Measurement of Rotation Rate of a Small Motor using a Raspberry Pi

The object of this example is to measure the rotation rate of an electric motor utilising photodiodes as a light source and the photo resistor as a sensor. A MCP3800 multichannel analogue - digital converter is used for voltage measurements down to 10µs intervals. The control of the motor is performed using a ld293 chip.

Equipment
. Raspberry Pi 3
. LD293 motor control IC
. DC hobby motor size 130 with fan
‘ KLS6-3537 3mm CdS photosensitive resistor
. MCP3008 8-channel ADC IC
. 3V white led array +holder
. 2x breadboard
. breadboard supply +motor voltage supply
‘ 1k ohm resistor
. micro button
. 20 Dupont male to female wires
‘ 17 breadboard wires

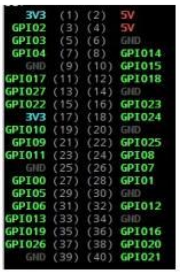
Elegoo.com provide excellent starter kits for Arduino and Raspberry Pi a selection can be found at https://www.elegoo.com/collections/arduino-learning-sets

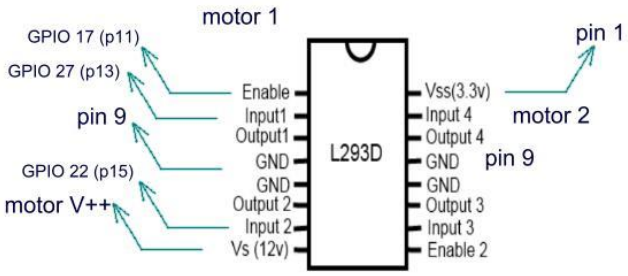
Setup of motor and ld293
DC Motor Control With Raspberry Pi and L293D.

The ld293 can be controlled using GPIO pins on the Raspberry Pi, it has the ability to control the speed and direction of 2 motors although in this case we only need to control 1. The Circuit Python pulseio library can be used to create pulse trains of specific frequency and duty cycles that are important for our use.

