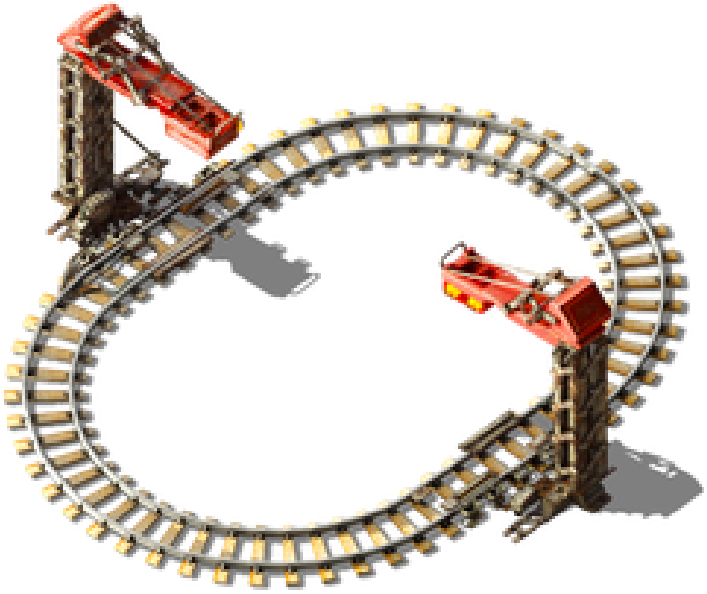
Traintorio

Informe técnico

Un trabajo presentado para la materia de
Proyectos y Diseño Electrónico



Krapp Ramiro

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

Hecho en LaTeX
Versión Alpha 0.1

Índice

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/traintorio

Introducción

Traintorio es un modelo del sistema de transporte público ferroviario de la provincia de Buenos Aires, diseñado para capacitar a alumnos de primaria, para que aprendan a transportarse emulando el sistema de tarjeta SUBE / molinillo.

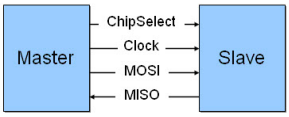
Fue desarrollado usando una placa de desarrollo DOIT ESP32 DevKit V1, conectado a un array de sensores RFID-RC552. Estos sensores intentan emular el molinillo de las estaciones de trenes, y para la emulación de la tarjeta sube, se usan unas tarjetas y/o llaveros con un código hexadecimal indentificador.

creo que la palabra no es modelo! es otra que saravia usa, que dice que significa que es una version horrosrosa de la realidad, despues le tengo que preguntar que palabra era.

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)

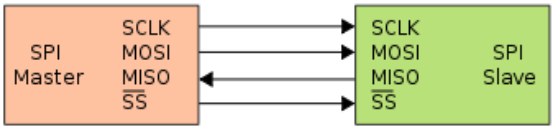


Figura 1: SPI master conectado a un único slave.

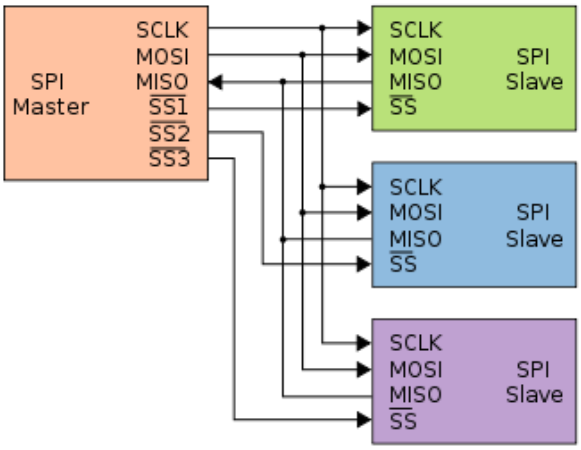


Figura 2: SPI master conectado a múltiples slaves.

