1111111111111111111111111111111111111111111111111111

#include "mbed.h"

#include "arm\_book\_lib.h"

int main()

{

    AnalogIn gasDetector(A0);        // MQ2 analog output

    AnalogIn tempSensor(A1);         // LM35 analog output

    DigitalOut alarmLed(D5);         // External LED

    alarmLed = OFF;

    float gasThreshold = 0.4f;       // MQ2 threshold (0.0–1.0)

    float tempThreshold = 35.0f;     // LM35 threshold in °C

    while (true) {

        float gasValue = gasDetector.read();             // 0.0–1.0

        float tempVoltage = tempSensor.read() \* 3.3f;    // Convert to voltage

        float tempCelsius = tempVoltage \* 100.0f;        // 10mV per °C

        if (gasValue > gasThreshold || tempCelsius > tempThreshold) {

            alarmLed = ON;

        } else {

            alarmLed = OFF;

        }

    }

}

2222222222222222222222222222222222222222222222222222222

#include "mbed.h"

#include "arm\_book\_lib.h"

int main()

{

    AnalogIn gasDetector(A0);        // MQ2 analog output

    AnalogIn tempSensor(A1);         // LM35 analog output

    DigitalOut alarmLed(D5);         // External LED

    float gasThreshold = 0.4f;       // MQ2 threshold

    float tempThreshold = 35.0f;     // Temp threshold in °C

    bool alarmState = OFF;           // Sticky alarm state

    while (true) {

        float gasValue = gasDetector.read();

        float tempVoltage = tempSensor.read() \* 3.3f;

        float tempCelsius = tempVoltage \* 100.0f;

        // 🔒 Latching logic — once triggered, stays ON

        if (gasValue > gasThreshold || tempCelsius > tempThreshold) {

            alarmState = ON;

        }

        alarmLed = alarmState;

    }

}

333333333333333333333333333333333333333333333333333333

#include "mbed.h"

#include "arm\_book\_lib.h"

// Sensors

AnalogIn gasSensor(A0);

AnalogIn tempSensor(A1);

// Output

DigitalOut alarmLed(D5);

// Buttons (each represents one digit)

DigitalIn button1(D2); // Digit 1

DigitalIn button2(D3); // Digit 2

DigitalIn button3(D4); // Digit 3

DigitalIn button4(D6); // Digit 4

// Thresholds

float gasThreshold = 0.4f;

float tempThreshold = 35.0f;

// State variables

bool alarmState = OFF;

bool emergencyMode = OFF;

int enteredCode[4];

int codeIndex = 0;

const int correctCode[4] = {1, 2, 3, 4};

// Debounce function

bool readButton(DigitalIn &button) {

if (button == 1) {

ThisThread::sleep\_for(200ms);

if (button == 1) return true;

}

return false;

}

int main() {

// Pull-downs for buttons

button1.mode(PullDown);

button2.mode(PullDown);

button3.mode(PullDown);

button4.mode(PullDown);

alarmLed = OFF;

while (true) {

float gasValue = gasSensor.read();

float tempVoltage = tempSensor.read() \* 3.3f;

float tempCelsius = tempVoltage \* 100.0f;

// Latching alarm mode

if ((gasValue > gasThreshold || tempCelsius > tempThreshold) && !emergencyMode) {

alarmState = ON;

}

// Emergency mode if both are triggered

if (gasValue > gasThreshold && tempCelsius > tempThreshold) {

emergencyMode = ON;

}

// Emergency flashing

if (emergencyMode) {

alarmLed = !alarmLed;

ThisThread::sleep\_for(200ms); // Flash rate

// Handle button inputs

if (readButton(button1)) enteredCode[codeIndex++] = 1;

if (readButton(button2)) enteredCode[codeIndex++] = 2;

if (readButton(button3)) enteredCode[codeIndex++] = 3;

if (readButton(button4)) enteredCode[codeIndex++] = 4;

// Check code

if (codeIndex == 4) {

bool correct = true;

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

if (enteredCode[i] != correctCode[i]) {

correct = false;

break;

