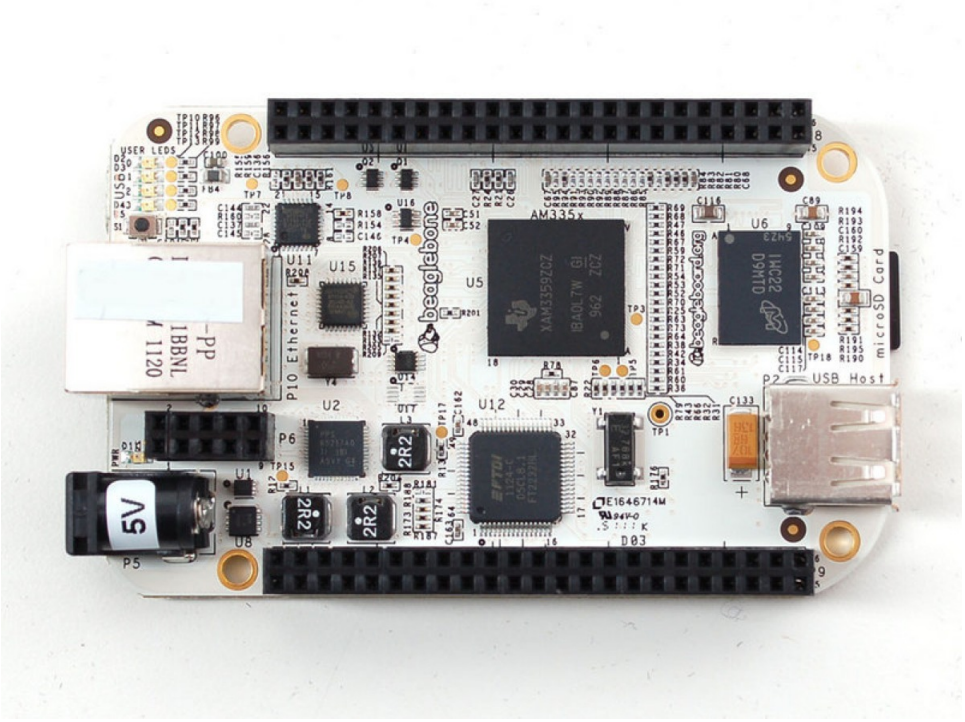
beaglebone



Beaglebone Guide: Using a Push button and LED with the Beaglebone

Jayneil Dalal(jayneil.dalal@gmail.com)

February 7, 2013

Abstract

In this guide, we will do a small project. The aim is to turn ON an LED when a push button is pressed by the user. This guide targets beginners who are just getting started on the Beaglebone. For the purpose of this guide, I have used an Ubuntu 12.04 64 bit system.

Specifications

Processor

- 720MHz super-scalar ARM Cortex-A8 (armv7a)
- 3D graphics accelerator ARM Cortex-M3 for power management
- 2x Programmable Realtime Unit 32-bit RISC CPUs

Connectivity

- USB client: power, debug and device
- USB host
- Ethernet
- 2x 46 pin headers 2x I2C, 5x UART, I2S, SPI, CAN, 66x 3.3V GPIO, 7x ADC

Software

- 4GB microSD card with Angstrom Distribution
- Cloud9 IDE on Node.JS with Bonescript library

Contents of the box

When you purchase a Beaglebone, you get the following items in the box as shown in Figure -1:

1. Beaglebone
2. USB cable
3. 4gb micro sd card

Figure-1: Box contents

Expansion Headers on the Beaglebone

The expansion headers on the beaglebone are comprised of two 46 pin connectors which are P8 and P9. All signals on the expansion headers are 3.3V unless otherwise indicated.

