University of California, Riverside
Mechanical Engineering Department
ME133: Introduction to Mechatronics

Dr. Fabio Pasqualetti TA: Gianluca Bianchin

Lab 7: March 7, 2018

Lab Report Due on March 13, 2018

OBJECTIVES:

1. Control the speed of a DC motor with Arduino.
2. Use the Arduino S/W to write simple program to change the speed of a DC motor with a PMW.
3. Write a program that increases or decreases the velocity of a DC motor based on the readings from a potentiometer.
4. Create a circuit and write an Arduino program to estimate the velocity of a DC motor using an encoder.

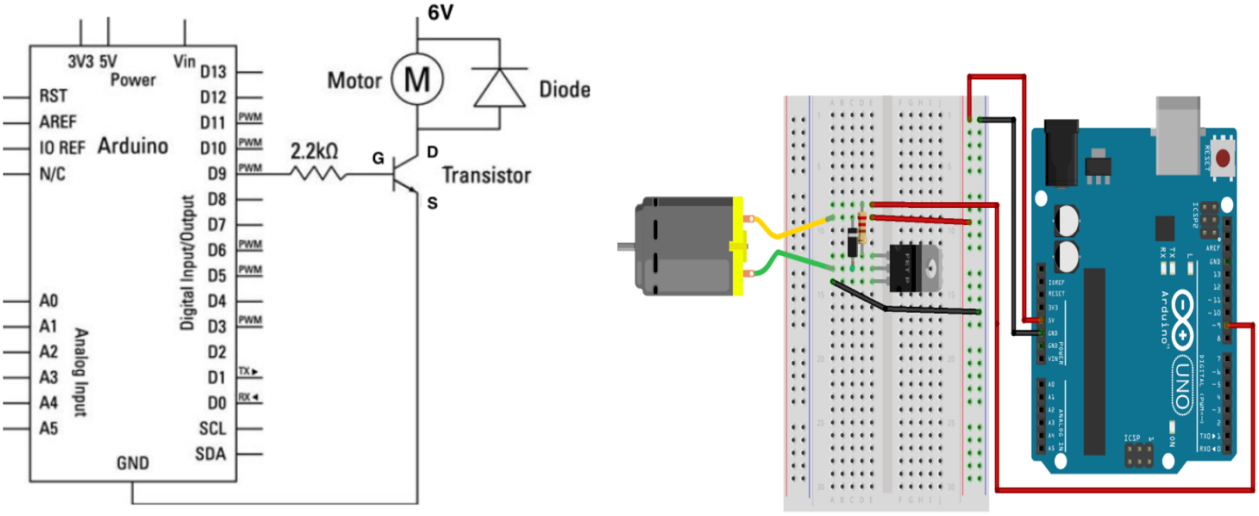
BILL OF MATERIALS:

1x	Arduino UNO
1x	PN2222 Transistor (IRF620 ?)
1x	1N4001 diode
1x	20K Ohm Potentiometer
2x	Pushbuttons
1x	220Ohm Ohm Resistor
2x	10K Ohm Resistor
1x	DC motor

MOTOR SPEED CONTROL:

Consider the circuit shown below to control a DC motor in a single direction. Most DC motors require current greater than 250mA. The maximum current of Arduino is 150mA. If motors are directly connected to the output of any pin, this can damage both the pin and the motor. To prevent damage, you need circuitry that acts as a bridge between your microcontroller and the motor. Some possibilities are transistors, H-bridges, and relays. In this lab we will use a transistor as shown below.

The diode provides a safe path for the inductive kickback of the motor. If the current in the inductor is suddenly switched off, the diode will briefly supply the voltage necessary to keep the current flowing in the short term.



When you put together the breadboard, there are two things to look out for.

