Librepass

Informe técnico

Un trabajo presentado para la materia de
Proyectos y Diseño Electrónico

Krapp Ramiro

Instituto tecnológico San Bonifacio
Departamento de electrónica
1 de abril de 2022

Hecho en LaTeX
Versión Alpha 0.1

Índice

1. Introducción **2**

2. Partes del proyecto **3**
2.1. El protocolo de comunicación SPI . . . 3
2.1.1. Ventajas y desventajas . . . 5
2.2. DOIT ESP32 DevKit v1 . . . 6
2.3. RFID . . . 7
2.3.1. Pinout del dispositivo . . . 8
2.3.2. Mapeo de Memoria . . . 9

3. Diagrama esquematico **10**

4. Codigo del programa **11**

5. Bitacoras Personales **16**
5.1. Krapp Ramiro . . . 16
5.1.1. 24/03/2022 . . . 16
5.1.2. 25/03/2022 . . . 16
5.1.3. 26/03/2022 . . . 16
5.1.4. 27/03/2022 . . . 17
5.1.5. 28/03/2022 . . . 17
5.1.6. 29/03/2022 . . . 17
5.1.7. 30/03/2022 . . . 17
5.1.8. 31/03/2022 . . . 17
5.1.9. 01/04/2022 . . . 18

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

Tengo un Repositorio en GitHub
https://github.com/KrappRamiro/librepass

Introducción

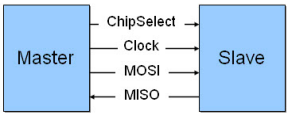
Librepass es un sistema Free and Open Source de seguridad para empresas Fue desarrollado usando una placa de desarrollo DOIT ESP32 DevKit V1, conectado a un array de lectores RFID-RC552.

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.

Partes del proyecto

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)

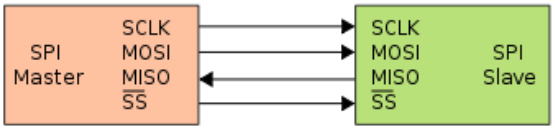


Figura 1: SPI master conectado a un único slave.

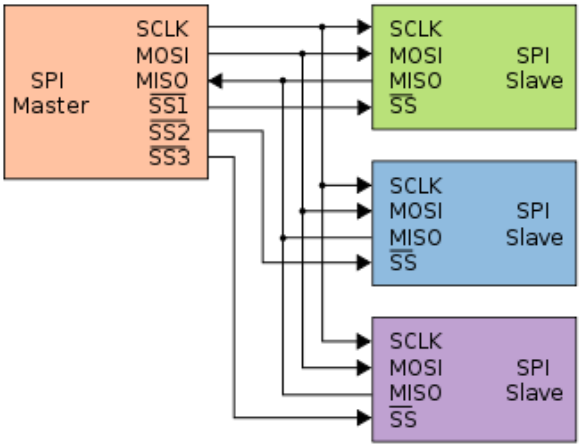


Figura 2: SPI master conectado a múltiples slaves.

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

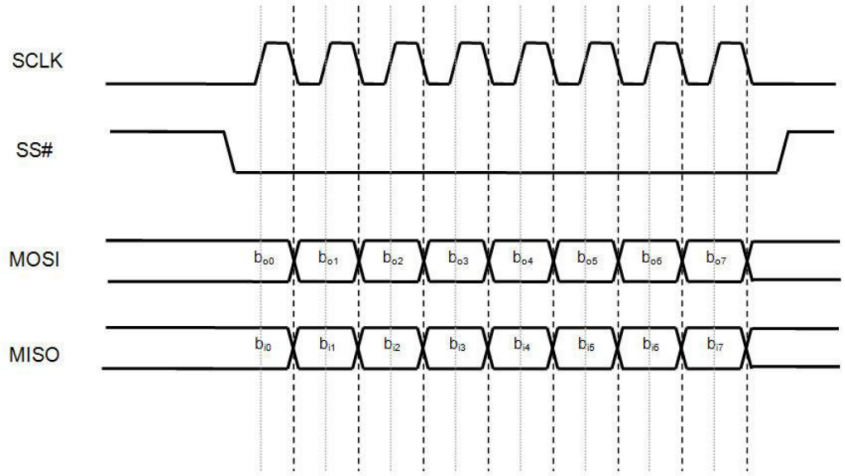


Figura 3: El timing de una comunicación SPI. En este ejemplo, La transmisión de datos por los canales MOSI y MISO es ejecutada por cada flanco descendente en la señal de clock en SCLK. En cambio, la lectura de datos es ejecutada por cada flanco ascendente. Esto se puede cambiar modificando el SPI mode

