

## Project 2: Adjust LED Brightness

### 1.Description

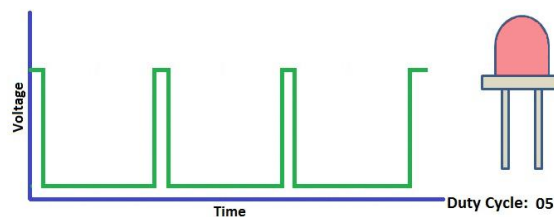
In previous lesson, we control LED on and off and make it blink.

In this project, we will control LED' s brightness through PWM simulating breathing effect.

PWM is a means of controlling the analog output via digital means.

Digital control is used to generate square waves with different duty cycles (a signal that constantly switches between high and low levels) to control the analog output. In general, the input voltages of ports are 0V and 5V.

What if the 3V is required? Or a switch among 1V, 3V and 3.5V? We cannot change resistors constantly. For this reason, we resort to PWM.



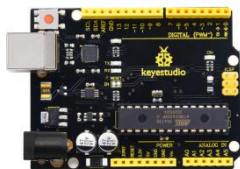
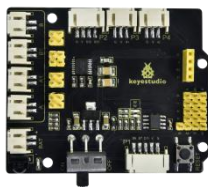



For the Arduino digital port voltage output, there are only LOW and HIGH, which correspond to the voltage output of 0V and 5V. You can define

# keystudio

LOW as 0 and HIGH as 1, and let the Arduino output five hundred 0 or 1 signals within 1s.

If all of the output five hundred are 1, that is 5V; if all of which are 0, that is 0V. If output 010101010101 in this way then the output port is 2.5V, which is like showing movie. The movie we watch are not completely continuous. It actually outputs 25 pictures per second. In this case, the human can't see it, neither does PWM. If we want different voltage, we need to control the ratio of 0 and 1. The more 0,1 signals output per unit time, the more accurate the control.

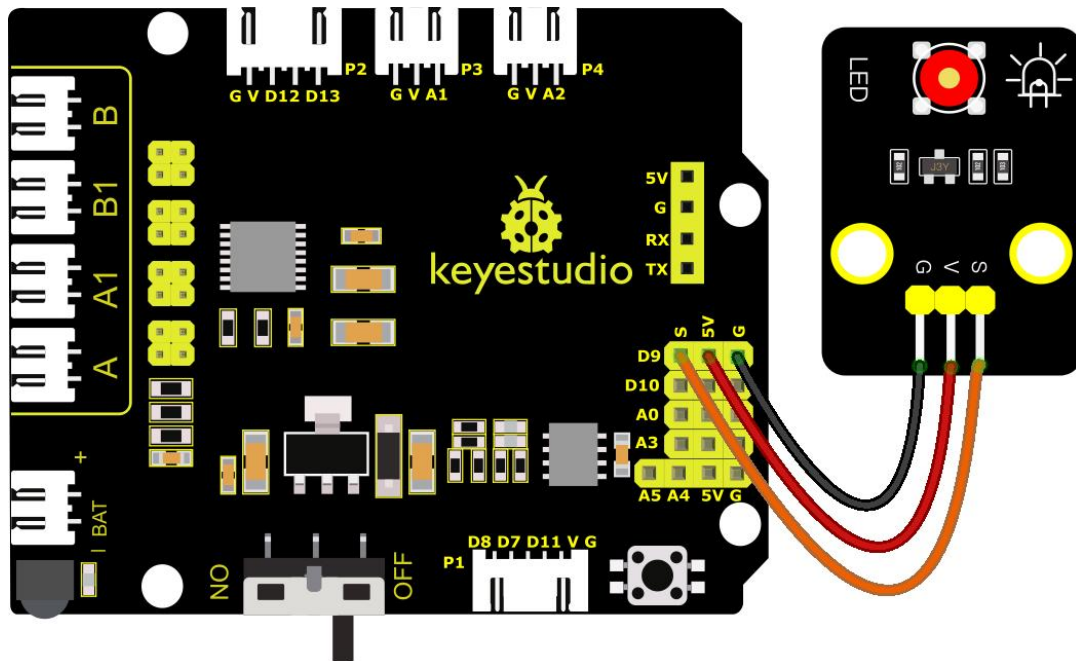
## 2.Components

Keystudio 4.0 Development Board *1	Keystudio 8833 Motor Driver Expansion Board *1	Red LED Module*1
		
3P F-F Dupont Wire*1	USB Cable*1	
		

# keyestudio

## 3.Wiring Diagram

Keep the wiring-up unchanged.