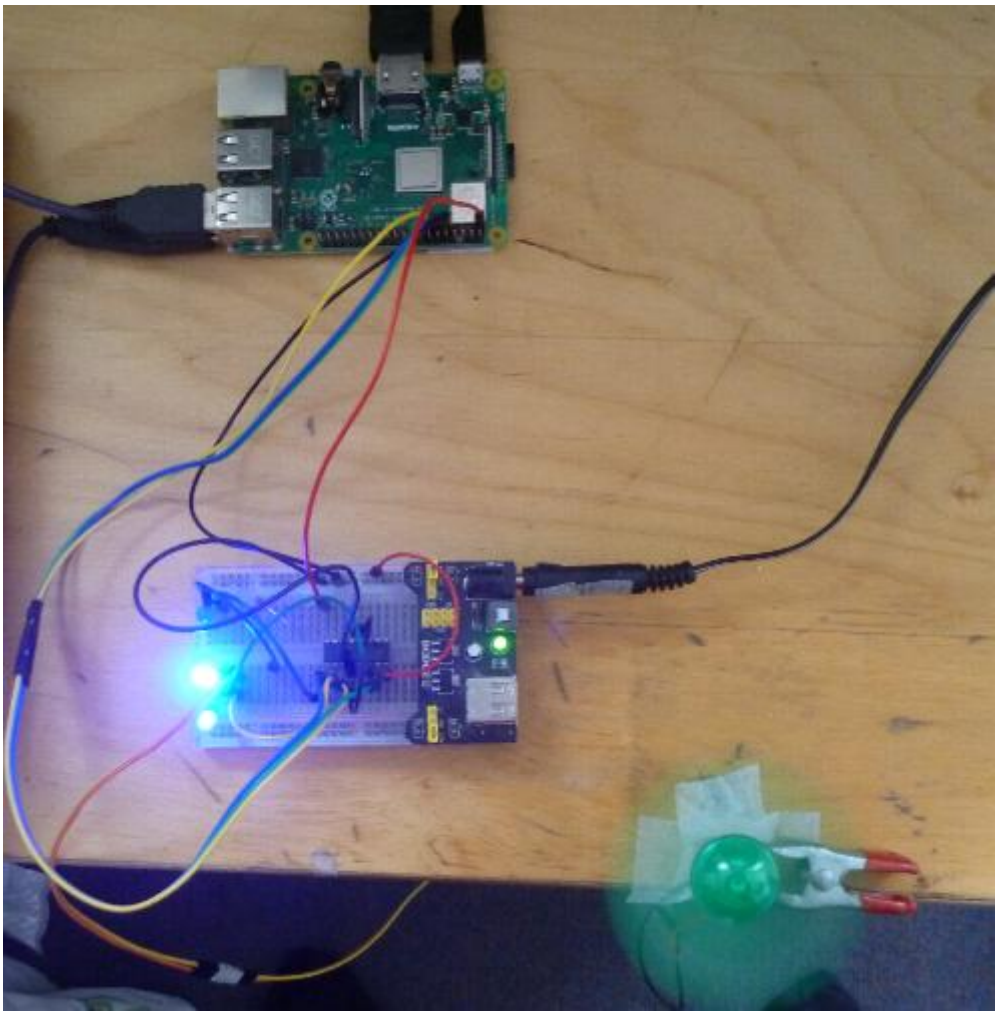
raspberry pi

GPIO pins





Above shows details of the connections between the ld293 chip and the Raspberry Pi. In this arrangement we expect to determine the direction of rotation of the motor using digital signals on GPIO 27 and GPIO 22 and we use GPIO 17 with a pulse modulated signal to control the speed with the duty cycle. The duty cycle is the proportion of time the voltage is at maximum and when averaged over time it is an average voltage output. The maximum value for duty cycle is 65535, giving maximum voltage and the minimum duty cycle is 0, resulting in a 0 volt output..



We can demonstrate this by running the following code, which you can copy and paste as motorcontrol.py

---

```
import time
import board
import pulseio
import digitalio

from digitalio import DigitalInOut, Direction
```

```
in1_pin = digitalio.DigitalInOut(board.D27)
in2_pin = digitalio.DigitalInOut(board.D22)

in1_pin.direction = Direction.OUTPUT
in2_pin.direction = Direction.OUTPUT

#forward
in1_pin.value = True
in2_pin.value = False
#reverse
#in1_pin.value = False
#in2_pin.value = True

# Initialize PWM output for the servo (on pin D17):
enabl = pulseio.PWMOut(board.D17, duty_cycle =0,frequency =1500,variable_frequency=False)


while True:
    enabl.duty_cycle =25000             # 0 <= duty cycle  <=65535
```

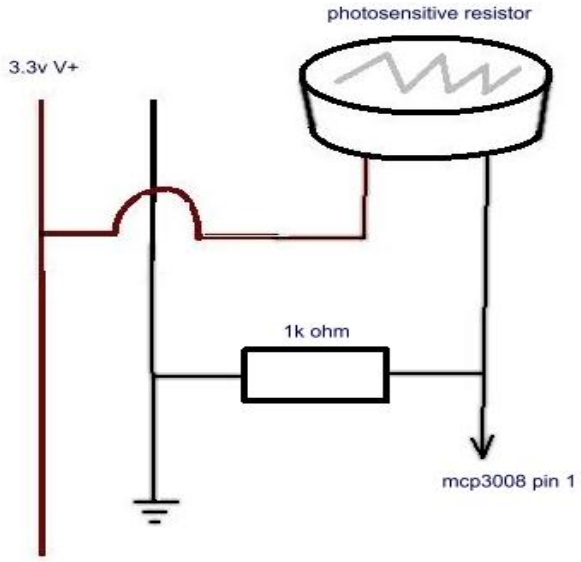
To reverse the the direction of motion comment out the forward in1_pin.value, in2_pin.value and un-comment reverse in1_pin.value, in2_pin.value.

To vary the motor speed change the value of enabl.duty_cycle within the while True loop.

Ref.
https://www.instructables.com/DC-Motor-Control-With-Raspberry-Pi-and-L293D/

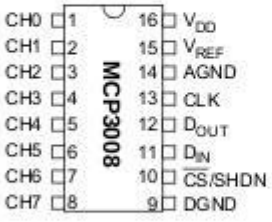
Setup and demonstration of photo resistor

We may use a CdS photo sensitive resistor to measure differences in light levels. It is a cheap high



sensitivity quick response sensor that is up to the task. We may use it in the form of a bridge with a 1k resistor, measurement of the voltage at the junction relates to the light on the sensor.
The MCp30008 chip is a multichannel ADC that can read up to 8 channels output and has 10bit resolution per channel, it uses SPI protocol for serial data transfer at rates up to 200ksps. The SPI bus requires 4 wires CLK, MOSI, MISO, CS. There is a single master signal initiator and there may be one or more slave devices feeding back data to the master. There are different configurations for multiple slaves but only one slave is active at any instant.