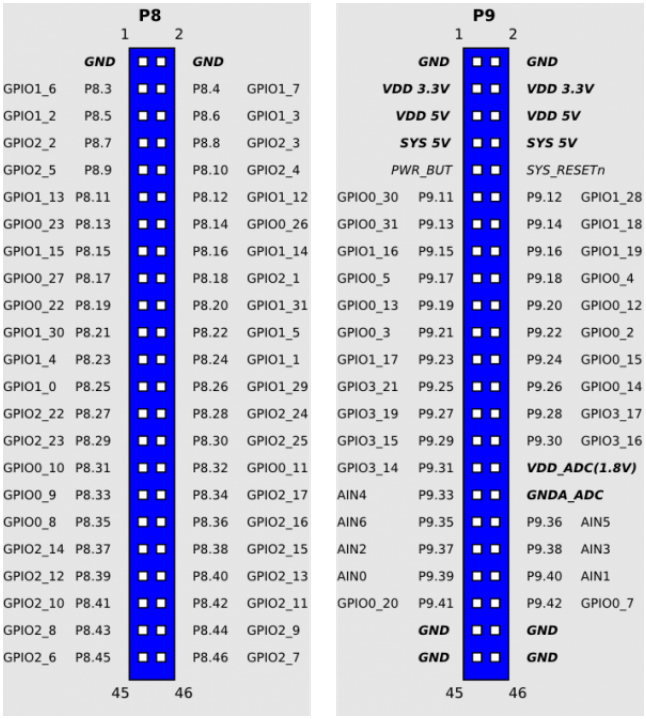


Figure-2: *Expansion headers on the Beaglebone*

Selecting the correct LED

- To make sure that you do not damage the GPIO pins on the Beaglebone, please use a LED whose rating should not exceed 3.3V/6mA:

  http://www.digikey.com/product-detail/en/HLMP-4700-C0002/516-2483-2-ND/1234840

- If you have an LED that is higher than the above mentioned ratings, please refer the refer the link below to select an appropriate current limiting resistor:

  http://www.cmiyc.com/tutorials/led-basics/

GPIO on the Beaglebone

The pins on the expansion header have multiple functions. To find out what is the default function of a pin, refer the beaglebone reference manual which can be downloaded from the link below:

http://beagleboard.org/static/BONESRM_latest.pdf

For example, Table-8 on page - 54 describes the default function of each pin on P8 expansion header under the '**SIGNAL NAME**' column. The '**CONN**' column describes the actual pin number as seen on the physical board. Tables-9,10 list the other possible functions of a particular pin on the P8 expansion header.

Once you have identified the pin number which you would like to use as a GPIO, you need to find out its corresponding reference number in the kernel. For example, if you would like to use pin 23 on P8 expansion header, then find out its default function as mentioned earlier. Note down the entire signal name. In this case, pin 23 is **GPIO1_4**. So any GPIO you come across would be referenced as GPIOM_N. Identify M,N. Use the formula below to find the corresponding reference number in the kernel:

Reference number = Mx32 + Y

Hence, pin 23 would be referenced as gpio 36 in the kernel.

Now, to change the function of a pin using the kernel you need to access the /sys/kernel/debug/omap_mux directory via the terminal on the beaglebone. Here your pin will be referenced by the name it is assigned in its mode 0. So, in table - 9, pin 23 is referenced as **gpmc_ad4** in mode 0. Then, identify the mode in which the pin can be used as GPIO. For pin 23, the mode is 7.

Pullup/Pulldown Resistors

Concept:

So, you want to use the GPIO on the beaglebone to read values externally connected devices(e.g. push button). The particular GPIO pin is set to be used in input mode. Now, think about the case when no input is being provided to that pin(lets say P2) as shown in Figure - 3 below. This is called a 'floating point' condition and the processor cannot determine what is the value of the input pin. In this case, even the noise in the environment can influence the pin value and give wrong results.

