3/6/2022

Real – Time Embedded Systems

Team#18

Reem Ahmed Sadek
Hager Khaled Karkar
Ahmed Essam ElDin
Zead Hani Ali

Contents

A. Introductory

This project aims to control the temperature using heater which is simulated as a green led and taking feedback from the potentiometer’s voltage to maintain the setpoint by using Queues to send data between tasks and making a communication between user through terminal to change the setpoint temperature and display the setpoint and the measured temperature on the lcd.

If the temperature is below the setpoint the green led will be turned on and if the temperature is above, it will be turned off. A Buzzer is used to indicate an alarm if the temperature exceeded a certain value to notify the user.

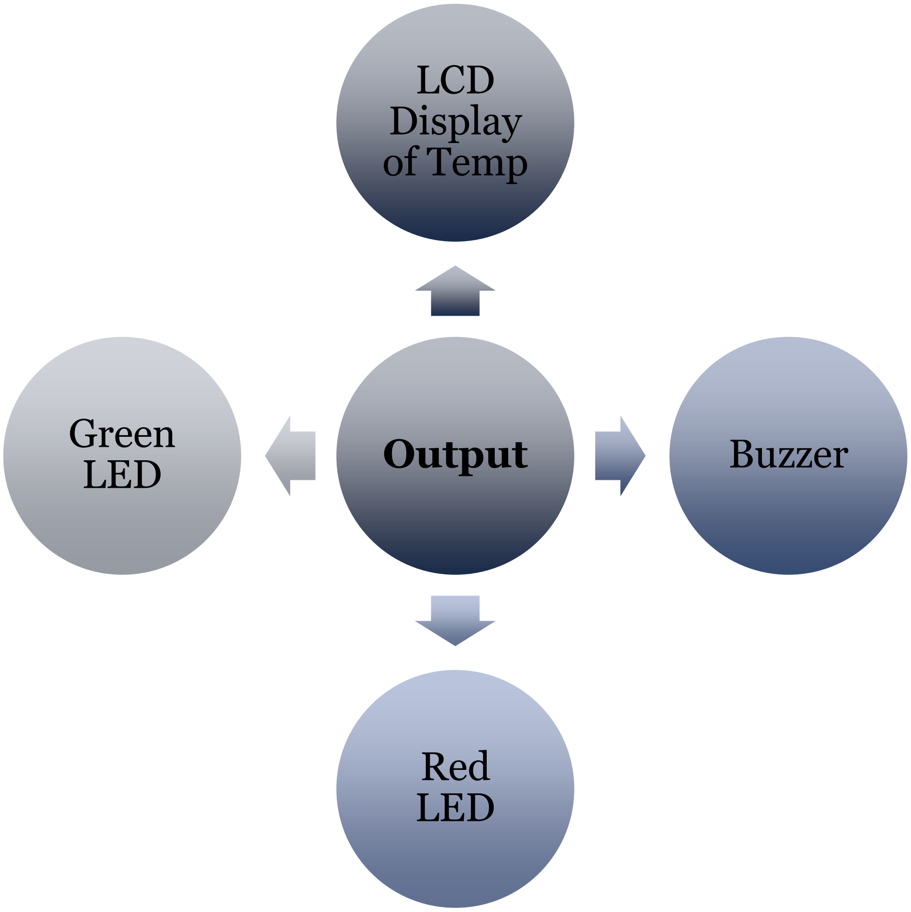
B. Connections & Data

1) Inputs to the system

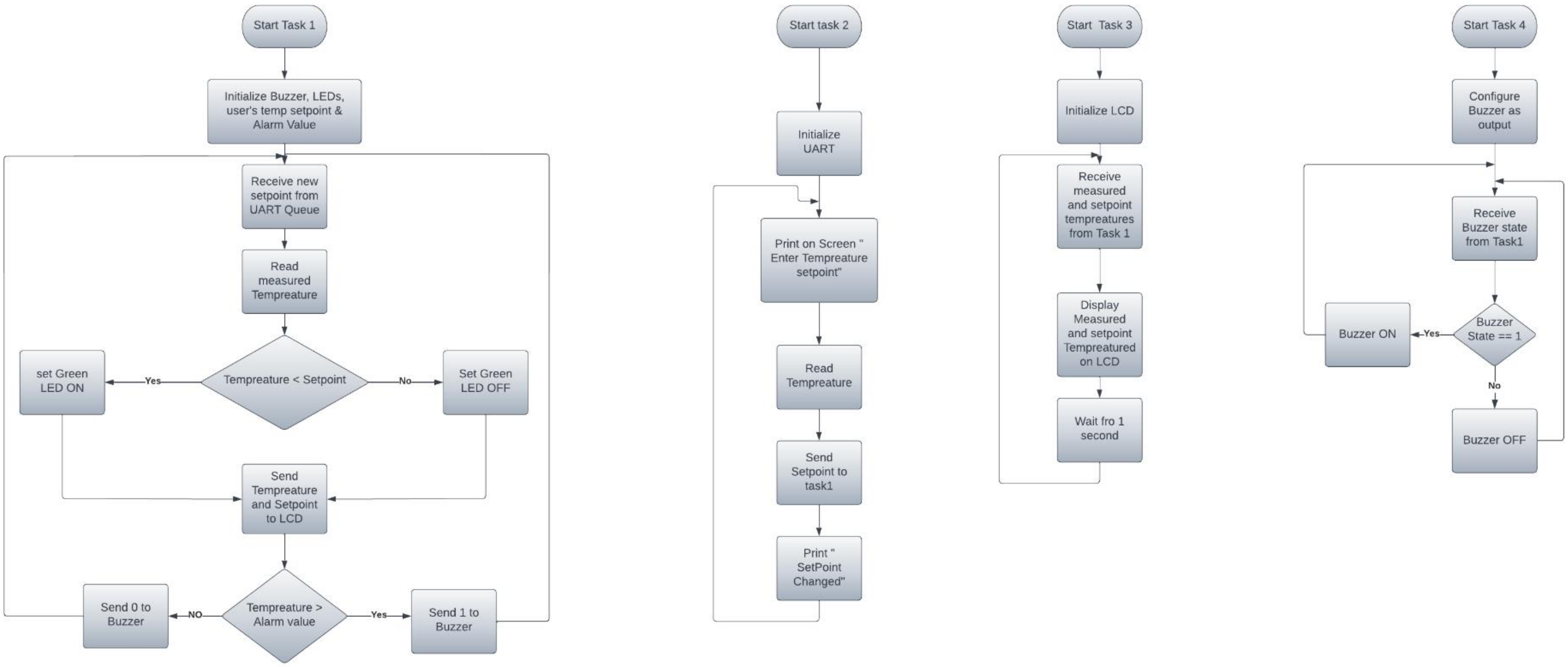
User's Setpoint tempreature

Potentiometer's Voltage

2) Outputs from the system



C. Flow Charts


Start Task 1
Initialize Buzzer, LEDs, user's temp setpoint & Alarm Value
Receive new setpoint from UART Queue
Read measured Tempreature
Tempreature < Setpoint
Yes
set Green LED ON
No
Set Green LED OFF
Send Tempreature and Setpoint to LCD
Tempreature > Alarm value
NO
Send 0 to Buzzer
Yes
Send 1 to Buzzer
Start task 2
Initialize UART
Print on Screen " Enter Tempreature setpoint"
Read Tempreature
Send Setpoint to task1
Print " SetPoint Changed"
Start Task 3
