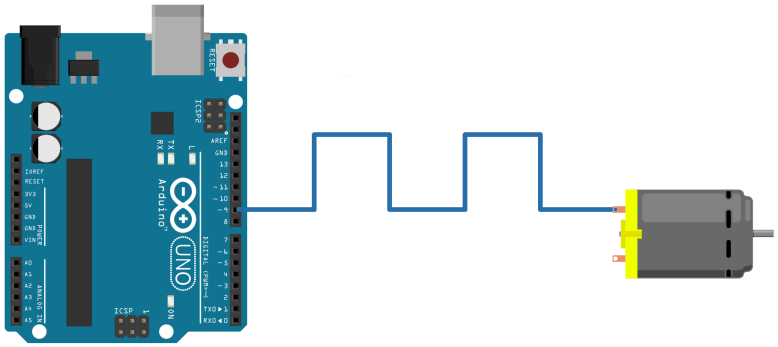


PWM & Speed Control of DC Brushed Motors

Lab 002

Electromechanical Systems: Software Interface

Professor Xiaohai R. Li
Report Written By Galib F. Rahman
EMT 2461 Section W358
October 31 2018

Table of Contents

Objective

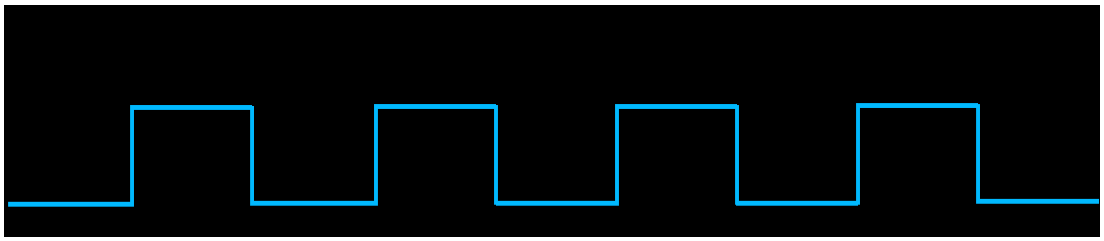
The objective of this laboratory exercise is to build an understanding of Pulse Width Modulation, often referred to as PWM - & implement a system in which a PWM signal controls the speed of a brushless DC motor.

Summary

Pulse Width Modulation (P.W.M.)

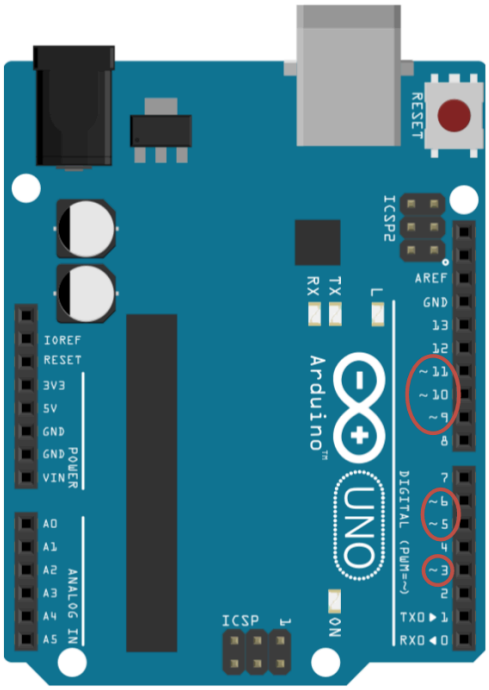
Pulse Width Modulation is essentially a digital signal in the form of a square wave, in which a series of *pulses* are transmitted at a certain frequency. The *duty cycle* of a PWM signal, refers to the percentage of how long a wave is 'high' over a period of time.

For example in the image below, the PWM signal is high half of the time and low half of the time in one period:



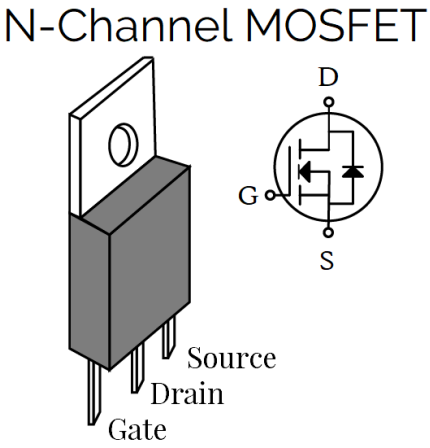
Thus one may state that the duty cycle of the PWM signal shown is 50%.