In our configuration the MCP3008 is the single slave and the connections are shown below.

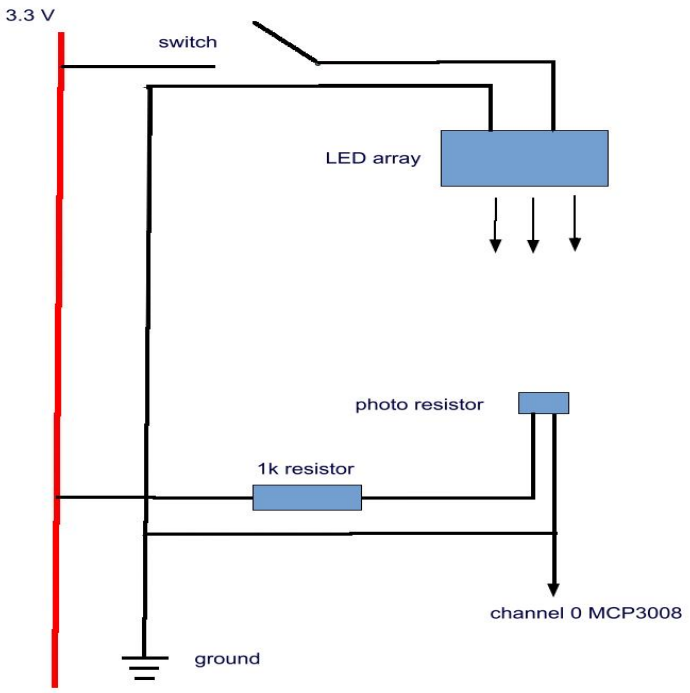


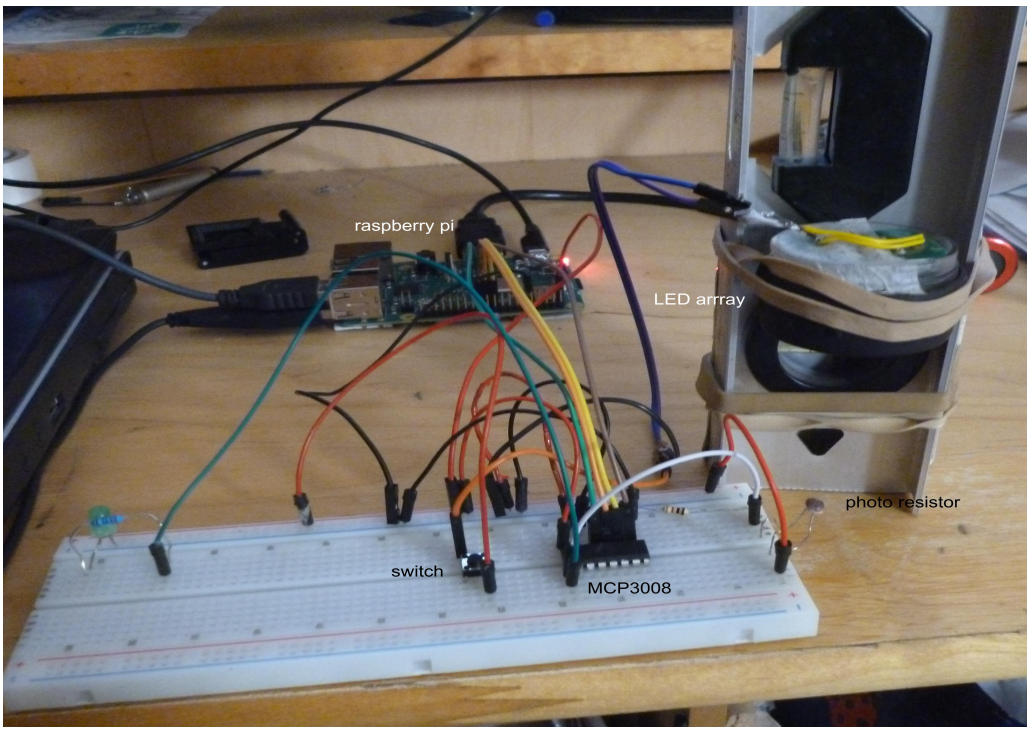
- **MCP3008 CLK** to **Pi SCLK** GPIO 11
- **MCP3008 DOUT** to **Pi MISO** GPIO 09
- **MCP3008 DIN** to **Pi MOSI** GPIO 10
- **MCP3008 CS** to CE0 GPIO 08
- **MCP3008 VDD** to **Pi 3.3V**
- **MCP3008 VREF** to **Pi 3.3V**
- **MCP3008 AGND** to **Pi GND**
- **MCP3008 DGND** to **Pi GND**
- **MCP3008 CH0** to low voltage side of photoresistor

