ENME 461 Lab 2- Actuators: DC and Stepper Motors

1. Introduction

This lab studies two of the most common actuators in mechatronic systems; the DC motor and the stepper motor. Functionality of the motors is introduced and issues are explored in driving both types of motors and controlling them using a digital microcontroller.

To complete the lab, show all completed projects to your teaching assistant.

2. Objectives

- To develop an understanding of DC and stepper motor fundamentals.
- To understand and apply semiconductor electronic elements to switch a DC motor on and off.
- To understand the concept of and take preventative measures against back EMF.
- To apply pulse width modulation to control the speed of a DC motor.
- To use a dedicated driver circuit for a stepper motor.
- To understand and apply driving methods for a stepper motor using a microcontroller.

3. Project One: Switching a DC motor on and off

The objective of this project is to turn the motor on for two seconds and turn it off for two seconds continuously using a digital microcontroller. Unlike an LED, motors are generally powered from a source other than the microcontroller. The microcontroller performs the controlling action. This calls for additional switching elements. In this lab we use a transistor as the switching element.

3.1. DC Motor

DC motors convert direct current electrical energy into rotational mechanical motion. DC motors are widely used in mechatronic applications including robotic manipulators, transport mechanisms, machine tools, etc.

The operation of a DC motor is based on electromagnetism. For a permanent magnet DC motor (such as the motor in the lab), the stator, which is the non-rotating part of the motor, is a permanent magnet that generates a magnetic field with a constant magnetic flux. The rotor, which is the rotating part of the motor, is a coil of wire placed in this field. When the motor is powered, current runs through the coil, creating a second magnetic field. The interaction of these two fields creates forces which generate torque for motor rotation.

A main characteristic of all DC motors is the generation of back electromotive force (EMF). This is the voltage created due to the rotation of a current carrying coil in a magnetic field. The back EMF opposes the powering voltage.

The DC motor used in the lab is the hobby grade permanent magnet DC motor shown in Figure 1. The motor is rated for operation between 3 and 6 V and draws currents in the range of 200mA without a load. It is part of a three-wheel moving platform created to build moving robot type devices.