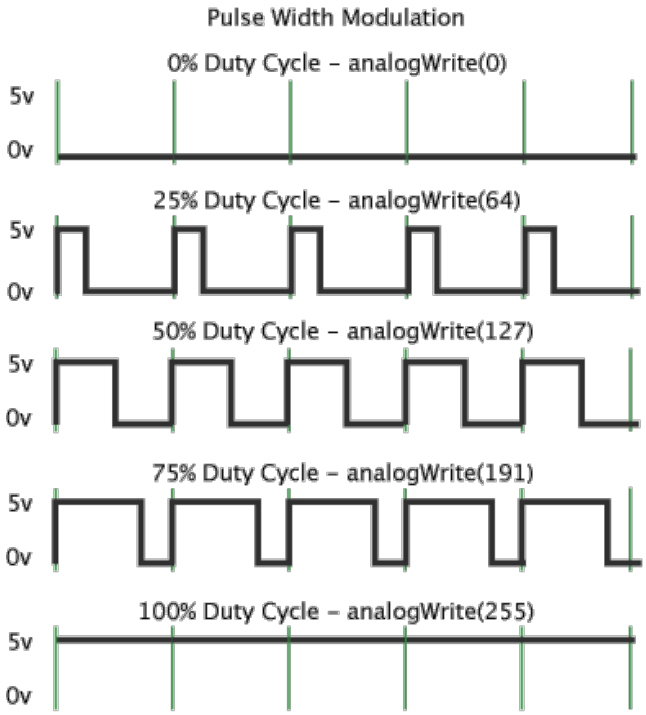
- Make sure that the transistor is the right way around.
- The striped end of the diode should be towards the +5V power line

The transistor acts like a switch that controls the power to the motor.
Arduino pin D9 is used to turn the transistor on and off.

SPEED CONTROL USING PWM

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED.

In the graphic below, the green lines represent a regular time period. This duration or period is the inverse of the PWM frequency. In other words, with Arduino's PWM frequency at about 500Hz, the green lines would measure 2 milliseconds each. A call to analogWrite() is on a scale of 0 - 255, such that analogWrite(255) requests a 100% duty cycle (always on), and analogWrite(127) is a 50% duty cycle (on half the time) for example.



TASK 1: DC Motor Speed Control

Write Arduino code that

a) Asks the user to enter a value between 0 and 255 and returns an error if not in this range
b) Controls the speed of the motor based on the entered value.
c) Demonstrate to the TA that code is properly working

Use the pseudo-code below:

```
(initialization)   //Set pin 9 as motor control pin

void setup()
{
        //Set Motor control pin as output Pin
        //Start Serial Communications
  Serial.println("Enter a Value between 0-255");
}


void loop()
{
(if condition)    //If something is entered into the Serial Monitor
  {
        //Take the first Integer entered and set the speed variable to the entered value
        //Write the 'speed' value to motor control pin
  }
}
```

TASK2

Connect two pushbuttons to the Arduino. Wire a circuit and write a program that increases the speed of the motor when the first button is pressed and decreases the speed of the motor when the second button is pressed. Be sure that the speed remains between feasible values (0-255). Display the current value of the speed variable on the serial plotter.

a) Demonstrate to the TA that code is properly working

b) Update your code so that the user can input increments of 1

TASK 3

Connect a potentiometer to the Arduino. Write a program that continuously reads the value of the potentiometer and set the speed of the motor accordingly.
Notice that analog inputs have values between 0 - 1023, while analog outputs take values over the interval 0 - 255.

a) Demonstrate to the TA that code is properly working

TASK 4

For a fixed load, the shaft speed of a DC motor is proportional to the applied voltage. Wire a circuit and write a program that continuously estimates the speed of the motor and visualizes it for the user through the serial plotter. To change the speed of the motor use the configuration in Task 2. Make sure you use a LARGE capacitor to cancel out the slowly changing variations in your output.

a) Demonstrate to the TA that code is properly working

TASK 5

For a fixed load, the shaft speed of a DC motor is proportional to the applied voltage. Wire a circuit and write a program that

a) Continuously estimates the speed of the motor and visualizes it for the user through the serial plotter (this should be a value between 0-255).

b) Updates the control input with one component proportional to the difference between the desired speed and the actual speed, and one component proportional to the INTEGRAL of the difference between the desired speed and the actual speed.

c) Demonstrate to the TA that code is properly working

NOTE:

- To change the speed of the motor, use the configuration in Task 2.
- Make sure you use a LARGE capacitor to cancel out the slowly changing variations in your output.