Project 7

Pulsating Lamp

Project 7 - Pulsating Lamp

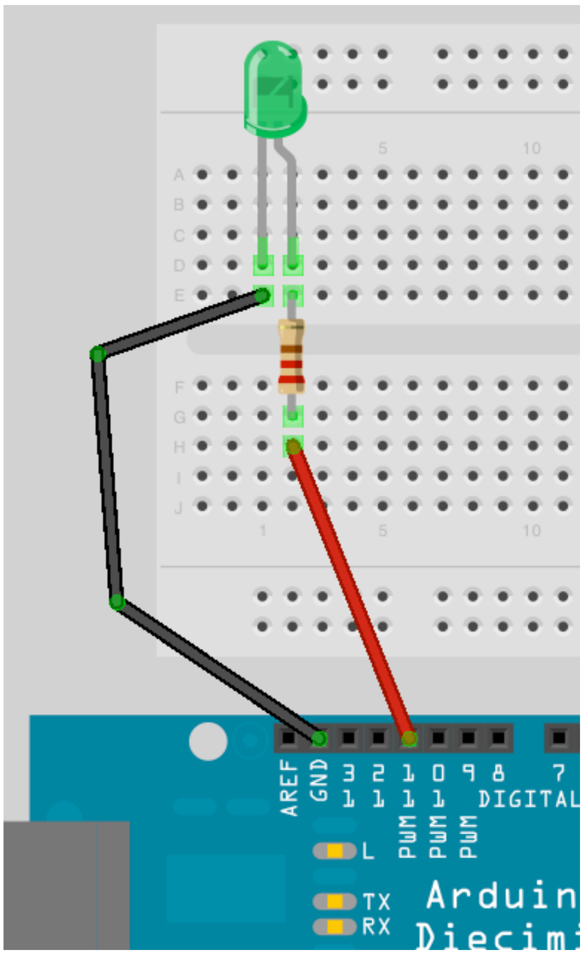
We are now going to delve further into a more advanced method of controlling LED's. So far we have simply turned the LED on or off. How about being able to adjust it's brightness too? Can we do that with an Arduino? Yes we can.

Time to go back to basics.

What you will need

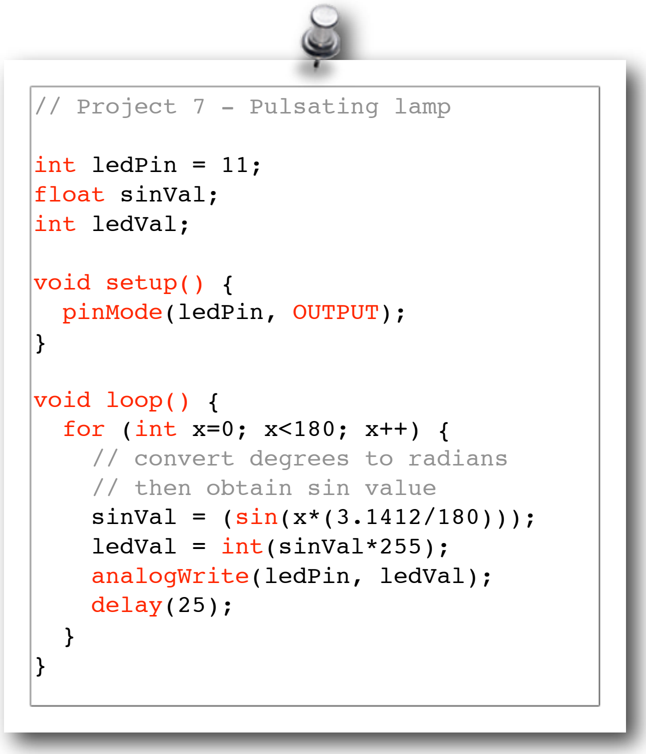
Green Diffused LED	
220Ω Resistor	

Connect it up



Enter the code

Enter this simple program.



```
// Project 7 - Pulsating lamp

int ledPin = 11;
float sinVal;
int ledVal;

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

void loop() {
  for (int x=0; x<180; x++) {
    // convert degrees to radians
    // then obtain sin value
    sinVal = (sin(x*(3.1412/180)));
    ledVal = int(sinVal*255);
    analogWrite(ledPin, ledVal);
    delay(25);
  }
}
```

Verify and upload. You will now see your LED pulsate on and off steadily. Instead of a simple on/off state we are now adjusting it's brightness. Let's find out how this works.

Project 7 - Code Overview

The code for this project is very simple, but requires some explanation.

```
// Project 7 - Pulsating lamp

int ledPin = 11;
float sinVal;
int ledVal;

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

void loop() {
  for (int x=0; x<180; x++) {
    // convert degrees to radians
    // then obtain sin value
    sinVal = (sin(x*(3.1412/180)));
    ledVal = int(sinVal*255);
    analogWrite(ledPin, ledVal);
    delay(25);
  }
}
```

We first set up the variables for the LED Pin, a float (floating point data type) for a sine wave value and ledVal which will hold the integer value to send out to Pin 11.

The concept here is that we are creating a sine wave and having the brightness of the LED follow the path of that wave. This is what makes the light pulsate in that way instead of just fade up to full brightness and back down again.

We use the sin() function, which is a mathematical function to work out the sine of an angle. We need to give the function the degree in radians. We have a for loop that goes from 0 to 179, we don't want to go past halfway as this will take us into negative values and the brightness value we need to put out to Pin 11 needs to be from 0 to 255 only.

The sin() function requires the angle to be in radians and not degrees so the equation of x*(3.1412/180) will convert the degree angle into radians. We then transfer the result to ledVal, multiplying it by 255 to give us our value. The result from the sin() function will be a number between -1 and 1 so we need to multiply that by 255 to give us our maximum brightness. We 'cast' the floating point value of sinVal into an integer by the use of int() in the statement

```
ledVal = int(sinVal*255);
```

Then we send that value out to Digital Pin 11 using the statement

```
analogWrite(ledPin, ledVal);
```

But, how can we send an analog value to a digital pin? Well, if we take a look at our Arduino and look at the Digital Pins you can see that 6 of those pins (3, 5, 6, 9, 10 & 11) have PWM written next to them. Those pins differ from the remaining digital pins in that they are able to send out a PWM signal.

PWM stands for Pulse Width Modulation. PWM is a technique for getting analog results from digital means. On these pins the Arduino sends out a square wave by switching the pin on and off very fast. The pattern of on/offs can simulate a varying voltage between 0 and 5v. It does this by changing the amount of time that the output remains high (on) versus off (low). The duration of the on time is known as the 'Pulse Width'.

For example, if you were to send the value 0 out to Pin 11 using analogWrite() the ON period would be zero, or it would have a 0% Duty Cycle. If you were to send a value of 64 (25% of the maximum of 255) the pin would be ON for 25% of the time and OFF for 75% of the time. The value of 191 would have a Duty Cycle of 75% and a value of 255 would have a duty cycle of 100%. The pulses run at a speed of approx. 500Hz or 2 milliseconds each.

So, from this we can see in our sketch that the LED is being turned on and off very fast. If the Duty Cycle was 50% (a value of 127) then the LED would pulse on and off at 500Hz and would display at half the maximum brightness. It is basically an illusion that we can use to our advantage by allowing us to use the digital pins to output a simulated analog value to our LED's.

Note that even though only 6 of the pins have the PWM function, you can easily write software to give a PWM output from all of the digital pins if you wish.

Later on we will revisit PWM as we can utilise it to create audible tones using a piezo sounder.