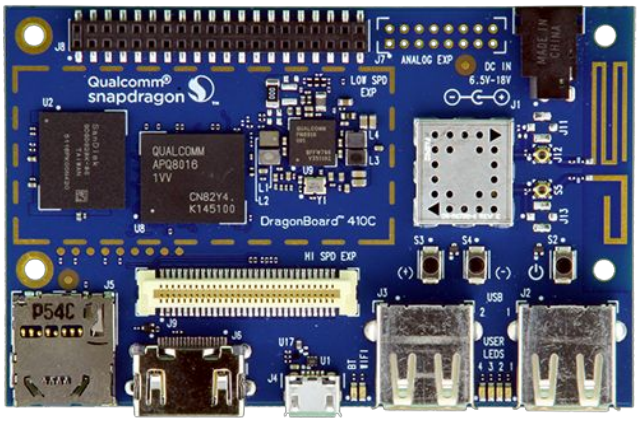


DragonBoard™ 410c

Module 4

Stepper Motors

Table of Contents

Stepper Motors

When working with robotics, motors among several other things have got to be some of the most important components you will chose for a project. This document with compare a variety of different motors widely used in DIY projects, especially DIY projects centered around robotics. It will take a deeper look at the stepper motors and what they are made of. It will then talk about the H-Bridge integrated circuit chip, why it is necessary for this projects and how it is used. Lastly, this document will guide you through the process of building a circuit capable of running a stepper motor. Schematics and code will be provided in order to gain a greater understanding of the stepper motor, as well as to facilitate the step by step instructions in this documents.

Name: Stepper Motors

Project Description: Students will be required to watch all videos and do some research on stepper motors. Students will then use the amplifier they built in Module 3 along with an H-Bridge IC chip to build a circuit for their stepper motor. Students can use the provided stepper motor code, and with small alterations, make a variety of programs centered around the stepper motor.

Code Access:
The code for this module is found at the following link:
https://github.com/IOT-410c/IOT-DB410c-Course-3/tree/master/Modules/Module_4_Stepper_Motors/Lesson_3_Build_and_Demo

To obtain this code, issue the following command:
git clone https://github.com/IOT-410c/IOT-DB410c-Course-3

Then inside of Android Studios, import the project found at
IOT-DB410c-Course-3/Modules/Module_4_Stepper_Motors/**Lesson_3_Build_and_Demo**

Note:

- Ensure the DragonBoard™ 410c is connected to a display
- Ensure the correct power source is connected to the device (DC 12V, 2A)
    - **WARNING**: Exceeding the recommended power could damage the device
- Ensure the device has USB debugging enabled (Android users only)
- Ensure the DragonBoard™ 410c is connected to a computer

Parts List:

a. Stepper motor
b. H-Bridge
c. Amplifier from M3
d. Breadboard
e. Wires

---

1 - Stepper Motors

This section will teach the basics of a stepper motor. The following questions will be answered: what they are, what makes them different from other motors, the components of the motor, and how exactly do they work.

1.1 - What Makes Them Different from Other Motors

In a regular DC motor, two wires (ground and power) give the motor continuous rotation. Although they can reach high speeds, it can only be altered by rapidly turning on and off the power (pulse width modulation). Therefore, it has low accuracy and circuit complexity.

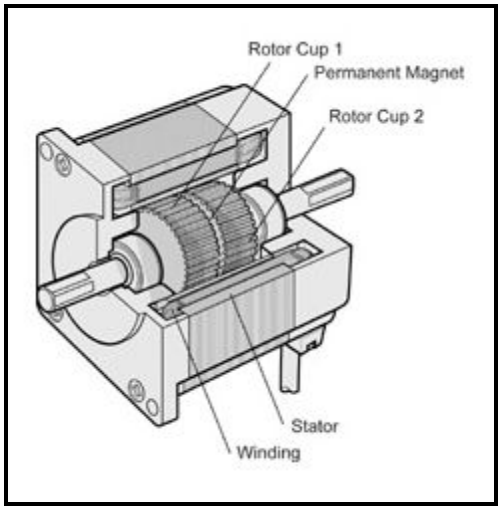
A servo motor is essentially a DC motor that also has a control unit. The servo motor can be controlled more efficiently with the control circuit accurately controlling the speed. However, servo motors are limited to a 180 degrees angle of rotation back and forth.

