Drone Propeller Thrust Analysis

PROJECT REPORT

DRZADINSKI, NATHAN

Table of Contents

Introduction

I am currently developing an open-source drone as a personal project. I am making all the avionics from scratch and currently planning to use an Arduino as the microcontroller. I already have radio communication working, gyroscopic sensor data, and an ultrasonic sensor for height. One of the main things I want in my drone is some autonomous features, like hovering, auto-landing, auto-takeoff, etc. I can get height and vertical velocity from the ultrasonic sensor. To control my drone as precisely as possible I need to know all the different thrust values at all PWM values. I am using 4 QWinOut A2212/6t 2200KV brushless DC motors with 9 in. propellers. The ESC I am using runs using the same signal as a servo motor. There are 180 different outputs for my Arduino to give the ESC. My goal for this project is to map out how the thrust from a single propeller is related to the Servo PWM signal from the Arduino. I am testing this using a 20Kg maximum load cell with a Hx711 module interpreter that is connected to an Arduino. This Arduino also controls the ESC. My code runs a test and analysis of such test data. The test runs through all 180 servo PWM signals and gets 10 force values from the load cell. This data is stored to a csv file. The analysis takes a csv file of data and creates a graph showing Force and PWM vs Time. This data should help me to code autonomous functions as precisely as possible.

Inputs and Outputs

Inputs

The user only needs to input 3 different things:

1. The user must use numeric integers to navigate menu
2. The user must input a filename without extension to save data/visual to
3. The user must input a csv filename without its extension to read data from for analysis task

Outputs

There are only two outputs:

1. Data is output to a file with user inputted filename.
2. Menu setup or “task complete”

User-Defined Functions

write_read(x)

Found in MotorTestSystem.py

1. write_read sends number ‘x’ as a string to the Arduino
2. Then it reads a force value from the Arduino
3. write_read returns force value from the Arduino

test(filename)

Found in MotorTestSystem.py

1. test calls send_data_to_csv with filename and labels for file data
2. Then it runs through 180 servo PWM signals ten times
    a. Each time test calls write_read writing servo PWM signal
    b. Gets force data back from write_read
    c. Checks if force data is flawed
    d. Calculates time
    e. Calls send_data_to_csv with current PWM value, force value, and time
3. Then calls write_read with 181 which resets an error value in the Arduino code
4. Calls write_read with 0 to turn off motor
5. Finally, test prints “Task Complete!”, there is nothing returned

send_data_to_csv(filename,input)

Found in recordData.py

1. send_data_to_csv checks if force data error occurred and if so skips to end of function
2. function opens csv file with filename with append type of interaction
3. function adds input to csv file.
    a. Input is a list of three data points: PWM, Force, and Time, in that order
4. send_data_to_csv returns nothing

linegraphg(infilename,outfilename)

Found in Analysis.py

1. open infilename csv file
2. make three lists, one for each data point, PWM, Force, and Time
    a. Skip label line (1st line)
    b. check data for force error (redundant)
    c. Force is in grams
3. convert three lists to numpy arrays
4. graph PWM vs Time and Force vs Time on same graph
    a. PWM vs time will be the straight line reminiscent of y=x

linegraphN(infilename,outfilename)