The sofware requires us to load the Adafruit CircuitPython MCP3xxx module. The instructions for this is found at https://github.com/adafruit/Adafruit_CircuitPython_MCP3xxx.

Further reading about MCP3008 can be found in the document provided by Adafruit “mcp3008-spi-adc.pdf” and the technical specification document “MCP3008.pdf” from Microchip Technology Inc.

We must now provide an electronically controllable light source. For this we can use a 3.5v white light led array that can be obtained from a cheap torch. If this is setup with a switch in its supply it will suffice.





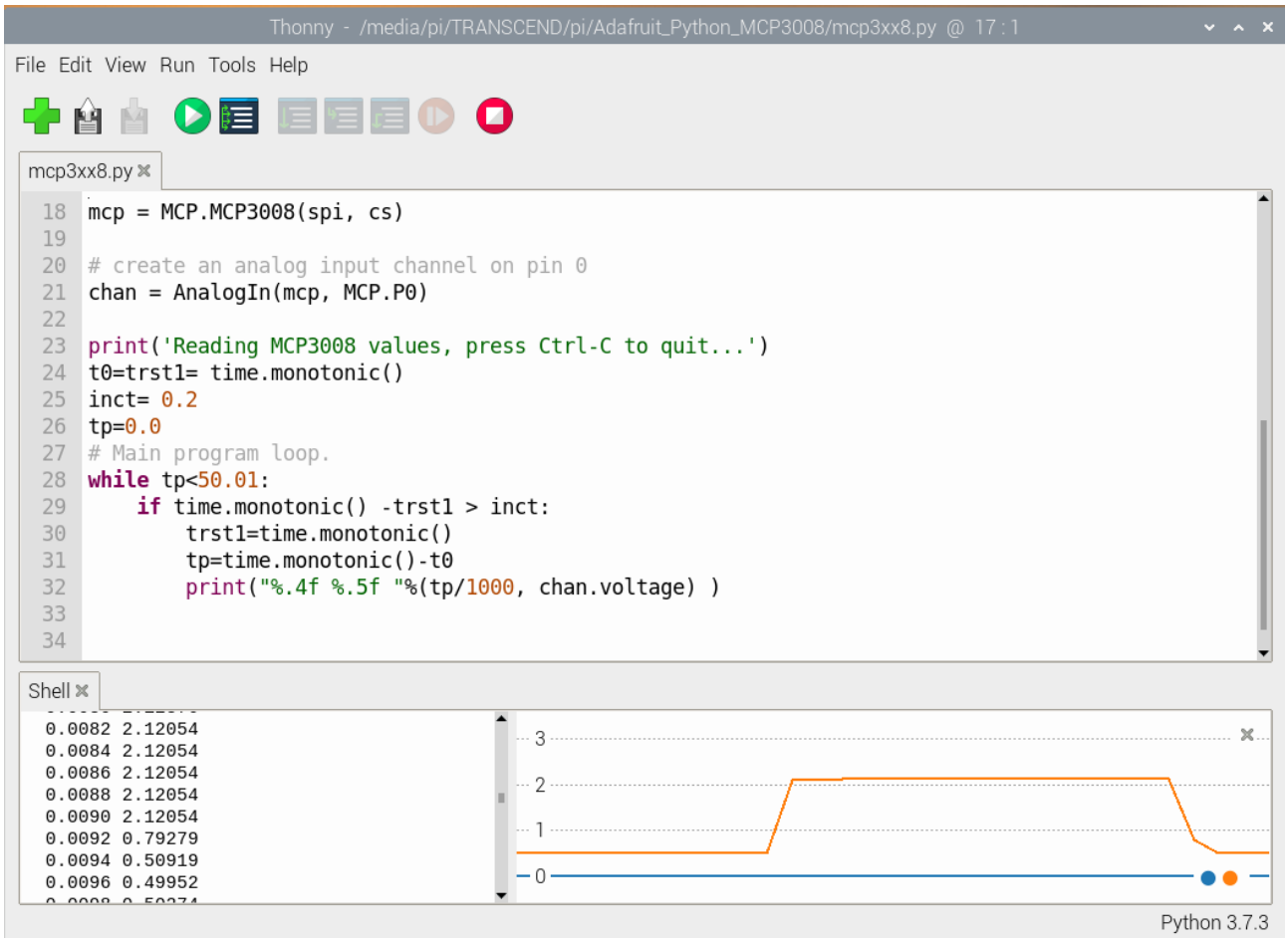
Software.