A stepper motor is very much like a servo motor but is run by electromagnetism. They require an external micro-controller, such as an Arduino or the Dragonboard 410c to individually power the electromagnets inside it. In this way, a stepper motor can be very precisely controlled for position and speed by having pre-defined step angles through 360 degrees. However, their pitfall is that they cannot reach high speeds and have low efficiency.
http://www.modmypi.com/blog/whats-the-difference-between-dc-servo-stepper-motors

	Stepper motor	DC motor	DC motor/ encoder	Servo
Cost $$$	low	low	high	medium
Speed	medium	high	high	medium/slow
Accuracy	high	low	high	medium
Circuit complexity	medium	low	low	low
Programming complexity	high	low	medium/high	low

1.2 - Typical Components of the Stepper Motor

The typical components of the stepper motor are the rotor cups and poles, the permanent magnetic shaft, stator coils and poles, shaft bearings, and the housing.



- rotor cups/poles - located in “core” of the motor
    - heavily magnetic, has ridges for the “steps”
- stator coils
    - coils on the rim that induce a field on the poles that interact with the rotor which allows for the motor to rotate around

1.3 - Deeper Look at How They Work

Stepper motors have something called a step degree. Mentioned earlier, a step represents a ridge on the rotor. And so the step degree is 360

As voltage is applied to the coil of wires located in the stator. This induces an electric current. This electric current creates a magnetic field (electromagnetism). Each coil acts like an electromagnet.

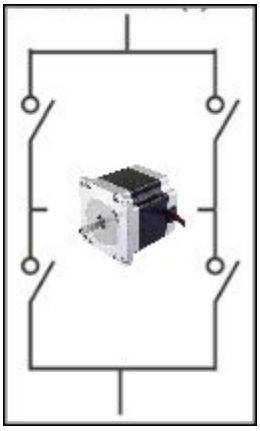
If the left electromagnet is powered on it acts as a north pole and the right electromagnet acts as a south pole. This causes the rotor, which is essentially the magnet in the middle of the stator, to rotate one step. Then the horizontal electromagnets are switched off and the vertical magnets repeat the same procedure. By the method, the motor begins to rotate. For a more detailed explanation, refer to the following website.
http://www.explainthatstuff.com/how-stepper-motors-work.html

It is to be noted that this carries very low efficiency. Power consumption is independent of load, therefore, the stepper motor draws maximum current even when there is no work being done.

2 - Controlling the Stepper Motor

This lesson will teach you about some important components in controlling your stepper motor. We will also be going over how to get everything working.

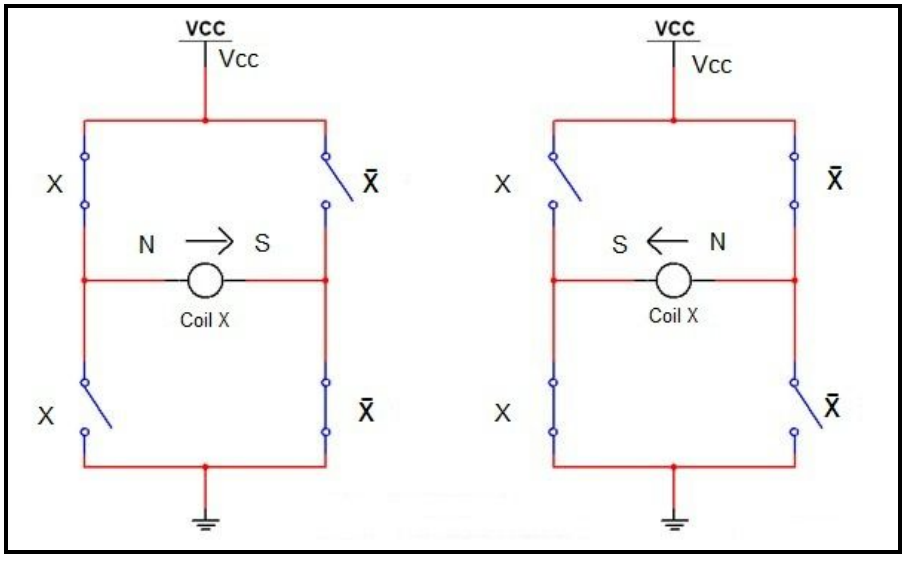
2.1 - H-Bridge IC Chip



As mentioned before, the H-Bridge is basically the spine of the stepper motor. This is because this IC determines the direction the motor will spin (allows for bidirectional turns instead of a one way spin).

