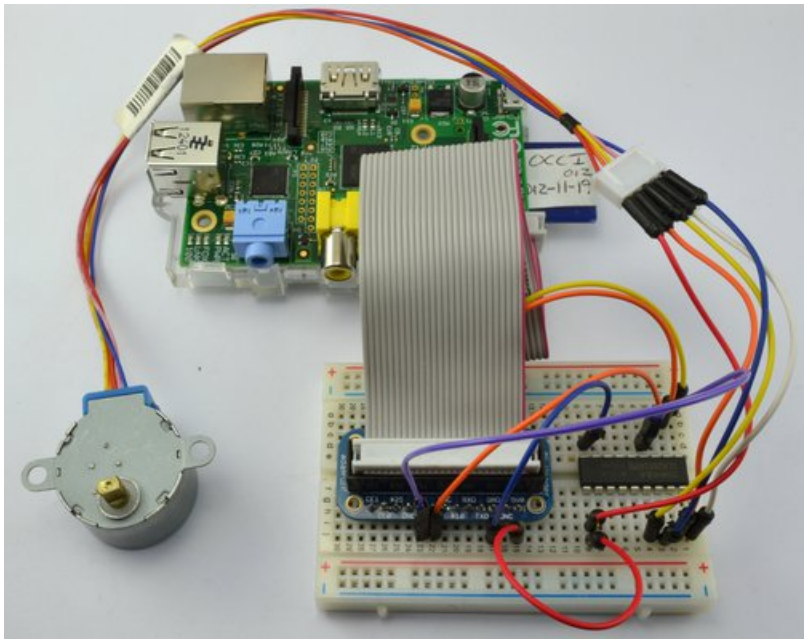
adafruit learning system

Adafruit's Raspberry Pi Lesson 10. Stepper Motors

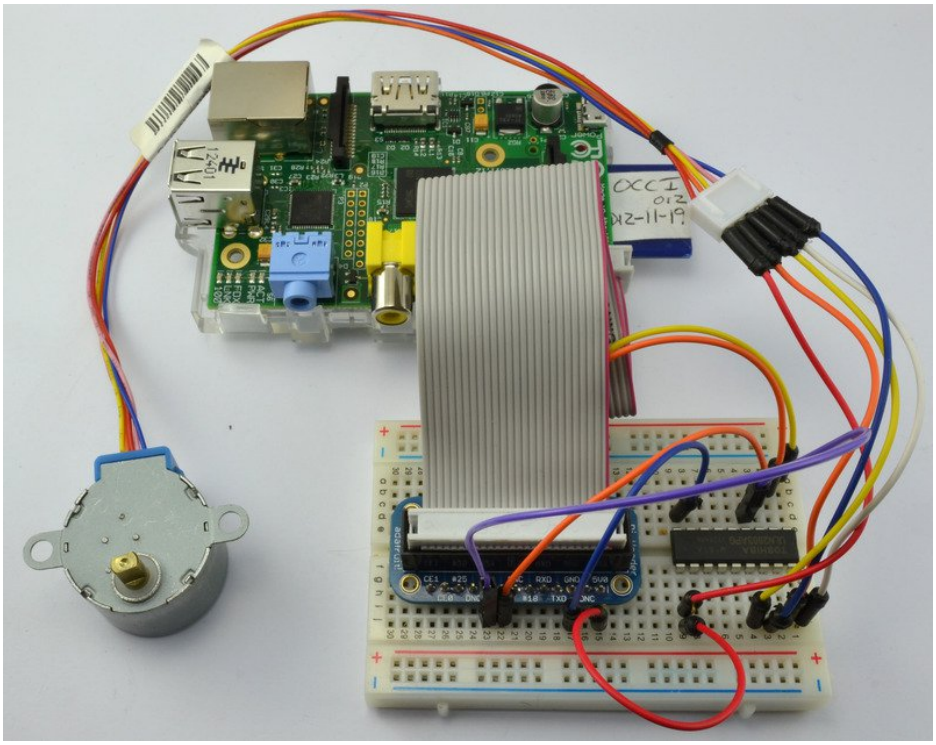
Created by Simon Monk



Last updated on 2019-03-27 08:58:11 PM UTC

Overview

Stepper motors fall somewhere in between a regular DC motor (Lesson 9 (https://adafru.it/aWI)) and a servo motor (Lesson 8 (https://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 (https://adafru.it/aWI).

