Real-Time and Embedded Systems Design – Project

Team 17

Eman Khaled 18P9713

Omar Hussien 18P1265

Farah Essam 18P3448

Mohamed Mostafa 18P9474

Shehab El Din Adel 18P3863

# Project files link

[Team17\_ProjectFiles](https://drive.google.com/drive/folders/1Ts34PHMl35FpKD1mGMrkU_K7KtCiO32r?usp=sharing)

# Video link

# Introduction

In this project, we will implement a ON/OFF temperature controller application using several techniques like FreeRTOS, Queues, Potentiometer as a temperature sensor, and Buzzers using Multitasking based approaches. Mainly, it is required to control the temperature of an oven using ON-OFF type controller. The setpoint temperature is input on the computer device and connected through UART PuTTy Software. There is no need to stop the application in order to modify the setpoint value. The temperature of the oven is measured using a potentiometer. The buzzer is activated when the temperature exceeds the alarm value. The heater is activated when its value is less than the set point.

# Flowchart

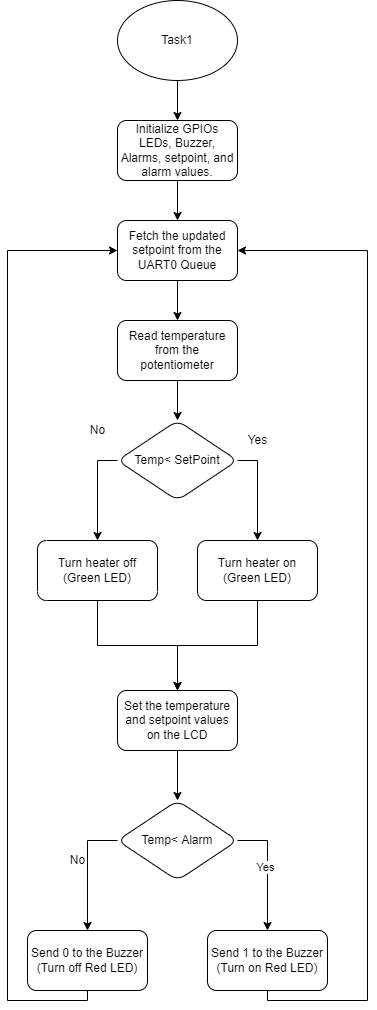
****

Figure - Task1 Flowchart

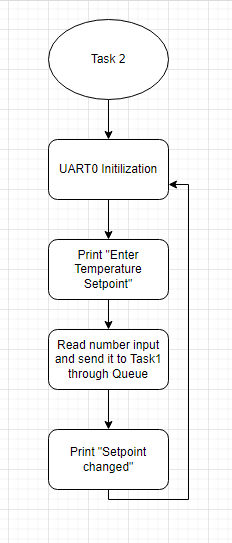


Figure - Task2 Flowchart

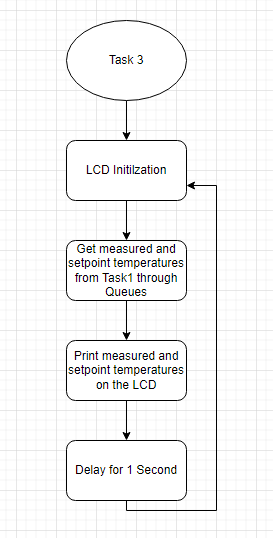


Figure - Task 3 Flowchart

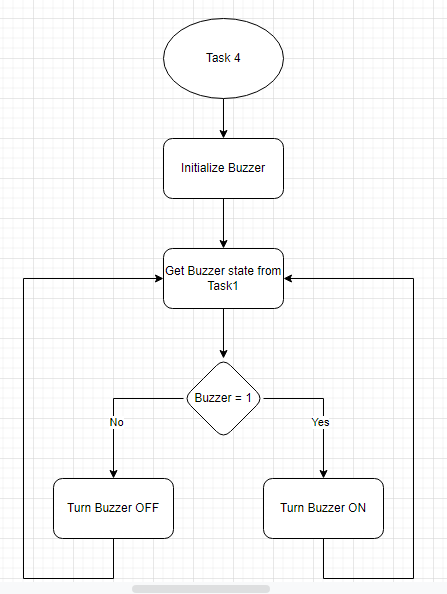


Figure - Task4 Flowchart

# Files & Tasks Description

* Task1
  + The main task that compares received temperature with the setpoint that was entered
  + Turn the Buzzer ON or OFF through Queue
  + Turn the heater ON or OFF through Queue
* Task2
  + UART0 that is used for communication between the TivaC and the computer device
  + Changing the setpoint temperature and send the updated setpoint temperature to main Task1 through the UART0 Queue.
* Task3
  + LCD that is used to show the measured temperature from the potentiometer.
  + Display the setpoint received from the main Task1 through the LCD queue.
* Task4
  + Buzzer that check if task1 has sent through the Buzzer Queue 1 or 0 in order to turn ON or OFF the buzzer respectively.
* gpiosInit
  + Initialize the GPIO ports and UART0
* Main
  + Creates the UART0, LCD, and Buzzer Queues.
  + Initializes the GPIOs and UART0 through gpiosInit function
  + Creates the Tasks mentioned above.

# Code Snippets

## LCD Functions

//Functions to initialize the LCD

void lcdCmd(unsigned char command);

void lcdStart(void);

void lcdData(unsigned char data);

//Function to clear the LCD

void lcdClear (void);

//Move the cursor to the first or second line

