Adafruit's Raspberry Pi Lesson 10. Stepper Motors

Created by Simon Monk
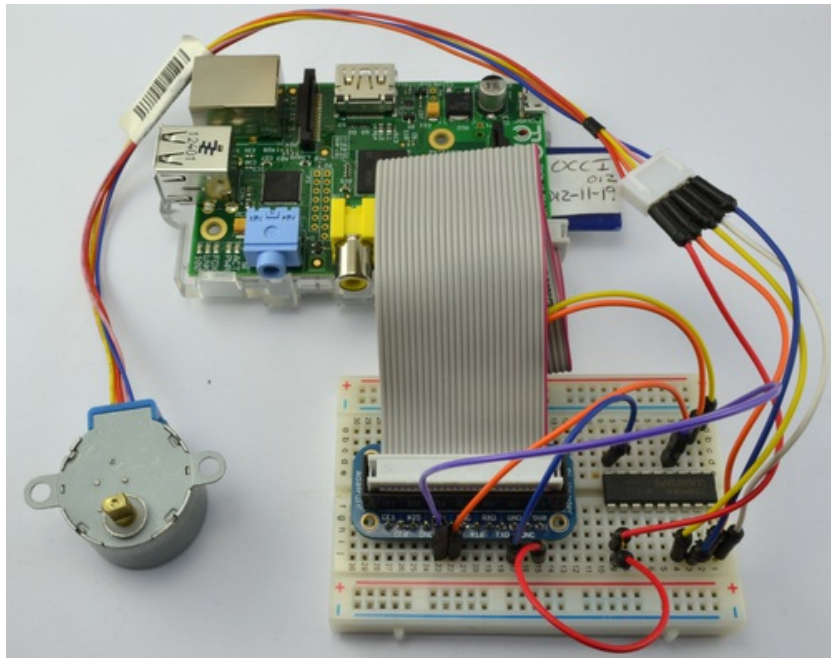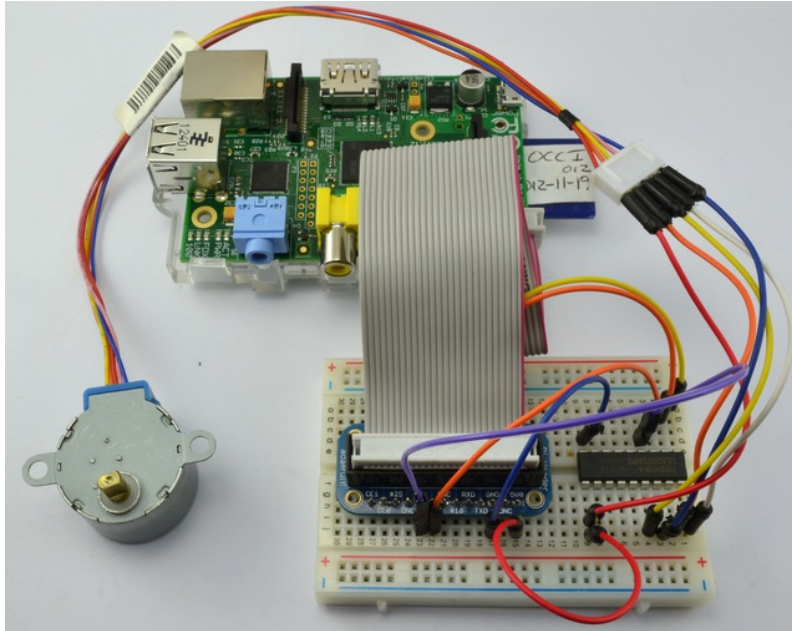
Guide Contents

Overview

Stepper motors fall somewhere in between a regular DC motor (Lesson 9 (http://adafru.it/aWI)) and a servo motor (Lesson 8 (http://adafru.it/aWJ)]). They have the advantage that they can be positioned accurately, moved forward or backwards one 'step' at a time, but they can also rotate continuously.

