LIAM FRASER

Liam is the creator of the RaspberryPiTutorials YouTube channel. He is currently studying Computer Science at the University of York and has a special interest in embedded systems.
liamfraser.co.uk

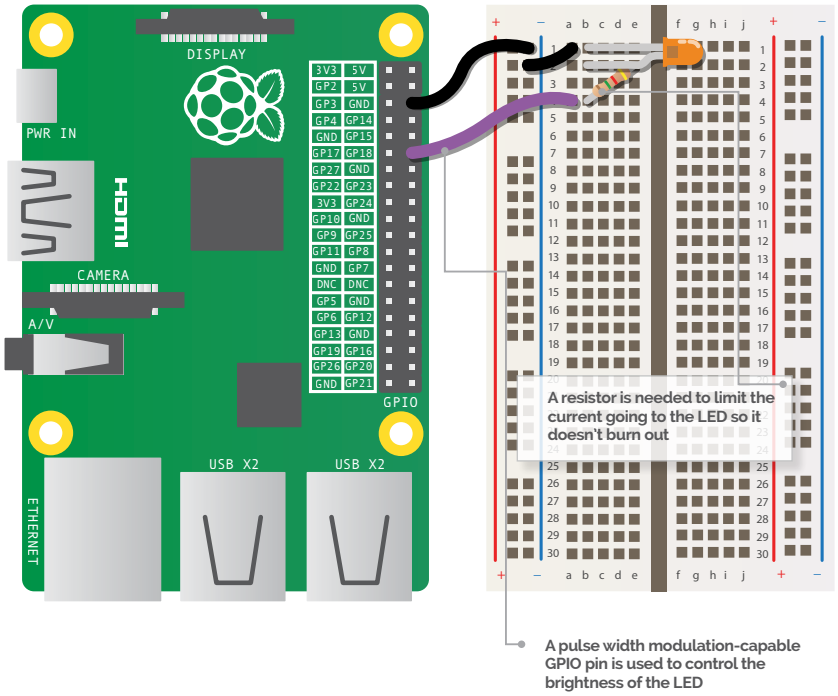
MAKE A PWM CANDLE LANTERN

You'll Need

- A coloured LED
- Breadboard
- Female-to-male jumper cables
- Resistor (100 ohm to 330 ohm)

Set a romantic mood with your Raspberry Pi by simulating a flickering candle effect using pulse-width modulation…

This tutorial is intended as a gentle – not to mention romantic – introduction to GPIO (general-purpose input and output) pins on your Raspberry Pi, and how to control them in Python. We'll be creating our romantic candle-like mood lighting using a random number generator to make an LED flicker at different intervals. In addition, its brightness will be varied using a technique called PWM (pulse-width modulation), which effectively controls what percentage of the time the LED is turned on. We will also take a look at the output of the pins on an oscilloscope, so that we can see how the code translates to the electrical signals that make things tick.



>STEP-01

Pick a resistor for your LED

A resistor will limit the current that flows through the LED. Different colour LEDs have different current limits, so you'll need to check the specifications where possible. 100 ohm or 220 ohm will definitely work, though your LEDs might end up being dimmer than usual. The equation for working out resistance is as follows:

R = (3.3V – LED VOLTAGE) / LED CURRENT

Our yellow LED needs a voltage of 1.8V – 2.2V and has a typical current of 20mA, so: **R = (3.3V – 2.0V) / 0.02** (which is 65 ohms). A resistor with a value between 65 and 130 ohms is ideal here, but a lower value will make your LED brighter.

>STEP-02

Setup the breadboard

Unplug your Pi and follow the breadboard illustration setup. Make sure you use the same GPIO pin we have, as only a couple are capable of pulse-width modulation (on the B+). We're using GPIO number 18 for PWM, which is described as **PCM_CLK/PWM0**.

The circuit path, as shown in the illustration, is **GPIO 18 > resistor > LED positive**. Finally, the LED negative leg goes into ground. The positive leg of an LED is usually longer. The negative side will have a flat edge rather than a circular one.