The Lesson will also show you how to use an alternative driver chip, the ULN2803

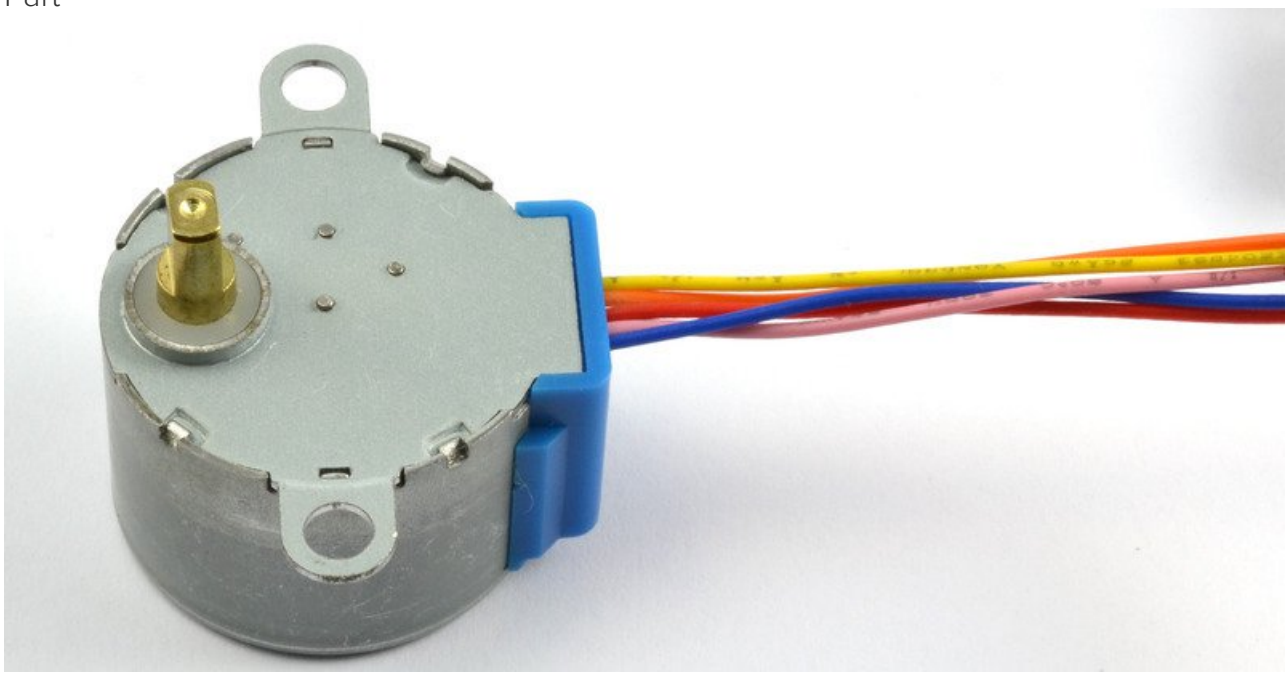
For this project, it does not really matter if you use a L293D or a ULN2803. The lower cost of the ULN2803 and the four spare outputs, that you could use for something else, probably make it the best choice if you don't have either chip.

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.

