Pulse Width Modulation (PWM)
Cornerstone Electronics Technology and Robotics II

- **Administration:**
  - Prayer
- **PicBasic Pro Programs Used in This Lesson:**
  - General PicBasic Pro Program Listing: http://www.cornerstonerobotics.org/picbasic.php
  - Lab 1 pwm1 as a .pdf file: http://www.cornerstonerobotics.org/code/pwm1.pdf
  - Lab 2 hpwm1 as a .pdf file: http://www.cornerstonerobotics.org/code/hpwm1.pdf
  - Lab 2 hpwm2 as a .pdf file: http://www.cornerstonerobotics.org/code/hpwm2.pdf
  - Lab 3 hpwm3 as a .pdf file: http://www.cornerstonerobotics.org/code/hpwm3.pdf
  - Lab 4 pwm_hpwm1 as a .pdf file: http://cornerstonerobotics.org/code/pwm_hpwm1.pdf

- **Pulse Width Modulation (PWM):**
  - **Introduction:** Electric motor speed is controlled by increasing and decreasing the effective voltage to the motor. More voltage and the motor rotates faster, less voltage and the motor slows down. The instinctive way to control motor speed is to connect a rheostat between the motor and the power supply to adjust the voltage, but this method has several shortcomings which will be discussed in this lesson. A better method of managing motor speed is to switch the power to the motor on and off very quickly. Rather than slowing a motor down by reducing the motor supply voltage from the full rated voltage, PWM produces digital pulses at the full motor rated voltage. Pulse Width Modulation, or PWM, is a technique for creating the series of on-off pulses. This on-off square wave pattern changes the portion of the time the signal is on versus the time that the signal is off (Figure 1).

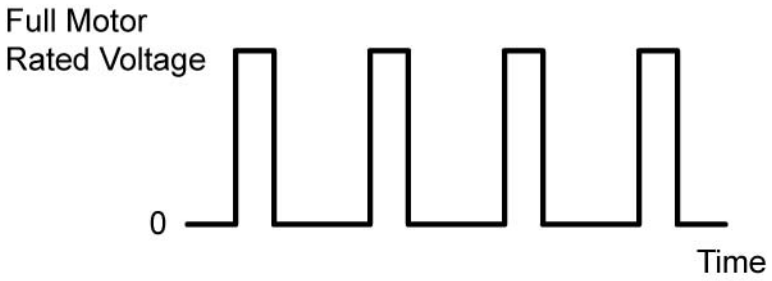


Figure 1: Sample Pulse-Width-Modulation (PWM) Signal Waveform

- **Problems Using a Rheostat to Control a Motor:** As noted above, to reduce the speed of a motor is seems natural to place a rheostat between the motor and the power supply, but this method has some inherent problems:
  - One difficulty with this method is that the voltage drop across the resistor represents energy lost in the form of heat which is a waste of valuable energy, especially from a battery power source (Figure 2).

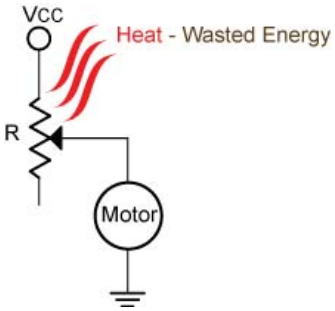


Figure 2: Motor Control Using a Rheostat (NOT Recommended)

  - Motor torque is directly related to the voltage supplied to the motor. The rheostat reduces the voltage to the motor which proportionally reduces the torque the motor delivers.
  - Another shortcoming is this – when the motor is running and you set the rheostat for a very slow speed then restart the motor, the motor may not rotate. It takes more energy to start a motor than to keep it running.
- **Duty Cycle:**
  - Definition for our purposes: The percentage of time a motor is on.
  - Formula definition of duty cycle:

Duty Cycle = 100% x Pulse Width/Period

Where:

Duty Cycle in (%)
Pulse Width = Time the signal is in the ON or high state (sec)
Period = Time of one cycle (sec)

