**GSM BASED DOOR ALERT SYSTEM  
 (SOURCE CODE)**/\*

\* GSM-Based Door Alert System

\* Real-time SMS notification when door opens

\*

\* Components:

\* - Arduino Uno

\* - GSM Module (SIM800L/SIM900A)

\* - Magnetic Door Sensor (Reed Switch)

\* - Power Supply

\*

\* Connections:

\* GSM Module:

\* VCC -> 3.7V-4.2V (External power supply recommended)

\* GND -> GND

\* TXD -> Pin 2 (Arduino RX)

\* RXD -> Pin 3 (Arduino TX)

\*

\* Magnetic Door Sensor:

\* One wire -> Pin 4 (Digital Input)

\* Other wire -> GND

\* Internal pull-up resistor enabled

\*

\* LED Indicator (Optional):

\* Anode -> Pin 13

\* Cathode -> GND (through 220Ω resistor)

\*/

#include <SoftwareSerial.h>

// Pin Definitions

#define DOOR\_SENSOR\_PIN 4 // Magnetic door sensor input pin

#define GSM\_RX\_PIN 2 // GSM module TX connects here

#define GSM\_TX\_PIN 3 // GSM module RX connects here

#define LED\_PIN 13 // Status LED pin

#define BUZZER\_PIN 8 // Optional buzzer pin

// GSM Configuration

SoftwareSerial gsmSerial(GSM\_RX\_PIN, GSM\_TX\_PIN);

// System Variables

bool doorState = HIGH; // Current door state (HIGH = closed, LOW = open)

bool lastDoorState = HIGH; // Previous door state

bool systemReady = false; // GSM module initialization status

unsigned long lastDebounceTime = 0; // For debouncing

unsigned long debounceDelay = 50; // Debounce delay in milliseconds

unsigned long lastAlertTime = 0; // Last alert timestamp

unsigned long alertCooldown = 5000; // Minimum time between alerts (5 seconds)

// Configuration - MODIFY THESE VALUES

const String PHONE\_NUMBER = "+1234567890"; // Replace with your phone number

const String ALERT\_MESSAGE = "ALERT: Door has been opened! Time: ";

const String SYSTEM\_READY\_MSG = "Door Security System is now active and monitoring.";

void setup() {

// Initialize serial communication

Serial.begin(9600);

gsmSerial.begin(9600);

// Configure pins

pinMode(DOOR\_SENSOR\_PIN, INPUT\_PULLUP); // Enable internal pull-up resistor

pinMode(LED\_PIN, OUTPUT);

pinMode(BUZZER\_PIN, OUTPUT);

// Initial LED state

digitalWrite(LED\_PIN, LOW);

Serial.println("=== GSM Door Alert System Starting ===");

Serial.println("Initializing GSM module...");

// Initialize GSM module

initializeGSM();

// Read initial door state

doorState = digitalRead(DOOR\_SENSOR\_PIN);

lastDoorState = doorState;

Serial.println("System ready. Monitoring door status...");

Serial.println("Door Status: " + String(doorState == HIGH ? "CLOSED" : "OPEN"));

// Send system ready notification

if (systemReady) {

sendSMS(PHONE\_NUMBER, SYSTEM\_READY\_MSG);

}

// Indicate system is ready

blinkLED(3, 200);

}

void loop() {

// Read current door sensor state

int reading = digitalRead(DOOR\_SENSOR\_PIN);

// Debounce the sensor reading

if (reading != lastDoorState) {

lastDebounceTime = millis();

}

if ((millis() - lastDebounceTime) > debounceDelay) {

// If the reading has been stable for debounceDelay

if (reading != doorState) {

doorState = reading;

// Door state changed

if (doorState == LOW) { // Door opened (sensor triggered)

handleDoorOpened();

} else { // Door closed

handleDoorClosed();

}

}

}

lastDoorState = reading;

// Check for incoming GSM messages (optional feature)

if (gsmSerial.available()) {

handleGSMResponse();

}

// Check for serial commands (for testing/debugging)

if (Serial.available()) {

handleSerialCommands();

}

delay(100); // Small delay for stability

}

void initializeGSM() {

Serial.println("Testing GSM module connection...");

// Test AT command

gsmSerial.println("AT");

delay(1000);

if (waitForResponse("OK", 5000)) {

Serial.println("GSM module responded successfully");

} else {

Serial.println("GSM module not responding. Check connections.");

return;

}

// Set SMS text mode

Serial.println("Setting SMS text mode...");

gsmSerial.println("AT+CMGF=1");

delay(1000);

if (waitForResponse("OK", 5000)) {

Serial.println("SMS text mode set successfully");

} else {

Serial.println("Failed to set SMS text mode");

return;

}

// Check network registration

Serial.println("Checking network registration...");

gsmSerial.println("AT+CREG?");

delay(2000);

// Check signal strength

Serial.println("Checking signal strength...");

gsmSerial.println("AT+CSQ");

delay(1000);

systemReady = true;

Serial.println("GSM module initialized successfully!");

}

void handleDoorOpened() {

Serial.println("ALERT: Door opened!");

// Visual and audio indication

digitalWrite(LED\_PIN, HIGH);

soundBuzzer(3, 100);

// Check cooldown period to prevent spam alerts

if (millis() - lastAlertTime > alertCooldown) {

if (systemReady) {

String message = ALERT\_MESSAGE + getCurrentTime();

sendSMS(PHONE\_NUMBER, message);

lastAlertTime = millis();

Serial.println("SMS alert sent successfully");

} else {

Serial.println("GSM not ready. Alert not sent.");

}

} else {

Serial.println("Alert cooldown active. SMS not sent.");