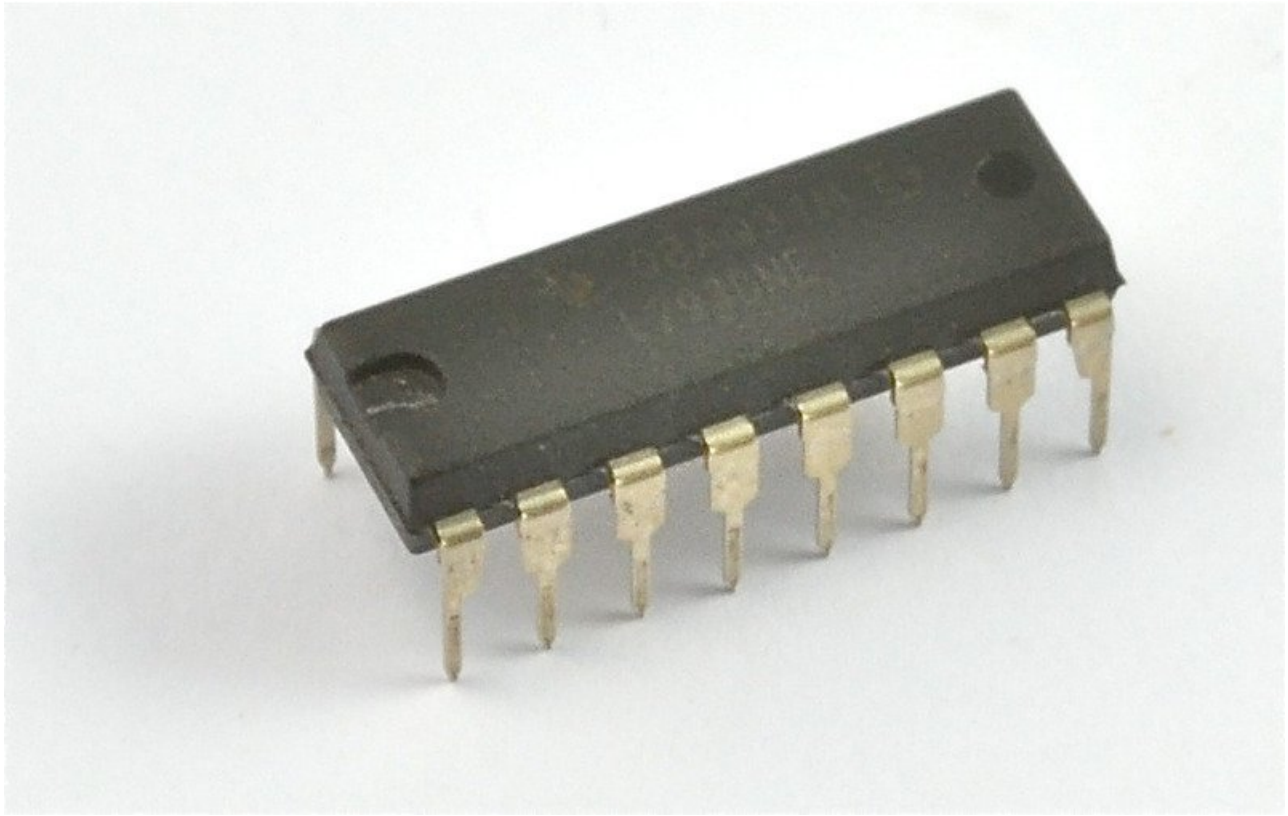
Parts

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

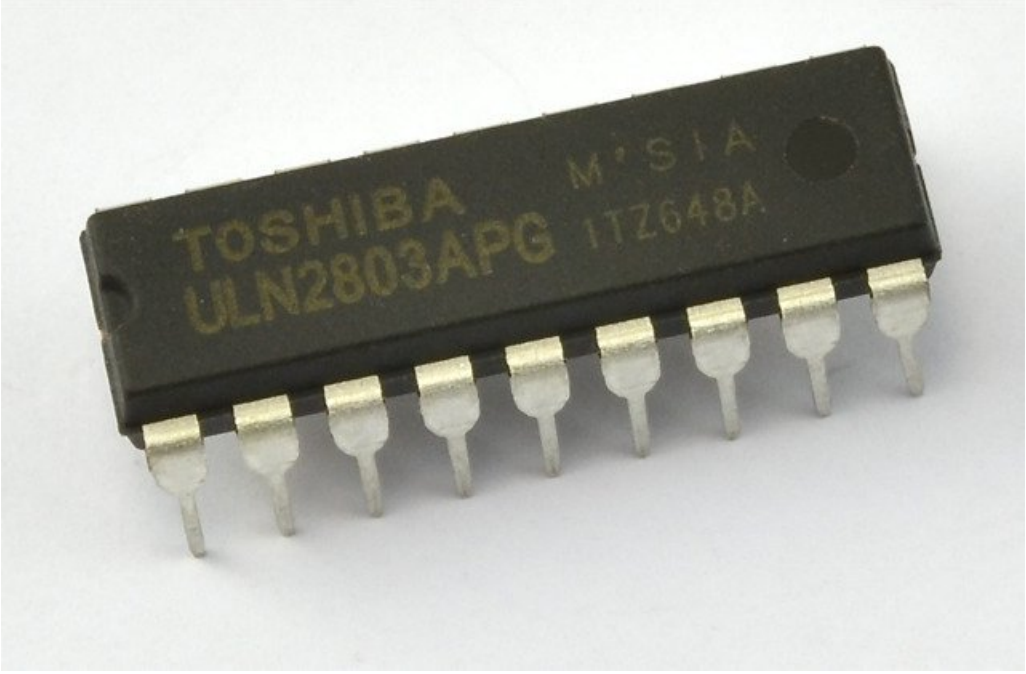
Part



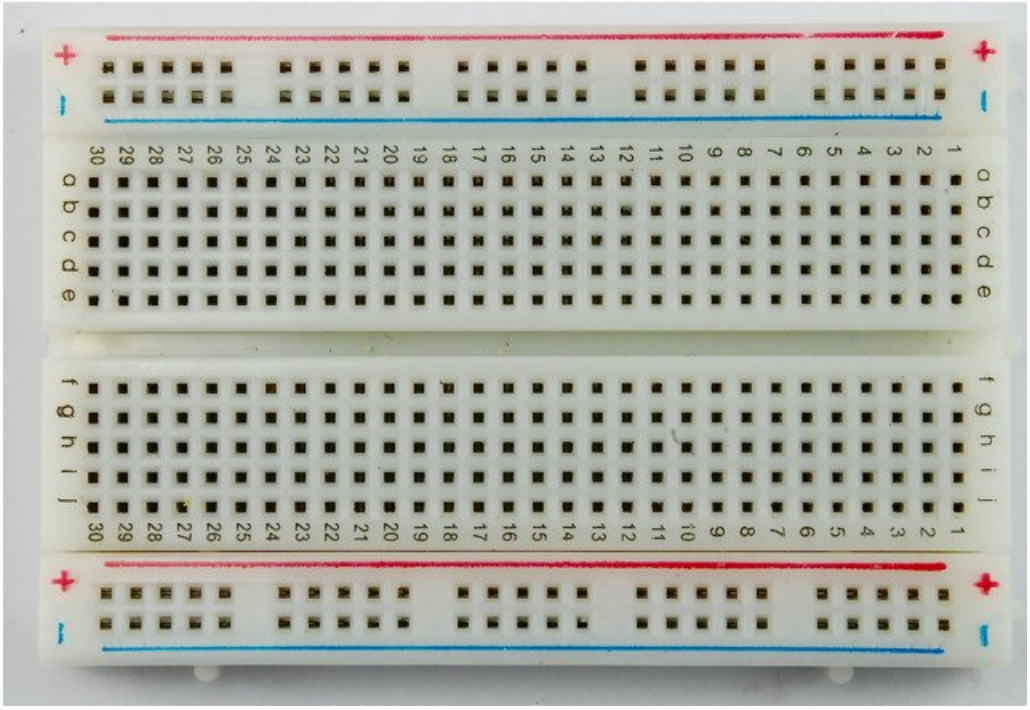
5V Stepper Motor



L293D IC

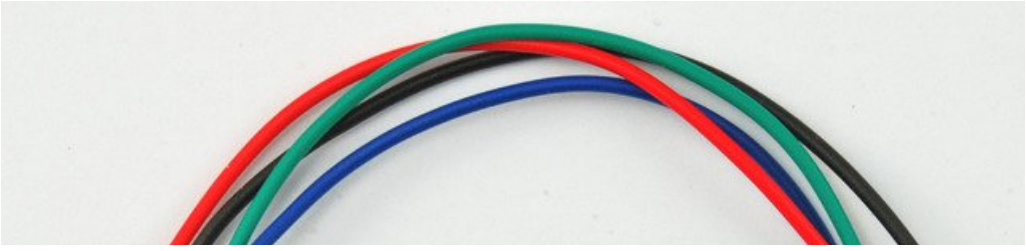


