```
# -*- coding: utf-8 -*-
"""
Created on Mon Jan  6 12:31:42 2020

@author: Wes Newton
"""
import nidaqmx
import matplotlib.pyplot as plt
import time
import numpy as np
import time
from tkinter import *
from tkinter import ttk
import csv

class Digital_Input():
    def __init__(self, channel):
        self.SW = nidaqmx.Task()
        resource = 'cDAQ1Mod5/line' + str(channel) + ':' + str(channel)
        self.SW.di_channels.add_di_chan(resource)

    def Read_SW(self):
        value = self.SW.read()
        return value

    def Free(self):
        self.SW.close()

class Analog_Output():
    def __init__(self, channel):
        self.channel = nidaqmx.Task()
        self.channel.ao_channels.add_ao_voltage_chan('cDAQ1Mod8/ao0')

    def output(self, voltage):
        self.channel.write(voltage)

    def cleanup(self):
        self.channel.write(0.0)
        self.channel.close()

class Motor_Control:
    def __init__(self, parent):
        # front panel
        self.myParent = parent
        self.control_panel = ttk.Frame(parent)
        self.control_panel.pack()

        self.btn_start = Button(self.control_panel)
        self.btn_start.configure(text="Start", background = "green")
        self.btn_start.bind("<Button-1>", self.start)
        self.btn_start.grid(row = 1, column = 1, pady = 4, padx = 4)

        self.btn_exit = Button(self.control_panel)
        self.btn_exit.configure(text='Exit', background = "blue")
        self.btn_exit.bind("<Button-1>", self.exit)
        self.btn_exit.grid(row = 2, column = 1, pady = 5)
```

```
        self.lbl_status = Label(self.control_panel)
        self.lbl_status.configure(text = "Waiting ...")
        self.lbl_status.grid(row = 4, column = 1, pady = 5)

        self.lbl_tach = Label(self.control_panel)
        self.lbl_tach.configure(text = "Speed (Counts)")
        self.lbl_tach.grid(row = 5, column = 1, pady = 5)

        self.lbl_tach_sp = Label(self.control_panel)
        self.lbl_tach_sp.configure(text = "Set Point(counts")
        self.lbl_tach_sp.grid(row = 6, column = 1, pady = 5)

        self.lbl_volts = Label(self.control_panel)
        self.lbl_volts.configure(text = "volts")
        self.lbl_volts.grid(row = 7, column = 1, pady = 5)

        #Digital i/o and analog output
        self.Stop_SW = Digital_Input(1)
        self.Start_SW = Digital_Input(0)
        self.Motor = Analog_Output(0)
        self.Up_spd = Digital_Input(2)
        self.Dn_spd = Digital_Input(3)

        self.Tach_In = nidaqmx.Task() #configure here to use all methonds
        self.Tach_In.di_channels.add_di_chan('cDAQ1Mod5/line5:5')

    def start(self, event):
        voltage = 0.00
        running = True
        set_point = 25
        up = False
        down = False

        self.lbl_tach_sp.configure(text = str(set_point))
        self.lbl_status.configure(text='Start..')
        self.control_panel.update_idletasks()
        self.Motor.output(voltage)

        Kp = .2  # Proportional term
        Kd = 0.0  # Derivative Term
        Ki = .02  # Integral Term
        error1_prev = 0
        error1_sum = 0

        self.P= []   # for plotting later
        self.I =[]
        self.D = []
        self.PID = []
        self.SP = []
        self.CT = []

        while running:
            self.Motor.output(voltage)
            Pulse = 0
            Count = self.Tach_In.read(200)
```

```
            for i in range(0, 199):
                if Count[i]==True and Count[i+1] != True:
                    Pulse +=1

            # PID control
            error1 = set_point - Pulse
            p = error1 * Kp
            i = error1_sum * Ki
            d = error1_prev * Kd
            PID = p + i + d
            error1_prev = error1
            error1_sum += error1

            self.P.append(p)
            self.I.append(i)
            self.D.append(d)
            self.PID.append(PID)
            self.SP.append(set_point)
            self.CT.append(Pulse)

            voltage = max(min(10.0, PID), 00.0)
            stop = self.Stop_SW.Read_SW()
            root.update_idletasks()
            if stop == True:
                running = False
                self.Motor.output(0.0)
                self.write_file()
            up_prev = up
            up = self.Up_spd.Read_SW()
            if up != up_prev and up == True:
                set_point += 1
            dn_prev = down
            down = self.Dn_spd.Read_SW()
            if down != dn_prev and down == True:
                set_point -= 1

            self.lbl_volts.configure(text = str(voltage))
            self.lbl_tach.configure(text=str(Pulse))
            self.lbl_tach_sp.configure(text = str(set_point))
            root.update_idletasks()

    def stop(self, event):
        self.lbl_status.configure(text='Stop..')
