Lab

* Main Task 4

1. Continuously read the temperature sensor and display the temperature in degrees Celsius.

**Main Task Code:**

#include "mbed.h"

#include "arm\_book\_lib.h"

// Temperature analog input

AnalogIn lm35Sensor(A1);

// Serial interface to PC

BufferedSerial uart(USBTX, USBRX, 9600);

char serialOut[64];

int main() {

while (true) {

// Convert voltage to °C

float adcVoltage = lm35Sensor.read() \* 3.3f;

float tempInC = adcVoltage \* 100.0f;

int roundedTemp = (int)tempInC;

int chars = sprintf(serialOut, "Temperature: %d°C\r\n", roundedTemp);

uart.write(serialOut, chars);

ThisThread::sleep\_for(1s);

}

}

1. Continuously read the potentiometer value and display it on the serial terminal.

**Main Task Code:**

#include "mbed.h"

#include "arm\_book\_lib.h"

// Potentiometer connected to analog input

AnalogIn knobInput(A2);

// Serial interface

BufferedSerial uart(USBTX, USBRX, 9600);

char outBuffer[64];

int main() {

while (true) {

// Read analog voltage (0.0 to 1.0)

float knobLevel = knobInput.read();

int levelPercent = static\_cast<int>(knobLevel \* 100);

int count = sprintf(outBuffer, "Potentiometer: %d%%\r\n", levelPercent);

uart.write(outBuffer, count);

ThisThread::sleep\_for(500ms); // Update rate: 0.5s

}

}

1. Display the output status of the gas sensor, indicating whether it detects gas or not.

**Main Task Code:**

#include "mbed.h"

#include "arm\_book\_lib.h"

// MQ2 sensor digital output

DigitalIn gasInput(D2);

// Serial interface

BufferedSerial uart(USBTX, USBRX, 9600);

char gasMessage[64];

int main() {

gasInput.mode(PullDown); // Ensure default LOW state

while (true) {

if (gasInput.read() == 1) {

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

} else {

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

}

uart.write(gasMessage, strlen(gasMessage));

ThisThread::sleep\_for(1s);

}

}

1. Display a message on the serial terminal indicating the source of the alarm, such as "Temperature Alarm" or "Gas Alarm."

**Main Task Code:**

#include "mbed.h"

#include "arm\_book\_lib.h"

// Sensor inputs

AnalogIn gasAnalog(A0); // Gas (MQ2)

AnalogIn tempAnalog(A1); // Temperature (LM35)

// Serial communication

BufferedSerial uart(USBTX, USBRX, 9600);

char alertBuffer[64];

// Alert thresholds

const float gasLimit = 0.4f;

const float tempLimit = 35.0f;

int main() {

while (true) {

float gasLevel = gasAnalog.read(); // 0.0–1.0

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

float tempC = voltage \* 100.0f;

bool gasWarning = gasLevel > gasLimit;

bool tempWarning = tempC > tempLimit;

if (gasWarning && tempWarning) {

sprintf(alertBuffer, "Gas and Temperature Alarm!\r\n");

} else if (gasWarning) {

sprintf(alertBuffer, "Gas Warning!\r\n");

} else if (tempWarning) {

sprintf(alertBuffer, "Temperature Warning!\r\n");

} else {

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

}

uart.write(alertBuffer, strlen(alertBuffer));

ThisThread::sleep\_for(1s);

}

}

1. Ensuring all sensor readings are updated on the serial terminal without delays.

**Main Task Code:**

#include "mbed.h"

#include "arm\_book\_lib.h"

// Sensor inputs

AnalogIn tempInput(A1); // LM35 temperature sensor

AnalogIn gasInput(A0); // MQ2 gas sensor

AnalogIn potInput(A2); // Potentiometer

// Serial communication

BufferedSerial uart(USBTX, USBRX, 9600);

char systemStatus[128];

// Safety thresholds

const float gasCutoff = 0.4f;

const float tempCutoff = 35.0f;

int main() {

while (true) {

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

float gasVal = gasInput.read();

float potVal = potInput.read();

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

int gasPercent = static\_cast<int>(gasVal \* 100.0f);

int potPercent = static\_cast<int>(potVal \* 100.0f);

const char\* systemState;

if (gasVal > gasCutoff && tempC > tempCutoff) {

systemState = "Gas and Temperature Alarm!\";

} else if (gasVal > gasCutoff) {

systemState = "Gas Warning!";

} else if (tempC > tempCutoff) {

systemState = "Temperature Warning!";

} else {

systemState = "System OK";

}

int length = snprintf(systemStatus, sizeof(systemStatus),

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

tempInt, gasPercent, potPercent, systemState);

uart.write(systemStatus, length);

ThisThread::sleep\_for(200ms);

}

}