```
// #define BLYNK_PRINT Serial
#include <FS.h>
#include <ESP8266WiFi.h>
#include <SPI.h>
#include <MFRC522.h>
#include <BlynkSimpleEsp8266.h>
// Pinos
#define RST_PIN D3
#define SS_PIN D8
#define PIRpin D1
#define LEDpin D0
#define BUZZERpin 1
#define RELEpin D2
// Autentificação
char auth[] = "64f4e332b18e4f0ab7eb20b7fdf7958b";
char ssid[] = "fe";
char pass[] = "fefe1234";
// Variáveis
boolean alarme;
boolean ativo;
boolean disparo;
boolean estadoAnterior;
boolean estadoLED = false;
boolean estadoRele = true;
unsigned long timerLeituraRFID = 0;
WidgetLCD lcd(V1); // Inicia o LCD do app
BlynkTimer timer; // Cria classe BlynkTimer como nome timer
MFRC522 mfrc522(SS_PIN, RST_PIN);
File arquivo; // Cria classe File com nome arquivo
int timerPIR = 1;
int timerRFID = 2;
int timerLED = 3;
int timerBUZZER = 4;
int timerATIVAR = 5;
```

```
BLYNK_CONNECTED() { // Sincroniza as entradas dos botões do app
 Blynk.syncVirtual(V0);
 Blynk.syncVirtual(V2);
}
BLYNK_WRITE(V0) { // Ao clicar no V0 no app executa os comandos
 int alarmeInt = param.asInt(); // Transforma a variável do botão do app em int
 alarme = (bool)alarmeInt; // Transforma a variável em booleana
 alarme = !alarme; // Inverte o estado do alarme
 arquivo = SPIFFS.open("/alarme", "w"); // Abre arquivo para escrita
 arquivo.write(alarme); // Salva o estado do alarme SPI - Flash File System
 arquivo.close(); // Fecha arquivo aberto
 if (alarme != estadoAnterior) {
 mudarAlarme(); // Muda estado do alarme conforme a variável
 lcd.print(0, 1, " via aplicativo "); // Escreve no LCD do app
 }
 estadoAnterior = alarme; // Salva estado anterior do alarme, para ver se ouve real modificação
 arquivo = SPIFFS.open("/anterior", "w");
 arquivo.write(estadoAnterior); // Salva estado anterior
 arquivo.close();
}
BLYNK_WRITE(V2) { // Ao clicar no V2 no app executa os comandos
 estadoRele = !estadoRele;
 arquivo = SPIFFS.open("/rele", "w");
 arquivo.write(estadoRele);
 arquivo.close();
 digitalWrite(RELEpin, !estadoRele);
}
void setup() {
 //Serial.begin(9600);
 boolean fileBegin = false;
 while (fileBegin != true) {
 fileBegin = SPIFFS.begin(); // Inicia a memória flash SPI do ESP8266
 delay(500);
 }
```

```
arquivo = SPIFFS.open("/alarme", "r"); // Lê valor salvo na memória SPI do ESP8266
alarme = arquivo.read();
arquivo.close();
arquivo = SPIFFS.open("/anterior", "r");
estadoAnterior = arquivo.read();
arquivo.close();
arquivo = SPIFFS.open("/ativo", "r");
ativo = arquivo.read();
arquivo.close();
arquivo = SPIFFS.open("/disparo", "r");
disparo = arquivo.read();
arquivo.close();
arquivo = SPIFFS.open("/rele", "r");
estadoRele = arquivo.read();
arquivo.close();
pinMode(PIRpin, INPUT); // Declara os pinos como entradas e saídas
pinMode(LEDpin, OUTPUT);
pinMode(BUZZERpin, OUTPUT);
pinMode(RELEpin, OUTPUT);
digitalWrite(LEDpin, ativo); // Passa os valores lidos dos arquivos para ligar ou desligar as saídas
digitalWrite(BUZZERpin, disparo);
digitalWrite(RELEpin, !estadoRele);
Blynk.begin(auth, ssid, pass); // Inicia a conecção com o cloud server Blynk
SPI.begin();
mfrc522.PCD_Init(); // Inicia o leitor de cartão RFID
lcd.clear(); // Limpa o LCD do app
timerRFID = timer.setInterval(50L, leituraRFID); // Inicia timer de leitura do RFID
delay(25);
timerPIR = timer.setInterval(100L, leituraPIR); // Inicia timer de leitura do PIR
timerBUZZER = timer.setInterval(20000L, buzzerOn); // Tempo paradesativar o alarme após movimento antes de disparar o Buzzer
timerATIVAR = timer.setInterval(20000L, ativarAlarme); // Tempo parasair do local antes de disparar o alarme;
timerLED = timer.setInterval(200L, LEDblink); // Faz piscar o LEDapenas
timer.disable(timerBUZZER);
timer.disable(timerATIVAR);
```

```
 timer.disable(timerLED);
 mudarAlarme();
 if (disparo == HIGH) {