Copy the following and run it as MCP3xxx8.py.

```
# Simple example of reading the MCP3008 analog input channels
# Author: Tony DiCola
# revised Tunde Adeyemo 11/20
# License: Public Domain
import time
import busio
import digitalio
import board
import adafruit_mcp3xxx.mcp3008 as MCP
from adafruit_mcp3xxx.analog_in import AnalogIn
# create the spi bus
spi = busio.SPI(clock=board.SCK, MISO=board.MISO, MOSI=board.MOSI)

# create the cs (chip select)
cs = digitalio.DigitalInOut(board.D8)

# create the mcp object
mcp = MCP.MCP3008(spi, cs)

# create an analog input channel on pin 0
```

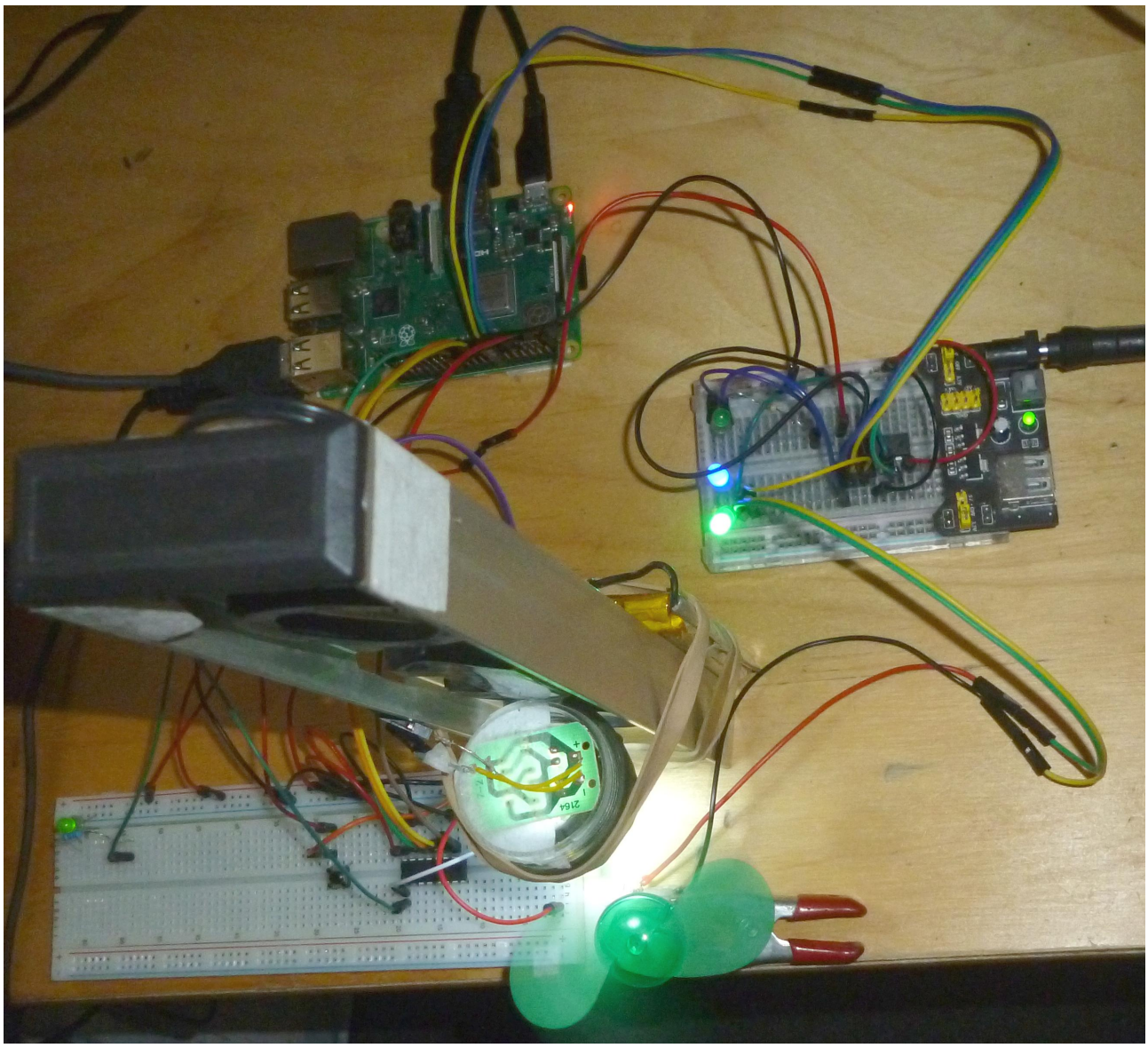
```
chan = AnalogIn(mcp, MCP.P0)

print('Reading MCP3008 values, press Ctrl-C to quit...')
t0=trst1= time.monotonic()
inct= 0.2
tp=0.0
# Main program loop.
while tp<50.01:
    if time.monotonic() -trst1 > inct:
        trst1=time.monotonic()
        tp=time.monotonic()-t0
        print("%.4f %.5f "%(tp/1000, chan.voltage) )
```