In this lesson you will learn how to control a stepper motor using your Raspberry Pi and the same L293D motor control chip that you used with the DC motor in Lesson 9 (http://adafru.it/aWI).
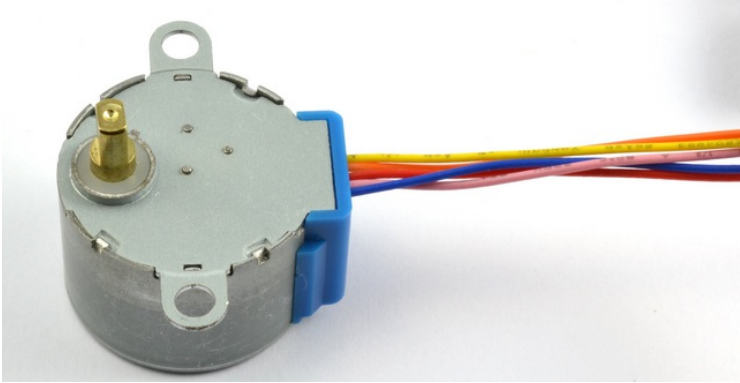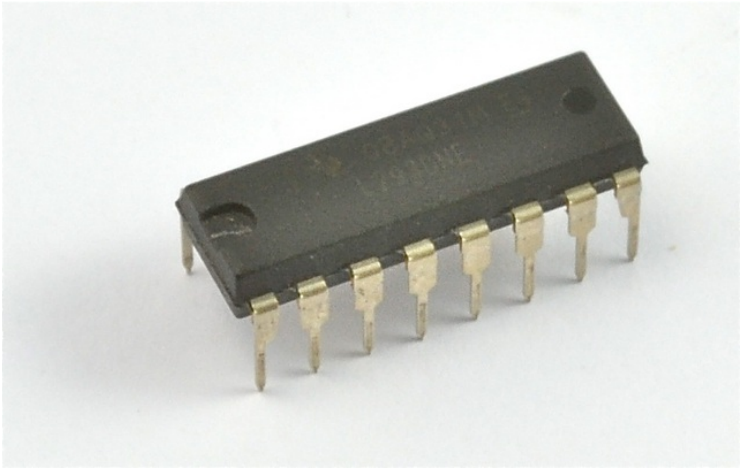
The Lesson will also show you how to use an alternative driver chip, the ULN2803

For this project, it does not really matter if you use a L293D or a ULN2803. The lower cost of the ULN2803 and the four spare outputs, that you could use for something else, probably make it the best choice if you don't have either chip.

The motor is quite low power and suffers less from the surges in current than DC motors and servos (which use DC motors). This project will therefore work okay powered from the 5V line of the Raspberry Pi, as long as the Pi is powered from a good supply of at least 1A.

Parts

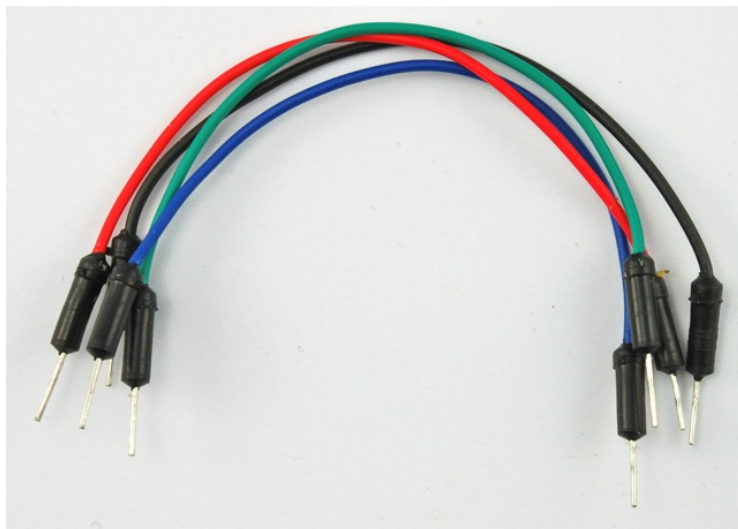To build the project described in this lesson, you will need the following parts.