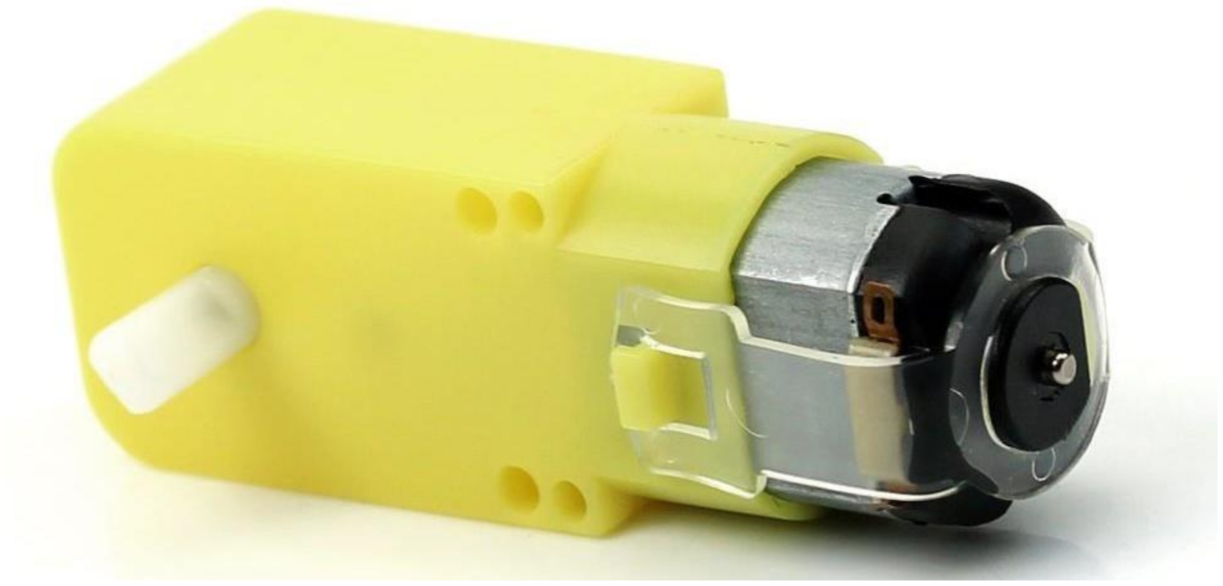


Figure 1- DC motor in 461 Labs

3.2. Transistor and diode

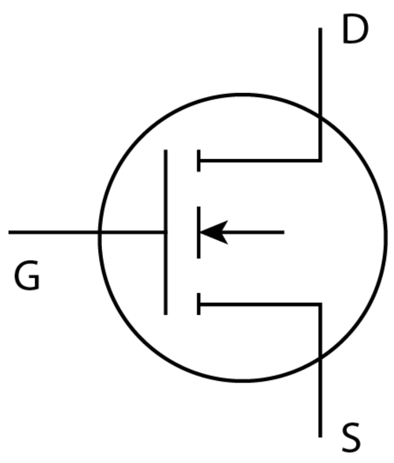
A transistor is a solid-state (semi-conductor based) switch that opens or closes a circuit. Switching action in a transistor is due to the change in its electrical characteristics. Transistors are three terminal devices. One terminal is used as the control input, another is connected to the load voltage and the third is usually grounded. There are several types of transistors available but the two types that are used most widely are the bipolar junction transistor (BJT) and the metal-oxide semi-conductor field effect transistor (MOSFET). Since this lab uses the MOSFET, the discussion here is focused on this type of transistor.

The three terminals of the MOSFET are called the *gate*, the *drain*, and the *source*. These are shown in the MOSFET symbol in Figure 2.

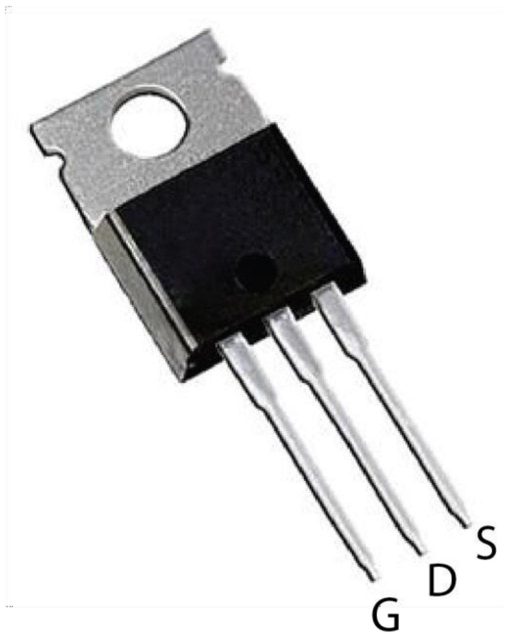
MOSFETs also have some variety, but the most common type (the MOSFET in the lab) is the n-type enhanced MOSFET.

- MOSFETS are *voltage* controlled devices. A *voltage* applied to the *gate* controls the operation of the MOSFET.
- When the transistor is off, the drain source resistance is very high and no current flows between *drain* and *source*.
- When the transistor is fully on, the drain-source resistance is very low and current flows from ***drain to source***.

The *gate* is insulated from the *drain-source* circuit. This is indicated by the dashed line in Figure 2a. No current flows into the MOSFET from the *gate*.



a) MOSFET Symbol. Applying voltage to the gate (G) allows current to flow from drain (D) to source (S).



b) The IRF 520 Power MOSFET in the lab

Figure 2- MOSFET symbol and picture

A typical implementation of transistors in DC motor control is as switches that turn DC motors on and off from a digital source.

