CSE 411

Real-Time and Embedded system design

Lab project

Submitted by: Team 15

Ahmed Mamdouh	**19P5326**
Hussam Elsayed	**18P6670**
Nadin Ahmed Ali	**18P6472**
Mirna Al-Amir	**15P8150**
Andrew Bolus	**18P7917**

Submitted to:

Dr. Sherif Hammad

Eng . Ahmed Mohamed

Table of Contents

.... 1

1. Introduction 3

2. Wiring and components.. 4

3. Flow chart 7

4. code.. 8

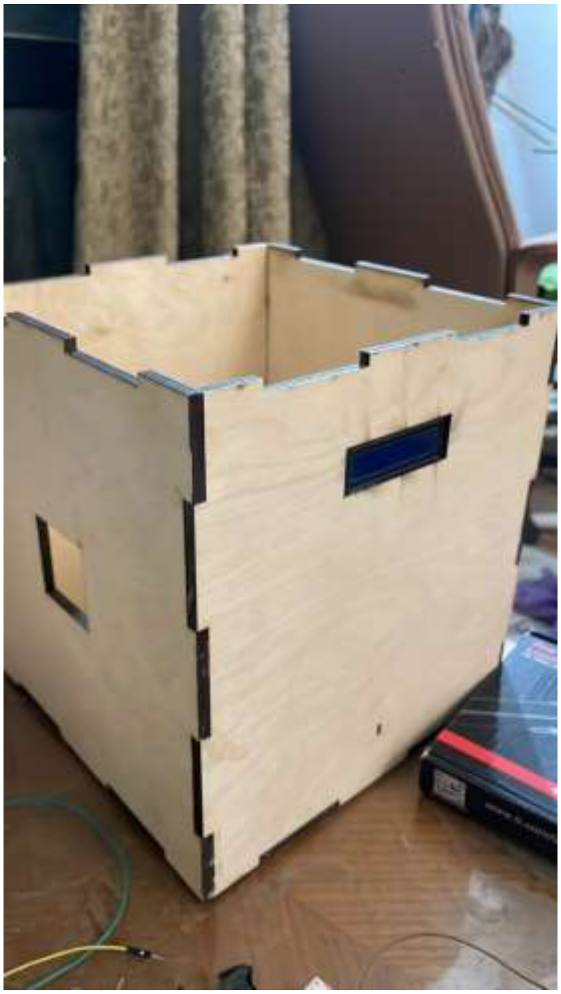
5. Working video 12

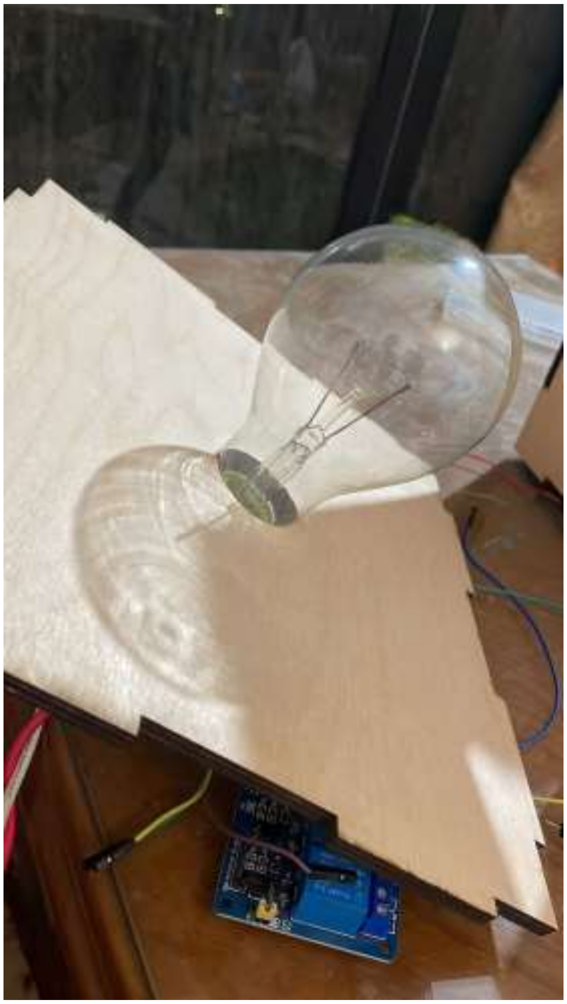
6. Contribution table... 13

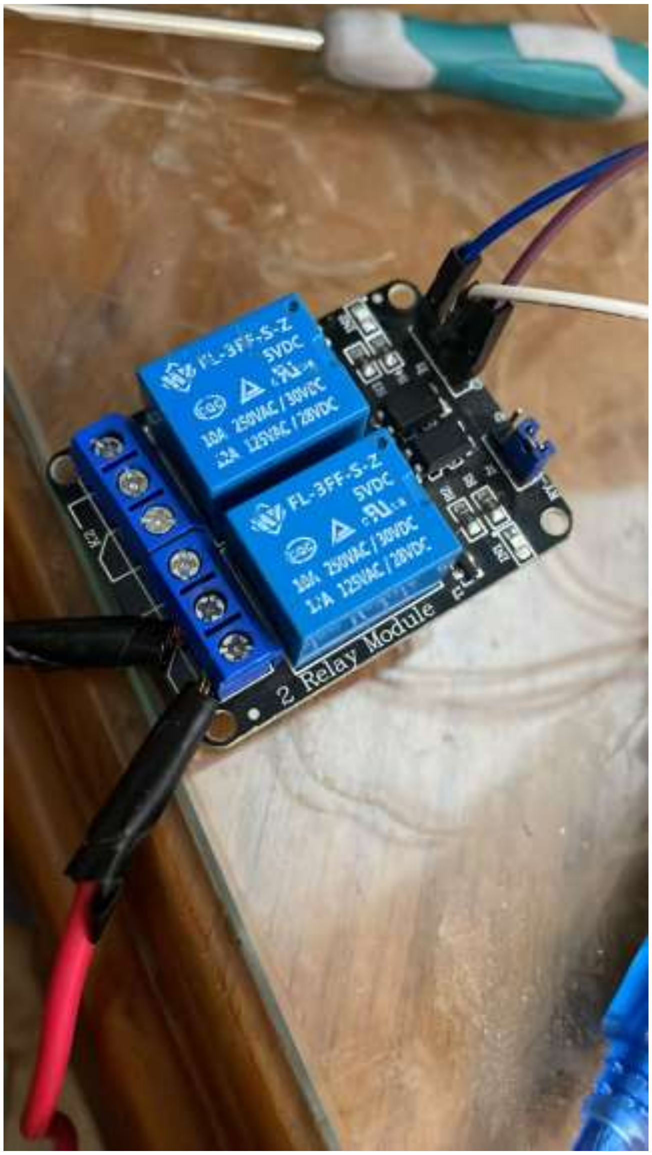
1. Introduction

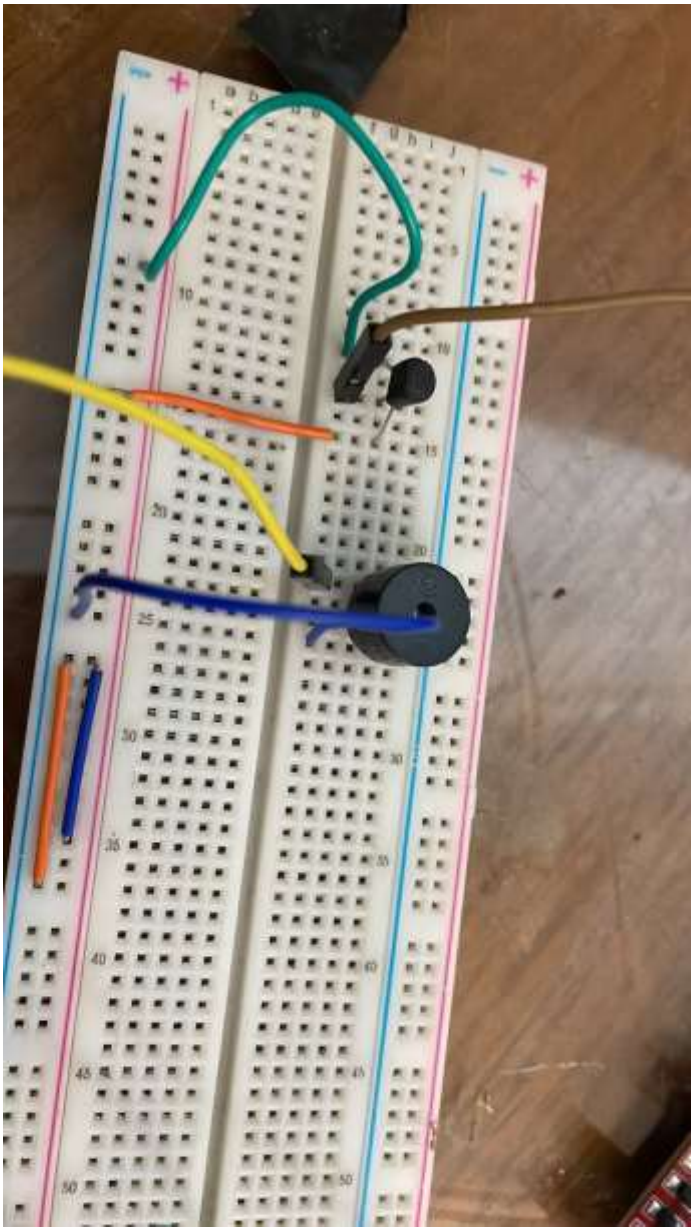
The goal of this project is to create and programme a simple On-Off temperature controller. A potentiometer is utilised as a temperature sensor, an LED is used as a heater, and the setpoint is entered via the keyboard.The heater (LED) starts working whenever the user sets a setpoint. Until the temperature sensor is replaced.To put the heater to sleep, the (potentiometer) detects the setpoint value. The current setpoint, as well as the one chosen, is on an LCD, the data is shown in real time.

