Lab Report 5: Revolutions per Minute

Max Huggins – UCA Department of Physics
Astronomy

February 9, 2021

Abstract

An experiment was done to determine the revolutions per minute (RPM) of a cooling fan at various duty cycles. The Raspberry Pi was programmed to control the fan's power input using pulse width modulation (PWM). RPM was measured using an infrared (IR) sensor and an IR LED.

1 Introduction

This experiment consisted of two main parts. Sensing revolutions of a fan using an IR sensor and LED and controlling average power input to the fan using PWM.
The same principles used in the previous experiment, Determining Acceleration Due to Gravity, for determining revolutions are similar for finding revolutions of the fan blades. The IR sensor is normally HIGH, but goes LOW once the IR LED is blocked. The Pi senses for this and records data accordingly. The average power sent to the fan is controlled by sending a digital pulse to a transistor in the switch configuration. Each pulse allows a current to flow from the supply voltage through the fan through the transistor to ground. The length of the pulse is referred to as the duty cycle and the number of pulses per second is the frequency. The circuit in figure 1 shows the setup. A consideration that had to be made was the oscillatory behavior of the PWM techniques used. The fan, while in transition from one duty cycle to the next will experience a change in equilibrium state of the fan. This means oscillations from the previous duty cycle can cause unusual RPM data collection. To minimize this a certain wait time was used after changing of duty cycles before more RPM data was collected. Figure 2 shows RPM vs. duty cycle at different wait times. Figure 3 shows a wait time of 17 seconds specifically. It can be seen that longer wait times allow oscillations from the previous duty cycle to settle.

2 Experimental Setup

Here are several figures showing schematics, plots, and charts for the experiment.

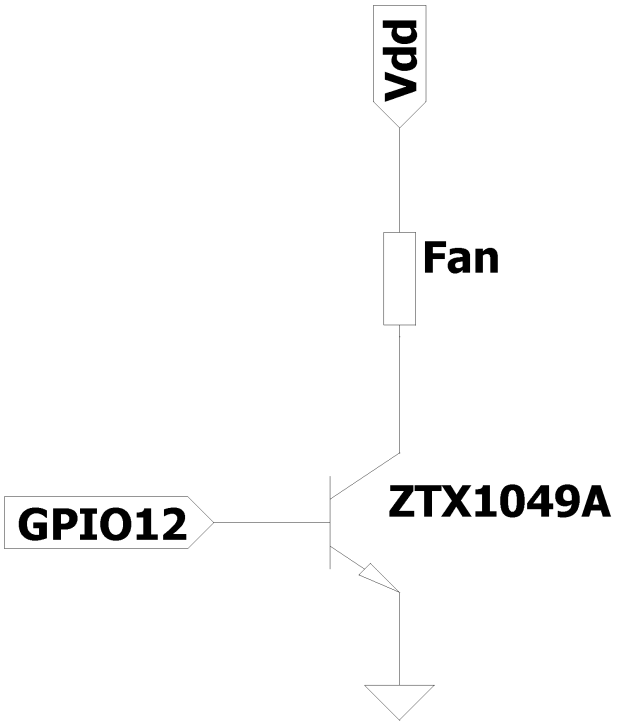

Figure 1: Schematic used for power handling.

