CSE 347

Temperature On/OFF Controller Embedded Project

Submitted By:-

Ahmed Sherif Hosny	17P8028

Submitted to: -

Dr. Sherif Hammad
Eng. Ahmed Nofal
Eng. Mohamed Tarek
Eng. Ahmed Tawfik
Eng. Norhan Ghoniem

Table of Contents

Table of figures

Demonstration Video

https://drive.google.com/drive/folders/1PyxxMzkPavQAVDA4MGcnLlBMUjYa-vAR?usp=sharing

Introduction

This project aims to control the temperature using heater and taking feedback from the temperature sensor 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 heater 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.

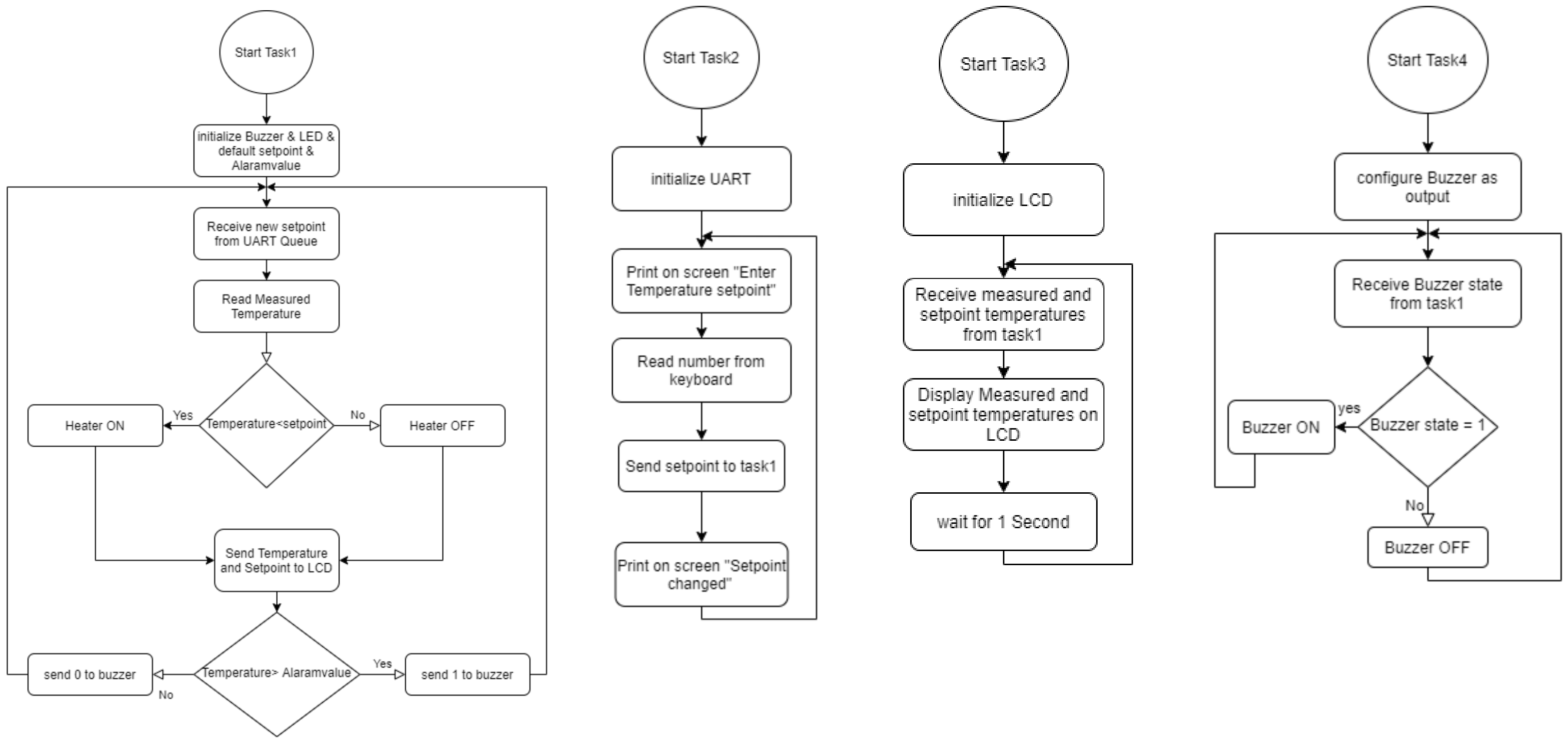
Inputs

- Temperature sensor
- User's setpoint temperature

Output

- LCD display of temperatures
- Buzzer
- Led
- Heater

Flow chart



Start Task1
initialize Buzzer & LED & default setpoint & Alaramvalue
Receive new setpoint from UART Queue
Read Measured Temperature
Temperature<setpoint
Yes
Heater ON
No
Heater OFF
Send Temperature and Setpoint to LCD
Temperature> Alaramvalue
Yes
send 1 to buzzer
No
send 0 to buzzer
Start Task2
initialize UART
Print on screen "Enter Temperature setpoint"
Read number from keyboard
Send setpoint to task1
Print on screen "Setpoint changed"
Start Task3
initialize LCD
Receive measured and setpoint temperatures from task1
Display Measured and setpoint temperatures on LCD
wait for 1 Second
Start Task4
configure Buzzer as output
Receive Buzzer state from task1