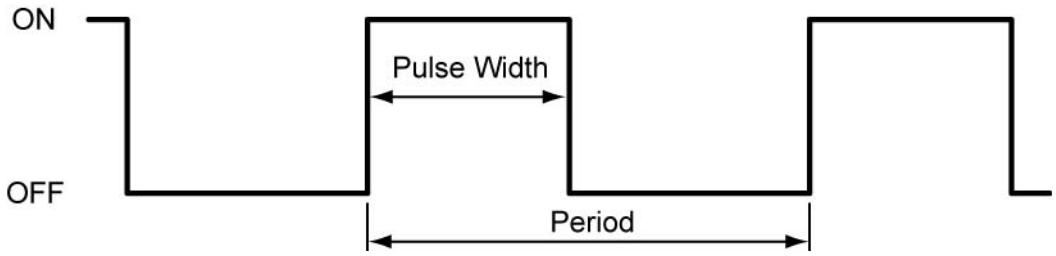


Figure 3: PWM Signal Waveform Terms

  - When the duty cycle is 0%, the load (motor) is fully off; when the duty cycle is 100%, the load is fully on.

- Examples:

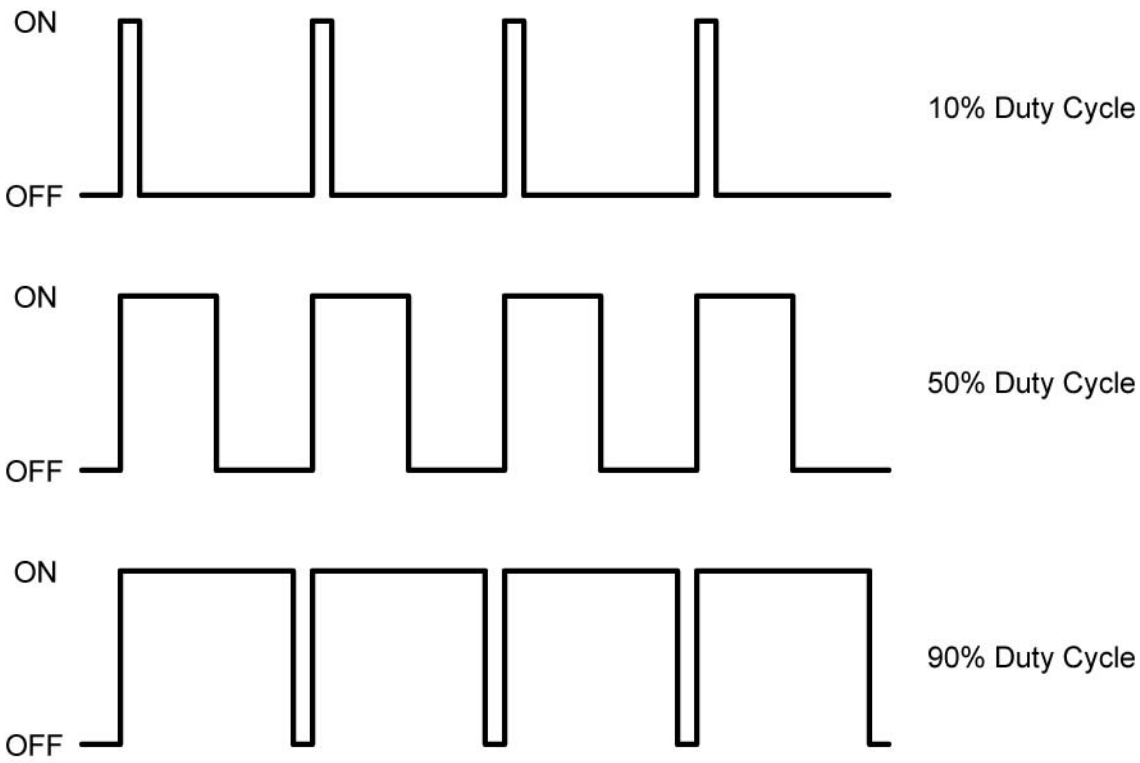


Figure 4: Examples of Several Duty Cycles

- The lesson continues on the next page.

- Example Duty Cycle Calculation: A PWM waveform is +12 volts for 40 us and 0 volts for 10 us (Figure 5). What is the duty cycle for the PWM signal?