void lcdLine(uint8\_t line);

//Display characters on the LCD

void lcdDisplay(char\* name);

## ADC Functions

//Initialize the ADC

void adc\_init(void);

//Function to read the temperature from the potentiometer

unsigned int adc\_read (void);

## toString Function

Used to convert a variable to string in order to be displayed on the LCD.

void toString(char tim, char text[])

{

    // initialize text [0,0]

    for (int j = 0; j < 2; j++)

    {

        text[j] = '0';

    }

    // put numbers in char array

    int i = 2;

    while (tim != 0)

    {

        i--;

        text[i] = ((tim % 10) + '0');

        tim /= 10;

    }

    text[2] = '\0'; // add null terminator

}

## Task1 Function

void TASK1(void \*pvParameters)

{

    typedef struct Message

    {

        char Txt1[4];

        char Txt2[4];

    } AMessage;

    AMessage msg;

    char \*on;

    char \*off;

    on = 1;

    off = 0;

    unsigned char setpoint = 30; //Setpoint default temperature

    unsigned AdcValue; //Value read from the ADC

    unsigned char Temperature; //Temperature that is read from the potentiometer

    float mV; //Value of ADC after being converted to Celcius

    unsigned const char AlarmValue = 50;

    adc\_init(); //ADC Initilization

    while (1)

    {

        xQueueReceive(xUARTQueue, &setpoint, 0); // Receive data

        AdcValue = adc\_read();                   // Read ADC

        mV = 147 - (247 \* AdcValue) / 4096;      // Temp in C

        Temperature = (int)mV;                   // Temp as integer

        if (Temperature < setpoint)

        {                              // If cold

            GPIO\_PORTE\_DATA\_R |= 0x02; // Heater LED ON

        }

        else // If hot

        {

            GPIO\_PORTE\_DATA\_R &= ~0x02; // Heater Led OFF

        }

        toString(Temperature, msg.Txt1);      // Measured value

        toString(setpoint, msg.Txt2);         // setpoint

        xQueueSend(xLCDQueue, &msg, 0);       // Send via Queue

        if (Temperature > AlarmValue)         // Alarm?

            xQueueSend(xBuzzerQueue, &on, 0); // buzzer ON

        else

            xQueueSend(xBuzzerQueue, &off, 0); // Buzzer off

    }

}

# Conclusion

In the end we were able to implement the functionality of the ON/OFF Temperature controller successfully, and learnt how to use several components like the Buzzer and Potentiometer with the FreeRTOS in addition to the Queues in order to be able to implement the project’s functionality.