By using a PWM signal to supply power to a load, such as an LED or a DC brushless motor, we can vary the the average voltage supplied. Thus, we may vary the brightness of a LED or the speed of a brushless motor by controlling the average voltage supplied by varying the duty cycle of a PWM signal.The greater the duty cycle, the greater the average voltage supplied to a load - and vice versa. For example, if we were to supply an LED a PWM signal with a duty cycle of 100% it would remain high throughout the entire duration and be at its maximum brightness - as opposed to a duty cycle of 50% where the brightness of the LED would be half as much- in comparison. This technique of varying duty cycles enables us to control motor speeds, without the use of resistors or potentiometers, which often result in power loss via thermal energy.



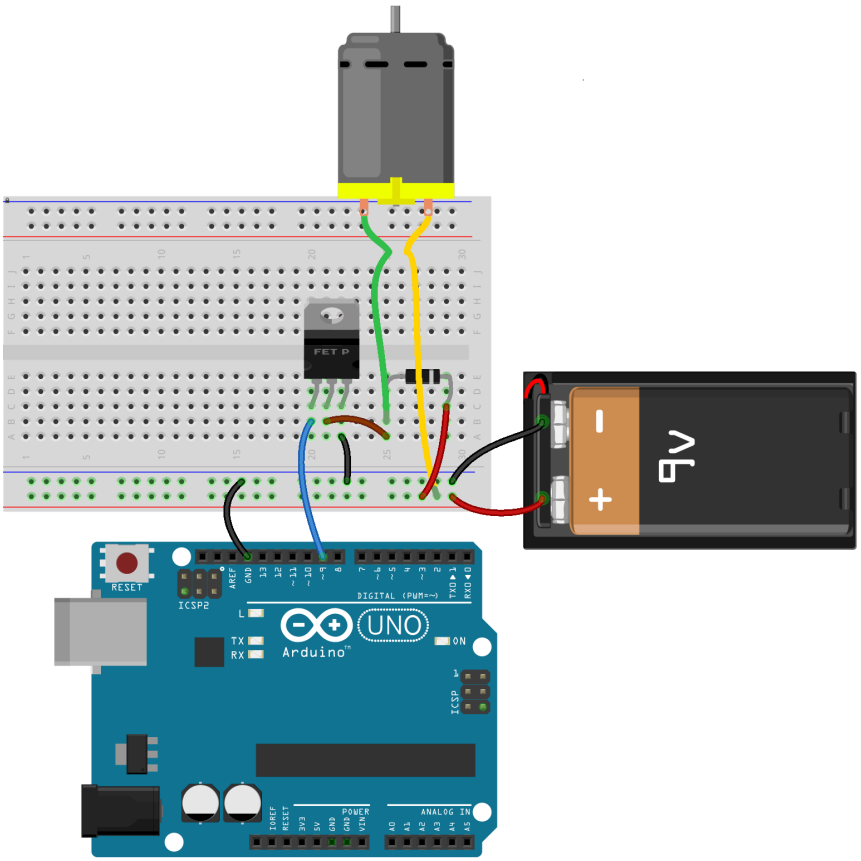
The Arduino UNO possesses the capability of providing PWM signals using select digital pins preceded by a tilde (**~**). These pins (**D3,D9,D10,D11**) output a signal at approximately 490 Hz, where as **D5** & **D6** have a frequency of 980 Hz.(See diagram left) In this lab, we used a N-channel IRF520 MOSFET to control the brushless DC motor, as opposed to the recommended NPN 2N2222 transistor alongside an external 9V power source.

