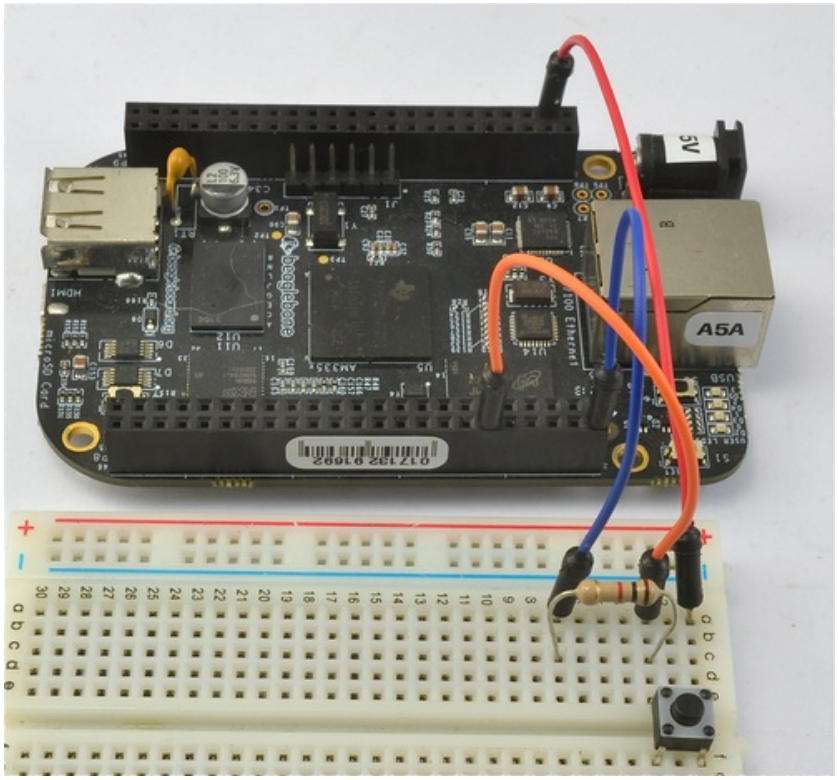
adafruit learning system

Connecting a Push Button to BeagleBone Black

Created by Simon Monk

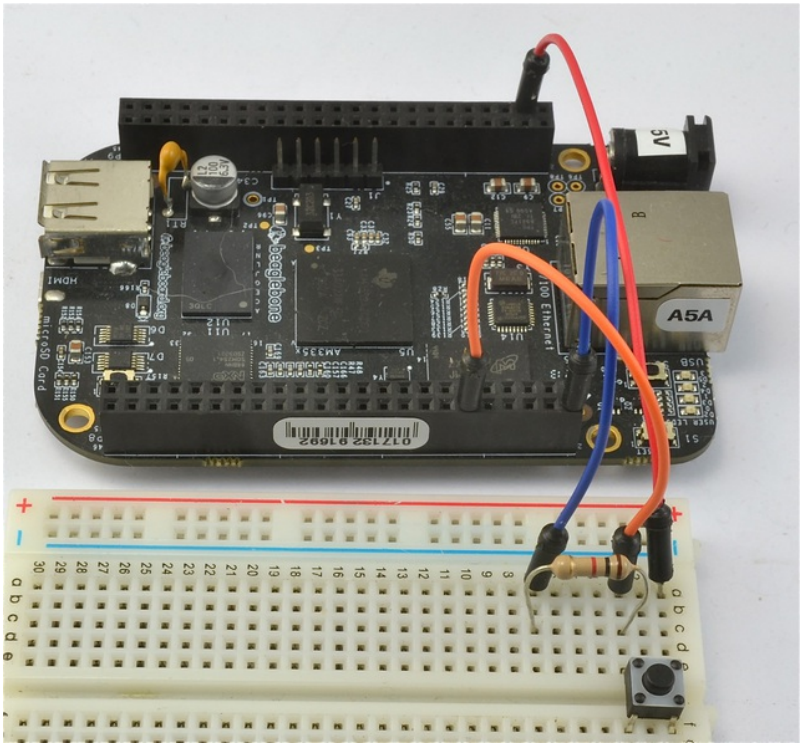


Last updated on 2014-02-16 11:45:07 AM EST

Guide Contents

Overview

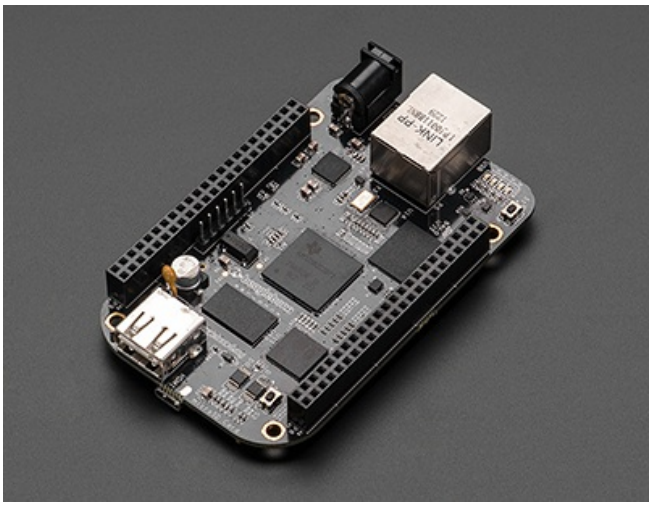
In this tutorial, you will learn how to connect a push button switch to a BeagleBone Black and have it display a message when the button is pressed.

