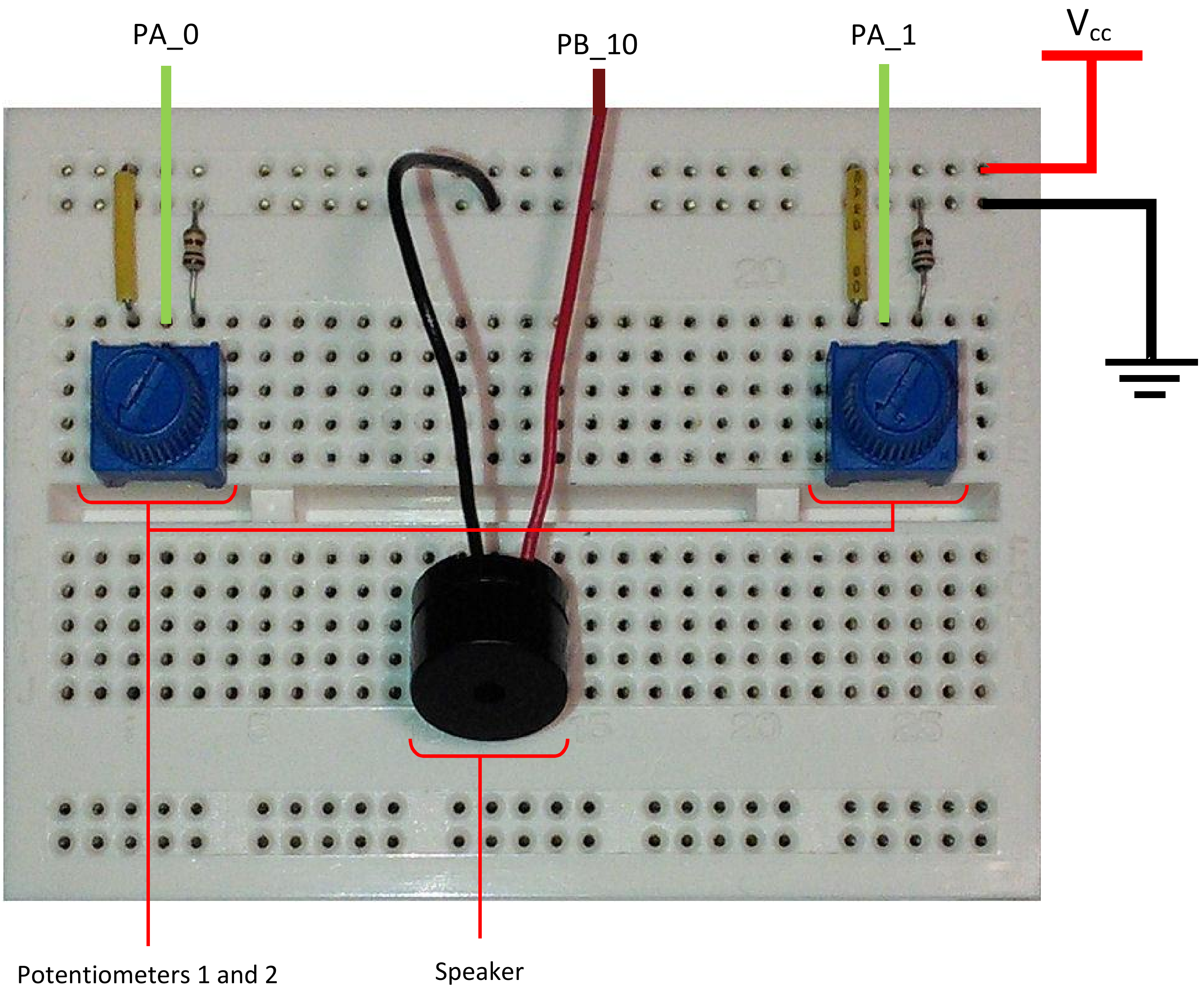
LAB EXERCISE: ANALOG INPUT AND OUTPUT

OVERVIEW

In this lab, you will learn to generate audio waves using PWM, and use two potentiometers to tune the volume and pitch of the audio.



