Lab 4: Use of Digital to Analog converter along with microcontroller.
Report Due: 13th Oct
Demo Due: 16th Oct

Learning Objective: In this lab, you are going to

1. Learn about generating basics of waveforms and measuring them using oscilloscope.
2. Explore generating low frequencies for various sounds and music.
3. Use DAC chip to create different waveforms.

Components Needed:
Pi 3 Board, resistors, buzzer. We will use a DAC chip MCP4725 for all the lab activities below. You are allowed to use the adafruit open source library, whose detail can be found here: https://learn.adafruit.com/mcp4725-12-bit-dac-with-raspberry-pi .

Programming requirements:
This Lab application can be written by python or C.

Lab Activities:

1. **Generating Square Wave:** A simple square wave is a waveform in which the amplitude strictly alternates between a fixed minimum and maximum. A square wave can be created by programming Raspberry Pi to be available at GPIO output pin by using digitalwrite().
   In lab 1, when you blink your LED, you are sending a square wave out from the pin to the LED. Here is a simple C solution to generate a square wave with frequency 0.1MHz (1/(5+5 μs)):

```
Void SquareWave(){
    while(1){
        digitalWrite(Pin, 1)
        delayMicroseconds(5)
        digitalWrite(Pin, 0)
        delayMicroseconds(5)
    }
}
```

   By changing the delay period we can change the frequency of the square wave.
   Activity 1: Write code or modify the code above to generate digital signal waveform with different frequencies: 200 MHz, 600 MHz & 800 MHz.
   Activity 2: Generate triangular signal waveform of the above frequencies. Write another C program for triangular waveform.
   Question 1: Find out the highest frequency that a raspberry pi can generate using digital output and write down the answer and explanation in your lab report.

2. **Measuring signal waveforms using oscilloscope:** Waveforms are specified by their frequency and amplitude. While regular voltmeter can measure point value of a signal at

a time, oscilloscope can show the continuous values over a period of time of a signal. An oscilloscope can be used to display and analyze waveform of electronic signals. In other words, it draws a graph of a signal voltage as a function of time.

To view the waveform on the oscilloscope, connect the probe test cable to the signal source (red to output pin, black to ground pin). Most scopes will produce a two-dimensional graph with time on the x-axis and voltage on the y-axis.

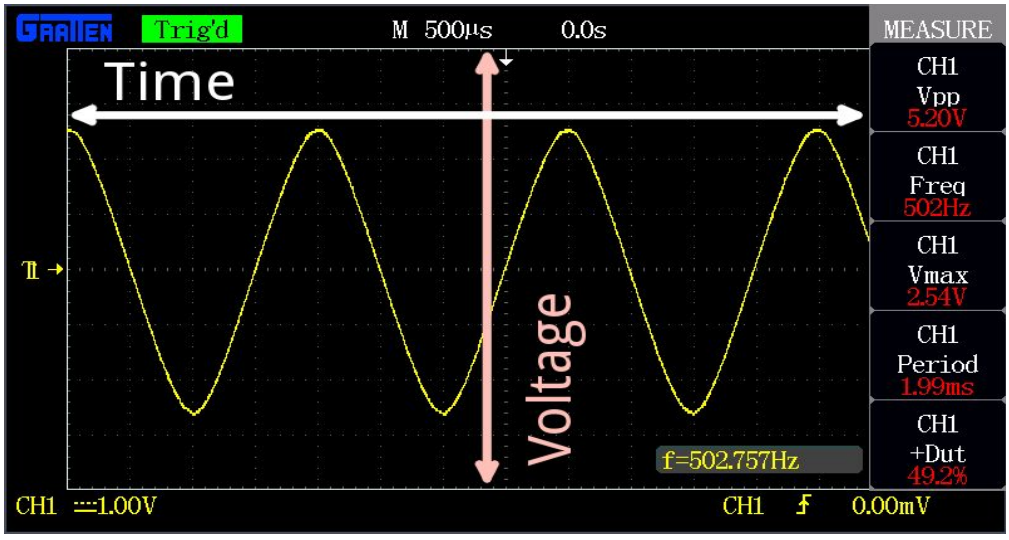


Figure a. A sample oscilloscope output provided by Sparkfun

On the x-axis, you can find out the timing characteristics, such as frequency, period, duty cycle, & rising/falling time, using a horizontal control to adjust time scale.

On the y-axis, you can find out the voltage characteristics, such as amplitude and maximum/minimum voltage, using a vertical control to adjust voltage scale.

In lab 5, you will be asked to build a oscilloscope to read and visualize the wave you generated in lab 4. We will discuss the detailed specification about oscilloscope in the next lab.

Activity 3: Read the square wave you generated in the part 1 with an oscilloscope to ensure that you have generated a wave with the desired frequency. Use control knobs on the oscilloscope panels to measure the highest voltage and the lowest voltage.

Question 2: Does the highest reading steadily stay at 5V? If not and there are noise, where does the noise come from?

3. **Generating Music using Raspberry Pi and output with a buzzer:** We will learn how to use a buzzer to output music and write a program to adjust the frequency of a square/sine wave based on user input.

   Generally speaking, a tone is a particular frequency of sound. When we apply an audible signal (voltage, frequency) to the small buzzer, it makes continuous sound. Buzzer is an actuator that converts frequency into sound. Please note that a voltage does not make a tone but frequency is the cause of tone. Let's make a buzzer to

produce a C4(middle C) tone. According to this document, “Middle C” in the music has a frequency 261 Hz (reference:https://pages.mtu.edu/~suits/notefreqs.html).