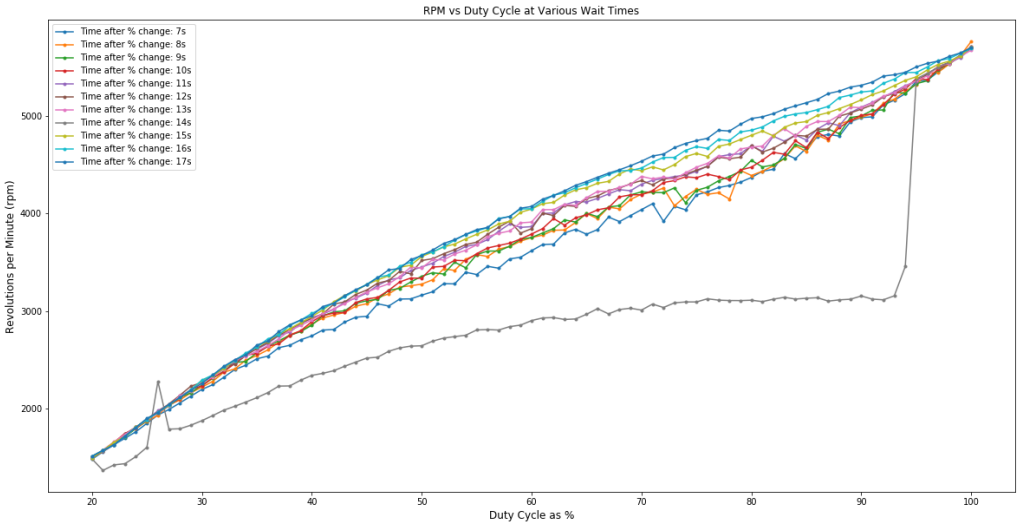

Figure 2: RPM vs duty cycle data for various wait times.

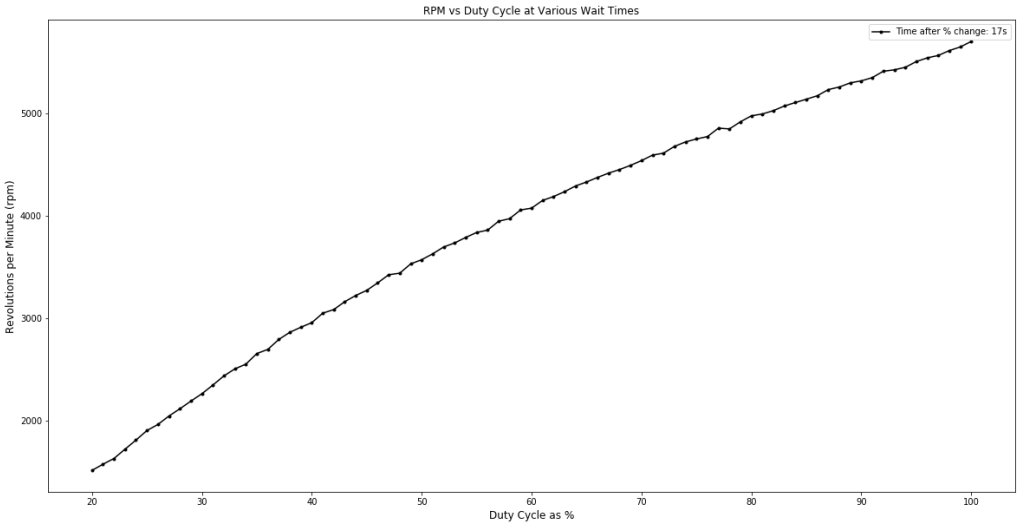

Figure 3: RPM vs duty cycle data for 17 seconds wait time.

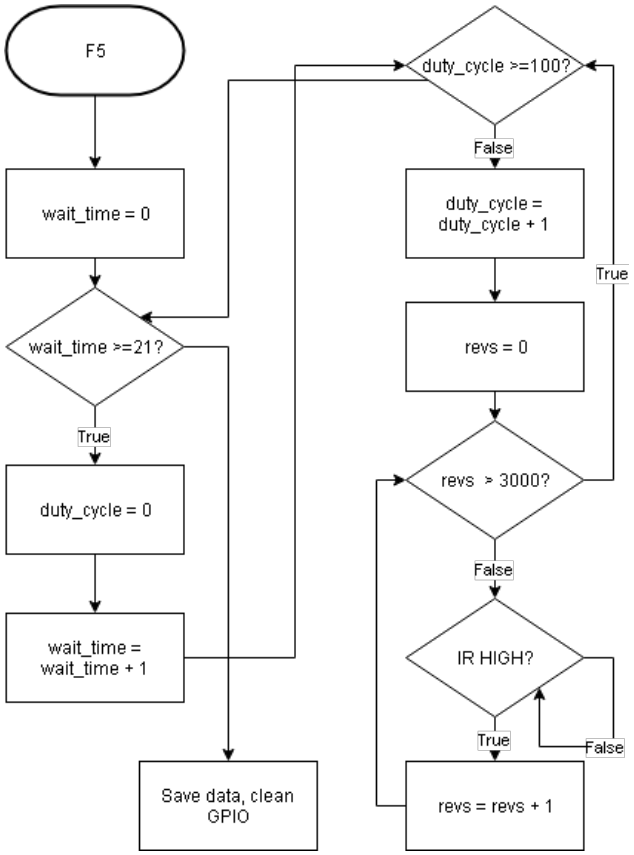

Figure 4: General flow of the script used for the Pi.

