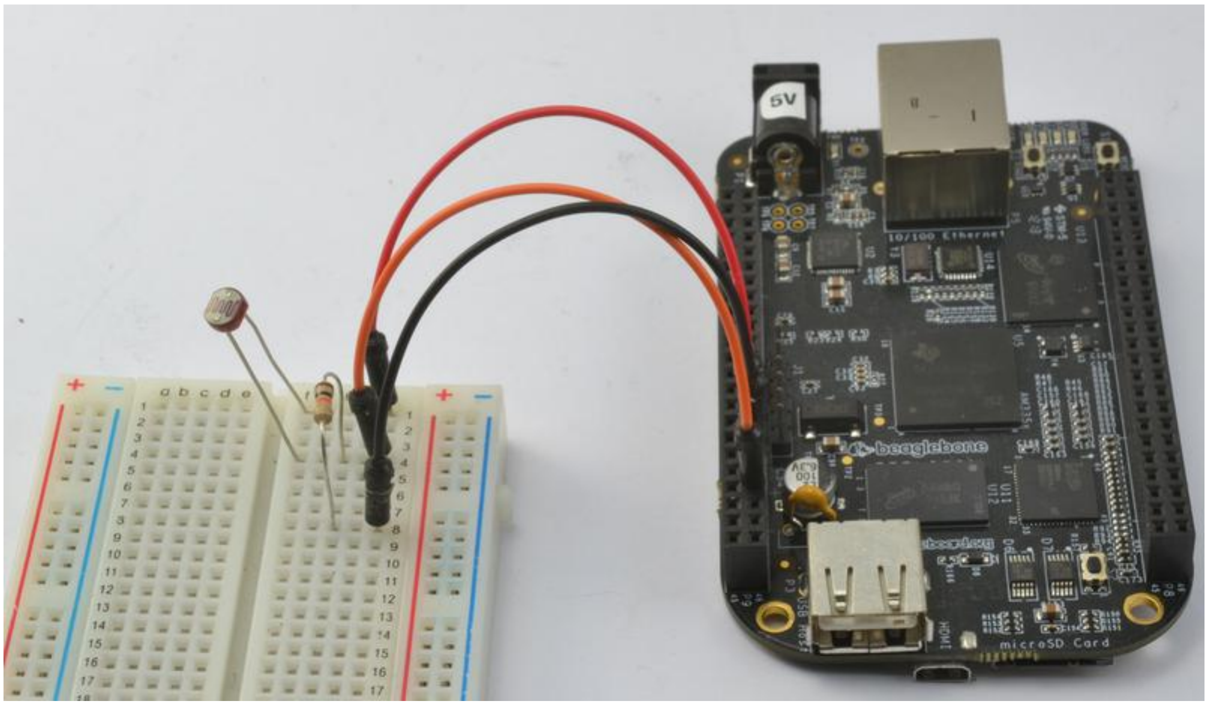
Measuring Light with a BeagleBone Black

Created by Simon Monk



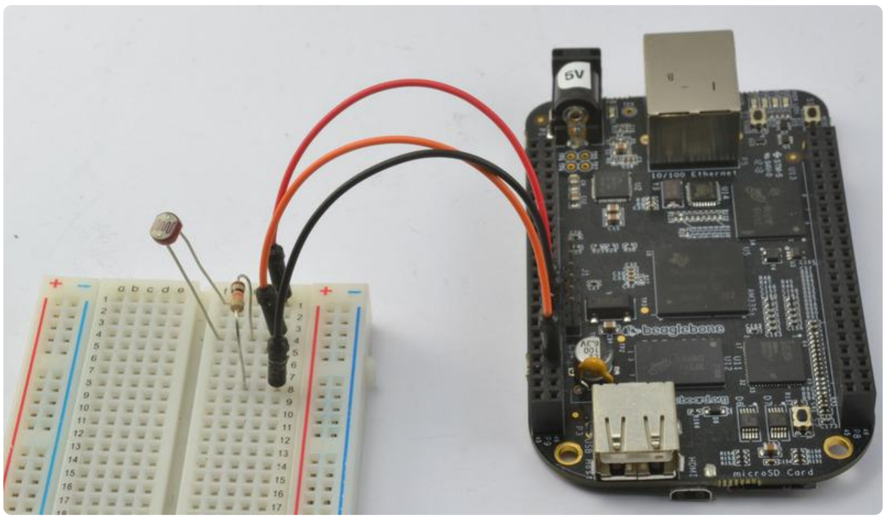
https://learn.adafruit.com/measuring-light-with-a-beaglebone-black