Then we need to generate a square wave that has a frequency equal to 261 Hz. This means the every 1 second contains 261 cycles of the waveform (1 Hz = 1 cycle per second). The following function will produce a 261 Hz square wave for 1 second:

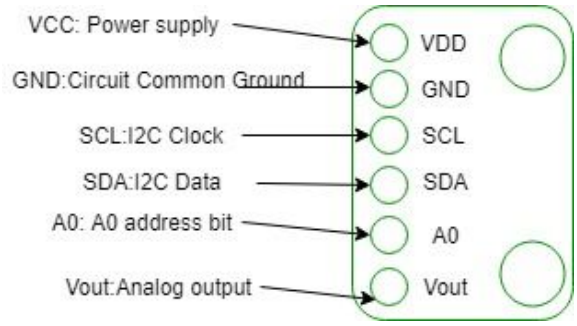
```
#include <wiringPi.h>
#include <stdio.h>

#define outputPin 0
#define C4 261
#define period 1000 //1 second
void tone(){
        long half_cycle = (long)(1000000/C4)
        long duration = (long)((period*1000)/(half_cycle*2));
        for(int i = 0; i < time; i++){
                digitalWrite(outputPin, HIGH);
                delayMicroseconds(half_cycle);
                digitalWrite(outputPin, LOW);
                delayMicroseconds(half_cycle);
        }
}
Void run(){
        tone();
        digitalWrite(outputPin, LOW);
        delay(20);
}
Int main(){
    if(wiringPiSetup() == -1){
        Return 1
    }
    pinMode(outputPin, OUTPUT);
    while(1){
        run();
    }
}
```

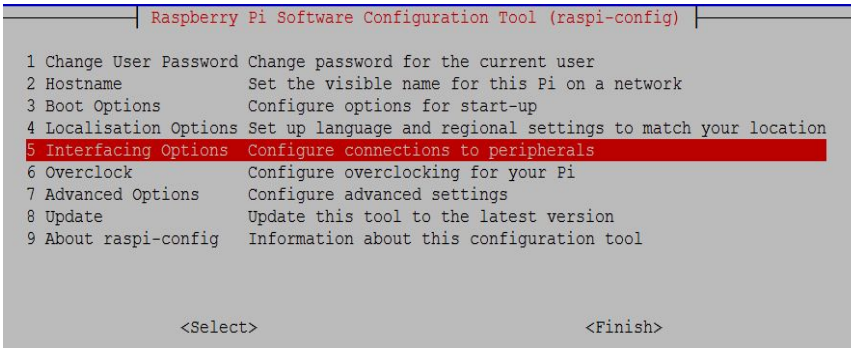
Activity 4: Modify the code above to let the small buzzer produce a piece of music. The **Tone()** function should be modified into **Tone(unsigned int frequency, unsigned int period)**, which will take a frequency input and a tone period. This tone function will be used multiple times to create the piece of music. Some sample music notes can be found at the end of the lab manual. Music notes can be hard coded in your main function.

4. **From Digital output to Analog output:** Now you are familiar with generating the digital output wave and how to adjust it to a correct frequency. In this section you will learn

about how to convert a digital signal into analog signal, and create waves other than a square wave.



We provided you a DAC MCP4725 12-bit chip, which will help you convert digital signal into analog. 12 bit means that it will accept up to 4096 possible inputs to provide a analog input, where 0 is 0V and 4096 is the full scale voltage (which determined by the voltage you supply to the VCC pin). According to the data sheet the voltage can be around 2.7V to 5.5V.



There are six pin on the chip package. SDA will send data from Raspberry pi to chip(0~4096), and the SCL(clock) will control the output rate.

You will need to enable your Raspberry Pi I2C functions via configuration in order to send signal via I2C ports. This can be done via the command “sudo raspi-config”. In the menu that appears, choose option 5, “Interfacing Options”. Then choose option P5, “I2C”. Lastly, Confirm that you want to enable the I2C interface And to control the I2C output, you can install Adafruit Python Library found Here.

Activity 5: Use the code (python) frame we provided below to generate a sine wave, square wave, and triangle wave. Check that they are being generated as expected by displaying them on a oscilloscope.

```
sin_wave():
    t = 0.0
    tStep = 0.05
    while True:
        voltage = 2048*(1.0+0.5*math.sin(6.2832*t))
         dac.set_voltage(int(voltage))
         t += tStep
         time.sleep(0.0005)
```

Question 3: Explore and write down how you can convert a digital PWM (Pulse width modulation) into analog signal. (e.g. possible circuit design, software conversion)

5. **Building a Function generator using Raspberry Pi:** In this section, you are going to build a function generator using what you learned from sections 1- 4. The command window should display nothing until an external button is pressed. Then the system should ask for 3 input:

i. Shape of the waveform
ii. Frequency
iii. Max output Voltage

When the inputs are confirmed, the raspberry pi should output the correct wave with correct characteristic continuously until the button is pressed again. Then the system should ask for 3 inputs again (continuing the cycle).

Question 4: What is the max frequency you can produce by your raspberry pi functional generator? Write it down in your report.

Question 5: Explain your design for the functional generator (you can use diagram to visualize your system state machine, what function you implemented, etc)

Demonstration: You will need to demo both music buzzer (part 4 activity) and functional generator(part 5 activity) to your TA in the lab (Their hardware implementations should not conflict with each other, so put both projects on the breadboard and run the programs one by one). Also, answer the four questions in the lab manual and submit your solution to ecampus by the deadline.

Sample Music note can be used in part 3: format: tone(time(second))
(all middle):
G(0.75) A(0.25) G(0.5) F(0.5) E(0.5) F(0.5) G(1) D(0.5) E(0.5) F(1) E(0.5) F(0.5) G(1)
G(0.75) A(0.25) G(0.5) F(0.5) E(0.5) F(0.5) G(1) D(1) G(1) E(1) C(1)