Wheel Encoder

Purpose

- To introduce the concept of an encoder for determining rotational speed and angle
- To develop familiarity with the IRL 510 power MOSFET to control power devices
- To reinforce the concept of Pulse-Width Modulation to controlling the speed of a DC motor
- To explore the pulse counting capability of the Basic Stamp II microcontroller

Components

Qty.	Item
1	OEM Basic Stamp II microcontroller and serial port cable
1	1M Ω resistors
1	1N4148 diode
1	IRL510 power MOSFET
1	DC motor
1	photointerrupter module (Optek OPB960T51)
1	encoder disk

Introduction

In this lab you will investigate how rotational speed and rotational angle can be determined using a rotary encoder. A rotary encoder consists of a disk with alternating opaque and clear radial regions. In operation, a light source is positioned on one side of the disk, and a photosensitive device, such as a phototransistor is positioned on the other side of the disk. As the disk rotates, the passage of the opaque and clear regions of the encoder disk alternately block and allow light to impinge on the receiver, which produces corresponding voltage pulses. The rotational speed of the encoder disk can be determined by counting pulses during a known time period. The angle of rotation corresponds directly to the number of pulses, since the number of pulses per revolution is constant.

We will also look at controlling the speed of a DC motor using the concept of Pulse-Width Modulation (PWM).

Procedure

Function Generator Output to Control Duty Cycle

1. Set up the function generator (FG) to output a 1 kHz square wave (remember to set the output termination to HIGH Z). Look at the signal on the ‘scope.
2. Set the amplitude to 8 V p-p
3. Offset the waveform by 4 V, so that you have a 0 to 8 V square wave.
4. Select the ‘% Duty’ function on the FG by pressing the Shift key, then the Offset key.

5. Rotate the knob and see what happens to the output waveform when you vary the duty cycle. **What are the limits you can set the duty cycle to? What does "duty cycle" mean? Describe in your own words.**

DC Motor Speed Contol Using Duty Cycle

6. Build the circuit shown in Figure 1. Don't forget the diode. **Important note**: MOSFET'^{s} are very sensitive to static electricity. Make sure that you are not carrying static charge before you handle these devices. It is best to work on a properly grounded anti-static surface with an anti-static bracelet on your wrist. If these precautions are not available, then discharge yourself by touching a grounded metal surface (such as the frame of the bench) before you handle a MOSFET. Always handle a MOSFET by its large metal tab and by its leads.

   The MOSFET you will be using in this lab is the IRL510. It is a power MOSFET that is fully "on" when the gate to source voltage, V_{GS}, is 5 V, which is the nominal voltage level for a 'high' digital output. (Note: most power MOSFET's don't turn on fully until V_{GS} is about 10 V. Keep this in mind in your project work.) Its package style is an industry standard TO-220. This package is somewhat awkward to use in a solderless breadboard, because its leads are so large. To avoid damaging the solderless breadboard, insert the IRL510 so that its metal tab is ***parallel*** to one of the 5-hole rows, but with each lead in a separate row. To do this, you will have to bend the leads slightly (see Figure 1).

   Why is the diode needed across the motor leads?

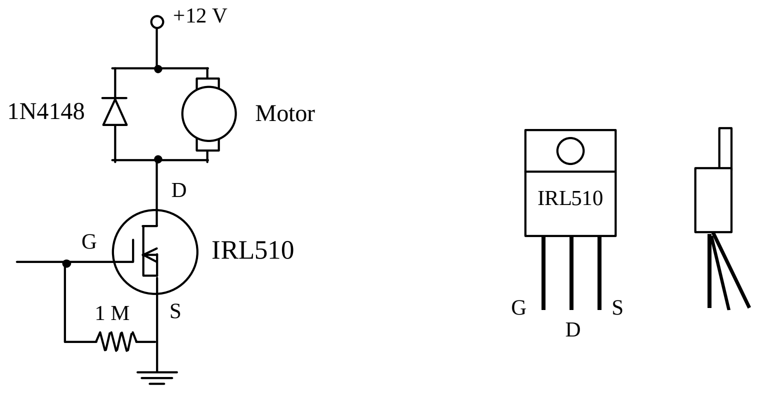

Figure 1 DC Motor Driver Circuit. The circuit uses an IRL510 power MOSFET. MOSFET's are very static sensitive, so handle them by the metal tab.

7. With the FG set up from Step 5, clip the red mini-hook lead from the function generator cable to the gate of the IRL510 and the black mini-hook lead to the common ground. Vary the duty cycle of the signal from the FG, and observe what happens to the motor. **Explain why the motor speed varies with the duty cycle.**

Speed Measurement Using an Encoder

8. Unhook the red lead of the FG from the gate of the IRL510. Attach the encoder wheel to the end of the motor shaft using the double-sided tape mounted on the encoder, and carefully center it.

9. Wire the optointerrupter module as shown in Figure 2 (please read the figure caption carefully). Double-check your wiring of power and ground before applying power (+5 V) to this unit. The output of the optointerrupter should be 5 V if nothing is blocking the slot, and close to 0 V when the slot is blocked. Verify this.

