Carleton UNIVERSITY

Department of Electronics

ELEC 4601: Laboratory 4

Programming with the mbed SDK

Introduction

In this lab, you will learn how to use the mbed software development kit (SDK). In Part A, you will write a program that replicates Lab 3 (controlling an RGB LED with a 5-way joystick), but with the help of the newly added mbed SDK. In Part B, you will write an audio player program that uses the speaker and the two potentiometers on the mbed application shield.

Hardware & Software

There are four tools used in this lab:

1. STM32 Nucleo F401RE development board (Arm Cortex-M4)
2. Arm mbed application shield
3. Keil µVision IDE
4. **mbed SDK**

mbed SDK

The mbed SDK provides startup code, C runtime, **libraries**, and peripheral APIs to expedite the development process. The many lines of code needed to setup GPIO ports and interrupt handlers, like in Lab 3, can been replaced with simple method calls from the included libraries—allowing us to create more complex and robust programs in less time.

STM32 Nucleo F401RE development board

The development board used in this lab is part of the STM32 family of microcontrollers by STMicroelectronics. It is useful for building prototypes and learning new concepts related to 32-bit Arm-based microprocessors. The F401RE features an Arm Cortex-M4 32-bit RISC CPU (+ 96 kB of SRAM, 512 kB of flash memory, a debugging interface, and various peripherals). The pinout (header) of this board, shown in Fig. 1, allows the programmer to access each pin of the microcontroller individually, or through an additional development shield (mbed).

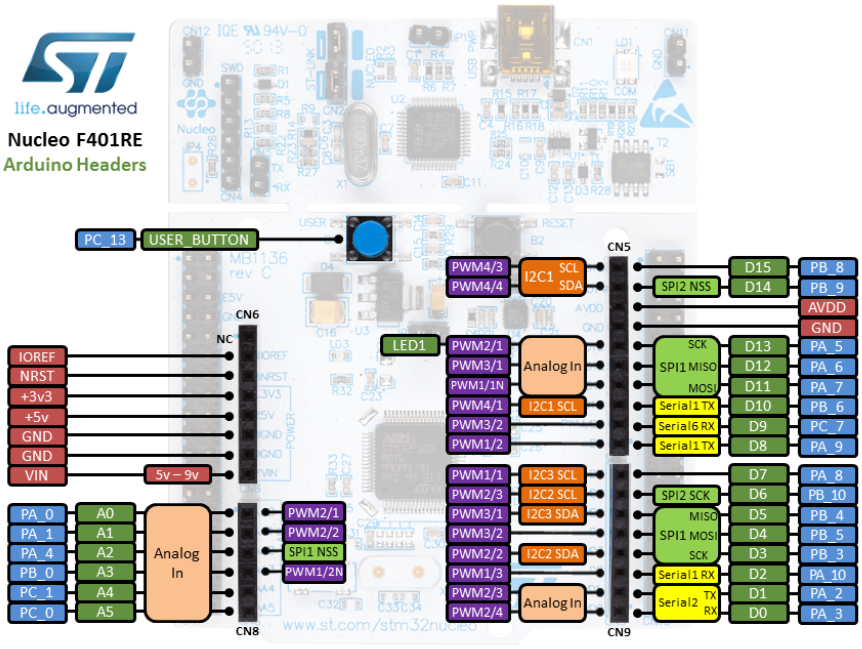

Figure 1 – Nucleo F401RE pin assignments

Arm mbed application shield

The mbed application shield shown in Fig. 2 connects to the Arduino-style header pins of the F401RE to add eight useful I/O tools to its functionality. This lab uses the **5-way joystick, RGB LED, speaker, and the two potentiometers**. The pin-mapping you will need to know when programming these tools is shown in Tab. 1.

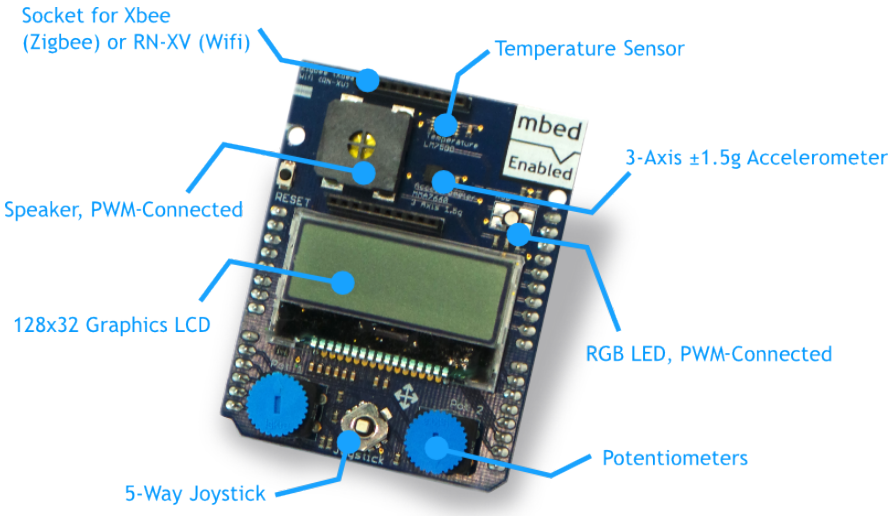

Figure 2 – Arm mbed application shield (labeled)