	Part
	5V Stepper Motor
	L293D IC
	ULN2803
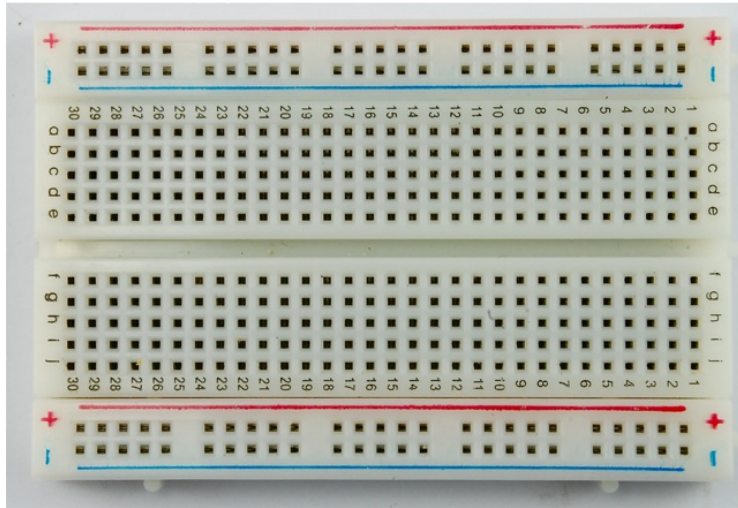
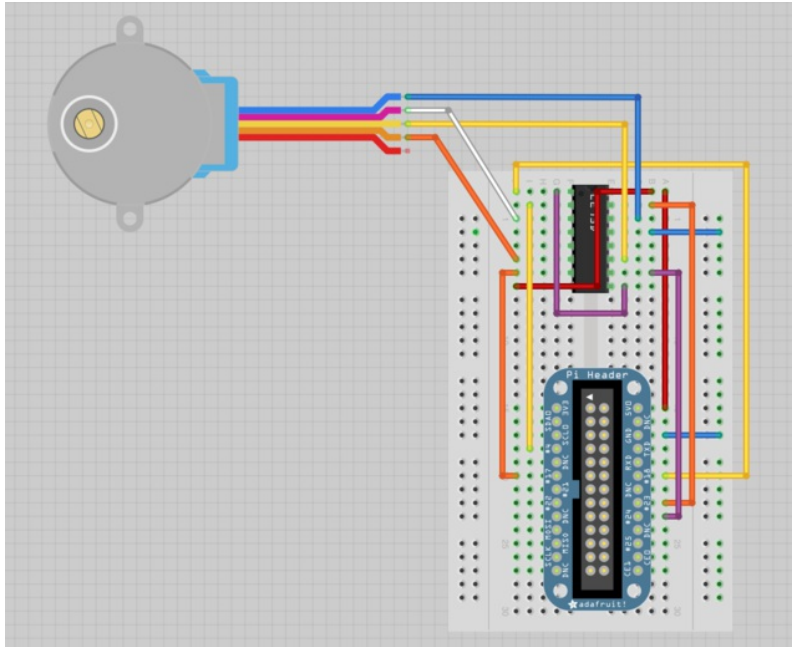|	
	Half-size Breadboard
	Raspberry Pi
	Jumper wire pack

Hardware (L293D)

The stepper motor has five leads, and we will be using both halves of the L293D this time. This means that there are a lot of connections to make on the breadboard.

The motor has a 5-way socket on the end. Push jumper wires into the sockets to allow the motor to be connected to the breadboard.


Note that the red lead of the Stepper motor is not connected to anything.

