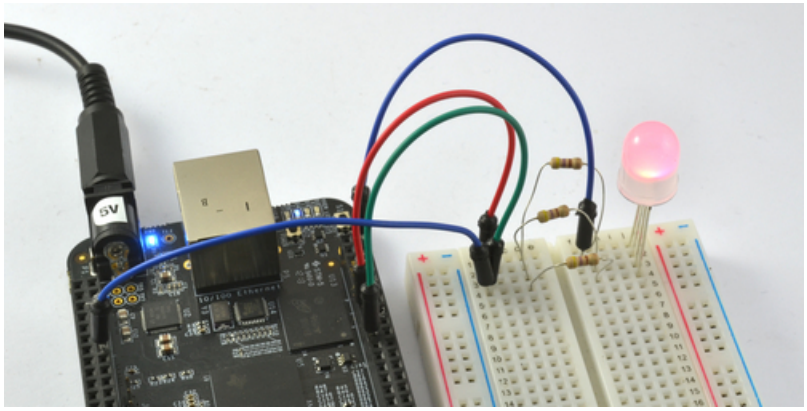
adafruit learning system

Fading a RGB LED on BeagleBone Black

Created by Simon Monk

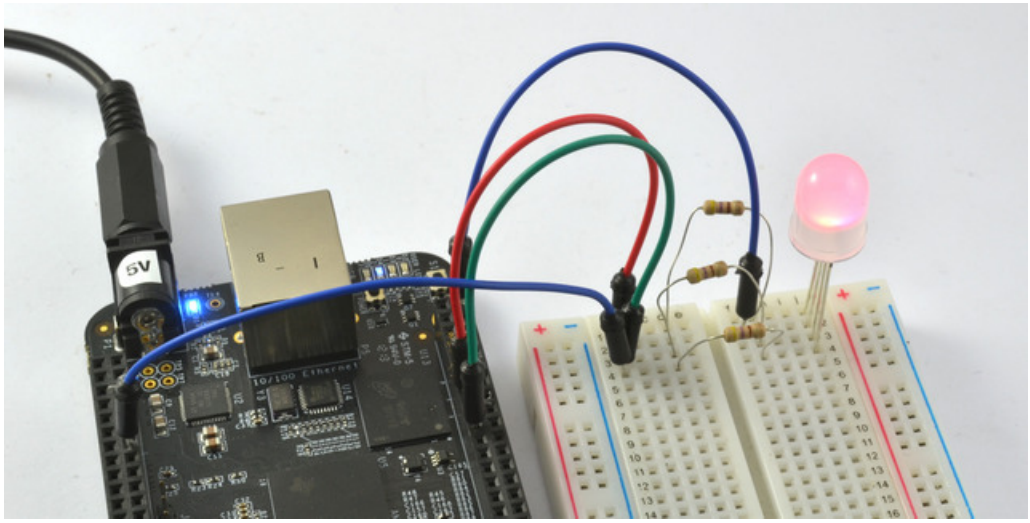


Last updated on 2018-08-22 03:36:28 PM UTC

Guide Contents

Overview

In this tutorial, you will learn how to control the color of an RGB LED using a BeagleBone Black (BBB).



Because the BBB runs Linux, there are many ways in which it can be programmed. In this tutorial we show how to control the power output of a GPIO pin and hence control the color of an RGB LED using Python.

The tutorial uses a technique called PWM (Pulse Wldth Modulation) to vary the brightness of each color channel between 0 and 100.

You will need

To make the project described in this tutorial, you will need the following:

BeagleBone Black

Diffused 10mm RGB LED common cathode or common anode

3 x appox. 470Ω resistors - you can use 220Ω to 1KΩ

Half-sized Breadboard

Male to male jumper leads

Installing the Python Library

This tutorial uses Ångström Linux, the operating system that comes pre-installed on the BBB.