Initialize LCD
Receive measured and setpoint tempreatures from Task 1
Display Measured and setpoint Tempreatured on LCD
Wait fro 1 second
Start Task 4
Configure Buzzer as output
Receive Buzzer state from Task1
Buzzer State == 1
Yes
Buzzer ON
No
Buzzer OFF

D. Files & Tasks Description

1) Files

- **Main.c**
  Main file including the 4 tasks, initialization function main function.
- **LCD.c**
  A Library file for implementing LCD functions.
- **ADC.c**
  A Library file for implementing ADC functions.

2) Functions

- **Main:**
  Where we initialized queues, GPIO ports and the 4 tasks.

- **Tiva_init( ):**
  Initialize GPIO Ports

- **Task 1:**
  Receives etpoint entry from putty, read ADC value, calculate temperature in Celsis and converting temperature to integer to be read and checked comparable to the setpoint.

  As well as turning on Green LED which implies turning the heater ON. If temreature is already higher then the green LED will be turned OFF.

- **Task 2:**
  Reading the entered number and printing it to the UART as well as displaying it to the putty terminal and set the total number to be passed as setpoint.

  It passes the setpoint value to the main task through xUARTQueue.

- **Task 3:**
  Receives the temperature and the setpoint values as well as selecting the LCDs first row.

  Printing the temperature and the measured value as well as selecting LCD's second row. Lastly print the setpoint and the entered value.

