Experiment 2: Increase Brigthness of LED while Button is pushed

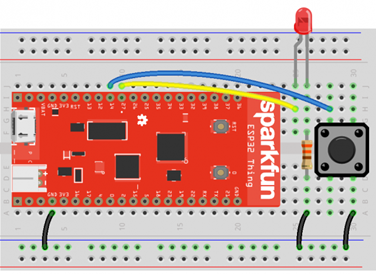
Pulse width modulation is the technique of switching a digital signal on and off very quickly to control a variety of electrical components.

By varying the amount of time a signal is on vs. off, you can vary the amount of electrical power provided to a component. We can use this, for example, to dim an LED or control the speed of a motor. To learn more about PWM, see [this tutorial](https://learn.sparkfun.com/tutorials/pulse-width-modulation).

In this example, we are going to

create a simple LED animation that occurs as long as a button is held down.

Hardware Connections



**⚡ Note:** It matters how you plug in your LED! Current can only flow in one direction through an LED, so pay careful attention to the leads. The short lead on the LED should be connected on the same row as the 330Ω resistor.

|  |  |
| --- | --- |
| [LED polarity diagram](https://cdn.sparkfun.com/assets/learn_tutorials/6/5/9/LED_drawing_01.png) | [How a push button is configured on the inside](https://cdn.sparkfun.com/assets/learn_tutorials/6/5/9/push_button_diagram_dk.png) |

**Note:** Buttons can be a little weird, if it's the first time you've used them. The pins across from each other are always connected, whereas the pins on the same side are only connected when you push the button.

**Note:** If you are using the [full-size breadboard](https://www.sparkfun.com/products/12615), the power rails are divided in the middle. This means that to get power to the whole power row, you will need to connect the two halves. See the picture below to see how to use jumper wires to connect the halves of the power rows.

Code: LED Animation

COPY CODE**import** machine

**import** sys

**import** utime

# Pin definitions

repl\_button **=** machine.Pin(0, machine.Pin.IN, machine.Pin.PULL\_UP)

repl\_led **=** machine.Pin(5, machine.Pin.OUT)

button **=** machine.Pin(14, machine.Pin.IN, machine.Pin.PULL\_UP)

pwm\_pin **=** machine.Pin(27, machine.Pin.OUT)

# Create a PWM object out of our pin object

pwm **=** machine.PWM(pwm\_pin)

# Slowly fade LED brightness

**while** True:

# If button 0 is pressed, turn on LED and drop to REPL

**if** repl\_button.value() **==** 0:

print("Dropping to REPL")

repl\_led.value(1)

sys.**exit**()

# Increase brightness of LED if button is held

**for** i **in** range(1024):

**if** button.value() **==** 0:

pwm.duty(i)

utime.sleep\_ms(2)

**else**:

pwm.duty(0)

Save the code with a name such as *pwm.py*. Open a command terminal on your host computer and navigate to the directory where you have *pwm.py* stored. Push button 0 on your ESP32 to ensure that it is in REPL mode, and enter the following commands into your terminal (don't forget to change <PORT> to your individual port number/location):

COPY CODEcp pwm.py main.py

ampy --port **<**PORT**>** put main.py

After the code has been uploaded, hold down the button (the one on the breadboard that is connected to pin 14). You should see the LED slowly brighten, turn off, and then begin the brightening animation over again as long as the button is held down.