**Part 1 (Sum= 10 marks)**

P1\_Q1\_(A): Using mbed SDK, write a line of code to configure the PB\_3 as a digital output.

DigitalOut COLOUR\_LED(PB\_3);

P1\_Q1\_(B): Using mbed SDK, write a line of code to configure the PC\_6 as an interrupt input

InterruptIn JOY\_DOWN(PC\_6);

P1\_Q1\_(C): Using mbed SDK, write two lines of code to attach an interrupt handler function named “my\_handler” to the rising and falling edge events on the pin you already configured as interrupt input in the previous question.

JOY\_DOWN.rise(my\_handler);

JOY\_DOWN.fall(my\_handler);

P1\_Q2: In part one of the lab, to turn a LED on, we set its state to “0”. What does that tell you about the hardware interface of the LEDs?

LEDs are active low meaning they are off when set to 1 and on when set to 0; this indicates that the hardware is a NAND gate

P1\_Q3: This lab provides essentially the same final functionality as Lab 3 (ignoring that the LEDs toggle in that lab instead). List 2 advantages and 2 disadvantage of using mbed SDK

Lab 3 was done using low level programming. Without general libraries, more specific hardware functions were able to be run. Lab 4 had libraries which meant it was easier to code and debug. However, we were limited by what libraries were included for us.

P1\_Q4: mbed SDK offers several component interfaces. Select three component interfaces that are not used in this lab and briefly explain what they are used for.

AnalogOut: used to set the voltage of an analog output pin

PortIn: specifies which pins of the GPIO are inputs and able to read the values of the pins

BusOut: combines numerous DigitalOut pins to write them simultaneously

**Part 2 (Sum= 10 marks)**

P2\_Q1\_(A): Using mbed SDK, write a line of code to configure the PA\_2 as a PWM (Pulse Width Modulation) output.

PwmOut my\_pwm(PA\_2);

P2\_Q1\_(B): Using mbed SDK, write a line of code to configure the PC\_4 as analog input.

AnalogIn Ainput(PC\_4);

P2\_Q1\_(C): If “my\_pwm” is the name of a PWM output, using mbed SDK, write a line of code to set its frequency to 100Hz.

my\_pwm.period(0.01);

P2\_Q1\_(D): Add another line of code to the previous question to set the duty cycle of the PWM output to %80.

my\_pwm.write(Ainput\*0.8);

P2\_Q1\_(E): If “my\_ticker” is the name of a ticker object and if “my\_handler” is the name of a handler function, using mbed SDK, write a line of code that calls the handler every 1 milliseconds.

my\_ticker.attach(my\_handler, 0.001)

P2\_Q2: To interface with the following elements of this lab, explain what GPIO configuration you used and why.

A\_ Speaker: PwmOut - (Pulse Width Modulation) allows for a digital signal to produce an analog signal which is required to drive the speaker output, by changing the duty cycle the speaker became louder/quieter.

B\_ Potentiometers: AnalogIn - used because it takes the resistance produced by the potentiometer as an analog value and converts it to be able to change the speed and volume of the speaker.

P2\_Q3\_(A): Explain what property of the PWM signal you changed to generate different notes, and how you did that.

Changed the frequency of the PWM signal which intern changed the sound of the produced tone. Did this using the following line of code “SPEAKER.period(note[k]);” where note[] is the array containing the various frequencies of the song and k is an integer used to step through the array .

P2\_Q3\_(B): Explain what property of the PWM signal you changed to control the volume of the audio, and how you did that.

Used the line of code SPEAKER.write(POT\_2\*0.8);

P2\_Q4: Explain what property of the TICKER you changed to control the beats. How did you change that for each note?

timer.attach(ticker\_ISR,beat[k]\*POT\_1); // pot 1 changes beat speed

Used k to loop through beat arry

P2\_Q4: List 2 advantages of using C++ for coding embedded processors?

* It has a standard library
* Ability to build reusable parts of code

P2\_Q5: Using the provided lab materials, explain the differences between the mbed API and CMSIS. How do they interface to each other (how does mbed access CMSIS and the underlying hardware)?

Mbed API controls the microprocessor peripherals (contains all the C++ code and libraries that we needed to complete the lab)

CMSIS is a low level component of Cortex-M; it gives access to the registers and interrupts.

They interface eachother since mbed API uses CMSIS to interface.