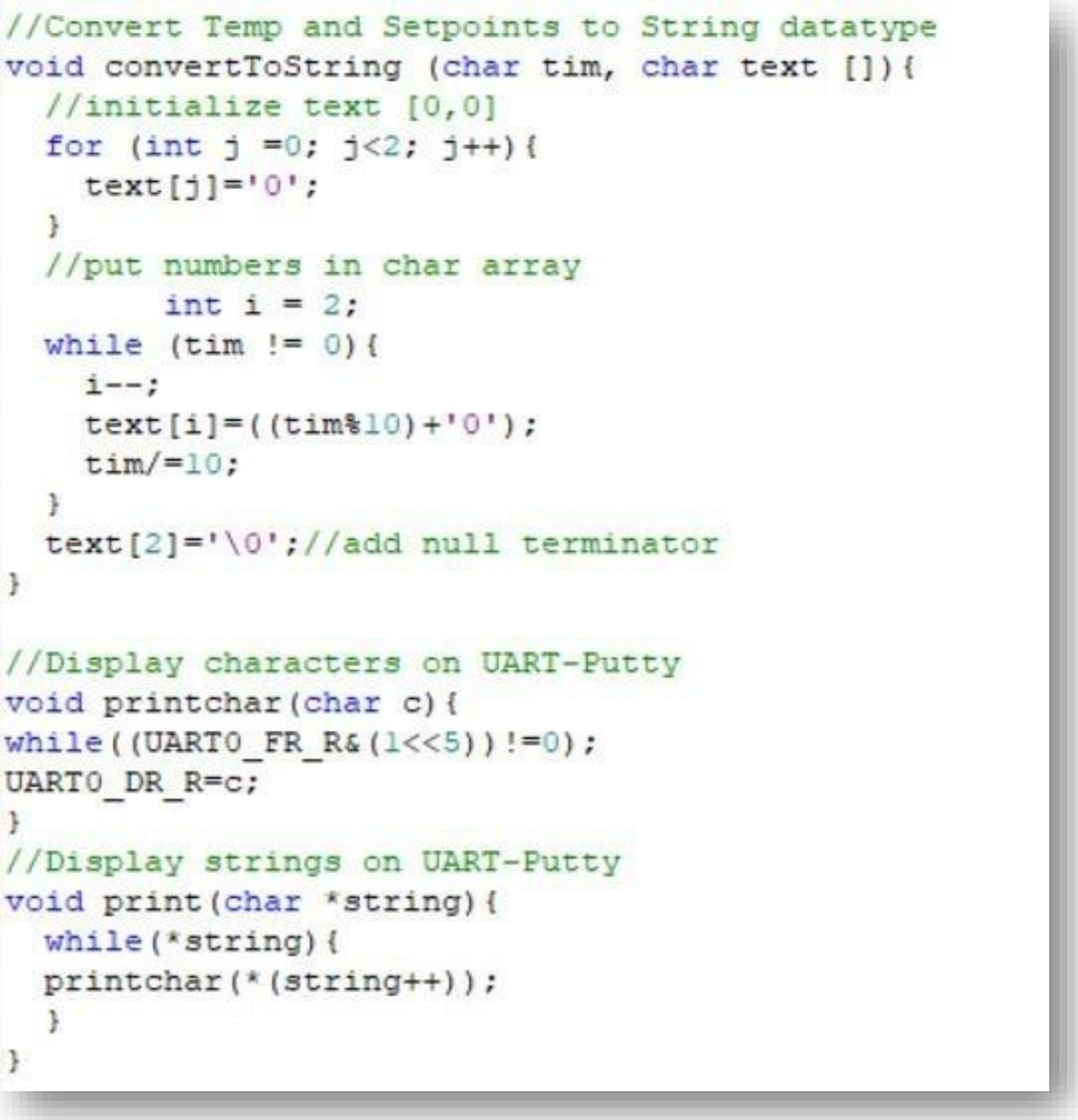
- **Task 4:**
  Used to initialize the buzzer task and receives the buzzer srare from the main task.

  If the sent value is ON then we'll turn on the RED LED and the Buzzer.

- **convertToString( ):**
  Convert Temp and setpoints to string datatype.

E. Code Snippets & Library Functions

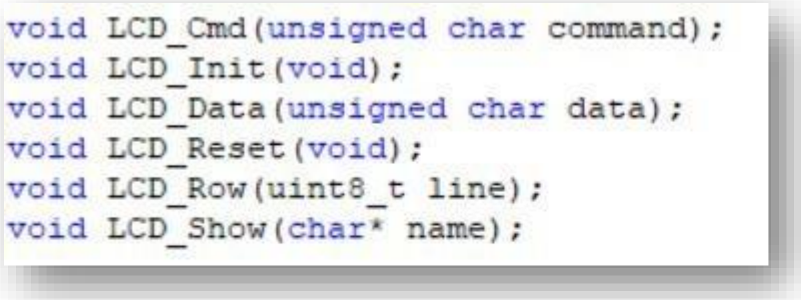
1) A function was made to receive variables and convert them to strings to be sent on the lcd.



```
//Convert Temp and Setpoints to String datatype
void convertToString (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
}

//Display characters on UART-Putty
void printchar(char c){
while((UART0_FR_R&(1<<5))!=0);
UART0_DR_R=c;
}
//Display strings on UART-Putty
void print(char *string){
  while(*string){
  printchar(*(string++));
  }
}
```

2) To use the LCD on tiva through Keil we made a LCD driver to control and display on the LCD using this API functions.

