Lab Report 4: Determining Acceleration Due to Gravity

Max Huggins – UCA Department of Physics
Astronomy

February 9, 2021

Abstract

An experiment was done to obtain a result for the acceleration due to gravity (g) on Earth's surface. This was done with an infrared (IR) sensor, IR LED, and a picket fence. A value of $(9.82 \pm .06)\frac{m}{s^2}$ was found for g. This is a % difference of .10% when compared to the known value of $9.80665\frac{m}{s^2}$ and this known value falls within the range allowed by the standard deviation.

1 Introduction

To determine g, an average velocity was found between each picket. Time data was taken between at the center of each picket and this could then be plotted with the average velocities. These velocities were determined with the IR sensor and LED. The time for determining velocities should not be the same. The center time is found by these steps:
1) Sensor LOW
2) Start time
3) Sensor HIGH
4) Stop time
5) Sensor LOW
Whereas the time for calculating average velocity over the distance of an opaque and transparent section was found this way:
1) Sensor LOW
2) Start time
3) Sensor HIGH
4) Sensor LOW
5) Stop time
The slope of the average velocities versus center times graph will give the acceleration (Refer to figure 1). This is because:

$$a = \frac{\Delta V}{\Delta T} \tag{1}$$

The program was first written to only find one value of g, but was modified such that ten trials could be run to determine an average with reduced error. Values for g are listed below.

Gravity ($\frac{m}{s^2}$)
9.744
9.915
9.823
9.817
9.745
9.854
9.874
9.809
9.740
9.895
9.764

2 Experimental Setup

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

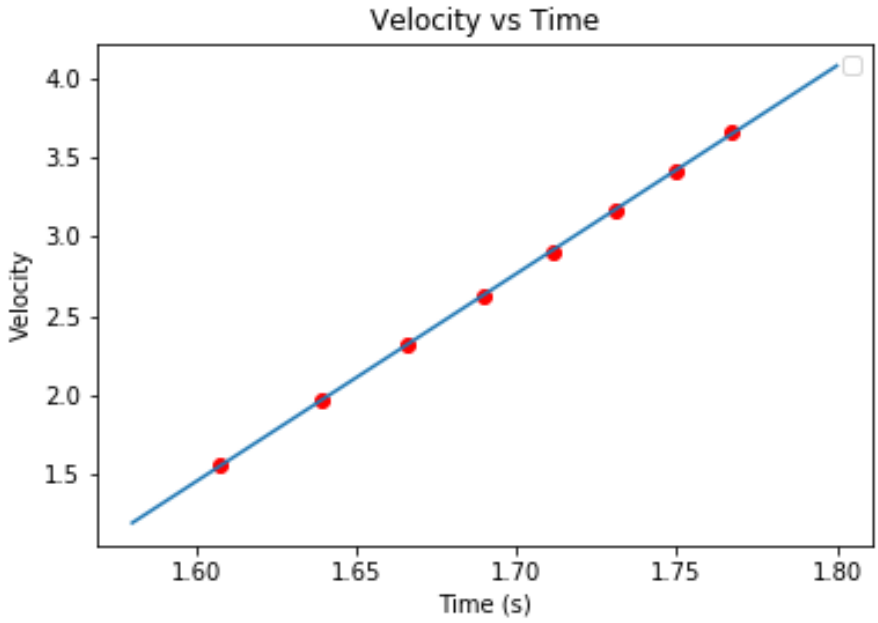

Figure 1: Example average velocity vs time graph.