no olvidarse de la daisy chained

poner las ventajas y desventajas que aparecen en wikipedia

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

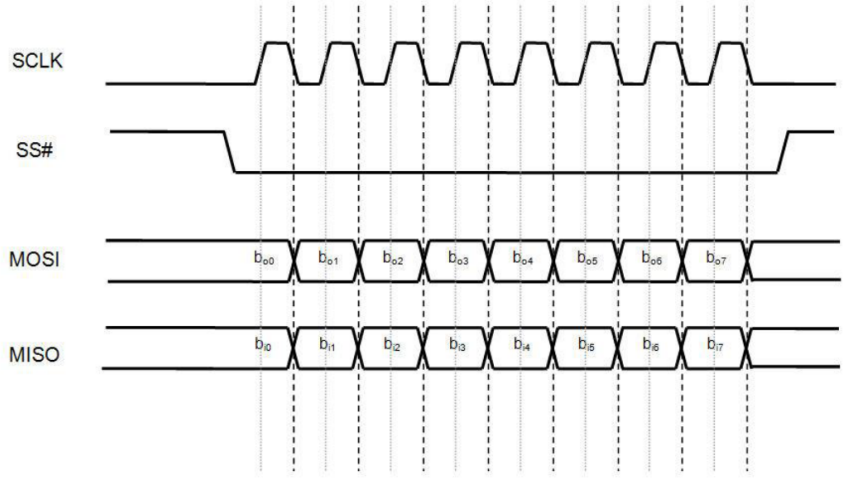


Figura 3: El timing de una comunicación SPI. En este ejemplo, La transmisión de datos por los canales MOSI y MISO es ejecutada por cada flanco descendente en la señal de clock en SCLK. En cambio, la lectura de datos es ejecutada por cada flanco ascendente. Esto se puede cambiar modificando el SPI mode

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)

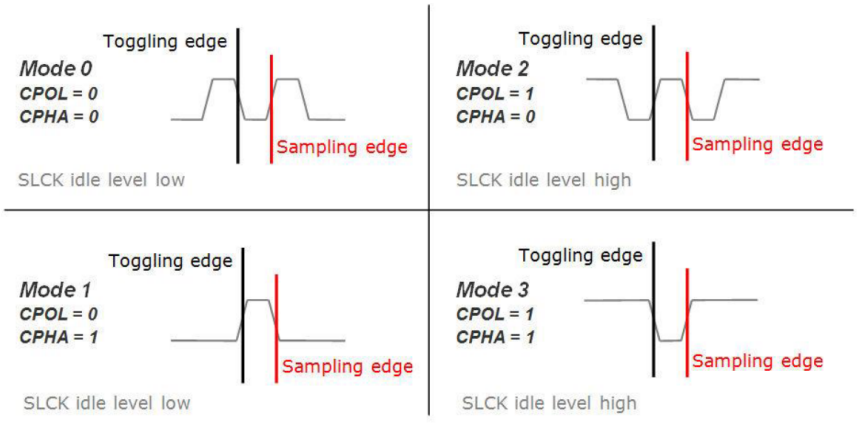


Figura 4: Los modos SPI son definidos con los parámetros CPOL (clock polarity) y CPHA (clock phase), que definen 3 parámetros: El flanco usado para envío de datos, el flanco usado para recepción de datos, y el estado de inactividad (idle) de SCLK

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.

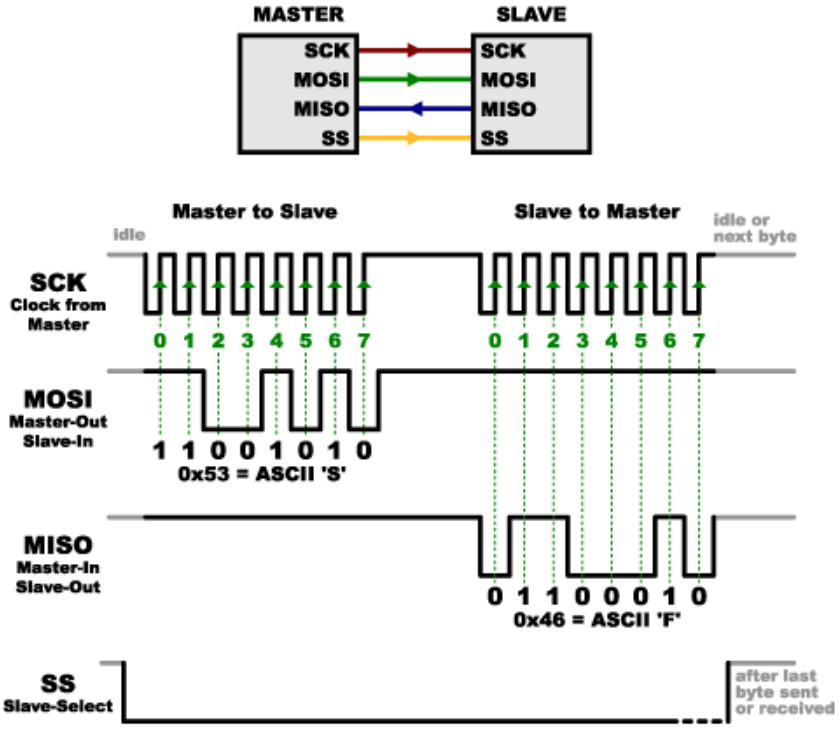
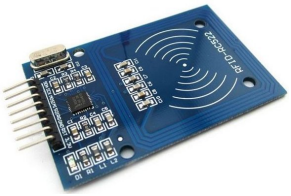