This is shown schematically in Figure 3 and explored in this project. When the transistor is off, no current flows from *drain* to *source* and the motor is off. Applying a small voltage V_{in} to the *gate* (using the microcontroller digital pin) turns the transistor on, current is allowed to flow to the motor through the *drain-source* connection which now has low resistance and the motor turns on.

A “flyback” diode is implemented in this circuit. Diodes (like LEDs from Lab 1) allow current to flow in one direction only, from the positive side (anode) to the negative side (cathode). The diode symbol and picture are shown in Figure 4. Note the band in the diode picture indicates the negative side (cathode).

When the transistor is switched off in the circuit of Figure 3, the diode prevents the back EMF voltage build-up (acting in the opposite direction of the power source) from damaging the transistor. The resistor at the *gate* drives the input to the ground and completely turns off the transistor.

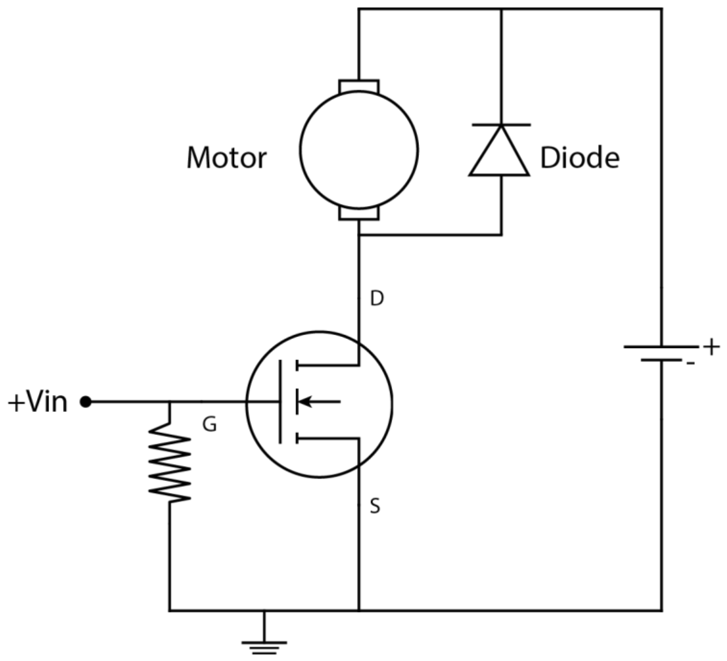


Figure 3- A transistor as a switch for DC motor control

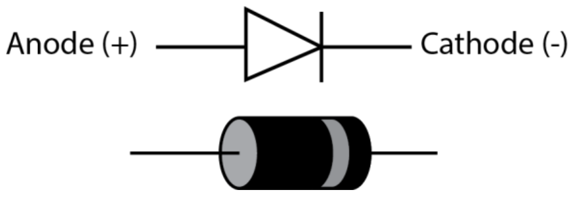


Figure 4- Diode symbol and picture. Current can only flow from anode to cathode. Note the band on the diode picture indicating the cathode.

3.3. Equipment

The equipment needed for this project are listed below.

- Arduino pack
- Battery pack
- IRF520 Power MOSFET
- Diode
- DC motor

3.4. Important Safety Considerations

- Transistors dissipate heat and they might get very hot when ratings are exceeded. Please exercise caution with these circuits. If things are not working as expected, immediately disconnect the power source.
- The motor is powered not from the microcontroller, but from an external battery pack. This is since, especially when driving a load, the power required to turn the motor may

draw more current than the Arduino can provide. To protect the microcontroller and provide enough power to the motor, it is best practice to power the motor externally.