ULN2803



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.

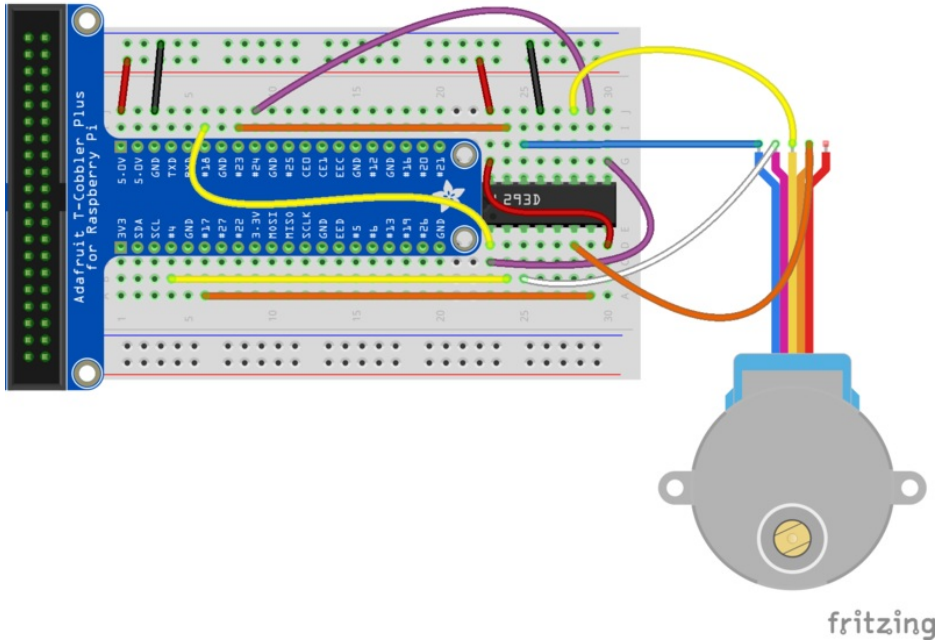
The motor has a 5-way socket on the end. Push jumper wires into the sockets to allow the motor to be connected to the breadboard.

