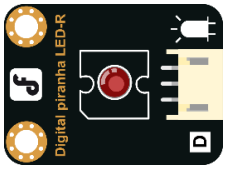
Project 5: Fading LED

In this chapter, instead of simply making the LED blink, we will learn how to adjust the brightness and add fading effect.

To achieve this, we are going to introduce the concept of PWM (pulse width modulation) and explain how it modulates analog signal by digital signal.

COMPONMENT LIST:

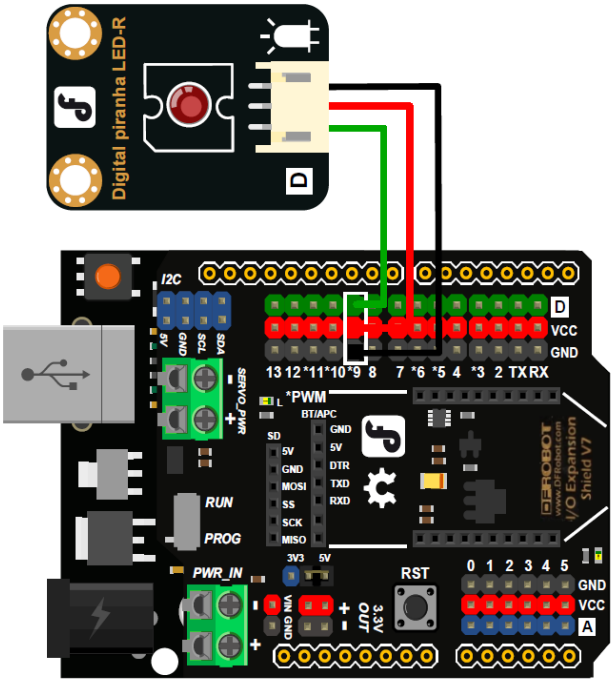
1× Digital Piranha LED-R



HARDWARE CONNECTIONS

Attach the Digital Piranha LED-R to digital pin 9.

Use the diagram below for reference:



Be sure that your power, ground and signal connections are correct or you risk damaging your components.

When you have connected the components and checked your connections, plug the 101 in to your PC with the USB cable so you can upload a program.

CODE INPUT

Sample Code 5-1:

```
int ledPin = 9;

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

void loop(){
  for (int value = 0 ; value < 255; value=value+1){
     analogWrite(ledPin, value);
     delay(5);
  }
  for (int value = 255; value >0; value=value-1){
     analogWrite(ledPin, value);
     delay(5);
  }
}
```

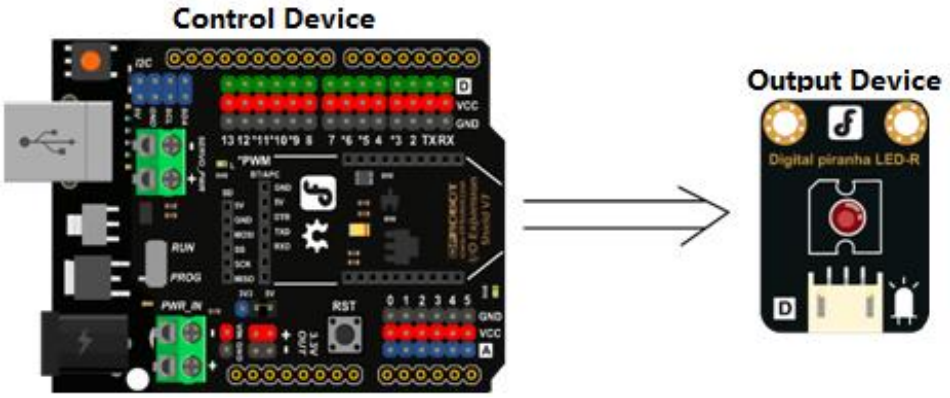
Use this sample code to implement the behavior we want.
You can copy and paste it in to the Arduino IDE, but if you want to develop you skills we recommend typing it out.

When you have finished, click "Verify" to check the code for syntax errors. If the code verifies successfully, you can upload it to your 101.

Once the code has uploaded, you should see the LED gradually brightening and dimming.

HARDWARE ANALYSIS (ANALOG OUTPUT)

This device is similar to experiment 1 where we made an LED blink. We also only have one output unit. The difference in this project is that we are using the LED for analog output, rather than digital output. Refer to the code section for more information.



CODE ANALYSIS

When we need to loop one line of code, we can use the "for" structure.