  Period = Pulse Width + Time OFF
  Period = 40 us + 10 us
  Period = 50 us

  Duty Cycle = 100% x Pulse Width / Period
  Duty Cycle = 100% x (40 us / 50 us)
  Duty Cycle = 80%

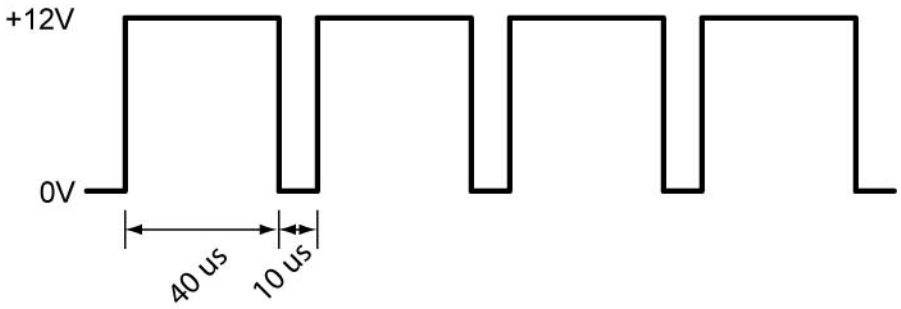


Figure 5: Duty Cycle Example Calculation

What is the frequency of this waveform?

Frequency = 1 / Period
Frequency = 1 / 50 us
Frequency = 1 / 0.00005 sec
Frequency = 20,000 Hz or 20 kHz

- Theoretical Average Voltage:
  - The actual voltage of a PWM signal depends upon the load.
  - Formula:

    Theoretical Average Voltage = $Voltage_{Full}$ x Duty Cycle

  - Example from Figure 5:

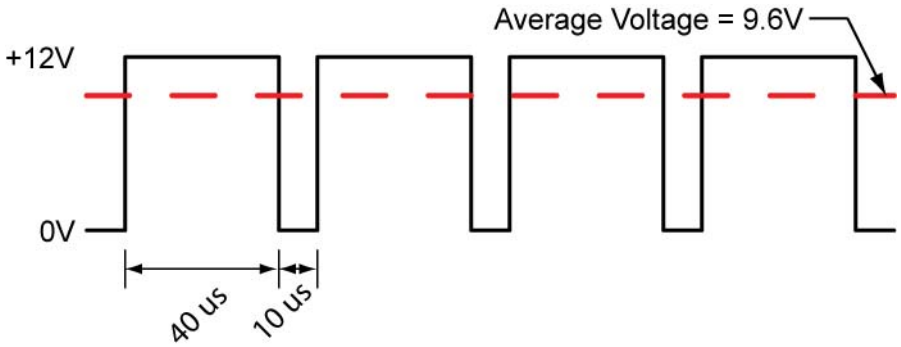


Figure 6: Average Voltage of Duty Cycle Example

Average Voltage = 12 Volts x 80%
Average Voltage = 9.6 Volts

- **Details about PWM:**
  - Since the motor is at full voltage during the ON pulse, the motor runs at full torque during the ON pulse.
  - A 25% duty cycle reduces the power applied to the motor close to 25%, yet the speed may not be reduced to 25% of the maximum speed.
  - A PWM signal is at a fixed frequency. A very low PWM frequency will make the motor rotate in a sequence of 'jerks', a slightly higher frequency will cause the motor to make an audible humming sound. The frequency of a PWM motor signal should be 20 kHz or more to ensure that the motor whine is outside the normal human hearing range (20 – 20,000 Hz). If your motor driver circuitry uses MOSFETs, the PWM frequency should never exceed the switching speed of the MOSFET.
  - Because of the weight of the motor armature and its inertia, the motor speed won't vary noticeable between each on and off pulse, provided the pulses are short enough and close enough together.
  - Pulse-Width Modulation (PWM) is a way of controlling the power of an electric circuit that wastes very little electrical energy. The signal is fed through a transistor switch which has little resistance while in the ON state. Power MOSFETs generally have a drain to source resistance in the ON-state ($R_{DS(ON)}$) measured in the hundredths of an ohm (Figure 7). Since the resistance is so small, little energy is consumed by the transistor switch while in the ON-state.