IMPLEMENTATION DETAILS

HARDWARE

NUCLEO F401RE BOARD

The Nucleo F401RE board pin descriptions are shown below:

Pin	Pin name in mbed API
Potentiometer 1	PA_0
Potentiometer 2	PA_1
PWM speaker	PB_10

AUDIO FREQUENCIES

A sound is essentially an air wave. The amplitude and the frequency of the wave decide the volume and the pitch of the sound respectively. The speaker (or headphone) inputs electrical signals (voltages), and use them to turn a coil into an electromagnet, which can either attract or repel the magnet that moves back and forth. The motion of the magnet will further push and pull a diaphragm and create air waves (just like a drum).

In this lab you will have to use the PWM output to generate electrical waves which can be turned into sound by the on-board speaker. Be aware that the frequency of the wave has to be in a specific range so that humans can hear it (see this reference: http://en.wikipedia.org/wiki/Audio_frequency)

PWM

Pulse width modulation (PWM) is a simple method of using a rectangular digital waveform to create an analog output. PWM uses the width of the pulse to represent an amplitude.

The period of the wave is usually kept constant, and the pulse width, or ON time is varied.

The duty cycle is the proportion of time that the pulse is ON or HIGH, and is expressed as a percentage:

Duty cycle = 100% * (pulse ON time)/(pulse period)

Whatever duty cycle a PWM stream has, there is an average value. If the ON time is small, the average value is low; if the ON time is large, the average value is high. Therefore, by controlling the duty cycle, we control the average output value (represented as the red line below).

Here are some examples showing, how to use the PWM:

```
PwmOut led(D5);         //define the PWM output
led = 0.5;              //set the duty cycle of PWM output to 50%
led.period(0.02)        //set the output frequency to 50Hz
```

WAVE GENERATION

We can use `for` loops to generate simple waveforms without using lookup tables. For example, for a saw-tooth wave we could use:

```
for(i=0; i<1; i+=0.05)
```

```
        out = i;
```

or, for a triangle wave use two `for` loops:

```
for(i=0; i<1 i+=0.025)
        out = i;
for(i=1; i>0; i-=0.025)
        out = i;
```

SOFTWARE FUNCTIONS

The functions that maybe used in this lab are listed below:

Function name	Description
Analog input	
AnalogIn (PinName pin)	Create an AnalogIn , connected to the specified pin
float read ()	Read the input voltage, represented as a float in the range [0.0, 1.0]
unsigned short read_u16 ()	Read the input voltage, represented as an unsigned short in the range [0x0, 0xFFFF]
operator float ()	An operator shorthand for read()
Other functions	
void wait (float s)	Wait for a number of seconds
void wait_ms (int ms)	Wait for a number of milliseconds
void wait_us (int us)	Wait for a number of microseconds
PWM output functions	
PwmOut (PinName pin)	Create a PwmOut connected to the specified pin.
Void write (float value)	Set the ouput duty-cycle, specified as a percentage (float)
float read ()	Return the current output duty-cycle setting, measured as a percentage (float)
void period (float seconds)	Set the PWM period, specified in seconds (float), keeping the duty cycle the same
void period_ms (int ms)	Set the PWM period, specified in milli-seconds (int), keeping the duty cycle the same
void period_us (int us)	Set the PWM period, specified in micro-seconds (int), keeping the duty cycle the same
void pulsewidth (float seconds)	Set the PWM pulsewidth, specified in seconds (float), keeping the period the same
void pulsewidth_ms (int ms)	Set the PWM pulsewidth, specified in milli-seconds (int), keeping the period the same
void pulsewidth_us (int us)	Set the PWM pulsewidth, specified in micro-seconds (int), keeping the period the same

PwmOut & operator= (float value)	A operator shorthand for write()
PwmOut & operator float ()	An operator shorthand for read()

YOUR APPLICATION CODE

- Define analog input, PWM and serial output ports
  - Two analog inputs for two potentiometers
  - One PWM output for the speaker
- Generate waves to the PWM output
  - Generate a saw-tooth sound wave (design the range to be from 320Hz to 8kHz)
  - Use one potentiometer to adjust the volume and the other one to tune the pitch