ME 220 LAB: Simple DC Motor Control with L293D using Arduino – *Part I*

Objective:
The objective of this lab is to connect your Arduino to L293D IC and eventually drive a small size permanent magnet (PM) direct current (DC) motor. In the remainder of the text, the word motor will be used to refer to a PM DC motor.

Pre-Lab:

The pin layout of L293(D) is given in figure 1.a. As mentioned in the data sheet, L293 is a quadruple half H-drive.

With a half H-drive (or half-bridge) you can control an inductive load, but only in one direction. In other words, you can control the flow of current over an inductive load in one direction, i.e. you can switch the current ON and OFF using a half bridge. If the inductive load is a DC motor, you can turn it ON or OFF at will. By controlling how fast you switch the current (i.e. by controlling the switching frequency), you can control the speed of the motor. Hence, you can control the speed control of 4 motors with a single L293 IC, but you cannot change their direction. This also implies that you can control 4 relays, most of the stepper motors or any similar inductive load with by using these 4 half bridges.
Simple math tells us that 4 half bridges should make 2 full bridges, where a full bridge can control the current over an inductive load in both directions! Hence, using two full bridges, you can interface two motors to L293D and control both the speed and the direction of rotation of them. The pin layout of the IC is organized nicely so that each half bridge is located towards opposite corners if the plastic package. Half bridges on the same side are connected through a common enable pin by the use of which you can create a full-bridge if needed. The functional table that is given in figure 1.b. If the bridges are enabled, the logic value at the input pins (A) are reflected to the corresponding output pins (Y). In case the pins are disabled (by pulling the enable pin to low), the outputs are inhibited.

L293 and L293D do have the same footprint, however, L293 has an output current of 1A per channel whereas this value is 600mA for L293D. The 'D' indicates that *flyback diodes* are included in the package; therefore, you do not need to add them to your circuit. *Flyback diodes* help the system to dissipate current without harming the rest of the system in case a load is instantaneously disconnected. Such a case is illustrated in Figure 2.

For small-scale robotic implementations, L293 is quite practical and low-cost. To drive loads that demand greater amounts of currents, you can combine bridges. One common way to do is to solder two ICs on top of each other. However, if you need to go over 1-1.5 A, then consider a proper driver designed for such loads since cooling might easily become your major problem if you use L293. If you push your L293 too much, cooling might become an issue, in this case check the datasheet, find out proper pins, and find a creative way to dissipate heat through these pins.

On the practical side and check out the application information given in figure 3. Use of both half- and full-bridges are illustrated in this example to drive motors in either unidirectional or bidirectional fashion. As mentioned above, the flyback diodes are not necessary along with L293D since they are present inside this IC.