 buzzerOn();
 }
}
void loop() {
 timer.run(); // Inicia os timers no loop;
 Blynk.run(); // Inicia o Blynk no loop
}
void LEDblink () { // Apenas faz piscar o led
 estadoLED = !estadoLED;
 digitalWrite(LEDpin, estadoLED);
}
void buzzerOn () { // Liga o buzzer e envia as mensagens de alerta para email e celular
 digitalWrite(BUZZERpin, HIGH);
 Blynk.email("ALARME DISPARADO!", "ALARME DISPARADO!");
 Blynk.notify("ALARME DISPARADO!");
 timer.disable(timerBUZZER);
}
void ativarAlarme () { // Ativa o alarme após os 20 segundos do alarme
 timer.disable(timerLED);
 timer.disable(timerATIVAR);
 digitalWrite(LEDpin, HIGH);
 ativo = true;
 arquivo = SPIFFS.open("/ativo", "w");
 arquivo.write(1); // Salva o valor do alarme na memória flash
 arquivo.close();
}
void leituraPIR () {
 if (ativo == false || disparo == true) { // Só executa a leitura caso o alarme esteja ativo e o buzzer esteja desligado
 return;
 }
 if (digitalRead(PIRpin) == HIGH) {
```

```
 lcd.clear();
 lcd.print(0, 0, " Movimento ");
 lcd.print(0, 1, " detectado ");
 disparo = true;
 arquivo = SPIFFS.open("/disparo", "w");
 arquivo.write(1);
 arquivo.close();
 timer.enable(timerLED);
 timer.restartTimer(timerBUZZER);
 timer.enable(timerBUZZER); // Ativa o timer de 20 segundos para dar tempo de desligar o alarme
 }
}
void leituraRFID () { // Códigos da leitura do cartão
 if (millis() < timerLeituraRFID) {
 return;
 }
 // Procura por cartao RFID
 if (!mfrc522.PICC_IsNewCardPresent()) {
 return;
 }
 // Seleciona o cartao RFID
 if (!mfrc522.PICC_ReadCardSerial()) {
 return;
 }
 // Mostra UID na serial
 String conteudo = "";
 byte letra;
 for (byte i = 0; i < mfrc522.uid.size; i++) {
 // Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
 // Serial.print(mfrc522.uid.uidByte[i], HEX);
 conteudo.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
 conteudo.concat(String(mfrc522.uid.uidByte[i], HEX));
 }
 // Serial.println();
```

```
 conteudo.toUpperCase();
 if (conteudo.substring(1) == "7C A0 17 A3" || conteudo.substring(1) == "EA 9B 77 89") { // Caso precise registrar mais cartões tirar comentários e
ativar Serial.begin para ver valores
 alarme = !alarme;
 arquivo = SPIFFS.open("/alarme", "w");
 arquivo.write(alarme);
 arquivo.close();
 mudarAlarme();
 estadoAnterior = alarme;
 arquivo = SPIFFS.open("/anterior", "w");
 arquivo.write(estadoAnterior);
 arquivo.close();
 lcd.print(0, 1, " via RFID ");
 timerLeituraRFID = millis() + 2000; // Intervalo para próxima leiturado RFID
 }
 else {
 lcd.clear();
 lcd.print(0, 0, "Cartão Inválido!");
 timerLeituraRFID = millis() + 2000; // Intervalo para próxima leitura
 }
}
void mudarAlarme () { // Muda estado do alarme
 lcd.clear();
 if (alarme == false) {
 Blynk.virtualWrite(V0, HIGH); // Mudar o estado do botão do app para o valor "ATIVAR"
 lcd.print(0, 0, "Alarme Desligado");
 disparo = false;
 arquivo = SPIFFS.open("/disparo", "w");
 arquivo.write(0);
 arquivo.close();
 ativo = false;
 arquivo = SPIFFS.open("/ativo", "w");
 arquivo.write(0);
 arquivo.close();
```

```
 timer.disable(timerLED);
 timer.disable(timerBUZZER);
 timer.disable(timerATIVAR);
 digitalWrite(BUZZERpin, LOW);
 digitalWrite(LEDpin, LOW);
 }
 else if (alarme == true && ativo == false) {
 Blynk.virtualWrite(V0, LOW); // Mudar o estado do botão do app para o valor "DESATIVAR"
 lcd.print(0, 0, " Alarme Ligado ");
 timer.enable(timerLED);
 timer.restartTimer(timerATIVAR);
 timer.enable(timerATIVAR);
 }
 else if (ativo == true && disparo == false) {
 Blynk.virtualWrite(V0, LOW);
 lcd.print(0, 0, " Alarme Ligado ");
 digitalWrite(LEDpin, HIGH);
 }
 else if (disparo == true) {
 lcd.print(0, 0, " Movimento ");
 lcd.print(0, 1, " detectado ");
 timer.enable(timerLED);
 digitalWrite(BUZZERpin, HIGH);
 }
}
```