/23/esp32-arduino-http-server-template-processing-with-multiple-placeholders/

- En base a esto, logre hacer una pagina web que muestre dos valores, humedad y temperatura (ambos conseguidos usando String(random())) y que se actualizen cada 10 segundos **sin tener que recargar la página.**. Por lo tanto, puedo decir que conseguí un webserver que funciona usando AJAX.

  Lo que habría que hacer es que ande con valores de verdad, como la UID de la tarjeta, y que funcione con eventos, y no intervalos.

- Leyendo un poco más de la documentacion de ESPAsyncWebServer, me doy cuenta que los eventos los tendría que armar en c++, y no en el JavaScript del documento HTML. De todas formas, parece haber un plugin para eventos, https://github.com/me-no-dev/ESPAsyncWebServer#async-event-source-plugin

- Para ir practicando un poco el tema del AJAX, voy a probar poniendo un <p>en la página que me tire la hora. Me voy a ayudar de esta guía https://randomnerdtutorials.com/esp32-date-time-ntp-client-server-arduino/

- Antes de que me olvide, me lo voy a anotar, porque no es la primera vez que cometo este error: processor se llama cuando te conectas a la pagina web, y el JavaScript se llama con los intervals, son dos cosas separadas

- Una nueva cosa en la lista de errores estupidos, haciendo el codigo de la hora, al copypastear parte de sistema de intervals en JavaScript, me olvide de cambiar el valor de getElementById("humedad") por getElementById("hora"), y eso hacía que:

  1. Hora se quedara en "Consiguiendo hora..."
  2. humidity se seteara en 16, y cada vez que la humedad se queria actualizar, tiraba un pequeño flash del otro valor, para inmediatamente cambiar al valor de getHour(). Fue por esto que me di cuenta de que el problema era el valor que puse en getElementById()

- Está el codigo de la hora terminado

6.1.0. 17/04/2022

- Agregue comentarios a todas las funciones, y ordene un poco el código

6.1.0. 18/04/2022

- Estuve trabajando en el informe, actualize toda la seccion del ESP32, agregando la tabla de pines, y las figuras para la version de 30 GPIO y 36 GPIO

6.1.0. 19/04/2022

- Estuve trabajando nuevamente en el informe, agregue informacion sobre el framework arduino

6.1.0. 21/04/2022

- Para usar eventos en el webserver, voy a seguir esta guia: https://randomnerdtutorials.com/esp32-web-server-sent-events-sse/

6.1.0. 28/04/2022

- Tengo andando los eventos, ahora cuando acerco la tarjeta RFID me muestra la UID en la página web.

- Para enviar info desde el cliente hacia el servidor, pense usar esto: https://randomnerdtutorials.com/esp32-esp8266-input-data-html-form/

- Al final, estuve probando en hacerlo sin usar forms, pero tengo que ver como enviar la informacion al servidor con un xmlhttprequest, esta página tiene buenos tutoriales https://www.w3schools.com/xml/ajax_intro.asp

6.1.0. 16/05/2022

- Puedo usar el status code 204 de HTTP, para usar forms

6.1.0. 19/05/2022

- Dividí el codigo en multiples archivos usando headers. Hice un header llamado krapp_utils, en el que metí todas mis funciones.
- Había tenido un problema con la compilacion, y el error estaba en que había metido tanto los archivos .h como los .cpp en el directorio include/, y esto no es la forma correcta. La forma correcta es meter los archivos .h en el include/. y los archivos .cpp en el src/.
- Más adelante tengo pensado hacer un archivo para la clase Empleado
- Lo pude hacer para la clase Empleado, con los archivos empleado.h y empleado.cpp

6.1.0. 30/06/2022

- Este último més estuve estudiando Flask y Bootstrap, con el objetivo de conseguir más experiencia como backend, ya que pensé que para hacer backend con Arduino iba a necesitar más experiencia. Con flask me hice 2 cursos de FreeCodeCamp, y conseguí una buena base de backend.