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)

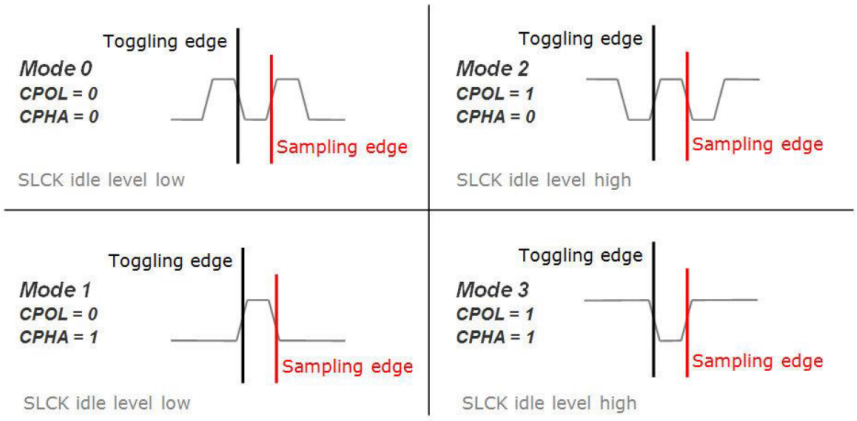


Figura 4: Los modos SPI son definidos con los parámetros CPOL (clock polarity) y CPHA (clock phase), que definen 3 parámetros: El flanco usado para envío de datos, el flanco usado para recepción de datos, y el estado de inactividad (idle) de SCLK

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.

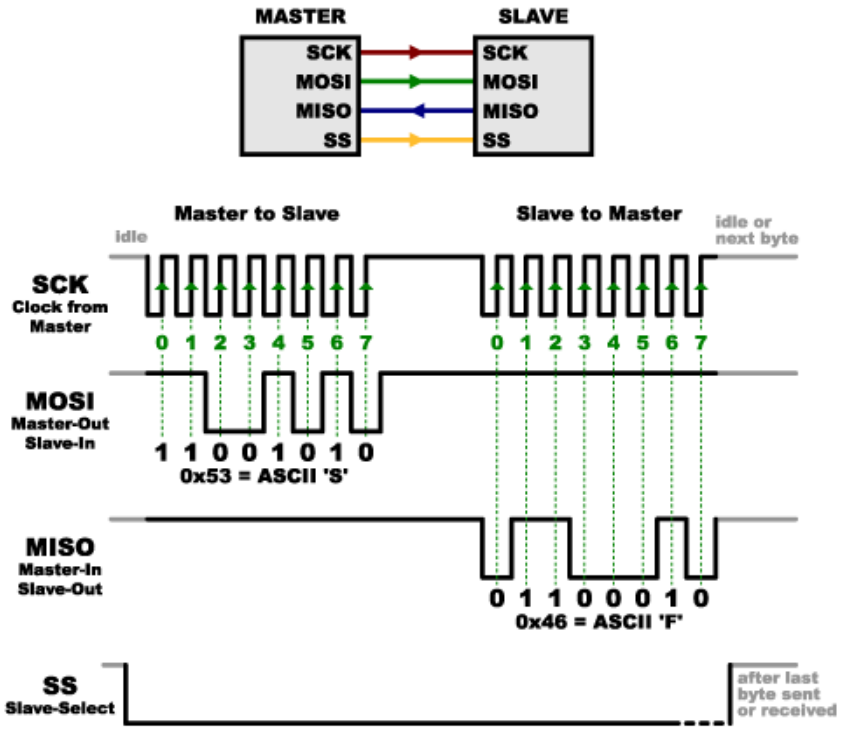


Figura 5: Grafico de comunicacion SPI

Ventajas y desventajas

ventajas y desventajas

DOIT ESP32 DevKit v1

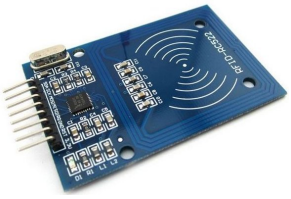
El kit de desarrollo DOIT ESP32 DevKit v1

