#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <IRremote.h>

#include <Servo.h>

#include <SoftwareSerial.h> // Include SoftwareSerial library for GSM

LiquidCrystal\_I2C lcd(0x27, 16, 2);

const int RECV\_PIN = 10;

const int LED\_PIN = 12; // Built-in LED on pin 13

const int GSM\_PIN = 6; // GSM control pin (turn on/off GSM module)

const int PIR\_PIN1 = 7; // PIR sensor pin 1

const int PIR\_PIN2 = 8; // PIR sensor pin 2

const int BUZZER\_PIN = 13; // Piezo buzzer pin

const int SERVO\_PIN = 11;

const int TOUCH\_PIN = 5;

Servo doorServo;

// Assign new pins for SoftwareSerial (GSM)

const int GSM\_TX = 3; // TX for GSM module

const int GSM\_RX = 2; // RX for GSM module

SoftwareSerial gsmSerial(GSM\_RX, GSM\_TX); // RX and TX pins for SoftwareSerial

const unsigned long Turn\_ON = 0xFA05FD02;

const unsigned long Turn\_Off = 0xA05FFD02;

const unsigned long Pass\_Key[] = { 0xA15EFD02, 0xA25DFD02, 0xE11EFD02, 0xB54AFD02 };

const int Pass\_Key\_LENGTH = 4;

unsigned long lastCodeReceived = 0;

unsigned long lastTime = 0;

const unsigned long debounceDelay = 200;

bool circuitOn = false;

bool passKeyMode = false;

bool touchSensorEnabled = true;

bool touchSensorState = LOW; // Correctly declare the touchSensorState variable

const int maxWrongAttempts = 3;

int KeyCounter = 0;

void setup() {

pinMode(TOUCH\_PIN, INPUT);

pinMode(PIR\_PIN1, INPUT);

pinMode(PIR\_PIN2, INPUT);

pinMode(BUZZER\_PIN, OUTPUT);

digitalWrite(BUZZER\_PIN, LOW); // Ensure buzzer is off initially

Serial.begin(9600);

gsmSerial.begin(9600); // Initialize GSM module

IrReceiver.begin(RECV\_PIN, DISABLE\_LED\_FEEDBACK); // Initialize IR Receiver

doorServo.attach(SERVO\_PIN);

doorServo.write(0); // Keep the servo at 0 degrees initially (door closed)

lcd.begin();

lcd.backlight();

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Setting Up");

}

void loop() {

if (IrReceiver.decode()) {

unsigned long currentTime = millis();

unsigned long codeReceived = IrReceiver.decodedIRData.decodedRawData;

if (codeReceived != lastCodeReceived || (currentTime - lastTime) > debounceDelay) {

lastCodeReceived = codeReceived;

lastTime = currentTime;

Serial.println(codeReceived, HEX);

if (codeReceived == Turn\_ON) {

circuitOn = true;

passKeyMode = false;

touchSensorEnabled = true;

KeyCounter = 0;

lcd.clear();

lcd.setCursor(2, 0);

lcd.print("System ON");

} else if (codeReceived == Turn\_Off) {

circuitOn = false;

passKeyMode = false;

touchSensorEnabled = true;

KeyCounter = 0;

lcd.clear();

lcd.setCursor(2, 0);

lcd.print("System Off");

doorServo.write(0); // Close the door

} else if (circuitOn && passKeyMode) {

handlePassKey(codeReceived);

}

IrReceiver.resume();

}

}

if (circuitOn && touchSensorEnabled) {

handleTouchSensor();

}

if (circuitOn) {

handleMotionDetection();

}

}

// Function to handle pass key input

void handlePassKey(unsigned long codeReceived) {

bool passKeyMatched = false;

for (int i = 0; i < Pass\_Key\_LENGTH; i++) {

if (codeReceived == Pass\_Key[i]) {

passKeyMatched = true;

break;

}

}

if (passKeyMatched) {

lcd.clear();

lcd.setCursor(1, 0);

lcd.print("Welcome Home");

passKeyMode = false;

touchSensorEnabled = false;

KeyCounter = 0;

doorServo.write(90); // Open the door

delay(5000); // Keep the door open for 5 seconds

doorServo.write(0); // Close the door

circuitOn = false;

lcd.clear();

lcd.setCursor(2, 0);

lcd.print("System OFF");

} else {

KeyCounter++;

if (KeyCounter < maxWrongAttempts) {

lcd.clear();

lcd.setCursor(0, 1);

lcd.print("Attempts Left: ");

lcd.print(maxWrongAttempts - KeyCounter);

}

if (KeyCounter >= maxWrongAttempts) {

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Max Attempts Reached");

lcd.setCursor(0, 1);

lcd.print("System Locked");

// Add any additional logic here if needed when the max attempts are reached

}

}

}

// Function to handle touch sensor input

void handleTouchSensor() {

touchSensorState = digitalRead(TOUCH\_PIN);

if (touchSensorState == HIGH) {

Serial.println("Touch detected");

lcd.clear();

lcd.setCursor(2, 0);

lcd.print("Enter Pass Key");

passKeyMode = true;

delay(500);

}

}

void handleMotionDetection() {

if (digitalRead(PIR\_PIN1) == HIGH || digitalRead(PIR\_PIN2) == HIGH) {

delay(100); // Add a small delay to filter out noise

if (digitalRead(PIR\_PIN1) == HIGH || digitalRead(PIR\_PIN2) == HIGH) { // Check again after the delay

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Motion Detected");

for (int i = 0; i < 3; i++) {

digitalWrite(BUZZER\_PIN, HIGH);

Serial.println("Buzzer ON"); // Debugging statement

delay(500); // Buzzer ON for 500ms

digitalWrite(BUZZER\_PIN, LOW);

Serial.println("Buzzer OFF"); // Debugging statement

delay(500); // Buzzer OFF for 500ms

}

MakeCall(); // Trigger the call

delay(5000); // Delay to avoid multiple triggers

lcd.clear();

lcd.setCursor(2, 0);

lcd.print("THIEF ALERT");

}

}

}

// Function to make a call

void MakeCall() {

Serial.println("Calling, please wait....");

sendCommand("ATD+923123363292;", 1000); // Replace with your recipient's number

delay(1000);

Serial.println("Calling successfully");

}

// Function to send AT commands

void sendCommand(String command, int delayTime) {

gsmSerial.println(command);

delay(delayTime);

}