در اینجا توضیحات بخش‌های مختلف کد به زبان فارسی آمده است تا بهتر متوجه عملکرد آن شوید.

**کتابخانه‌ها و تعریف‌های اولیه**

cpp

Copy code

#include "config.h"

#include <SPI.h>

#include <MFRC522.h>

#include <SoftwareSerial.h>

**توضیح:** در این قسمت، کتابخانه‌های مورد نیاز برای برنامه اضافه می‌شوند:

* SPI: برای ارتباط با RFID.
* MFRC522: برای کار با ماژول RFID.
* SoftwareSerial: برای ارتباط با ماژول SIM800 (درگاه سریال نرم‌افزاری).

**تعریف شماره تلفن و پایه‌ها**

cpp

Copy code

const String PHONE = "+989121231212"; // edit

#define rxPin 15

#define txPin 4

#define BAUD\_RATE 9600

#define RST\_PIN 22

#define SS\_PIN 21

**توضیح:**

* PHONE: شماره تلفن که پیام‌ها به آن ارسال می‌شود.
* rxPin و txPin: پایه‌های ورودی و خروجی که برای ارتباط سریال با ماژول SIM800 استفاده می‌شود.
* RST\_PIN و SS\_PIN: پایه‌های مربوط به ماژول RFID.

**ایجاد enum و آرایه‌ها برای مدیریت کاربران**

cpp

Copy code

enum User { HOSSEIN, AMIN, EHSAN, YOUNES, UNKNOWN };

unsigned long userStartTime[4] = {0}; // Timers for each user

unsigned long userStopTime[4] = {0};

int userPresence[4] = {0}; // Presence flag for each user (0 = out, 1 = in)

const String userUIDs[4] = {

"72 71 88 51", // Hossein

"5D D2 FD A9", // Amin

"E9 89 5F B3", // Ehsan

"79 55 63 B3" // Younes

};

**توضیح:**

* enum User: یک نوع داده برای شناسایی کاربران. مقادیر HOSSEIN، AMIN، EHSAN و YOUNES معرف کاربران مختلف هستند.
* userStartTime[] و userStopTime[]: زمان‌های شروع و پایان حضور کاربران.
* userPresence[]: وضعیت حضور کاربر (0 = خروج، 1 = ورود).
* userUIDs[]: UID های مربوط به کارت‌های RFID کاربران.

**ایجاد Feeds برای ارتباط با Adafruit IO**

cpp

Copy code

AdafruitIO\_Feed \*Text = io.feed("text"); // edit

AdafruitIO\_Feed \*Final[4] = { io.feed("final\_H"), io.feed("final\_A"), io.feed("final\_E"), io.feed("final\_Y") }; // edit

AdafruitIO\_Feed \*userFeeds[4] = { io.feed("Hossein"), io.feed("Amin"), io.feed("Ehsan"), io.feed("Younes") }; // edit

**توضیح:**

* این قسمت برای ایجاد ارتباط با پلتفرم Adafruit IO استفاده می‌شود.
* Text برای ارسال پیام‌های عمومی است.
* Final[] و userFeeds[]: برای ذخیره اطلاعات حضور و غیاب کاربران استفاده می‌شود.

**تنظیمات اولیه در setup()**

cpp

Copy code

void setup() {

Serial.begin(9600);

Serial.println("ESP32 serial initialize");

sim800.begin(BAUD\_RATE, SERIAL\_8N1, rxPin, txPin);

Serial.println("SIM800L serial initialize");

// Initialize SMS settings

sim800.print("AT+CMGF=1\r"); // SMS text mode

delay(1000);

// Initialize RFID

SPI.begin();

mfrc522.PCD\_Init();

Serial.println("Scan a RFID tag");

// Connect to Adafruit IO

io.connect();

while (io.status() < AIO\_CONNECTED) {

Serial.print(".");

delay(500);

}

Serial.println(io.statusText());

}

**توضیح:**

* این تابع برای تنظیمات اولیه استفاده می‌شود:
  + شروع ارتباط سریال با ESP32 و SIM800.
  + تنظیم ماژول SIM800 در حالت پیام کوتاه.
  + راه‌اندازی SPI و ماژول RFID.
  + اتصال به Adafruit IO.