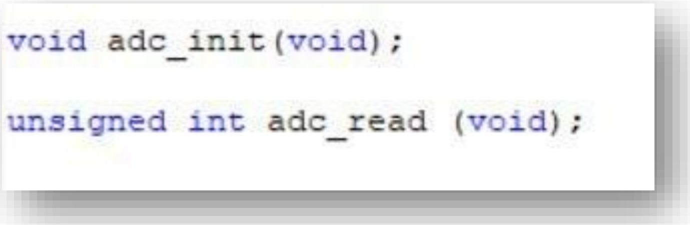
- ✓ LCD_Cmd & LCD_Init & LCD_Data: to initialize the LCD.
- ✓ LCD_Reset: to clear the lcd.
- ✓ LCD_Row: to move the cursor on the first or second line.
- ✓ LCD_Show: to display string on the LCD.



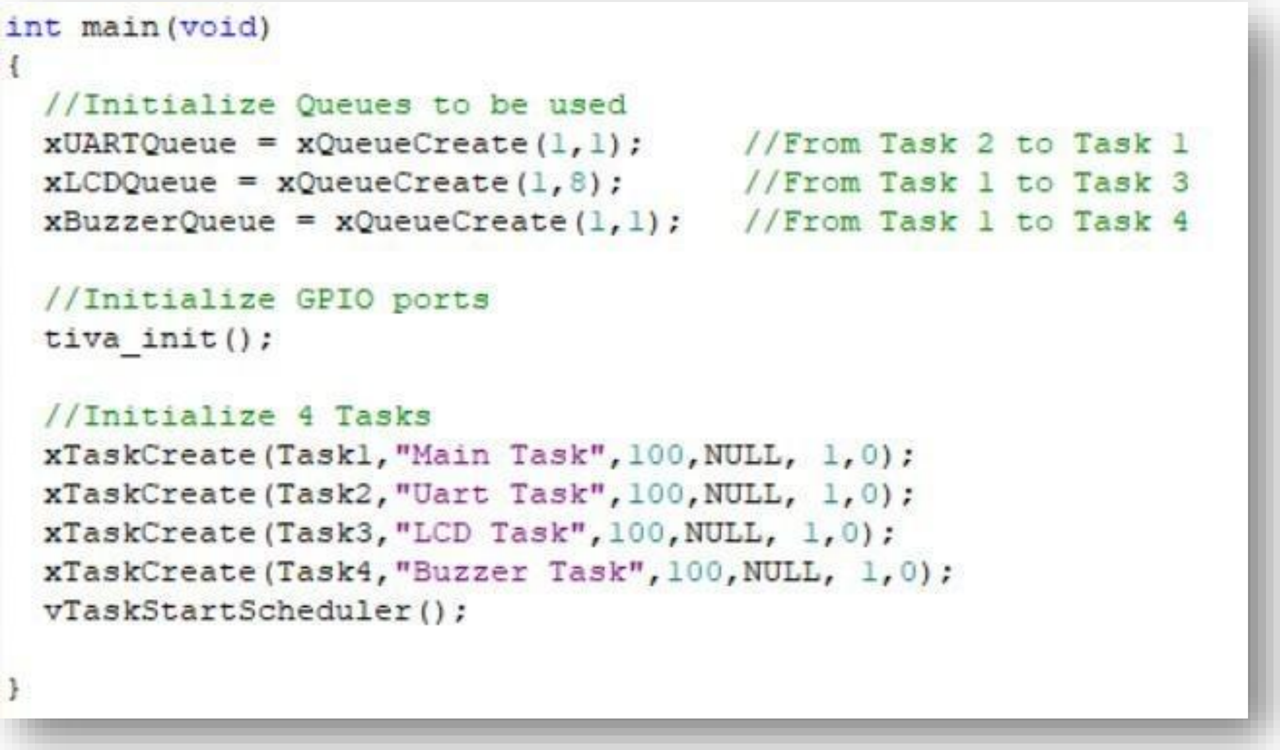
```
void LCD_Cmd(unsigned char command);
void LCD_Init(void);
void LCD_Data(unsigned char data);
void LCD_Reset(void);
void LCD_Row(uint8_t line);
void LCD_Show(char* name);
```

3) To use the Temperature sensor an ADC library was made to read and convert readings from the sensor.