Buzzer state = 1
yes
Buzzer ON
No
Buzzer OFF

Task Diagram

As seen, we got for tasks in which all of them have the same priority so the scheduler will run each task for a time slice then move to the next task to run it and then the cycle repeats.

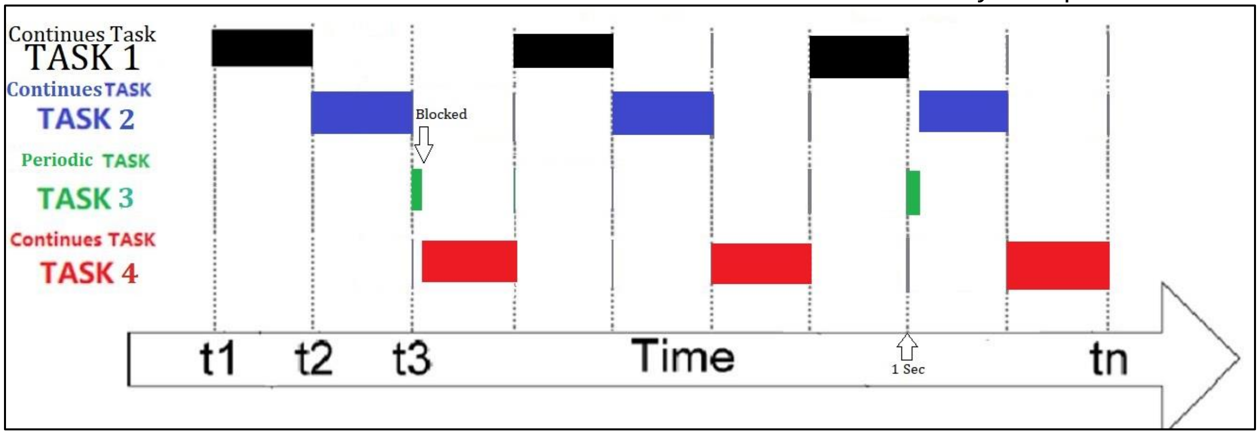


Figure 1

Description of Files & Tasks

Files:

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

Functions:

Task1: Main Controller function for comparing temperature with setpoint and set on or off the heater and the buzzer through the Queue send and recieve.

Task2: UART Controller function for displaying the communication between System and laptop and changing the setpoint temperature and send the new setpoint to task1 through UART queue.

Task3: LCD Controller Function for displaying on LCD the measured temperature from the sensor and the setpoint by receiving the data from task1 through LCD Queue.

Task4: Buzzer controller function to check if task1 sent on the queue data to set on the buzzer or not by receiving data from task1 through Buzzer Queue.

GPIO_init: An initialization function to initialize GPIO ports and UART.

Main: main function to create queues and tasks for the program and start the scheduler.

Hardware Connections

LCD

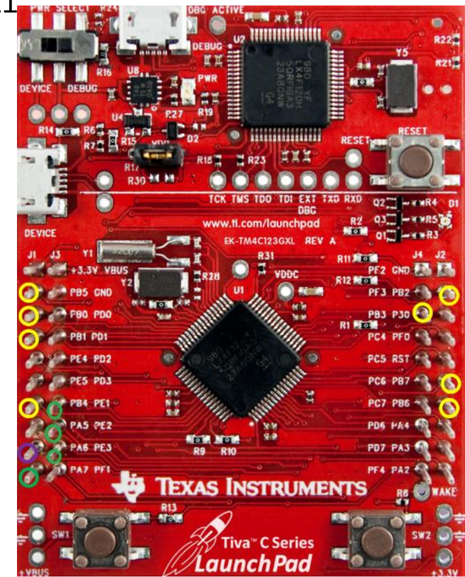
VSS	Ground	D0	PB0
VDD	5 v	D1	PB1
Vo	Potentiometer	D2	PB2
RS	PA6	D3	PB3
RW	Ground	D4	PB4
E	PA7	D5	PB5
A	5 v	D6	PB6
K	Ground	D7	PB7