"for" loops are written in the following structure:

```
for(init; condition; update){
    statement(s);
}
```

Here is the flow:

1. The "init" statement is executed first and only once.
2. The "condition" is evaluated. If it is false, stop the loop.
3. The "statement(s)" is executed.
4. The "update" is executed. Go back to step 2.

Let's return to our code:

```
for (int value = 0; value < 255; value=value+1){
        ...
}
for (int value = 255; value >0; value=value-1){
        ...
}
```

These two “for” loops make the value increase from 0 to 255 and then decrease from 255 to 1. Now let’s take a look at the function analogWrite().

Digital pins only have two values: 0 or 1. How can we send an analog value to a digital pin? We can use this function. Four digital pins on the 101 are marked with the symbol “~”, which means they can send out PWM signal.

The function is written in the following form:

analogWrite(pin,value)

The function analogWrite() is used to write an analog value between 0 and 255 to the PWM pins. analogWrite() can only be used for PWM pins, which are pins 3, 5, 6, 9 on an 101.

PWM stands for pulse width modulation. Very simply, PWM allows us to achieve analog results with digital signals by switching the signals HIGH and LOW very rapidly. Using this, we can simulate voltages between 0v and 5v.
If you were to look at the PWM signal on an oscilloscope, you would see a square wave. The top of the square is the HIGH signal, and the bottom is LOW. The greater the width of the square, the longer the HIGH pulse – hence the name pulse width modulation.

Let’s explore PWM in more detail by examining five square waves.

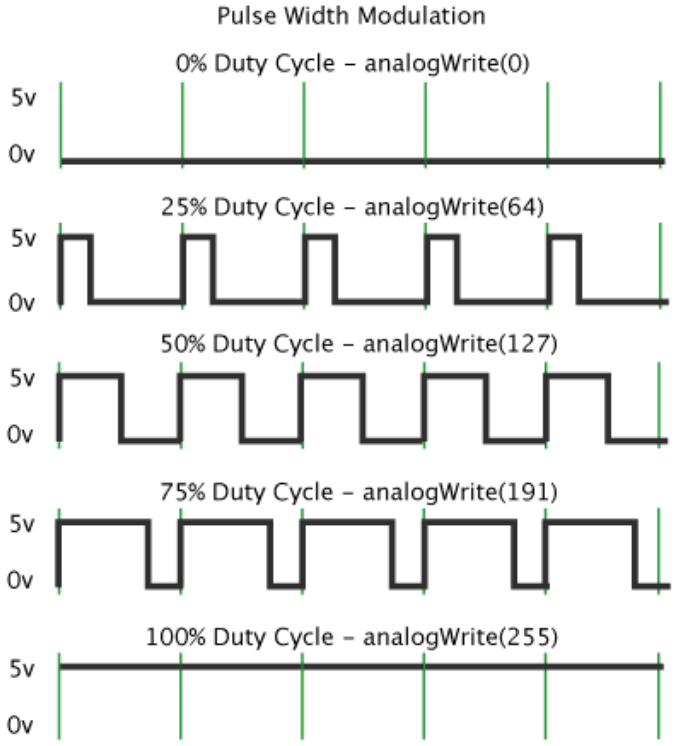


Fig 4-2 PWM Illustrations

Green vertical lines on the graph represent an interval.The black line represents the electric pulse. Each value included in the function of "analogWrite(value)" can be mapped to a certain percentage, which is also called Duty Cycle.The duty cycle refers the length of time the pulse goes from LOW to HIGH and HIGH to LOW. The duty cycle of the first square wave is 0% and the corresponding value is 0. If the duty cycle is 100%, the analog value will be 255. The longer the duty cycle, the brighter the light is.

To help you better understand, we can try to manually set the duty cycle by setting the time length of HIGH digital signal in each cycling period. When the period goes short, which makes LED blink fast enough, we will perceive it as brightness.

CODE INPUT

```
int ledPin = 10;
void setup() {
     pinMode(ledPin,OUTPUT);
}
void loop(){
  for (int duration = 0 ; duration < 20; duration=duration+1){
     digitalWrite(ledPin, HIGH);
     delay(duration);
     digitalWrite(ledPin, LOW);
     delay(20-duration);
}
  for (int duration = 20 ; duration > 0; duration=duration-1){
     digitalWrite(ledPin, HIGH);
     delay(duration);
     digitalWrite(ledPin, LOW);
     delay(20-duration);
  }
}
```

PWM's electronic applications include adjusting the brightness of an LED, or varying the speed of a DC motor. Can you think of any other applications for it?

With the same hardware connections we've used in this session, the LED can achieve completely different effects by using a different code.
Have a play with the code and see if you can make different lighting effects. Use your imagination and have fun with it!