3 Conclusion

A method called pulse width modulation was used to control power input to a fan from the Raspberry Pi and the revolutions per minute was measured from the methods used in a previous experiment. These methods involved an IR sensor and LED. Considerations had to be made to dampen oscillations from changes in equilibrium states.
Considerations to be made for furthering this are:
1) Increasing wait times
2) Interpolating data for further processing (f.e. error analysis)
3) Using a power supply that can output enough current over long periods of time to ensure that the fan is operating in similar conditions over time.

4 Appendix

This is the main code.

```
import RPi.GPIO as GPIO
import time

FAN = 12
IR = 37

GPIO.setmode(GPIO.BOARD)
GPIO.setup(IR,GPIO.IN)
GPIO.setup(FAN,GPIO.OUT)

freq = 1
lowest_duty = 20
my_pwm = GPIO.PWM(FAN, freq)

duty_cycle = 0
my_pwm.start(duty_cycle)
pause_time = 2

try:
    for p in range(0,21,1):
        RPMs = []
        duty_cycles = []
        my_pwm.ChangeDutyCycle(lowest_duty)
        time.sleep(30)
        for i in range(lowest_duty,101,1):
            revs = 0
            duty_cycle = i
            print('the duty cycle is:', duty_cycle)
            my_pwm.ChangeDutyCycle(duty_cycle)
            time.sleep(p)
            start_time = time.time()
            while revs <= 3000:
                IR_state = GPIO.input(IR)
                if IR_state == False:
```

```
                    while IR_state == False:
                        IR_state = GPIO.input(IR)
                        pass
                    revs = revs + 1
            total_time = (time.time() - start_time) / 60
            RPM = (revs / total_time) / 7
            RPMs.append(RPM)
            duty_cycles.append(i)
        print(RPMs)
        file = open('./RPMDATA/RPMs{}.txt'.format(p), 'w')
        for n in range(len(RPMs)):
            #Write the data as comma delimites
            file.write(str(duty_cycles[n]) + ',' + str(RPMs[n]) + '\n')
        #always close the file you are using

        file.close()

except KeyboardInterrupt:
    my_pwm.stop()
    print('toast got burnt')

finally:
    GPIO.cleanup()
```

This is the data processing script.

```
# -*- coding: utf-8 -*-
"""
Created on Fri Apr  5 14:49:07 2019

@author: maxhu
"""

import numpy as np
import matplotlib.pyplot as plt

RPMs = [[] for i in range(0,21)]
duty_cycles = [[] for i in range(0,21)]

for i in range(0,21):
    lines = np.loadtxt('../RPMDATA/RPMs{}.txt'.format(i), delimiter=',')
    for line in lines:
        RPMs[i].append(line[1])
        duty_cycles[i].append(line[0])

plt.figure(figsize=(20,10))
for i in range(7,18):
    plt.plot(duty_cycles[i],RPMs[i], '.-', label = 'Time after % change: {}s'.
   format(i))
plt.legend(loc='best')
plt.xlabel('Duty Cycle as %', fontsize=(12))
plt.ylabel('Revolutions per Minute (rpm)', fontsize=(12))
plt.title('RPM vs Duty Cycle at Various Wait Times', fontsize=(12))
plt.savefig('RPM_Data')

```

```
plt.figure(figsize=(20,10))
plt.plot(duty_cycles[17],RPMs[17], '.-', color = 'black', label = 'Time after
    % change: 17s')
plt.legend(loc='best')
plt.xlabel('Duty Cycle as %', fontsize=(12))
plt.ylabel('Revolutions per Minute (rpm)', fontsize=(12))
plt.title('RPM vs Duty Cycle at Various Wait Times', fontsize=(12))
plt.savefig('RPM_Data_Single')
```