  Despues me di cuenta de que en realidad, lo que estaba pasando, es que estaba enfocando mal el proyecto. Segun me comentaron varios amigos que trabajan de arquitectura de sistemas, y de backend developer, no puedo guardar toda la info de los usuarios en el ESP32, es una locura, porque se llega a perder el ESP o se llega a estropear, y chau toda la info.

  Entonces decidí cambiar de paradigma. En vez de hacer todo monolítico en el ESP32, decidí hacer las cosas bien, como lo hacen los desarrolladores de IOT, y me voy a pasar a Cloud.

  Para ya tener una buena base, anduve estudiando y haciendo apuntes del Internet Model, de los protocolos TCP/IP, de HTTP y del modelo REST y las API's RESTful.

- Se me acaba de ocurrir una idea buenisima. Puedo hacer una pagina web con flask, que se conecte a AWS y desde ahí actualice la lista de usuarios permitidos. Y aparte, puedo hacer que el ESP32 mande una request a una API REST para ver si el usuario esta autorizado. Entonces, me ahorro el uso de MQTT, es buenisimo.

  En resumen, el ESP32 como un cliente que consulte a una API REST si el usuario esta autorizado, y que la API REST lea de una base de datos que este en AWS. Y que esa base de datos pueda actualizarse usando un front hecho con flask Con esto en base, decidí hacer el cambiazo con AWS, y me voy a pasar a AWS IOT Core

- Voy a empezar con lo sencillo, consultar a una API con el ESP32, para eso voy a usar mockapi https://mockapi.io/. Para poder consultar la API, voy a seguir un tutorial https://techtutorialsx.com/2017/05/19/esp32-http-get-requests/

6.1.0. 01/07/2022

Empece con un curso de AWS

6.1.0. 06/07/2022

- Termine el curso de AWS
- En el curso de AWS, se comenta de un servicio de Database llamado Aurora Serverless, que solamente corre cuando lo necesito. Es como AWS Lambda, pero para bases de datos. Creo que es lo que voy a usar, pero voy a tener que investigar más con respecto a precio. Es más barato que Aurora normal, y resulta más eficiente en cuanto a precio.
- Para usar algo como Heroku, puedo usar Elastic Beanstalk (que manía rara que tiene AWS de poner nombres raros) Es un servicio para hacer deploy y escalamiento de aplicaciones webs y servicios. El tema es que no esta tan pensado para production.
- Tambien podria directamente usar Lambda serverless functions.
- Podria usar AWS Artifact, para generar un informe de que AWS tiene global compliance con un monton de organizaciones, ta bueno para el informe.
- Para la parte de costos y presupuestos, el curso tiene toda una parte explicando todos los servicios que hay. Hay algunos pagos, y otros gratis. Esta bueno

6.1.0. 30/09/2022

- A dia de hoy, ya tengo una webapp funcionando con Flask como backend, y el frontend hecho con los templates de Jinja2 y con Bootstrap. Esta webapp tiene una API REST la cual permite registrar los accesos del cliente ESP32, y estos accesos se guardan en una base de datos PostgreSQL. Los mismos se pueden ver en las distintas pestañas de la webapp. Lo proximo es trabajar en el cliente ESP32 y el resto de cosas que quedan.

6. Referencias

[1] RFID 4u. *RFID Regulations*. URL: https://rfid4u.com/rfid-regulations/.

[2] Code Academy. *What is REST?* URL: https://www.codecademy.com/article/what-is-rest.

[3] Amazon. *AWS IoT Core*. URL: https://aws.amazon.com/iot-core/.

[4] Amazon. *Getting started with AWS IoT Core*. URL: https://docs.aws.amazon.com/iot/latest/developerguide/iot-gs.html.

[5] Amazon. *What is cloud computing?* URL: https://aws.amazon.com/what-is-cloud-computing/.

[6] Corelis. *SPI Tutorial*. URL: https://www.corelis.com/education/tutorials/spi-tutorial/.

[7] Zerynth docs. *DOIT Esp32 DevKit v1 reference*. URL: https://testzdoc.zerynth.com/reference/boards/doit_esp32/docs/.