Last updated on 2021-11-15 06:02:05 PM EST

Table of Contents

Overview

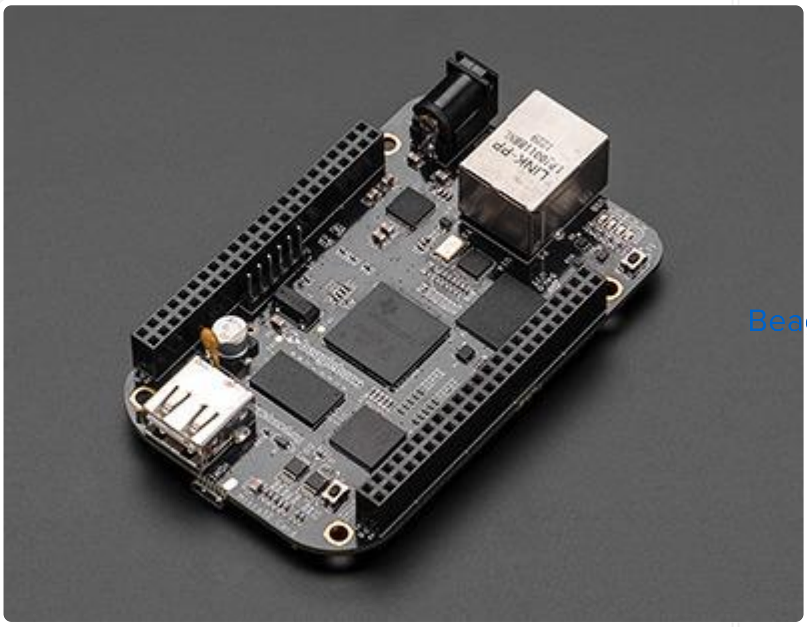
In this tutorial, you will learn how to connect a photoresistor to a BeagleBone Black.

