Universitatea
Transilvania
din Brașov
FACULTATEA DE INGINERIE ELECTRICĂ
ȘI ȘTIINȚA CALCULATOARELOR

INFORMATICA APLICATA II

Sistem Antifurt – Proiect

“SecureTag”

(Documentatie)

Realizat de: Lupsa Eduard-Daniel

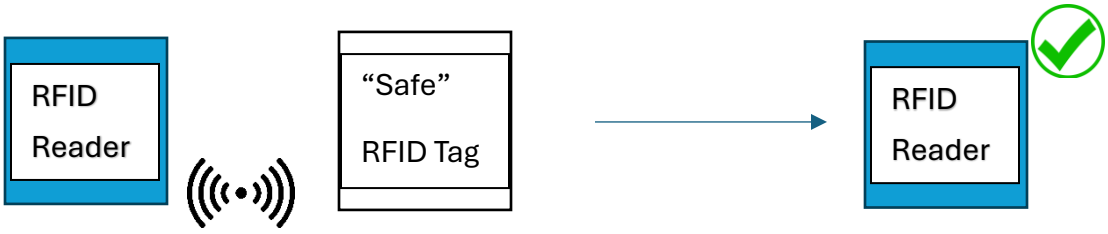
Anul II, ETTI, 4LF633

Cuprins

I.INFORMATII GENERALE

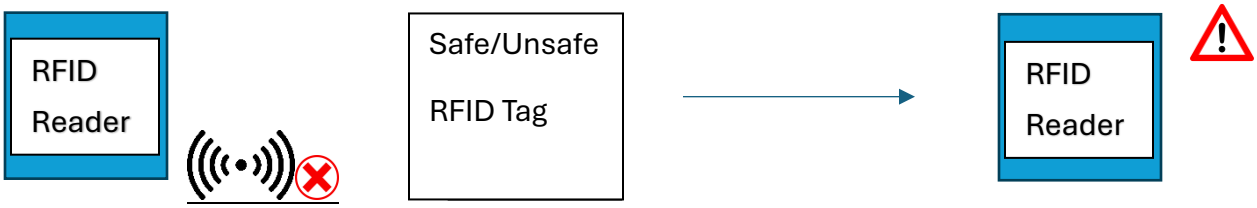
I.1.Titlul proiectului

Numele “SecureTag” este unul sugestiv, deoarece acesta reprezinta functionalitatea de baza a sistemului:



(distanta minima intre cititor si tag-ul “safe”)

(statusul sistemului este “safe”)



(distanta prea mare intre cititor si tag-ul safe/

alt tag necunoscut a fost apropiat)

(sistemul alarmeaza utilizatorul)

I.2.Scopul Proiectului

Scopul acestui proiect a fost de a realiza un sistem antifurt de baza care este usor de inteles si adaptabil, in functie de nevoie.

Proiectul este format din componente hardware si o pagina web pentru a vizualiza status-ul sistemului.

De asemenea, proiectul m-a ajutat mult personal, deoarece am avut oportunitatea de a invata lucruri noi, folositoare, care ma vor ajuta in viitor.

I.3.Obiective/Functii

-crearea unui sistem usor de inteles si folositor, care permite oricarui utilizator sa il foloseasca/modifice, cu un nivel minim de cunostiinte necesare

-functionarea cititorului RFID independenta, indiferent de celalalte parti ale proiectului(sistemul poate fi folosit in aproape orice situatie)

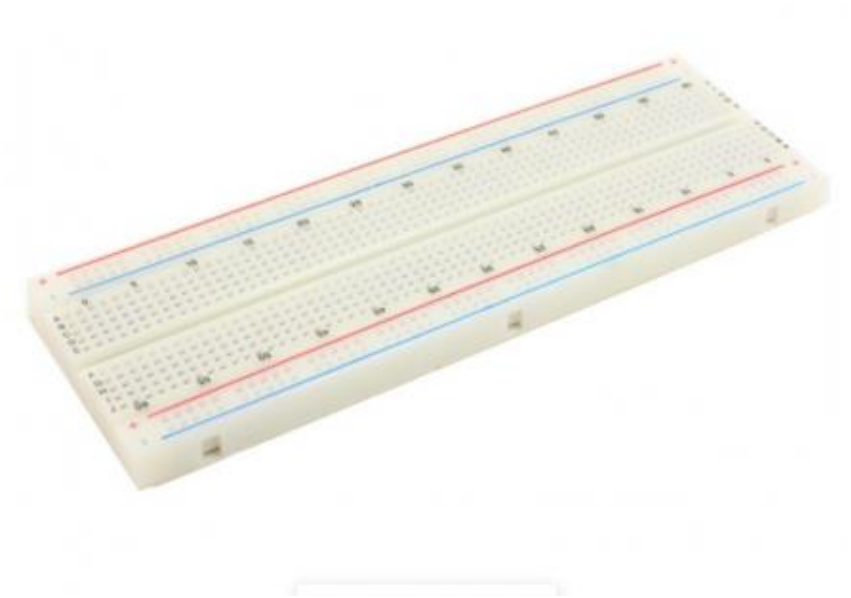
-afisarea statusului sistemului pe mai multe dispozitive simultan folosind o interfata web, accesibila in interiorul retelei create de catre placa de dezvoltare