Follow the instructions here, to install the Python IO BBIO library. http://learn.adafruit.com/setting-up-io-python-library-on-beaglebone-black (https://adafru.it/cgh)

Wiring

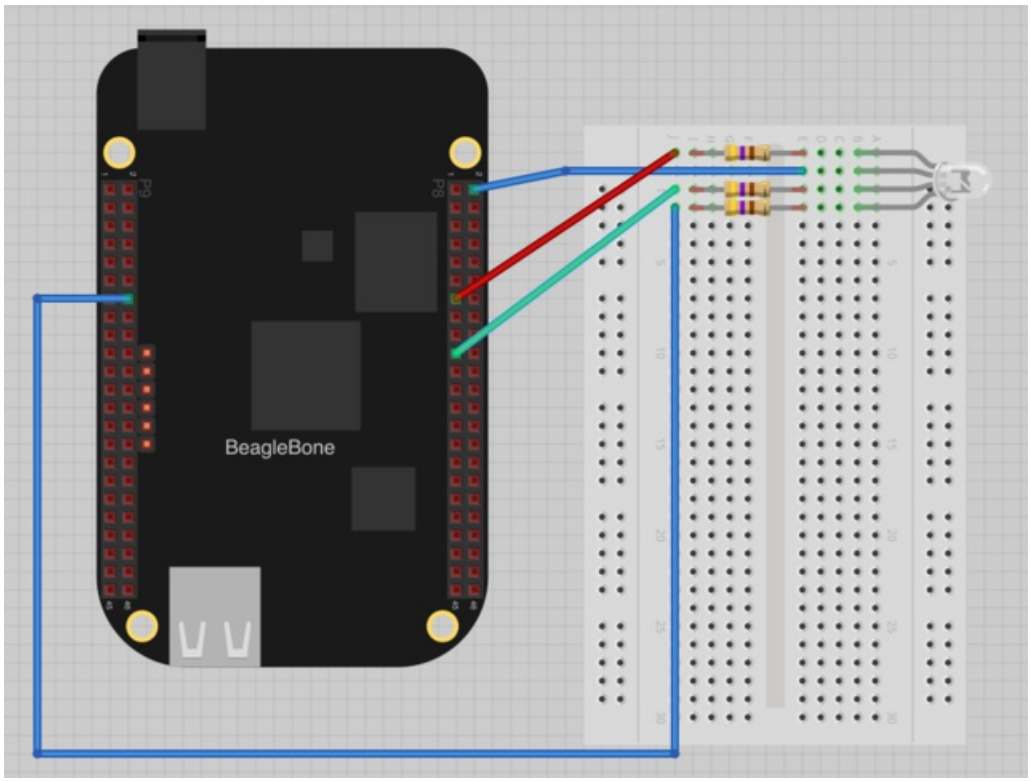
This tutorial is compatible with both common cathode and common anode LEDs.

Common anode LEDs have the positive connections of the red, green and blue LEDs connected together to one lead (the longest lead). A common cathode LED has the negative connections connected together (again, the longest lead).

Select the next step according to the type of RGB LED that you have.

Wiring (Common Cathode LED)

Wire up the breadboard using the header leads as shown below.



Push the LED leads into the breadboard, with the longest (common negative) lead on the second row of the breadboard. It does not matter which way around the resistors go.