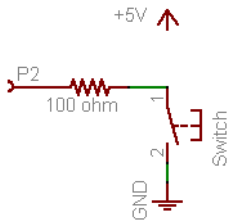


Figure-3: No pullup resistor

Hence, to avoid this condition the pins have a pullup/pull down resistor connected to them to prevent this from happening . Pullup is used to denote that the pin is connected to Vcc(3.3V in this case) via a resistor and pulldown is used to denote the pin that is connected to GND via a resistor. In Figure - 4 below, a 10k ohms pullup resistor is used. The value is high so that when the switch is pressed and the GND is connected to 5V, the current passing through is very less and does not burn the wire or damage the circuit. Beaglebone has internal pullup and pulldown resistors.

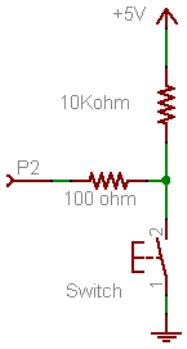


Figure-4: Pullup resistor connected

Push button

A pushbutton is a simple switch mechanism which permits user generated changes in the state of a circuit. Pushbutton usually comes with four legs. As seen from the Figure - 6 below, legs are always connected in groups of two. When the pushbutton is pressed all the 4 legs are connected.

Figure-5: Push Button

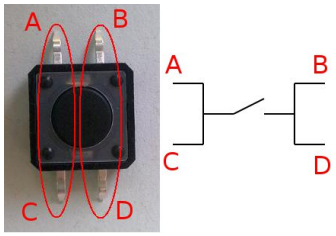


Figure-6: Working of push button

The pushbutton used in this guide can be purchased from here:

https://www.sparkfun.com/products/97

Changing the mode of a pin on the expansion header

As discussed earlier, the pins on the expansion headers have multiple functions including gpio, i.e., the pins are multiplexed. If the default mode of that pin is gpio, great or else the mode needs to be changed explicitly. First let us have a look at the pin register(part of the main control module) description in the AM335x Technical Reference which can be downloaded from the link below:

http://www.ti.com/lit/ug/spruh73g/spruh73g.pdf

7	6	5	4	3	2	1	0
Reserved	conf_<module>_<pin>_slewctrl	conf_<module>_<pin>_rxactive	conf_<module>_<pin>_putypesel	conf_<module>_<pin>_puden	conf_<module>_<pin>_mmode		
R-0h	R/W-0h	R/W-1h	R/W-0h	R/W-0h	R/W-0h		

LEGEND: R/W = Read/Write; R = Read only; W1toCl = Write 1 to clear bit; -n = value after reset

Table 9-58. conf_<module>_<pin> Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	Reserved	R	0h	
19-7	Reserved	R	0h	
6	conf_<module>_<pin>_slewctrl	R/W	0h	Select between faster or slower slew rate 0: Fast 1: Slow Reset value is pad-dependent.
5	conf_<module>_<pin>_rxactive	R/W	1h	Input enable value for the PAD 0: Receiver disabled 1: Receiver enabled
4	conf_<module>_<pin>_putypesel	R/W	0h	Pad pullup/pulldown type selection 0: Pulldown selected 1: Pullup selected Reset value is pad-dependent.
3	conf_<module>_<pin>_puden	R/W	0h	Pad pullup/pulldown enable 0: Pullup/pulldown enabled 1: Pullup/pulldown disabled Reset value is pad-dependent.
2-0	conf_<module>_<pin>_mmode	R/W	0h	Pad functional signal mux select. Reset value is pad-dependent.

Figure-7: Pin register description for AM335x

As seen from the Figure - 7 above, Bits 0-2 are used to change the mode of a pin. Bit-3 is used to enable or disable a pullup/pulldown resistor. Bit-4 will decide whether that particular pin will use pullup or pulldown resistor. Once you have the beaglebone up and running, execute the following command in its terminal

```
$ cat /sys/kernel/debug/omap_mux/gpmc_ad4
```

You will get an output similar to the one shown below

```
name: gpmc_ad4.gpio1_4 (0x44e10810/0x810 = 0x0027), b NA, t NA
mode: OMAP_PIN_INPUT_PULLDOWN | OMAP_MUX_MODE7
signals: gpmc_ad4 | mmc1_dat4 | NA | NA | NA | NA | NA | gpio1_4
```

