Pulse Width Modulation

The method used by the RPi.GPIO library to produce an "analog" output is called Pulse Width Modulation (PWM). The GPIO pin actually uses a digital output, but generates a series of pulses. The width of the pulses are varied. The larger the proportion of the time that the pulse stays high, the greater the power delivered to the output, and hence the brighter the LED as shown in Figure 9-6.

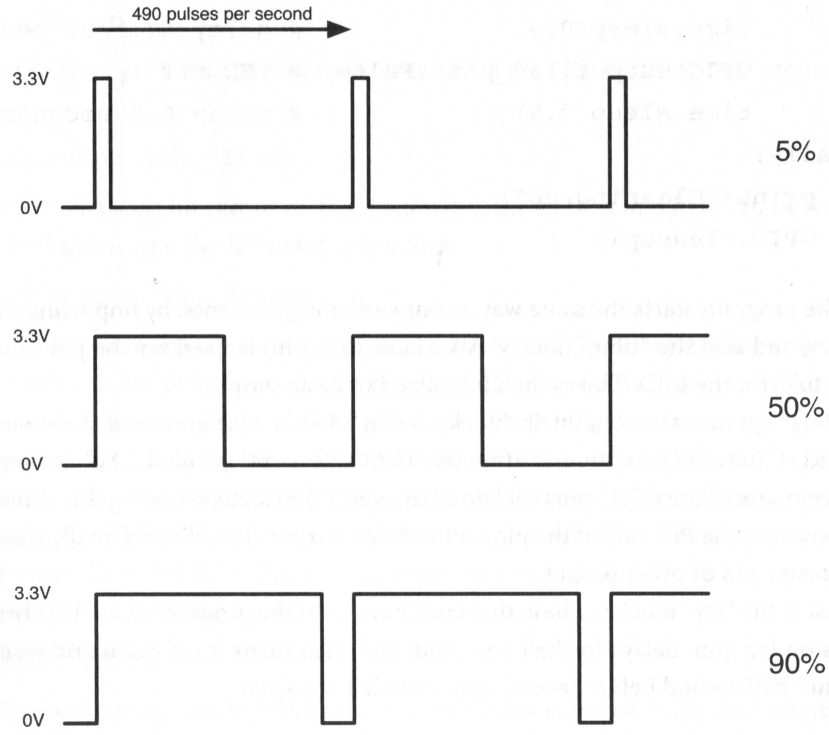

Figure 9-6 *Pulse Width Modulation.*

The proportion of the time that the pulse is HIGH is called the duty and this is often expressed as a percentage.

Even though the LED is actually turning on and off, it happens so fast that your eye is just fooled into thinking the LED is brighter or dimmer depending on the length of the PWM pulse.

With the LED connected to pin 18 as you did in Figure 9-5, open the program 09_pwm.py in IDLE and then run it. Use IDLE3 for Python 3 and run it as superuser as you did in the previous section. The first thing the program does is to prompt you for a brightness level between 1 and 100 as shown below:

```
Enter Brightness (0 to 100):0
Enter Brightness (0 to 100):50
Enter Brightness (0 to 100):100
```

Try a few different values and see how the brightness of the LED changes.

Setting up a GPIO output to do PWM is a little different from using it as a simple on/off digital output.

```
import RPi.GPIO as GPIO

led_pin = 18
GPIO.setmode(GPIO.BCM)
GPIO.setup(led_pin, GPIO.OUT)

pwm_led = GPIO.PWM(led_pin, 500)
pwm_led.start(100)

try:
    while True:
      duty_s = input("Enter Brightness (0 to 100):")
      duty = int(duty_s)
      pwm_led.ChangeDutyCycle(duty)
finally:
    print("Cleaning up")
    GPIO.cleanup()
```

After setting the pin to be an output as normal, you then need to create a PWM channel using the line below:

```
pwm_led = GPIO.PWM(led_pin, 500)
```

The second parameter (500) specifies the number of pulses per second. Having created the channel, PWM is now started at 100 percent on using the line:

```
pwm_led.start(100)
```