Below is the screen shot of an example of running it in Thonny while pressing the switch.



We are now ready to put it all together to measure the motor speed.

Connect both circuits to the Raspberry Pi, ensuring the ground and 3.3v power sources are interconnected. Place the motor such that its fan shades the photo resistor when in rotation and short-circuit the switch to set the the LED array to permanently on.



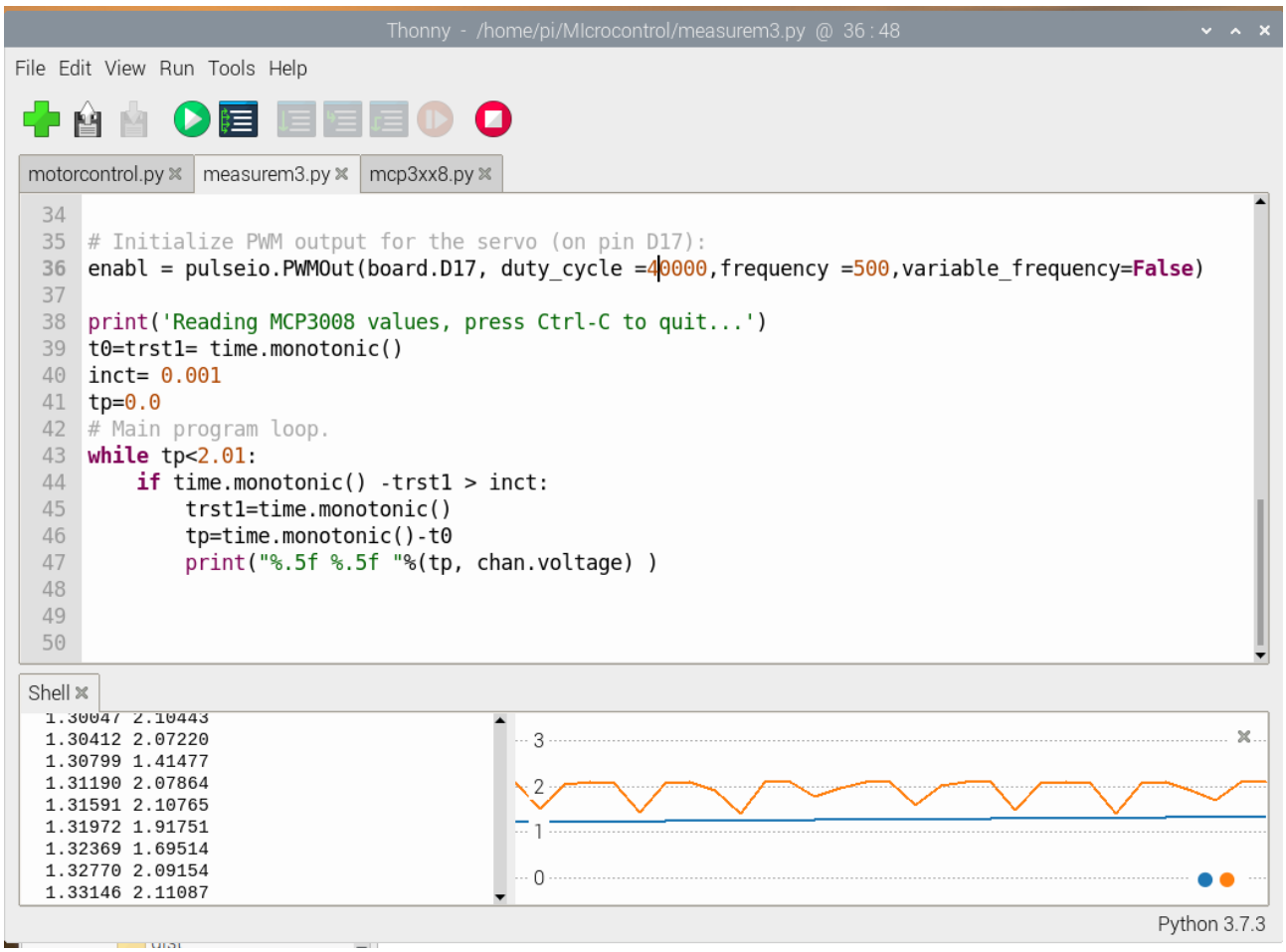
We now meld our two programs.