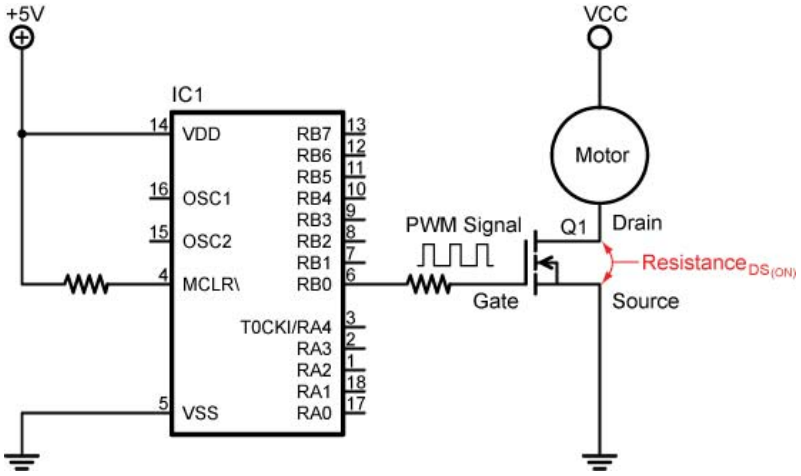


Figure 7: The Drain to Source Resistance in the ON-state is Measured in the Hundredths of an Ohm

  - Check out how your motor controller copes with forward and reverse. Some controllers use the PWM for speed and another pin to set direction (forward or reverse). Other controllers achieve it all in the one PWM signal e.g. duty cycles < 50% are reverse and > 50% are forward.

- **New PicBasic Pro Commands:**
  - **PWM**
    - Format:

      PWM *Pin*,*Duty*,*Cycle*

      Outputs a pulse width modulated pulse train on *Pin*. The *Duty* cycle for each PWM cycle ranges from 0 (0%) to 255 (100%). This PWM cycle is repeated *Cycle* times. *Pin* may be a constant, 0 - 15, or a variable that contains a number 0 – 15 (e.g. B0) or a pin name (e.g. PORTA.0).
    - The *Cycle* time of **PWM** is dependent upon the oscillator frequency. If a 4MHz oscillator is used, each *Cycle* is about 5ms long. If a 20MHz oscillator is used, each *Cycle* is about 1ms in length. Defining an OSC value has no effect on **PWM**. The *Cycle* time always changes with the actual oscillator speed.
    - If you want continuous PWM output and the PICmicro MCU has PWM hardware, **HPWM** may be used instead of **PWM**. *Pin* is made an output just prior to pulse generation and reverts to an input after generation stops. The **PWM** output on a pin looks like so much garbage, not a beautiful series of square waves. A filter of some sort is necessary to turn the signal into something useful. An RC circuit can be used as a simple D/A converter:
    - Example:

      PWM 7,127,100 ‘Sends a 50% duty cycle PWM
      ‘signal out Pin RB7 for 100 cycles
  - Perform Pulse Width Modulation LAB 1 – pwm1.pbp
  - HPWM:
    - Format:

      HPWM Channel,*Dutycycle*,Frequency

      Outputs a PWM signal using the PICs PWM hardware which is available on some PICs including the PIC16F88.
    - The default HPWM Channel 1 on the 16F88 is RB0. In the specification sheet, it is referred to as CCP1. RB3 is an alternative pin for CCP1, but the PIC must be reconfigured before this pin can be used. See the green PicBasic Pro Compiler manual by microEngineering Labs, Inc. for details.
    - The *Duty*cycle for each **HPWM** cycle ranges from 0 (0%) to 255 (100%), similar to Duty for the **PWM** command.
    - Frequency sets the desired frequency of the PWM. The lowest frequency for the PIC16F88 at an oscillator setting of 4 MHz is 245 Hz and the highest is 32,767 Hz.
    - The HPWM command runs in the background of your program, meaning it will continue to run while the program is executing other commands.
  - Complete Pulse Width Modulation LAB 2 – hpwm1.pbp
- **Motor Calibration:**
  - Since our robotic cars are equipped with differential drive, it is essential that both motors operate at the same rpm when navigating forward or backward. The dc motors must be calibrated to ensure both motors rotate at the same rpm.
  - Perform Pulse Width Modulation LAB 3 – PWM Calibration
  - Perform Motor Control Pulse Width Modulation LAB 4 – Using PWM for Speed Control with the SN754410