> Use the colors of the leads to identify them, not the position from which they emerge from the motor.
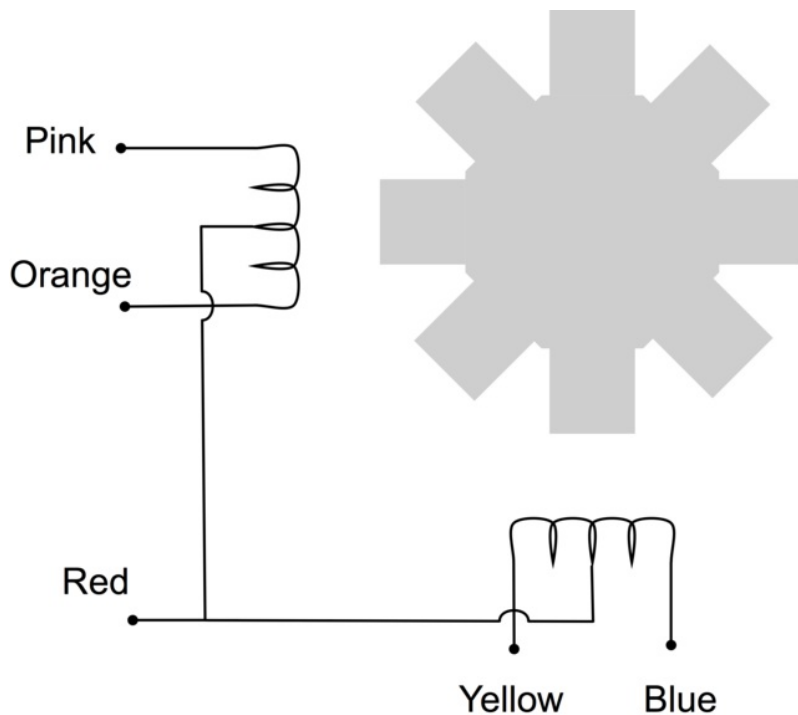
Hardware (ULN2803)

If you are using a ULN2803, then all five of the stepper motor leads are used.

The motor has a 5-way socket on the end. Push jumper wires into the sockets to allow the motor to be connected to the breadboard.
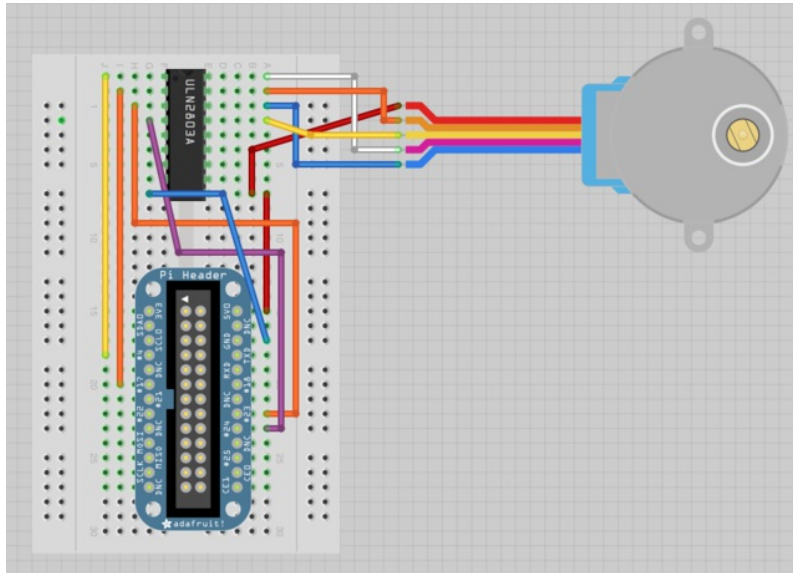
Stepper Motors

Stepper motors us a cogged wheel and electro magnets to nudge the wheel round a 'step' at a time.

