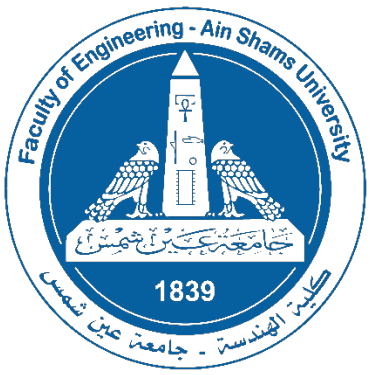
CSE 411 – Real-Time and Embedded System Design

Project Documentation



Computer Engineering and Software Systems
Ain Shams University – Faculty of Engineering
2021 – 2022

Team No. 26

Ahmad Salama Abdelaziz Salama
18P8805@eng.asu.edu.eg – Group 1 – Section 1 – UEL

Muhammad Ashraf Ali Bahgat
18P8083@eng.asu.edu.eg – Group 1 – Section 1 – UEL

Ahmed Bakry Abbas Masoud
1807211@eng.asu.edu.eg – Group 1 – Section 1

Ahmed Khaled Saad Ali
1809799@eng.asu.edu.eg – Group 1 – Section 1

Video Demo Link: *https://github.com/vadrif-draco/asufecse411tasks/issues/10*

Project GitHub Link: *https://github.com/vadrif-draco/asufecse411tasks/tree/project*

Project Thesis

As the project document says, we are required to design an ON-OFF temperature control sensor to control a heater with a multi-tasking queue-based FreeRTOS-based implementation. The system should turn off the heater when heating reaches the pre-defined setpoint and should kick in when the temperature is below it. Not much more is required for this system than this simple ON-OFF system, which is also called a BANG-BANG controller because it either BANGs ON or BANGs OFF, and this needs only the set-point value based on which it turns on or off.

The setpoint value for the temperature is entered using the PC keyboard onto the UART terminal. We have complete control over that value at any time (hence the multi-tasking approach), and we simulate temperature state changes (above setpoint, below setpoint, critical value) with LEDs as well as display everything on an LCD. The temperature measurement itself is simulated via a potentiometer. When the measured value is above the arbitrary danger level, the buzzer triggers.

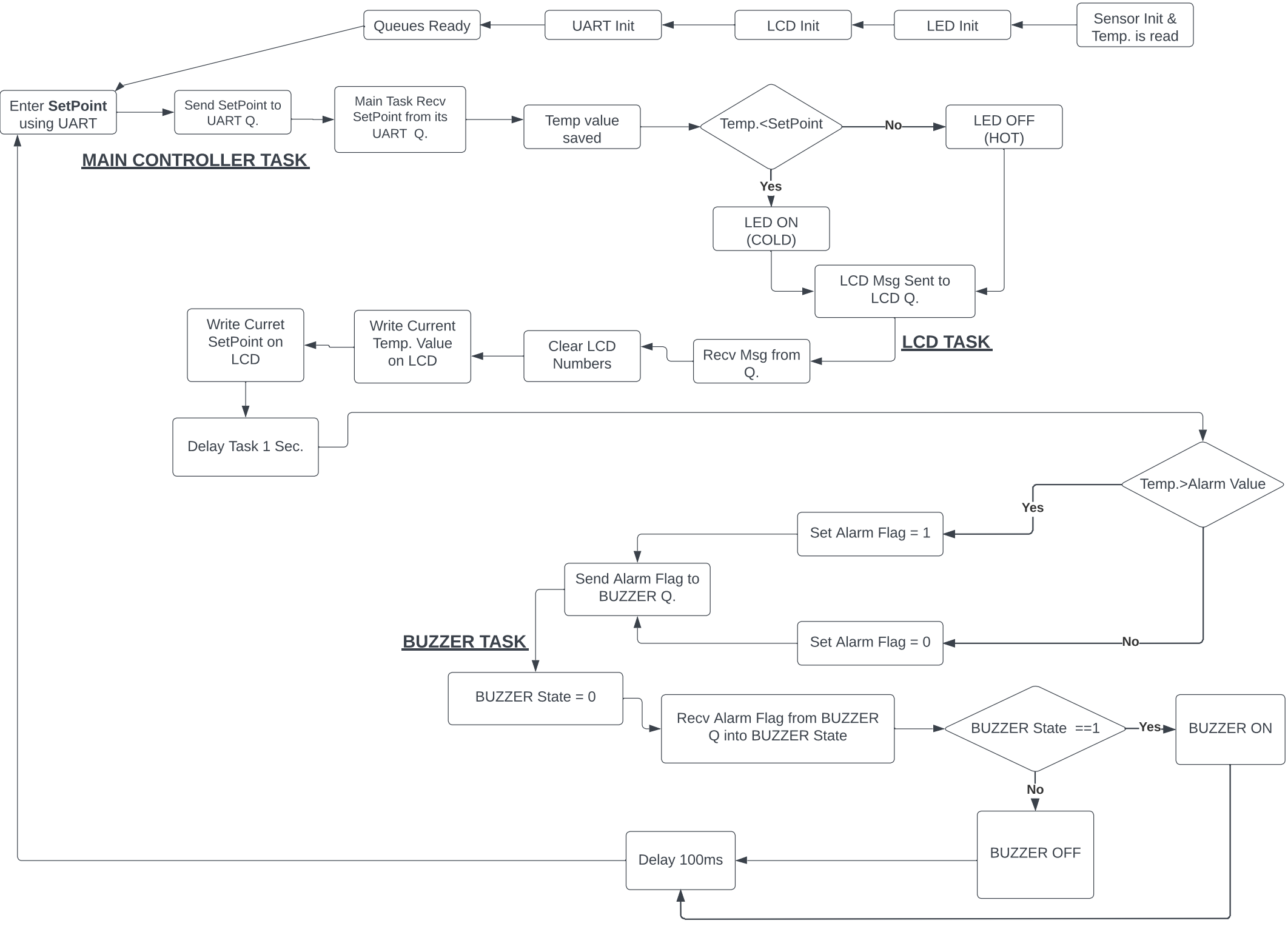
The aim of this project is to build this system using our knowledge with FreeRTOS and perhaps TivaWare as well on the TM4c123GH6PM microcontroller launchpad Tiva-C using the tasks and queues systems in FreeRTOS.

