***Rapid Embedded Systems***

***Design and Programming Course***

**LAB 2**

**Interrupts**

**Issue 1.0**

Contents

[1 Introduction 1](#_Toc35440393)

[1.1 Lab overview 1](#_Toc35440394)

[2 Requirements 1](#_Toc35440395)

[2.1 Software and Hardware 1](#_Toc35440396)

[3 Hardware Setup 2](#_Toc35440397)

[3.1 Pin Layout 2](#_Toc35440398)

[4 Software Interfaces 3](#_Toc35440399)

[4.1 InterruptIn/InterruptOut Interfaces 3](#_Toc35440400)

[4.2 Low Power Consumption 3](#_Toc35440401)

[5 Your Application Code 4](#_Toc35440402)

[5.1 Toggle the Lights 4](#_Toc35440403)

[5.2 Counter 4](#_Toc35440404)

[6 Additional references 4](#_Toc35440405)

# Introduction

## Lab overview

In this lab, we will implement two functions. For the first function, the RGB LEDs are controlled by the status of the buttons. For the second, the number of times each button has been pressed controls the LEDs. By the end of this lab you will get some insight and practical experience with the Mbed API for Interrupts.

# Requirements

## Software and Hardware

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

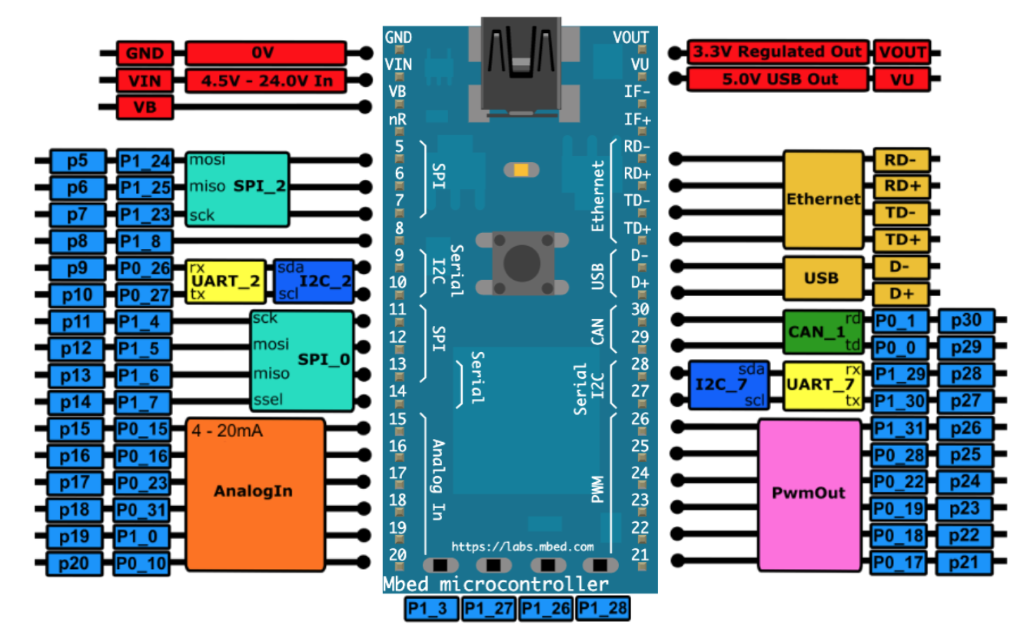
* **Visual Studio code,** or any other text editor.
* **Mbed Simulator**, an offline simulator of an mbed microcontroller and hardware components.

Two code skeletons, which include some support for implementing the tasks in 5.1 and 5.2, should be found in the same folder as this manual.

# Hardware Setup

## Pin Layout

In this experiment, we are going to use the Mbed Simulator target. Pin descriptions can be found below:



*Figure 1: The Mbed Simulator board pin descriptions*

In the Simconfig.json file, connect the push buttons, implemented in the “Getting Started” guide, and the LEDs to their respective pin as defined in the table below.

|  |  |
| --- | --- |
| Pin | Mbed Pin Name |
| Button 1 | p5 |
| Button 2 | p6 |
| Button 3 | p7 |
| Button 4 | p8 |
| Red LED | p9 |
| Blue LED | p10 |
| Yellow LED | p11 |

# Software Interfaces

## InterruptIn/InterruptOut Interfaces

The InterruptIn interface is used to trigger an event when a digital input pin changes. For example:

InterruptIn button\_press(Input Pin);

void button\_ISR(){

Led\_out = !Led\_out;

}

int main(){

button\_press.rise(&button\_ISR);

while(1); // waiting for interrupts

}

Other functions are listed below:

|  |  |
| --- | --- |
| Function name | Description |
| void rise (void(\*fptr)(void)) | Attach a function to call when a rising edge occurs on the input |
| template<typename T>  void rise (T \*tptr, void (T : :\*mptr)(void)) | Attach a member function to call when a rising edge occurs on the input |
| void fall (void(\*fptr)(void)) | Attach a function to call when a falling edge occurs on the input |
| template<typename T >  void fall (T \*tptr, void(T::\*mptr)(void)) | Attach a member function to call when a falling edge occurs on the input |
| void mode (PinMode pull) | Set the input pin mode |
| void enable\_irq () | Enable IRQ |
| void disable\_irq () | Disable IRQ |

## Low Power Consumption

Use the sleep mode to reduce the power consumption of your application. For example, to use the Sleep-on-Exit feature you can use the wait-for-interrupt operation:

\_\_WFI() ; //go to sleep.

However, wait-for-interrupt operation is not yet implemented in the Mbed Simulator.

We can use: wait\_ms(); instead.

# Your Application Code

In this lab task, you will use the Mbed API for Interrupts in order to complete two exercises.

## Toggle the Lights

In the first exercise, you need to use the Mbed API functions to:

* Define an InterruptIn and ISR for each bit from input
* Setup the interrupt service routine, so that the RGB LED is used to indicate when a button is pressed.
  + Button 1: toggles RED
  + Button 2: toggles GREEN
  + Button 3: toggles BLUE
  + Button 4: lights up WHITE (RED, GREEN and BLUE)
* Put the processor into sleep mode upon exiting from the ISR

Try to issue an interrupt on different signal edges (rising edge or falling edge). What changes?

## Counter

In the second exercise, you will need to complete the skeleton code so that it counts the number of times a button has been pressed and lights up the RGB LED, which correspond to the button which has been pressed the most.

* Button 1 corresponds to the Red LED
* Button 2 corresponds to the Green LED
* Button 3 corresponds to the Blue LED
* Button 4 is used to reset all the counters and turn off all the LEDs.

Also, if several buttons have been pressed the same number of times, their respective LEDs should all be ON simultaneously.

**If you are stuck you can look at the code solution provided with the lab for some help.**

# Additional references

**Documentation for Mbed API for Interrupts**

<https://os.mbed.com/docs/mbed-os/v5.13/apis/interruptin.html#related-content>