Buzzer

Buzzer	PE2
Red Led	PE0

Heater

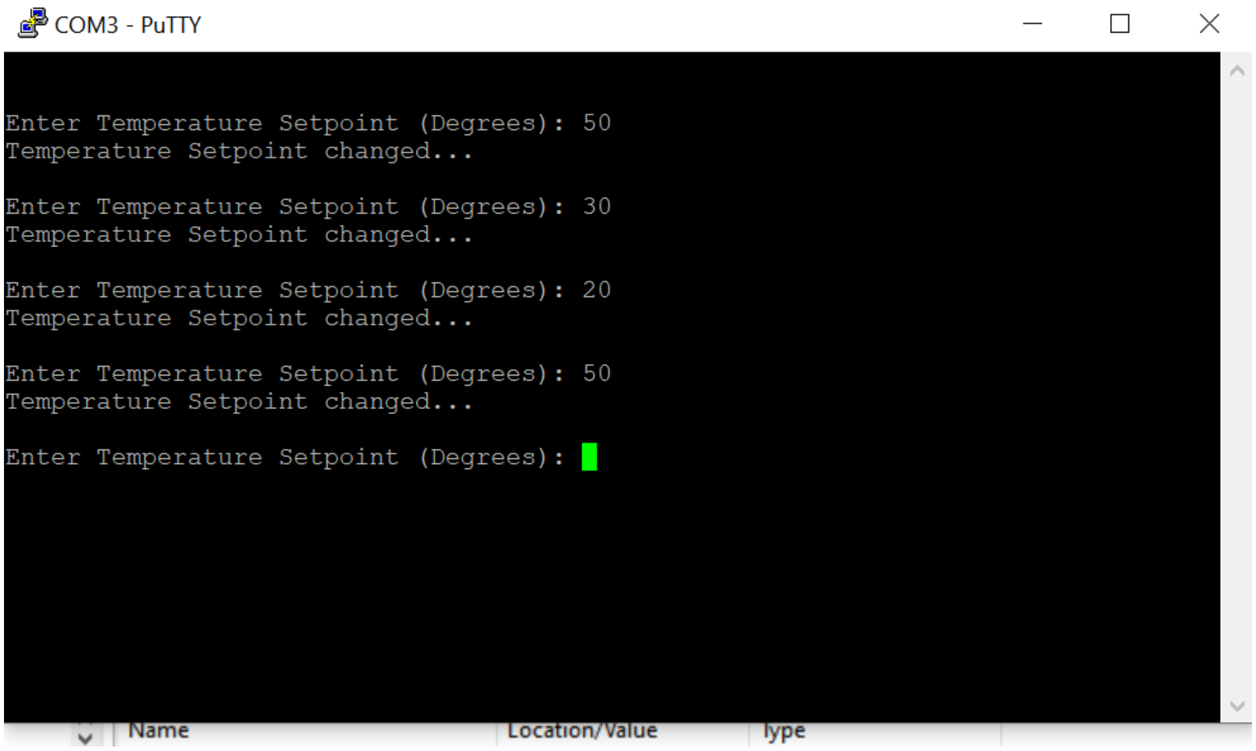
Green Led	PE1



Code Output

Terminal Output

This is the output the is shown on the terminal screen on the laptop to set the setpoint of the temperature in the program and displaying that the setpoint changed after pressing enter.



```
Enter Temperature Setpoint (Degrees): 50
Temperature Setpoint changed...

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

Enter Temperature Setpoint (Degrees): 20
Temperature Setpoint changed...

Enter Temperature Setpoint (Degrees): 50
Temperature Setpoint changed...

Enter Temperature Setpoint (Degrees):
```

Figure 2

LCD Output

This is the output on the LCD screen in which the measured temperature and the setpoint temperature are displayed on the screen.

