ME 120 Basic Pulse Width Modulation
Winter 2015 Gerald Recktenwald
v: March 30, 2015 gerry@pdx.edu

1 Basic PWM Properties

Pulse Width Modulation or PWM is a technique for supplying electrical power to a load that has a relatively slow response. The supply signal consists of a train of voltages pulses such that the width of individual pulses controls the effective voltage level to the load. Both AC and DC signals can be simulated with PWM. In these notes we will describe the use of PWM on an Arduino for controlling LEDs and DC motors.

The PWM pulse train acts like a DC signal when devices that receive the signal have an electromechanical response time that is slower than the frequency of the pulses. For a DC motor, the energy storage in the motor windings and the inertia of the rotating mass in the rotor and winds effectively smooths out the energy bursts delivered by the input pulses so that the motor experiences a lesser or greater electrical power input depending on the widths of the pulses. For an LED, using PWM causes the light to be turned on and off at frequency higher than our eyes can detect. We simply perceive the light as brighter or dimmer depending on the widths of the pulses in the PWM output.

Figure 1 shows a voltage signal comprised of pulses of duration τ_o that repeat every τ_c units of time. The output of a PWM channel is either V_s volts during the pulse or zero volts otherwise. If this signal is supplied as input to a device that has a response time much larger than τ_c, the device will experience the signal as an approximately DC input with an effective voltage of

$$V_{\text{eff}} = V_s \frac{\tau_o}{\tau_c} \tag{1}$$

The ratio τ_o/τ_c is called the *duty cycle* of the square wave pulses. The effective DC voltage supplied to the load is controlled by adjusting the duty cycle.

2 Using PWM on an Arduino

An Arduino Uno has 14 digital input/output (I/O) pins[1]. Conventional, i.e., not PWM, operation of the digital I/O pins is controlled with the `pinMode`, `digitalRead` and `digitalWrite` functions. The `pinMode` function is used to configure a pin as an input or output. When a digital I/O pin is configured as an input, `digitalRead` reads the state of the pin, which will be either `HIGH` or `LOW`.

[1] `http://arduino.cc/en/Main/ArduinoBoardUno`

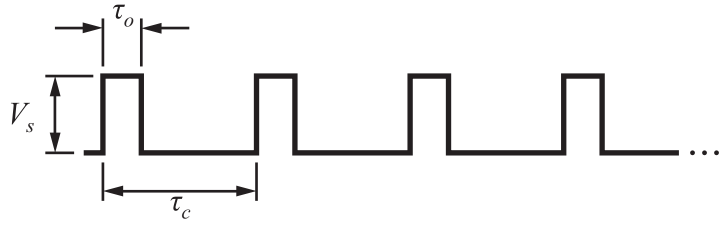

Figure 1: Nomenclature for definition of PWM duty cycle.

```
int PWM_out_pin = 9;   //  Must be one of 3, 5, 6, 9, 10, or 11
             //  for Arduino Uno
void setup() {
  pinMode(PWM_out_pin, OUTPUT);
}

void loop() {
  byte PWM_out_level;

  PWM_out_level = ...  //  Code logic to set output level

  analogWrite( PWM_out_pin, PWM_out_level);
}
```

Listing 1: Skeleton of an Arduino sketch to demonstrate the use of `pinMode` and `analogWrite` in controlling PWM output.

In an Arduino sketch, `HIGH` is a predefined constant that is evaluated as "true" in a conditional expression, and is equivalent to a numerical value of 1. Electrically, a value of `HIGH` means the pin voltage is close to 5 V. Similarly, the constant `LOW` is interpreted as "false" in conditional expressions, it is numerically equivalent to 0, and electrically the pin voltage is 0. When a digital I/O pin is configured for output, `digitalWrite` is used to set the pin voltage to `HIGH` or `LOW`.

On an Arduino Uno, PWM output is possible on digital I/O pins 3, 5, 6, 9, 10 and 11. On these pins the `analogWrite` function is used to set the duty cycle of a PWM pulse train that operates at approximately 500 Hz[2]. Thus, with a frequency $f_c = 500\,\text{Hz}$, the period is $\tau_c = 1/f_c \sim 2\,\text{ms}$. As with conventional digital I/O, the `pinMode` function must be called first to configure the digital pin for output.