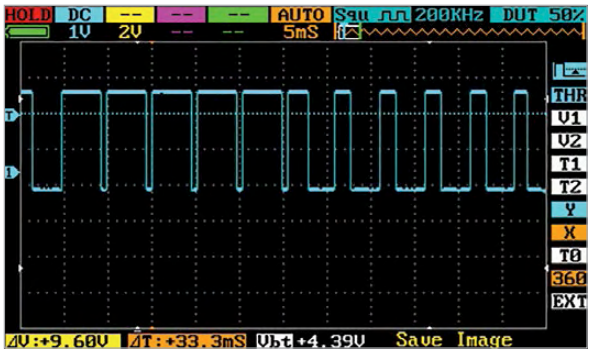
>STEP-03

Get coding!

Once you've wired up the project, power up your Pi and begin coding using an editor of your choice (or opening a terminal and typing **`nano candle.py`** will do).

Once we've imported the libraries we need, the **`setup`** function organises our program and starts PWM

Below This oscilloscope trace illustrates how the brightness of the LED is controlled



for us. The **flicker** function sets a random brightness by calling the **set_brightness** function, then sleeps for a random time. This function is then wrapped up in an infinite **while** loop within the **loop** function, which handles the all-important cleanup of the GPIO library when **CTRL+C** is pressed by the user.

>STEP-04

Test your creation

Exit your editor and run the code by typing **sudo python2 candle.py** into your terminal (you need root privileges to access the GPIO pins). Now that you've tested it, you can exit with **CTRL+C** and we'll make it run at boot. This way, the Pi can run headless and not need any user interaction.

At the terminal, type: **sudo nano /etc/rc.local**, then add the following line: **python2 /home/pi/candle.py &** (but make sure you put this in the line before **exit 0**). Don't forget to save the changes.

The ampersand means the script will go to the background and let the boot process continue. Notice how sudo isn't required because **rc.local** is executed as root. Reboot the Pi with **sudo reboot** to verify that it works.

>STEP-05

Packaging it up

Now that the script is started when the Pi boots, you could package it up into a nice container using a portable phone charger as a power supply. Arts and crafts are out of the scope of this tutorial, but there are plenty of candle holders that can be fashioned out of paper if you search the internet. Paper is ideal, especially with lots of holes in, since the LED probably isn't throwing out much light.

>STEP-06

Presentation, presentation, presentation

The candlelight project is surprisingly effective, but presentation is key in matters of the heart, so you may want to spruce up your project before you use it on a loved one. Pretty lanterns are available very cheaply from most department stores; just make sure you select one that obscures the view of the interior. If the lantern isn't big enough to fit the Pi and breadboard, solder the resistor to the LED and hide the Pi behind it.

Below A candle lantern that obscures the view of the inside is perfect for disguising the LED and hiding your Pi

Candle.py

Language >PYTHON

```
import RPi.GPIO as GPIO
import time
import random

# Set the PWM output we are using for the LED
LED = 18

def setup():
        global pwm

        # GPIO uses broadcom numbering (GPIO numbers)
        GPIO.setmode(GPIO.BCM)
        # Set the LED pin as an output
        GPIO.setup(LED, GPIO.OUT)

        # Start PWM on the LED pin at 200Hz with a
        # 100% duty cycle. At lower frequencies the LED
        # would flicker even when we wanted it on solidly
        pwm = GPIO.PWM(LED, 200)

        # Start at a brightness of 100%
        pwm.start(100)

def set_brightness(new_brightness):
        # Sets brightness of the LED by changing duty cycle
        pwm.ChangeDutyCycle(new_brightness)

def flicker():
    # We want a random brightness between 0% and 100%.
    # Then then we'll hold it for a random time
    # between 0.01 and 0.1 seconds to get a nice flicker
    # effect. Play with these values to make the effect
    # suit your liking
    set_brightness(random.randrange(0, 100))
    time.sleep(random.randrange(1, 10) * 0.01)

# The wrapper around the flicker function makes sure the
# GPIO hardware is cleaned up when the user presses CTRL-C

def loop():
    try:
        while True:
            flicker()
    except KeyboardInterrupt:
        pass
    finally:
        GPIO.cleanup()

# setup the hardware
setup()

# start the flickering
loop()
```