EGR 262

Fundamental Circuits Lab

File: N262L5

**Lab # 5**

**Pulse Width Modulation (PWM)**

A. **Objectives**

 The objectives of this laboratory are to introduce the student to:

1. Pulse width modulation (PWM)
2. Writing programs for the Arduino UNO to produce PWM signals
3. Capturing PWM signals on an oscilloscope
4. Using PWM signals to vary the brightness of an LED and the speed of a motor

B. **Materials**

 Breadboard with Arduino UNO

 Adaptor (120V AC to 12 VDC, 1000 mA)

 TIP32C PNP Bipolar Junction Transistor (BJT)

 7406 Open-collector Inverter

 DC Motor (12,150 RPM, 6-12V)

 LED

 220Ω resistor

 2.2 kΩ resistor

 Oscilloscope

 DC Power Supply

 WaveStar software for capturing oscilloscope images

C. **Introduction**

 Output current for digital pins on the Arduino UNO is limited to about 40mA. Output voltage is limited to 5V. One way to work with higher currents and voltages is to use open-collector logic gates and transistors.

 **Open-collector logic gates**

 TTL logic gates has two types of outputs:

1. Totem-pole outputs (most common type)
	* Output HIGH voltage is typically 5V
	* Output current typically < 16 mA
	* Input current typically < 1.6 mA (from the Arduino perhaps)
	* Example: 7404 Inverter (NOT) gate
2. Open-collector gates
	* Output HIGH can be “pulled up” to 30V or 40V using an external pull-up resistor
	* Output current typically < 40 mA
	* Input current typically < 1.6 mA (from the Arduino perhaps)
	* Will not function without a external pull-up resistor. 2.2kΩ is a common value for pull-up resistors.
	* Example: 7406 Inverter (NOT) gate

 ***Illustration:***

  

Open-collector gate:

* 0V or 12V delivered to the load
* up to 40 mA delivered to the load
* a 2.2k resistor is used to pull the output voltage up to 12V

Standard gate (totem pole):

* 0V or 5V delivered to the load
* up to 16 mA delivered to the load

 **Transistors**

 Note that open-collector gates can provide up to about 40 mA. What if the load is a motor that requires 1A? One option is to use a transistor. Transistors are often used to amplify signals, but they are also used as switches. The circuit below shows how a Bipolar Junction Transistor (BJT) can be used as a switch to drive a motor requiring 12V and 1A of current.



**IMotor**

**IB**

 Notes:

* When the output of the 7406 is LOW it creates a base current, IB, of a few mA which “turns ON” the TIP32C, making it act like a short from E to C, which provides 12V to the motor.
* When the output of the 7406 is HIGH, it is “pulled up” close to 12V resulting in a small IB which turns OFF the TIP32C, making it act like an open from E to C, which turns off the motor.
* The TIP32C is rated for up to 100V and 3A.

The following two circuits will be built and tested in lab:

**Arduino**

**UNO**

**D11**

**220 Ω**

**LED**

**Circuit 1**

**Arduino**

**UNO**

**D11**

**Circuit 2**

**7406**

**Rp**

**2.2k**

**12V**

**TIP32C**

**E**

**B**

**C**

**12V DC**

**Motor**

**Iin**

**Iout**

**Imotor**

D. **Pre-Lab Tasks**

1. Write a C++ program for Lab #5 to produce a PWM output according to the following specifications:
	* Begin with the program in the Lab 5 presentation (slide 18 perhaps) which produces a PWM signal on D11 based on user input.
	* Prompt the user to enter a value from 0 to 255. If an incorrect value is entered, display an error message and loop back to allow the user to again enter a value.
	* If a correct value is entered, indicate that a PWM signal was sent to D11 and also display the duty cycle (as a percentage) and the analog (DC) voltage.
	* The ouput for a valid input of 64 might look something like the following:

***Enter a value for the width of the pulse (0 to 255):***

***Value read = 64***

***PWM signal sent to D11***

***Duty cycle = 64/255 = 25.20%***

***DC voltage on D11 should be 1.25 V***

***Enter a value for the width of the pulse (0 to 255):***

1. Show the pinout for the 74LS06.
2. Show the pinout for the TIP32C (or use the image below)



1. Use ***Fritzing*** to draw a ***breadboard layout*** and ***schematic*** for Circuit 2.
	1. Begin with the standard EGR262 breadboard layout (EGR262Breadboard.fzz)
	2. Save your breadboard layout and schematic for Lab 5 (.fzz file)
	3. Export the breadboard as a jpg image to insert in your lab report.
	4. Export the schematic as a jpg image to insert in your lab report.
	5. Fritzing parts for Lab 5 (also shown below):
		* **Motor**: Use the part ***DC Motor*** from the ***Core - Output*** library.
		* **TIP32C**: Use the part ***Mystery Part – 3 pins*** from the ***Core – Basic*** library. Rename it as TIP32C using the Inspector. You can also edit the part and change its ***Title*** to TIP32C (under the MetaData tab) and change its pins (under the Connectors tab). Change Pin1 to B (Base), Pin 2 to C (Collector), and Pin 3 to E (Emitter). It will be added to the ***Mine*** library when you save it.
		* **74LS06**: Use the part ***DIP – 8 pins*** (the image says IC) from the ***Core – ICs*** library. Use the Inspector to change the number of pins to 14 and the label to 74LS06. You can also save it to the ***Mine*** library.
		* **12V Power Supply**: In the ***breadboard view***, you can simply show two wires from the breadboard to be connected to the 12V power supply. Note that the ground must be common for the 5V supply and the 12V supply. However, do not connect the +12V lead to the 5V power rails! It is only connected to the emitter of the TIP32C. In the schematic view, you can use the part ***DC Power*** in the ***Core – Schematic View*** library. You can rename it as 12 V as shown:



E. **In-Lab Tasks**

1. **Turn on the oscilloscope before the computer**
	* The oscilloscope must be turned on before the computer in order to capture oscilloscope images using Wavestar.
	* If the computer is already on, turn on the oscilloscope and restart the computer.
2. **Testing the Program**
	* Download your program and run the program for Lab 5.
	* View the output on D11 using the oscilloscope. The PWM output should appear and should change every time you enter a valid number into the program. You might want to have the instructor look at the output before you begin capturing oscilloscope images.
	* Run the program for two invalid inputs (one < 0 and one > 255). Also run it for two valid inputs. Capture the computer output showing these 4 runs to include in your report.
	* If any changes were made to the program, save the program under a different name (L5v2, perhaps) and include a copy in the Post-Lab section.
3. **Measuring Output Voltages**
	* Measure and record the output voltage with a voltmeter for the following 28 inputs: 0, 1, 1X, 2X, 24X, 254, and 255, where X is the last digit in your Student ID. Also record the value predicted by your program. A sample table for X = 6 is shown below:

|  |  |  |
| --- | --- | --- |
| Input | VDC (V) predicted by program | VDC (V) measured using voltmeter |
| 0 |  |  |
| 1 |  |  |
| 16 |  |  |
| 26 |  |  |
| … |  |  |
| 236 |  |  |
| 246 |  |  |
| 254 |  |  |
| 255 |  |  |

1. **Capturing Oscilloscope Images**
	* Capture oscilloscope images for the following 5 inputs: 2X, 8X, 14X, 20X, and 24X, where X is the last digit in your Student ID.
	* Also record the predicted value of D from your program for each of the 5 inputs.
	* For each image:
		+ Use a vertical scale of 1V/div and line up the bottom of the waveform with a major division.
		+ Display perhaps 1-2 complete periods of the waveform.
		+ Add two cursors to measure TH (TH will be the difference, Delta, between the cursors).
		+ Capture the oscilloscope image. Copy and paste it into your lab report.
		+ Label the images as indicated below in your report and calculate D for each image.



1. **Using PWM to control the brightness of an LED**
	* Construct Circuit 1. (You should leave your 7-segment display circuit on the breadboard for later labs, but disconnect buttons circuits or other previous connections from D11.)
	* Verify that the LED brightness changes as the program input changes from 0 to 255.
	* Estimate the brightness of the LED (perhaps on a scale of 1-10) for the following 5 inputs: 0, 50, 100, 150, 200, and 255. Record the results.
	* Demonstrate the circuit to the instructor.
2. **Using PWM to control the speed of a DC motor**
	* ***Warning***: Before connecting the 12V supply to the circuit, ask the instructor to check your connections. The +12V connection only goes to the emitter on the TIP32C, not to the 5V power rail. If you connect it to the 5V power rail, you may damage the Arduino UNO.
	* Construct Circuit 2. Note that the ground for the 12V supply and the ground for the Arduino must be common.
	* Verify that the speed of the motor changes as the program input changes from 0 to 255.
	* Estimate the speed of the motor (perhaps using slow, fast, very fast, etc) for the following 5 inputs: 50, 100, 150, 200, and 255. Also find the highest value of the input where the motor stops. Record the results.
	* When the input is 255 (max motor speed), use an ammeter to measure Iin, Iout, and Imotor.
	* Demonstrate the circuit to the instructor.
3. **Comments**
	* Record comments for any problems encountered or lessons learned in this lab.

F. **Post-Lab Tasks**

1. Discuss the performance of the program.
2. Compare the measured DC voltages to the DC voltages calculated by your program.
3. Use Excel to graph measured VDC versus input (0 to 255). Does the voltage vary linearly? Perform a regression analysis (add a ***linear trendline***) and display both the formula and the value of R2. What does the value of R2 imply?
4. Use a table to compare the values of D predicted by your program and the values calculated from the oscilloscope images. Discuss the results.
5. Discuss the results of using PWM to control the brightness of an LED.
6. Discuss the results of using PWM to control the speed of a motor. Why does the motor stop turning when the input value is too low?
7. Discuss the currents measured in the motor circuit. Are all values within specifications? Specifically:
* Compare Iin to the max input current for the 7406 and the max output current for D11 on the Arduino.
* Compare Iout to the max output current for the 7406.
* Compare Imotor to the max current for the TIP32C.
1. Could the motor have been controlled directly from D11 without using the 7406 and TIP32C? Why or why not?
2. Include the final version of your program. Highlight any changes that were made and discuss them. If no changes were made, clearly say so.

G. **Report**

 A lab report is due 1 week after the date of the experiment.

* The lab report must be your own work. Copying data, tables, graphs, circuits, etc., from other students is not allowed and will result in grades of 0 on the lab.
* Be sure to follow good practices for presenting all tables and graphs. See the Presentation for Lab #1 for examples.
* The lab report should consist of the following sections:
1. Title Page (include course number & title, lab number & title, date, and your name)
2. Pre-Lab Tasks
	* Include instructions or headings for all items to make the report easy to follow.
3. In-Lab Tasks
* Include instructions or headings for all items to make the report easy to follow.
* Be sure to include comments from lab as well as measured data.
1. Post-Lab Tasks
* Include instructions or headings for all items to make the report easy to follow.