By energizing the coils in the right order, the motor is driven round. The number of steps that the stepper motor has in a 360 degree rotation is actually the number of teeth on the cog.

The motor we are using has 8 steps, but then the motor also incorporates a reduction gearbox of 1:64 that means that it needs 8 x 64 = 512 steps.

In this lesson, we do not use the common Red connection. This connection is only provided if you are using a different type of drive circuit that does not allow the current in each coil to be reversed. Having a center connection to each coil means that you can either energise the left or right side of the coil, and get the effect of reversing the current flow without having to use a circuit that can reverse the current.

Since we are using a L293D that is very good at reversing the current, we do not need this common connection, we can supply current in either direction to the whole of each of the coils.
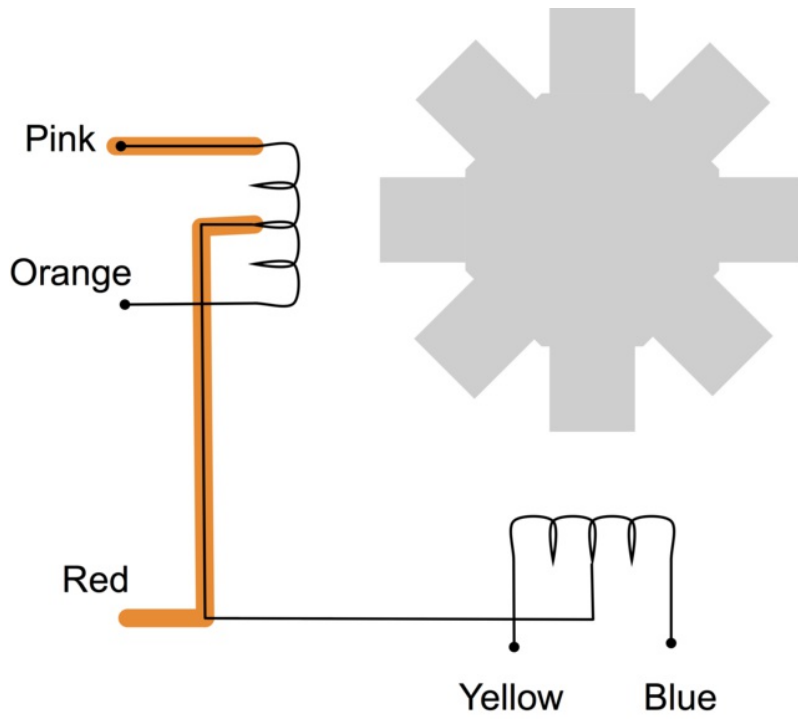
ULN2803