-si desigur mentinerea unui cost redus de-alungul proiectului

II.INFORMATII HARDWARE

II.1.Componente folosite

-Breadboard 830 puncte (~10 RON)



-Fire mama-mama(~7 RON)

-Fire tata-tata(~5 RON)

-Placa dezvoltare ESP32(~35 RON)

Detalii:

- Microcontroller cu frecvență de 240MHz, dual-core - Tensilica LX6;
- Mufa micro-USB;
- Memorie SRAM 520KB;
- Emițător HT40, protocol 806.11BGN;
- Bluetooth dual-mode (clasic și BLE);
- Memorie Flash 16MB;
- Tensiuni de alimentare: 2.2V-3.6V;
- Temperaturi de lucru de la -40°C to la 125°C;
- Antenă PCB inclusă (dispune și de conexiune pentru atenă externă);
- Rate de transfer: 150 Mbps@11n HT40, 72 Mbps@11n HT20, 54 Mbps@11g, and 11 Mbps@11b;
- Putere de emisie: 19.5 dBm@11b, 16.5 dBm@11g, 15.5 dBm@11n;
- Conectivitate UDP la 135Mbps;
- Consum curent în modul deep sleep: 2.5 μA.

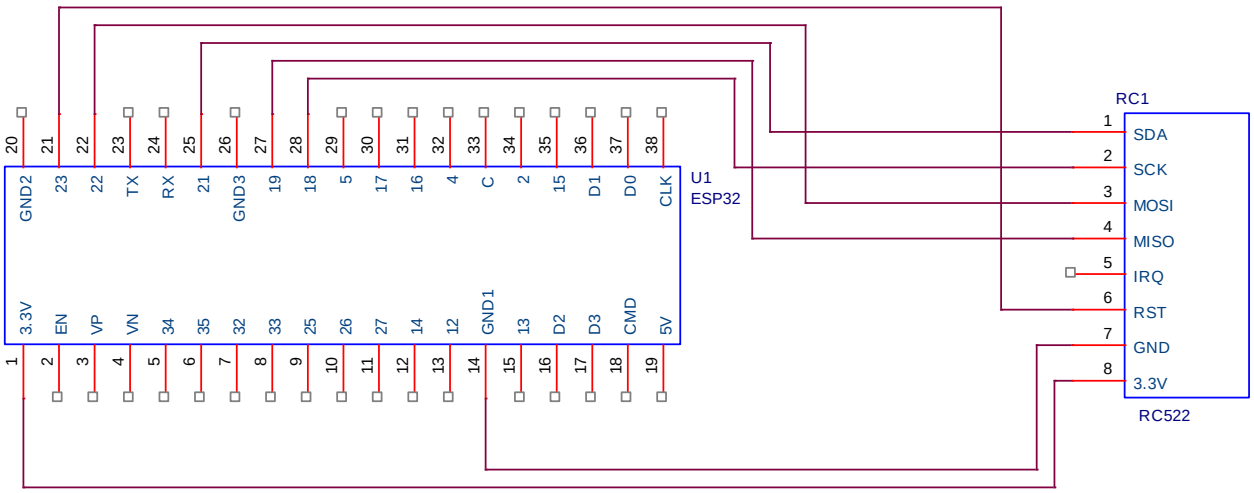
-Modul RFID RC522(~10 RON)

Detalii:

- Tensiune de alimentare: 3.3 V;
- Curent idle: 10 - 13 mA;
- Curent de sleep: 80 uA;
- Curent maxim: 30 mA;
- Frecvență de funcționare: 13.56MHz.
- Carduri suportate: S50, S70, UltraLight, Pro și Desfire.
- Versiune firmware: **0x12**;
- Dimensiune circuit: 40 x 60 mm;
- Protocoale de comunicare:
    - RS232 Serial UART : până la 1228.8 kBd;
    - SPI: până la 10 MBit/s;
    - I^2C: până la 400 kBd în Fast Mode și până la 3400 kBd în High-Speed Mode.
- Buffer FIFO;
- Moduri flexibile de întrerupere;
- Temperatură optimă de funcționare: -25 °C - +85 °C

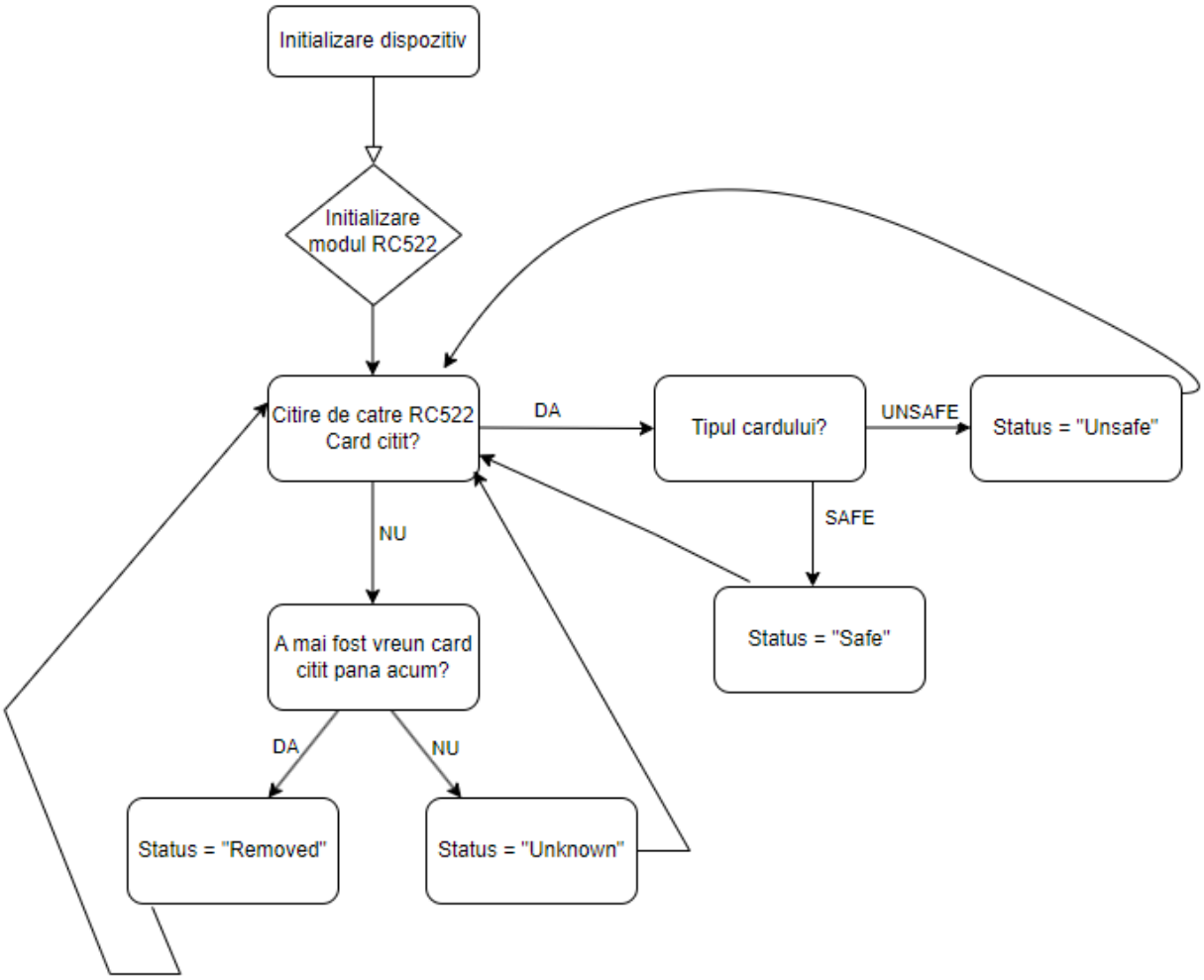
Costul se aproximeaza la 67 RON.