This tutorial works with all versions of Raspberry Pi (rev 1, 2, A, B, B+ and Zero) except the compute module which has no headers.

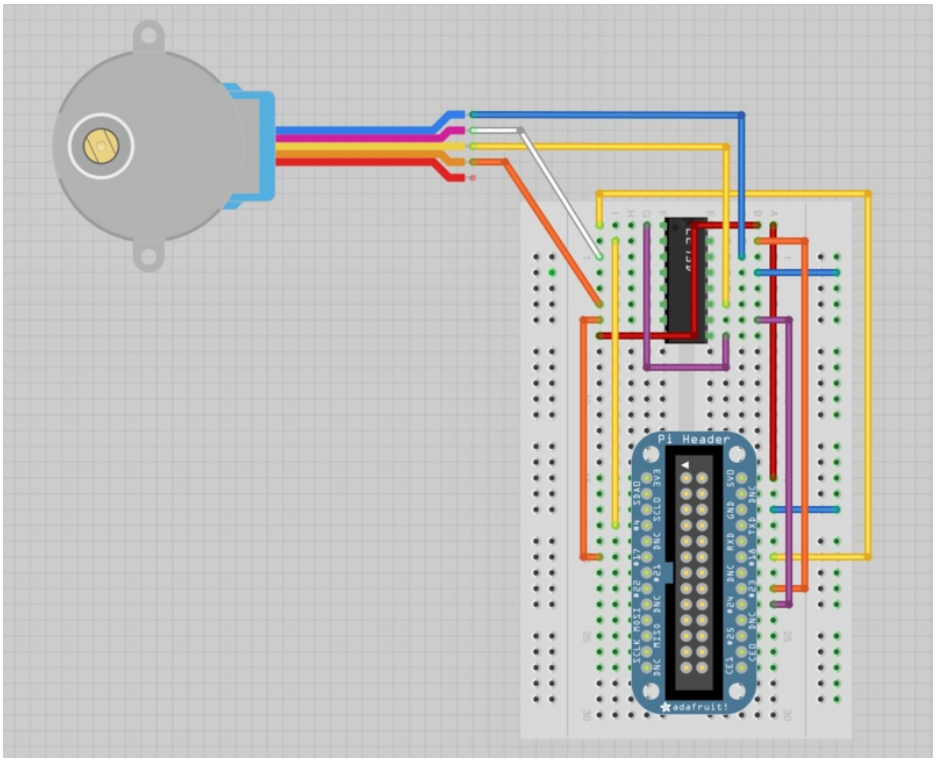
Use the colors of the leads to identify them, not the position from which they emerge from the motor.

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

40-Pin (A, B, B+ and Zero) Cobbler Plus Schematic



20-Pin (Rev 1 and Rev2) Cobbler Schematic


Pi Header

Hardware (ULN2803)

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

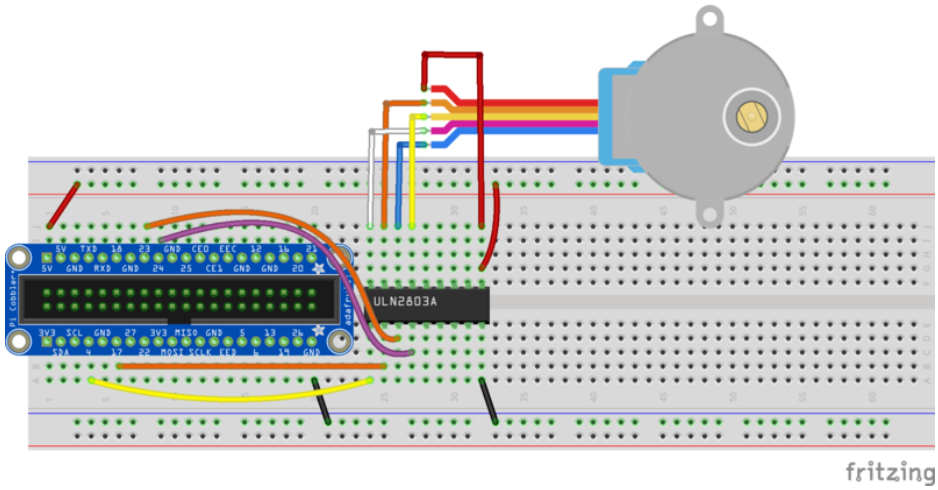
The motor has a 5-way socket on the end. Push jumper wires into the sockets to allow the motor to be connected to the breadboard.