The skeleton of a sketch in Listing 1 shows the basic code components for using PWM on an Arduino. Two integer variables, `PWM_out_pin` and `PWM_out_level` are used to indicate the pin number and output level respectively. In the `setup` function, the statement

```
pinMode(PWM_out_pin, OUTPUT);
```

configures the `PWM_out_pin` for output. The `OUTPUT` symbol is a constant defined by the Arduino programming tools (the IDE or Integrated Development Environment). The statement

```
analogWrite( PWM_out_pin, PWM_out_level);
```

sets the `PWM_out_pin` pin to the value of the PWM duty cycle and hence the effective voltage according to Equation (1). The parameter that sets the duty cycle is an 8-bit unsigned integer, i.e., a value in the range $0 \leq$ `PWM_out_level` ≤ 255.

The digital output voltage of an Arduino Uno is either 0 V or 5 V. Thus, in Equation (1), $V_s = 5\,\text{V}$. The PWM output level specified with the `analogWrite` is an 8-bit value that corresponds to an effective voltage range of 0 to 5 V. Figure 2 depicts the relationships between the PWM output parameters. The quantities in Figure 2 are linearly related. Thus,

$$\texttt{PWM\_out\_level} = 255 \times \frac{\tau_o}{\tau_c} = 255 \times \frac{V_{\text{eff}}}{V_s}$$

Remember that the second input to `analogWrite` is an unsigned 8-bit value, so `PWM_out_level` should be an `byte` variable type. Since $V_s = 5\,\text{V}$ always, for routine calculations, we can use the simple formula

$$\texttt{PWM\_out\_level} = \frac{255}{5} \times V_{\text{eff}} \tag{2}$$

[2]See `http://www.arduino.cc/en/Tutorial/PWM` and `http://arduino.cc/en/Tutorial/SecretsOfArduinoPWM`.

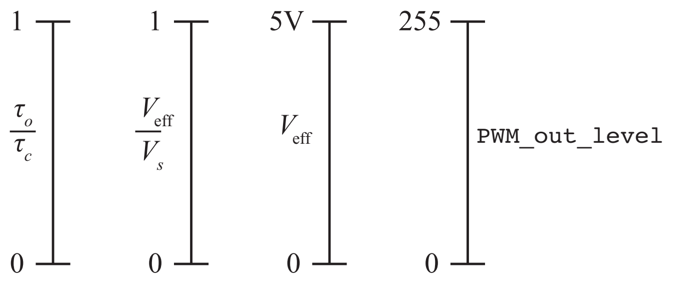

Figure 2: Scaling relationships for PWM parameters. `PWM_out_level` is the 8-bit value used as the second parameter to the `analogWrite` function.

Therefore, to supply an effective voltage of 3 V use

$$\texttt{PWM\_out\_level} = \frac{255}{5} \times 3 = 153$$

3 Implementation Examples

We now show how PWM can be applied to two practical situations: controlling the brightness of an LED, and controlling the speed of a DC motor. The Arduino code for both cases is the same.

3.1 Blinking an LED

The very first program for most Arduino users is to blink an LED on and off. The wiring diagram for the LED is shown on the left side of Figure 3. The basic blink program is available as `blink.pde`, and is part of the standard installation of the Arduino Integrated Development Environment (IDE). Make the following menu selections from an open Arduino window

File → Examples → Basics → Blink

Note that the `blink.pde` program *does not* use PWM.

3.2 LED Brightness

Blinking an LED involves sending a signal to the LED that is either on or off, true or false, 5V or 0V. Adjusting the brightness of an LED requires the sending voltage values in the range between 0V and 5V using the PWM technique described earlier.

Example 3.1 Modifying the Blink Program to use PWM

Controlling the brightness of an LED with PWM is straightforward. The LED circuit is connected to one of the PWM output pins and the duty cycle for the PWM output is controlled with the `analogWrite` command.

