1111111111111111111111111111

#include "mbed.h"

#include "arm\_book\_lib.h"

AnalogIn tempSensor(A1);

BufferedSerial pc(USBTX, USBRX, 9600);

char buffer[64];

int main() {

    while (true) {

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

        float temperatureC = voltage \* 100.0f;

        int tempInt = temperatureC;  // Round down to integer

        int len = sprintf(buffer, "Temperature: %d C\r\n", tempInt);

        pc.write(buffer, len);

        ThisThread::sleep\_for(1000ms);

    }

}

2222222222222222222222222222222222222222222

#include "mbed.h"

#include "arm\_book\_lib.h"

AnalogIn pot(A2);                      // Potentiometer connected to A2

BufferedSerial pc(USBTX, USBRX, 9600); // Serial to PC

char buffer[64];

int main() {

    while (true) {

        float value = pot.read();  // Read range: 0.0 to 1.0

        int percentage = value \* 100;

        int len = sprintf(buffer, "Potentiometer: %d %%\r\n", percentage);

        pc.write(buffer, len);

        ThisThread::sleep\_for(500ms);  // Update every half second

    }

}

33333333333333333333333333333333333333333333333333

#include "mbed.h"

#include "arm\_book\_lib.h"

DigitalIn gasSensor(D2);               // Digital output from MQ2

BufferedSerial pc(USBTX, USBRX, 9600); // Serial port

char buffer[64];

int main() {

    gasSensor.mode(PullDown); // Ensure clean LOW when idle

    while (true) {

        if (gasSensor == 1) {

            sprintf(buffer, "Gas Detected!\r\n");

        } else {

            sprintf(buffer, "No Gas\r\n");

        }

        pc.write(buffer, strlen(buffer));

        ThisThread::sleep\_for(1000ms); // Update every 1 second

    }

}

4444444444444444444444444

#include "mbed.h"

#include "arm\_book\_lib.h"

// Inputs

AnalogIn gasSensor(A0);       // MQ2 analog output

AnalogIn tempSensor(A1);      // LM35 analog output

// Serial

BufferedSerial pc(USBTX, USBRX, 9600);

char buffer[64];

// Thresholds

const float gasThreshold = 0.4f;   // Adjust based on calibration

const float tempThreshold = 35.0f; // Celsius

int main() {

    while (true) {

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

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

        float tempCelsius = tempVoltage \* 100.0f;

        // Check alarm conditions

        bool gasAlarm = gasValue > gasThreshold;

        bool tempAlarm = tempCelsius > tempThreshold;

        if (gasAlarm && tempAlarm) {

            sprintf(buffer, "Gas & Temperature Alarm!\r\n");

        } else if (gasAlarm) {

            sprintf(buffer, "Gas Alarm!\r\n");

        } else if (tempAlarm) {

            sprintf(buffer, "Temperature Alarm!\r\n");

        } else {

            sprintf(buffer, "No Alarm\r\n");

        }

        pc.write(buffer, strlen(buffer));

        ThisThread::sleep\_for(1000ms); // Update every second

    }

}

555555555555555555555555555555555555555555

#include "mbed.h"

#include "arm\_book\_lib.h"

// Analog Sensors

AnalogIn tempSensor(A1);     // LM35

AnalogIn gasSensor(A0);      // MQ2 AOUT

AnalogIn potSensor(A2);      // Potentiometer

// Serial

BufferedSerial pc(USBTX, USBRX, 9600);

char buffer[128];

// Thresholds

const float gasThreshold = 0.4f;

const float tempThreshold = 35.0f;

int main() {

    while (true) {

        // Read sensors

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

        float gasValue = gasSensor.read();

        float potValue = potSensor.read();

        int potPercent = potValue \* 100;

        // Determine alarm status

        const char\* status;

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

            status = "Gas & Temp Alarm";

        } else if (gasValue > gasThreshold) {

            status = "Gas Alarm";

        } else if (tempC > tempThreshold) {

            status = "Temp Alarm";

        } else {

            status = "No Alarm";

        }

        // Safe formatted print (no float formatting)

        int tempInt = static\_cast<int>(tempC);

        int gasInt = static\_cast<int>(gasValue \* 100);  // % for gas too if needed

        int len = snprintf(buffer, sizeof(buffer),

            "Temp: %d C | Gas: %d%% | Pot: %d%% | Status: %s\r\n",

            tempInt, gasInt, potPercent, status);

        pc.write(buffer, len);

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

    }

}