Figura 5: Grafico de comunicacion SPI

Ventajas y desventajas

ventajas y desventajas

RFID



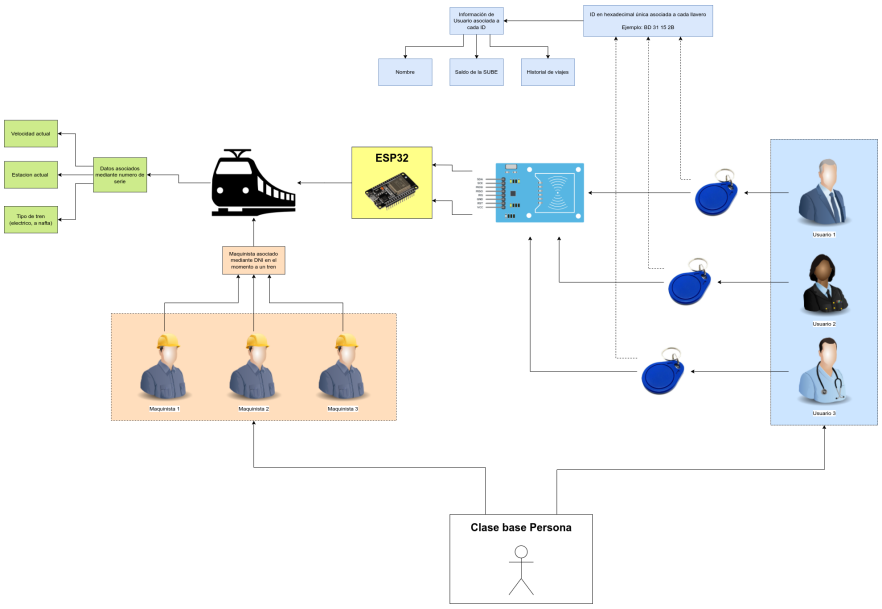
Segun Wikipedia[5], RFID o identificación por radiofrecuencia (del inglés Radio Frequency Identification) es un sistema de almacenamiento y recuperación de datos remotos que usa dispositivos denominados etiquetas, tarjetas o transpondedores RFID. El propósito fundamental de la tecnología RFID es transmitir 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.