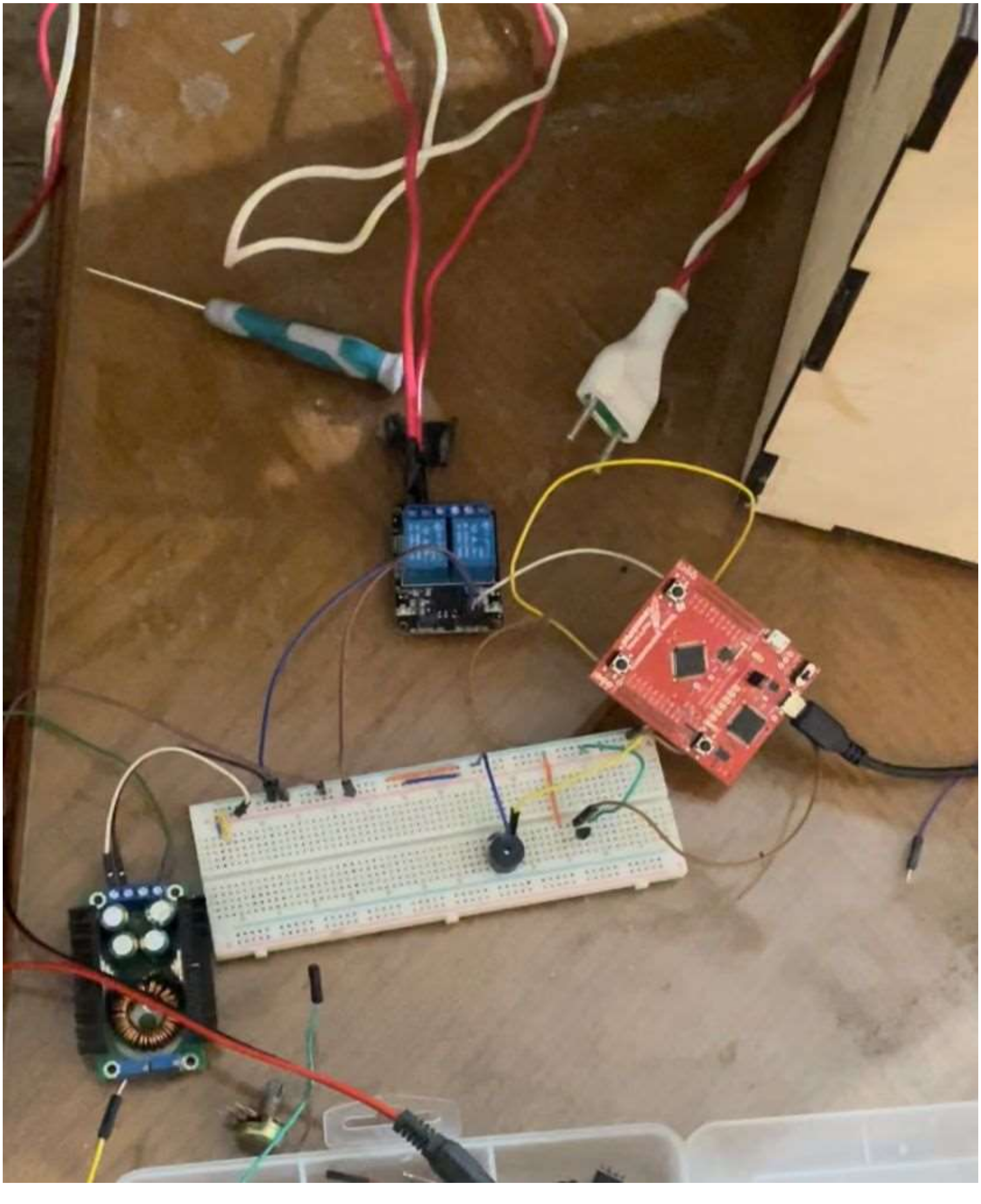
2. Wiring and components



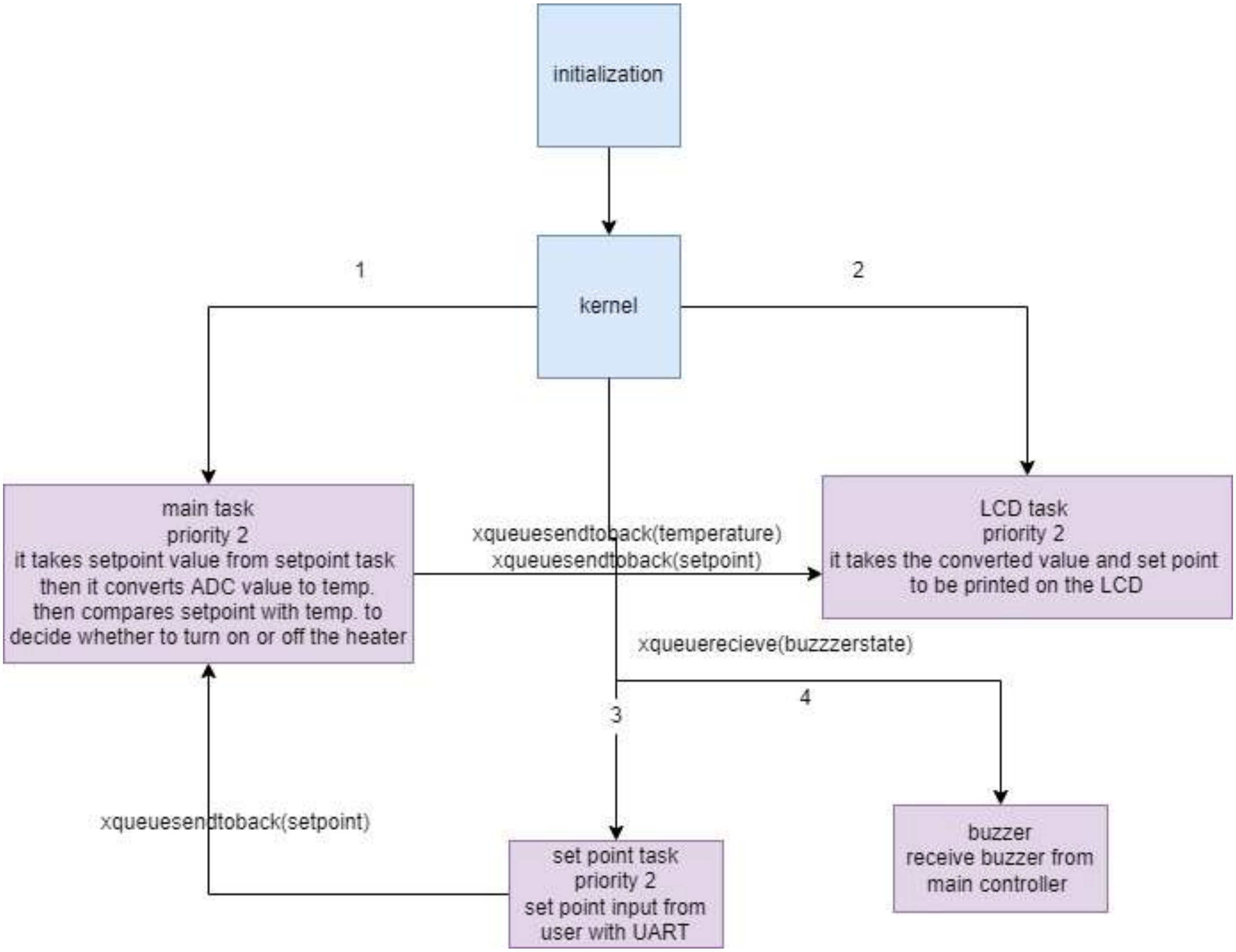



FL-3FF-S-Z
5VDC
10A 250VAC / 30VDC
12A 125VAC / 28VDC
2 Relay Module

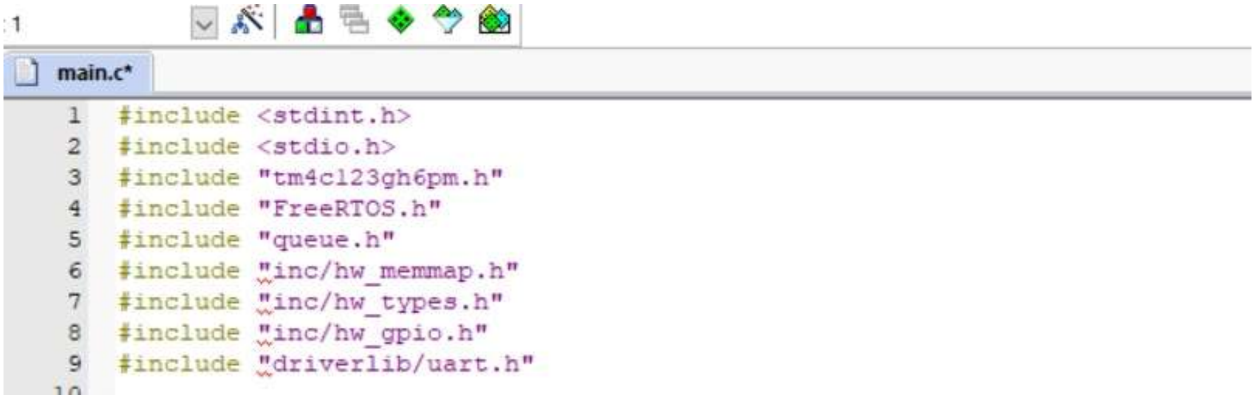




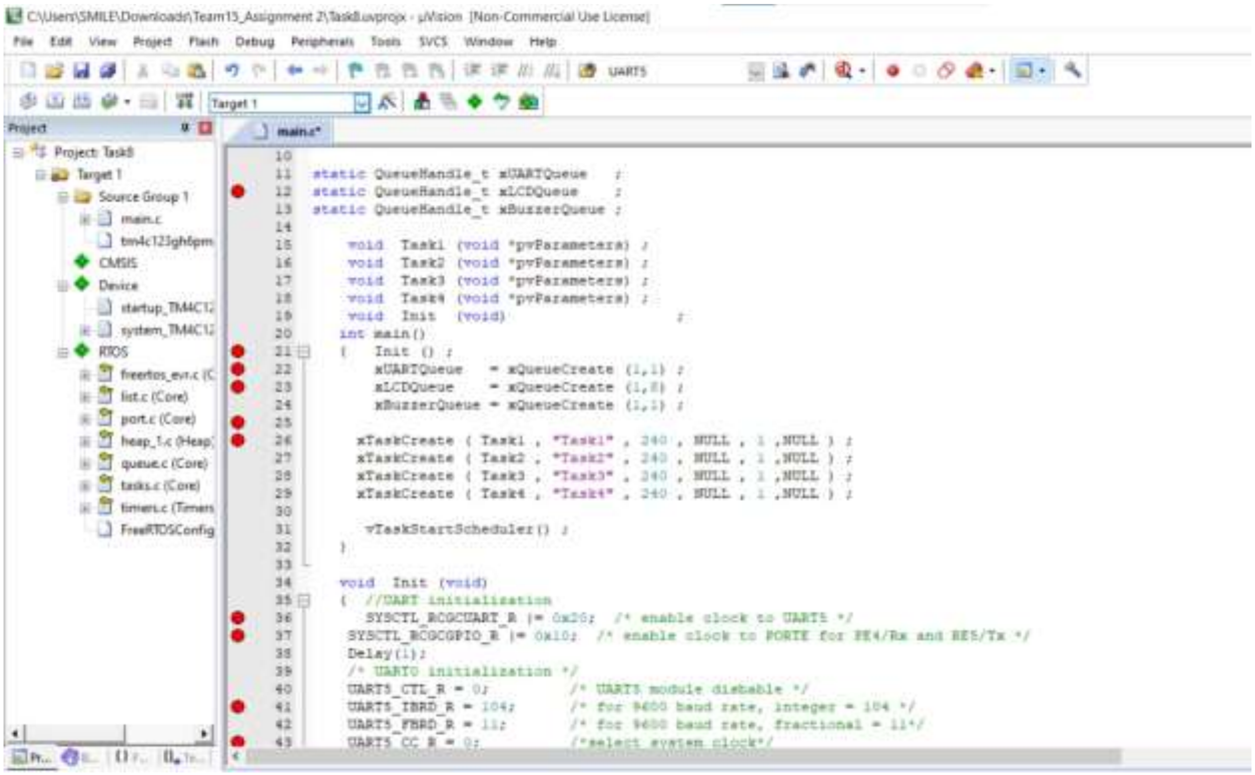
3. Flow chart


initialization
kernel
1
2
main task
priority 2
it takes setpoint value from setpoint task
then it converts ADC value to temp.
then compares setpoint with temp. to
decide whether to turn on or off the heater
xqueuesendtoback(temperature)
xqueuesendtoback(setpoint)
LCD task
priority 2
it takes the converted value and set point
to be printed on the LCD
xqueuerecieve(buzzzerstate)
3
4
xqueuesendtoback(setpoint)
set point task
priority 2
set point input from
user with UART
buzzer
receive buzzer from
main controller