        self.Motor.output(0.0)

    def exit(self, event):
        self.lbl_status.configure(text='Exiting...')
        self.Motor.output(0.0)
        self.Motor.cleanup()
        self.Stop_SW.Free()
        self.Start_SW.Free()
        self.Tach_In.close()
        self.myParent.destroy()

    def write_file(self):
        with open('pid_data.csv', 'w', newline ='') as csvfile:
```

```
            writer = csv.writer(csvfile, delimiter = ',')
            for i in range(0, len(self.P)):
                writer.writerow([self.P[i], self.I[i], self.D[i], self.PID[i],
                                 self.SP[i], self.CT[i]])


root = Tk()
root.geometry("300x300")
motor = Motor_Control(root)
root.mainloop()
```

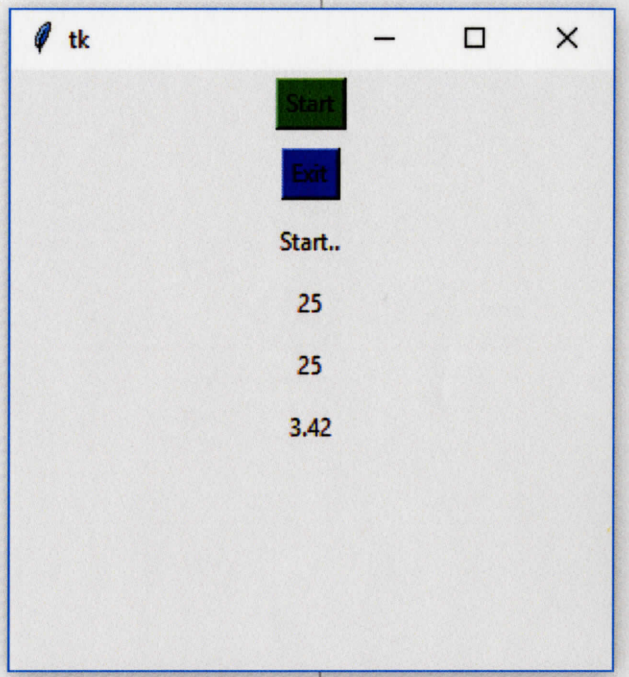
Spyder (Python 3.7)

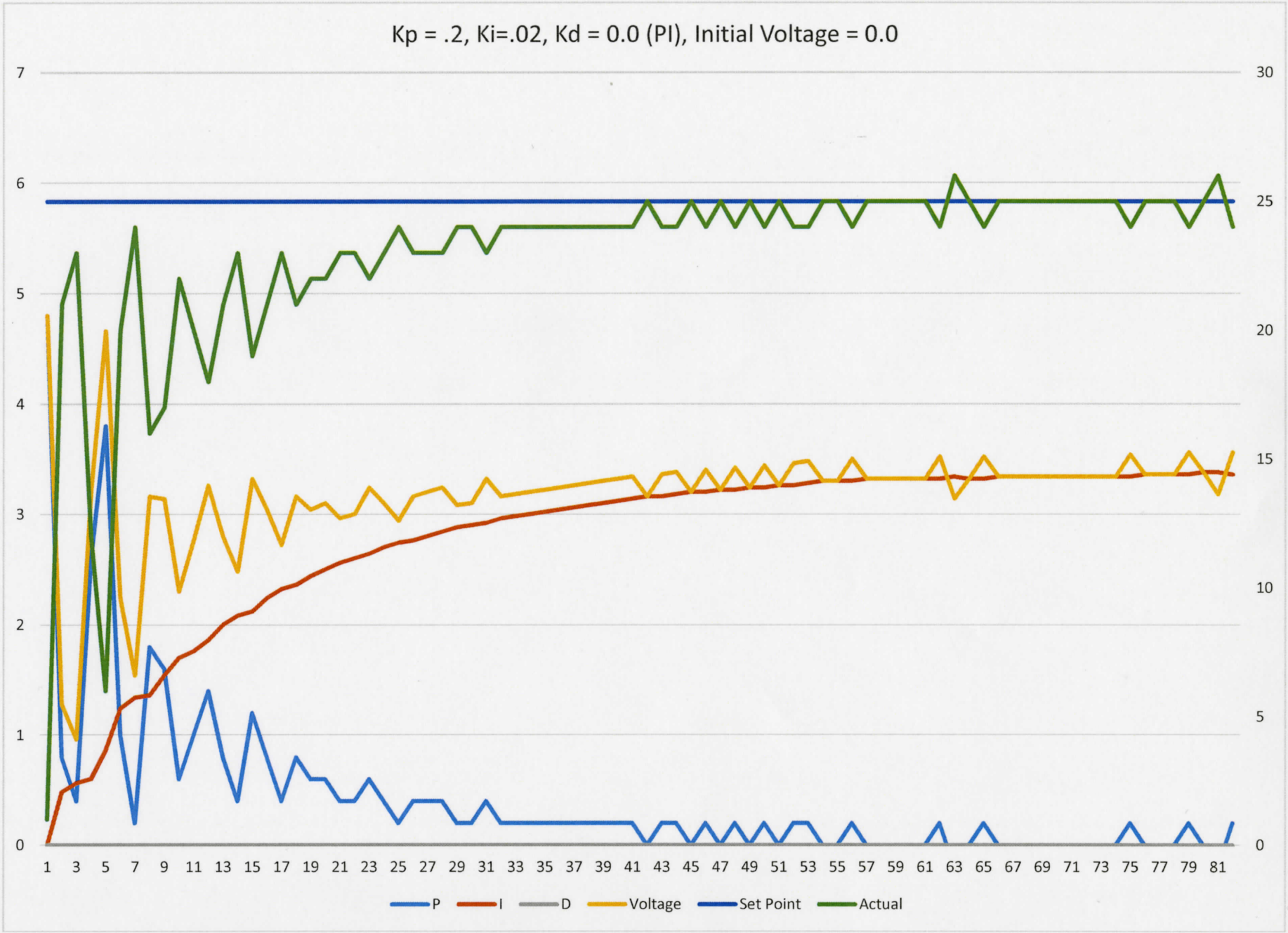
File Edit Search Source Run Debug Consoles Projects Tools View Help

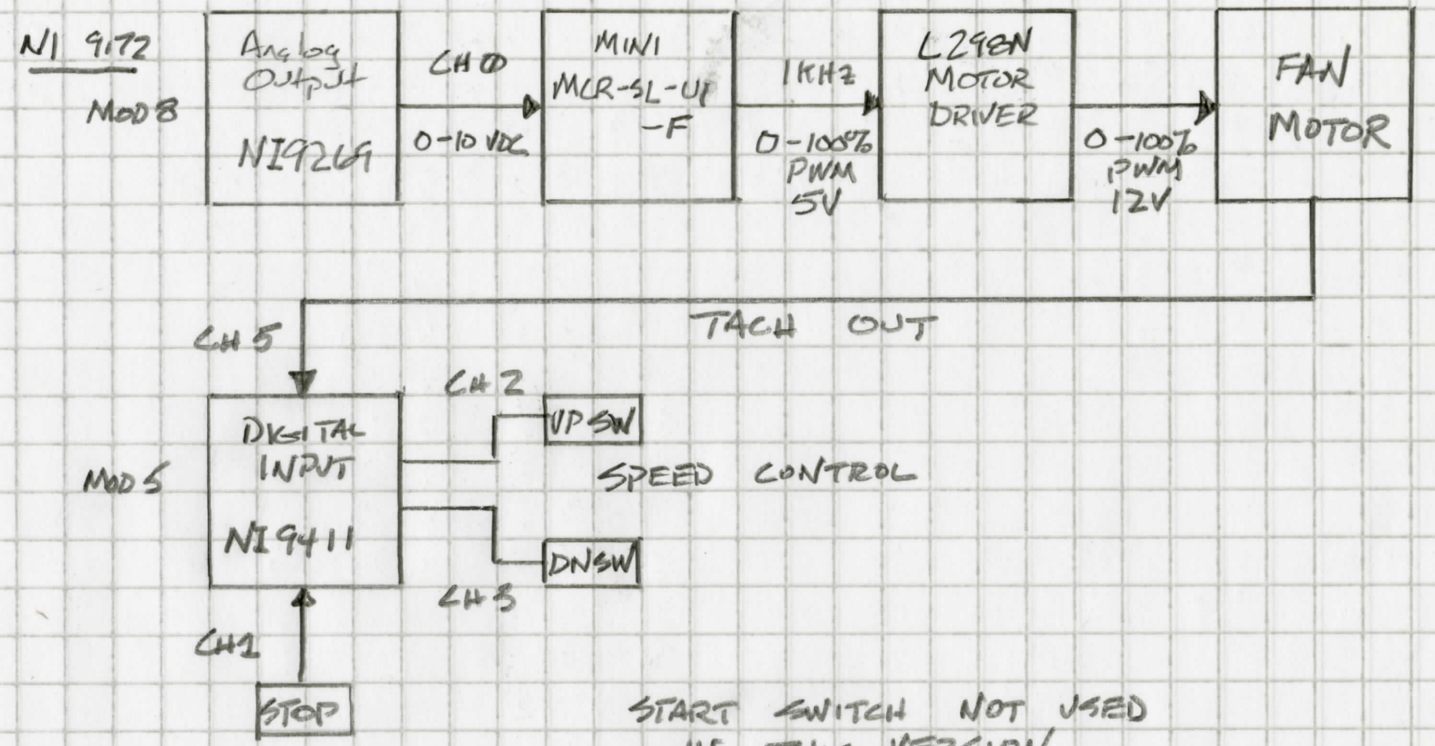