}

}

void handleDoorClosed() {

Serial.println("Door closed");

digitalWrite(LED\_PIN, LOW);

// Optional: Send door closed notification

// Uncomment the lines below if you want notifications for door closing too

/\*

if (systemReady && (millis() - lastAlertTime > alertCooldown)) {

String message = "INFO: Door has been closed. Time: " + getCurrentTime();

sendSMS(PHONE\_NUMBER, message);

lastAlertTime = millis();

}

\*/

}

void sendSMS(String phoneNumber, String message) {

Serial.println("Sending SMS to: " + phoneNumber);

Serial.println("Message: " + message);

// Set SMS recipient

gsmSerial.println("AT+CMGS=\"" + phoneNumber + "\"");

delay(1000);

// Send the message

gsmSerial.print(message);

delay(500);

// End SMS with Ctrl+Z (ASCII 26)

gsmSerial.write(26);

delay(2000);

// Wait for confirmation

if (waitForResponse("OK", 10000)) {

Serial.println("SMS sent successfully!");

blinkLED(2, 100);

} else {

Serial.println("Failed to send SMS");

blinkLED(5, 50); // Fast blink indicates error

}

}

bool waitForResponse(String expectedResponse, unsigned long timeout) {

String response = "";

unsigned long startTime = millis();

while (millis() - startTime < timeout) {

if (gsmSerial.available()) {

char c = gsmSerial.read();

response += c;

Serial.print(c); // Echo GSM response to serial monitor

if (response.indexOf(expectedResponse) != -1) {

return true;

}

}

}

return false;

}

void handleGSMResponse() {

String response = "";

while (gsmSerial.available()) {

char c = gsmSerial.read();

response += c;

Serial.print(c);

}

// Check for incoming SMS notification

if (response.indexOf("+CMTI:") != -1) {

Serial.println("\nIncoming SMS detected");

// Optional: Add code to read and process incoming SMS

}

}

void handleSerialCommands() {

String command = Serial.readString();

command.trim();

command.toUpperCase();

if (command == "TEST") {

Serial.println("Testing SMS functionality...");

sendSMS(PHONE\_NUMBER, "Test message from Door Alert System");

}

else if (command == "STATUS") {

Serial.println("=== System Status ===");

Serial.println("GSM Ready: " + String(systemReady ? "YES" : "NO"));

Serial.println("Door State: " + String(doorState == HIGH ? "CLOSED" : "OPEN"));

Serial.println("Phone Number: " + PHONE\_NUMBER);

Serial.println("===================");

}

else if (command == "RESET") {

Serial.println("Resetting GSM module...");

systemReady = false;

initializeGSM();

}

else if (command.startsWith("PHONE:")) {

// Format: PHONE:+1234567890

String newPhone = command.substring(6);

Serial.println("Phone number would be updated to: " + newPhone);

Serial.println("(Note: Restart required for permanent change)");

}

else {

Serial.println("Available commands:");

Serial.println("TEST - Send test SMS");

Serial.println("STATUS - Show system status");

Serial.println("RESET - Reset GSM module");

Serial.println("PHONE:+number - Set phone number");

}

}

String getCurrentTime() {

// Simple timestamp (you can enhance this with RTC module)

unsigned long currentTime = millis();

unsigned long seconds = currentTime / 1000;

unsigned long minutes = seconds / 60;

unsigned long hours = minutes / 60;

seconds = seconds % 60;

minutes = minutes % 60;

hours = hours % 24;

String timeStr = String(hours) + ":" +

String(minutes) + ":" +

String(seconds) + " (uptime)";

return timeStr;

}

void blinkLED(int times, int delayMs) {

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

digitalWrite(LED\_PIN, HIGH);

delay(delayMs);

digitalWrite(LED\_PIN, LOW);

delay(delayMs);

}

}

void soundBuzzer(int times, int delayMs) {

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

digitalWrite(BUZZER\_PIN, HIGH);

delay(delayMs);

digitalWrite(BUZZER\_PIN, LOW);

delay(delayMs);

}

}

// Optional: Function to check GSM module status periodically

void checkGSMStatus() {

Serial.println("Checking GSM status...");

gsmSerial.println("AT+CREG?"); // Check network registration

delay(1000);

gsmSerial.println("AT+CSQ"); // Check signal strength

delay(1000);

// Read responses

while (gsmSerial.available()) {

Serial.write(gsmSerial.read());

}

}

/\*

\* INSTALLATION AND SETUP INSTRUCTIONS:

\*

\* 1. Hardware Connections:

\* - Connect GSM module VCC to external 3.7V-4.2V power supply

\* - Connect GSM GND to Arduino GND and power supply GND

\* - Connect GSM TXD to Arduino pin 2

\* - Connect GSM RXD to Arduino pin 3

\* - Connect door sensor one wire to Arduino pin 4

\* - Connect door sensor other wire to Arduino GND

\* - Optional: Connect LED to pin 13 and buzzer to pin 8

\*

\* 2. Software Configuration:

\* - Update PHONE\_NUMBER with your mobile number (include country code)

\* - Customize ALERT\_MESSAGE as needed

\* - Insert SIM card in GSM module (ensure it has SMS capability)

\*

\* 3. Testing:

\* - Upload code to Arduino

\* - Open Serial Monitor (9600 baud)

\* - Wait for "System ready" message

\* - Test by opening/closing door

\* - Use serial commands: TEST, STATUS, RESET

\*

\* 4. Troubleshooting:

\* - If GSM doesn't respond: Check power supply and connections

\* - If no SMS received: Verify phone number format and SIM card credit

\* - If false alerts: Adjust debounceDelay value

\* - Check serial monitor for detailed status messages

\*/