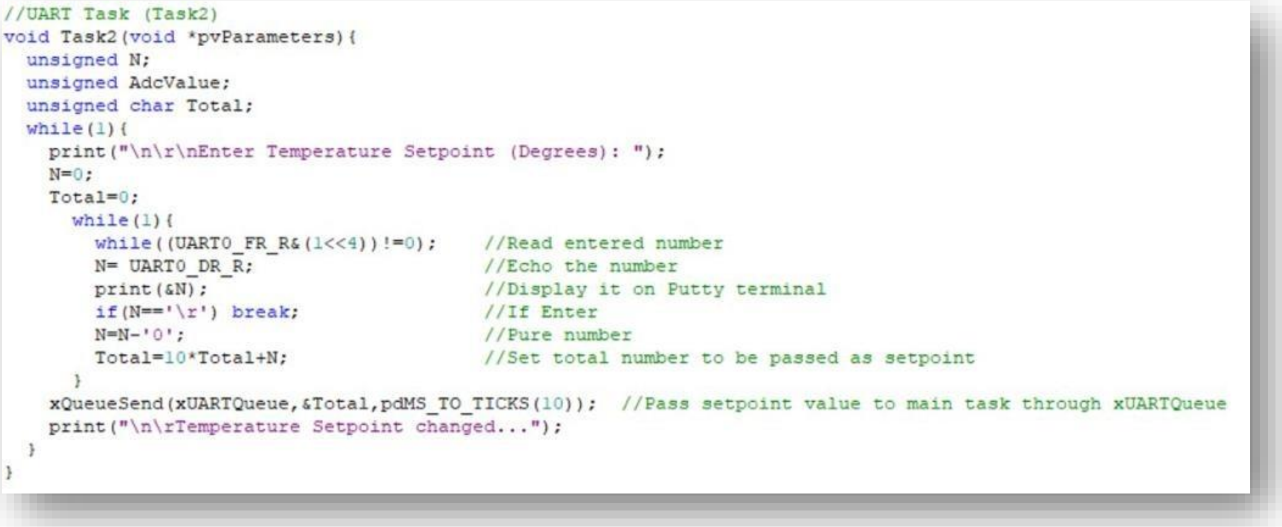
- ✓ adc_init: to initialize the ADC.
- ✓ adc_read: to read potentiometer's temperature.



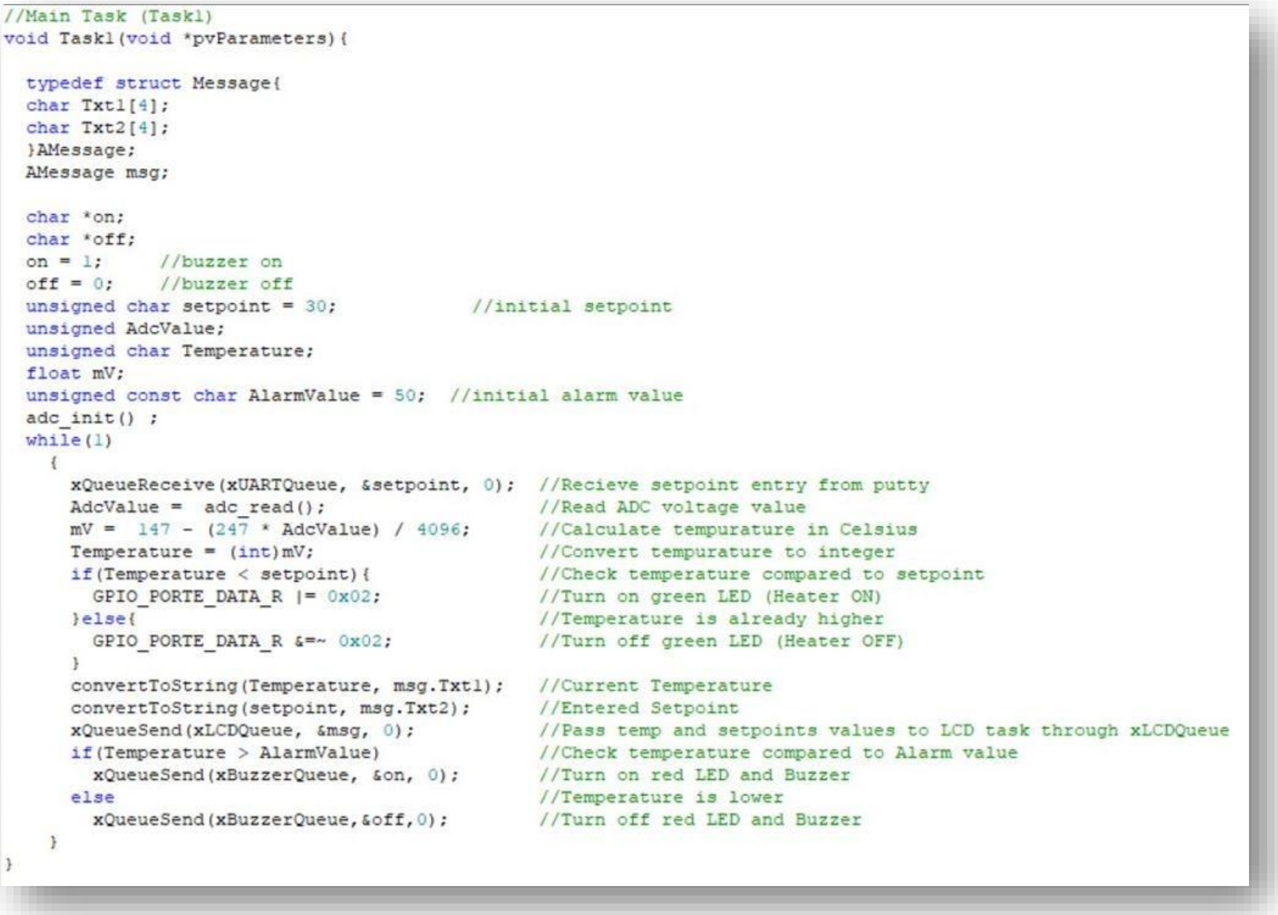
```
void adc_init(void);

unsigned int adc_read (void);
```