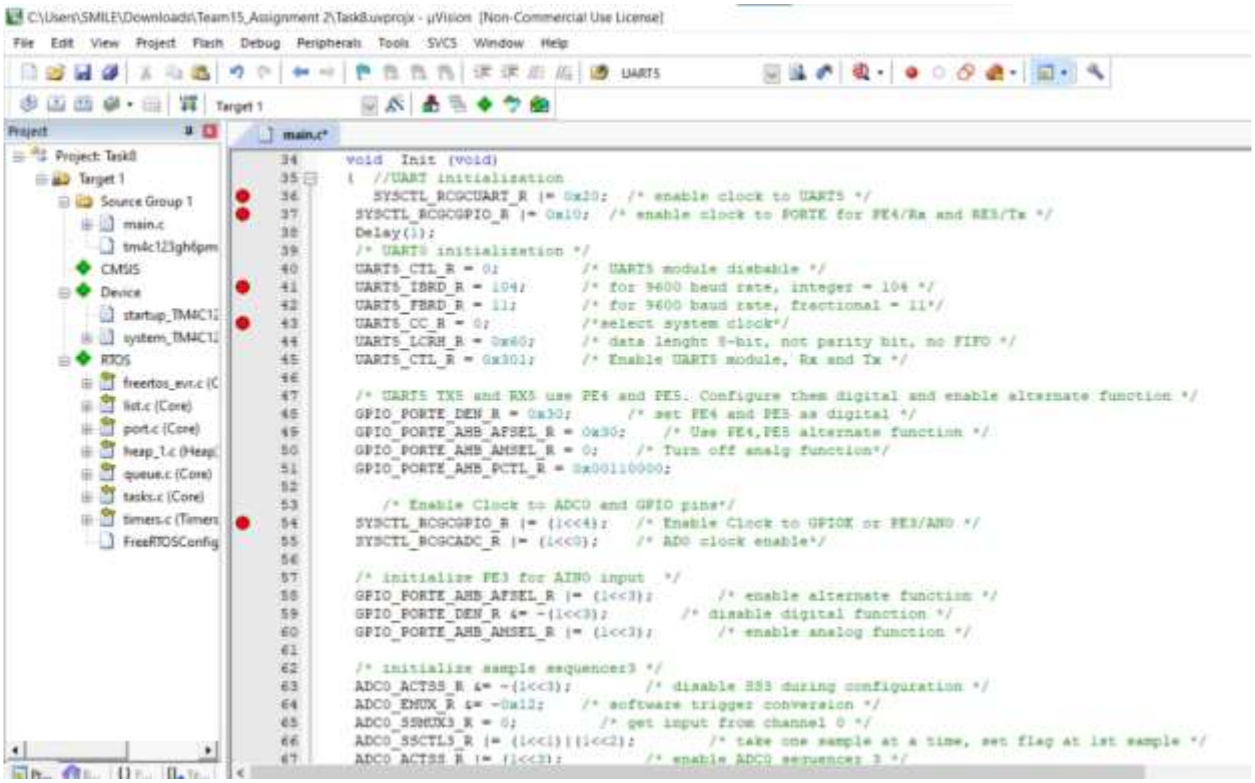
4. code



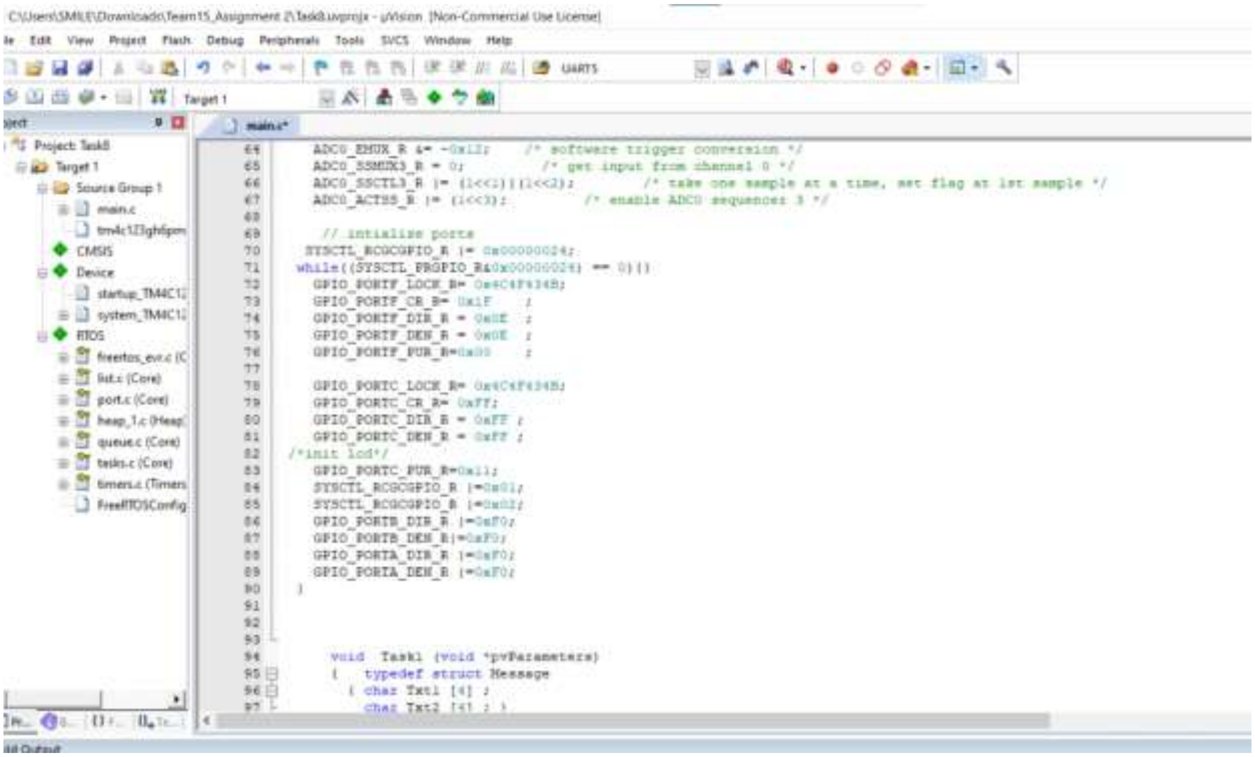
```c
#include <stdint.h>
#include <stdio.h>
#include "tm4c123gh6pm.h"
#include "FreeRTOS.h"
#include "queue.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_gpio.h"
#include "driverlib/uart.h"
```