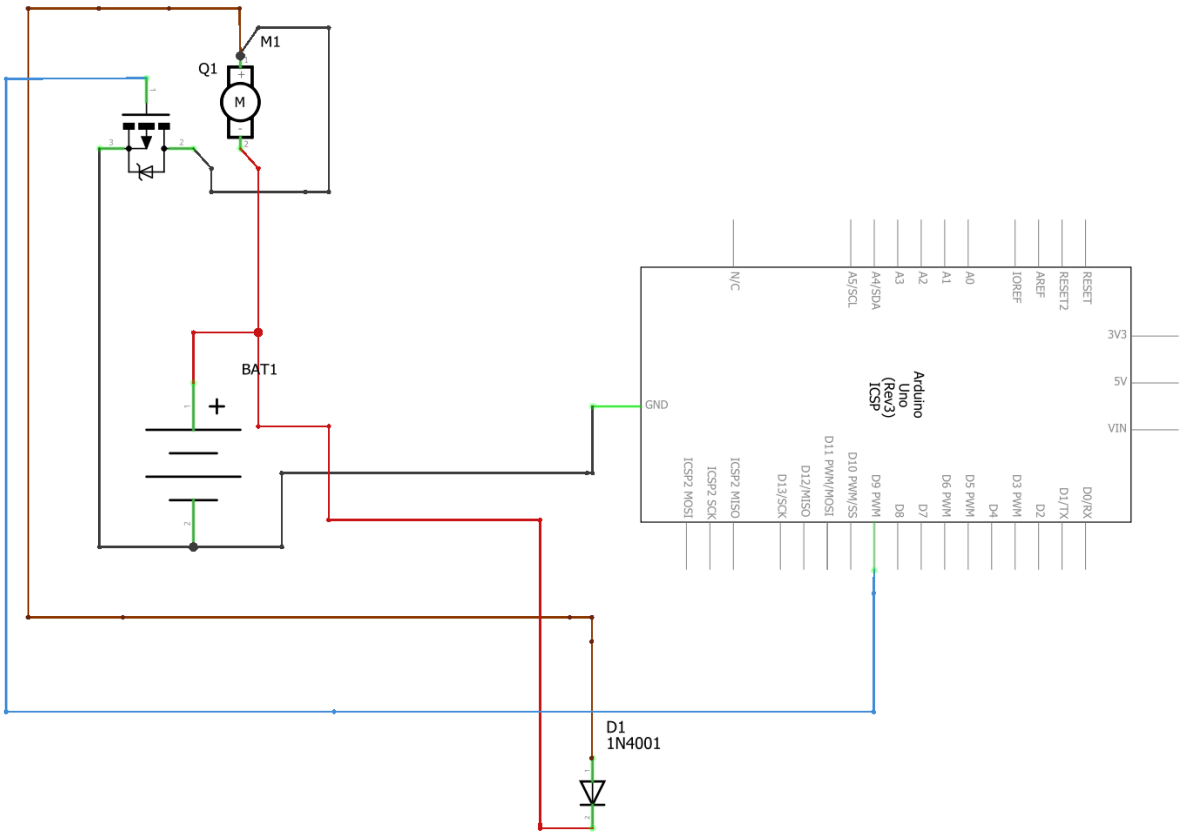
MOSFETs



MOSFETs are **M**etal **O**xide **F**ield **E**ffect **T**ransistors. The field effect refers to the voltage controlled characteristics of this electronic component, as opposed to a Bipolar Junction Transistor (BJT), which is a current controlled device. MOSFETs may be categorized into two modes; Depletion and Enhancement. Depletion mode MOSFETs act like a normally closed switch, where current flows when no voltage is applied. However, a negative applied voltage will stop the current flow and behave as an open switch. An Enhancement mode MOSFET, on the other hand, behaves similarly to a variable resistor. Enhancement mode MOSFETs can be placed into two types; N-Channel & P-Channel. The pinout of the MOSFET are identified as the Gate,Drain, and Source (as shown right). When a voltage is applied between the gate and source, current is allowed to flow between the drain and source. Like a variable resistor, the voltage applied varies the resistance between the drain and source. The lower the voltage, applied to the gate, the higher the resistance.When the voltage is increased and reaches the threshold voltage, the resistance decreases at an accelerated rate. This resistance, between the drain and source is referred to as the $R_{DS}{}^{ON}$ (stated in the datasheet), which was 0.27 Ω for the IRF520 MOSFET, used in the constructed circuit, in this exercise.

Circuit Diagram & Schematic





Source Code

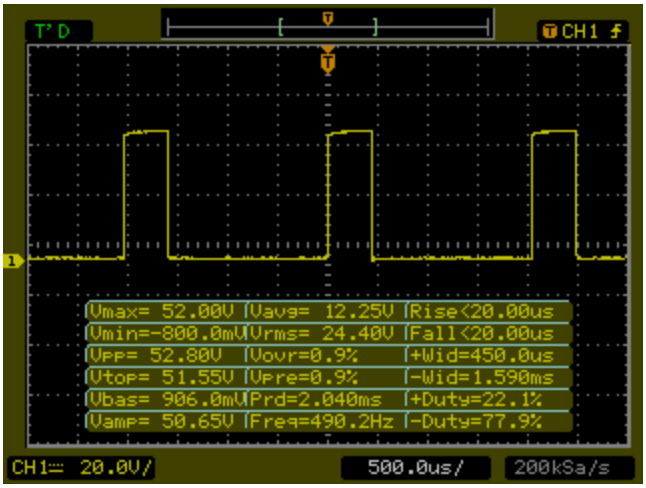
```
/*
***PWM and Speed Control of DC Brushed Motors
***EMT 2461 | Professor X.Li | Lab 003 v.0.1a
***Date: October 14 2018 Sunday 9:00 AM
***Program Developed by Galib F. Rahman
*/


int motorPin = 9;   //PWM Pin 9
int motorSpeed;    //Motor Speed for Brush Motor
char setSpeed;     //
void setup() {
 Serial.begin(9600);

}

void loop() {
  if (Serial.available()){
     char setSpeed=Serial.read();
     if(setSpeed>='0' && setSpeed<='9'){
      int motorSpeed=  map(setSpeed,'0','9',0,255);
      analogWrite(motorPin,motorSpeed);
      Serial.println("");
      Serial.print("Motor Speed is ");
      Serial.print(motorSpeed);

     }

    else{
     Serial.print("Unexpected Input ");
     Serial.println(setSpeed);


   }
  }

}
```