Listing 2 shows two programs for blinking an LED with an Arduino. The program on the left (`blink.pde`) is a slight variation on the `Blink.pde` program that comes as a standard part of the Arduino distribution. `Blink.pde` uses the `digitalWrite` command to turn an LED on and off. During the "on" state the digital output pin is supplied with 5V.

```
//  File:  Blink.pde
//
//  Turns on an LED on for one second,
//  then off for one second, repeatedly.
//  For digital I/O the brightness is set
//  to maximum during the "on" phase.

//  Use pins be 0, 1, ..., 13 for digital I/O
int LED_pin = 11;



void setup() {
  pinMode(LED_pin, OUTPUT);
}

void loop() {
  digitalWrite(LED_pin, HIGH);
  delay(1000);
  digitalWrite(LED_pin, LOW);
  delay(1000);
}
```

```
//  File:  Blink_dim.pde
//
//  Turns on an LED on for one second,
//  then off for one second, repeatedly.
//  Use PWM to control variable level of
//  brightness of the "on" phase.

//  Use pins 3, 5, 6, 9, 10 or 11 for PWM
int LED_pin = 11;
//  must be in the range 0 <= level <= 255
int level = 20;

void setup() {
  pinMode(LED_pin, OUTPUT);
}

void loop() {
  analogWrite(LED_pin, level);
  delay(1000);
  analogWrite(LED_pin, 0);
  delay(1000);
}
```

Listing 2: Arduino using PWM to control the brightness of an LED.

The `Blink_dim.pde` program on the right hand side of Listing 2 uses the `analogWrite` function to supply a variable voltage level to the LED.

□

Example 3.2 PWM Control of LED Brightness

In this example, the brightness of the LED is set to three levels in an endlessly repeating sequence. This allows us to verify that the PWM control of brightness is working as it should. The LED circuit is the same.

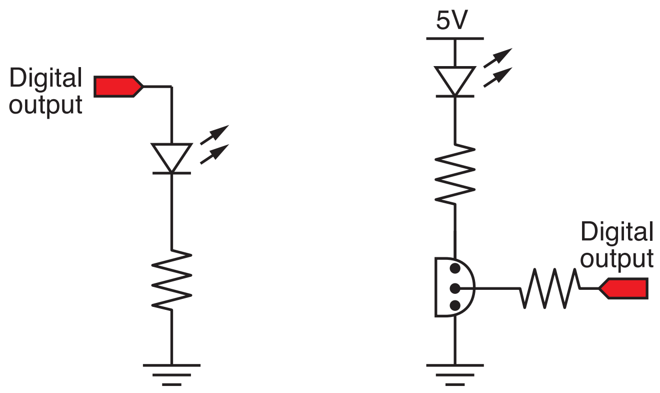

Figure 3: Electrical circuit for the basic blink program on the left, and the transistor-controlled blink program. Note that the program to drive these two circuits can be identical as long as the same digital output pin is used for both circuits.

```
//   File:  LED_three_levels.pde
//
//   Use PWM to control the brightness of an LED
//   Repeat a pattern of three brightness levels

int  LED_pin = 11;             //  must be one of 3, 5, 6, 9, 10 or 11 for PWM

void setup() {
  pinMode(LED_pin, OUTPUT);    // Initialize pin for output
}

void loop() {

  int  dtwait = 1000;          //  Pause interval, milliseconds
  int  V1=20, V2=220, V3=120;  //  8-bit output values for PWM duty cycle

  analogWrite(LED_pin, V1);
  delay(dtwait);

  analogWrite(LED_pin, V2);
  delay(dtwait);

  analogWrite(LED_pin, V3);
  delay(dtwait);
}
```

Listing 3: Arduino using PWM to control the brightness of an LED.

The `LED_three_levels.pde` sketch in Listing 3 causes an LED to glow at three different brightess levels specified by the variables `V1`, `V2` and `V3`. Each level is held for a time interval specified by the `dtwait` variable.