Diagrama esquematico

El diagrama esquematico fue generado usando drawio



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

```
/*
TODO Hacer el registro de viajes de cada pasajero

*/

#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;

class Tren {
   /* Si alguien se pregunta por qué 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.  */
private:
   int speed = 0; // velocidad, en km/h
   String serialNumber; // numero de serie, que va a identificar al tren
   String currentStation;
   String trainType; // esta var se refiere si es a nafta, si es electrico, etc

public:
   Tren(String serialNumber, String trainType)
   {
      this->serialNumber = serialNumber;
      this->trainType = trainType;
   }
   // Para los getters tenia dos opciones, o retornaba un struct, o hacia una funcion
   // para cada variable
   int getSpeed()
   {
      return speed;
   }
   String getSerialNumber()
   {
      return serialNumber;
   }
   String getCurrentStation()
   {
```

```
51          return currentStation;
52      }
53      String getTrainType()
54      {
55          return trainType;
56      }
57
58      void travelToStation(String stationName)
59      {
60          currentStation = stationName;
61          // TODO Hacer algo parecido con la funcion que tenes en Pasajero
62      }
63  };
64
65  class Persona {
66      // Esta clase sirve como padre para las clases Maquinista y Pasajero
67      // IDEA: Hacer que las personas puedan morir, y que se invalide la SUBE.
68      // Por ejemplo,         if (!persona.isAlive) {allowTransaction(false)}
69  private:
70      String name;
71      bool isAlive = true;
72      String dni;
73
74  public:
75      void kill()
76      {
77          isAlive = false;
78      }
79  };
80
81  class Maquinista : public Persona { // clase que hereda de Persona
82  private:
83      String name;
84      float salary;
85      int seniority; // el seniority se piensa con los años de antiguedad
86  public:
87  };
88
89  class Pasajero : public Persona { // clase que hereda de Persona
90  private:
91      String nombre;
92      int sube_id;
93      float sube_saldo;
94
95  public:
96      void travelToStation(String stationName)
97      {
98          // TODO hay que hacer la transaccion
99
100         /*
101         Como deberia ser esto? tendria que ser así:
102         1- Calcular distancia a la estacion
103         2- Cobrar 5 pesos por cada estacion
104
105         Para calcular la estación, lo que haría sería armar un vector de
106         estaciones, algo asi:
107
108         ["temperley", "lomas de zamora", "banfield", "remedios de escalada", "etc"]
109
110         1 - Llamar a una funcion getCurrentStation() que retorne un String
111         de la estacion actual
112         2 - sabiendo la estacion actual, se podría hacer un getIndex()
113         https://www.geeksforgeeks.org/how-to-find-index-of-a-given-element-in-a-vector-in-cpp/
114         Entonces se haria un getIndex(estacionActual) - getIndex(estacionDestino),
115         y el resultado de esa operacion es la distancia entre las estaciones.
116
117         OJO: Esa operacion puede dar resultados negativos, por eso habria que guardarlo
118         en una variable, checkear si es negativa, y en ese caso pasarla a positivo
```

```
119
120         3 - Llamar a la funcion calcularPasaje(int price_per_estation, int distance)
121
122
123         */
124     }
125 };
126
127 #include <MFRC522.h> //library responsible for communicating with the module RFID-RC522
128 #include <SPI.h> //library responsible for communicating of SPI bus
129 #define SS_PIN 21
130 #define RST_PIN 22
131 #define SIZE_BUFFER 18
132 #define MAX_SIZE_BLOCK 16
133 #define greenPin 12
134 #define redPin 32
135 // used in authentication
136 MFRC522::MIFARE_Key key;
137 // authentication return status code
138 MFRC522::StatusCode status;
139 // Defined pins to module RC522
140 MFRC522 mfrc522(SS_PIN, RST_PIN);
141
142 // reads data from card/tag
143 void readingData()
144 {
145     // prints the technical details of the card/tag
146     mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid));
147
148     // prepare the key - all keys are set to FFFFFFFFFFFFh
149     for (byte i = 0; i < 6; i++)
150         key.keyByte[i] = 0xFF;
151
152     // buffer for read data
153     byte buffer[SIZE_BUFFER] = { 0 };
154
155     // the block to operate
156     byte block = 1;
157     byte size = SIZE_BUFFER; // authenticates the block to operate
158     status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key, &(mfrc522.uid)); // line
    ↪  834 of MFRC522.cpp file
159     if (status != MFRC522::STATUS_OK) {
160         Serial.print(F("Authentication failed: "));
161         Serial.println(mfrc522.GetStatusCodeName(status));
162         digitalWrite(redPin, HIGH);
163         delay(1000);
164         digitalWrite(redPin, LOW);
165         return;
166     }
167
168     // read data from block
169     status = mfrc522.MIFARE_Read(block, buffer, &size);
170     if (status != MFRC522::STATUS_OK) {
171         Serial.print(F("Reading failed: "));
172         Serial.println(mfrc522.GetStatusCodeName(status));
173         digitalWrite(redPin, HIGH);
174         delay(1000);
175         digitalWrite(redPin, LOW);