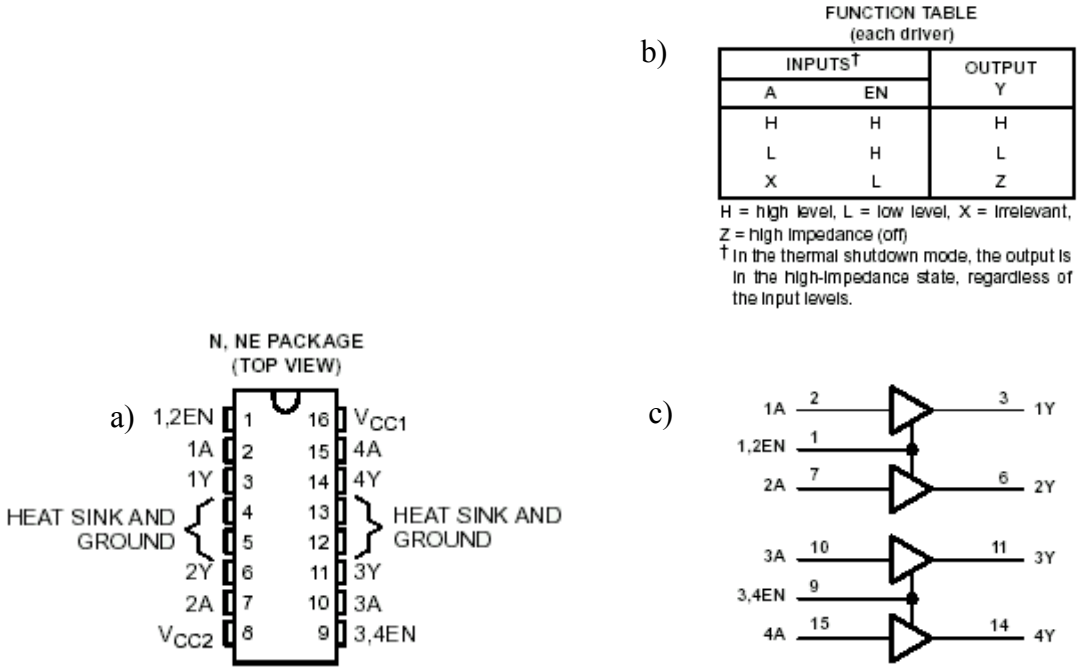

Figure 1. L293(D) a) Pin Layout b) function table c) logical diagram

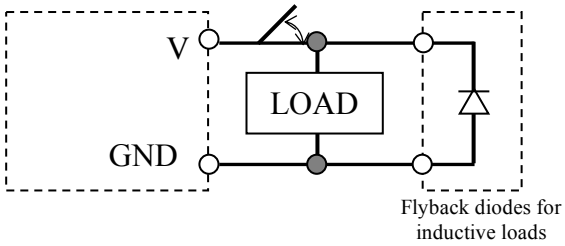

Figure 2. Use of flyback diode with inductive loads

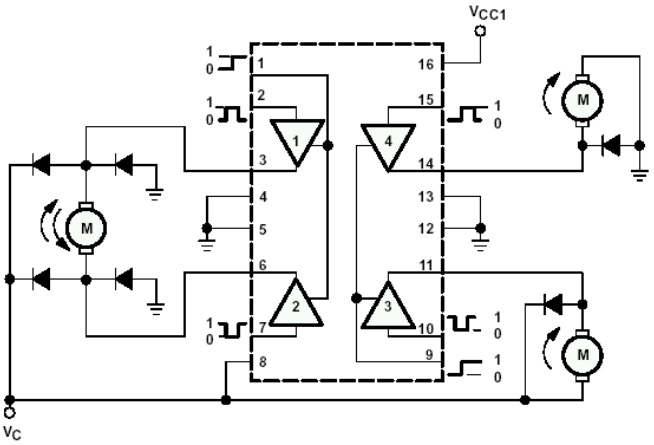

Figure 3. Possible use of half- and full-bridges.

ME 220 LAB: Simple DC Motor Control with L293D using Arduino

Lab:

1. In the first part of the lab, you will build a so called, LED Motor! A LED motor is simply composed of two LEDs and a 1K resistor. It is good practice if you solder one and keep it handy for DC motor applications. Figure 4 illustrates the schematic of a LED motor.

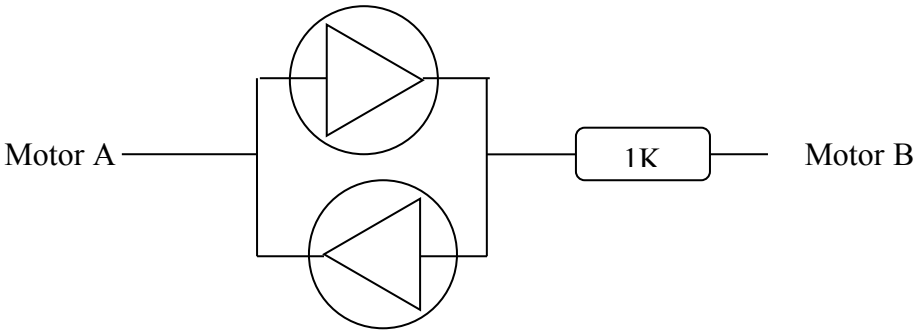

Figure 4. Schematic of a LED motor.