Editor - C:\Users\220000177\Desktop\Python37\Fan_Motor_Speed_Control.py

Feet_To_Meters.py | scratch_6.py | Fan_Motor_Speed_Control.py

```
112             Count = self.Tach_In.read(200)
113             for i in range(0, 199):
114                 if Count[i]==True and Count[i+1] != True:
115                     Pulse +=1
116
117             # PID control
118             error1 = set_point - Pulse
119             p = error1 * Kp
120             i = error1_sum * Ki
121             d = error1_prev * Kd
122             PID = p + i + d
123             error1_prev = error1
124             error1_sum += error1
125
126             self.P.append(p)
127             self.I.append(i)
128             self.D.append(d)
129             self.PID.append(PID)
130             self.SP.append(set_point)
131             self.CT.append(Pulse)
132
133             voltage = max(min(10.0, PID), 00.0)
134             stop = self.Stop_SW.Read_SW()
135             root.update_idletasks()
136             if stop == True:
137                 running = False
138                 self.Motor.output(0.0)
139                 self.write_file()
140             up_prev = up
141             up = self.Up_spd.Read_SW()
142             if up != up_prev and up == True:
143                 set_point += 1
144             dn_prev = down
145             down = self.Dn_spd.Read_SW()
146             if down != dn_prev and down == True:
```




Kp = .2, Ki=.02, Kd = 0.0 (PI), Initial Voltage = 0.0
7
6
5
4
3
2
1
0
30
25
20
15
10
5
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81
P
I
D
Voltage
Set Point
Actual


NI 9172
MOD 8
Analog Output
NI9264
CH 0
0-10 VDC
MINI MCR-SL-UI -F
1KHz
0-100% PWM 5V
L298N MOTOR DRIVER
0-100% PWM 12V
FAN MOTOR
TACH OUT
CH 5
CH 2
DIGITAL INPUT
NI9411
MOD 5
UP SW
SPEED CONTROL
DN SW
CH 3
CH 1
STOP
START SWITCH NOT USED IN THIS VERSION
START IS PERFORMED ON GUI