## 4.Test Code

```
/**
 *
 * keyestudio 4wd BT Car
 * lesson 2.1
 * pwm
 * http://www.keyestudio.com
 */
int ledPin = 9; // Define the LED pin at D9
int value;

void setup () {
  pinMode (ledPin, OUTPUT); // initialize ledpin as an output.
}

void loop () {
  for (value = 0; value <255; value = value + 1)
```

# keystudio

```
{
  analogWrite (ledPin, value); // LED lights gradually light up
  delay (5); // delay 5ms
}
for (value = 255; value > 0; value = value-1)
{
  analogWrite (ledPin, value); // LED gradually goes out
  delay (5); // delay 5ms
}
}
//*****
```

## 5.Test Result

After successfully uploading the code to the V4.0 board, connect the wirings according to the wiring diagram, and use a USB cable to connect the computer to power the board. After powering on, you will see that the LED gradually changes from bright to dark, like human's breath, rather than turning on and off immediately.

## 6.Code Explanation

If we need to repeat a certain statement, we could use for statement.

For statement format is shown below:

```
for (① cycle initialization; ② condition is true ④ cycle adjustment statement) {
  ③ loop body statement;
}
```

# keystudio

FOR cyclic sequence:

Round 1: 1 → 2 → 3 → 4

Round 2: 2 → 3 → 4

...

Until number 2 is not established, "for" loop is over,

After knowing this order, go back to code:

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

The two "for" statements make value increase from 0 to 255, then reduce from 255 to 0, then increase to 255....infinitely loop

There is a new function in the following ----- analogWrite()

We know that digital port only has two state of 0 and 1. So how to send an analog value to a digital value? Here, this function is needed. Let's observe the Arduino board and find 6 pins marked "~" which can output PWM signals.

Function format as follows:

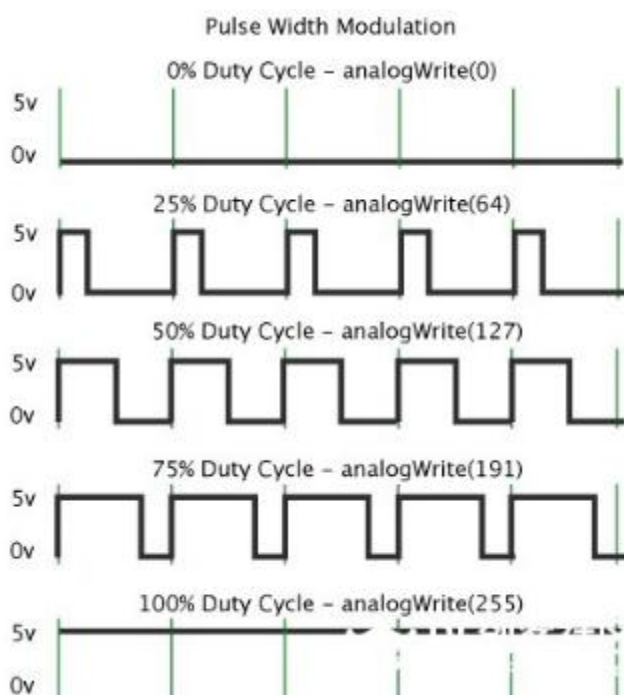
**analogWrite(pin,value)**

# keyestudio

`analogWrite()` is used to write an analog value from 0~255 for PWM port, so the value is in the range of 0~255. Attention that you can only write the digital pins with PWM function, such as pin 3, 5, 6, 9, 10, 11.

PWM is a technology to obtain analog quantity through digital method. Digital control forms a square wave, and the square wave signal only has two states of turning on and off (that is, high or low levels). By controlling the ratio of the duration of turning on and off, a voltage varying from 0 to 5V can be simulated. The time turning on (academically referred to as high level) is called pulse width, so PWM is also called pulse width modulation.

Through the following five square waves, let's learn more about the PWM.



# keyestudio

In the above figure, the green line represents a period, and the value of `analogWrite()` corresponds to a percentage which is called Duty Cycle as well. Duty cycle implies the ratio of time occupied by the high level in the cycle. From top to bottom, the duty cycle of first square wave is 0% and its corresponding value is 0.

The LED brightness is lowest, that is, light off. The more time the high level lasts, the brighter the LED. Therefore, the last duty cycle is 100%, which corresponds to 255, and LED is the brightest. And 50% is the brightest half, 25% means darker.

PWM is more used for adjusting the LED' s brightness or the rotation speed of motors.

It plays a vital role in controlling smart robot cars. I believe that you cannot wait to learn the next project.

## 7.Extension Practice

Let' s modify the value of delay and remain the pin unchanged, then observe how the LED changes.

```
//*****  
/*  
keyestudio 4wd BT Car  
lesson 2.2  
pwm  
http://www.keyestudio.com  
*/  
int ledPin = 9; // Define the LED pin at D9  
void setup () {  
    pinMode(ledPin, OUTPUT); // initialize ledpin as an output.
```

# keyestudio

```
}

void loop () {
  for (int value = 0; value < 255; value = value + 1) {
    analogWrite (ledPin, value); // LED lights gradually light up
    delay (30); // delay 30MS
  }
  for (int value = 255; value > 0; value = value - 1) {
    analogWrite (ledPin, value); // LED gradually goes out
    delay (30); // delay 30MS
  }
}

//*****
```

Upload the code to the development board, then the LED will blink more slowly.