}

}

if (correct) {

emergencyMode = OFF;

alarmState = OFF;

alarmLed = OFF; // ✅ Force LED OFF

ThisThread::sleep\_for(300ms); // Optional: prevent flicker

}

codeIndex = 0; // Reset input

}

} else {

// Normal mode (no emergency)

alarmLed = alarmState;

ThisThread::sleep\_for(100ms);

}

}

}

44444444444444444444444444444444444444444444

#include "mbed.h"

#include "arm\_book\_lib.h"

// Sensors

AnalogIn gasSensor(A0);

AnalogIn tempSensor(A1);

// Output

DigitalOut alarmLed(D5);       // Main alarm LED

DigitalOut lockoutLed(D7);     // Blinks slowly during lockout

// Buttons (each represents one digit)

DigitalIn button1(D2); // Digit 1

DigitalIn button2(D3); // Digit 2

DigitalIn button3(D4); // Digit 3

DigitalIn button4(D6); // Digit 4

// Thresholds

float gasThreshold = 0.4f;

float tempThreshold = 35.0f;

// States

bool alarmState = OFF;

bool emergencyMode = OFF;

bool lockout = false;

int enteredCode[4];

int codeIndex = 0;

const int correctCode[4] = {1, 2, 3, 4};

int wrongAttempts = 0;

Timer lockoutTimer;

// Debounce helper

bool readButton(DigitalIn &button) {

    if (button == 1) {

        ThisThread::sleep\_for(200ms);

        if (button == 1) return true;

    }

    return false;

}

int main() {

    // Configure pull-downs for all buttons

    button1.mode(PullDown);

    button2.mode(PullDown);

    button3.mode(PullDown);

    button4.mode(PullDown);

    alarmLed = OFF;

    lockoutLed = OFF;

    while (true) {

        float gasValue = gasSensor.read();

        float tempVoltage = tempSensor.read() \* 3.3f;

        float tempCelsius = tempVoltage \* 100.0f;

        // Normal alarm activation

        if ((gasValue > gasThreshold || tempCelsius > tempThreshold) && !emergencyMode) {

            alarmState = ON;

        }

        // Emergency mode activation

        if (gasValue > gasThreshold && tempCelsius > tempThreshold) {

            emergencyMode = ON;

        }

        // 🔐 LOCKOUT MODE

        if (lockout) {

            alarmLed = OFF;

            lockoutLed = !lockoutLed;

            ThisThread::sleep\_for(1000ms);  // Slow blink 1s

            if (lockoutTimer.elapsed\_time() >= 60s) {

                // End lockout

                lockout = false;

                wrongAttempts = 0;

                lockoutLed = OFF;

                lockoutTimer.stop();

                lockoutTimer.reset();

            }

            continue;  // Skip everything else during lockout

        }

        // 🚨 EMERGENCY MODE

        if (emergencyMode) {

            alarmLed = !alarmLed;

            ThisThread::sleep\_for(200ms);  // Fast blink

            // Handle code input via buttons

            if (readButton(button1)) enteredCode[codeIndex++] = 1;

            if (readButton(button2)) enteredCode[codeIndex++] = 2;

            if (readButton(button3)) enteredCode[codeIndex++] = 3;

            if (readButton(button4)) enteredCode[codeIndex++] = 4;

            // Check code after 4 inputs

            if (codeIndex == 4) {

                bool correct = true;

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

                    if (enteredCode[i] != correctCode[i]) {

                        correct = false;

                        break;

                    }

                }

                if (correct) {

                    // Reset everything

                    emergencyMode = OFF;

                    alarmState = OFF;

                    alarmLed = OFF;

                    wrongAttempts = 0;

                    ThisThread::sleep\_for(300ms);  // prevent flicker

                } else {

                    wrongAttempts++;

                    if (wrongAttempts >= 5) {

                        lockout = true;

                        lockoutTimer.start();

                    }

                }

                codeIndex = 0;  // Reset entered code

            }

        } else {

            // Normal mode: LED based on alarm state

            alarmLed = alarmState;

