Southern New Hampshire University

CS 350 Milestone One PWM Lab Guide

In this milestone, you will build on your previous work with Raspberry Pi and embedded systems by writing Python code to blink an LED using pulse width modulation (PWM). PWM allows the programmer to vary both the pulse frequency and the pulse's duty cycle (sometimes referred to as the duration). In the case of an LED, PWM can be used to fade the light in or out or blink it at a specific frequency, but the principles of PWM can be useful for many other applications, such as operating motors, motor controllers, or charge controllers.

In Module One, you created a circuit for a controllable LED through a simple on/off command. Now, you will use the same circuit you created in the Module One assignment and apply some software changes to use PWM to control the LED. This lab is critical in developing your embedded programming skills and will serve as a foundation for future assignments.

Step 1: Initial Software Setup
You will be using the Milestone1.py file provided for this module. Examining the file provided, you will see the following:

1. The PWM frequency for GPIO line 18 is set to 60 Hz (the same frequency as many HD video screens and monitors) with the following command:

   ```
   pwm18 = GPIO.PWM(18, 60)
   ```

2. The PWM output is started with a 50% duty cycle (it is on half the time and off half the time) with the following command:

   ```
   pwm18.start(50)
   ```

When you run this script, you will see the LED turn on, similar to the tests you ran in Module One, but the LED is only powered for half the time. Because the frequency is set to 60 Hz, it is only perceived as being slightly less bright than being on 100% of the time.

Step 2: Understanding Changes in Frequency
In this step, you will see what happens when you change the frequency for the PWM instance. Using an editor of your choice, edit the Milestone1.py file to reduce the PWM frequency on GPIO line 18. This process may require you to transfer an updated script back to your Raspberry Pi if the editor is not on it.

Your goal here is to reduce the frequency in small enough increments to determine when you can visually see the LED start to blink.

Step 3: Understanding Changes in Duty Cycle
In this step, you will change the duty cycle for the PWM independently of the frequency. Using an editor of your choice, edit the Milestone1.py file to change the PWM frequency on GPIO line 18 back to 60 Hz.

Your goal here is to reduce the duty cycle of the PWM function on GPIO line 18 in small enough increments so that you can tell when there has been a perceptible change in the brightness of the LED.

Step 4: Using PWM to Fade the LED In and Out
In this step, you will modify the code in the Milestone1.py template further. You will update the duty cycle for the PWM function on GPIO line 18 in small steps, going from 0 to 100 and then back again.

Begin by commenting out lines 46–53 to make this section of the template look like the following:

```
## Start the PWM instance with a 50% duty cycle
#pwm18.start(50)
#
## Wait for input to stop the program
#input('Press return to stop:')
#
## Stop the PWM
#pwm18.stop()
```

Now, you will need to add some code in its place, including two simple loops.

```
# Start the PWM instance on GPIO line 18 with 0% duty cycle

      pwm18.start(0)

# Loop from 0 to 100 in increments of 5, and update the duty cycle
# Accordingly, pausing 1/10th of a second between each update for duty
cycle in range (0, 100, 5):

      pwm18.ChangeDutyCycle(dutyCycle)
      time.sleep(0.1)

# Loop from 100 to 0 in increments of -5, and update the duty cycle
# Accordingly, pausing 1/10th of a second between each update for duty
cycle in range (100, 0, -5):

      pwm18.ChangeDutyCycle(dutyCycle)
      time.sleep(0.1)

# Stop the PWM instance on GPIO line 18

      pwm18.stop()
```

Your goal here is to understand the impact these changes in the duty cycle have on the LED's behavior.

Step 5: Building Up More PWM Capabilities
To complete this portion of Milestone One, you must put the actions to fade the LED in and out (the for-loop code from Step 4) into a loop that will run until it is broken by a keyboard interrupt (Ctrl-C). Your code should use a try/except construct to catch the keyboard interrupt and ensure the code is cleaned up appropriately.

You can reference the SimplyBlink.py script from Module One for an example of the try/except block necessary to complete this portion of the Milestone.

Step 6: Recording Your Success
Once you have completed Step 5, run your Milestone1.py script and take a short video (no more than 30 seconds) where you announce your name while recording what your LED is doing while your script is running. This will be submitted as evidence of your completion of Milestone One.

Lab Questions

1. At what frequency can you see the LED start to blink?
2. At what duty cycle is the intensity of the LED perceptibly diminished from the initial 50% duty cycle?
3. When changing the duty cycle of the PWM, the loop used an increment of 5 every tenth of a second. Was this perceptibly smooth? If not, what could you change to improve the visual response? Why?
4. What function sets the PWM frequency for a GPIO line?
5. What function sets the duty cycle for a GPIO line?