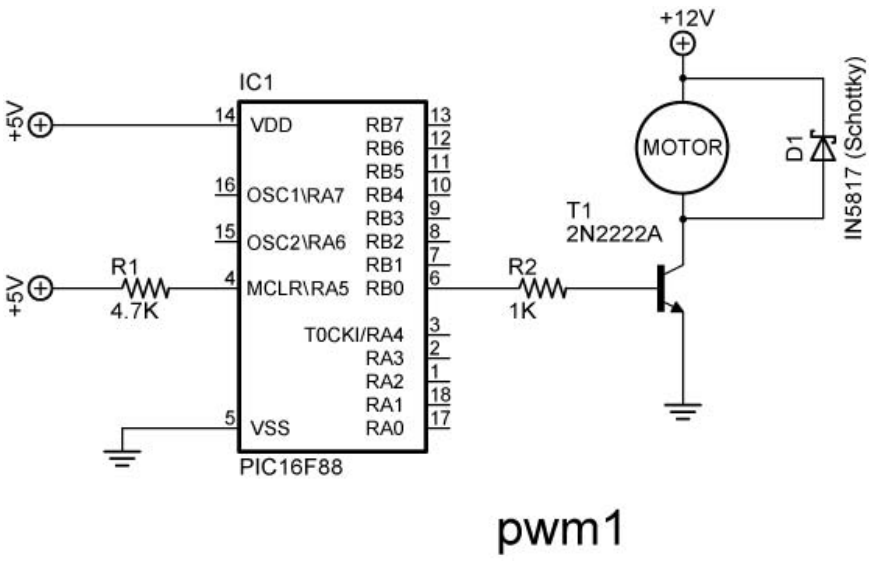
Cornerstone Electronics Technology and Robotics II
Pulse Width Modulation LAB 1 – pwm1.pbp

- **Purpose:** The purpose of this lab is to acquaint the student the PicBasic Pro command **PWM** and how to make basic connections of a motor to a PIC programmed with **PWM**.

- **Apparatus and Materials:**

    - 1 – Breadboard with +5VDC and +12VDC Power Supply
    - 1 – PIC16F88
    - 1 – 1K Resistors
    - 1 – 4.7K Resistor
    - 1 – 2N2222 NPN Transistor
    - 1 – 1N5817 Schottky Diode
    - 1 – 12 Volt DC Motor

- **Procedure:**
    - Wire the circuit below on your breadboard:
    - Open **pwm1.pbp** from your folder and download to the PIC. The **pwm1.pbp** program changes the motor speed to three levels.
    - Run the program and observe the motor speed changes
    - Now save **pwm1.pbp** as **pwm10.pbp**.
    - Experiment by changing the values of Duty first, and then Cycle in the **PWM** command.