We looked at the L293D in Lesson 9. The ULN2803 is a very useful chip.
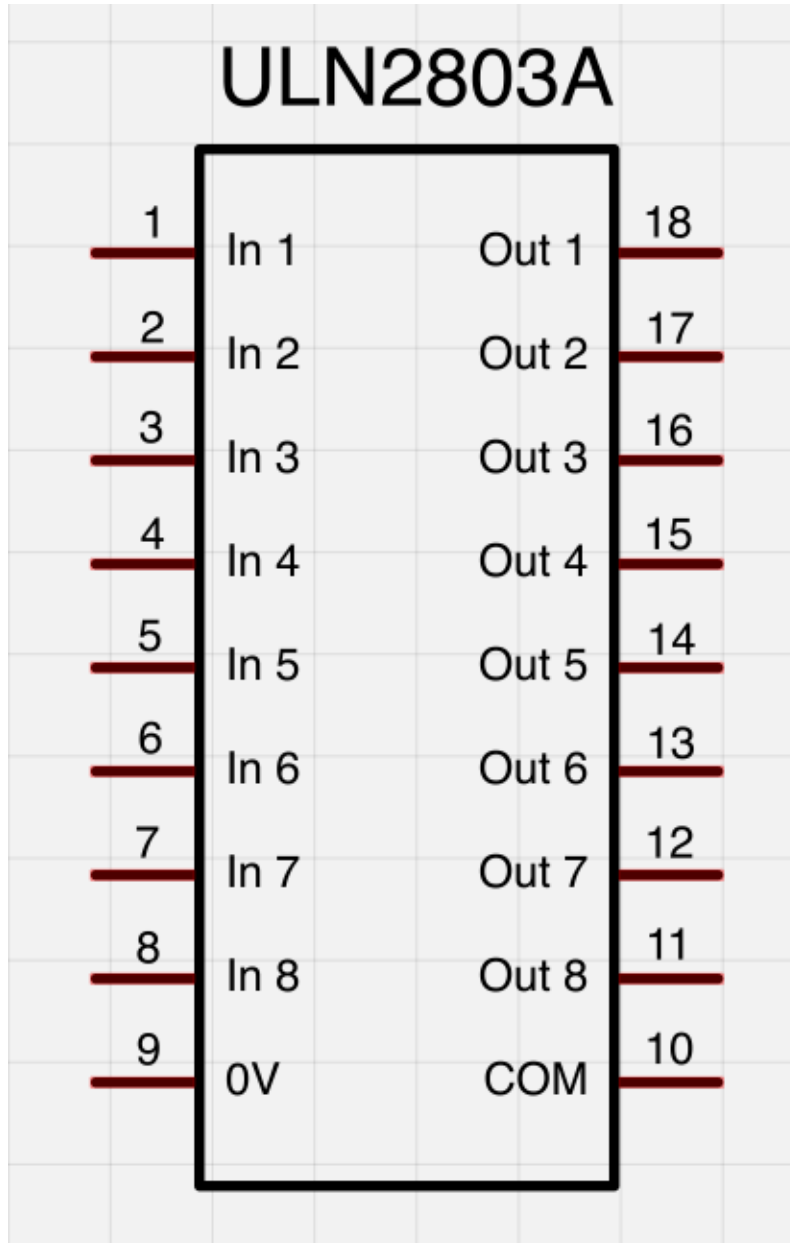
Whereas the L293D effectively has four outputs whose polarity can be reversed, the ULN2803 has eight outputs that amplify the weak signals from the raspberry Pi GPIO pins allowing them to switch much higher currents.

However, unlike the L293D an output from the ULN2803 can only sink current, so the common positive red lead of the stepper motor is used. So, rather than use the whole of the coil between say the Pink and Orange leads, just the half of the coil between the common Red

connection and the Pink connection is energized.

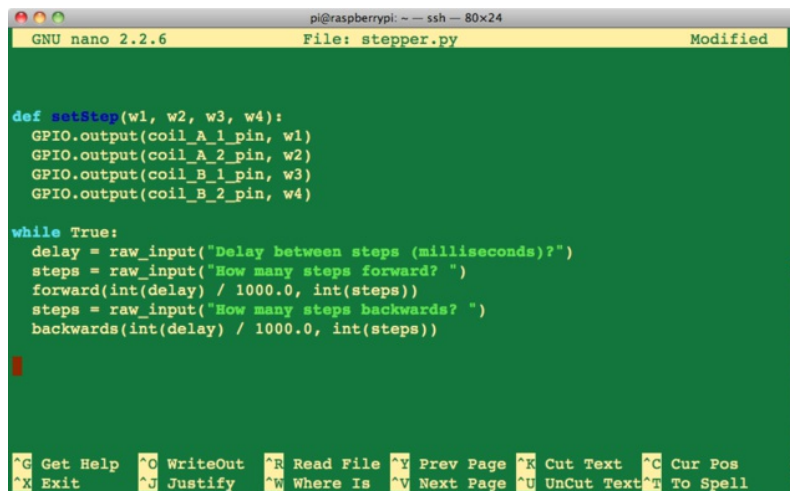http://learn.adafruit.com/adafruits-raspberry-pi-lesson-10-stepper-motors

Software

The software is exactly the same whether you use the L293D or ULN2803 chips.