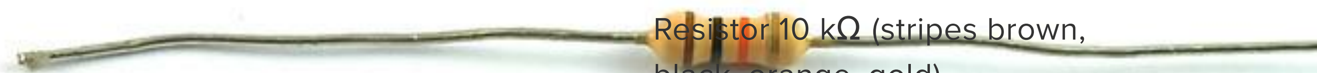


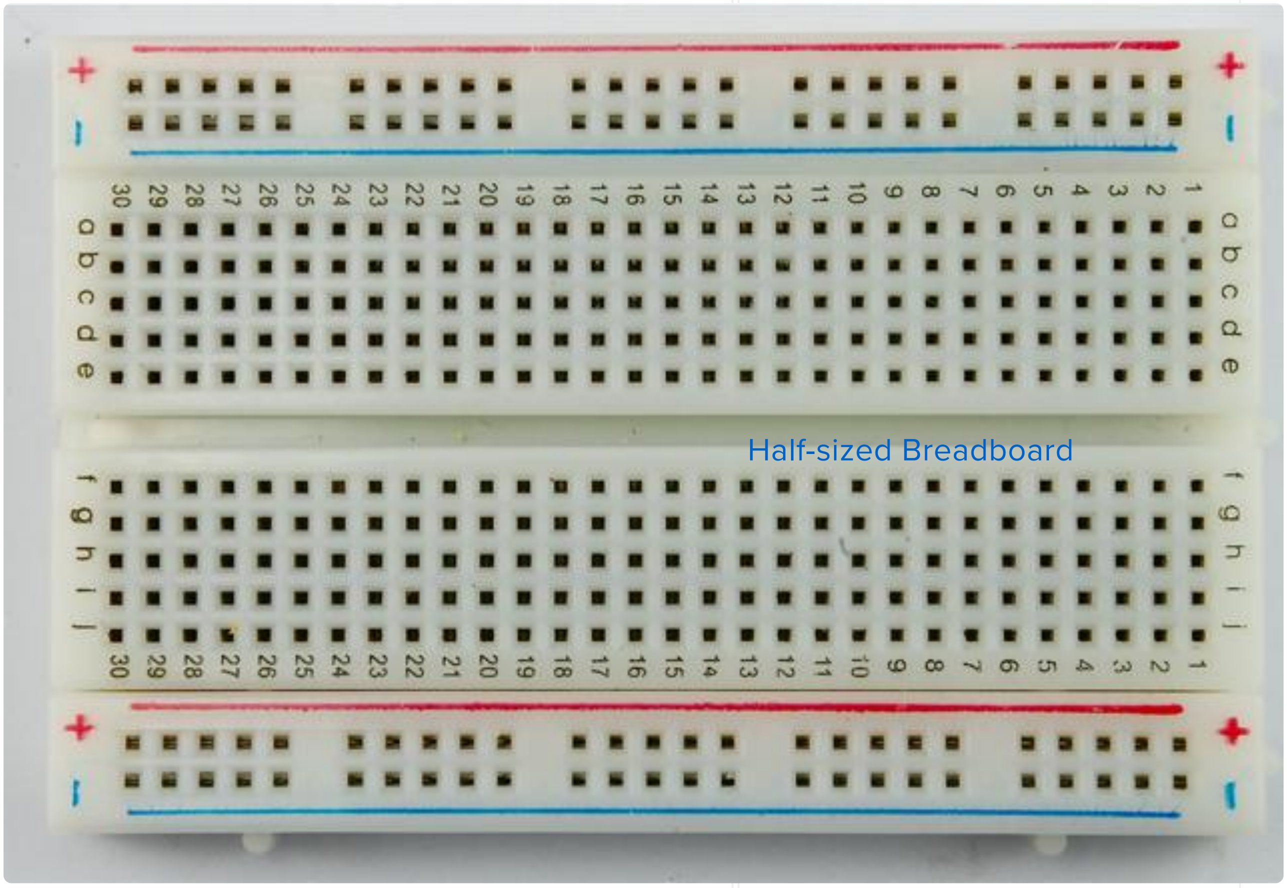
Because the BBB runs Linux, there are many ways in which it can be programmed. In this tutorial we show how to read analog values with a photoresistor light sensor using Python.

You Will Need

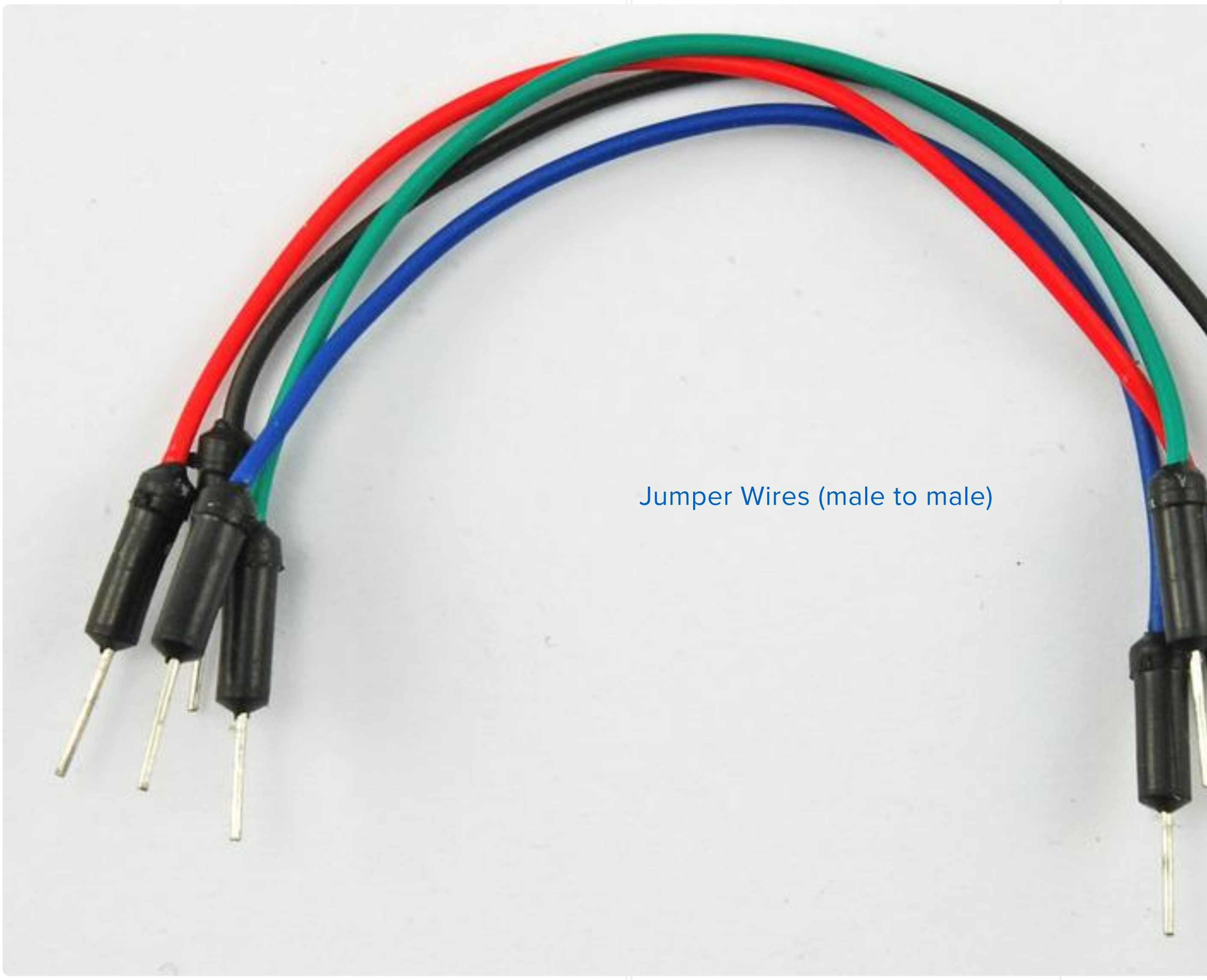
To try out this tutorial, you will need:

	BeagleBone Black

	Photoresistor
	Resistor 10 kΩ (stripes brown, black, orange, gold)



Half-sized Breadboard



Jumper Wires (male to male)

The photoresistor used has a dark resistance in excess of 200 kΩ and under bright light, the resistance falls to 1 or 2 kΩ. You can use other photoresistors, but you may need to change the value of the 10 kΩ resistor in order to get good resolution for your light measurements.

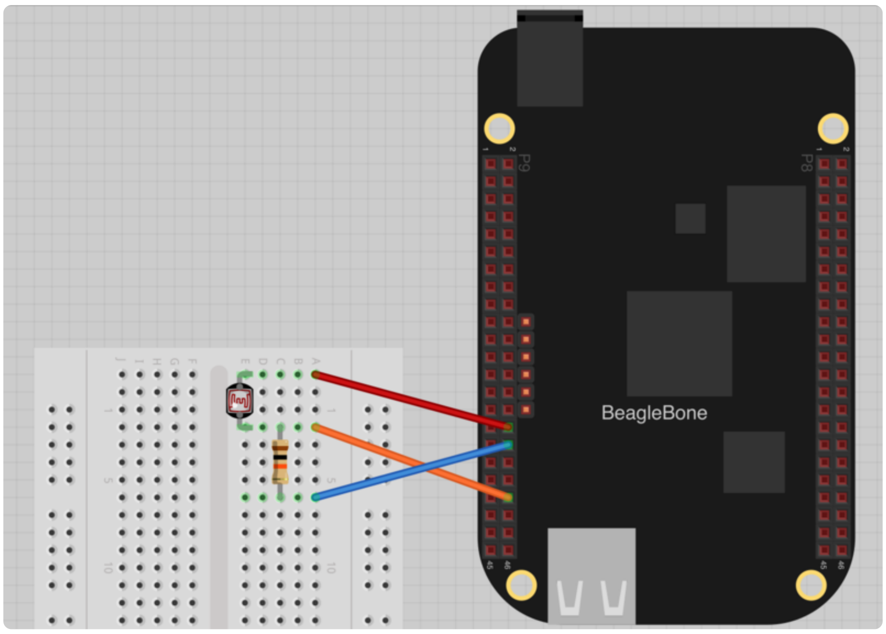
Installing the Python Library

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

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



Both the photoresistor and the resistor can be placed either way around. They are arranged in what is called a "voltage divider" arrangement.