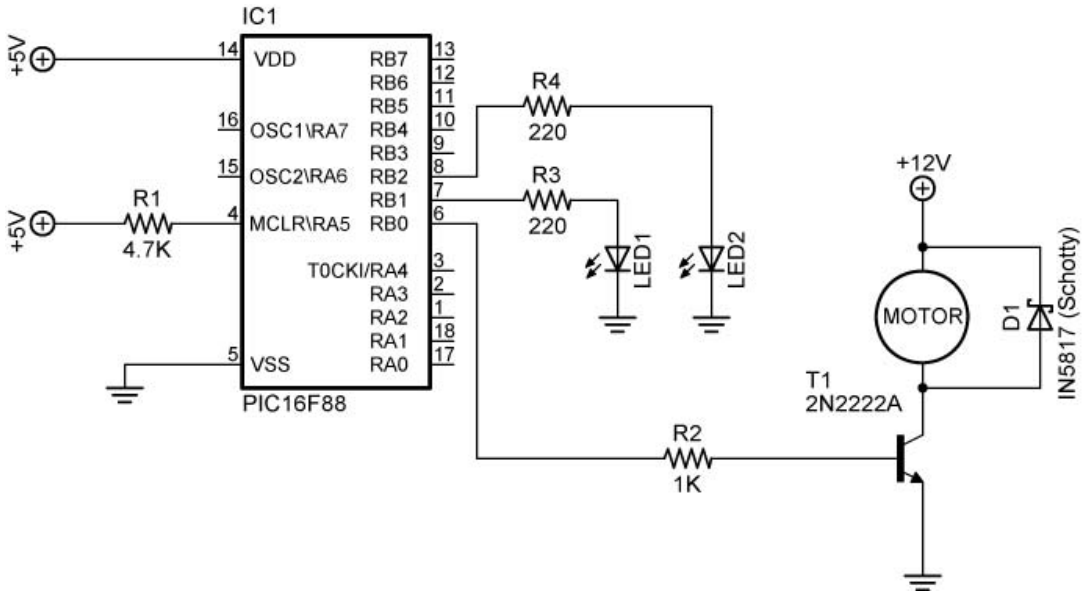
Circuit for pwm1.pbp

Cornerstone Electronics Technology and Robotics II
Pulse Width Modulation LAB 2 – hpwm1.pbp

- **Purpose:** The purpose of this lab is to acquaint the student with the command **HPWM** and how it runs in the background while the program executes other commands.

- **Apparatus and Materials:**
  - 1 – Breadboard with +5VDC and +12VDC Power Supply
  - 1 – PIC16F88
  - 1 – 1K Resistors
  - 1 – 4.7K Resistor
  - 1 – 2N2222A NPN Transistor
  - 1 – 1N5817 Diode
  - 1 – 12 Volt DC Motor – The Class Used the Jameco #155855 Gearhead 72 RPM Motor

- **Procedure:**
  - Wire the circuit as shown below.
  - Load **hpwm1.pbp** into the PIC16F88. The **hpwm1.pbp** program runs the HPWM command in the background while executing other PicBasic Pro commands.
  - Take note that the motor continues to run as the program executes the other program commands.



Circuit for hpwm1.pbp

  - Now save **hpwm10.pbp** as **hpwm10.pbp** and change the Dutycycle values.
  - Open **hpwm2.pbp** and load into the PIC16F88. This program changes the value of the Dutycycle from 90 to 255 and then to 0. The motor speed responds accordingly.

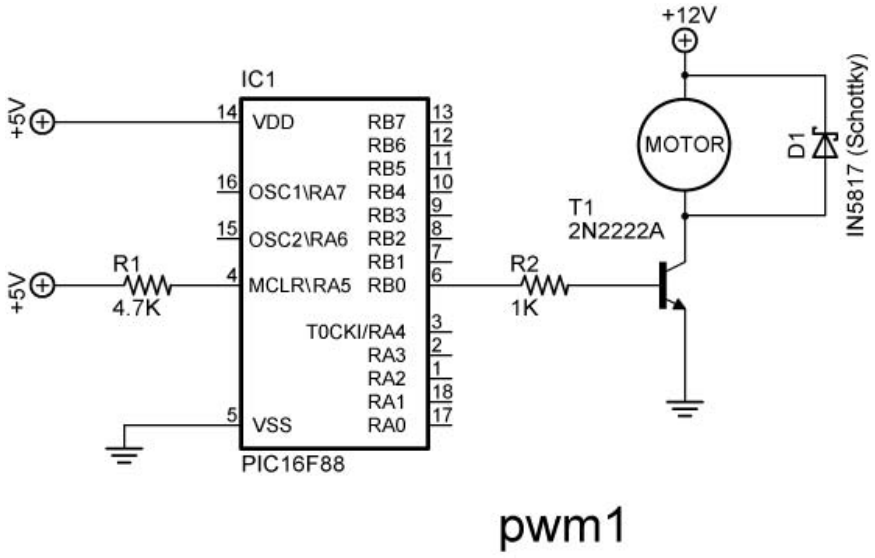
Cornerstone Electronics Technology and Robotics II
Pulse Width Modulation LAB 3 – PWM Calibration

- **Purpose:** The purpose of this lab is to acquaint the student calibrating motor rotational speeds driven by PWM.