RFID

Segun Wikipedia[5]:

"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 6: Distintos tags RFID

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**

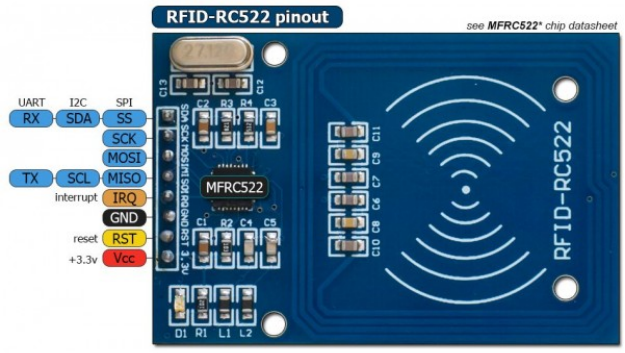


Figura 7: El pinout del lector RFID-RC552. Se puede notar como este dispositivo está adaptado para funcionar con 3 protocolos distintos, comunicación por UART, comunicación por I2C y comunicacion por SPI

Mapeo de Memoria

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

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

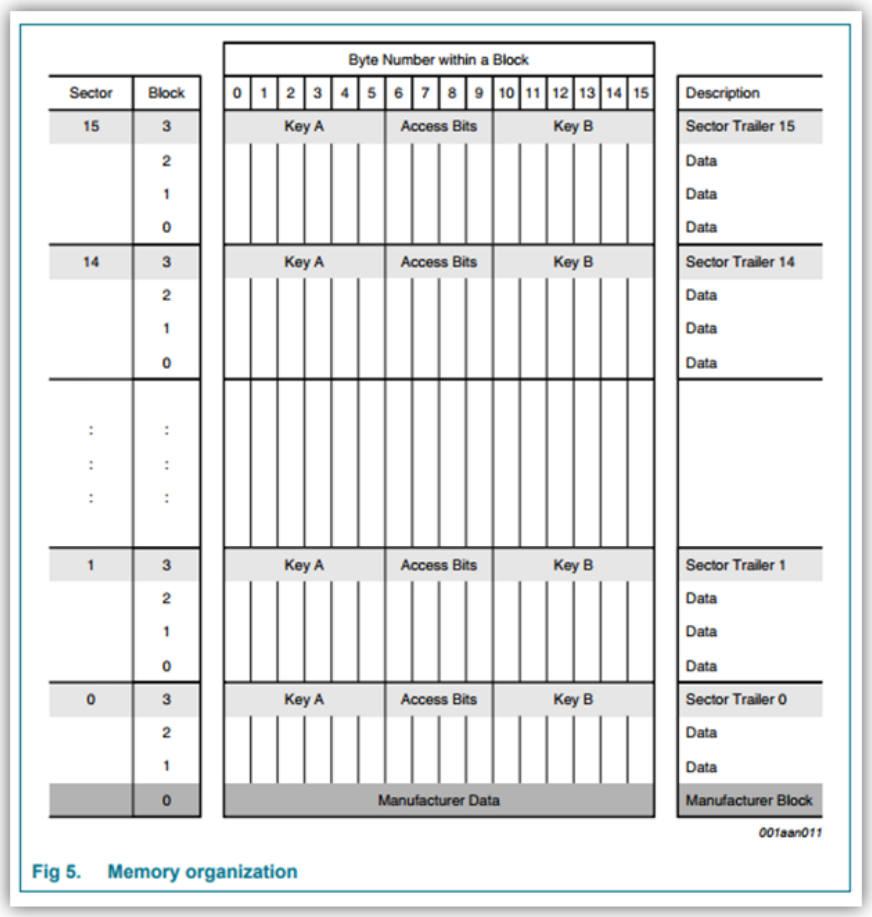
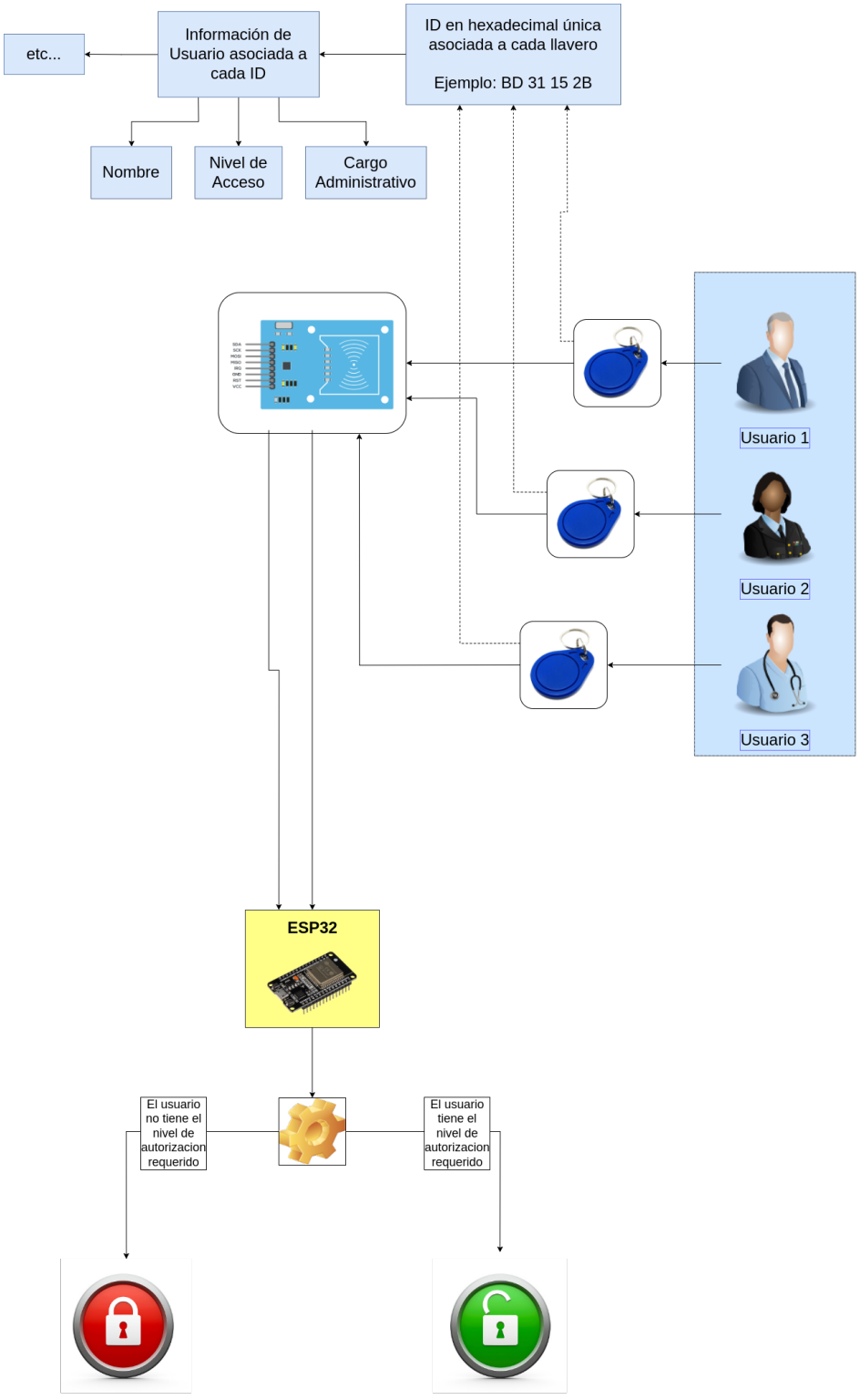


Fig 5. Memory organization

Diagrama esquematico



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 <Arduino.h>
#include <iostream>
#include <vector>

/* Se usa std::vector en reemplazo de usar `using namespace std` por una muy
buena razon, y es que se evita el namespace pollution. Si no sabes qué es eso,
te recomiendo personalmente este post,  es corto, sencillo, y bien explicado
para principiantes:
https://www.thecrazyprogrammer.com/2021/01/better-alternatives-for-using-namespace-std-in-c.html
*/
using std::vector;

/*
   Si alguien se pregunta por qué, en las clases, las variables estan en private,
   la respuesta es muy sencilla:
   Es porque no se desea que se modifiquen las variables de forma manual.
   Esto es porque esa práctica es propensa a errores, ya que se podría introducir
   un valor inadecuado y generar algun problema.