Using four GPIOs from your board to connect to the switches (transistors) on the H-Bridge IC, you can control which direction the motor turns. The current will begin flowing at the top and then will go either right or left, then cross the middle (thus turning the motor) and then reach the bottom. This is done by closing and opening the switches, which will be explained more in the next couple sections.

2.2 - Circuit Layout



The H-Bridge is a simple circuit that has 4 switches that can be individually turned on and off. When each is turned on, it carries a voltage across a load that is located in the middle of the bridge.

The top end of the bridge is connected to a power supply and the bottom end is grounded. The load in the middle in this application is a bipolar stepper motor.

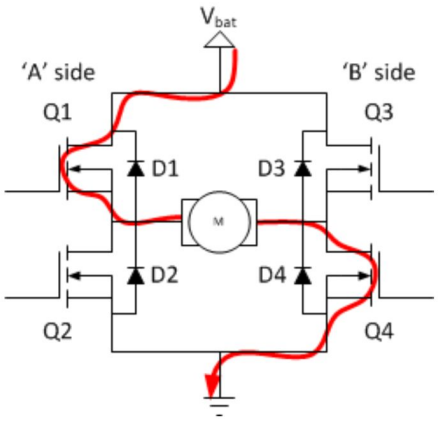
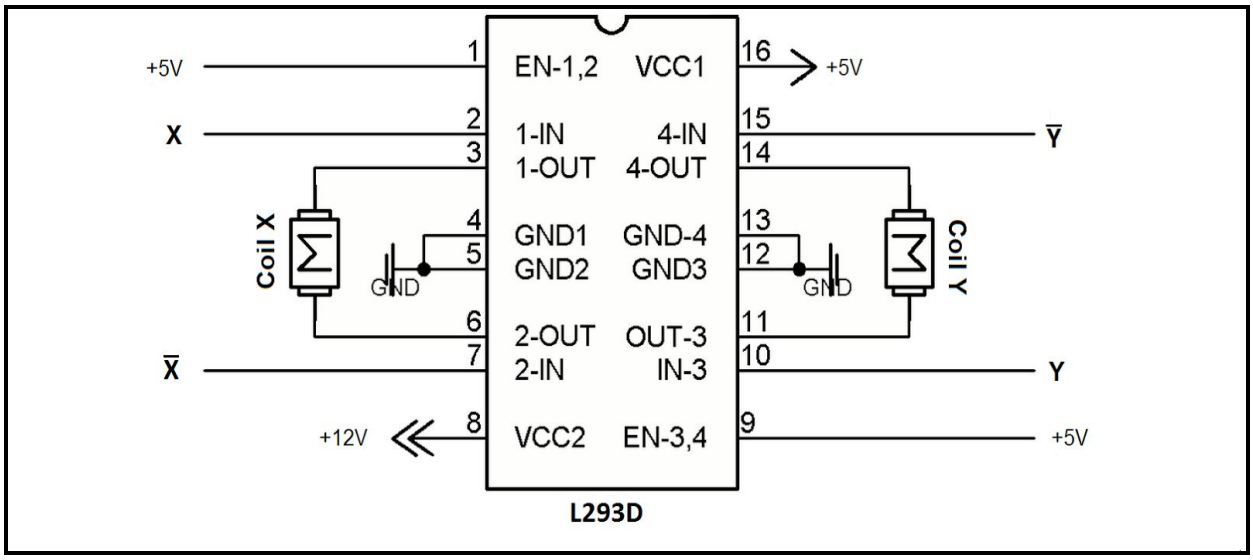


Diagram of a bipolar stepper motor

When Q1 and Q4 are turned on, the left lead is powered and the right lead is grounded. The motor starts spinning as soon as the current starts flowing through the load. If Q2 and Q3 are turned on, the motor starts spinning in the opposite direction. Remember to **never** turn off q2 and q1 or q3 and q4. Doing so will cause a "shoot-through" and will destroy the H-Bridge. thttp://www.modularcircuits.com/blog/articles/h-bridge-secrets/h-bridges-the-basics/



Above is the IC we will be using. Most of the pins are self explanatory. The DragonBoard™ 410c's GPIOs will be connected to the the four X/$\overline{X}$ and Y/$\overline{Y}$ pins. This will all you to control the direction of the stepper motor using 1's and 0's as on/off signals in this circuit.