- **Apparatus and Materials:**

    - 1 – RPM Meter
    - 1 – Breadboard with +5VDC and +12VDC Power Supply
    - 1 – PIC16F88
    - 1 – 1K Resistors
    - 1 – 4.7K Resistor
    - 1 – 2N2222A NPN Transistor
    - 1 – 1N5817 Schottky Diode
    - 2 – DC Motors of the Same Model

- **Procedure:**
    - Use the same circuit that was used in Lab 1 for Lab 3.



Circuit for PWM and HPWM Calibration

- Motor 1 Calibration with **PWM** Command:
    - Label the motors Motor 1 and 2 and connect the power supply positive lead to the + terminal on Motor 1 and the negative lead to the other terminal on Motor 1. This will be considered forward for Motor 1. Make sure the car does in fact travel forward, not in reverse.
    - Using **pwm10.pbp**, empirically determine the smallest Duty value before Motor 1 stops rotating. This number will be used as the smallest Duty value in the remaining discussion.

- Calculate Input Values:
  - Divide the range from the smallest Duty value to 255 into 9 almost equal parts. For example, if the smallest Duty value is 140, subtract 140 from 255 to yield 115. Divide 115 by 9 to get 12.77, or rounded, 13. Start the input values at 140. Write each of these input values down. Now add 13 to 140 to get the second input value, 153. To calculate the remaining input values, continue adding 13 to the previous total until you arrive at approximately 255. In this example, the input values would be: 140, 153, 166, 179, 192, 205, 218, 231, 244, and 257.
- Setting the Calibration Graph Scales:
  - Set the horizontal scale from your smallest Duty value (at the graph origin) to 255. Adjust the placement of the 255 value so that the lines on the scale make sense when plotting the other input values.
  - Program the PIC16F88 with the Duty value 255 so you can determine the highest rpm that you will measure. Set the vertical scale using this rpm value as the maximum value. Again, adjust this maximum value so the lines on the scale make sense.
- For each input value, measure the rpm of Motor 1 and plot the coordinates on the calibration graph. Label this plot as Motor 1 Forward.
- Switch the polarity of the motor leads and repeat the whole calibration process for Motor 1 in reverse. Plot the results on the same calibration graph as Motor 1 Forward.

o Motor 2 Calibration with **HPWM** Command:
- Connect the positive lead to the - terminal and the negative lead to the + terminal on Motor 2. This will be considered forward for Motor 2.
- Repeat the whole calibration process for Motor 2 except use **hpwm3.pbp**. Plot the results on a new calibration graph and label this plot as Motor 2 Forward.
- Switch the polarity of the motor leads and repeat the whole calibration process for Motor 2 in reverse. Plot the results on the same calibration graph as Motor 2 Forward.
- Save these plots for motor calibration in the coming weeks.