There are a few modifications to help it work in a fashion that would make it easier for us to get our results.

The main program loop is allowed to run for just over 2s, this is to prevent an endless sream of printed results particularly since we reduce the intervals between reads to 1 millesecond, also it prevents the scale of the graph printed in Thonny from getting out hand. The following program can be copied and saved as maeasureem3.py and then run.

```
# measurem3.py ; similtaneosly control motor and measure rotation speed
# author Tunde Adeyemo 01/21
# License: Public Domain
import time
import board
```

```
import pulseio
import digitalio
import busio
import adafruit_mcp3xxx.mcp3008 as MCP
from adafruit_mcp3xxx.analog_in import AnalogIn
# create the spi bus
spi = busio.SPI(clock=board.SCK, MISO=board.MISO, MOSI=board.MOSI)

# create the cs (chip select)
cs = digitalio.DigitalInOut(board.D8)

# create the mcp object
mcp = MCP.MCP3008(spi, cs)

# create an analog input channel on pin 0
chan = AnalogIn(mcp, MCP.P0)

from digitalio import DigitalInOut, Direction

in1_pin = digitalio.DigitalInOut(board.D27)
in2_pin = digitalio.DigitalInOut(board.D22)

in1_pin.direction = Direction.OUTPUT
in2_pin.direction = Direction.OUTPUT

#forward
in1_pin.value = True
in2_pin.value = False
#reverse
#in1_pin.value = False
#in2_pin.value = True

# Initialize PWM output for the servo (on pin D17):
enabl = pulseio.PWMOut(board.D17, duty_cycle =25000,frequency
=500,variable_frequency=False)
print('Reading MCP3008 values)
t0=trst1= time.monotonic()
inct= 0.001
tp=0.0
# Main program loop.
while tp<2.01:
    if time.monotonic() -trst1 > inct:
        trst1=time.monotonic()
        tp=time.monotonic()-t0
        print("%.5f %.5f "%(tp, chan.voltage) )
```



As before we can use the value of enabl.duty_cycle to control the speed of the motor, this time we set it within the declaration of the enable pin.

Above shows the result of running the program in Thonny. The first column of the shell window gives time in seconds from t0, the second column gives the value read in volts. In the plot is seen the distinctive dips in voltage as the sensor is periodically shaded by the fan of the motor. The data output in the shell window can be copied analysed for it's periodicity and from that we can work out the fan speed.

Links:
http://www.farnell.com/grh/?mpn=711 ADAFRUIT INDUSTRIES&CMP=os_pdf-datasheet
http://www.ti.com/product/l293d?qgpn=l293d
https://ww1.microchip.com/downloads/en/DeviceDoc/21295d.pdf
http://www.klsele.com/admin/product_upload/20190322144057KLS6-3537..pdf