II.2.Schema hardware



ESP32	RC522
3.3V	3.3V
GND	GND
23	RST
22	MOSI
21	SDA
19	MISO
18	SCK

II.3.Schema logica



III.INFORMATII SOFTWARE

Codul sursa al programului este valabil pe pagina personala de GitHub: GitHub

(https://github.com/EddyTX/Sistem-Antifurt)

III.1.Fisiere incluse

```
#include <WiFi.h>
#include <Arduino.h>
#include <SPI.h>
#include <MFRC522.h>
#include <ESPUI.h>
#include "esp32_ap.h"
```

“Wifi.h”

-Gestionarea conexiunilor WiFi pe ESP32.

“Arduino.h”

-Funcții de bază pentru programarea pe platforma Arduino.

“SPI.h”

-Comunicare SPI pentru periferice.

“MFRC522.h”

-Controlul modulelor RFID RC522.

“ESPUI.h”

-Crearea unei interfețe grafice pentru ESP32 printr-un server web.

“esp32_ap.h”

-Configurarea punctului de acces WiFi pentru ESP32

III.2.Parti de cod importante

Dupa cum am mentionat, intregul cod sursa se gaseste pe pagina mea de GitHub. Dar, mai jos am atasat niste secvente importante din codul sursa.

```
#define SS_PIN 21
#define RST_PIN 22
#define SCK_PIN 18
#define MOSI_PIN 23
#define MISO_PIN 19
```

-definirea pinilor pentru comunicarea cu ESP32

```
MFRC522 mfrc522(SS_PIN, RST_PIN);
```
- Instantierea cititorului

```
byte safeCardUID[4] = {0x62, 0x15, 0x48, 0x55};
```
-“hard-codarea” a cardului safe

```
void createESP32AccessPoint() {
  // Configure and start the Access Point
  WiFi.softAP(ssid, password, 6, 0);
  Serial.println("Access Point started!");
  Serial.print("SSID: ");
  Serial.println(ssid);
  Serial.print("Password: ");
  Serial.println(password);

  // Print the ESP32's IP address
  Serial.print("ESP32 IP Address: ");
  Serial.println(WiFi.softAPIP());
}
```

-functie pentru a crea un Access Point al ESP-ului cu SSID-ul “ESP32_Network” si parola “123456789”

```
// Look for new cards
if ( !mfrc522.PICC_IsNewCardPresent()) {
  return;
}
if ( !mfrc522.PICC_ReadCardSerial()) {
  return;
}

bool result = true;
uint8_t buf_len=4;
cpid(&id);
Serial.print("NewCard ");
PrintHex(id.uidByte, id.size);
Serial.println("");
```

-Scanarea si detectarea de noi carduri

```
if (memcmp(id.uidByte, safeCardUID, 4) == 0) {
  Serial.println("Safe card");
  state = "Safe";
  ESPUI.updateLabel(1, state);
} else {
  Serial.println("Not Safe card");
  state = "Unsafe";
  ESPUI.updateLabel(1, state);
}
```

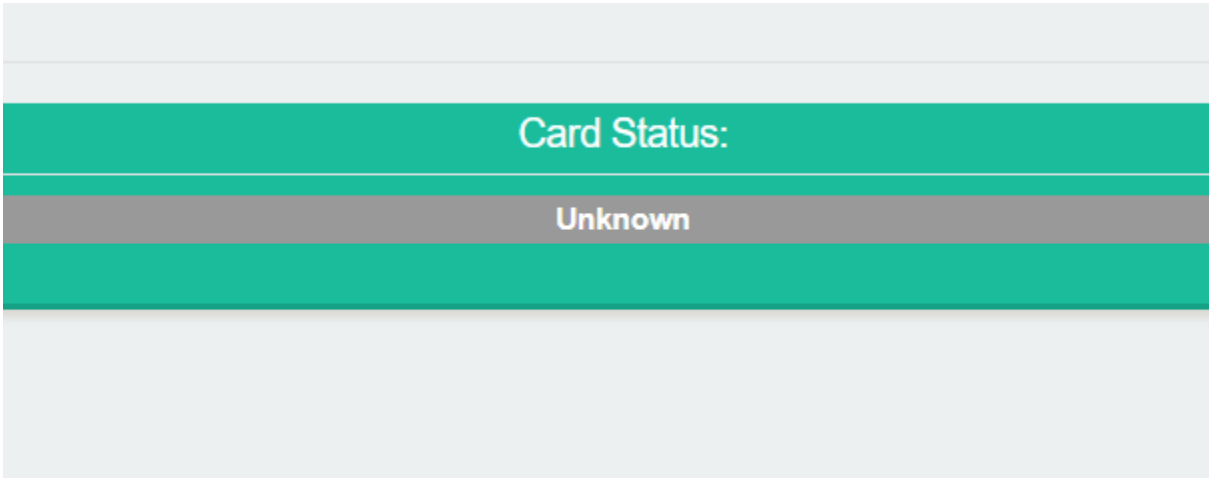
-Verificarea daca cardul detectat este cel safe

```
while(true){
  control=0;
  for(int i=0; i<3; i++){
    if(!mfrc522.PICC_IsNewCardPresent()){
      if(mfrc522.PICC_ReadCardSerial()){
        control |= 0x16;
      }
      if(mfrc522.PICC_ReadCardSerial()){
        control |= 0x16;
      }
      control += 0x1;
    }
    control += 0x4;
  }

  if(control == 13 || control == 14){
    //card is still there
  } else {
    break;
  }
}

Serial.println("CardRemoved");
state = "Removed";
ESPUI.updateLabel(1, state);
delay(2000);
```