   Por eso se usan funciones public, normalmente llamadas setters, que permiten
   asignar y leer los valores, y que establecen un margen de valores seguros.
*/

/*
class Empleado {
private:
   String name;
   bool isAlive = true;
   String dni;
   int clearanceLevel;
   String cargoAdministrativo;

public:
   void setLifeStatus(bool lifeStatus)
   {
      this->isAlive = lifeStatus;
   }
   void setName(String name)
   {
      this->name = name;
   }
   void setName(String dni)
   {
      this->dni = dni;
   }
   void setName(int clearanceLevel)
   {
```

```
51          this->clearanceLevel = clearanceLevel;
52       }
53       void setName(String cargoAdministrativo)
54       {
55          this->cargoAdministrativo = cargoAdministrativo;
56       }
57    };
58    */
59
60    #include <MFRC522.h> //library responsible for communicating with the module RFID-RC522
61    #include <SPI.h> //library responsible for communicating of SPI bus
62    #define SS_PIN 21
63    #define RST_PIN 22
64    #define SIZE_BUFFER 18
65    #define MAX_SIZE_BLOCK 16
66    #define greenPin 12
67    #define redPin 32
68    // used in authentication
69    MFRC522::MIFARE_Key key;
70    // authentication return status code
71    MFRC522::StatusCode status;
72    // Defined pins to module RC522
73    MFRC522 mfrc522(SS_PIN, RST_PIN);
74
75    // reads data from card/tag
76    void readingData()
77    {
78       // prints the technical details of the card/tag
79       mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid));
80
81       // prepare the key - all keys are set to FFFFFFFFFFFFh
82       for (byte i = 0; i < 6; i++)
83          key.keyByte[i] = 0xFF;
84
85       // buffer for read data
86       byte buffer[SIZE_BUFFER] = { 0 };
87
88       // the block to operate
89       byte block = 1;
90       byte size = SIZE_BUFFER; // authenticates the block to operate
91       status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key, &(mfrc522.uid)); // line
    ↪  834 of MFRC522.cpp file
92       if (status != MFRC522::STATUS_OK) {
93          Serial.print(F("Authentication failed: "));
94          Serial.println(mfrc522.GetStatusCodeName(status));
95          digitalWrite(redPin, HIGH);
96          delay(1000);
97          digitalWrite(redPin, LOW);
98          return;
99       }
100
101      // read data from block
102      status = mfrc522.MIFARE_Read(block, buffer, &size);
103      if (status != MFRC522::STATUS_OK) {
104         Serial.print(F("Reading failed: "));
105         Serial.println(mfrc522.GetStatusCodeName(status));
106         digitalWrite(redPin, HIGH);
107         delay(1000);
108         digitalWrite(redPin, LOW);
109         return;
110      } else {
111         digitalWrite(greenPin, HIGH);
112         delay(1000);
113         digitalWrite(greenPin, LOW);
114      }
115
116      Serial.print(F("\nData from block ["));
117      Serial.print(block);
```

