Embedded Hardware Design

Monsoon 2017

Experiment 4

Date : 31-08-2017
Topics: PWM, Motor Control and Optocoupler

Pulse Width Modulation-

In the last lab, we used *analogRead()* function to read values which were between 0 and 5V on the analog pins (A0-A5) by invoking the ADC. But what if we want to do the opposite, i.e, generate a voltage between 0 and 5V on an output port? As you may have guessed by now, there is something called *analogWrite()* which does exactly that! But it isn't used on analog pins. You may have noticed that some digital pins have a tilde (~) sign before their pin number. To be specific, they are: 3, 5, 6, 9, 10 and 11. The ~ denotes that those pins have PWM enabled and can be used to generate any voltage using the *analogWrite()* function.

PWM is a technique for getting analog results by digital means. Basically, a square wave is generated on those pins whose pulse width (duration of "on time") simulates the voltage levels between full on (5 Volts) and off (0 Volts). To get varying analog values, you change that pulse width.

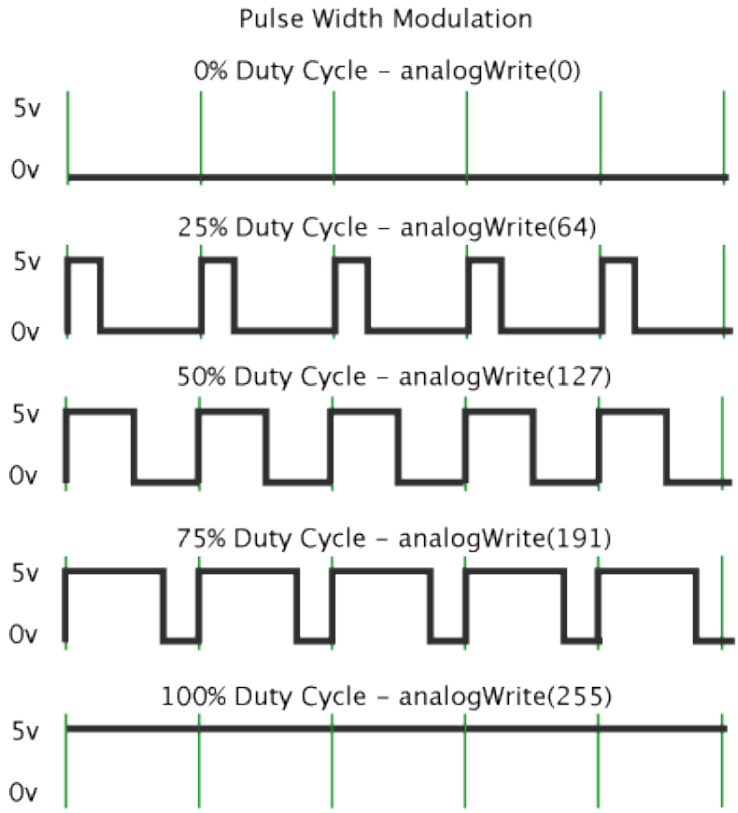

Illustration 1: PWM concept

In analog, 0 meant 0V and 1023 was the maximum value you could read which meant 5V. However, when it comes to PWM, 0 means 0V and 255 is the maximum value you can give to your *analogWrite()* function (for 5V). For example, 50% duty cycle or 127 would give you *0.5*5V=2.5V*.
The applications of PWM are numerous! In this lab, we'll use it to control the speed of the motor. **Functions-**

1. *analogWrite(pin, val):* Any pin ∈ {3, 5, 6, 9, 10, 11} and val ∈ [0, 255]
2. *map(val, fromLow, fromHigh, toLow, toHigh):* If you want to read an analog signal and based on that reading change the PWM value, then map function is useful.

Note that arduino pins 5, 6 provide PWM with frequency ~1kHz and 3, 9, 10, 11 at about 500Hz. We can implement PWM on any pin by using *delayMicroseconds()*. This method is called Bit-banging.

Motor Control with Arduino-

A DC motor is the most common type of motor in use. They're extensively used in the field of robotics. DC motors have two leads- positive and negative. They always have a voltage and RPM reading written on them. The motors that we'll be using today are 9V, 2400 rpm rated. But have you ever heard of a microcontroller working beyond 5V? No! So how should we interface this with our simple 5V Uno? The answer is a power transistor. We can give 9V from a power supply to the collector and control the base of that transistor using our PWM pin.