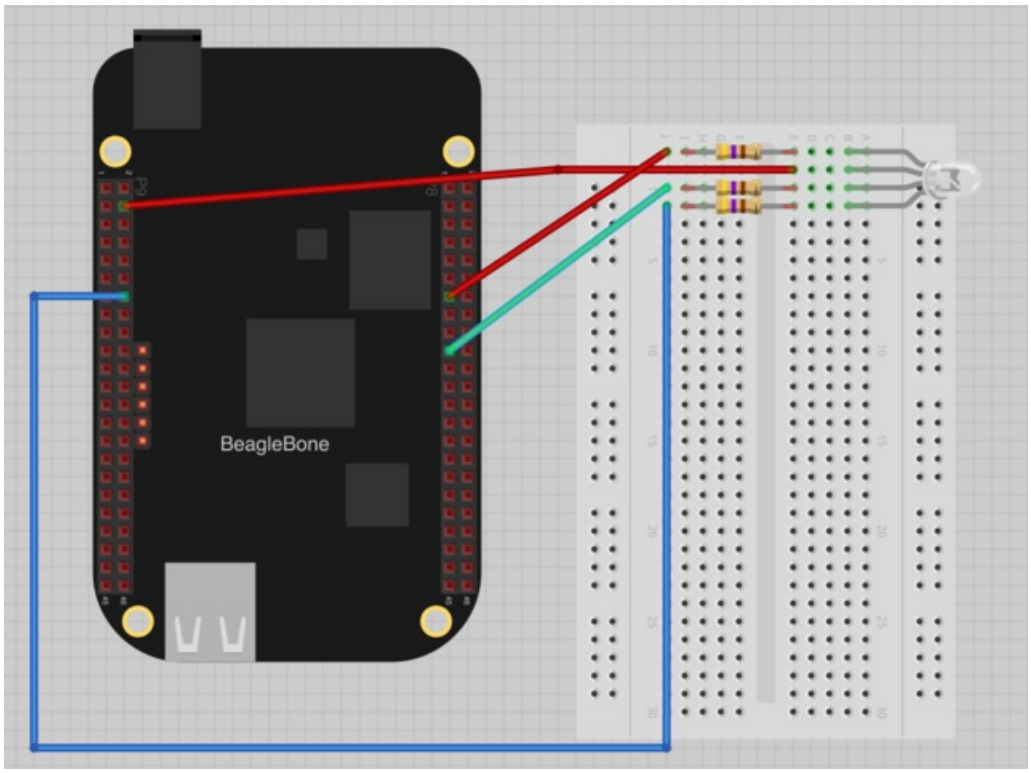
The top two connections on the BBB expansion header we are using (P8) are both GND. The three outputs to control the brightness of the red, green and blue channels are connected to sockets P8_13, P8_19 and P9_14 of the BBB.

The pins are numbered left to right, 1, 2 then on the next row down 3,4 etc. You can also use the pin P9_16 for PWM output.

You can find out about all the pins available on the P8 and P9 connecters down each side of the BBB here: http://stuffwemade.net/hwio/beaglebone-pin-reference/ (https://adafru.it/cgi)

Wiring (Common Anode LED)

Wire up the breadboard using the header leads as shown below.



Push the LED leads into the breadboard, with the longest (common positive) lead on the second row of the breadboard. It does not matter which way around the resistors go.

The common anode of the LED is connected to 3.3V. The three outputs to control the brightness of the red, green and blue channels are connected to sockets P8_13, P8_19 and P9_14 of the BBB.

The pins are numbered left to right, 1, 2 then on the next row down 3,4 etc. You can also use the pin P9_16 for PWM output.

You can find out about all the pins available on the P8 and P9 connecters down each side of the BBB here: http://stuffwemade.net/hwio/beaglebone-pin-reference/ (https://adafru.it/cgi)

The Python Console

Before writing a Python program to control an LED, you can try some experiments from the Python console to control the LED's color and make sure that everything is working.

To launch the Python Console type:

```
# python
Python 2.7.3 (default, Apr  3 2013, 21:37:23)
[GCC 4.7.3 20130205 (prerelease)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

First, we need to import the library, in particular its PWM feature that will allow us to control the brightness of the three color channels. so enter the command:

```
>>> import Adafruit_BBIO.PWM as PWM
```

To turn on the red component of the LED type:

```
>>> PWM.start("P8_13", 0)
```

Next turn on the Green by doing:

```
>>> PWM.start("P8_19", 0)
```

The LED should now look yellow / orange.

Finally lets turn on the blue channel using:

```
>>> PWM.start("P9_14", 0)
```

The LED should now be whitish in color as all three color channels are at full brightness.

Lets now turn all the LEDs off again by sending the following commands:

```
>>> PWM.stop("P8_13")
>>> PWM.stop("P8_19")
>>> PWM.stop("P9_14")
```

Writing a Program

To make the LED cycle through a range of colors, we can write a short program. First, exit the Python console by typing:

```
>>> exit()
```

This should take you back to the Linux prompt.

Enter the following command to create a new files called fade.py

