ECEN 5803

Mastering Embedded System Architecture

(Fall-2023)

PROJECT-1 : MODULE 3

RTOS THREADS

Date

10/7/2023

Guided By

Prof. Timothy Scherr

Submitted By

Ajay Kandagal

Kshitija Dhondage

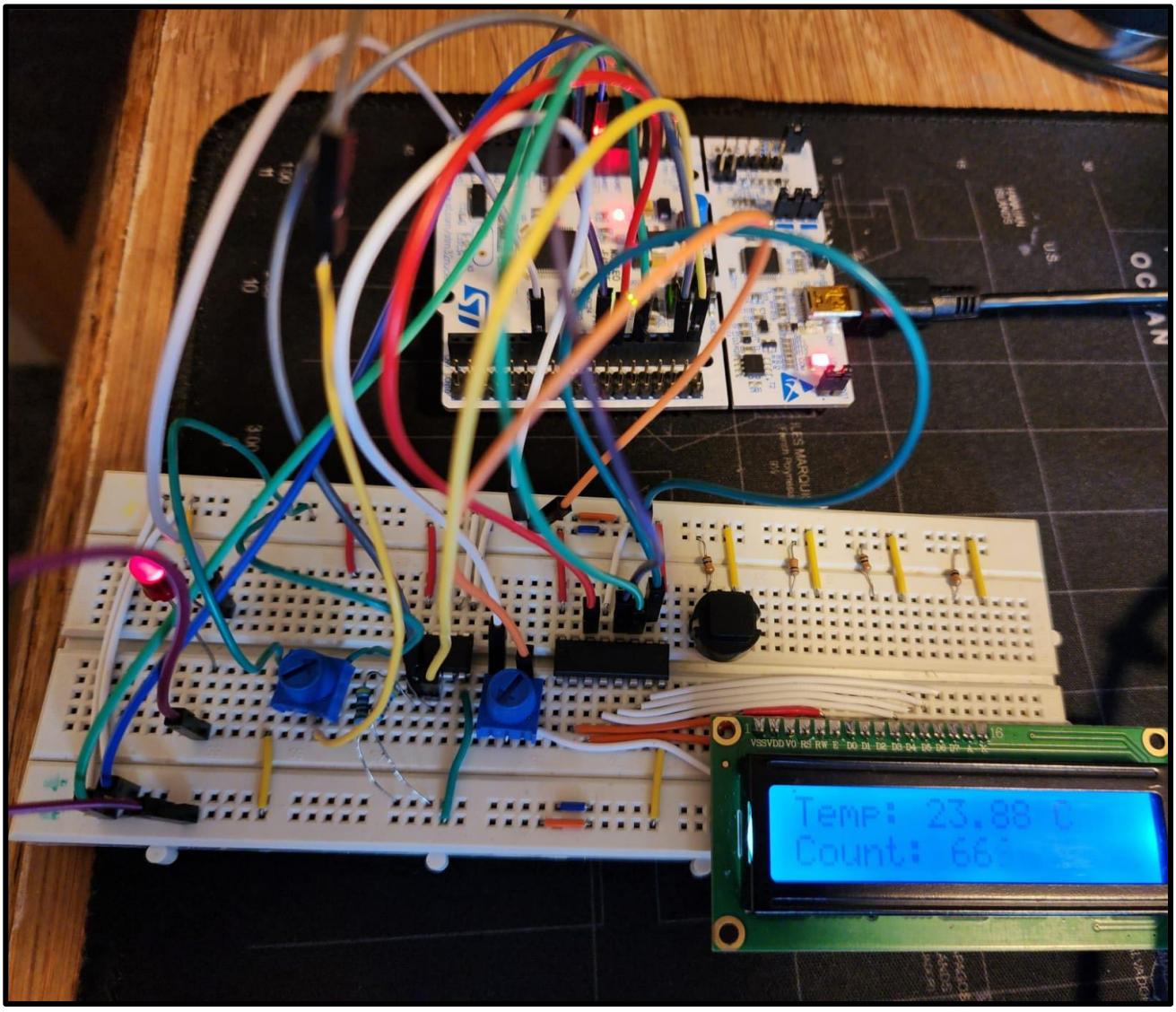
Que. What temperature is displayed on the LCD?

Ans: The temperature sensor sends temperature data via I2C, which is displayed on the PC terminal through UART and simultaneously on an LCD screen using SPI.

So, the same temperature value is displayed on the LCD and PC terminal.

Code Output:

Hardware Setup and LCD output:



PC terminal Output:

Appendix

```
/**
 * @file    main.cpp
 * @brief   Code for Project 1 Module 3: RTOS THREADS
 * @version v1.00
 * @date    7, October, 2023
 * @authors Kshitija Dhondage (ksdh4214@colorado.edu)
 *              Ajay Kandagal (ajka9053@colorado.edu)
 * @details The code uses CMSIS RTOS V1.
 *          In this code the four threads are implementedand and they run concurrently.
 *          The following four threads are implemented:
 *              1. Thread to display the temperature on the LCD and to the PC
```

```
 *              2. Thread to adjust the brightness of the externel RGB LED using a potentiometer
 *              3. Thread to display an incrementing counter on the LCD and to the PC
 *              4. Thread Blink on board LED
 * @Reference https://www.keil.com/pack/doc/CMSIS/RTOS/html/usingOS.html
 */

/** Header files */
#include "mbed.h"
#include "rtos.h"
#include "DS1631.h"
#include "NHD_0216HZ.h"
#include "pindef.h"
#include "cmsis_os.h"

/** Serial UART interface */
Serial pc_terminal(SERIAL_TX, SERIAL_RX);

/** Set the temp sensor DS1631 , I2c_SDA, I2c_SCL, and address */
DS1631 sensor(PB_9, PB_8, 0x90);

/** Set LCD parameters: CS, MOSI, SCL for SPI interface */
NHD_0216HZ LCD_display(PB_6, PA_7, PA_5);

/** on-board LED is sending out the digital signal */
DigitalOut Internel_LED1(LED1);

/** MCU takes the analog data from the potentiometer */
AnalogIn Pot(PA_0);
```

```
/** MCU sends the PWM output to the external LED */
PwmOut Externel_LED(PB_10);

/** Mutex ID */
osMutexId LCD_mutex;
/** Declar Mutex */
osMutexDef(LCD_mutex);

/**
 * @brief     temp_thread fucntion to display the temp sensed from DS1631 via I2c interface
              to LCD display via SPI interface and PC terminal via UART interface
 * @param[in] optional parameter
 * @returns   None
 */
void temp_thread(void const *args)
{
        float temperature = 0.0;
        while(1)
        {
                temperature = sensor.read();

                osMutexWait(LCD_mutex, osWaitForever);
                pc_terminal.printf("\n\r Temperature: %0.2f C\n\r",temperature);
                LCD_display.set_cursor(0,0);
                LCD_display.printf("Temp: %0.2f C",temperature);
                osMutexRelease(LCD_mutex);
```

```
                wait_ms(40);
            }
}

/**
 * @brief     Adjust_brightness thread fucntion to control externel LED brightness
              via PWM by reading the Potentiometer value via analog interface
 * @param[in] optional parameter
 * @returns   None
 */
void adjust_brightness(void const *args)
{
  while (1)
  {
    /** Read the POT value */
    float potValue = Pot.read();

    /** Set the LED period */
    Externel_LED.period_us(4.0f);

    /** Adjust LED brightness based on POT value */
    for (float brightness = 0.0f; brightness < potValue; brightness += 0.01f)
    {
      Externel_LED.write(brightness * potValue);
    }

    /** Wait for a period */
    wait(1.0);
```

```
  }
}



/**
 * @brief     led1_thread fucntion to blink the on board LED1
 * @param[in] optional parameter
 * @returns   None
 */
void led1_thread(void const *args)
{
          while(1)
          {
            Internel_LED1 = 1;
    wait(2.0);
    Internel_LED1 = 0;
                    wait(1.0);
  }
}

/**
 * @brief     cont_thread fucntion to display the increment counter value
            to LCD display via SPI interface and PC terminal via UART interface
 * @param[in] optional parameter
 * @returns   None
 */
void count_thread(void const *args)
{
```

```
 static int count=0;
        while(1)
        {
                osMutexWait(LCD_mutex, osWaitForever);
                LCD_display.set_cursor(0,1);
                LCD_display.printf("Count: %d", count);
                pc_terminal.printf("\n\r Count = %d \n\r", count);
                osMutexRelease(LCD_mutex);

                wait_ms(500);
                count++;
        }
}

/** Defining a four threads named
*   1. led1_thread
*   2. adjust_brightness
*   3. temp_thread
*   4. count_thread
*   with normal priority and a stack size of 1024 bytes
*/
osThreadDef(led1_thread,       osPriorityNormal, 1024);
osThreadDef(adjust_brightness, osPriorityNormal, 1024);
osThreadDef(temp_thread,       osPriorityNormal, 1024);
osThreadDef(count_thread,      osPriorityNormal, 1024);

/**
 * @brief Main function Entry point function
```

```
 *
 * @param[in] Void
 *
 * @return    int
 */
int main()
{
        /** LCD initilization */
        LCD_display.init_lcd();

        osMutexId mutex_id;
        /** Create a mutex to protect the single access to LCD */
 LCD_mutex = osMutexCreate(osMutex(LCD_mutex));

        osThreadId id1, id2, id3,id4;

        /** Create the temp_thread thread */
        id1 = osThreadCreate(osThread(temp_thread), NULL);
        if(id1 == NULL)
        {
   pc_terminal.printf("\n\r Failed to create Temp thread\n\r");
  }

        /** Create the count_thread */
 id2 = osThreadCreate(osThread(count_thread), NULL);
        if (id2 == NULL)
        {
    pc_terminal.printf("\n\r Failed to create Count thread\n\r");
```

```
  }

        /** Create the led1_thread */
        id3 = osThreadCreate(osThread(led1_thread), NULL);
        if (id3 == NULL)
        {
    pc_terminal.printf("\n\r Failed to create Led1 thread\n\r");
  }

        /** Create the adjust_brightness thread */
        id4 = osThreadCreate(osThread(adjust_brightness), NULL);
        if (id4 == NULL)
        {
    pc_terminal.printf("\n\r Failed to create adjust brightness thread\n\r");
  }

        /** sleep mode to reduce the power consumption */
        while(1)
        {
                __wfi();
        }
}
```