This tutorial works with all versions of Raspberry Pi (rev 1, 2, A, B, B+ and Zero) except the compute module which has no headers.

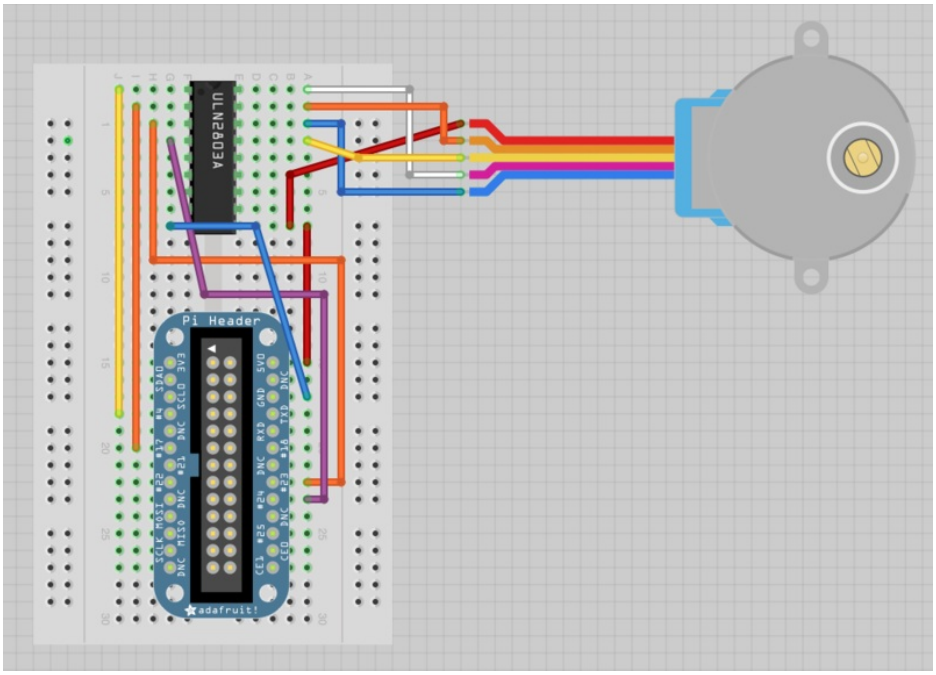
This setup cannot be used with anything but 5-pin (unipolar) stepper motors!

Although the code below mentions pin 18 of the GPIO connector being used as an Enable pin, this is only required when using the L293D.