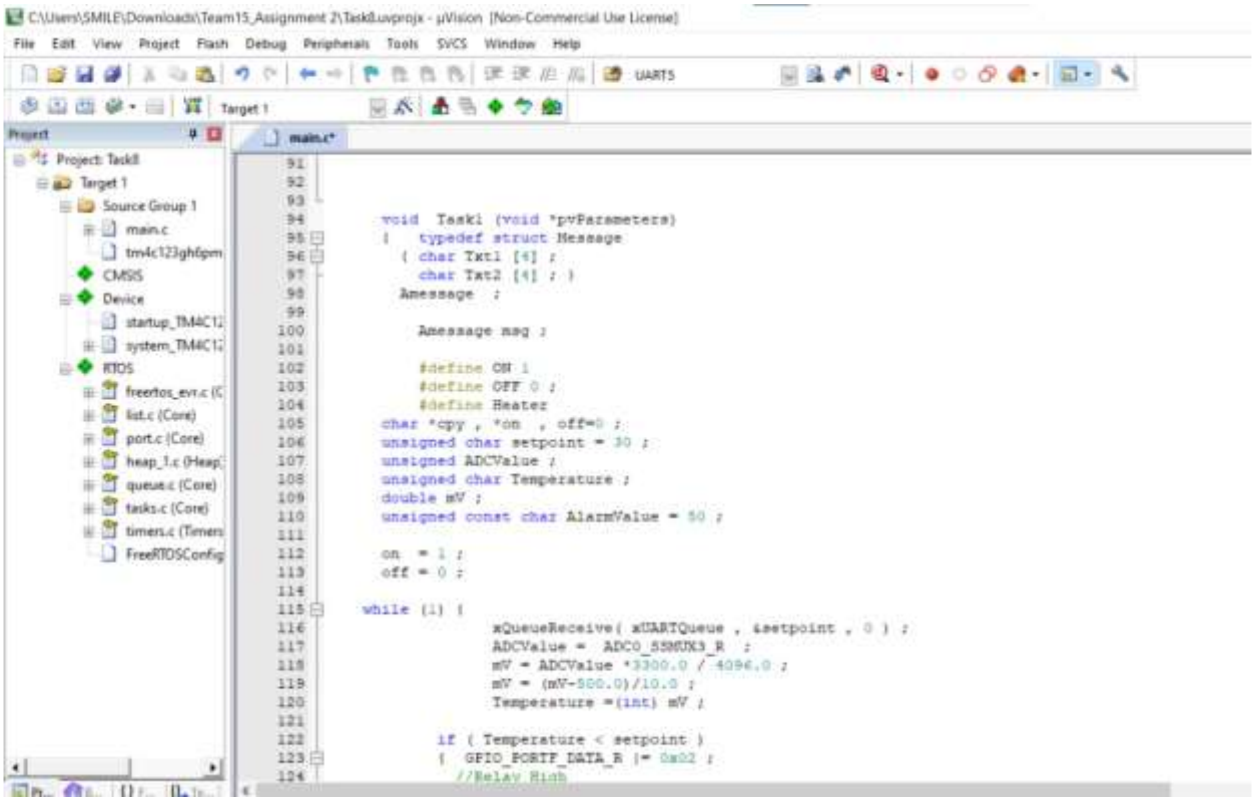
```c

static QueueHandle_t xUARTQueue    ;
static QueueHandle_t xLCDQueue     ;
static QueueHandle_t xBuzzerQueue ;

    void  Task1 (void *pvParameters) ;
    void  Task2 (void *pvParameters) ;
    void  Task3 (void *pvParameters) ;
    void  Task4 (void *pvParameters) ;
    void  Init  (void)                        ;
   int main()
   {   Init () ;
       xUARTQueue    = xQueueCreate (1,1) ;
       xLCDQueue     = xQueueCreate (1,8) ;
       xBuzzerQueue = xQueueCreate (1,1) ;

     xTaskCreate ( Task1 , "Task1" , 240 , NULL , 1 ,NULL ) ;
     xTaskCreate ( Task2 , "Task2" , 240 , NULL , 1 ,NULL ) ;
     xTaskCreate ( Task3 , "Task3" , 240 , NULL , 1 ,NULL ) ;
     xTaskCreate ( Task4 , "Task4" , 240 , NULL , 1 ,NULL ) ;

      vTaskStartScheduler() ;
   }

   void  Init (void)
   {  //UART initialization
      SYSCTL_RCGCUART_R |= 0x20;  /* enable clock to UART5 */
    SYSCTL_RCGCGPIO_R |= 0x10;  /* enable clock to PORTE for PE4/Rx and RE5/Tx */
    Delay(1);
    /* UART0 initialization */
    UART5_CTL_R = 0;          /* UART5 module disbable */
    UART5_IBRD_R = 104;       /* for 9600 baud rate, integer = 104 */
    UART5_FBRD_R = 11;        /* for 9600 baud rate, fractional = 11*/
    UART5_CC_R = 0;           /*select system clock*/
```


C:\Users\SMILE\Downloads\Team15_Assignment 2\Task8.uvprojx - µVision [Non-Commercial Use License]

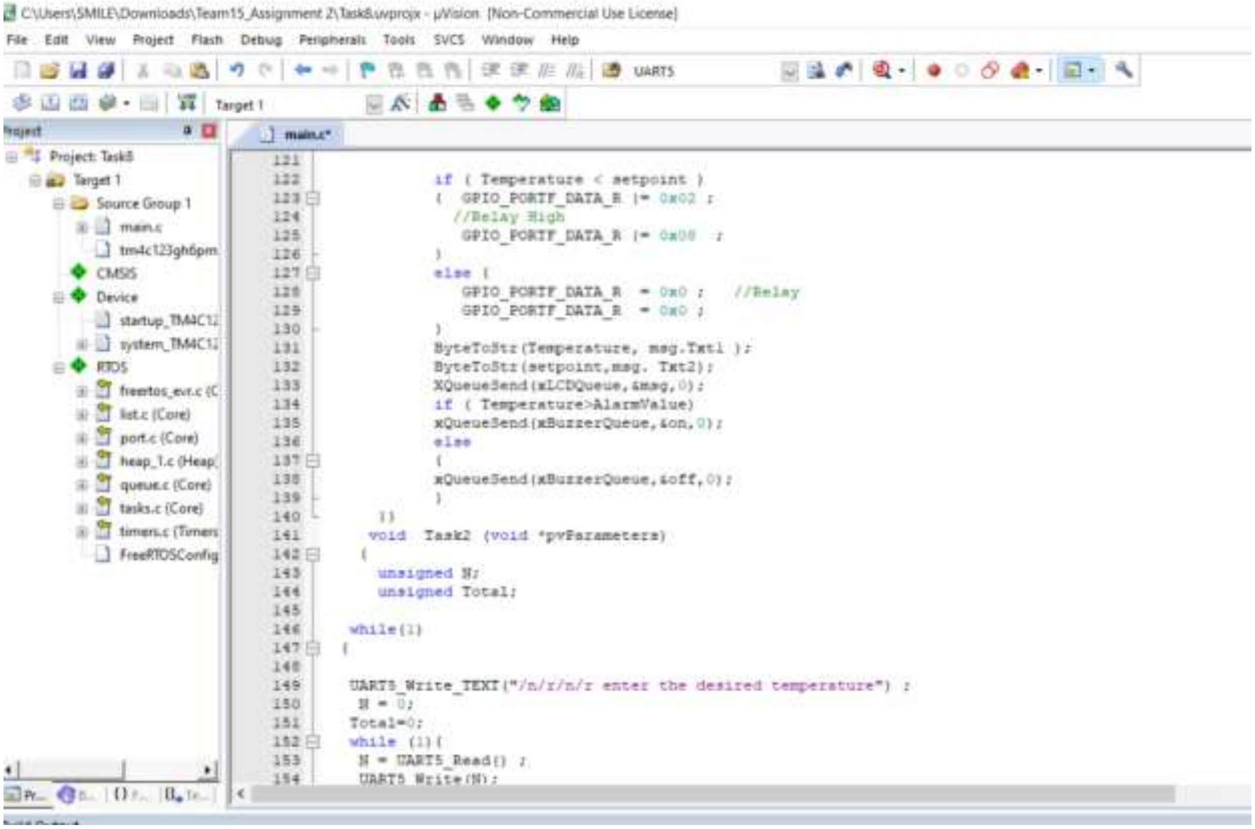
```
void  Init (void)
{  //UART initialization
   SYSCTL_RCGCUART_R |= 0x20;  /* enable clock to UART5 */
 SYSCTL_RCGCGPIO_R |= 0x10;  /* enable clock to PORTE for PE4/Rx and RE5/Tx */