The orange lead to pin P9.40, which is also analog input 1 (AIN1). Note that only certain pins can be used as analog inputs (see http://learn.adafruit.com/setting-up-io-python-library-on-beaglebone-black/adc (https://adafru.it/ch0))

Connect the blue lead from pin 34 of P9 to the bottom of the resistor and the red lead from pin 32 of P9 to the top connection of the photoresistor.

The pins are numbered left to right, 1, 2 then on the next row down 3,4 etc. 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)

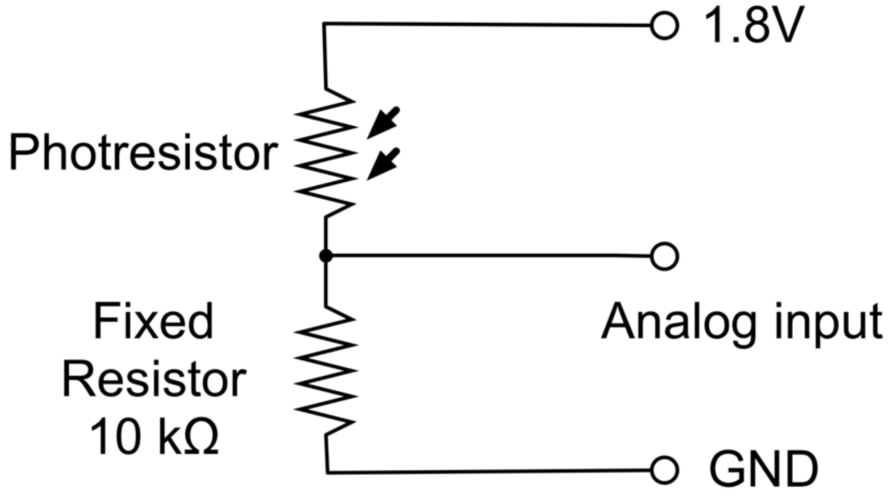
Photoresistors

The photocell used is of a type called a photoresistor, sometimes called an LDR - Lig ht Dependent Resistor. As the name suggests, these components act just like a resistor, except that the resistance changes in response to how much light is falling on them.

The photoresistor used has a dark resistance in excess of 200 kΩ and under bright light, the resistance falls to 1 or 2 kΩ.

To convert this varying value of resistance into something we can measure on BBB's analog input, it need to be converted into a voltage between 0 and 1.8V.

The simplest way to do that is to combine it with a fixed resistor, in this case of 10 kΩ in an arrangement called a voltage divider.



The voltage at the analog input will be pulled up towards 1.8V by the photoresistor and down towards 0V by the fixed resistor. Exactly who is willing this tug-or-war will depend on the resistance of the photoresistor.

If the photoresistor is brightly illuminated, then its resistance will fall so it will pull the voltage up closer to 1.8V. If, on the other hand, the photoresistor is in the dark, the resistance will increase, so the fixed resistor will dominate and the analog input will be pulled close to 0V.

The VDD_ADC1 pin (P9, pin 32) is a steady 1.8V designed to be used to provide a reference voltage for situations like this.

The Python Console

Before writing a Python program to measure and report the light level, we can try some experiments in the Python Console.

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.
&gt;&gt;&gt;
```

First, we need to import the library, so enter the command:

```
&gt;&gt;&gt; import Adafruit_BBIO.ADC as ADC
```

Now enter the command below into the Python Console to setup the ADC (Analog to Digital Convertor).

```
&gt;&gt;&gt; ADC.setup()
```

You can now read the value at the analog input (between 0 and 1) using the command below. Try it a few times while moving your hand closer to and further away from the photoresistor. The readings should change in response to your hand movements. Cover the photoresistor completely and the reading should drop almost to 0.