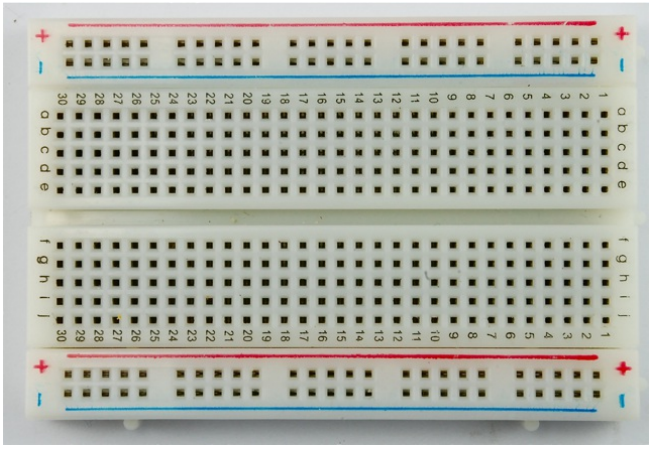
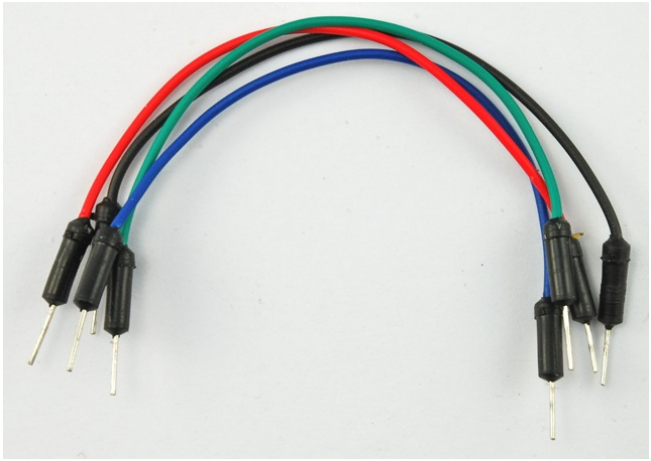


Because the BBB runs Linux, there are many ways in which it can be programmed. In this tutorial we show how to use an input pin using Python.

You will need

To complete this tutorial, you will need:

	BeagleBone Black
	Push Switch
	1 kΩ resistor

	Half-size Breadboard
	Male to Male Jumper wires such as http://www.adafruit.com/products/759