-Determina daca cardul inca este prezent pe cititor sau a fost mutat

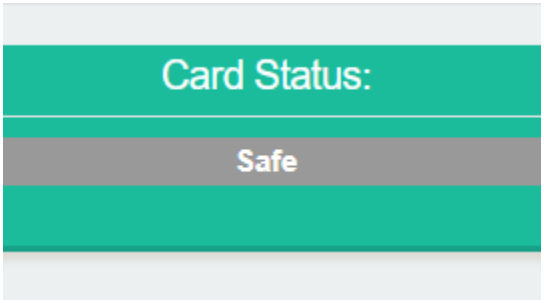
Fragmente pentru actualizarea statusului pe interfata web:

Initial, label-ul arata statusul “Unknown”



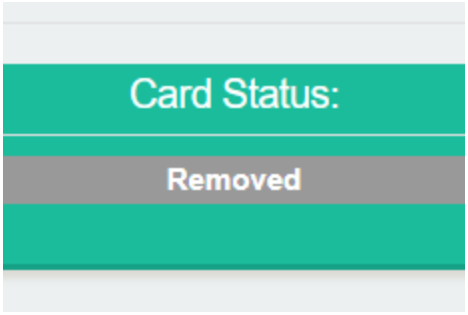
Dupa ce cardul safe a fost apropiat, se apeleaza functia de actualizare a statusului si se schimba in “Safe”.

```
state = "Safe";
ESPUI.updateLabel(1, state);
```



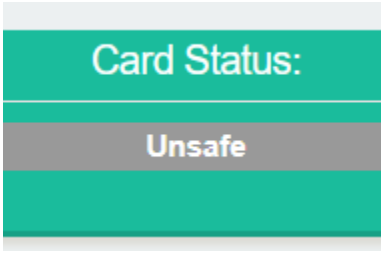
Cand cardul a fost retras de pe cititor, se actualizeaza statusul in “Removed”.

```
state = "Removed";
ESPUI.updateLabel(1, state);
```