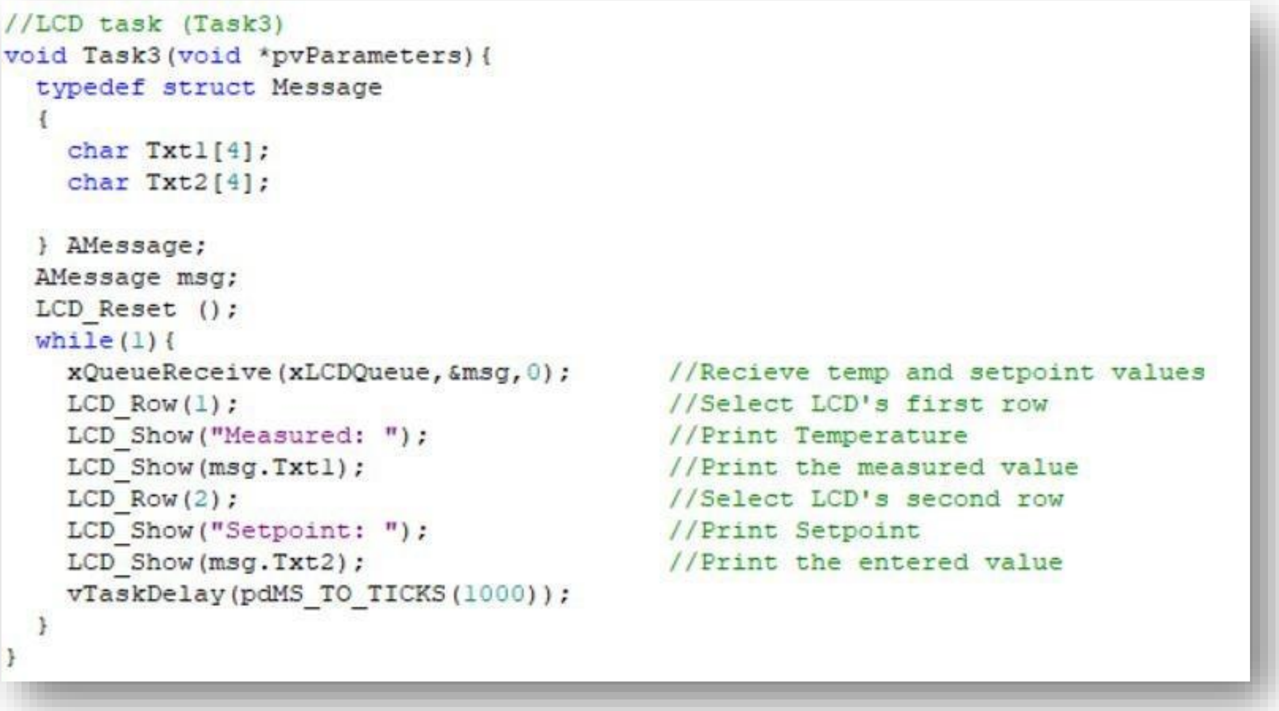
```
int main(void)
{
  //Initialize Queues to be used
  xUARTQueue = xQueueCreate(1,1);     //From Task 2 to Task 1
  xLCDQueue = xQueueCreate(1,8);      //From Task 1 to Task 3
  xBuzzerQueue = xQueueCreate(1,1);   //From Task 1 to Task 4

  //Initialize GPIO ports
  tiva_init();

  //Initialize 4 Tasks
  xTaskCreate(Task1,"Main Task",100,NULL, 1,0);
  xTaskCreate(Task2,"Uart Task",100,NULL, 1,0);
  xTaskCreate(Task3,"LCD Task",100,NULL, 1,0);
  xTaskCreate(Task4,"Buzzer Task",100,NULL, 1,0);
  vTaskStartScheduler();

}
```



