***Rapid Embedded Systems***

***Design and Programming Course***

**LAB 6**

**Real-Time Operating Systems**

**Issue 1.0**

Contents

[1 Introduction 1](#_Toc35965049)

[1.1 Lab overview 1](#_Toc35965050)

[2 Requirements 1](#_Toc35965051)

[2.1 Software and Hardware 1](#_Toc35965052)

[3 Hardware Setup 2](#_Toc35965053)

[3.1 Pin Layout 2](#_Toc35965054)

[4 Real-Time Operating Systems 3](#_Toc35965055)

[4.1 RTOS APIs 3](#_Toc35965056)

[4.2 Your Application Code 4](#_Toc35965057)

[5 Additional references 4](#_Toc35965058)

[6 Troubleshooting 4](#_Toc35965059)

# Introduction

## Lab overview

In this lab, you will integrate the functions developed in the previous modules and run them concurrently in the Mbed RTOS. Each of the previous tasks will be executed in one thread that is scheduled by the RTOS. There are four threads:

1. Displaying the temperature on the LCD

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

3. Displaying an incrementing counter on the LCD

4. Blinking an LED

# Requirements

## Software and Hardware

In this lab, we will be using the following hardware and software:

* **Mbed Studio**, an IDE designed to streamline development and prototyping using Mbed enabled microcontrollers and development boards

**OR**

**Mbed online compiler**, which can be found here: <https://ide.mbed.com/compiler>

* **NUCLEO-F401RE**, or another suitable Mbed OS 5 compatible development board. A full list of compatible devices can be found here: <https://os.mbed.com/platforms>, note that outputs may have to be reconfigured for devices following a different standard.
* A breadboard, 2 x 100Ω resistor, 2 x 1kΩ resistor, 1 x 10kΩ potentiometer, 1 x temperature sensor (DS1631), 1 x LCD (NHD\_0216HZ), 1 x shift register (74HC595N), 3 x RGB LED and 1 x LED.

The code skeleton, which includes some assistance for implementing the task in 4.2, should be found in the same folder as this manual

# Hardware Setup

## Pin Layout

In this experiment, we are going to use the Nucleo F401RE Board, for which pin descriptions can be found below:



Figure 1: The NUCLEO F401RE board pin descriptions

On 2 breadboards, build the circuits in Figure 2, using the hardware components listed in 2 .

 

Figure 2: Circuit Layouts

Connect the LEDs and the pins on the circuits to their relative pin which are defined in the table below.

|  |  |
| --- | --- |
| **Pin** | **Pin name in Mbed API** |
| Potentiometer 1  | A0 |
| LED | D4 |
| RED LED | D3 |
| GREEN LED | D5 |
| BLUE LED | D6 |
| LCD SPI COTI | D11 |
| LCD SPI SCK (SCLK) | D13 |
| LCD SPI CS | D10 |
| Temperature sensor I2C SCL | D15 |
| Temperature sensor I2C SDA | D14 |
| USB UART TX | D1 |
| USB UART RX | D0 |

# Real-Time Operating Systems

In this lab, we will integrate four simple functions and run them concurrently using the Mbed RTOS.

## RTOS APIs

The RTOS APIs handle creation and destruction of threads in Arm Mbed OS 5, as well as mechanisms for safe interthread communication. In this lab, you will be introduced to two elements of the Mbed RTOS APs:

* **Thread**
* **Mutex**

The **Thread** class allows defining, creating and controlling parallel tasks.

A **Mutex** is used to synchronize the execution of threads, for example to protect the access to a shared resource. Note that the Mutex methods cannot be called from interrupt service routines. It can break the thread safety mechanism and introduce race conditions into an otherwise safe piece of code.

Additional information about the RTOS API with example can be found in the Additional references.

## Your Application Code

* Create four threads to:
	1. Display the temperature on the LCD
	2. Adjust the brightness of the RGB LED using a potentiometer
	3. Display an incrementing counter on the LCD
	4. Blink an LED
* Note that:
	+ Every thread contains an infinite loop
	+ Tasks are scheduled in a cooperative manner, namely, threads can go to waiting state after each update, to let the next thread be scheduled
	+ Since the LCD must be accessed exclusively, we need a mutex to protect its single access.
* In the main program:
	+ Clear the LCD display
	+ Start all threads
	+ Use sleep mode to reduce power consumption

If you experience any issues with these tasks you can always look at the solution code or refer to the troubleshooting section of the document.

# Additional references

**RTOS APIs in Mbed Studio**

<https://os.mbed.com/docs/mbed-os/v5.13/apis/rtos.html>

# Troubleshooting

If you experience a problem when trying to print a word to the lcd using SPI, it could be because your init\_lcd method is not correct, if you were to use the solution provided and you still can not print any word in the lcd do the following:

Make a while loop in main.cpp and inside add the init\_lcd() method along with the print\_lcd() , this will ensure that the lcd initializes at least one time as well as the execution of the printing function.