```
118     Serial.print(F("]: "));
119
120     // prints read data
121     for (uint8_t i = 0; i < MAX_SIZE_BLOCK; i++) {
122        Serial.write(buffer[i]);
123     }
124     Serial.println(" ");
125  }
126
127  void writingData()
128  {
129
130     // prints thecnical details from of the card/tag
131     mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid));
132
133     // waits 30 seconds dor data entry via Serial
134     Serial.setTimeout(30000L);
135     Serial.println(F("Enter the data to be written with the '#' character at the end \n[maximum of 16
    ↪  characters]:"));
136
137     // prepare the key - all keys are set to FFFFFFFFFFFFh
138     for (byte i = 0; i < 6; i++)
139        key.keyByte[i] = 0xFF;
140
141     // buffer para armazenamento dos dados que iremos gravar
142     // buffer for storing data to write
143     byte buffer[MAX_SIZE_BLOCK] = "";
144     byte block; // the block to operate
145     byte dataSize; // size of data (bytes)
146
147     // recover on buffer the data from Serial
148     // all characters before chacactere '#'
149     dataSize = Serial.readBytesUntil('#', (char*)buffer, MAX_SIZE_BLOCK);
150     // void positions that are left in the buffer will be filled with whitespace
151     for (byte i = dataSize; i < MAX_SIZE_BLOCK; i++) {
152        buffer[i] = ' ';
153     }
154
155     block = 1; // the block to operate
156     String str = (char*)buffer; // transforms the buffer data in String
157     Serial.println(str);
158
159     // authenticates the block to operate
160     // Authenticate is a command to hability a secure communication
161     status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
162        block, &key, &(mfrc522.uid));
163
164     if (status != MFRC522::STATUS_OK) {
165        Serial.print(F("PCD_Authenticate() failed: "));
166        Serial.println(mfrc522.GetStatusCodeName(status));
167        digitalWrite(redPin, HIGH);
168        delay(1000);
169        digitalWrite(redPin, LOW);
170        return;
171     }
172     // else Serial.println(F("PCD_Authenticate() success: "));
173
174     // Writes in the block
175     status = mfrc522.MIFARE_Write(block, buffer, MAX_SIZE_BLOCK);
176     if (status != MFRC522::STATUS_OK) {
177        Serial.print(F("MIFARE_Write() failed: "));
178        Serial.println(mfrc522.GetStatusCodeName(status));
179        digitalWrite(redPin, HIGH);
180        delay(1000);
181        digitalWrite(redPin, LOW);
182        return;
183     } else {
184        Serial.println(F("MIFARE_Write() success: "));
```

```
185         digitalWrite(greenPin, HIGH);
186         delay(1000);
187         digitalWrite(greenPin, LOW);
188      }
189   }
190
191   // menu to operation choice
192   int menu()
193   {
194      Serial.println(F("\nChoose an option:"));
195      Serial.println(F("0 - Reading data"));
196      Serial.println(F("1 - Writing data\n"));
197
198      // waits while the user does not start data
199      while (!Serial.available()) { };
200
201      // retrieves the chosen option
202      int op = (int)Serial.read();
203
204      // remove all characters after option (as \n per example)
205      while (Serial.available()) {
206         if (Serial.read() == '\n')
207            break;
208         Serial.read();
209      }
210      return (op - 48); // subtract 48 from read value, 48 is the zero from ascii table
211   }
212
213   void setup()
214   {
215      Serial.begin(9600);
216      SPI.begin(); // Init SPI bus
217      pinMode(greenPin, OUTPUT);
218      pinMode(redPin, OUTPUT);
219
220      digitalWrite(greenPin, HIGH);
221      digitalWrite(redPin, HIGH);
222      delay(500);
223      digitalWrite(greenPin, LOW);
224      digitalWrite(redPin, LOW);
225
226      // Init MFRC522
227      mfrc522.PCD_Init();
228      Serial.println("Approach your reader card...");
229      Serial.println();
230   }
231
232   void loop()
233   {
234      // Aguarda a aproximacao do cartao
235      // waiting the card approach
236      if (!mfrc522.PICC_IsNewCardPresent()) {
237         return;
238      }
239      // Select a card
240      if (!mfrc522.PICC_ReadCardSerial()) {
241         return;
242      }
243
244      // Dump debug info about the card; PICC_HaltA() is automatically called
245      // mfrc522.PICC_DumpToSerial(&(mfrc522.uid));</p><p> //call menu function and retrieve the desired option
246      int op = menu();
247
248      if (op == 0)
249         readingData();
250      else if (op == 1)
251         writingData();
252      else {
```

```
253         Serial.println(F("Incorrect Option!"));
254         return;
255     }
256 
257     // instructs the PICC when in the ACTIVE state to go to a "STOP" state
258     mfrc522.PICC_HaltA();
259     // "stop" the encryption of the PCD, it must be called after communication with authentication, otherwise
    ↪  new communications can not be initiated
260     mfrc522.PCD_StopCrypto1();
261 }
```

Bitacoras Personales

Krapp Ramiro

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.

25/03/2022

Hacer las urls mas chicas con o tiny

- 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

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 sensor RFID-RC552

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.

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/

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/
- tambien 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

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

31/03/2022

- Encontre una nueva version de la libreria MFRC552, 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 sensor

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.

Referencias

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

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

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

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

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

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

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

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

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

[10] 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.

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

[12] 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.

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

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

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