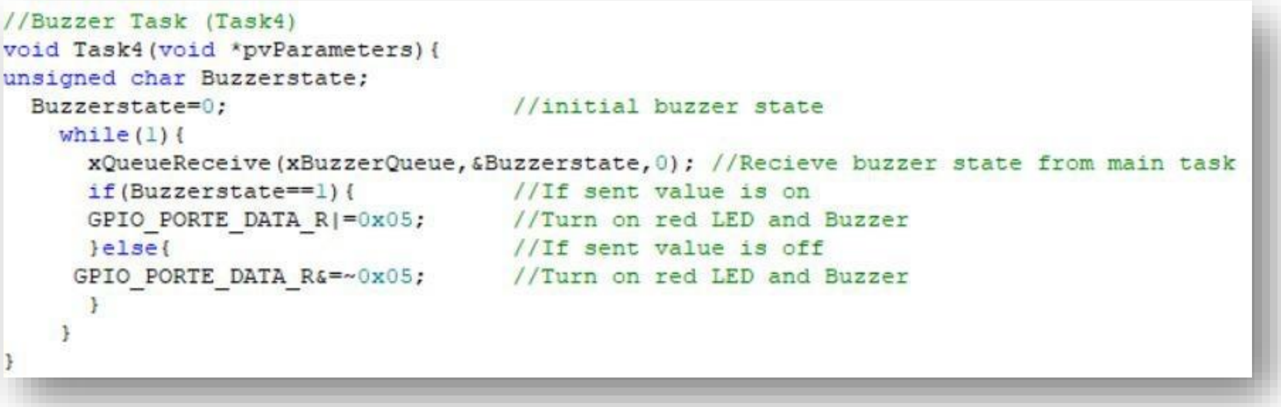
```
//UART Task (Task2)
void Task2(void *pvParameters){
  unsigned N;
  unsigned AdcValue;
  unsigned char Total;
  while(1){
    print("\n\r\nEnter Temperature Setpoint (Degrees): ");
    N=0;
    Total=0;
      while(1){
        while((UART0_FR_R&(1<<4))!=0);    //Read entered number
        N= UART0_DR_R;                    //Echo the number
        print(&N);                        //Display it on Putty terminal
        if(N=='\r') break;                //If Enter
        N=N-'0';                          //Pure number
        Total=10*Total+N;                 //Set total number to be passed as setpoint
      }
    xQueueSend(xUARTQueue,&Total,pdMS_TO_TICKS(10));  //Pass setpoint value to main task through xUARTQueue
    print("\n\rTemperature Setpoint changed...");
  }
}
```



```
//Main Task (Task1)
void Task1(void *pvParameters){

  typedef struct Message{
  char Txt1[4];
  char Txt2[4];
  }AMessage;
  AMessage msg;

  char *on;
  char *off;
  on = 1;     //buzzer on
  off = 0;    //buzzer off
  unsigned char setpoint = 30;            //initial setpoint
  unsigned AdcValue;
  unsigned char Temperature;
  float mV;
  unsigned const char AlarmValue = 50;  //initial alarm value
  adc_init() ;
  while(1)
    {
      xQueueReceive(xUARTQueue, &setpoint, 0);  //Recieve setpoint entry from putty
      AdcValue =  adc_read();                   //Read ADC voltage value
      mV =  147 - (247 * AdcValue) / 4096;      //Calculate tempurature in Celsius
      Temperature = (int)mV;                    //Convert tempurature to integer
      if(Temperature < setpoint){               //Check temperature compared to setpoint
        GPIO_PORTE_DATA_R |= 0x02;              //Turn on green LED (Heater ON)
      }else{                                    //Temperature is already higher
        GPIO_PORTE_DATA_R &=~ 0x02;             //Turn off green LED (Heater OFF)
      }
      convertToString(Temperature, msg.Txt1);   //Current Temperature
      convertToString(setpoint, msg.Txt2);      //Entered Setpoint
      xQueueSend(xLCDQueue, &msg, 0);           //Pass temp and setpoints values to LCD task through xLCDQueue
      if(Temperature > AlarmValue)              //Check temperature compared to Alarm value
        xQueueSend(xBuzzerQueue, &on, 0);       //Turn on red LED and Buzzer
      else                                      //Temperature is lower
        xQueueSend(xBuzzerQueue,&off,0);        //Turn off red LED and Buzzer
    }
}
```