2.3 - Stepper Sequence

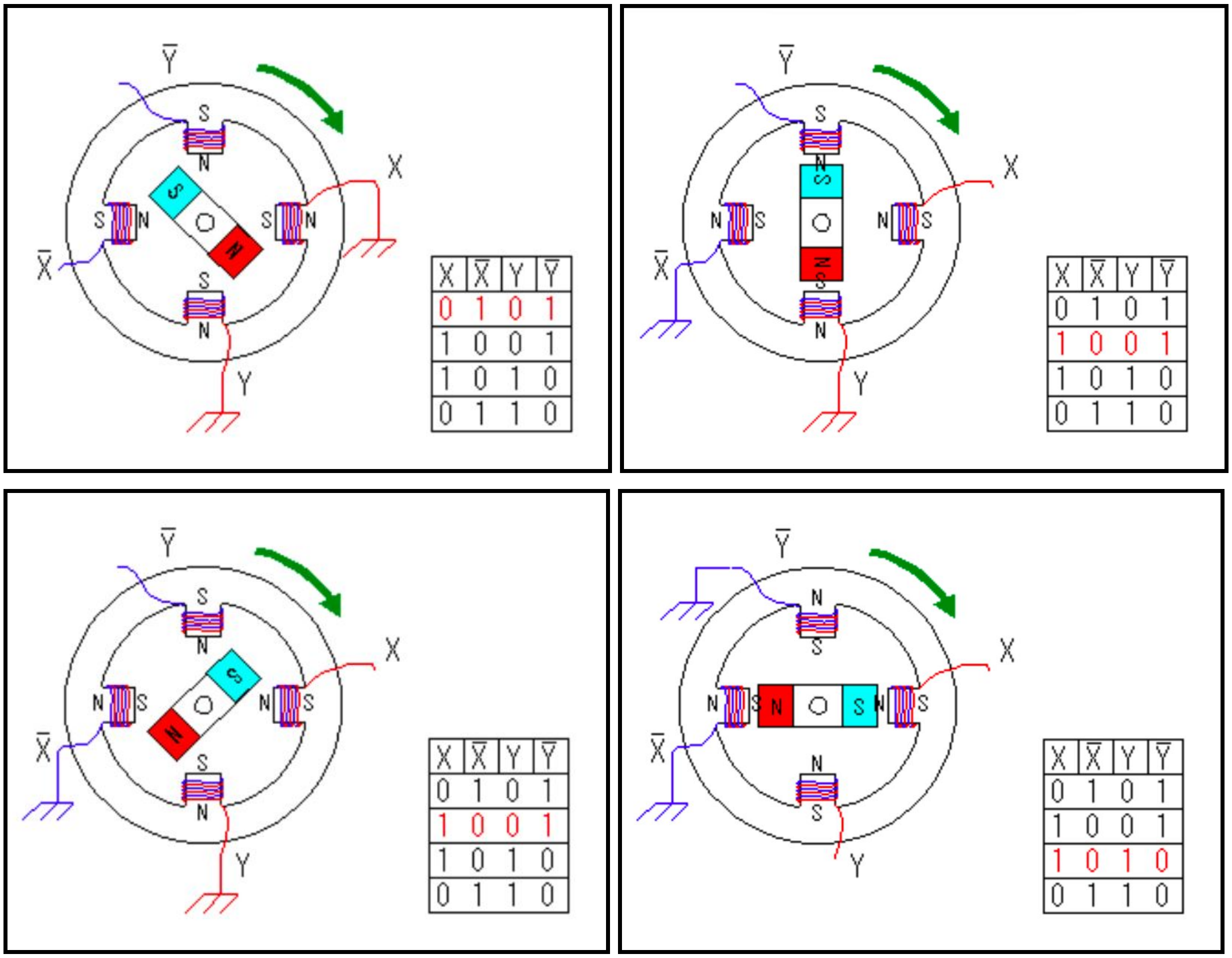
This section will go over the code to allow you to control your stepper motor as it requires a certain sequence the motor has to receive (GPIOs send) for it to work. Since we have plenty of GPIOs, we can just use four of them instead of building or using some kind of flip-flops/switches.