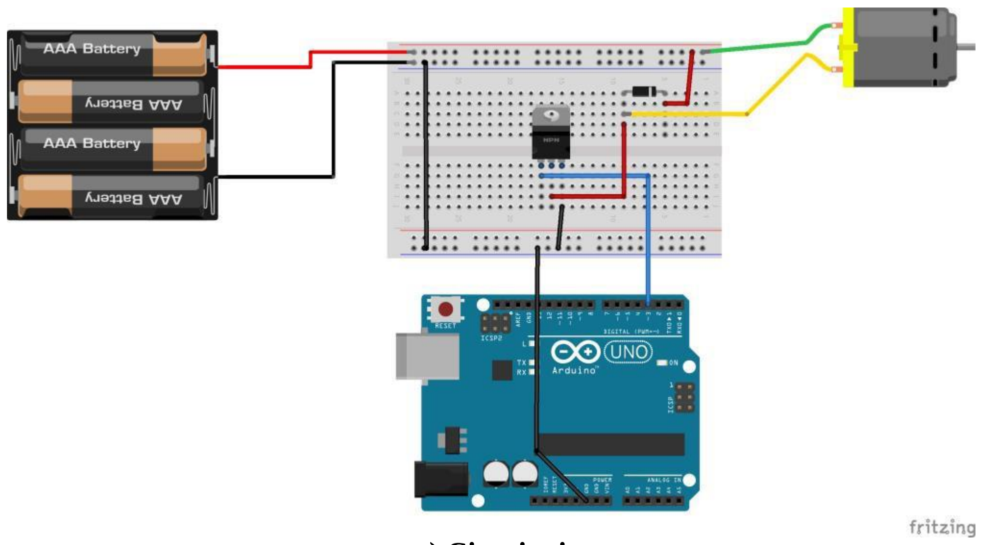
- Carefully note how the diode is connected. Note that the diode here **PREVENTS** any drain source connection when the transistor is off, protecting the transistor.
- Carefully note the transistor pin arrangement.

3.5. Breadboard and circuit schematic

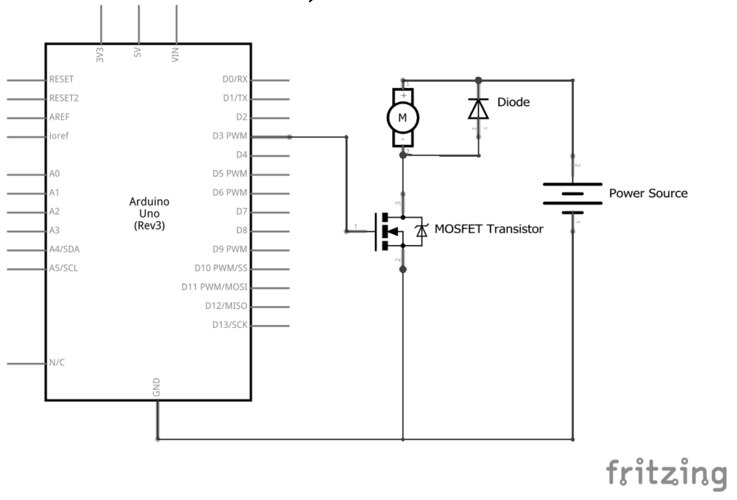
The breadboard and circuit schematic are shown below in Figure 5. Please be careful when building the circuit.

3.6. Program Concept

The concept is similar to the LED blink project of Lab 1. Power digital pin 3 on and off with delays of 2 seconds in between to turn the motor on and off. You can also experiment with switching the polarity of the motor to make it turn in the opposite direction.



a) Circuit view



b) Schematic view

Figure 5- Motor switch circuit and schematic

4. Project Two: Turn a DC motor on and off using a pushbutton

The objective is to demonstrate how a DC motor can be turned on at the press of a button. The circuit and programming concept for push button controlled elements were studied in Lab 1. This project builds on that information to control a DC motor.

5. Project Three: Controlling the speed of a DC motor using a potentiometer

The objective of this project is to control the speed of rotation of a DC motor with a potentiometer. The motor turns faster or slower with the turning of a potentiometer knob.

5.1. Pulse Width Modulation

In general, with a constant load on the motor shaft, changing the voltage input to the motor changes its speed of rotation. The battery pack in the lab however does not produce a time varying voltage; neither do other DC power sources. Also, microcontrollers are digital devices producing signals that are only on and off or maximum voltage and zero.