Found in Analysis.py

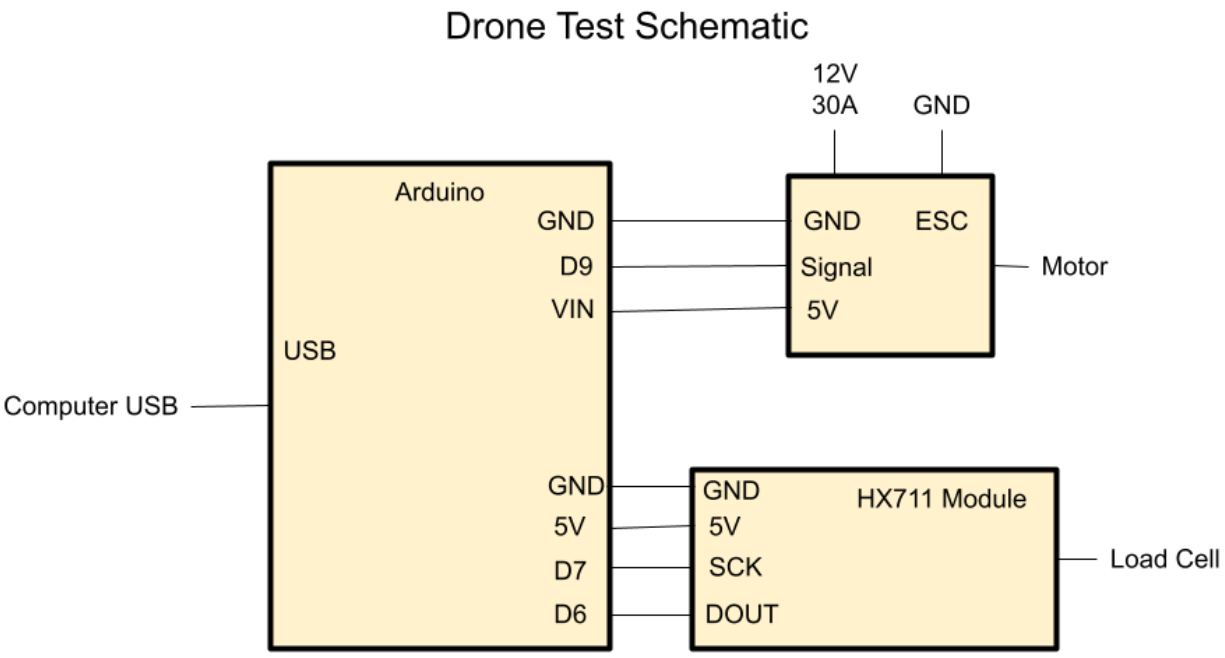
(Same as linegrapghg but force data is in Newtons)

1. open infilename csv file
2. make three lists, one for each data point, PWM, Force, and Time
    a. Skip label line (1st line)
    b. check data for force error (redundant)
    c. Force is converted from grams to Newtons
3. convert three lists to numpy arrays
4. graph PWM vs Time and Force vs Time on same graph
    a. PWM vs time will be the straight line reminiscent of y=x