The following chart describes how to control the motor (these were given to us, 0 = open switch and 1 = closed switch). This chart will be important as we will be implementing it later on!

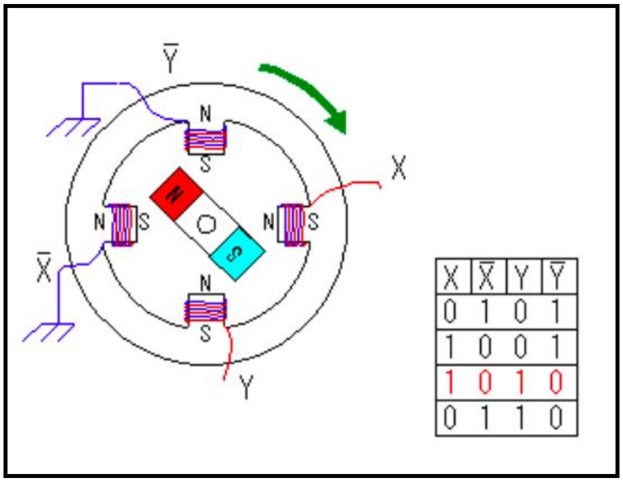
Clockwise				Counterclockwise			
X	$\overline{X}$	Y	$\overline{Y}$	X	$\overline{X}$	Y	$\overline{Y}$
0	1	0	1	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	1	0	1	0
0	1	1	0	1	0	0	1

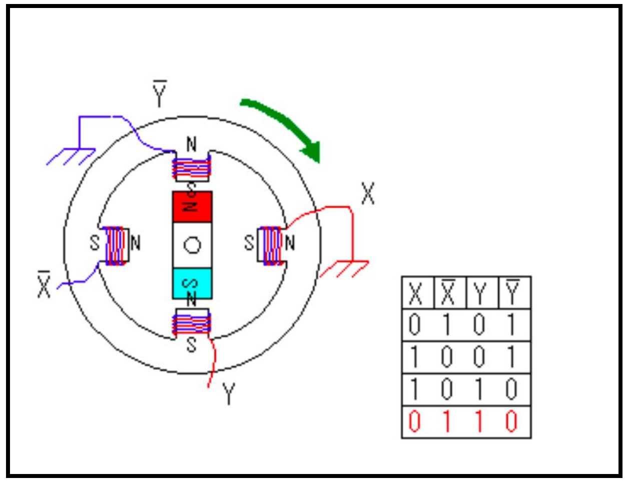
Note: Remember the bar over the X and Y means **not** (sometimes written ~X, ~Y or !X, !Y) which is the opposite of X and Y (i.e. if X = 0, $\overline{X}$ = 1).

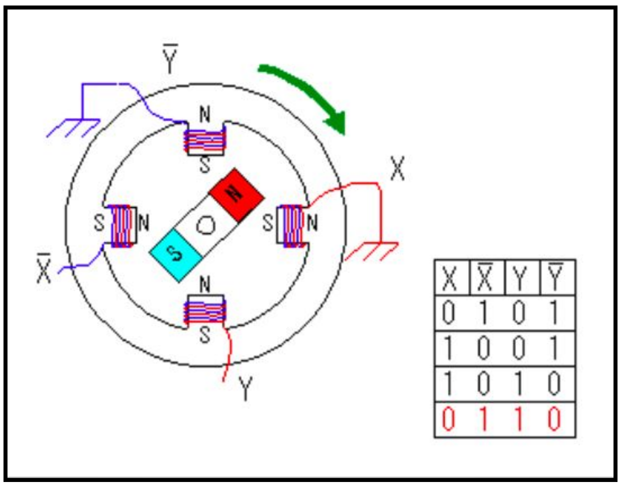
Below shows the process of the forward motion (read left to right, top to bottom). Basically, theses four inputs will control the magnets inside the motor that act like gates (the coils). When the correct signal is given (depending on where the rotor cup/pole is), the gates will “open” (the magnets will attract/push) and the rotor will spin through the open gate until it stops at the next one. And so you can continue the cycle to complete a full rotation.