Prepare two LED motors. Once you are done with the LED motors, let us inspect it. Then you can move on to the next step.

2. At this stage you will handle two interrupts on Arduino. Now, connect two push buttons to digital input pins 2 & 3 as illustrated in figure 5.

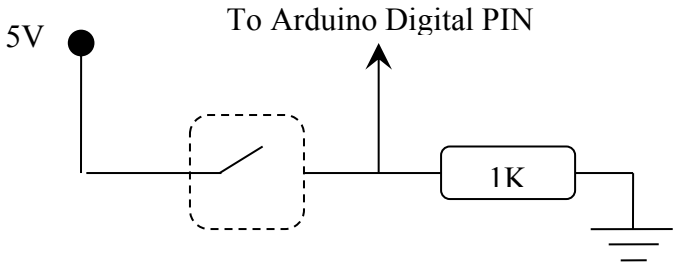

Figure 5. Interfacing a button / switch to Arduino to provide digital input.

Given this configuration when the button is pressed, Arduino is going to read HIGH and it will read LOW at other times.

3. Now make sure that you can handle digital pin interrupts. Read the Arduino help on interrupts. Write interrupt handlers for pins (2 & 3 on UNO) such that when you press the button, Arduino sends "Button X is pressed" and sends "Button X is released" when you press and release the buttons where X is the PIN number on which the press or release is sensed. Once you are done, let us inspect you code before you proceed to the next step.
Hint: Go to Arduino UNO product page and checkout the interrupt pins.

4. At this step you are to build your driving circuitry on the breadboard using L293D and you will interface it to Arduino as shown in figure 5. Instead of a real motor, for the time being interface you LED motor to L293D. You will be allowed to use a real DC motor only after we inspect that you can successfully control the LED motor (shown as M1 in Figure 6).

Pay attention to the power supplies (Figure 6). You can use the +5V from Arduino for pin 16 on L293D, whereas Motor Power Supply on pin 8 should be externally provided to the circuitry where the ground of the external supply is connected to the ground of the remaining system when you use an electric motor. For the sake of LED motor you can use 5V from Arduino but once you use a real motor attach a proper power supply to pin 8.

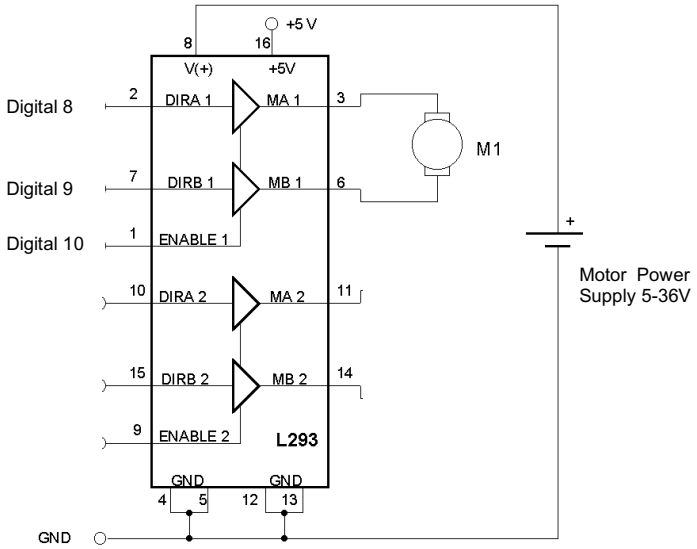

Figure 6. L293D interfacing to Arduino.

5. Finally, write a program that acts according to the truth table given in table 1. Observe how the motor acts when you play with buttons and try to relate this behavior of the circuitry with the pre-lab reading.