            ThisThread::sleep\_for(100ms);

        }

    }

}

5555555555555555555555555555555555555555555555

#include "mbed.h"

#include "arm\_book\_lib.h"

// Sensors

AnalogIn gasSensor(A0);

AnalogIn tempSensor(A1);

// Outputs

DigitalOut alarmLed(D5);       // Emergency LED

DigitalOut lockoutLed(D7);     // Lockout blinking LED

// 4 push buttons for 4-digit code

DigitalIn button1(D2); // 1

DigitalIn button2(D3); // 2

DigitalIn button3(D4); // 3

DigitalIn button4(D6); // 4

// Thresholds

const float gasThreshold = 0.4f;

const float tempThreshold = 35.0f;

// Code and state

const int correctCode[4] = {1, 2, 3, 4};

int enteredCode[4];

int codeIndex = 0;

bool alarmState = OFF;

bool emergencyMode = OFF;

bool lockout = false;

bool lockoutLedState = false;

bool lockoutTimerStarted = false;

int wrongAttempts = 0;

Timer lockoutTimer;

Timer blinkTimer;

// Debounce button

bool readButton(DigitalIn &btn) {

    if (btn == 1) {

        ThisThread::sleep\_for(150ms);

        return btn == 1;

    }

    return false;

}

int main() {

    // Configure buttons

    button1.mode(PullDown);

    button2.mode(PullDown);

    button3.mode(PullDown);

    button4.mode(PullDown);

    alarmLed = OFF;

    lockoutLed = OFF;

    while (true) {

        // Lockout mode

        if (lockout) {

            // Start timers only once

            if (!lockoutTimerStarted) {

                lockoutTimer.start();

                blinkTimer.start();

                lockoutTimerStarted = true;

            }

            // Blink lockout LED slowly

            if (blinkTimer.elapsed\_time() >= 500ms) {

                lockoutLedState = !lockoutLedState;

                lockoutLed = lockoutLedState;

                blinkTimer.reset();

            }

            // After 60 seconds, unlock

            if (lockoutTimer.elapsed\_time() >= 60s) {

                lockout = false;

                wrongAttempts = 0;

                lockoutLed = OFF;

                lockoutLedState = false;

                lockoutTimer.stop();

                lockoutTimer.reset();

                blinkTimer.stop();

                blinkTimer.reset();

                lockoutTimerStarted = false;

            }

            ThisThread::sleep\_for(50ms);

            continue;

        }

        // Read sensor values

        float gasValue = gasSensor.read();

        float tempC = tempSensor.read() \* 3.3f \* 100.0f;

        // Normal alarm mode

        if ((gasValue > gasThreshold || tempC > tempThreshold) && !emergencyMode) {

            alarmState = ON;

        }

        // Emergency mode if both triggered

        if (gasValue > gasThreshold && tempC > tempThreshold) {

            emergencyMode = ON;

        }

        // EMERGENCY MODE

        if (emergencyMode) {

            alarmLed = !alarmLed;

            ThisThread::sleep\_for(200ms);  // Fast blink

            // Code entry

            if (readButton(button1)) enteredCode[codeIndex++] = 1;

            if (readButton(button2)) enteredCode[codeIndex++] = 2;

            if (readButton(button3)) enteredCode[codeIndex++] = 3;

            if (readButton(button4)) enteredCode[codeIndex++] = 4;

            if (codeIndex == 4) {

                bool correct = true;

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

                    if (enteredCode[i] != correctCode[i]) {

                        correct = false;

                        break;

                    }

                }

                if (correct) {

                    emergencyMode = OFF;

                    alarmState = OFF;

                    alarmLed = OFF;

                    wrongAttempts = 0;

                } else {

                    wrongAttempts++;

                    if (wrongAttempts >= 5) {

                        lockout = true;

                    }

                }

                codeIndex = 0;

            }

        } else {

            // Normal alarm (non-emergency)

            alarmLed = alarmState;

            ThisThread::sleep\_for(100ms);

        }

    }

}