***Rapid Embedded Systems***

***Design and Programming Course***

**LAB 1**

**Digital I/O**

**Issue 1.0**

Contents

[1 Introduction 1](#_Toc25141183)

[1.1 Lab overview 1](#_Toc25141184)

[2 Requirements 1](#_Toc25141185)

[2.1 Software and Hardware 1](#_Toc25141186)

[3 Hardware Setup 2](#_Toc25141187)

[3.1 Pin Layout 2](#_Toc25141188)

[4 DigitalIn/DigitalOut Interfaces 3](#_Toc25141189)

[4.1 Introduction to DigitalIn/DigitalOut Interfaces 3](#_Toc25141190)

[4.2 Application code 3](#_Toc25141191)

[5 BusIn/BusOut Interfaces 4](#_Toc25141192)

[5.1 Introduction to BusIn/BusOut Interfaces 4](#_Toc25141193)

[5.2 Application code 5](#_Toc25141194)

[6 Appendix 5](#_Toc25141195)

[6.1 Create a Switch 5](#_Toc25141196)

[7 Additional references 6](#_Toc25141197)

# Introduction

## Lab overview

In this lab, we will implement two functions in which the RGB LEDs are controlled by the status of the buttons (switches). We will use the DigitalIn/DigitalOut and the BusIn/BusOut interfaces to assist with this.

By the end of this lab you will get some insight and practical experience with the Mbed API for Digital I/O.

# Requirements

## Software and Hardware

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

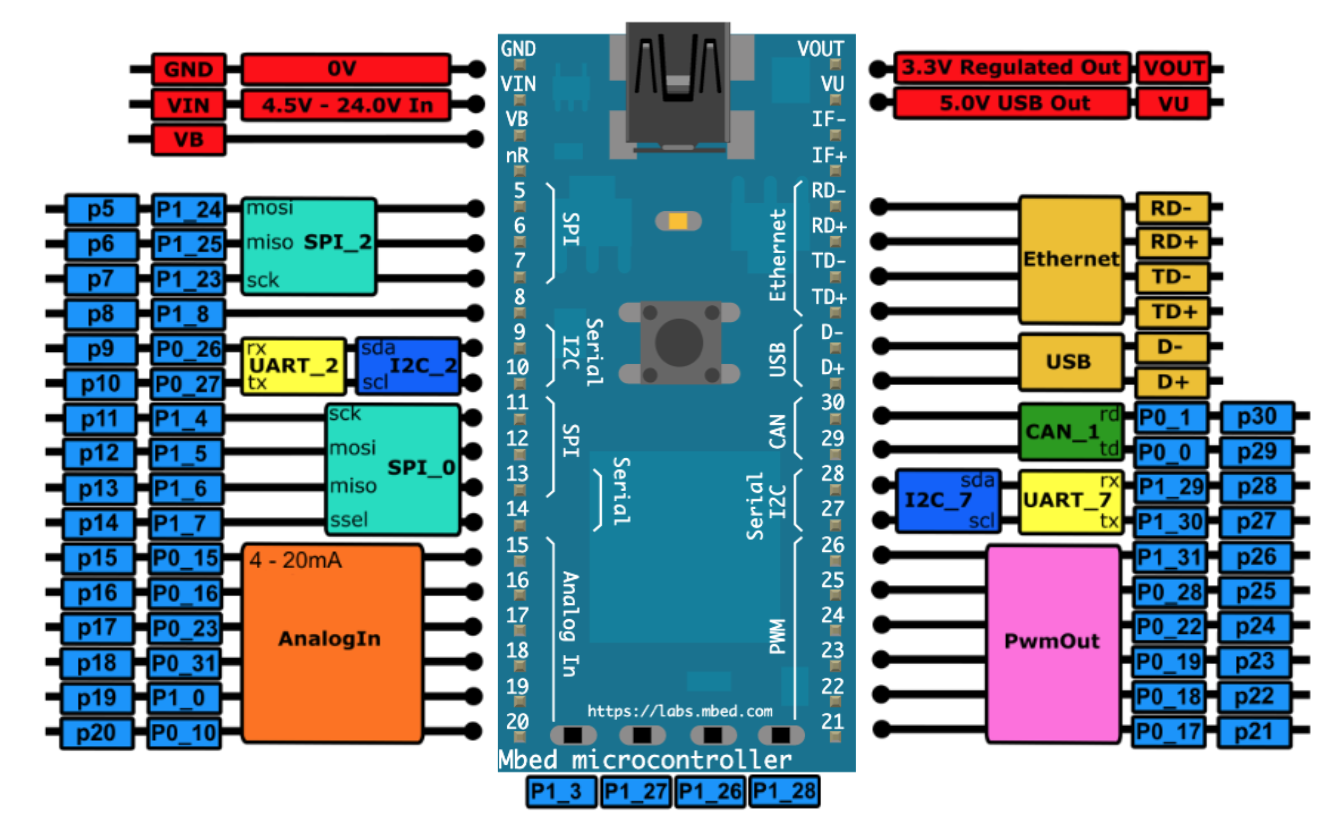
* **Visual Studio Code,** or any other text editor.
* **Mbed Simulator**, an offline simulator of an mbed microcontroller and hardware components.

The code skeleton, which includes some support for implementing the task in at the end of the lab 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. The pin descriptions for the simulated board can be found below:



*Figure 1: The Mbed Simulator board pin descriptions*

Create and add the switch (button) component of the simulator as explained in part “6. Appendix”

In the Simconfig.json file, connect the switches 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 |

# DigitalIn/DigitalOut Interfaces

In this task, you will be implementing a function in which you can control RGB LEDs using the status of buttons. This will be achieved using the DigitalIn/DigitalOut Interfaces.