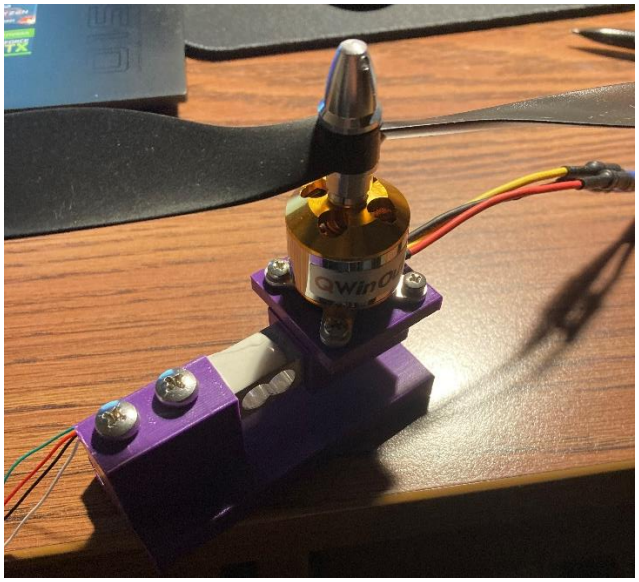
User Manual

1. Setup

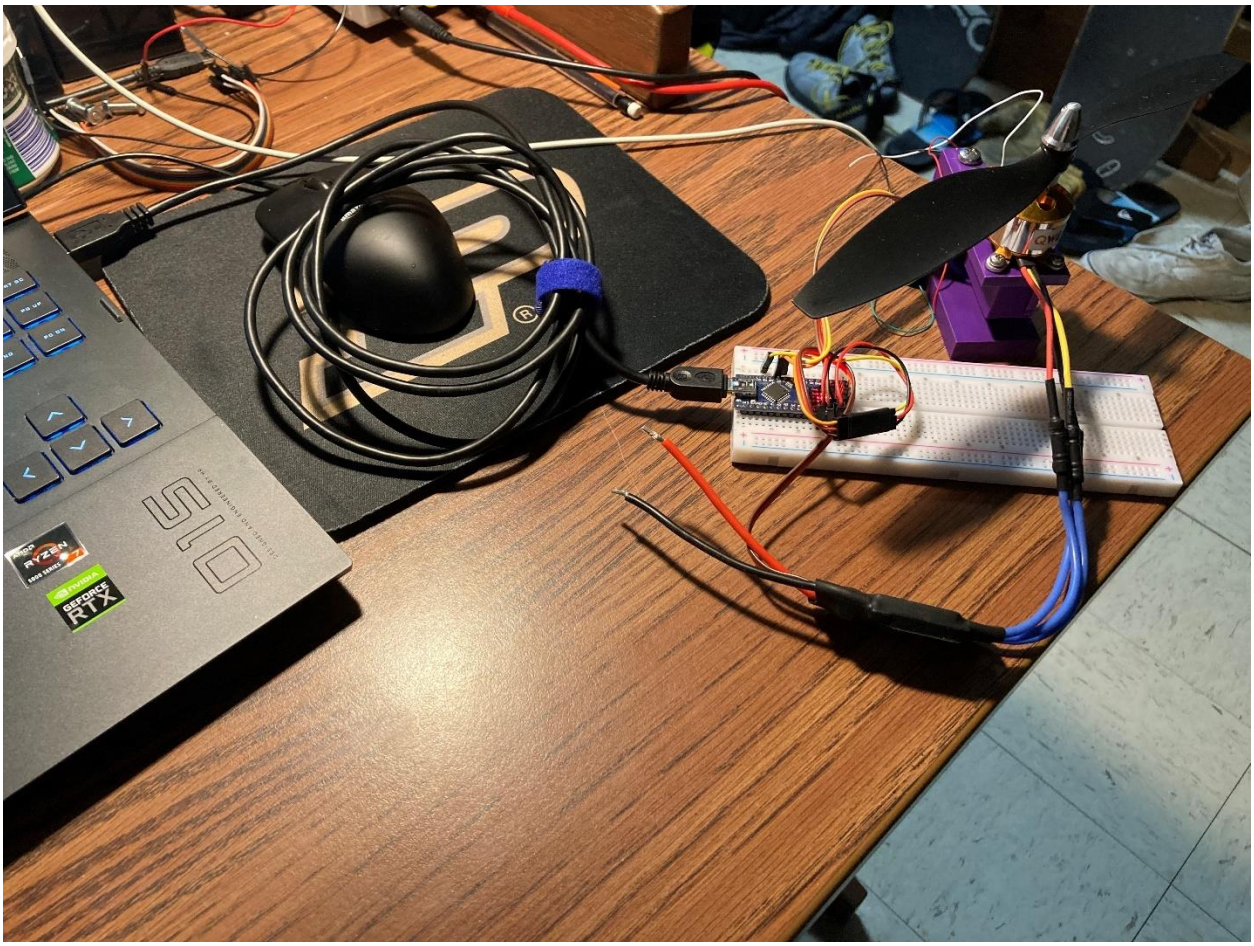
- Set up necessary circuit (See circuit diagram)



- Set up mechanical aspects
  - Connect motor to propeller upside down as to push air down (test this before full test!)
  - Set up load cell test stand



- Connect a USB cable from the Arduino to the COM3 port of your computer.
    - If no COM3 then you must edit the Arduino variable in the MotorTestSystem.py file. ‘COM3’ must be replaced with the name of the port the Arduino is attached to.
- The final hardware setup should look something like: ***ESC Power isn’t connected yet***