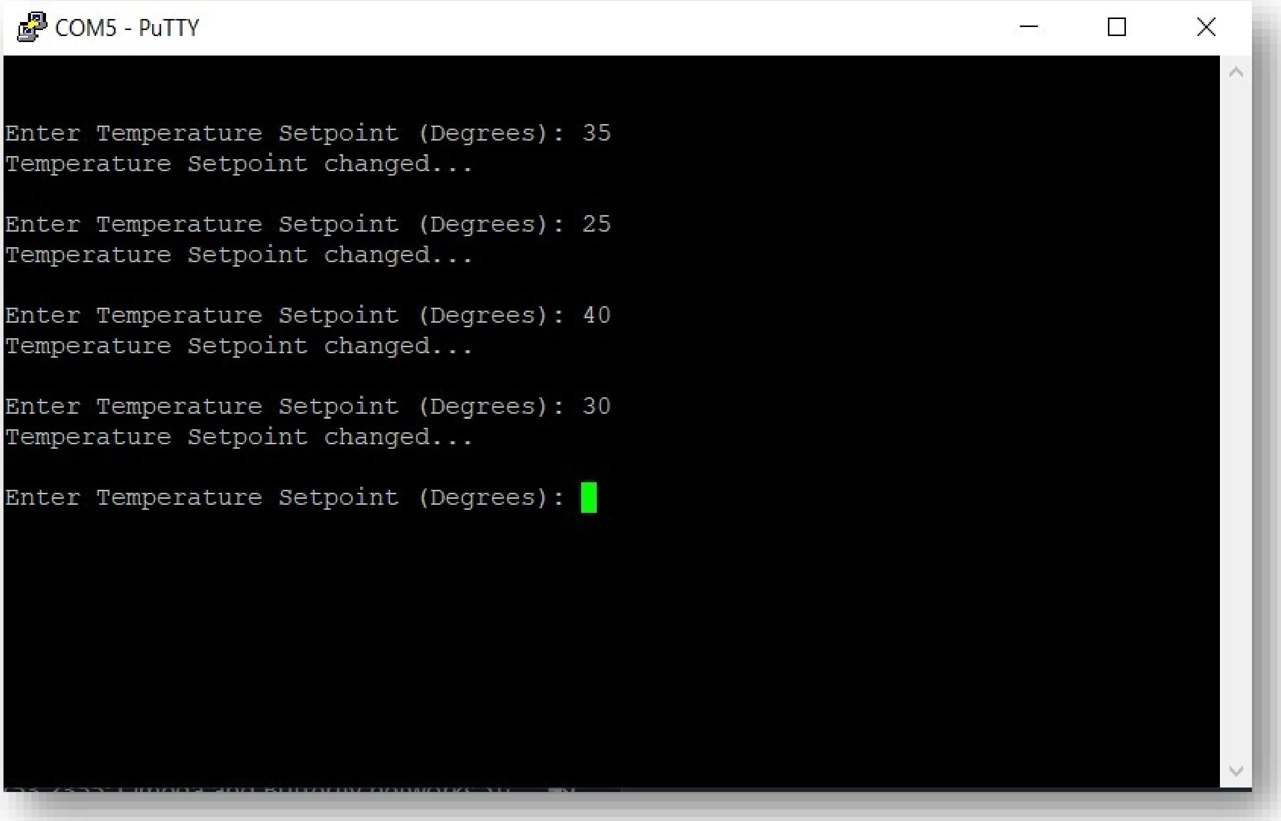


```
//LCD task (Task3)
void Task3(void *pvParameters){
  typedef struct Message
  {
    char Txt1[4];
    char Txt2[4];

  } AMessage;
  AMessage msg;
  LCD_Reset ();
  while(1){
    xQueueReceive(xLCDQueue,&msg,0);      //Recieve temp and setpoint values
    LCD_Row(1);                           //Select LCD's first row
    LCD_Show("Measured: ");               //Print Temperature
    LCD_Show(msg.Txt1);                   //Print the measured value
    LCD_Row(2);                           //Select LCD's second row
    LCD_Show("Setpoint: ");               //Print Setpoint
    LCD_Show(msg.Txt2);                   //Print the entered value
    vTaskDelay(pdMS_TO_TICKS(1000));
  }
}
```



```
//Buzzer Task (Task4)
void Task4(void *pvParameters){
unsigned char Buzzerstate;
  Buzzerstate=0;                   //initial buzzer state
    while(1){
      xQueueReceive(xBuzzerQueue,&Buzzerstate,0); //Recieve buzzer state from main task
      if(Buzzerstate==1){          //If sent value is on
      GPIO_PORTE_DATA_R|=0x05;     //Turn on red LED and Buzzer
      }else{                       //If sent value is off
     GPIO_PORTE_DATA_R&=~0x05;     //Turn on red LED and Buzzer
      }
    }
}
```

F. Terminal Output



```
Enter Temperature Setpoint (Degrees): 35
Temperature Setpoint changed...

Enter Temperature Setpoint (Degrees): 25
Temperature Setpoint changed...

Enter Temperature Setpoint (Degrees): 40
Temperature Setpoint changed...

Enter Temperature Setpoint (Degrees): 30
Temperature Setpoint changed...

Enter Temperature Setpoint (Degrees):
```

G. Video & Source Code

https://drive.google.com/drive/folders/1seMktyz66e_OWYONyguYc4U9bTAaaZKi?usp=sharing