Installing the Python Library

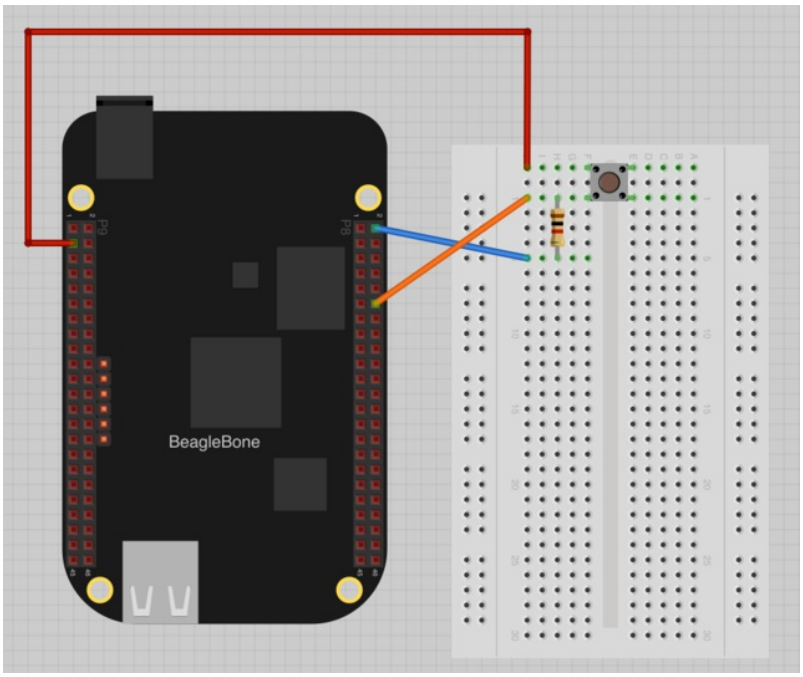
This tutorial uses Ångström Linux, the operating system that comes pre-installed on the BBB.

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

Wiring

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

> Its a good idea to reset the board by powering it down, so that all the GPIO pins are set to their input state, before wiring up the breadboard. Otherwise, if the pin being used as an input was last used as an output, this could damage the board.



The pins on the switch are spaced slightly wider in one direction than the other, so they will only fit the correct way around over the gap down the centre of the breadboard.

The top two connections on the right hand BBB expansion header (as the board is pictured above -- P8) are both GND. The blue lead is connected from this GND (0V) connection to one end of the resistor. The red lead is connected to pin 3 of the other connector (3.3V) and the orange lead to pin 12 (P8.12), which is the right-hand connector on the sixth row down.

The resistor 'pulls down the input pin (P8.12) so that it is at 0V (GND) until the push button is pressed, at which point it will be at 3.3V.

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/ (http://adafru.it/cgi).

The Python Console

Before writing a Python program to use the switch, you can try some experiments from 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.
>>>
```

> If you are using Ubuntu or Debian, you may need to launch python with 'sudo python' to use GPIO.

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

```
>>> import Adafruit_BBIO.GPIO as GPIO
```

Let's now set the pin we are going to use to be an input:

```
>>> GPIO.setup("P8_12", GPIO.IN)
```

At this point, do not press the switch, but type:

```
>>> GPIO.input("P8_12")
0
```

Now hold down the push switch and run the Python line again, by pressing the up cursor key and RETURN.

```
>>> GPIO.input("P8_12")
1
```

So, this experiment has shown us that when the key is pressed, requesting the input value from the pin will have a value of 1 and when it is not pressed, it will be 0.

Writing a Program

To make a message appear each time the button is pressed, we are going to write a short Python program, so 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 switch.py

```
# nano switch.py
```

Now paste the code below into the editor window.

```
import Adafruit_BBIO.GPIO as GPIO
import time

GPIO.setup("P8_12", GPIO.IN)

old_switch_state = 0

while True:
    new_switch_state = GPIO.input("P8_12")
    if new_switch_state == 1 and old_switch_state == 0 :
        print('Do not press this button again!')
        time.sleep(0.1)
    old_switch_state = new_switch_state
```

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

To start the program, enter the command:

```
# python switch.py
```

Each time you press the button, you should see a message.

```
# python switch.py
Do not press this button again!
Do not press this button again!
```

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

We only want the message to appear when the button is pressed. To prevent the message appearing continuously as long as the button is held down the variable old_switch_state is used and the message only displayed when the switch goes from not being pressed to being pressed.

The time.sleep command is a simple way to avoid switch bounce, which would cause the message to appear twice if the switch contacts did not close cleanly when the button is pressed.

Next Steps

You can change the code so that instead of printing a message, it performs some other action. You could for example have it control LED flashing, by combining the code and hardware from this tutorial with that of http://learn.adafruit.com/blinking-an-led-with-beaglebone-black (http://adafru.it/clG)

The same setup can be used for any size button and switch!

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 (http://adafru.it/caH) at Adafruit.