To overcome the issue of creating a signal that has a value between zero and maximum, a technique known as pulse width modulation (PWM) is employed. A PWM signal is a square-wave signal with fixed amplitude and frequency, but the width (time) of the on and off portions can be varied. A square wave PWM signal is shown in Figure 6. As shown in the figure, the duty cycle of a PWM signal represents the percentage of time the signal is on.

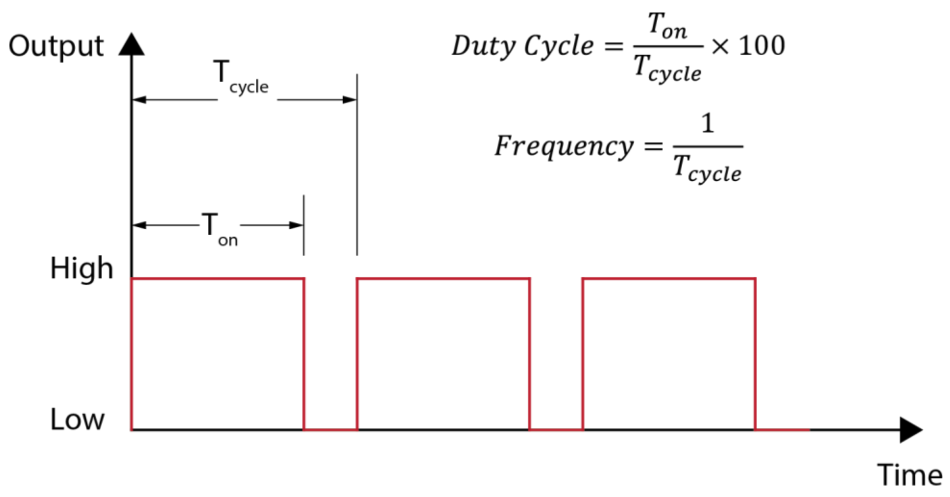

Output
T_{cycle}
$Duty\ Cycle = \frac{T_{on}}{T_{cycle}} \times 100$
$Frequency = \frac{1}{T_{cycle}}$
T_{on}
High
Low
Time

Figure 6- PWM Signal

Many microcontrollers, including the one in the lab, have PWM modules built into them which can be called programmatically to generate a PWM signal. Normally the period of the pulse of the PWM signal is much smaller than the time needed by the motor (or other loads controlled by the PWM) to react. Therefore, the motor does not respond to the each on and off cycle of the PWM signal, but to the average value of the PWM signal.

On the Arduino UNO, digital pins 3,5,6 and 9,10,11 can produce a PWM output. These pins are marked by a ~ sign next to them. The command to generate a PWM signal in the Arduino IDE is:

```
analogWrite(pin, value)
```

Where “pin” is the digital pin from which the PWM signal is generated and “value” is a number between 0 and 255, corresponding to the duty cycle of the signal. A value of 255 creates a 100% duty cycle, where the signal is always on, a value of 127 generates a 50% duty cycle, where the signal is on 50% of the time and off 50% of the time and a value of 0 creates a signal that is always off. It is not required to call `pinMode()` to set the pin as an output before calling `analogWrite()`.

5.2. Equipment

Same as Project One with the addition of a potentiometer.

5.3. Breadboard and schematic circuit view

Same as Project One, with the addition of a potentiometer.

5.4. Program Concept

The concept combines the potentiometer experiment from Lab 1 and the PWM concept here. The analog potentiometer voltage reading must be converted to a value between 0 and 255 which becomes the duty cycle of the PWM signal. Applying this PWM signal to the transistor gate controls the rate at which the gate opens and closes, thereby the rate at which the motor is powered on and off, therefore creating the desired speed.

You can use the map function y=map(x, low1,high1, low2, high2) which maps a number x from the range [low1,high1] to the range [low2,high2] and stores it in y to map analog readings of the potentiometer to the duty cycle of the PWM signal.