```
nano fade.py
```

Now paste the code below into the editor window.

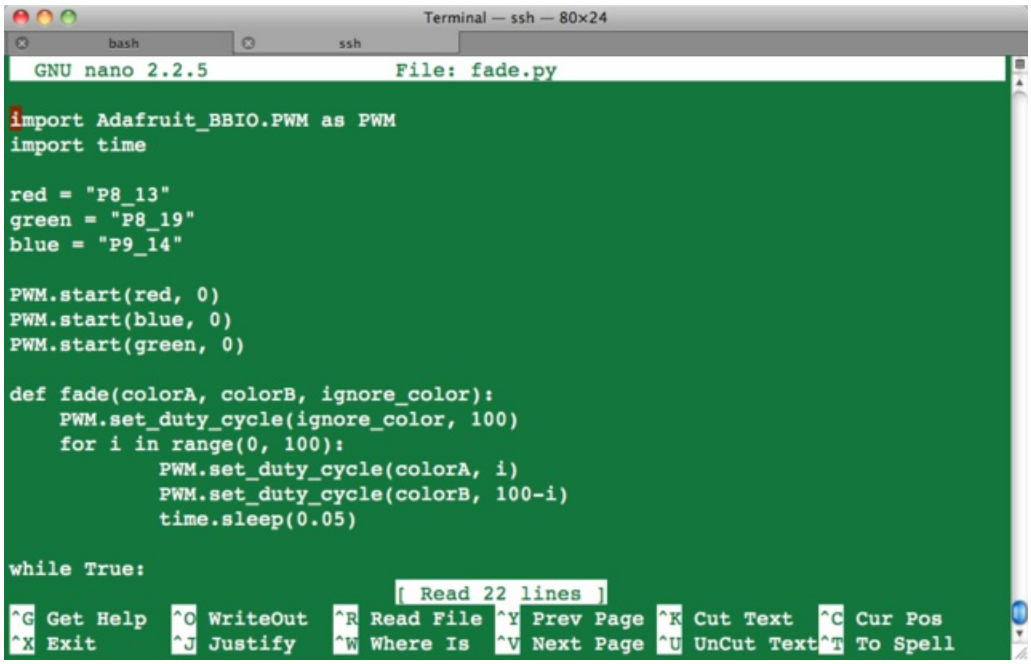
```
import Adafruit_BBIO.PWM as PWM
import time

red = "P8_13"
green = "P8_19"
blue = "P9_14"

PWM.start(red, 0)
PWM.start(blue, 0)
PWM.start(green, 0)

def fade(colorA, colorB, ignore_color):
    PWM.set_duty_cycle(ignore_color, 100)
    for i in range(0, 100):
     PWM.set_duty_cycle(colorA, i)
     PWM.set_duty_cycle(colorB, 100-i)
     time.sleep(0.05)

while True:
    fade(red, green, blue)
    fade(green, blue, red)
    fade(blue, red, green)
```



Save and exit the editor using CTRL-x and the Y to confirm.

To start the program, enter the command:

```
$ python fade.py
```

When you want to stop the program, use CTRL-c.

You will notice that the LED will remain frozen in its last color. Since we are not stopping the PWM channels, they will run in the background.

The program fades colors in pairs. First from red to green, then from green to blue and finally from blue to red, before continuing the cycle again.

The set_duty_cycle changes the brightness of the led color. In the Adafruit_BBIO.PWM library, brightest is 0 and 100 means the LED for that color is off.

The program above assumes that you are using a common cathode LED. If you are using a common anode LED, then change these lines:

```
PWM.set_duty_cycle(ignore_color, 100)