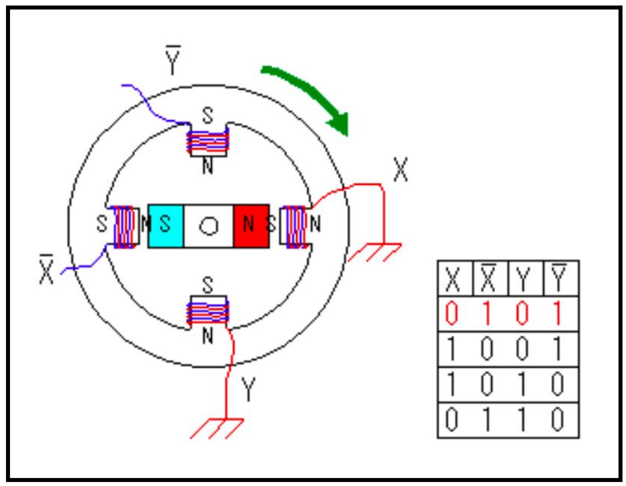


Ȳ
X
X̄
Y
S
N
X X̄ Y Ȳ
0 1 0 1
1 0 0 1
1 0 1 0
0 1 1 0

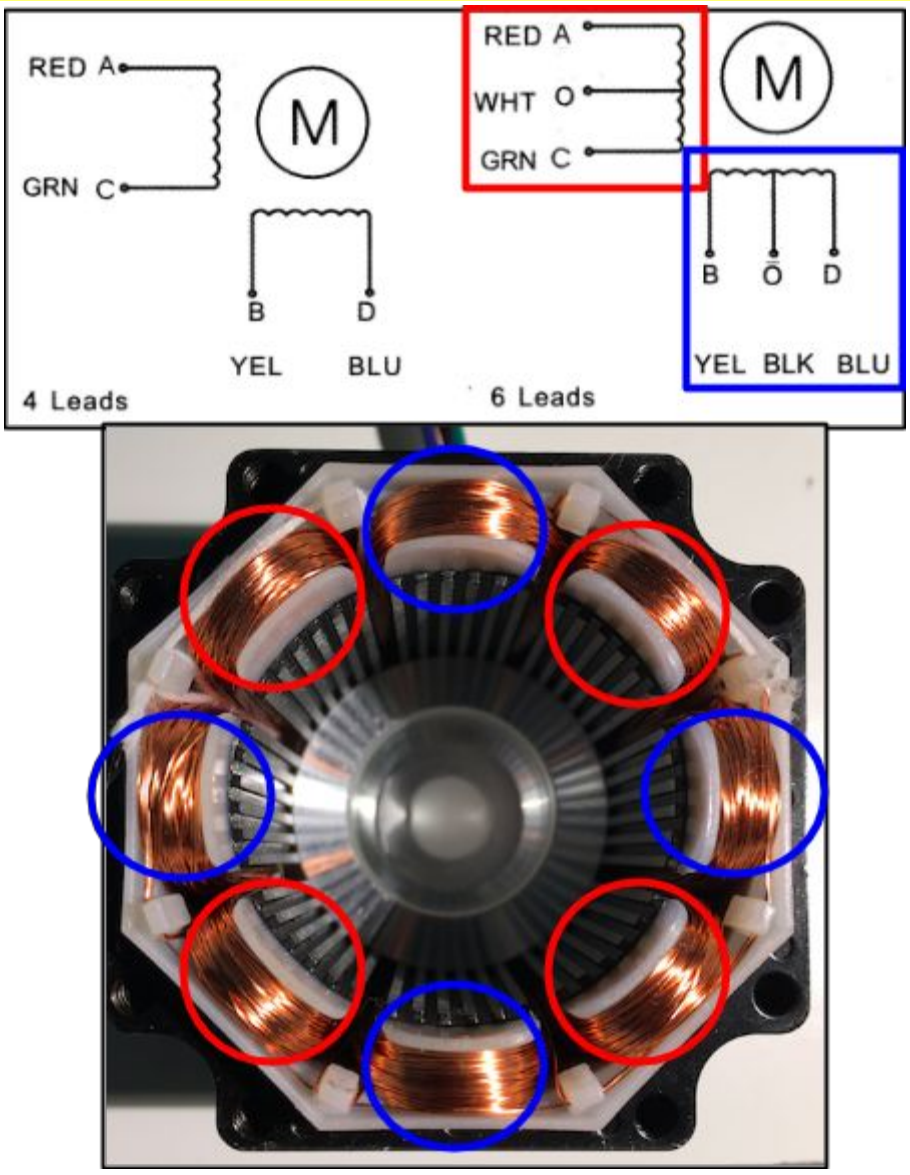

Ȳ
X
X̄
Y
N
S
N
S
N
S
N
S
X X̄ Y Ȳ
0 1 0 1
1 0 0 1
1 0 1 0
0 1 1 0


Ȳ
X
X̄
Y
N
S
S
N
S
N
N
S
N
S
X X̄ Y Ȳ
0 1 0 1
1 0 0 1
1 0 1 0
0 1 1 0


Ȳ
X
X̄
Y
N
S
S
N
S
N
N
S
N
S
X X̄ Y Ȳ
0 1 0 1
1 0 0 1
1 0 1 0
0 1 1 0


Ȳ
X
X̄
Y
S
N
S
N
S
N
S
N
S
N
X X̄ Y Ȳ
0 1 0 1
1 0 0 1
1 0 1 0
0 1 1 0

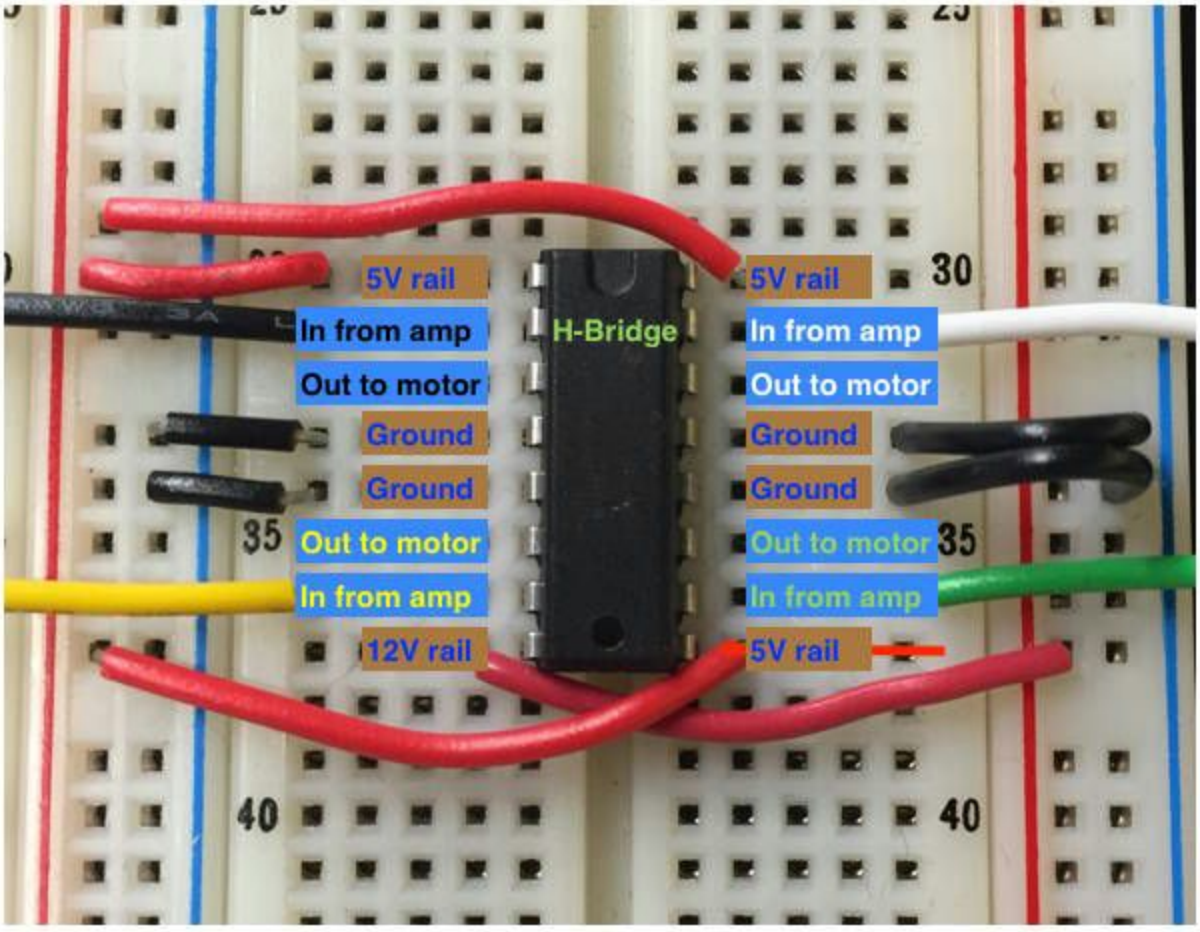
Now, the diagram below shows you how the stepper motor's wires work. If you bought the same one we are using, you'll notice there are six wires that will control the magnets (coils). The red/white/green is one set while the blue/black/yellow set is another (they will be turned on interchangeably, also color coded in the diagram). Basically, you'll want to choose a higher torque and longer charge time, or a lower torque and faster charge time (higher speed). And so using the red/green + blue/yellow setup gives you a higher torque (but sacrifices speed) while a white/green + black/yellow setup will give you a higher speed (but sacrifices torque).