- Make sure all files, excluding the Arduino files, are in one folder
- Run Menu.py, menu like this should appear in the terminal:

```
Pick a task:
1) Test
2) Analyze Data
3) End Program
```

2. Run Test

- Select the Test function in main menu by inputting “1”

```
Pick a task:
1) Test
2) Analyze Data
3) End Program
1
```

- Input desired file name to save data in, **WITHOUT FILE EXTENSION**

```
Please input desired save file name without extension
test1
```

- When the test has been completed Task Complete! Will be displayed and then the main menu will come back up, (This could take a few minutes)

```
Test Complete!
```

3. Get Test Output

- You can find your data in a csv file with the filename given during Run Test

4. Run Analysis

- Select the Analyze Data option in main menu by inputting “2”

```
Pick a task:
1) Test
2) Analyze Data
3) End Program
2
```

- Choose between force being measured in grams or Newtons

```
Pick an analysis force unit:
1) Grams
2) Newtons
1
```

- Input desired csv file name that houses test data to analyze, **WITHOUT FILE EXTENSION**

```
Input input filename
10better
```

- Input desired file name to save graphical analysis in, **WITHOUT FILE EXTENSION**

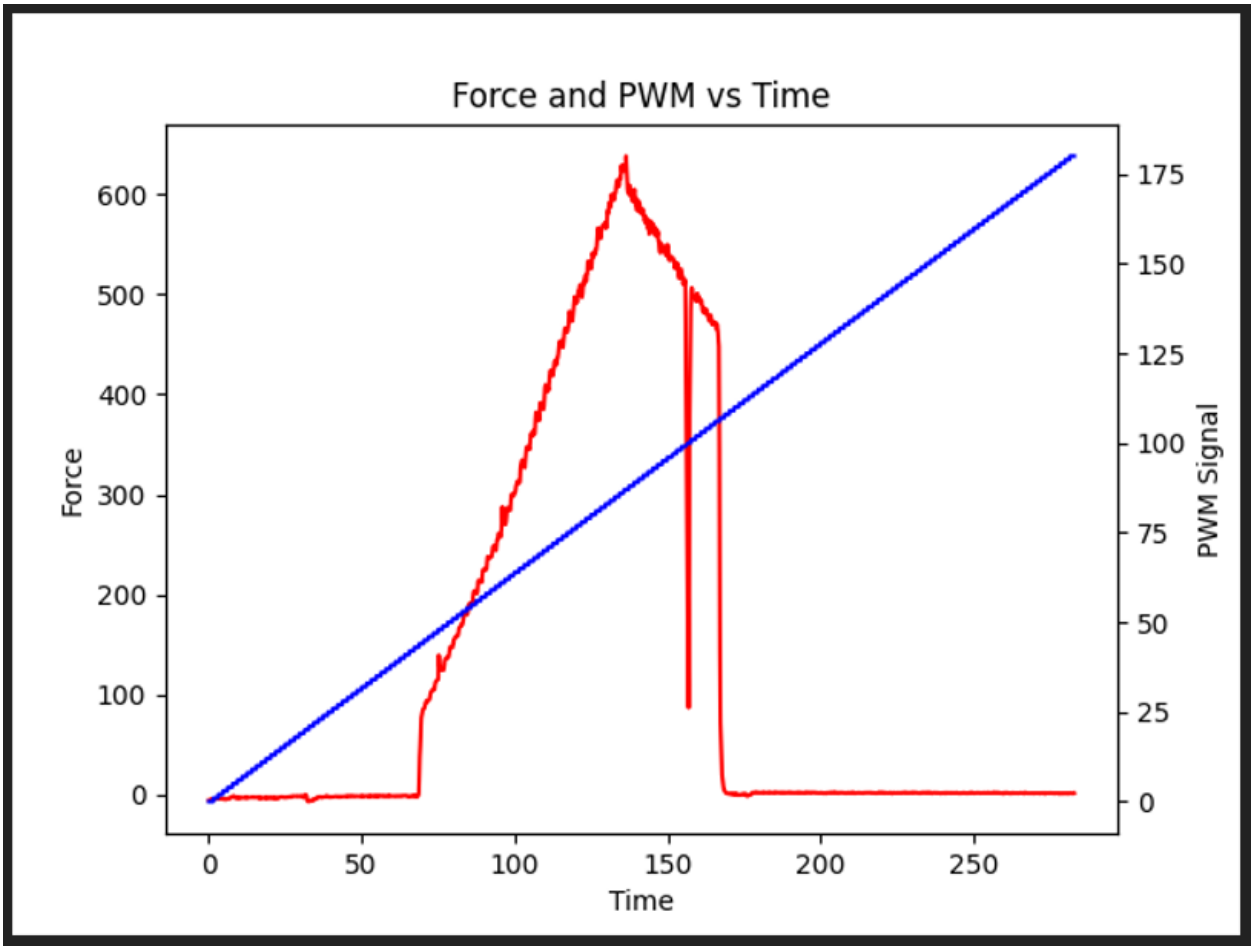
```
Input output filename
out
```

- When the test has been completed Task Complete! Will be displayed and then the main menu will come back up, (This could take a few minutes)

```
Analysis Completed!
```

5. Get Analysis Output

- You can find your graphical analysis png file with the filename given during Run Analysis
- Example Graph: (Red = Force, Blue = PWM)



6. Quit Program

- Select the Quit Program option in main menu by inputting “3”

```
Pick a task:
1) Test
2) Analyze Data
3) End Program
3
PS C:\Users\ndrza>
```

Appendix

Python Files

Menu.py

```
"""

================================================================================
ENGR 13300 Fall 2021

Program Description
    Test and analyze data on the strength of a propeller motor for open source
drone.

******* Important note:
    Code will not run unless Arduino plugged into Com 3. If you delete