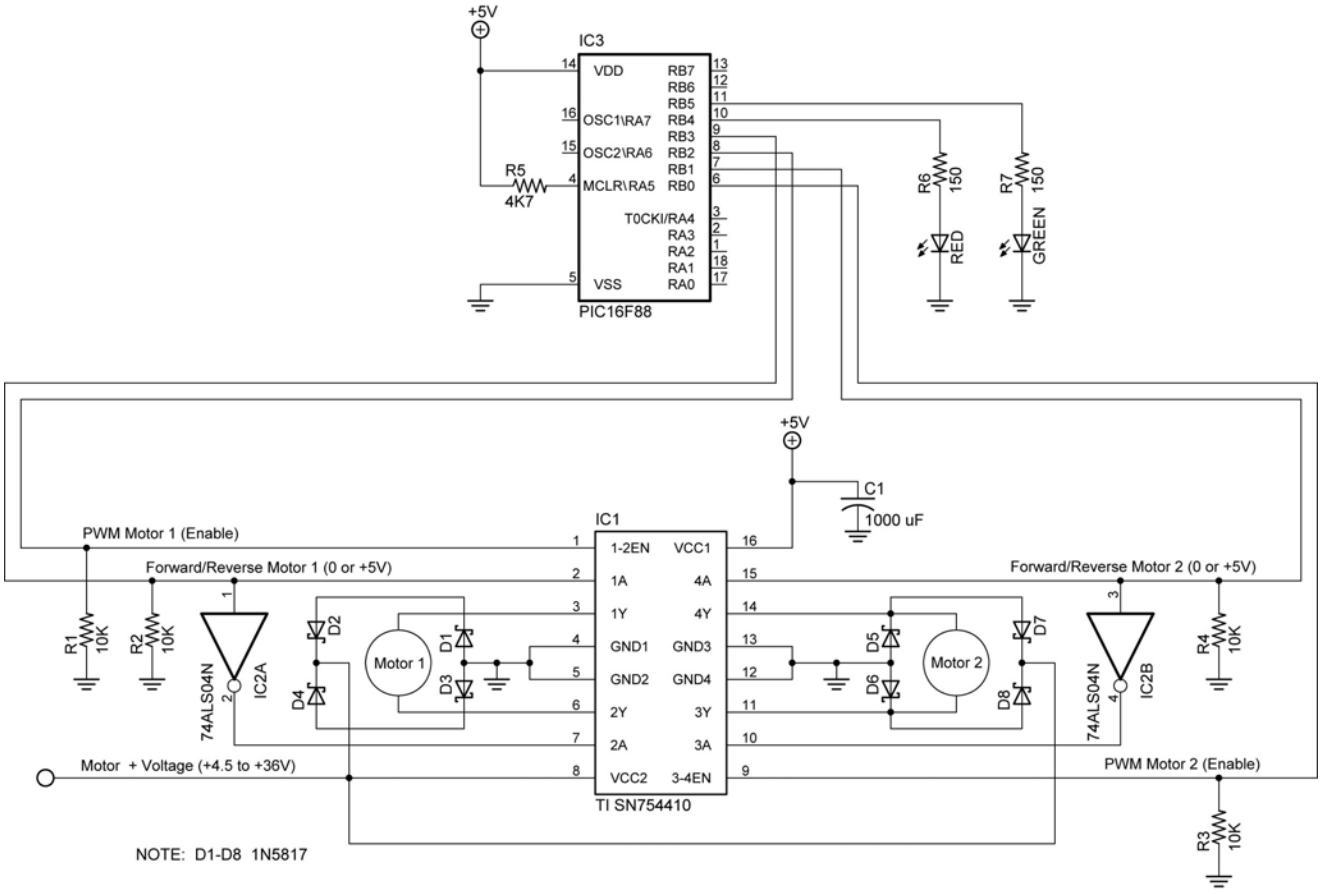
Cornerstone Electronics Technology and Robotics II
Pulse Width Modulation LAB 4 – Using PWM for Speed Control with the SN754410

- **Purpose:** The purpose of this lab is to have the student use PWM to change the speed of their robotic car.

- **Apparatus and Materials:**

  - 1 – Robotic Car Platform
  - 2 – Gearhead DC Motors, Jameco #155855
  - 1 – SN754410 Quadruple Half-H Driver, Pololu #0024
  - 1 – 74LS04 Hex-Inverter
  - 1 – PIC16F88
  - 8 – 1N5817 Diodes
  - 1 – 4.7K Resistor
  - 4 – 10K Resistors
  - 2 – 150 Ohm Resistors
  - 2 - LEDs
  - 1 – 1000 uF Capacitor

- **Procedure:**
  - Wire the following circuit on the robotic car breadboard. This is the identical circuit that was used in the H-Bridge lesson.
  - Program the PIC16F88 with **pwm_hpwm_sn754410.pbp**.



Texas Instrument SN754410 H-Bridge Motor Driver

- **Challenge:**
    - Using the calibration data just collected, program the robotic car to travel forward along the straight taped line.
    - Use the calibration graphs to make the robotic car travel at ½ its maximum speed.
    - Program the robotic car to follow the straight taped line in reverse.