```
&gt;&gt;&gt; ADC.read("P9_40")
0.547777771949768O7
&gt;&gt;&gt; ADC.read("P9_40")
0.55000001192092896
&gt;&gt;&gt; ADC.read("P9_40")
0.55166667699813843
&gt;&gt;&gt; ADC.read("P9_40")
0.39388889074325562
&gt;&gt;&gt;
```

Note: Using the up arrow will repeat the last line that you entered in the Python Console.

Writing a Program

To automate this, we can write a short Python program to repeatedly display the analog reading (between 0 and 1) and the voltage that this corresponds to, printing values once per second.

Exit the Python Console by typing:

```
&gt;&gt;&gt; exit()
```

This should take you back to the Linux prompt.

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

```
nano light.py
```

Now paste the code below into the editor window.

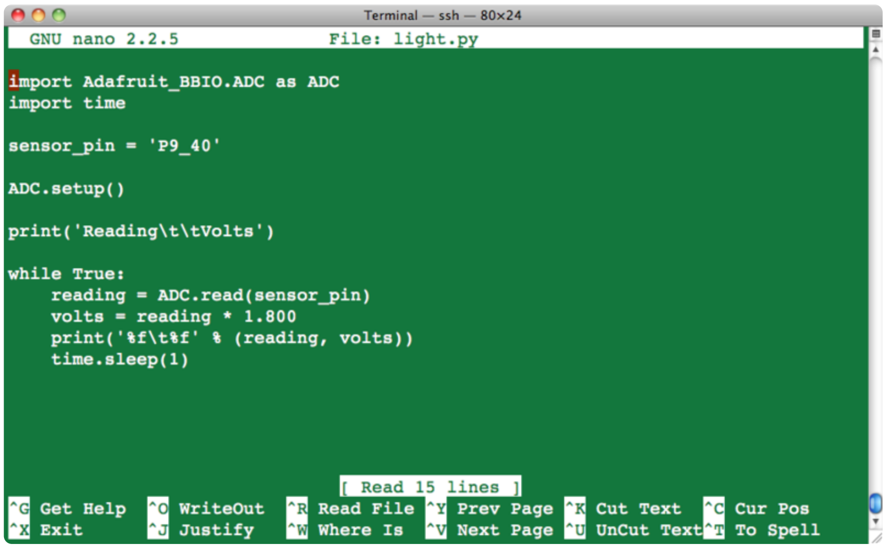
```
import Adafruit_BBIO.ADC as ADC
import time

sensor_pin = 'P9_40'

ADC.setup()

print('Reading\t\tVolts')

while True:
    reading = ADC.read(sensor_pin)
    volts = reading * 1.800
    print('%f\t%f' % (reading, volts))
    time.sleep(1)
```



```
Terminal — ssh — 80×24
  GNU nano 2.2.5              File: light.py

import Adafruit_BBIO.ADC as ADC
import time

sensor_pin = 'P9_40'

ADC.setup()

print('Reading\t\tVolts')

while True:
    reading = ADC.read(sensor_pin)
    volts = reading * 1.800
    print('%f\t%f' % (reading, volts))
    time.sleep(1)

                             [ Read 15 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

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

To start the program, enter the command:

```
python light.py
```

You will then see a series of readings.

```
Reading             Volts
0.000556    0.001000
0.000000    0.000000
0.026111    0.047000
0.448889    0.808000
0.449444    0.809000
```

```
0.452222   0.814000
0.730000   1.314000
0.738889   1.330000
0.722778   1.301000
0.726111   1.307000
0.445556   0.802000
0.401667   0.723000
0.717778   1.292000
```

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

The readings taken are not in any useful units of light intensity. Photoresistors are not carefully calibrated sensors. If you wanted to make a light meter, with absolute measurement of light intensity in meaningful units, you would need to create a lookup table that related readings with readings taken from a properly calibrated light meter.

Warning: The analog inputs of the BBB operate at 1.8V. So do not be tempted to connect the red lead to a higher voltage.

Next Steps

Now that you can measure the light, you could do other things with it like using an 'if' command in the program to display a message when it gets dark.

You could also use the project as a light logger, using Python to write the readings to a file, each accompanied by a time stamp. If the readings are written one per line, with a comma between the time and the reading, then you will be able to import it directly into a spreadsheet and produce charts from the data.

You could also use other types of resistive sensor in place of the photoesistor, such as:

- Force sensitive resistor: http://www.adafruit.com/products/166 (http://adafru.it/166)
- Ribbon sensor: http://www.adafruit.com/products/178 (http://adafru.it/178)

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.