## Introduction to DigitalIn/DigitalOut Interfaces

The DigitalIn interface is used to reads the value of a digital input pin. The logic level is either 1 or 0.

The DigitalOut interface is used to configure and control a digital output pin by setting the pin to a logic level 0 or 1.

You can use any of the numbered Arm Mbed pins as a DigitalIn or DigitalOut, for example:

DigitalIn mybutton (Input Pin);

DigitalOut Led\_out(Output Pin);

int main(){

if (mybutton)

Led\_out = 1;

}

The DigitalIn/Out interfaces have a set of member functions which can be found in figure 2.

|  |  |
| --- | --- |
| Function name | Description |
| int read () | Read the input, represented as 0 or 1 (int) for DigitalIn  Return the output setting, represented as 0 or 1 (int) for DigitalOut |
| Operator int() | Shorthand for read () |
| void mode (PinMode pull) | Set the input pin mode. Parameters: PullUp, PullDown, PullNone, OpenDrain. |
| int is\_connected () | Return the output setting, represented as 0 or 1 (int) |
| void write (int) | Set the output, specified as 0 or 1 (int) |
| DigitalOut& operator = (int value) | Shorthand for write () |

Figure 2: Member function of DigitalIn and DigitalOut API

The DigitalInOut interface is a bidirectional digital pin, we can use this interface to read the value of a digital pin when set as an input(), as well as write the value when set as an output().

## Application code

In this exercise you are required to write the function *ControlLED\_DigitalIO* in the code skeleton provided, using the DigitalIn, DigitalOut Interfaces, and boolean algebra.

The buttons represent an unsigned binary number N with Button 1 being the Least Significant Bit (LSB) and button 4 the Most Significant Bit (MSB). Your function should turn on the right combination of RGB LEDs for all ranges of N as specified below:

*Note: The following table assumes that you have used current sourcing when connecting the LEDs.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N | 0-3 | 4-5 | 6-7 | 8-11 | 12-14 | 15 |
| Red LED | ON | OFF | ON | OFF | OFF | ON |
| Yellow LED | OFF | ON | ON | ON | OFF | ON |
| Blue LED | OFF | OFF | OFF | ON | ON | ON |

1. Define the Input and Output pins in the code skeleton.
2. After checking that the Boolean equation corresponding to each RGB LED is:

RED LED= (!b4&&(!b3||b2)) || (b4&&b3&&b2&&b1)

Yellow LED= (b4^b3) || (b4&&b3&&b2&&b1)

BLUE LED= b4

Write and test the function *ControlLED\_DigitalIO*

# BusIn/BusOut Interfaces

In this task, you will be implementing the same function as in section 4, but using the BusIn/BusOut interfaces instead of the DigitalIn/DigitalOut interfaces.

## Introduction to BusIn/BusOut Interfaces

The BusIn interface is used to combine DigitalIn pins to read them at once, for example:

BusIn My\_Bus\_In(Input Pin 1, Input Pin 2, Input Pin 3);

int main(){

if(My\_Bus\_In == 0b001)

Led\_out = 1;

}

Similarly, the BusOut interface is used to combine DigitalOut pins to write them at once, for example:

BusOut My\_Bus\_Out(Output Pin 1, Output Pin 2, Output Pin 3);

int main(){

My\_Bus\_Out = 0b010;

}

Note that the order of the pins in the constructor is the reverse order of the pins in the byte order. So, if you have BusOut(a,b,c,d,e,f,g,h), then the order of bits in the byte would be “hgfedcba” with “a” being bit 0, “b” being bit 1, “c” being bit 2 and so on.

## Application code

In this exercise, you are required to write the function *ControlLED\_BusIO* using the BusIn, BusOut interfaces and a switch statement. The function should satisfy the same requirements as specified in 4.2.

1. Define the input and output buses in the code skeleton.
2. Using the guidelines shown in figure 3, write and test the function, *ControlLED\_BusIO.*

void ControlLED\_BusIO (){

switch (buttonsbus){

case 0 ... 3:

ledsbus=0b0100;

break;

// write code her//

}

}

Figure 3: ControlLed\_BusIO code skeleton

1. Compare the interfaces DigitalIn/DigitalOut and BusIn/BusOut. In which cases will it be better to use the BusIn/BusOut interfaces over the DigitalIn/DigitalOut interfaces?

If you have any issues – refer to the code solution files provided for some help.

# Appendix

## Create a Switch

We are going to create a new component and add it to the Mbed Simulator as it was explained in the Getting Started Lab.

1. Find two pictures of the same dimensions, one picture of switch which is ON and another one of the same switch when it is OFF.
2. Save these two pictures to *viewer/img* and rename them to *switch\_on.png* and *switch\_off.png*.
3. Add in viewer/js-ui/viewer.js:

{ component: 'Switch', name: 'Switch',  pins: [ 'Switch' ] },

inside var components = [ … ]

1. Add the *switch.js* provided file to *viewer/js-ui/components*.
2. Add in viewer/viewer.html:

<script type="text/javascript" src="/js-ui/components/switch.js">.

</script>

under   <!-- UI handling -->  and below the line:

<script type="text/javascript" src="/js-ui/components/base\_component.js"></script>

# Additional references

**Documentation Mbed API for Digital I/O**

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