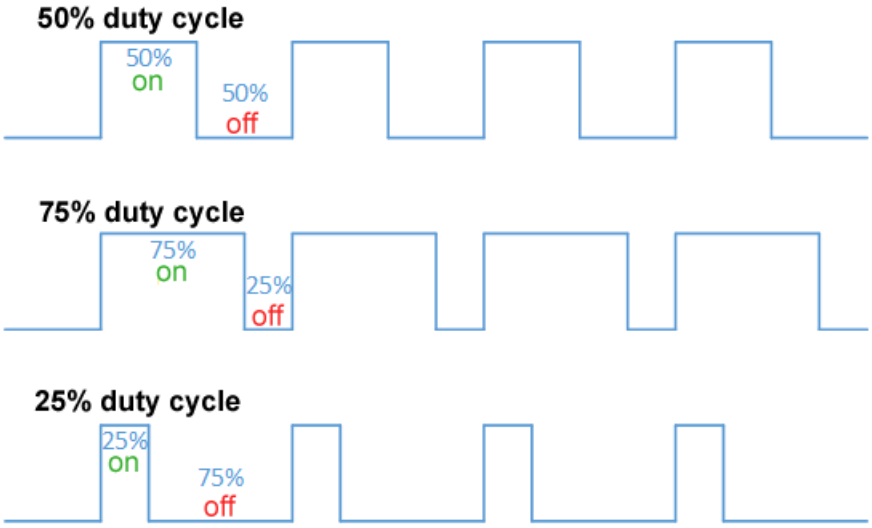
UNIVERSITY OF MINNESOTA DULUTH

CS 4995/5995 – LAB 08: PWM, PSEUDO-ANALOG AND SERVOS

INTRO TO PWM

PWM (Pulse-Width Modulation) is a simple method of producing pseudo-analog values using digital signals. Simply speaking, PWM toggles an output HIGH and LOW at a specified frequency. We can control the voltage of the pseudo-analog signal by varying the “duty-cycle” or the on time of the signal as described below:



If a PWM signal has a 50% duty cycle, the voltage read by a multimeter in DC mode will be 50% of the maximum voltage (assuming a high enough frequency).

The frequency and duty cycle can be calculated as follows:

Frequency = 1 / (on_time + off_time)
Duty cycle = on_time * frequency

Servos are a special case, as they use PWM as a control scheme rather than an analog signal. Servos require a PWM frequency of 50Hz, and an on_time between 0.5ms (2.5% duty cycle) and 2.5ms (12.5% duty cycle) for 0 deflection and full deflection respectively.

In this lab, we will be using PWM to control the brightness of an LED and the rotation of a servo based on a potentiometer.

USEFUL TOOLS

- Interpolation – Converts a value from one range to another
  - You can use numpy interp to accomplish this
  - An example of how to use numpy interp can be found in *servo.py*
- Raspberry Pi IO Library
  - This is the library you will use to generate PWM signals on digital pins.
  - An example which sweeps the servo back and forth can be found in *servo.py*

YOUR TASKS

1. Complete the potentiomer_read function – This function should read the value from the analog port where you connected your potentiometer and return it. The ADC has already been initialized for you and can be accessed through the *potentiometer_adc* variable.
2. Complete the *main* function. Once done, this function should do the following:
   a. Initialize two PWM channels. One for the servo and one for the LED. The servo channel should be set with a frequency of 50Hz and the LED channel should be set with a frequency of 100Hz.
   b. Retrieve the current value of the potentiometer using your potentiometer_read function.
   c. Using the interpolate function, convert the potentiometer value to an on time between 0.5ms and 2.5ms (2.5% and 12.5% duty cycle), and pass it to the servo PWM channel
   d. Using the interpolate function, convert the same potentiometer value to an on time between 0ms and 10ms (0% and 100% duty cycle), and pass it to the LED PWM channel.
   e. Loop forever (with a 10ms delay between loops)

EXERCISES

Step 1: Setup the Hardware

With the Raspberry Pi OFF, attach the grove shield to the Raspberry Pi 40 pin GPIO header. Be careful not to bend pins. Next, with the Pi either ON or OFF, attach the potentiometer to an analog port, and the Servo and LED module to two digital ports.

Step 2: Download the code

Copy the program files from our github (if you haven't already):

```
$ git clone https://github.umn.edu/akhan/S_IoT_F_21.git
```

If you already have the repository downloaded (you should), instead run "git pull" from within the repository to retrieve the latest changes. If it warns you about overwriting your code, you can save it by running the following commands

```
$ git add *
$ git commit -m "Saving Lab 7"
$ git pull
```

Step 3: Run the setup

Run the setup scripts by running the following commands:

```
$ cd Lab_08_PWM
$ chmod +x setup.sh
$ ./setup.sh
```

Step 4: Complete the Code

Open lab8.py in the text editor of your choice. Complete the code as described above, then demonstrate your code to the TA.