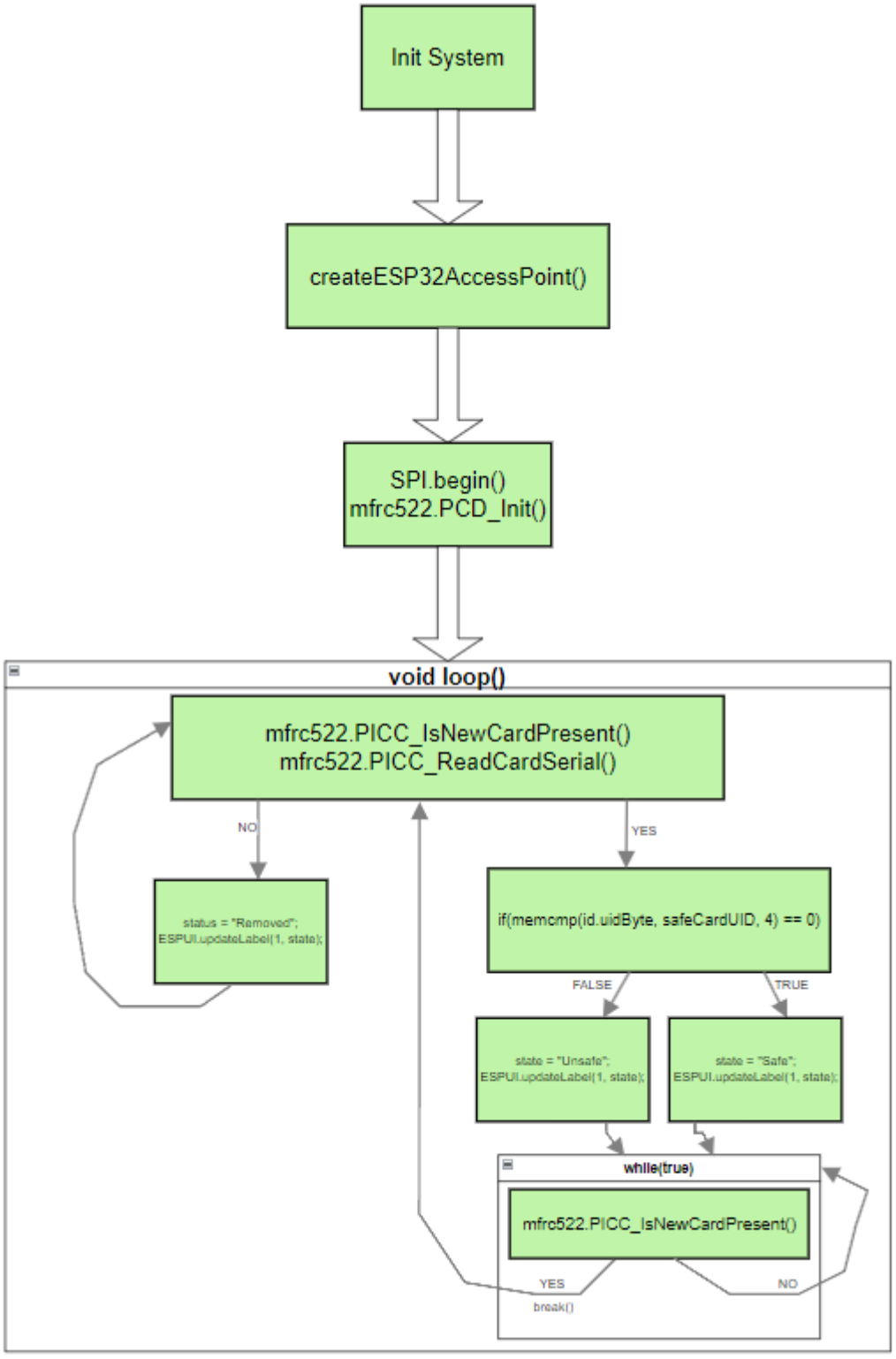


Cand un card “Unsafe” a fost apropiat, statusul se actualizeaza in “Unsafe”.

```
Serial.println("Not Safe card");
state = "Unsafe";
ESPUI.updateLabel(1, state);
```



III.3.Diagrama software



IV.INCHEIERE

IV.1.Probleme intampinate

In mare parte, nu au fost probleme majore care sa impiedice foarte mult realizarea proiectului. Insa, au fost diferite obstacole pe care le-am putut rezolva relativ usor. De exemplu:

- Cititorul RFID nu functiona
  Initial, primul cititor folosit nu dadea niciun rezultat. Schimbarea acestuia cu unul nou a rezolvat problema, probabil era defect.
- ESP32 nu intra corect in modul de Bootloader
  Placa de dezvoltare nu reusea sa intre in modul Bootloader, deci fisierele nu se incarcau corect in memoria sa ceea ce facea ca proiectul sa nu functioneze. Am rezolvat aceasta problema accesand fisierul “platformio.ini” si schimband in configuratie la board din “esp32dev” in “esp32cam”.

- Nu se putea realiza o conexiune FTP
  Initial, ESP32 trebuie sa se conecteze la un server FTP de pe calculatorul personal pentru a modifica un fisier “logs.txt”, iar dupa programul prelua statusul din acest fisier si il afisa pe ecran. Aceasta conexiune a mers la inceput, dar dupa a incetat din a functiona si nu am putut sa aflu deloc motivul. Din aceasta cauza am decis sa folosesc o interfata Web, care a fost mult mai simplu si accesibil.

IV.2.Bibliografie

https://www.handsontec.com/dataspecs/RC522.pdf

https://chatgpt.com

https://randomnerdtutorials.com/esp32-web-server-arduino-ide/

https://github.com/s00500/ESPUI/blob/master/examples/completeExample/completeExample.cpp

https://github.com/dplasa/FTPClientServer

https://github.com/miguelbalboa/rfid/issues/188