**حلقه اصلی loop()**

cpp

Copy code

void loop() {

handleSIM800(); // Process incoming SMS

io.run(); // Process Adafruit IO

processRFID(); // Process RFID scans

delay(1000);

}

**توضیح:**

* این تابع حلقه اصلی برنامه است که به صورت مداوم اجرا می‌شود.
  + handleSIM800(): پردازش داده‌های ورودی از ماژول SIM800 (مانند پیام‌های دریافتی).
  + io.run(): اجرای توابع مرتبط با Adafruit IO.
  + processRFID(): پردازش کارت‌های RFID.

**پردازش RFID در processRFID()**

cpp

Copy code

void processRFID() {

if (mfrc522.PICC\_IsNewCardPresent() && mfrc522.PICC\_ReadCardSerial()) {

String content = getTagUID();

User currentUser = identifyUser(content);

if (currentUser != UNKNOWN) {

handleUserEntryExit(currentUser);

} else {

Serial.println("Unknown user detected.");

}

}

}

**توضیح:**

* این تابع برای پردازش کارت‌های RFID استفاده می‌شود. اگر کارتی شناسایی شود:
  + UID کارت از طریق getTagUID() دریافت می‌شود.
  + identifyUser() برای شناسایی کاربر بر اساس UID استفاده می‌شود.
  + ورود یا خروج کاربر توسط handleUserEntryExit() پردازش می‌شود.

**تابع استخراج UID کارت در getTagUID()**

cpp

Copy code

String getTagUID() {

String content = "";

for (byte i = 0; i < mfrc522.uid.size; i++) {

content += (mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");

content += String(mfrc522.uid.uidByte[i], HEX);

}

content.toUpperCase();

Serial.println("UID tag: " + content);

return content.substring(1); // Remove leading space

}

**توضیح:**

* این تابع UID کارت RFID را به دست آورده و آن را به صورت یک رشته (string) برمی‌گرداند.

**شناسایی کاربر بر اساس UID در identifyUser()**

cpp

Copy code

User identifyUser(const String &uid) {

for (int i = 0; i < 4; i++) {

if (uid == userUIDs[i]) return static\_cast<User>(i);

}

return UNKNOWN;

}

**توضیح:**

* این تابع UID کارت را بررسی کرده و کاربر مربوط به آن را بر اساس آرایه userUIDs[] شناسایی می‌کند. اگر UID نامعتبر باشد، UNKNOWN برگردانده می‌شود.

**پردازش ورود و خروج کاربر در handleUserEntryExit()**

cpp

Copy code

void handleUserEntryExit(User user) {

if (userPresence[user] == 0) {

userPresence[user] = 1;

userStartTime[user] = millis();

sendNotification(user, "Vared Shod");

} else {

userPresence[user] = 0;

userStopTime[user] = millis();

sendNotification(user, "Kharje Shod");

Final[user]->save(userStopTime[user] - userStartTime[user]);

}

}

**توضیح:**

* این تابع وضعیت ورود یا خروج کاربر را مدیریت می‌کند:
  + اگر کاربر وارد شده باشد، وضعیت حضور آن به 1 تغییر می‌کند و زمان ورود ثبت می‌شود.
  + اگر کاربر خارج شود، وضعیت به 0 تغییر کرده و زمان خروج ثبت می‌شود.

**ارسال پیام و ذخیره در Adafruit IO در sendNotification()**

cpp

Copy code

void sendNotification(User user, const String &action) {

String userName = getUserName(user);

String message = userName + " " + action;

Serial.println(message);

Text->save(message);

userFeeds[user]->save(userPresence[user] == 1 ? "1" : "0");

Reply(userName[0] + String(userPresence[user] == 1 ? " in" : " out"));

}

**توضیح:**

* این تابع برای ارسال پیام (چه در سریال، چه در SMS و چه در Adafruit IO) استفاده می‌شود.
* نام کاربر و عملی که انجام شده (ورود یا خروج) نمایش داده و ذخیره می‌شود.

**دریافت نام کاربر از enum در getUserName()**

cpp

Copy code

String getUserName(User user) {

switch (user) {

case HOSSEIN: return "Hossein";

case AMIN: return "Amin";

case EHSAN: return "Ehsan";

case YOUNES: return "Younes";

default: return "Unknown";

}

}

**توضیح:**

* این تابع نام کاربر را بر اساس مقادیر enum User برمی‌گرداند.

**پردازش پیام‌های دریافتی در `parseData()**