 Delay(1);
 /* UART0 initialization */
 UART5_CTL_R = 0;           /* UART5 module disbable */
 UART5_IBRD_R = 104;        /* for 9600 baud rate, integer = 104 */
 UART5_FBRD_R = 11;         /* for 9600 baud rate, fractional = 11*/
 UART5_CC_R = 0;            /*select system clock*/
 UART5_LCRH_R = 0x60;       /* data lenght 8-bit, not parity bit, no FIFO */
 UART5_CTL_R = 0x301;       /* Enable UART5 module, Rx and Tx */

 /* UART5 TX5 and RX5 use PE4 and PE5. Configure them digital and enable alternate function */
 GPIO_PORTE_DEN_R = 0x30;       /* set PE4 and PE5 as digital */
 GPIO_PORTE_AHB_AFSEL_R = 0x30;     /* Use PE4,PE5 alternate function */
 GPIO_PORTE_AHB_AMSEL_R = 0;    /* Turn off analg function*/
 GPIO_PORTE_AHB_PCTL_R = 0x00110000;

    /* Enable Clock to ADC0 and GPIO pins*/
 SYSCTL_RCGCGPIO_R |= (1<<4);   /* Enable Clock to GPIOE or PE3/AN0 */
 SYSCTL_RCGCADC_R |= (1<<0);    /* AD0 clock enable*/

 /* initialize PE3 for AIN0 input  */
 GPIO_PORTE_AHB_AFSEL_R |= (1<<3);        /* enable alternate function */
 GPIO_PORTE_DEN_R &= ~(1<<3);          /* disable digital function */
 GPIO_PORTE_AHB_AMSEL_R |= (1<<3);        /* enable analog function */

 /* initialize sample sequencer3 */
 ADC0_ACTSS_R &= ~(1<<3);          /* disable SS3 during configuration */
 ADC0_EMUX_R &= ~0x12;     /* software trigger conversion */
 ADC0_SSMUX3_R = 0;           /* get input from channel 0 */
 ADC0_SSCTL3_R |= (1<<1)|(1<<2);           /* take one sample at a time, set flag at 1st sample */
 ADC0_ACTSS_R |= (1<<3);          /* enable ADC0 sequencer 3 */
```