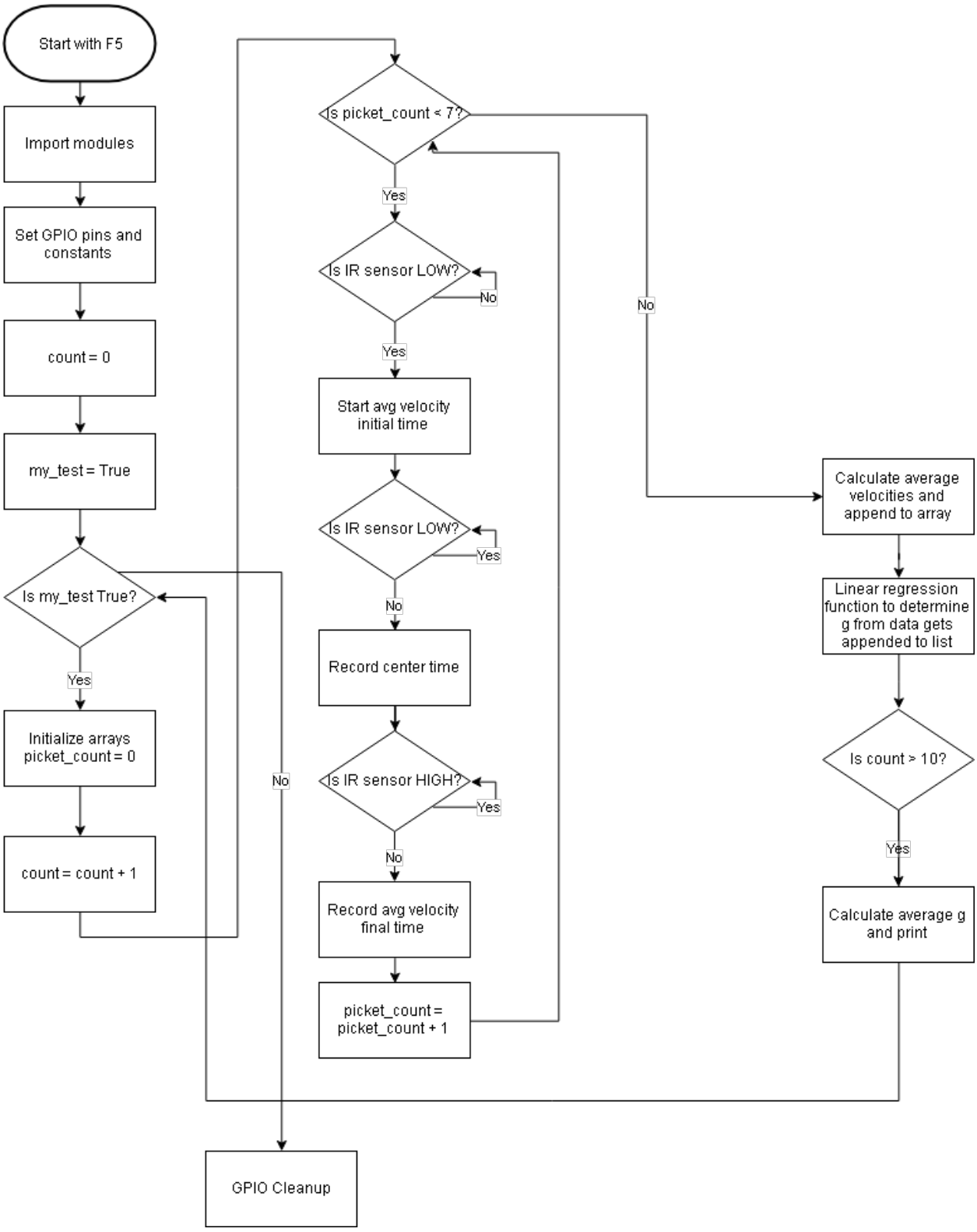

Figure 2: Flow of Python script.

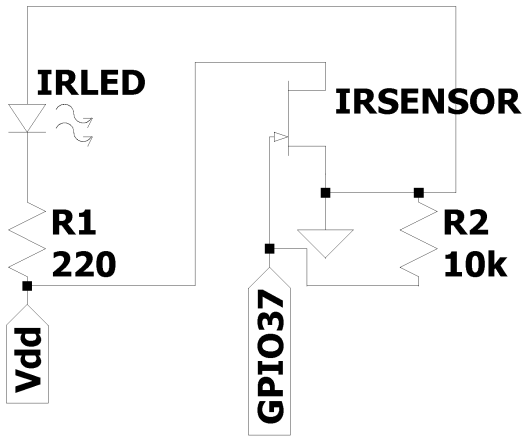

Figure 3: Schematic for setup.

3 Conclusion

This experiment was a relatively simple setup, which could now be expanded upon in order to obtain a better value for g. Future attempts at this experiment should consider the following changes:
1) A rig to drop the picket fence for you (to eliminate irregular drops)
2) Increase number of drops
3) A picket with more sections
4) Error analysis including air drag
This changes would allow a more accurate value for g.
Also, skills learned here can now be applied to other applications when interfacing with the Raspberry Pi.

4 Appendix

This is the main code.

```
import RPi.GPIO as GPIO
import time
import matplotlib.pyplot as plt
import numpy as np
import LinearRegressionClass as linreg

y = .050
IR = 37

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

g_values = []

try:
```

```
    my_test = True
    count = 0
    while my_test == True:
        count = count + 1
        plot_t = []
        avg_v = []
        del_t = []
        picket_count = 0
        IR_state = GPIO.input(IR)
        start_total = time.time()
        while picket_count < 7:
            IR_state = GPIO.input(IR)
            if IR_state == False:
                start_t = time.time()
                while IR_state == False:
                    IR_state = GPIO.input(IR)
                    pass
                plot_t.append(time.time() - start_total)
                IR_state = GPIO.input(IR)
                while IR_state == True:
                    IR_state = GPIO.input(IR)
                    pass
                del_t.append(time.time() - start_t)
                picket_count = picket_count + 1

        for i in range(0,7):
            v = (y) / del_t[i]
            avg_v.append(v)
        B = linreg.linear_regression(plot_t, avg_v)
        g_values.append(B)
        if count > 10:
            g_avg = np.mean(np.array(g_values))
            my_test = False
            print(g_avg)


except KeyboardInterrupt:
    print("that can't be good...")

finally:
    GPIO.cleanup()
    print('Isaac cleaned the oven.')
```

This is the import for the linear regression function.

```
# -*- coding: utf-8 -*-
"""
Created on Fri Mar 29 13:57:05 2019

@author: maxhu
"""
import numpy as np

def linear_regression(x_values, y_values):
##    y_values = []
##    x_values = []
#    lines = np.loadtxt('LOADINTEXTFILEHERE.txt', delimiter=',')
#    for line in lines:
#        x_values.append(line[0])
#        y_values.append(line[1])
    N = len(x_values)
    sum_y = 0
    sum_x = 0
    sum_xy = 0
    sum_x_squared = 0

    for i in range(0,N):
        sum_y = sum_y + y_values[i]
        sum_x = sum_x + x_values[i]
        sum_xy = sum_xy + x_values[i]*y_values[i]
        sum_x_squared = sum_x_squared + x_values[i]**2

    Delta = N*sum_x_squared - (sum_x)**2
    global A
    global B
    A = (sum_x_squared*sum_y - sum_x*sum_xy) / Delta
    B = (N*sum_xy - sum_x*sum_y) / Delta

    deviation = 0

    for i in range (0,N):
        deviation = deviation + (y_values[i] - A - B*x_values[i])**2

    sigma_y = np.sqrt(deviation / (N-2))

    sigma_A = sigma_y*np.sqrt(sum_x_squared/Delta)
    sigma_B = sigma_y*np.sqrt(N/Delta)

    print('The best estimate for B is: ', B)
    return B

def my_fit(x):#Plot this in your figure for linear regression line
    return A + B*x
```