IEEE Arduino Pulse Width Modulation Workshop

Pulse width modulation (PWM) is a way to change the average power supplied by a signal. In figure 1, if the voltage is 5 volts while the wave is high and the duty cycle is 50%, then the average voltage will be 2.5 volts. This is because the signal is 5 volts for half of the time, so it averages out to 2.5 volts (5 volts * 50% = 2.5 volts).

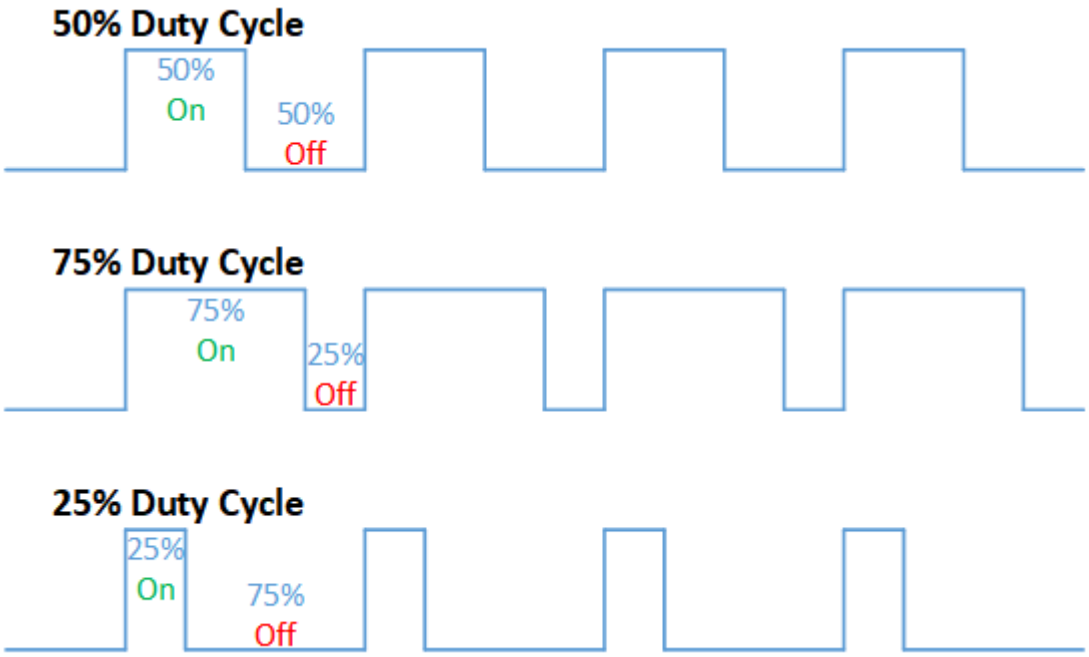

Figure 1. Duty cycle example.

Note: The output pins that can do pulse width modulation on the Arduino Uno are output pins 3, 5, 6, 9, 10, and 11. They have a wave next to them to indicate that they can be pulse width modulated outputs as shown in figure 2 below.

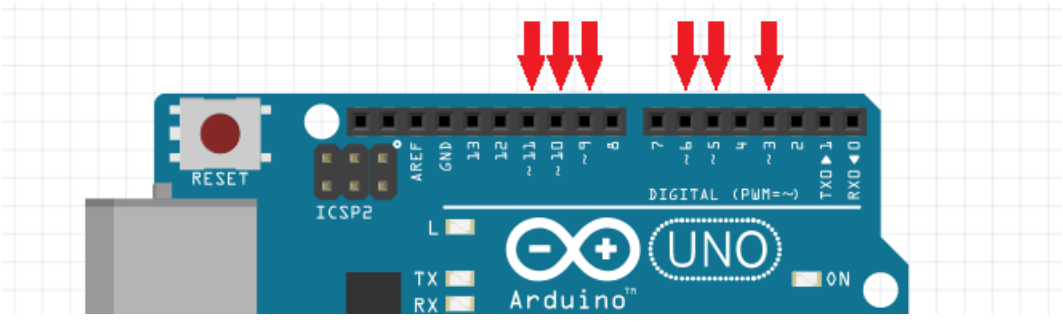

Figure 2. PWM pins on the Arduino Uno

Part 1: Changing the Brightness of an LED

Pulse width modulation can be useful for controlling circuit components in different ways. For instance, you can change the brightness of an LED by using pulse width modulation. If the duty cycle increase, the LED gets brighter, but if the duty cycle decrease, the LED becomes dimmer. Set up the hardware as shown in figure 3 and use the code in figure 4 to change the brightness of the LED.