C:\Users\SMILE\Downloads\Team15_Assignment 2\Task8.uvprojx - µVision [Non-Commercial Use License]

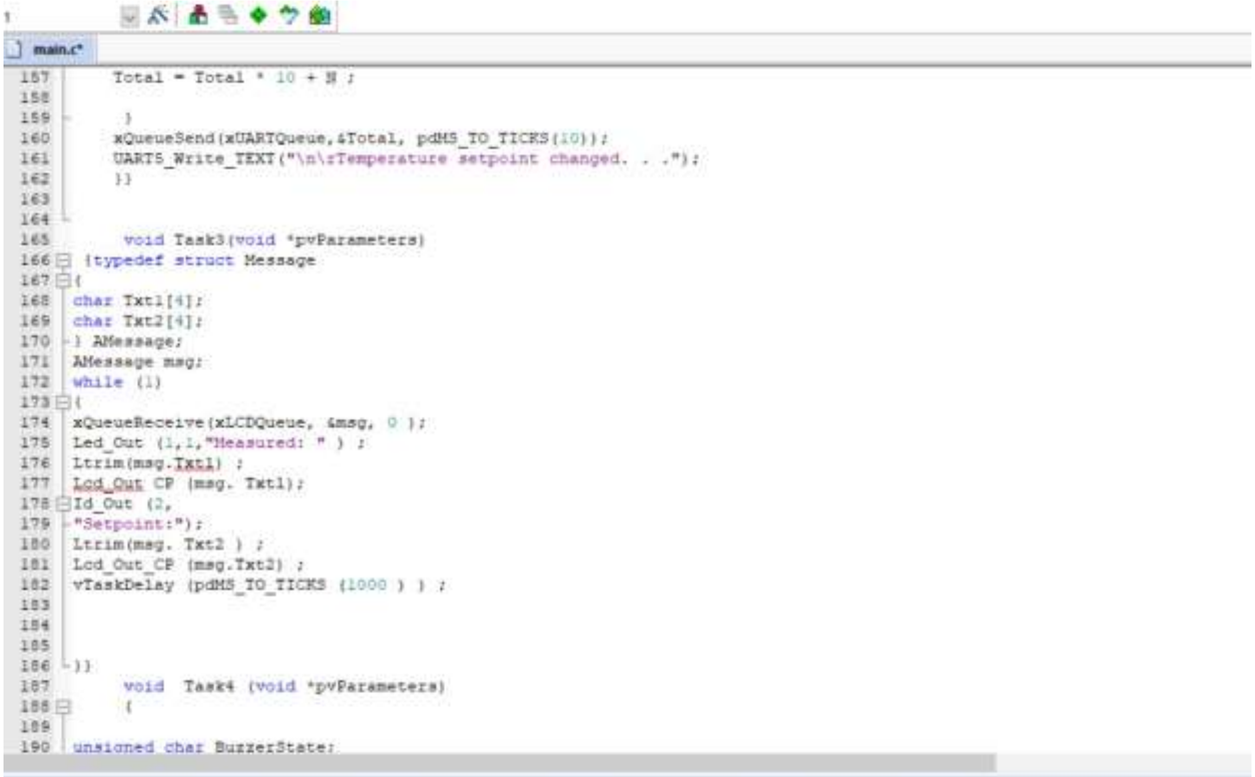
```
    ADC0_EMUX_R &= ~0x12;     /* software trigger conversion */
    ADC0_SSMUX3_R = 0;           /* get input from channel 0 */
    ADC0_SSCTL3_R |= (1<<1)|(1<<2);           /* take one sample at a time, set flag at 1st sample */
    ADC0_ACTSS_R |= (1<<3);          /* enable ADC0 sequencer 3 */

     // intialize ports
   SYSCTL_RCGCGPIO_R |= 0x00000024;
  while((SYSCTL_PRGPIO_R&0x00000024) == 0){}
    GPIO_PORTF_LOCK_R= 0x4C4F434B;
    GPIO_PORTF_CR_R= 0x1F    ;
    GPIO_PORTF_DIR_R = 0x0E  ;
    GPIO_PORTF_DEN_R = 0x0E  ;
    GPIO_PORTF_PUR_R=0x00    ;

    GPIO_PORTC_LOCK_R= 0x4C4F434B;
    GPIO_PORTC_CR_R= 0xFF;
    GPIO_PORTC_DIR_R = 0xFF ;
    GPIO_PORTC_DEN_R = 0xFF ;
 /*init lcd*/
    GPIO_PORTC_PUR_R=0x11;
    SYSCTL_RCGCGPIO_R |=0x01;
    SYSCTL_RCGCGPIO_R |=0x02;
    GPIO_PORTB_DIR_R |=0xF0;
    GPIO_PORTB_DEN_R|=0xF0;
    GPIO_PORTA_DIR_R |=0xF0;
    GPIO_PORTA_DEN_R |=0xF0;
  }



       void  Task1 (void *pvParameters)
       {    typedef struct Message
         { char Txt1 [4] ;
           char Txt2 [4] ; }
```