40-Pin (A, B, B+ and Zero) Cobbler Plus Schematic

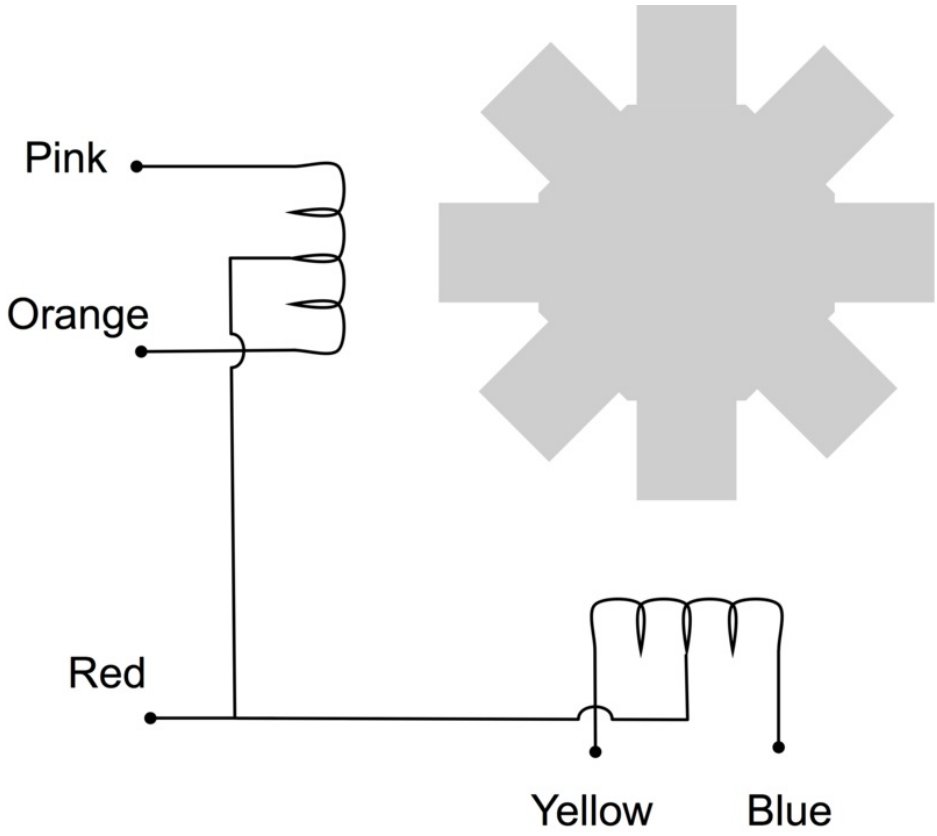


20-Pin (Rev 1 and Rev2) Cobbler Schematic


ULN2803A
Pi Header
adafruit!

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.

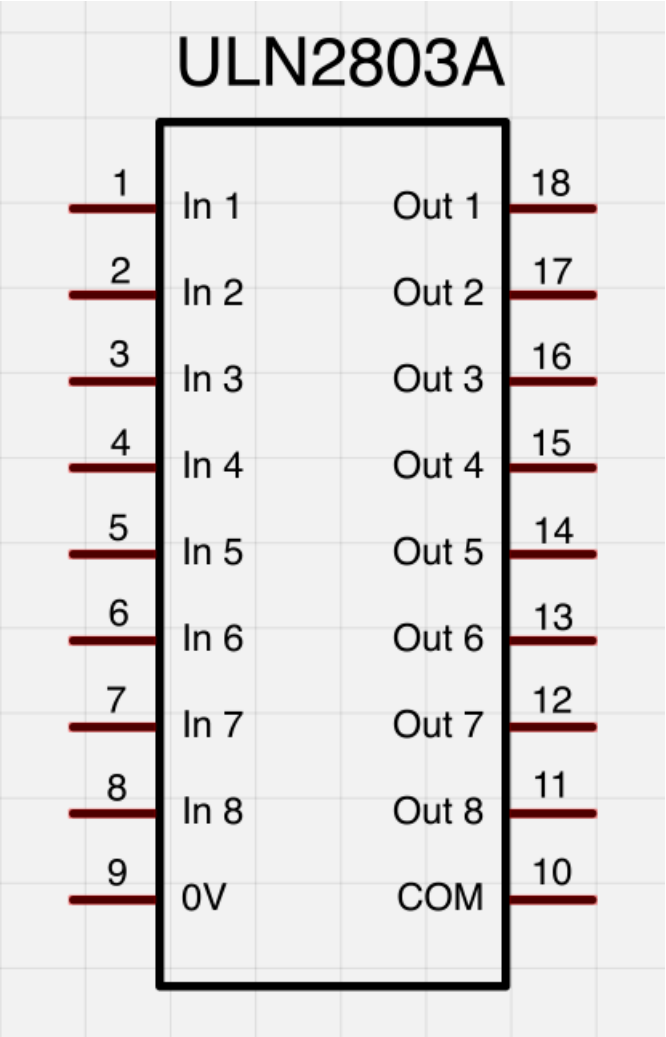
The motor we are using has 8 steps, but then the motor also incorporates a reduction gearbox of 1:64 that means that it needs 8 x 64 = 512 steps.

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.