MotorTestSystem import and call then can run analysis section

Assignment Information
    Assignment:     Individual Project
    Author:         Nathan Drzadinski, ndrzadin@purdue.edu
    Team ID:        LC3 - 26 (e.g. LC1 - 01; for section LC1, team 01)

Contributor:    Name, login@purdue [repeat for each]
    My contributor(s) helped me:
    [ ] understand the assignment expectations without
        telling me how they will approach it.
    [ ] understand different ways to think about a solution
        without helping me plan my solution.
    [ ] think through the meaning of a specific error or
        bug present in my code without looking at my code.
    Note that if you helped somebody else with their code, you
    have to list that person as a contributor here as well.

ACADEMIC INTEGRITY STATEMENT
I have not used source code obtained from any other unauthorized
source, either modified or unmodified. Neither have I provided
access to my code to another. The project I am submitting
is my own original work.
================================================================================
"""
import MotorTestSystem as MTS
import Analysis as A

def main():
```

```
    # Menu runs infinite until given command to end
    while True:
        # A basic Menu Structure
        in1 = input("Pick a task:\n1) Test\n2) Analyze Data\n3) End Program\n")
        if in1 == '1' or in1.lower == "test":
            in2 = input("Please input desired save file name without
extension\n")
            MTS.Test(in2)
        elif in1 == '2' or in1.lower == "analyze data":
            in3 = input("Pick an analysis force unit:\n1) Grams\n2) Newtons\n")
            if in3 == '1':
                inf = input("Input input filename\n")
                outf = input('Input output filename\n')
                A.linegraphg(inf,outf)
            elif in3 == '2':
                inf = input("Input input filename\n")
                outf = input('Input output filename\n')
                A.linegraphN(inf,outf)
        elif in1 == '3' or in1.lower == "end program":
            break
        else:
            print("Error: User input\n")

if __name__ == "__main__":
    main()
```