6. Project Four: Half-step driving a stepper motor

The objective of this project is to drive a stepper motor using the half-step drive mode. The stepper motor will turn continuously using the half-step drive method.

6.1. Stepper motor

A stepper motor is a motor that can move in small, angular increments or steps. Stepper motors are conveniently driven from a digital source since they operate with digital pulses. They are also popular in position control applications without position measurement since the nominal position of the motor can be controlled by the number of pulse signals sent to the motor. One application of stepper motors is in positioning tables.
The stepper motor in the lab is the cheap and popular 28BYJ-48, shown along with the ULN2003 breakout board in Figure 7. The remainder of this section focuses on issues specific to this motor.

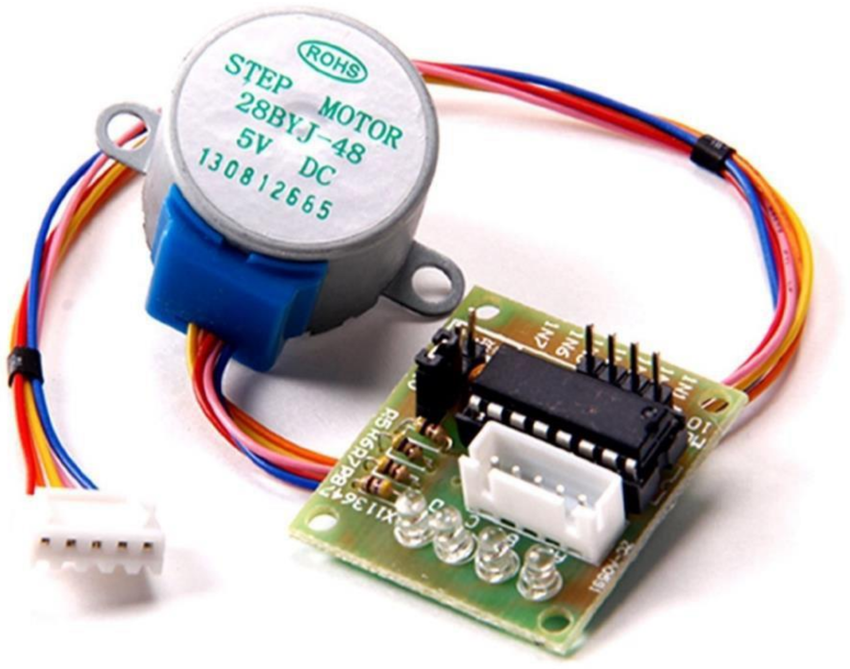


Figure 7- The geared stepper motor in the ENME 461 lab

6.2. Motor specifications

This motor has four coil windings spread around a permanent magnet rotor as shown schematically in Figure 8. These four coils are connected to the Blue, Pink, Yellow and Orange leads coming out of the motor. The Red cable is connected to the common tap.

This arrangement creates a ***four-phase*** stepper motor where phase refers to a separately activated coil winding.

This motor is also ***unipolar***, indicating it has a common centre tap connected to the red wire and that the polarity of the phases does not change. The other characteristics of this stepper motor are listed in Table 1, as provided by the manufacturer.