______________ □

Example 3.3 Transistor-based PWM Control of LED Brightness

The Arduino digital output pins can supply a maximum current of 40 mA. In many applications, higher current loads need to be switched. For modestly larger current loads, say up to 1A, simple NPN transistors can be used. For higher currents, MOSFETs or relays are necessary. In this example, the basic principle of using a transistor to switch a load is demonstrated. The load is an LED, which does not *need* a transistor for switching.

The Arduino sketch is the same as in the preceding examples. The only difference is in the electrical circuit, where the PWM output pin is tied to the base of an NPN transistor, as shown in the right half of Figure 3.

______________ □

Continue Writing Here:

Example 3.4 PWM Control of a DC Motor

- DC motor circuit: Transistor control; Diode snubber
- Sketch

______ □

3.3 Hardware Limits

Each digital output pin of an Arduino Uno can supply no more than 40 mA. To drive modestly higher current loads, the PWM output can be used with a transistor that switches the load. That strategy works with small DC motors. For example, a P2N2222 transistor from ON Semiconductor has a current limit of 600 mA, a factor of 30 higher than a digital pin of an Arduino.

Controlling higher power AC or DC loads introduces the potential for electrical hazards, and should not be attempted without substantial experience and abundant safety precautions.

A MOSFET (Metal Oxide Field Effect Transistor) device can be used to control DC loads at higher currents. Arduino shields using MOSFET controls can be purchased from Sparkfun [3] or other vendors.

One solution for controlling resistive AC loads such as heaters is a product called the PowerSwitchTail, which is available from the manufacturer (`http://powerswitchtail.com/`) or Adafruit (`http://www.adafruit.com/products/268`).

3.4 Further Reading

The *Secrets of Arduino PWM* on the official Arduino web site[4] explains some of the low-level details of Arduino PWM, and how to take fine level control over the PWM outputs.

[3]See, e.g., the Power Driver Shield `http://www.sparkfun.com/products/10618` or the mini FET Shield `http://www.sparkfun.com/products/9627`

[4]`http://arduino.cc/en/Tutorial/SecretsOfArduinoPWM`

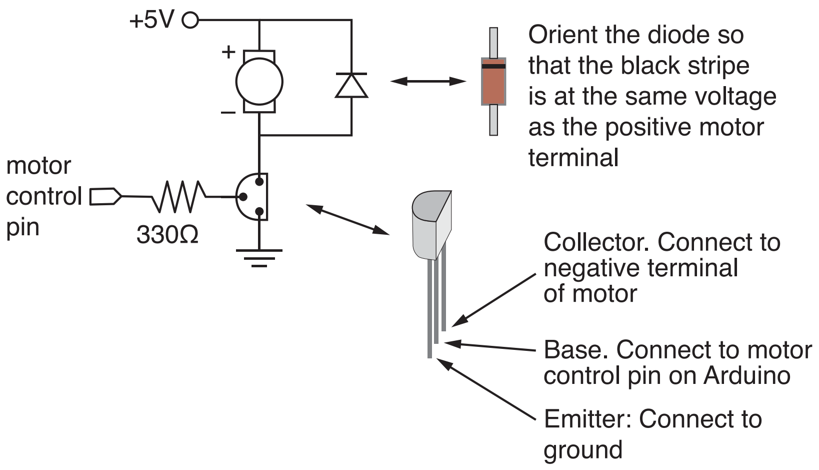
+5V
+
–
motor control pin
330Ω
Orient the diode so that the black stripe is at the same voltage as the positive motor terminal
Collector. Connect to negative terminal of motor
Base. Connect to motor control pin on Arduino
Emitter: Connect to ground

Figure 4: Electrical circuit for using PWM to control the speed control of a DC motor.

3.5 Study Questions

1. What is the effective voltage V_{eff} for $V_s = 5V$ and duty cycles of 25 percent, 38 percent, and 64 percent?

2. A small DC motor is connected to digital output pin 7 of an Arduino Uno. What are the Arduino commands to set the effective output voltage to 2V, 3V, and 4.5V? Only write the `analogWrite( )` command.