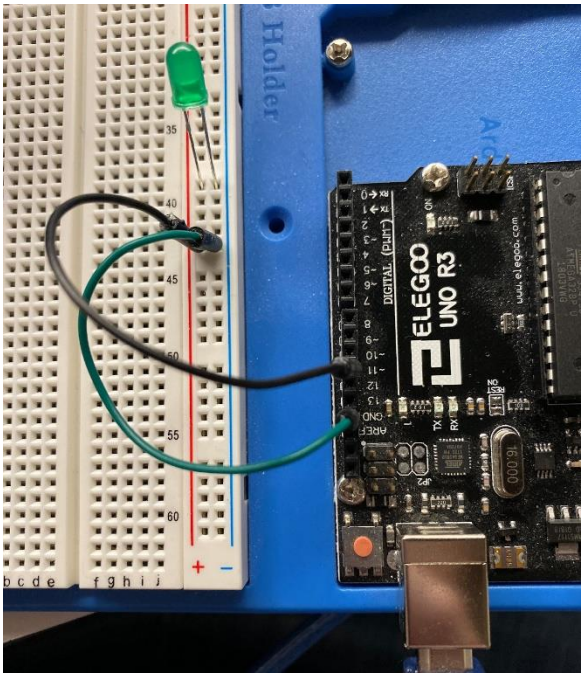

Figure 3. Hardware setup. The gray wire goes from the positive side (longer lead) of the LED to pin 11. The green wire goes from the negative side of the LED to ground (GND).

```
int led_pin = 11;
int brightness;

void setup() {
    pinMode(led_pin, OUTPUT);
    brightness = 100 ;

    //Brighness must be between 0 and 255
    //255 is a 100% duty cycle (1 x 5 volts = 5 volts average)
    //191 is a 75% duty cycle (.75 x 5 volts = 3.75 volts average)
    //127 is a 50% duty cycle (.5 x 5 volts = 2.5 volts average)
    //64 is a 25% duty cycle (.25 x 5 volts = 1.25 volts average)
    //0 is a 0% duty cycle (0 x 5 volts = 0 volts average)
}

void loop() {

  analogWrite(led_pin,brightness);

}
```

Figure 4. Code to change the brightness of an LED using PWM.

Part 2: Fading an LED

Pulse width modulation can also be used to "fade" an LED. Fading is when the LED slowly gets dimmer until it is off and then slowly gets brighter again. To fade an LED set up the hardware as shown in figure 3 and use the code in figure 5.

```
int led_pin = 11;

void setup() {

  pinMode(led_pin, OUTPUT);

}
void loop() {

  //Fading the LED
  for(int i=0; i<255; i++){
    analogWrite(led_pin, i);
    delay(5);
  }

  for(int i=255; i>0; i--){
    analogWrite(led_pin, i);
    delay(5);
  }
}
```

Figure 5. Code to fade an LED.

Part 3: Controlling a Servo Motor

Controlling motors is important in many real-world applications such as robotics. Servo motors are used for precision movements and can move from 0° to 180° so they can only move within a half-circle. For the next portion of the workshop, you will be making a servo move to specific positions (0°, 90°, 180°).

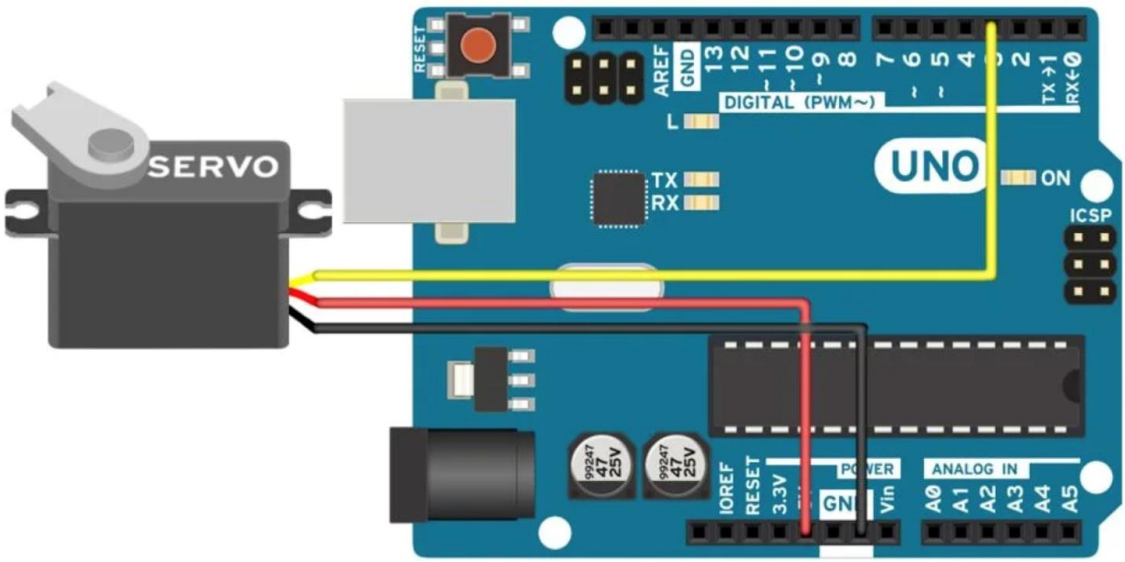

Figure 6. Hardware setup for a servo motor. For our servos red wire is connected to 5V, the brown wire is connected to ground, and the orange wire is connected to pin 3.

```
#include <Servo.h>

Servo Servo1;

void setup() {

   Servo1.attach(3);

}
void loop(){
   // Make servo go to 0 degrees
   Servo1.write(0);
   delay(1000);
   // Make servo go to 90 degrees
   Servo1.write(90);
   delay(1000);
   // Make servo go to 180 degrees
   Servo1.write(180);
   delay(1000);
}
```

Figure 7. Code to move the servo to 0°, 90°, and 180°.

Part 4: Sweeping a Servo Motor

Next, we will sweep the servo motor by changing its position to go from 0° to 180° and back again. To do this, we will use the same hardware setup in figure 6 above and the code from figure 8 below.

```
#include <Servo.h>

Servo myservo;

int pos = 0;
void setup() {
  myservo.attach(3);  // attaches the servo on pin 3 to the servo object
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) {
    // in steps of 1 degree
    myservo.write(pos);
    delay(5);
  }
  for (pos = 180; pos >= 0; pos -= 1) {
    myservo.write(pos);
    delay(5);
  }
}
```

Figure 8. Code to sweep a servo motor.