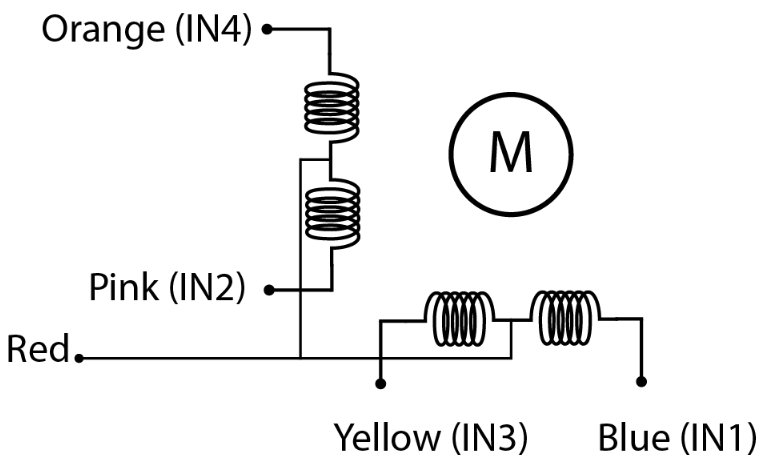


Figure 8- Stepper motor schematic

Table 1- Characteristics of the 28BYJ-48 Stepper Motor

Motor type	Unipolar stepper
Connection	5 lead connection
Voltage	5-12V DC
Frequency	100Hz
Drive mode	Half step recommended
Step Angle (not accounting for internal gears)	5.625° in half step drive mode
Gear ratio	63.68: 1

6.3. Drive methods

Each step of rotation in a stepper motor is the result of an input voltage pulse. The stepwise rotation of the stepper is the result of collective pulses in a pulse train. The motion of a stepper motor therefore depends on how the coils (phases) are activated using the pulse train. There are four possible ways of driving a stepper motor:

- Wave Drive actuates each of the coils in turn. The pattern for the motor in the labs is shown in Table 2. **If this motor were non-geared**, due to the arrangement of the coils, wave drive would accomplish a full 360 degree rotation of the motor shaft in 32 steps.

Table 2- Wave drive actuation pattern

Lead wire color	**Steps**			
	Step 1	**Step 2**	**Step 3**	**Step 4**
Blue	ON	OFF	OFF	OFF
Pink	OFF	ON	OFF	OFF
Yellow	OFF	OFF	ON	OFF
Orange	OFF	OFF	OFF	ON

- Half Step drive alternates between activating one phase and two phases at a time. The pattern for the 28BYJ-48 Stepper is shown in Table 3. If this motor were non-geared, this drive method would accomplish a full rotation (360 degrees) of the motor shaft in 64 steps.

Table 3- Half step drive actuation pattern

Lead wire color	**Steps**							
	Step 1	**Step 2**	**Step 3**	**Step 4**	**Step 5**	**Step 6**	**Step 7**	**Step 8**
Blue	ON	ON	OFF	OFF	OFF	OFF	OFF	ON
Pink	OFF	ON	ON	ON	OFF	OFF	OFF	OFF
Yellow	OFF	OFF	OFF	ON	ON	ON	OFF	OFF
Orange	OFF	OFF	OFF	OFF	OFF	ON	ON	ON

- Full Step drive activates two phases of the motor at any given time.
- Microstepping drive varies the current applied to each phase in small steps in a sine wave pattern, resulting in a smooth rotation compared to the other techniques.

The 28BYJ-48's manufacturer recommends driving it in the half step mode where it has a stride angle of 5.625° per step, indicating a full rotation is equivalent to 64 steps.

It is important to note however that, as mentioned in the recommended video, this motor is geared down internally using a combination of gears with an overall ratio of 63.68: 1, creating a much higher resolution of rotation.

In the first three drive methods, reversing the order in which the coils are activated will reverse the direction of rotation.

6.4. ULN2003 breakout board

The board that is sold with the 28BYJ-48 includes a ULN2003 transistor array chip to amplify signals from the microcontroller. This board incorporates four LEDs (A,B,C,D) which turn on to indicate when each of the four coils (Blue, Pink, Yellow, Orange) is actuated.

Board connections are straightforward. IN1, IN2, IN3, IN4 represent Blue, Pink, Yellow, Orange coils, respectively. The (-) and (+) connections represent where the battery pack is connected. We also establish a common ground between the Arduino and the motor using the breadboard.

6.5. Equipment

- 28BYJ-48 Stepper motor
- ULN2003 Board
- Female/female and male/female jumper wires for power
- Battery pack

6.6. Breadboard view

The circuit breadboard view is shown in Figure 9. IN1, IN2, IN3, IN4 are connected to digital pins 2,3,4 and 5, respectively. The breadboard creates a common ground between the Arduino and the motor circuit.