176         return;
177     } else {
178         digitalWrite(greenPin, HIGH);
179         delay(1000);
180         digitalWrite(greenPin, LOW);
181     }
182
183     Serial.print(F("\nData from block ["));
184     Serial.print(block);
185     Serial.print(F("]: "));
```

```
186
187     // prints read data
188     for (uint8_t i = 0; i < MAX_SIZE_BLOCK; i++) {
189        Serial.write(buffer[i]);
190     }
191     Serial.println(" ");
192  }
193
194  void writingData()
195  {
196
197     // prints thecnical details from of the card/tag
198     mfrc522.PICC_DumpDetailsToSerial(&(mfrc522.uid));
199
200     // waits 30 seconds dor data entry via Serial
201     Serial.setTimeout(30000L);
202     Serial.println(F("Enter the data to be written with the '#' character at the end \n[maximum of 16
↪     characters]:"));
203
204     // prepare the key - all keys are set to FFFFFFFFFFFFh
205     for (byte i = 0; i < 6; i++)
206        key.keyByte[i] = 0xFF;
207
208     // buffer para armazenamento dos dados que iremos gravar
209     // buffer for storing data to write
210     byte buffer[MAX_SIZE_BLOCK] = "";
211     byte block; // the block to operate
212     byte dataSize; // size of data (bytes)
213
214     // recover on buffer the data from Serial
215     // all characters before chacactere '#'
216     dataSize = Serial.readBytesUntil('#', (char*)buffer, MAX_SIZE_BLOCK);
217     // void positions that are left in the buffer will be filled with whitespace
218     for (byte i = dataSize; i < MAX_SIZE_BLOCK; i++) {
219        buffer[i] = ' ';
220     }
221
222     block = 1; // the block to operate
223     String str = (char*)buffer; // transforms the buffer data in String
224     Serial.println(str);
225
226     // authenticates the block to operate
227     // Authenticate is a command to hability a secure communication
228     status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A,
229        block, &key, &(mfrc522.uid));
230
231     if (status != MFRC522::STATUS_OK) {
232        Serial.print(F("PCD_Authenticate() failed: "));
233        Serial.println(mfrc522.GetStatusCodeName(status));
234        digitalWrite(redPin, HIGH);
235        delay(1000);
236        digitalWrite(redPin, LOW);
237        return;
238     }
239     // else Serial.println(F("PCD_Authenticate() success: "));
240
241     // Writes in the block
242     status = mfrc522.MIFARE_Write(block, buffer, MAX_SIZE_BLOCK);
243     if (status != MFRC522::STATUS_OK) {
244        Serial.print(F("MIFARE_Write() failed: "));
245        Serial.println(mfrc522.GetStatusCodeName(status));
246        digitalWrite(redPin, HIGH);
247        delay(1000);
248        digitalWrite(redPin, LOW);
249        return;
250     } else {
251        Serial.println(F("MIFARE_Write() success: "));
252        digitalWrite(greenPin, HIGH);
```

```
253         delay(1000);
254         digitalWrite(greenPin, LOW);
255      }
256   }
257
258   // menu to operation choice
259   int menu()
260   {
261      Serial.println(F("\nChoose an option:"));
262      Serial.println(F("0 - Reading data"));
263      Serial.println(F("1 - Writing data\n"));
264
265      // waits while the user does not start data
266      while (!Serial.available()) { };
267
268      // retrieves the chosen option
269      int op = (int)Serial.read();
270
271      // remove all characters after option (as \n per example)
272      while (Serial.available()) {
273         if (Serial.read() == '\n')
274            break;
275         Serial.read();
276      }
277      return (op - 48); // subtract 48 from read value, 48 is the zero from ascii table
278   }
279
280   void setup()
281   {
282      Serial.begin(9600);
283      SPI.begin(); // Init SPI bus
284      pinMode(greenPin, OUTPUT);
285      pinMode(redPin, OUTPUT);
286
287      digitalWrite(greenPin, HIGH);
288      digitalWrite(redPin, HIGH);
289      delay(500);
290      digitalWrite(greenPin, LOW);
291      digitalWrite(redPin, LOW);
292
293      // Init MFRC522
294      mfrc522.PCD_Init();
295      Serial.println("Approach your reader card...");
296      Serial.println();
297   }
298
299   void loop()
300   {
301      // Aguarda a aproximacao do cartao
302      // waiting the card approach
303      if (!mfrc522.PICC_IsNewCardPresent()) {
304         return;
305      }
306      // Select a card
307      if (!mfrc522.PICC_ReadCardSerial()) {
308         return;
309      }
310
311      // Dump debug info about the card; PICC_HaltA() is automatically called
312      // mfrc522.PICC_DumpToSerial(&(mfrc522.uid));</p><p> //call menu function and retrieve the desired option
313      int op = menu();
314
315      if (op == 0)
316         readingData();
317      else if (op == 1)
318         writingData();
319      else {
320         Serial.println(F("Incorrect Option!"));
```

```
321         return;
322     }
323 
324     // instructs the PICC when in the ACTIVE state to go to a "STOP" state
325     mfrc522.PICC_HaltA();
326     // "stop" the encryption of the PCD, it must be called after communication with authentication, otherwise
↪  new communications can not be initiated
327     mfrc522.PCD_StopCrypto1();
328 }
```

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.

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/.