MotorTestSystem.py

```
"""
================================================================================
ENGR 13300 Fall 2021

Program Description
    Test and collect data on the strength of a propeller motor for open-source
drone.

Assignment Information
    Assignment:     Individual Project
    Author:         Nathan Drzadinski, ndrzadin@purdue.edu
    Team ID:        LC3 - 26 (e.g. LC1 - 01; for section LC1, team 01)

Contributor:    Name, login@purdue [repeat for each]
    My contributor(s) helped me:
    [ ] understand the assignment expectations without
        telling me how they will approach it.
    [ ] understand different ways to think about a solution
```

```
        without helping me plan my solution.
    [ ] think through the meaning of a specific error or
        bug present in my code without looking at my code.
    Note that if you helped somebody else with their code, you
    have to list that person as a contributor here as well.

ACADEMIC INTEGRITY STATEMENT
I have not used source code obtained from any other unauthorized
source, either modified or unmodified. Neither have I provided
access to my code to another. The project I am submitting
is my own original work.
================================================================================
"""
import serial
import time
from recordData import send_data_to_csv as sendcsv

# Declare arduino as a serial object
arduino = serial.Serial(port='COM3', baudrate=115200, timeout=.1)

def write_read(x):
    # Sends x to arduino
    arduino.write(bytes(x, 'utf-8'))
    time.sleep(0.05)
    # Reads data from arduino
    data = arduino.readline()
    return data

def Test(filename):
    # Label CSV file
    field = ['PWM','Force','Time']
    sendcsv(filename,field)
    # Find a start time so can see change over time
    start_time = time.time()
    # Iterates through most motor speeds
    for pwm in range(0,181):
        for i in range(0,9):
            # Looks for return data error
            try:
                fvalue = float(write_read(str(pwm)))
            except:
                fvalue = -10
            sendcsv(filename,[pwm,fvalue,time.time()-start_time])
    # Resets Arduino error variable
    write_read(str(181))
```

```
    # Turns Motor off at end of trial
    write_read(str(0))
    print("\n\nTest Complete!\n")

def main():
    pass

if __name__ == "__main__":
    main()
```

recordData.py

```
"""
================================================================================
ENGR 13300 Fall 2021

Program Description
    This file houses the function that writes the data to a csv file

Assignment Information
    Assignment:     Individual Project
    Author:         Nathan Drzadinski, ndrzadin@purdue.edu
    Team ID:        LC3 - 26 (e.g. LC1 - 01; for section LC1, team 01)

Contributor:    Name, login@purdue [repeat for each]
    My contributor(s) helped me:
    [ ] understand the assignment expectations without
        telling me how they will approach it.
    [ ] understand different ways to think about a solution
        without helping me plan my solution.
    [ ] think through the meaning of a specific error or
        bug present in my code without looking at my code.
    Note that if you helped somebody else with their code, you
    have to list that person as a contributor here as well.

ACADEMIC INTEGRITY STATEMENT
I have not used source code obtained from any other unauthorized
source, either modified or unmodified. Neither have I provided
access to my code to another. The project I am submitting
is my own original work.
================================================================================
"""
import csv

# Append relevent data to specificized csv file
def send_data_to_csv(filename, input):
```

```
    # This stops a data error from being added to the csv file
    if input[1] != '-10':
        with open(filename+".csv",'a') as csvfile:
            writer = csv.writer(csvfile,lineterminator = '\n')
            writer.writerow(input)

def main():
    pass

if __name__ == "__main__":
    main()
```