Table 1 – Pin mapping of the joystick and the RGB LED

Pin	mbed API
JOY_UP	PA_4
JOY_DOWN	PB_0
JOY_RIGHT	PC_0
JOY_CENTER	PB_5
JOY_LEFT	PC_1
RED_LED	PB_4
GREEN_LED	PC_7
BLUE_LED	PA_9
POT_1	PA_0
POT_2	PA_1
SPEAKER	PB_10

Keil µVision IDE

The µVision IDE combines project management, a run-time environment, build facilities, source code editing, and program debugging. The integrated debugging tool allows you to test, verify, and optimize the application code. As such, µVision is the only software tool you will need to program the STM32 microcontroller; it will be used throughout the remaining labs of this course. We strongly *suggest* you take the time to become familiar with the layout, as well as some basic C++ code, if you are not already.

Part A: Replicating Lab 3 with the mbed SDK

In this part of the lab, you will program the 5-way joystick on the mbed application shield to work as an interrupt source on the F401RE microcontroller. You will also program the RGB LED on the mbed shield to light up in response to an interrupt request in the following manner:

- Left press → Red light on — Left let go → Red light off
- Right press → Green light on — Right let go → Green light off
- Up press → Blue light on — Up let go → Blue light off
- Center press → White light on — Center press → White light off

The purpose of this part of the lab is to show how SDKs (like mbed) can make programming microprocessors much easier.

Procedure

A new code template has been provided for you on **cuLearn**, along with useful PowerPoint notes on interrupts, low-power features, digital inputs and outputs, and **software libraries**. You will need to finish the project, test it, and check with the TA before moving on to Part B. The following steps will guide you along the process:

1. Download and open the project "interrupt.uvprojx" in **Keil µVision**

2. In the **Project** window, expand "main.cpp" to see all of the included header files. Open "mbed.h" to get an idea of what is included with this SDK. *Take note that you will need to come back to some of these files later in the lab for information on how to use the methods they contain*

3. Familiarize yourself with "main.cpp" by reading the comments and C++ code already included. You will need to fill in code wherever you see the comment //Write your code here

4. Define the three LEDs in the **RGB LED** as digital outputs

5. Define the 4 switches you need in the **5-way joystick** as interrupt inputs. ***Hint:*** the **InterrputIn** interface in the mbed SDK (InterruptIn.h) is used to do this

6. Write the code for the four interrupt handlers so that a given LED will light up when its corresponding switch is pressed, and **turn off when that switch is let go**

7. Finish the **main()** method. The included comments will lead the way

8. Check if your code works. If it does, demo it to the TA

Part A Deliverables

Before moving on, make sure you have the following:

- Save the entire project for later reference (i.e., writing your lab report)

Part B: Using PWM and Timers to Make an Audio Player

In this part of the lab, you will program an audio player using the microcontroller's built-in timer, pulse-width modulation (PWM), and, of course, interrupts. The audio player program will play a simple yet elegant piece of music on the application shield's **speaker** and display the melody on the **RGB LED**. Two **potentiometers** will be used to adjust the speed and volume of the audio.

Procedure

You will need to finish the project, test it, and check with the TA before getting your checkout mark. The following steps will guide you along the process:

1. Open the project "timer_pwm.uvprojx" in **Keil µVision**

2. Familiarize yourself with "main.cpp" by reading the comments and C++ code already included. You will need to fill in code wherever you see the comment //Write your code here. Note that the notes and beat lengths have already been defined, along with the two arrays **note[]** and **beat[]** that define the song

3. Like in Part A, you will need to setup the tools that you will be using (i.e., the **RGB LED**, **speaker**, and two **potentiometers**). You will need to think about which interfaces are appropriate for each I/O. You will need to look into "mbed.h" and the other included header files for more information. If you are not familiar with **pulse-width modulation (PWM)**, you may want to take the time to get familiar with it. PWM essentially allows you to create an analog signal by digital means. Think about which components of this program could potentially benefit from PWM

4. Instantiate a **Ticker** object. Call it "ticker"

5. In **main()**, initialize "ticker"; it will be the (periodic) interrupt source for this program (i.e., whenever we get an interrupt request from "ticker," we will jump to the ISR and do what needs to be done). Nothing else is required in **main()**, as we're just waiting for interrupts

6. All that is left is to write the code for **ticker_ISR()**. The included comments will lead the way

7. Check if your code works. If it does, demo it to the TA

Part B Deliverables

Before moving on, make sure you have the following:

- Save the entire project for later reference (i.e., writing your lab report)

Lab report

Answer the questions in the **Google Form** posted on **cuLearn**.