Code Description

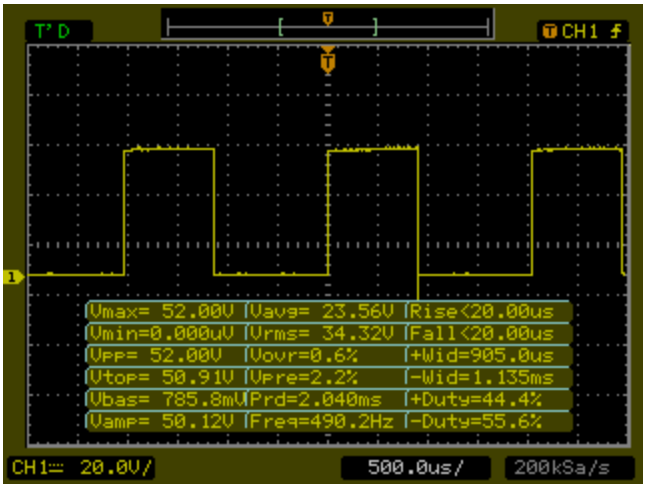
In the code shown above, we declared the PWM enabled pin 9 as motorPin, which will pulse a digital signal (varying in duty cycles later throughout the program). This pin is connected to the *gate* of the N-Channel MOSFET which will enable us to control the motor's speed or the current flow between the *drain* & *source*. Using serial communication the user will input a value ranging from '0' to '9' (stored in the variable *setSpeed*), which will vary the duty cycle of the PWM wave on pin 9. To generate this PWM signal, we will use the **analogWrite()** function which takes in two parameters; ***pin*** & ***value***. The inputted values are mapped (using the **map()** function) for the range of the second parameter used in the **analogWrite** function, ***value***. The ***value*** parameter ranges from 0-255, where the duty cycle is 0% at 0 and 100% at 255.This ***value*** is stored into the variable *motorSpeed*. The serial monitor will display the data stored in the variable *motorSpeed* after user input, if it is within the range - otherwise an error message would be displayed.

Results

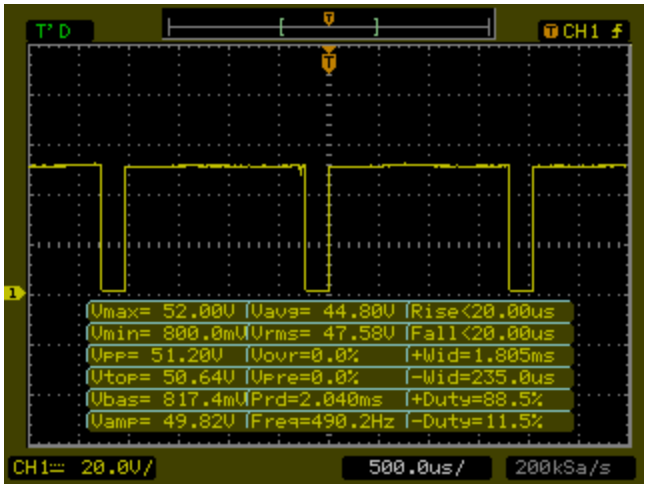
The data shown below are oscilloscope readings, displaying the PWM wave generated and it's attributes. (The duty cycle is shown as **+Duty**)



This waveform is the PWM signal generated on pin 9, when the user inputs *2*, the duty cycle reaches 22%.



This waveform is the PWM signal generated on pin 9, when the user inputs *4*, the duty cycle reaches 44.4%.



This waveform is the PWM signal generated on pin 9, when the user inputs *8*, the duty cycle reaches 88.5%.

Under ideal conditions, the following table displays the user input values and its corresponding ***value*** and calculated duty cycle.

User Input	Value	Duty Cycle
0	0	0
1	28.3	11%
2	56.6	22%
3	84.9	33%
4	113.2	44%
5	141.5	55.5%
6	169.8	66.6%
7	198.1	77.7%
8	226.4	88.8%
9	254.7	99.9%

Conclusion

In this exercise we successfully controlled a brushless DC motor's speed by using pulse width modulation (PWM). In addition we used the serial monitor as a means of providing the user access to adjust said speed and display the value passed into the **analogWrite()** function.

Diagrams & Schematics Designed

Using