The mode field tells whether the pin is being used as input or output, whether it is using pullup or pulldown resistor as well as what is the current mode in which the pin is being used. The signal field tells what are the different possible functions that this pin can have. Now, in the name field pay close attention to '0x0027'. The number is in hexadecimal and it indicates the current mux settings for the pin. So, to convert it to binary, just split the two digits and convert them individually. So, '2' in binary is 10 and '7' in binary is '111'. So '0x0027' in binary would be 10111 where the 1 on the left most side is the Most Significant Bit(M.S.B.) and the 1 on the right most side is Least Significant Bit(L.S.B.) . In the pin register, Bit -0 is the L.S.B and Bit - 6 is the M.S.B as the bits above it are all reserved. So, '10111' means the Bits 0-2 have value 1, Bit-3 has a value 0, Bit-4 has a value 0, Bit-5 has a value 1 and rest of the bits are zero padded(set to zero). This means that this particular pin has been configured to be used in mode-7 which is GPIO mode. Also, the pin is configured to use pull down resistors but is not using them currently.

Preparing the SD Card

The Beaglebone already comes with an sd card that is preloaded with a working Angstrom image. In any case should you want a newer image or want to program the sd card again, this section covers it all.

- First download the latest Angstrom image for Beaglebone from the link below:

  http://downloads.angstrom-distribution.org/demo/beaglebone/

  At the time of writing this guide, the latest image available for download was 'Angstrom-Cloud9-IDE-GNOME-eglibc-ipk-v2012.05-beaglebone-2012.11.22.img.xz'

- Now, identify the correct raw device name (like /dev/sde - not /dev/sde1) for the sd card

- Now unpack the image to the sd card by writing the following command in the terminal:

  ```
  $ xz -dkc Angstrom-Cloud9-IDE-GNOME-eglibc-ipk-v2012.05-beaglebone-2012.11.22.img.xz > /dev/sdX
  ```

  Here 'sdX' stands for the device id of the sd card.

Booting up the board

- To power up the beaglebone, connect it to the computer via the usb cable as shown in the Figure -8:

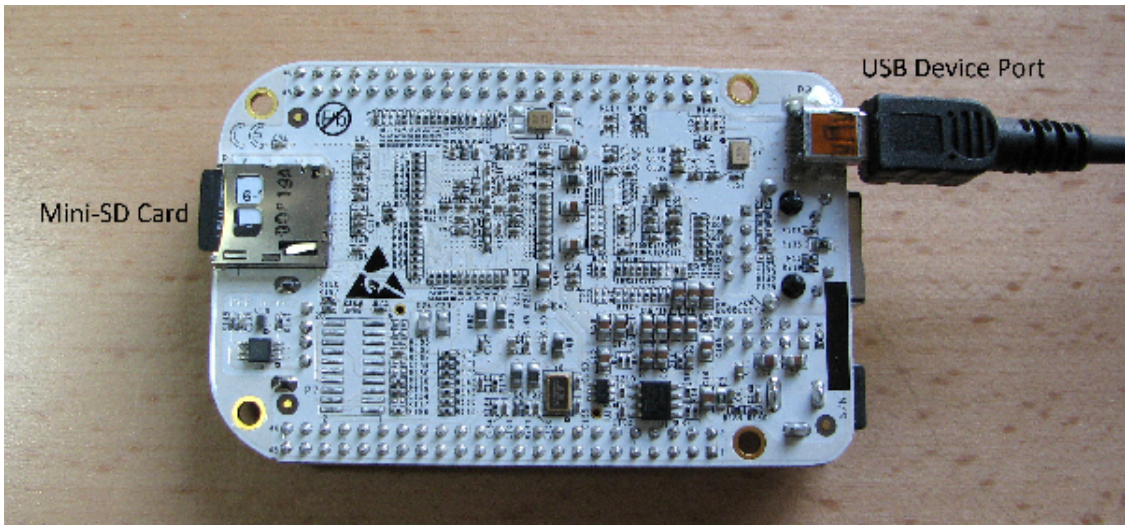


Figure-8: Powering up the Beaglebone

- Eject the Beaglebone via the disk utility program in Ubuntu. I tried ejecting it via the file manager but that did not work for me. Upon every boot, the Beaglebone is the "storage mode" by default. Hence, this step is done to switch it to "network mode".
- Access the beaglebone via the terminal:

```
$ screen /dev/ttyUSB1 115200
```