Table 1: Truth table to be implemented on Arduino

Inputs		Outputs		
Digital 2	Digital 3	Digital 10	Digital 9	Digital 8
0	X	L	L	L
1	0	H	L	H
1	1	H	H	L

Continue to the next part for a bit more programming and use real DC motors!

ME 220 LAB: Simple DC Motor Control with L293D using Arduino – *Part II*

Pre-Lab:

Now that you can control the direction of the LED Motor, now let's try to change the speed of the motor.

Changing the speed of the motor can be achieved by changing the voltage applied to the motor. Hence, an analog voltage can be applied to a DC electric motor in order to change its velocity. However, this is not preferred for reasons such as efficiency, and the need for a digital to analog converter (DAC) on the microcontroller. However, most of the popular microcontrollers do not come with a DAC built into them.

An alternative method for controlling the speed of a motor is achieved by applying a pulse width modulated (PWM) signal to the motor. A PWM signal is a square wave, where in common practice the frequency of the wave is fixed and duty cycle (DC) of the signal is changed in order to change. Some sample PWM signals with various DC values are illustrated in figure 7.

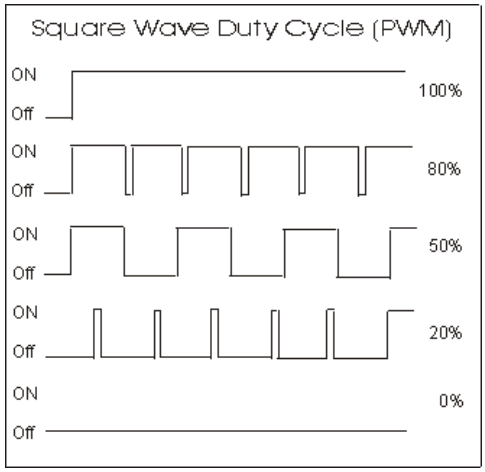

Figure 7. PWM wave forms with different duty cycles.

It is seen that, as DC increases, duration of full supply voltage applied to the motor will increase. Hence, in terms of the resultant effects on the motor, we might say that, increasing DC in a PWM signal will have a similar affect to increasing the amplitude of an analog signal.

Arduino Uno has more than a few PWM pins. The frequency of the PWM signal is constant and preset to 490 Hz. The duty cycle of a PWM pin is controlled by calling the *analogWrite(pin, value)* command.

Lab:

1. In this part of the lab, you will change the speed of the LED motor. Well, in this scenario, speed corresponds to light intensity on the LED. The higher the PWM DC, the brighter the LED will become in the selected direction.

You will continue using the truth table and the already connected buttons from Part I of this lab. However, in this case, Digital 10 pin will be configured as PWM output. You will be sending a proper DC from the serial monitor and observe that brightness of LED changes

2. Now, you are to develop your fully functional communication protocol. In other words, you have to control the speed and direction of each motor than can be connected to L293D from the serial monitor.

The protocol you will come up with should enable users to do the following by sending a single set of commands through the serial port:

- Apply a certain DC value to a specified motor
- Apply different DC values to each motor
- Stop an individual motor
- Stop all motors
- Change direction of a motor
- Change direction of both motors

Once you implement this, show us that it works on the LED motors. Evidently, you have to connect two LED motors to your Arduino. It is a similar process to the one shown in Figure 6.

3. Finally you will be able to connect a real motor to your circuit. Keep the LED motor on one side of the circuit and attach the real motor to the other side of L293D. Using your communication protocol, test different DC values for the motor.
IMPORTANT NOTE: At this stage before connecting the real motor to the circuit, make sure that Pin 8 of L293D is connected to an external power supply as mentioned in figure 6.

Starting from zero DC, if you keep increasing the DC value with small increments (i.e. 3-5% at a time) you will observe that there is a certain DC value below which the motor does not start turning.

Min Duty Cycle to start: ____%

If you apply the same procedure the other way around, (i.e. start from 100% and count down) you should find a similar DC value before the motor stops.

Min Duty Cycle to stop: ____%

Compare start and stop values and comment on their relative values.