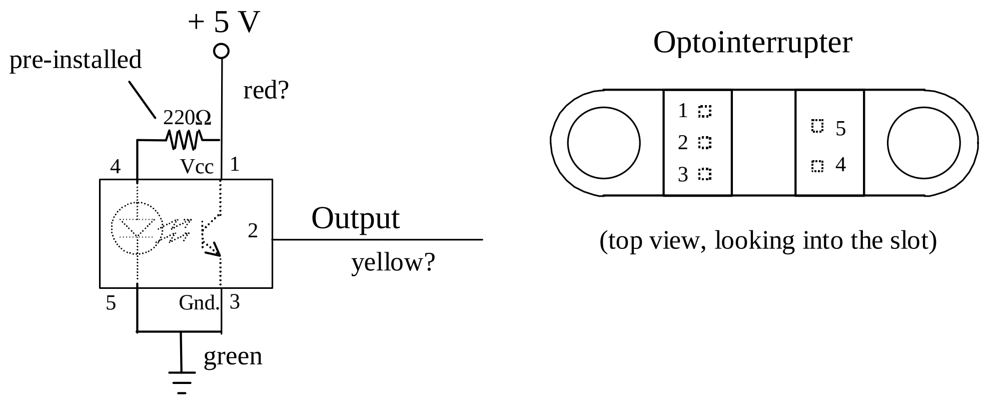

Figure 2 Optointerrupter Module. This device consists of an LED and a phototransistor mounted in a plastic housing separated by a slot. The output will be approximately 5 V when the slot is not blocked and approximately 0 V when the slot is blocked. The power and output lead wires may be different than the colors shown. To identify the power lead, look for the lead that connects to the resistor. Be aware that a current limiting resistor *must* be used in series with the LED of an optointerrupter. The optointerrupters you will use here in the lab *already* have the current limiting resistor installed to save you from having to buy us a new one if you were to forget the resistor. If you use a different optointerrupter at some later occasion, don’t forget to use a current limiting resistor!

10. Arrange the motor and optointerrupter, so that the encoder disk can rotate through the slot of the optointerrupter without touching its sides.

11. Re-connect the red mini-hook of the FG to the gate of the IRL510, and run the motor as in Step 7. Observe the output of the optointerrupter on the ‘scope. **Determine the speed of the motor at 5 different duty cycles that span the full range that the FG can output. Plot motor speed vs. % duty cycle in your report, and discuss your results.**

Basic Stamp Measurement of Encoder Pulses

12. Disconnect the red mini-hook of the FG and turn off the 12 V power supply. Insert the OEM Basic Stamp on the breadboard. Be ***very careful*** wiring power to V_{in} of the Stamp. We have lost several Stamps in the last few weeks, because some folks have been careless in making their power connections, so DOUBLE CHECK the power and ground connections before turning on the power supply. Connect the output of the optointerrupter to P0 of the Stamp.

13. We are going to use the COUNT function to determine the speed of the motor. The syntax of COUNT is:

```
COUNT pin, period, variable
```

Count the number of cycles (0-1-0 or 1-0-1) on the specified pin during *period* number of milliseconds and store that number in *variable.*

• **Pin**	is a variable/constant (0–15) that specifies the I/O pin to use. This pin will be placed into input mode by writing a 0 to the corresponding bit of the DIRS register.
• **Period**	is a variable/constant (1 to 65535) specifying the time in milliseconds during which to count.
• **Variable**	is a variable (usually a word) in which the count will be

14. Enter and run the following PBASIC program:

```
‘Tachometer Program
‘Put your name here
‘Put the date here

counts VAR word        ‘counts is a 16 bit variable
RPM VAR word           ‘RPM is a 16 bit variable
counts = 0             ‘initialize counts

loop:
   count 0, 1000, counts
   RPM = counts * 60
   debug dec ? RPM, 13       ‘try this statement without the “, 13” and see what happens
goto loop
```

Reconnect the red mini-hook of the FG to the gate of the IRL510. **Explain how this program works. Compare the speed you measure with the ‘scope with the value your program outputs. How well do these values agree? Explain the source of any discrepancy.**

15. Suppose the encoder were mounted to the shaft of a motor or the wheel of a vehicle. If the diameter of the wheel is 3 inches, write a PBASIC program that will indicate the rotational speed of the wheel and tell how far the vehicle has traveled. How accurately can you calculate the speed and distance? Quantify your answer.

    Run your program with the setup you just used. (In order to accomplish this, you will need to know how to work around the fact that PBASIC deals with integer arithmetic only. If you need to multiply a variable by a constant that has a fractional part, like π (pi) for example, use the */ operator, and represent the constant as a two-byte number. The whole number portion will be the upper byte, and the fractional portion will be the lower byte in units of 256. So for pi, the whole number portion is 3, so the upper byte will be $03. The lower byte will be 0.14159*256, which is about 36 or $24. So if you need to multiply a variable by pi, use the */ operator, and multiply by $0324.)