Analysis.py

```
"""
================================================================================
ENGR 13300 Fall 2021

Program Description
    This file houses all the functions that do different data analysis

Assignment Information
    Assignment:     Individual Project
    Author:         Nathan Drzadinski, ndrzadin@purdue.edu
    Team ID:        LC3 - 26 (e.g. LC1 - 01; for section LC1, team 01)

Contributor:    Name, login@purdue [repeat for each]
    My contributor(s) helped me:
    [ ] understand the assignment expectations without
        telling me how they will approach it.
    [ ] understand different ways to think about a solution
        without helping me plan my solution.
    [ ] think through the meaning of a specific error or
        bug present in my code without looking at my code.
    Note that if you helped somebody else with their code, you
    have to list that person as a contributor here as well.

ACADEMIC INTEGRITY STATEMENT
I have not used source code obtained from any other unauthorized
source, either modified or unmodified. Neither have I provided
access to my code to another. The project I am submitting
is my own original work.
================================================================================
"""
import csv
import numpy as np
```

```
import matplotlib.pyplot as plt

def linegraphg(infilename,outfilename):
    # Open csv file
    with open(infilename+'.csv') as csvfile:
        reader = csv.reader(csvfile)
        # Declare lists
        PWMlist = []
        Forcelist = []
        Timelist = []
        # Count removes csv header
        count =0
        for row in reader:
            if count != 0:
                # Removes error data (Redundant)
                if row[1] == '-10':
                    continue
                else:
                    PWMlist.append(float(row[0]))
                    Forcelist.append(float(row[1]))
                    Timelist.append(float(row[2]))
            count+=1
        # Make Lists into arrays for graphing
        Timelist = np.array(Timelist,dtype=object)
        Forcelist = np.array(Forcelist,dtype=object)
        PWMlist = np.array(PWMlist,dtype=object)
        # Graph data and save to input filename
        fig, axis = plt.subplots()
        axis.plot(Timelist,Forcelist, color = "red")
        axis.set_xlabel("Time")
        axis.set_ylabel('Force (grams)')
        axis2 = axis.twinx()
        axis2.plot(Timelist,PWMlist,color="blue")
        axis2.set_ylabel('PWM Signal')
        plt.title("Force and PWM vs Time")
        plt.savefig(outfilename+'.png')
        print('\n\nAnalysis Completed!\n')

# Same as grams but units converted to Newtons
def linegraphN(infilename,outfilename):
    with open(infilename+'.csv') as csvfile:
        reader = csv.reader(csvfile)
        PWMlist = []
        Forcelist = []
        Timelist = []
```

```
        count =0
        for row in reader:
            if count != 0:
                if row[1] == '-10':
                    continue
                else:
                    PWMlist.append(float(row[0]))
                    Forcelist.append(float(row[1])*0.0098) # unit conversion
                    Timelist.append(float(row[2]))
            count+=1
        Timelist = np.array(Timelist,dtype=object)
        Forcelist = np.array(Forcelist,dtype=object)
        PWMlist = np.array(PWMlist,dtype=object)
        fig, axis = plt.subplots()
        axis.plot(Timelist,Forcelist, color = "red")
        axis.set_xlabel("Time")
        axis.set_ylabel('Force (Newtons)')
        axis2 = axis.twinx()
        axis2.plot(Timelist,PWMlist,color="blue")
        axis2.set_ylabel('PWM Signal')
        plt.title("Force and PWM vs Time")
        plt.savefig(outfilename+'.png')
        print('\n\nAnalysis Completed!\n')
```

Arduino Code

MotorTestStandCode.ino

```
/*
 * Nathan Drzadinski
 * 10/26/2021
 * Motor Test Stand
 *
 */

#include <HX711.h>
#include <Servo.h>

HX711 scale;

Servo ESC;

// HX711 circuit wiring
const int LOADCELL_DOUT_PIN = 6;
const int LOADCELL_SCK_PIN = 7;
```

```
int error = 0; // Equals PWM jumps
int count = 0; // Equals num of trials at each speed
int x;
float calibration_factor = 111;
float units;

void setup() {
  Serial.begin(115200);
  Serial.setTimeout(1);
  ESC.attach(9);
  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  ESC.write(0);
  scale.set_scale(calibration_factor); //Adjust to this calibration factor
  scale.tare();  //Reset the scale to 0
}

void loop() {
  //Reads incoming Int and sets motor speed to correspond.
  while (!Serial.available());
  x = Serial.readString().toInt();
  if (((x>=error)&(x<181))||x==0){
    ESC.write(x);
  }
  count++;
  units = scale.get_units(), 5;

  Serial.print(units);

  //Stops occasional bit error
  if (count>10) {
    count = 0;
    error+=1; // Up by ten, must change to python step
  }
  //resets error at end of python code
  if (error > 180){
    error =0;
  }
}
```