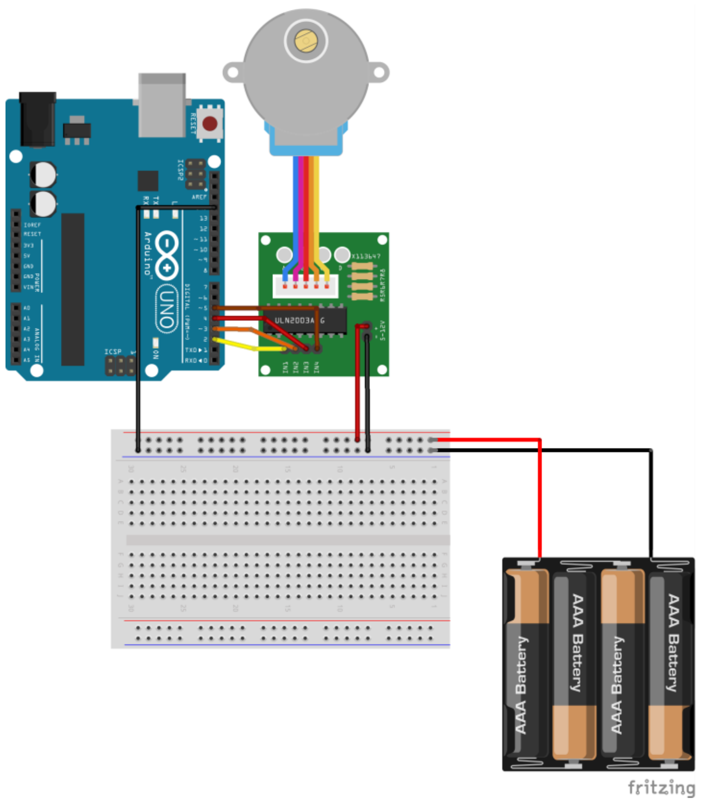


Figure 9- Breadboard view of the stepper motor circuit

6.7. Program concept

This program first sets the digital pins 2,3,4, 5 to be outputs as was done in Lab 1. In the loop, the pattern of Table 3 is recreated, turning digital pins 2,3,4 and 5 HIGH or LOW according to the steps, activating the appropriate coils.

A main consideration is to balance the execution time of the program with the response time of the stepper motor to create a smooth motion. If the program is executed at a very fast rate with little or no delay after each step, the motor will not have time to respond. If the program is slowed down using a larger delay after each step, the motor rotation will be painstakingly slow. Experiment with a delay time inserted between the steps (of Table 3) to find an appropriate number. Small numbers in the millisecond range should work best.

In a rudimentary programming approach, the loop can explicitly set the state of each pin at any point. A more elegant programming solution could use an array or a dedicated function separate from the loop function.

7. Project Five: Bouncing the stepper motor

The objective is to have the stepper motor rotate one full turn in the clockwise direction, then reverse direction and rotate a full turn in the counter-clockwise direction and continue in this fashion. To create the program, use the information in Table 1 to calculate the number of steps required for a full turn of the motor shaft. Then modify the program from Section 6 to turn the motor in the counter-clockwise direction.

8. References

Arduino References. (2017). Retrieved from https://www.arduino.cc/en/Reference/Array

Jouaneh, M. (2013). *Fundamentals of Mechatronics.* Stamford: CEngage Learning.

Stan. (2014, March). *28BYJ-48 Stepper Motor with ULN2003 driver and Arduino Uno*. Retrieved from 42 Bots: http://42bots.com/tutorials/28byj-48-stepper-motor-with-uln2003-driver-and-arduino-uno/

Timothy, H. (n.d.). *PWM*. Retrieved from https://www.arduino.cc/en/Tutorial/PWM