PWM.set_duty_cycle(colorA, i)
PWM.set_duty_cycle(colorB, 100-i)
```

to look like this:

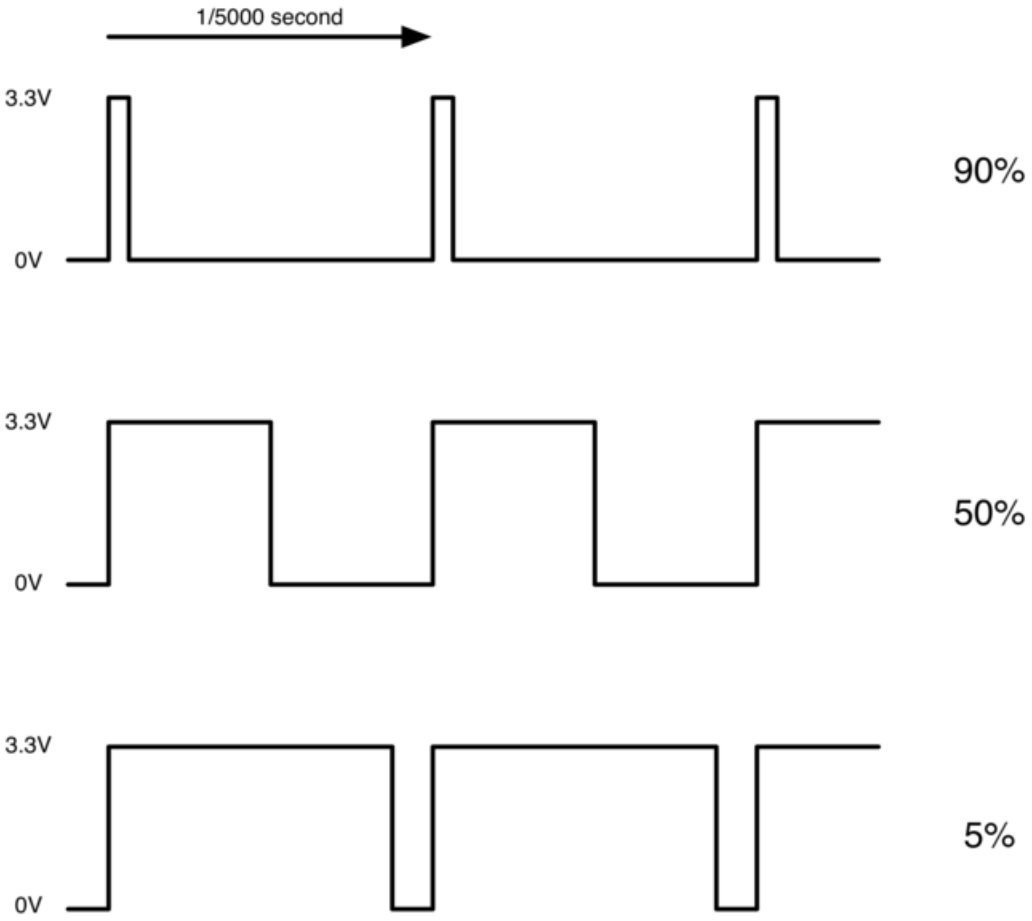
```
PWM.set_duty_cycle(ignore_color, 0)

PWM.set_duty_cycle(colorA, 100-i)
PWM.set_duty_cycle(colorB, i)
```

PWM

Pulse Width Modulation (or PWM) is a technique for controlling power. We also use it here to control the brightness of each of the LEDs.

The diagram below shows the signal from one of the GPIO pins on the BBB.



Every 1/5000 of a second, the PWM output will produce a pulse from 3.3V down to 0V. The length of this pulse is controlled by the 'set_duty_cycle' function.

If the output is high for 90% of the time then the load will get 90% of the power delivered to it. We cannot see the LEDs turning on and off at that speed, so to us, it just looks like the brightness is different.

Next Steps

You could try using analog readings, say from light or temperature to control the color of the LED. Perhaps, just set the red channel to 50% duty cycle and use a temperature reading to control the blue channel and the light reading to control the duty cycle of the green channel.

You can find tutorials for sensing temperature and light here (https://adafru.it/ciS) and here (https://adafru.it/ciT).

About the Author.
As well as contributing lots of tutorials about Raspberry Pi, Arduino and now BeagleBone Black, Simon Monk writes books about open source hardware. You will find his books for sale here (https://adafru.it/caH) at Adafruit.