3 - Build and Demo

This lesson will go over the circuit build of the motor as well as the code setup. At the end of the lesson, we have a demo for you to run so you can test your new, working stepper motor.

3.1 - Building the Circuit



3.2 - Setting up the Code

Let's take a look at the program we wrote to use the stepper motor. Remember, these files need to be in the same directory as your GPIOLibrary.py.

Here's the code:

```
from GPIOLibrary import GPIOProcessor
import time
import math

GP = GPIOProcessor()

try:
    # Stepper Motor Controls
    A1 = GP.getPin23() # Green
    A2 = GP.getPin24() # Black
    B1 = GP.getPin25() # White
    B2 = GP.getPin26() # Yellow

    A1.out()
    A2.out()
    B1.out()
    B2.out()

    # Delay time
    T = 0.001 # 1 ms

    # Stepper Sequence (Forward, Reverse)
    SS = [[[0,1,0,1], [1,0,0,1], [1,0,1,0], [0,1,1,0]],
          [[0,1,0,1], [0,1,1,0], [1,0,1,0], [1,0,0,1]]]

    # Forward/Reverse (0 = Forward, 1 = Reverse)
    FR = 0

    # Step Angle
    SA = 1.8  # 1.8 degrees per step

    while True:
        print 'Enter degrees:'
        x = input()

        # check if (-) degrees for reverse direction
        if x < 0:
            FR = 1
            x = abs(x)
        else:
            FR = 0

        x = int(x/SA) # calculate number of steps
```

```
        # run stepper sequence
        for i in range(0,x):
            A1.setValue(SS[FR][i%4][0])
            time.sleep(T)
            A2.setValue(SS[FR][i%4][0])
            time.sleep(T)
            B1.setValue(SS[FR][i%4][0])
            time.sleep(T)
            B2.setValue(SS[FR][i%4][0])
            time.sleep(T)

        print 'again? [y/n]:'
        r = raw_input()
        if r == 'n':
            break

finally:
    GP.cleanup()
```

Some key things to note:

- Green/Black wires and White/Yellow are opposites (hence Green/Black = A1/A2 and White/Yellow = B1/B2, and each of these comes is a GPIO output from the board).
- A delay time is how long the signal should be kept (it takes some time for the rotor to move). For this motor, we found **1ms** to be perfect. This will vary between motors (experiment)
- next we setup the stepper sequence for forward and reverse motion
- SA stands for Step angle (or degrees per step, this may vary motor to motor)
- the SA is important because the program asks you by how many degrees do you want to turn your motor
- finally, after the direction is determined, a loop is used to determine which sequence is needed and how many times it must be repeated for the motor to turn the desired number of degrees

3.3 - Demo of Stepper Motor

Now that everything is ready, let's run it!

After you get into your directory with the GPIOLibrary and the program, use **sudo python [filename].py** to run it.

Now, just input a number and your motor should turn the number of degrees you inputted (note there will be some rounding errors with the way the code is setup, so it won't always turn exactly X number of degrees but it will be close).

References