This project uses the Rpi.GPIO Library. If you have not already done so, you will need to install this (http://adafru.it/aTH).

To install the code, you can connect to your Pi using SSH (http://adafru.it/aWc) and open an editor window by typing:

```
$nano stepper.py
```

Then paste the code below into the editor window and save it using CTRL-X and then Y.


```python
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

enable_pin = 18
coil_A_1_pin = 4
coil_A_2_pin = 17
coil_B_1_pin = 23
coil_B_2_pin = 24

GPIO.setup(enable_pin, GPIO.OUT)
GPIO.setup(coil_A_1_pin, GPIO.OUT)
GPIO.setup(coil_A_2_pin, GPIO.OUT)
GPIO.setup(coil_B_1_pin, GPIO.OUT)
GPIO.setup(coil_B_2_pin, GPIO.OUT)

GPIO.output(enable_pin, 1)
```

```python
def forward(delay, steps):
  for i in range(0, steps):
    setStep(1, 0, 1, 0)
    time.sleep(delay)
    setStep(0, 1, 1, 0)
    time.sleep(delay)
    setStep(0, 1, 0, 1)
    time.sleep(delay)
    setStep(1, 0, 0, 1)
    time.sleep(delay)

def backwards(delay, steps):
  for i in range(0, steps):
    setStep(1, 0, 0, 1)
    time.sleep(delay)
    setStep(0, 1, 0, 1)
    time.sleep(delay)
    setStep(0, 1, 1, 0)
    time.sleep(delay)
    setStep(1, 0, 1, 0)
    time.sleep(delay)


def setStep(w1, w2, w3, w4):
  GPIO.output(coil_A_1_pin, w1)
  GPIO.output(coil_A_2_pin, w2)
  GPIO.output(coil_B_1_pin, w3)
  GPIO.output(coil_B_2_pin, w4)

while True:
  delay = raw_input("Delay between steps (milliseconds)?")
  steps = raw_input("How many steps forward? ")
  forward(int(delay) / 1000.0, int(steps))
  steps = raw_input("How many steps backwards? ")
  backwards(int(delay) / 1000.0, int(steps))
```
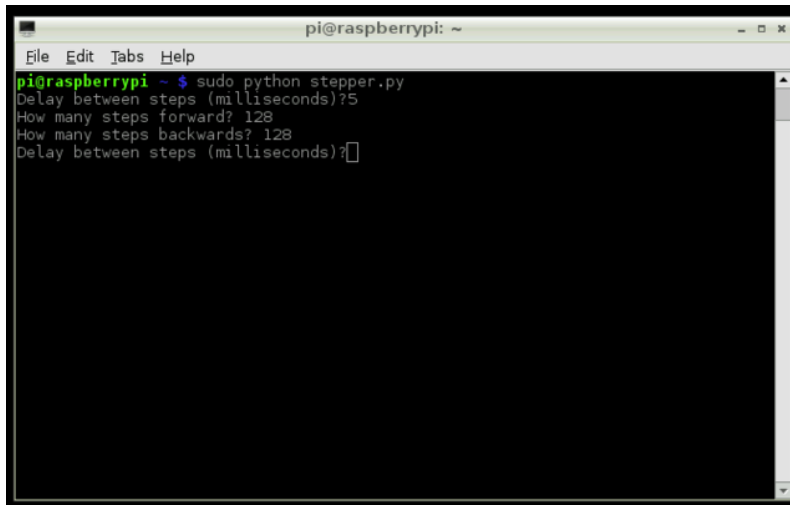
Configure and Test

The program needs to be run as super-user, so enter the following command into the SSH session.

```
$sudo python stepper.py
```

Enter a delay (5 is a good value) and then a number of steps (512 is a full rotation).

Experiment reducing the delay to find the maximum speed of the motor.