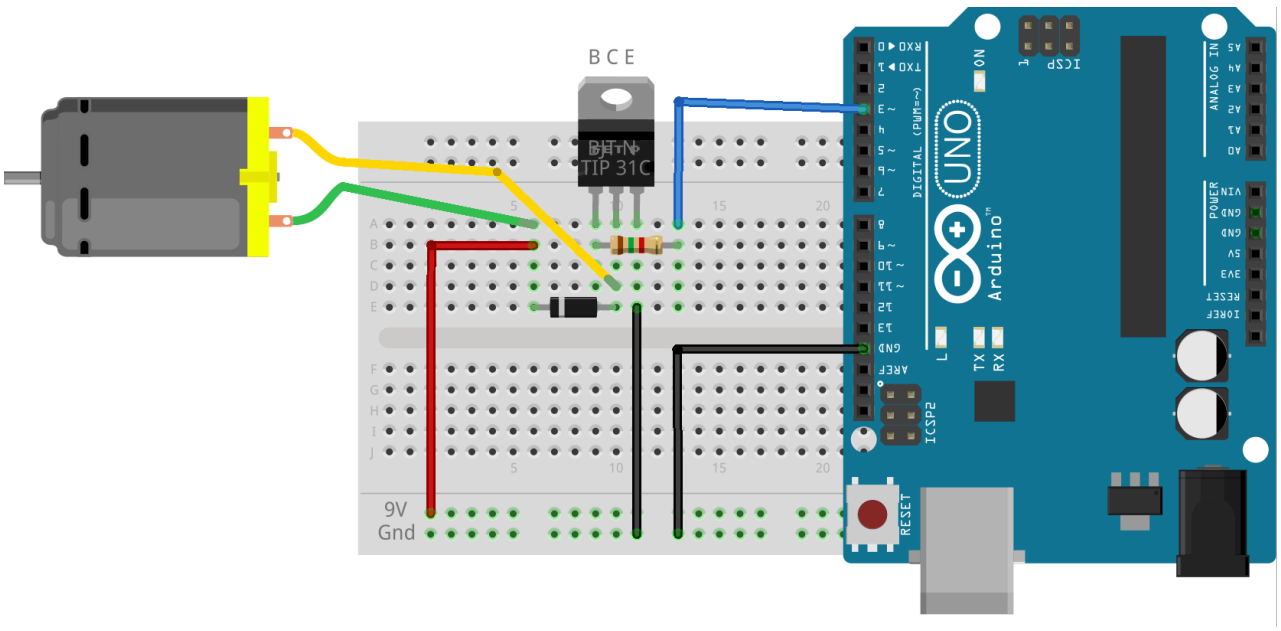

Illustration 2: Motor interfacing with Arduino

Optocoupler-

The optocoupler or optical isolator is a component that transfers electrical signals between two isolated circuits by using light. It has an IR diode and a phototransistor connected as shown-

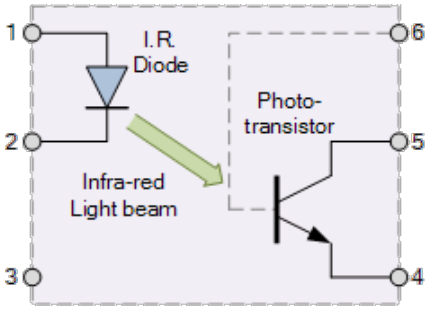

Illustration 3: A typial Optocoupler

When the diode is on, infrared beam is directed on the base of the phototransistor thereby turning the transistor ON. In the experimental setup for Lab 4, the motor is connected to a slotted disc. An optocoupler is set up at its rims such that when an IR diode is on one side and the phototransistor on the other. The slots of the disc pass through this optocoupler setup. As the motor spins, we'll get pulses on the collector of the phototransistor which will be used to calculate its speed.

Experiment 4A: Motor Speed Control

1. Set up all the components as per *Figure 2*. Use any PWM pin to interface.
2. Interface a potentiometer with Arduino on any analog pin.
3. Control the PWM width and hence the speed of the motor using the potentiometer as input (If the potentiometer is fully turned clockwise, speed of motor should be maximum).
4. Now, generate the following waveforms and check on the CRO-
   a) PWM with frequency of 1.5kHz at 40% duty cycle.
   b) PWM with frequency of 3kHz at 75% duty cycle.
5. In (4), you also need to output the voltage and count value of delay on the Serial Monitor.

Experiment 4B: Measurement of Speed

1. In addition to the setup shown in Figure 2, interface the IR LED and the phototransistor that is present on the motor setup.
2. First, apply 5V from the arduino to the base at 100% duty cycle. Check the output of the phototransistor on the CRO. Measure the time period of the waveform and hence calculate the speed in RPM. This gives the maximum speed that the motor can attain (for this experiment).
3. Write an ISR which is triggered by these pulses. Increment a variable (global and volatile) “count” every time the ISR is executed.
4. Write a main program to count the number of interrupts coming in a pre-determined time interval (using custom PWM frequency) and display the speed on the Serial Monitor.
5. Rotate the potentiometer and observe the change in speed displayed on the serial monitor.