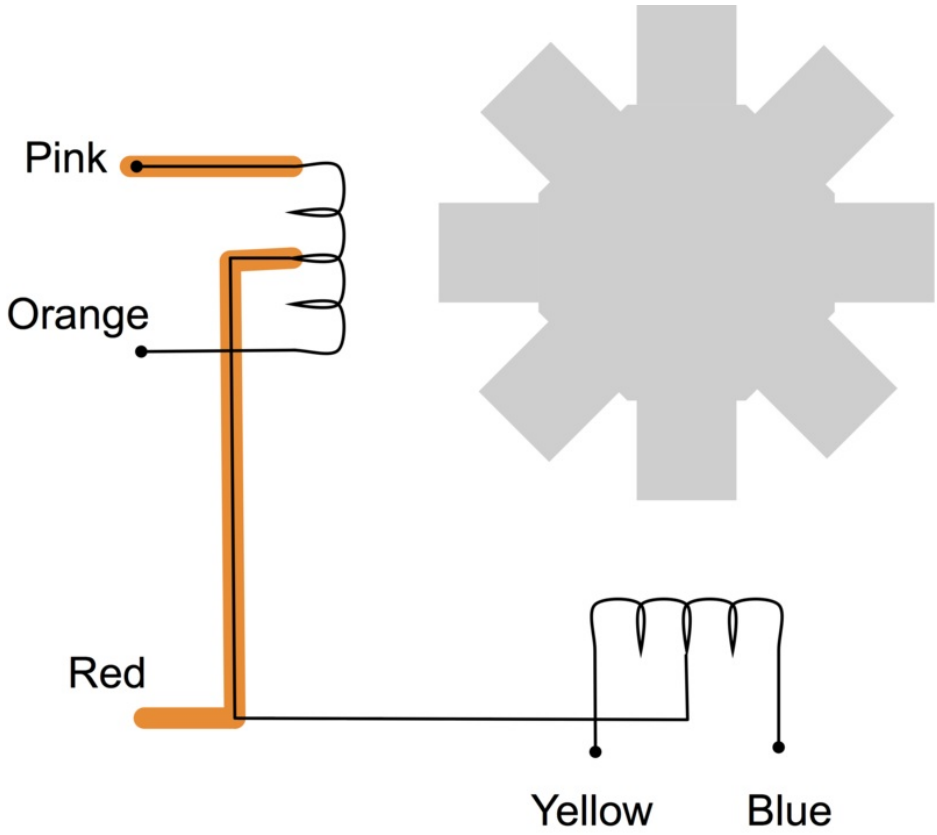
ULN2803

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.


Pink
Orange
Red
Yellow
Blue

Software

The Setup

We are using the CircuitPython Libraries that are part of adafruit-blinka. See CircuitPython Libraries on Raspberry Pi (https://adafru.it/Deo)to get a fresh Raspberry Pi setup.

If you have a running Raspberry Pi with an up to date copy of Raspbian you can simply run the following command to install adafruit-blinka.

The software is exactly the same on all 40-pin and 20-pin Raspberry Pi models. You can use the L293D or ULN2803 chips without any modification the following code:

```
$ sudo pip3 install adafruit-blinka
```

The Code

```
import time
import board
import digitalio

enable_pin = digitalio.DigitalInOut(board.D18)
coil_A_1_pin = digitalio.DigitalInOut(board.D4)
coil_A_2_pin = digitalio.DigitalInOut(board.D17)
coil_B_1_pin = digitalio.DigitalInOut(board.D23)
coil_B_2_pin = digitalio.DigitalInOut(board.D24)

enable_pin.direction = digitalio.Direction.OUTPUT
coil_A_1_pin.direction = digitalio.Direction.OUTPUT
coil_A_2_pin.direction = digitalio.Direction.OUTPUT
coil_B_1_pin.direction = digitalio.Direction.OUTPUT
coil_B_2_pin.direction = digitalio.Direction.OUTPUT

enable_pin.value = True

def forward(delay, steps):
    i = 0
    while 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)
        i += 1

def backwards(delay, steps):
    i = 0
    while 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)
        i += 1

def setStep(w1, w2, w3, w4):
    coil_A_1_pin.value = w1
    coil_A_2_pin.value = w2
    coil_B_1_pin.value = w3
    coil_B_2_pin.value = w4

while True:
    user_delay = input("Delay between steps (milliseconds)?")
    user_steps = input("How many steps forward? ")
    forward(int(user_delay) / 1000.0, int(user_steps))
    user_steps = input("How many steps backwards? ")
    backwards(int(user_delay) / 1000.0, int(user_steps))
```

When the steppers are not moving, they are still 'activated' and hold their position. This draws power. If you don't need

the steppers to 'hold' their position, you can call **setStep(0,0,0,0)** to release the coils. The motor will spin freely and wont draw a lot of current.

Configure and Test

Get the Code

Let's put this file right in your home directory for simplicity. The wget command makes things easy as this can be run from an internet connected Raspberry Pi.

```
$ wget https://raw.githubusercontent.com/adafruit/Adafruit_Learning_System_Guides/master/Raspberry_Pi_Ste
```

Running the Code

Confirm that the Pi connected to the Cobbler (or Cobbler Plus) and run the script.

The following command will start the program:

```
$ sudo python3 ./Raspberry_Pi_Stepper_Motors.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.