[8] Fireship. *Top 50+ AWS Services Explained in 10 Minutes*. URL: https://www.youtube.com/watch?v=JIbIYCM48to.

[9] Arduino Foundation. *A Brief Introduction to the Serial Peripheral Interface*. URL: https://www.arduino.cc/en/reference/SPI.

[10] Arduino Foundation. *Language Reference*. URL: https://www.arduino.cc/reference/en/.

[11] Free Software Foundation. *What is Free Software*. URL: https://www.fsf.org/about/what-is-free-software.

[12] Wikimedia Foundation. *Radio-frequency identification*. URL: https://en.wikipedia.org/wiki/Radio-frequency_identification.

[13] Wikimedia Foundation. *RFID*. URL: https://es.wikipedia.org/wiki/RFID.

[14] Wikimedia Foundation. *Serial Peripheral Interface*. URL: https://en.wikipedia.org/wiki/Serial_Peripheral_Interface.

[15] Mike Grusin. *Serial Peripheral Interface (SPI)*. URL: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/.

[16] ESP32 IO. *ESP32 RFID/NFC*. URL: https://esp32io.com/tutorials/esp32-rfid-nfc.

[17] Fernando Koyanagi. *ESP32 With RFID: Access Control*. URL: https://www.instructables.com/ESP32-With-RFID-Access-Control/.

[18] Exostiv Labs. *Introduction to I2C and SPI Protocols*. URL: https://www.exostivlabs.com/files/documents/Introduction-to-I2C-and-SPI-Protocols.pdf?/article/aa-00255/22/introduction-to-spi-and-ic-protocols.html.

[19] Rus Shuler. *How Does the Internet Work?* URL: https://web.stanford.edu/class/msande91si/www-spr04/readings/week1/InternetWhitepaper.htm.

[20] Arduino Get Started. *Arduino RFID/NFC*. URL: https://arduinogetstarted.com/tutorials/arduino-rfid-nfc.

[21] Arduino Get Started. *How to connect multiple spi sensors/devices with Arduino?* URL: https://arduinogetstarted.com/faq/how-to-connect-multiple-spi-sensors-devices-with-arduino.

[22] GNU Operating System. *¿Qué es el Software Libre?* URL: https://www.gnu.org/philosophy/free-sw.es.html.

[23] Massachusetts Institute of Technology. *The MIT License*. URL: https://mit-license.org/.

[24] techtutorialsx. *ESP32: HTTP GET Requests*. URL: https://techtutorialsx.com/2017/05/19/esp32-http-get-requests/.

[25] Random Nerd Tutorials. *DHT Sensor Web Server*. URL: https://randomnerdtutorials.com/esp32-dht11-dht22-temperature-humidity-web-server-arduino-ide/.

[26] Random Nerd Tutorials. *ESP32 Pinout Reference: Which GPIO pins should you use?* URL: https://randomnerdtutorials.com/esp32-pinout-reference-gpios/.

[27] Random Nerd Tutorials. *ESP32 Web Server using Server-Sent Events (Update Sensor Readings Automatically).* URL: https://randomnerdtutorials.com/esp32-web-server-sent-events-sse/.

[28] Random Nerd Tutorials. *Getting Started with the ESP32 Development Board.* URL: https://randomnerdtutorials.com/getting-started-with-esp32/.

[29] Random Nerd Tutorials. *How to use ESP32 Dual Core with Arduino IDE.* URL: https://randomnerdtutorials.com/esp32-dual-core-arduino-ide/.

[30] Random Nerd Tutorials. *Security Access using MFRC522 RFID Reader with Arduino.* URL: https://randomnerdtutorials.com/security-access-using-mfrc522-rfid-reader-with-arduino/.

[31] Tech TutorialsX. *ESP32 Arduino HTTP Server: Template processing with multiple placeholders.* URL: https://techtutorialsx.com/2018/07/23/esp32-arduino-http-server-template-processing-with-multiple-placeholders/.

[32] W3Schools. *AJAX - Server Response.* URL: https://www.w3schools.com/XML/ajax_xmlhttprequest_response.asp.