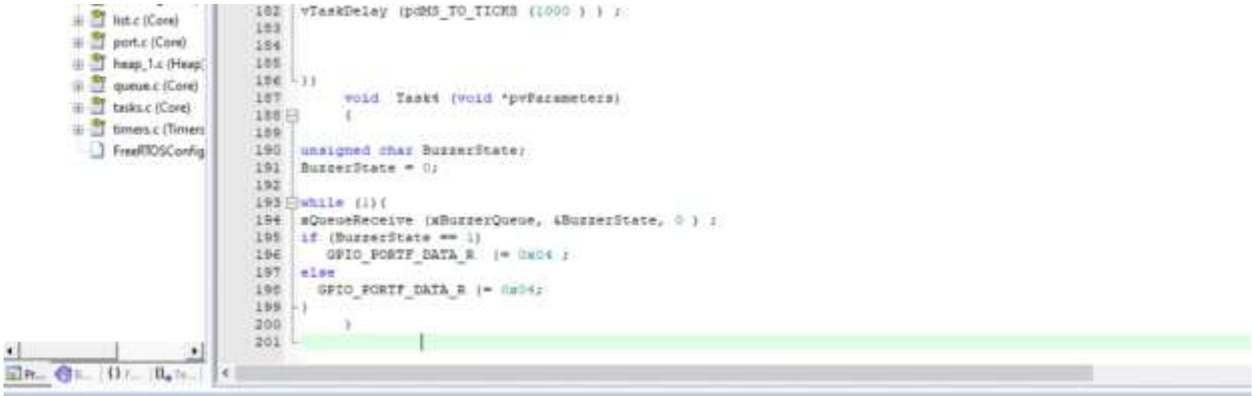
```



    void  Task1 (void *pvParameters)
    {   typedef struct Message
      { char Txt1 [4] ;
        char Txt2 [4] ; }
     Amessage  ;

       Amessage msg ;

       #define ON 1
       #define OFF 0 ;
       #define Heater
    char *cpy , *on  , off=0 ;
    unsigned char setpoint = 30 ;
    unsigned ADCValue ;
    unsigned char Temperature ;
    double mV ;
    unsigned const char AlarmValue = 50 ;

    on  = 1 ;
    off = 0 ;

  while (1) {
                 xQueueReceive( xUARTQueue , &setpoint , 0 ) ;
                 ADCValue =  ADC0_SSMUX3_R  ;
                 mV = ADCValue *3300.0 / 4096.0 ;
                 mV = (mV-500.0)/10.0 ;
                 Temperature =(int) mV ;

          if ( Temperature < setpoint )
          {  GPIO_PORTF_DATA_R |= 0x02 ;
            //Relay High
```



```

            if ( Temperature < setpoint )
            {  GPIO_PORTF_DATA_R |= 0x02 ;
              //Relay High
               GPIO_PORTF_DATA_R |= 0x08  ;
            }
            else {
               GPIO_PORTF_DATA_R  = 0x0 ;    //Relay
               GPIO_PORTF_DATA_R  = 0x0 ;
            }
            ByteToStr(Temperature, msg.Txt1 );
            ByteToStr(setpoint,msg. Txt2);
            XQueueSend(xLCDQueue,&msg,0);
            if ( Temperature>AlarmValue)
            xQueueSend(xBuzzerQueue,&on,0);
            else
            {
            xQueueSend(xBuzzerQueue,&off,0);
            }
     }}
   void  Task2 (void *pvParameters)
  {
    unsigned N;
    unsigned Total;

 while(1)
 {

 UART5_Write_TEXT("/n/r/n/r enter the desired temperature") ;
  N = 0;
 Total=0;
 while (1){
  N = UART5_Read() ;
  UART5_Write(N);
```



```
     Total = Total * 10 + N ;

      }
     xQueueSend(xUARTQueue,&Total, pdMS_TO_TICKS(10));
     UART5_Write_TEXT("\n\rTemperature setpoint changed. . .");
     }}


      void Task3(void *pvParameters)
 {typedef struct Message
{
char Txt1[4];
char Txt2[4];
} AMessage;
AMessage msg;
while (1)
{
xQueueReceive(xLCDQueue, &msg, 0 );
Led_Out (1,1,"Measured: " ) ;
Ltrim(msg.Txt1) ;
Lcd_Out CP (msg. Txt1);
Id_Out (2,
"Setpoint:");
Ltrim(msg. Txt2 ) ;
Lcd_Out_CP (msg.Txt2) ;
vTaskDelay (pdMS_TO_TICKS (1000 ) ) ;



}}
      void  Task4 (void *pvParameters)
      {

unsigned char BuzzerState;
```



```
vTaskDelay (pdMS_TO_TICKS (1000 ) ) ;



})
      void  Task4 (void *pvParameters)
      {

unsigned char BuzzerState;
BuzzerState = 0;

while (1){
xQueueReceive (xBuzzerQueue, &BuzzerState, 0 ) ;
if (BuzzerState == 1)
   GPIO_PORTF_DATA_R  |= 0x04 ;
else
  GPIO_PORTF_DATA_R |= 0x04;
}
      }

```

5. Working video

https://drive.google.com/drive/folders/162V3u2fq_rRpAv4UIl2C3wI0bI197hL_?usp=sharing

6. Contribution table

Name	ID	Task	Percentage
Ahmed Mamdouh	19P5326	integration	22%
Hussam Elsayed	18P6670	Main code	22%
Nadin Ahmed Ali	18P6472	UART communication	20%
Mirna Al-Amir	15P8150	wiring	18%
Andrew Bolus	18P7917	ADC	18%