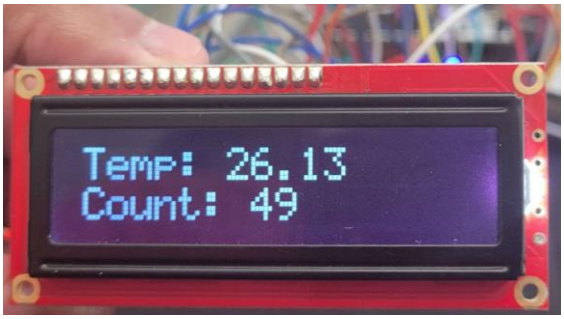
Michael Starks & Sovann Chak

Project 1 Module 3

ECEN 5803-002

1. What temperature is displayed on the LCD?



Appendix.

Module 3 Code:

```
/*----------------------------------------------------------------------------

LAB EXERCISE 12 - Real-Time Operating System

----------------------------------------

Integrate functions developed in previous modules and run then concurrently

in the mbed RTOS. The following four threads have to be implemented:


1. Display the temperature on the LCD and to the PC

2. Adjust the brightness of the RGB LED using a potentiometer

3. Display an incrementing counter on the LCD and to the PC

4. Blink an LED


 NOTE:  For this lab, the LCD, temp sensor and POT are virtual devices.
```

GOOD LUCK!

@file main.cpp

@brief Integrates functions developed in previous modules and run then concurrently

@author Michael Starks and Sovann Chak

@data October 6, 2023

@version 1.0

--/

#include "mbed.h"

#include "rtos.h"

#include "DS1631.h"

#include "NHD_0216HZ.h"

#include "pindef.h"

/*

Define the mutex

Define the LCD display and the temperature sensor

Define other inputs and outputs

*/

/* Write your code here */

/* Define the screen, temp sensor and mutex objects */

DS1631 temp_sensor(I2C_SDA,I2C_SCL,0x90);

NHD_0216HZ display(SPI_CS,SPI_MOSI,SPI_SCLK);

Mutex display_mutex;

```
/* Define the IO for led adjustment input and output and blinking led */
DigitalOut blink_led(LED2);
AnalogIn pot_in(A0);
PwmOut led(D3);



/*Display temperature on the LCD */
void temp_thread(void const *args){
float temp;
/*write your code here */
while(1) {
/* Read the temperature */
temp = temp_sensor.read();
/* Aquire the mutex to write to the screen */
display_mutex.lock();
/* Set the cursor to the correct row and column */
display.set_cursor(0,0);
/* Write to the display */
display.printf("Temp: %0.2f",temp);
/* Unlock the mutex */
display_mutex.unlock();
/* yield for another thread */
//osThreadYield();
osDelay(100);
}
}

/*Adjust the brightness of the RGB LED */
```

```
void adjust_brightness(void const *args){
led.period(.01);
/*write your code here */
while(1) {
/* Read and scale the v_in from the pot and set the duty cycle */
led = (pot_in.read()-0.5)/0.5;
/* yield for another thread */
//osThreadYield();
osDelay(10);
}
}

/*Blink an LED */
void led1_thread(void const *args){
/*write your code here */
while(1){
/* Toggle the LED */
blink_led = !blink_led;
/* sleep for a half second */
osDelay(500);
}
}

/*Display a counter on the LCD */
void count_thread(void const *args){
/*write your code here */
uint8_t count = 0;
while(1) {
/* Aquire the mutex to write to the screen */
```

```
display_mutex.lock();
/* Set the cursor to the correct row and column */
display.set_cursor(0,1);
/* Write to the display */
display.printf("Count: %d", count);
/* Unlock the mutex */
display_mutex.unlock();
/* Increment the counter */
count++;
/*sleep for a half second */
osDelay(250);
}

}

/*-------------------------------------------------------------------------
MAIN function
*-------------------------------------------------------------------------*/

int main(){
/*
Initialise and clear the LCD display
Start all threads
Wait for timer interrupt
*/


/*write your code here */
/* Initialize and clear the display */
```

```
display.init_lcd();
display.clr_lcd();

/* Set up the thread definitions */
osThreadDef(adjust_brightness,osPriorityNormal,256);
osThreadDef(led1_thread,osPriorityNormal,256);
osThreadDef(count_thread,osPriorityNormal,640);
osThreadDef(temp_thread,osPriorityNormal,512);

/* Create the threads */
osThreadCreate(osThread(adjust_brightness),NULL);
osThreadCreate(osThread(led1_thread),NULL);
osThreadCreate(osThread(count_thread),NULL);
osThreadCreate(osThread(temp_thread),NULL);

while(1){
/* Sleep the main thread */
__wfi();
}



}

/* ******************************ARM University Program Copyright (c) ARM Ltd
2014************************************ */
```