The components used are as follows:

- The launchpad and its LEDs
- Resistance(s)
- Transistor(s)
- Potentiometer to simulate the temperature sensor, interfaced with ADC
- LCD to display the state of the system
- Buzzer for alert
- PC Keyboard for input to UART

The technical details of the implementation can be found in the code itself (commented) as well as in the following pages of this document. Our implementation is mainly influenced by the book referred to in the project description.

The flow-chart describing the main logic of our implementation can be seen in the following page in full-size for high resolution reviewing.


Sensor Init & Temp. is read
LED Init
LCD Init
UART Init
Queues Ready
Enter SetPoint using UART
Send SetPoint to UART Q.
Main Task Recv SetPoint from its UART Q.
Temp value saved
Temp.<SetPoint
No
LED OFF (HOT)
Yes
LED ON (COLD)
MAIN CONTROLLER TASK
LCD Msg Sent to LCD Q.
LCD TASK
Recv Msg from Q.
Clear LCD Numbers
Write Current Temp. Value on LCD
Write Curret SetPoint on LCD
Delay Task 1 Sec.
Temp.>Alarm Value
Yes
Set Alarm Flag = 1
Send Alarm Flag to BUZZER Q.
BUZZER TASK
Set Alarm Flag = 0
No
BUZZER State = 0
Recv Alarm Flag from BUZZER Q into BUZZER State
BUZZER State ==1
Yes
BUZZER ON
No
BUZZER OFF
Delay 100ms

Code Documentation

The project code contains three principal parts/files, each split into a `.h` and `.c` pair, along with the `main.c` file. As is the norm, the `.h` files include all `#define` and function declarations that are later used in the `.c` files, where the implementation for all functions can be found. Each part serves a different purpose related to different tasks and their communication with the microcontroller. They are, with no specific order: LCD, Sensor, and UART.

LCD

Defines

Name	Value
EN	0x40
RS	0x20
clearDisplay	0x01
returnHome	0x02
displayOn	0x0A
cursorOn	0x0E
cursorBlink	0x0F
cursorOff	0xC
IncrementCursor	0x10|0x04
DecrementCursor	0x10|0x00
setTo4Bits	0x28
entryMode	0x06
firstLine	0x80
SecondLine	0xC0
LCD_SETDDRAMADDR	0x80
ShiftDisplyRight	0x10|0x08|0x04
ShiftDisplayLeft	0x10|0x08|0x00

Functions

Function Name	LCD_init
Return Type	void
Inputs	void
Description	Called before any LCD command to first initialize LCD.

Function Name	`LCD_Write4bits`
Return Type	`void`
Inputs	`char data, char control`
Description	Sent data is written to DRAM of LCD to either make a command or write data to screen.

Function Name	`LCD_cmd`
Return Type	`void`
Inputs	`char command`
Description	Takes a command and sends it to LCD.

Function Name	`LCD_data`
Return Type	`void`
Inputs	`char data`
Description	Takes input character and displays it on LCD and shifts cursor right.

Function Name	`LCD_WriteString`
Return Type	`void`
Inputs	`char* str`
Description	Takes input string and displays it on LCD.

Function Name	`gotoxy`
Return Type	`Void`
Inputs	`uint32_t x, uint32_t y`
Description	Moves cursor to specified location on LCD. First point on first line → (0, 0) First point on second line → (0, 1)

Function Name	DELAY
Return Type	void
Inputs	const int ms
Description	Pause the execution of the code for a set amount of time supplied (in ms).

Sensor

Functions

Function Name	initSensor
Return Type	void
Inputs	void
Description	Called before any sensor operations to first initialize sensor.

Function Name	readSensor
Return Type	uint32_t
Inputs	void
Description	Reads the current sensor value from the ADC and returns it.

UART

Functions

Function Name	UART0_INIT
Return Type	void
Inputs	uint32_t baudrate
Description	Called before any UART operations to first initialize UART0.

Function Name	SEND_CHAR_TO_UART0
Return Type	void
Inputs	char c
Description	Sends a character to UART0.

Function Name	READ_CHAR_FROM_UART0
Return Type	char
Inputs	void
Description	Reads a character from UART0.

Function Name	SEND_STR_TO_UART0
Return Type	void
Inputs	char *str
Description	Sends a string to UART0.

Function Name	clearUARTOutput
Return Type	void
Inputs	void
Description	Clears UART0 output.

main.c

The `main.c` file does not only contain the main function, which helps start all tasks, but also contains the task functions to be executed by FreeRTOS. It also contains a struct, `Message`, that contains the message to be displayed on the LCD. There are four tasks to be executed, `task1`, `LCDTask`, `UARTTask`, and `buzzerTask`.

Tasks

Task Name	task1
Priority	2
Description	Main task responsible for controlling the system. It provides the main functionality of turning off the LEDs and buzzer depending on the state of the system.

Task Name	UARTTask
Priority	2
Description	Task responsible for handling UART communication between the PC and the microcontroller.

Task Name	LCDTask
Priority	2
Description	Task responsible for updating the value of the temperature and setpoint displayed on the LCD.

Task Name	buzzerTask
Priority	2
Description	Task responsible for turning the buzzer on or off depending on the buzzer state value sent from task1.