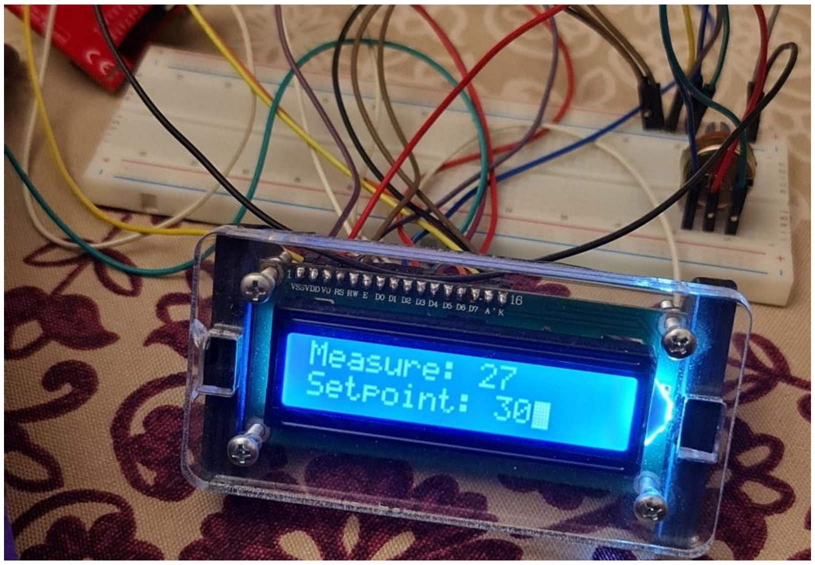


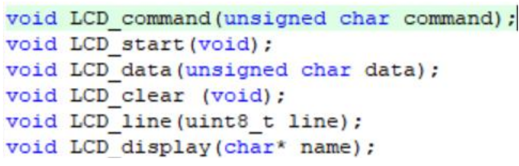
Figure 3

Libraries & Functions

LCD Library

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

- LCD_command & LCD_start & LCD_data: to initialize the LCD.
- LCD_clear: to clear the lcd.
- LCD_line: to move the cursor on the first or second line.
- LCD_display: to display string on the LCD.

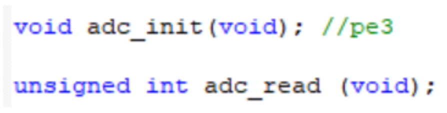


```
void LCD_command(unsigned char command);
void LCD_start(void);
void LCD_data(unsigned char data);
void LCD_clear (void);
void LCD_line(uint8_t line);
void LCD_display(char* name);
```

ADC Library

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 from the temperature sensor.

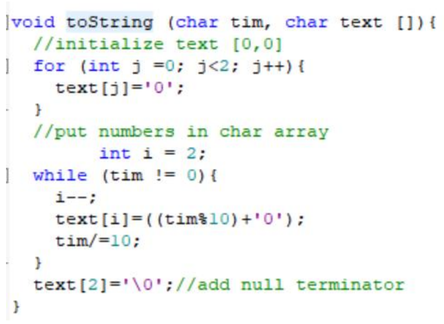


```
void adc_init(void); //pe3

unsigned int adc_read (void);
```

Convert to String function

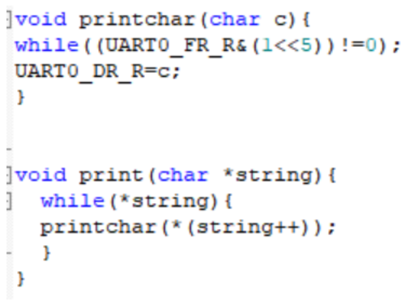
A function was made to receive variables and convert them to strings to be sent on the lcd.



```
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
}
```

Uart Functions

Those functions were made to send characters and strings to the terminal on putty on the laptop.



```
void printchar(char c){
while((UART0_FR_R&(1<<5))!=0);
UART0_DR_R=c;
}


void print(char *string){
  while(*string){
  printchar(*(string++));
  }
}
```

Problems & Solution

During implementing this project a few problems encountered me in the code and the hardware.

1. Finding Libraries for LCD & ADC.
2. Converting the code written in the textbook into keil.
3. Including Tivaware in keil.
4. Finding Temperature sensor and buzzer.
5. Connecting some of the hardware components on the tiva board and breadboard.

But with time I was able to solve these problems and successfully managed to run the code through the help from the TAs, my colleges and researching on the Internet.

1. With the help of my colleges, I found drivers for the LCD & ADC.
2. Through the internet I was able to find solutions on how to write the code properly.
3. From a YouTube video I managed to include the Tivaware library.
4. By asking around I found a shop that sells the components I needed.
5. By using YouTube and google I was able to connect all the hardware properly.

References

instruments, texas. (2020). *Tivware Peripheral Library*. texasinstruments.

Ibrahim, D. (2020). *Arm-Based Microcontroller Multitasking projects*. elsevier.

Market leading RTOS (Real Time Operating System) for embedded systems with Internet of Things extensions. FreeRTOS. (2021, May 12). https://www.freertos.org/index.html.