Note:- You can also use minicom. But this is just much easier! Also in most cases the virtual USB serial port is ttyUSB1. If it does not work, try ttyUSB0 .

- You should be greeted by an Angstrom login. The username for the same is 'root' and for password, just press 'ENTER'. You should see the following prompt:

```
root@beaglebone:~#
```

Mode -1 : GPIO pin configured to use internal pull up resistor

Setup

Please refer the figure below to see the connections:

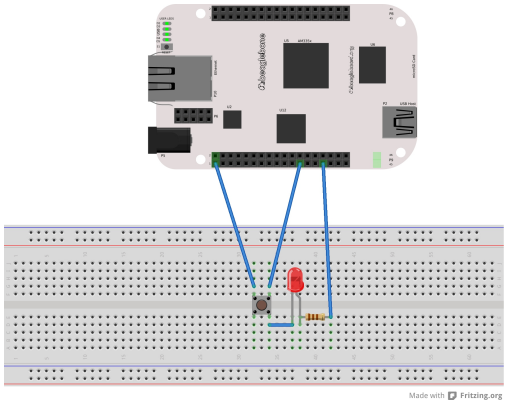


Figure-9: Connection Diagram

Only the pins from P8 expansion header are being used in this case. Connect the GND from pin-1 on the beaglebone to a leg-1 of the push button. Connect pin-25 on the beaglebone to leg-2(which is not connected to the previous leg-1) of the push button. Connect the cathode of the LED to leg-4(which is connected by default to leg-2) of the push button. Connect the anode of the LED to a 110 ohm resistor and the other end of the resistor to pin-29 on the beaglebone. It is recommended to use this mode.

Script

```
#!/bin/bash
#Open the GPIO port
#Pin no. 23 aka input pin
echo 17 > /sys/kernel/debug/omap_mux/gpmc_ad4
echo 36 > /sys/class/gpio/export
echo "in" > /sys/class/gpio/gpio36/direction
#Pin no. 29 aka output pin
echo 7 > /sys/kernel/debug/omap_mux/lcd_hysnc
echo 87 > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio87/direction
# Read forever
while :
do
# Read value of the GPIO pin
THIS_VALUE=‘cat /sys/class/gpio/gpio36/value‘
if [ "$THIS_VALUE" = "0" ]
then
echo 1 > /sys/class/gpio/gpio87/value
else
echo 0 > /sys/class/gpio/gpio87/value
fi
done
```

In the above script, '#' is used for comments(except for first line). Write the above script in your favorite text editor, save it(make sure to add the '.sh' extension at the end) and place it in on the sd card before you boot the beaglebone. Or you can use a text editor like 'nano' on the beaglebone and write the above script. In any case make sure the script is executable by typing the following command in the terminal:

```
$ chmod a+x <script_name>
```

In my case, I named the script 'gpiobutton(pullup).sh'

Steps

- Make sure that your kernel supports GPIO by typing the following commands in the terminal:

```
$ grep GPIOLIB /boot/config-‘uname -r‘
```

The output after running the above command should be as shown below:

```
CONFIG_ARCH_REQUIRE_GPIOLIB=y
CONFIG_GPIOLIB=y
```

Now run the following command in the terminal:

```
$ grep GPIO_SYSFS /boot/config-‘uname -r‘
```

The output after running the above command should be as shown below:

```
CONFIG_GPIO_SYSFS=y
```

- So, now run the script created earlier by typing the following command in the terminal:

```
$ ./gpiobutton(pullup).sh
```

You should get the output as shown below:

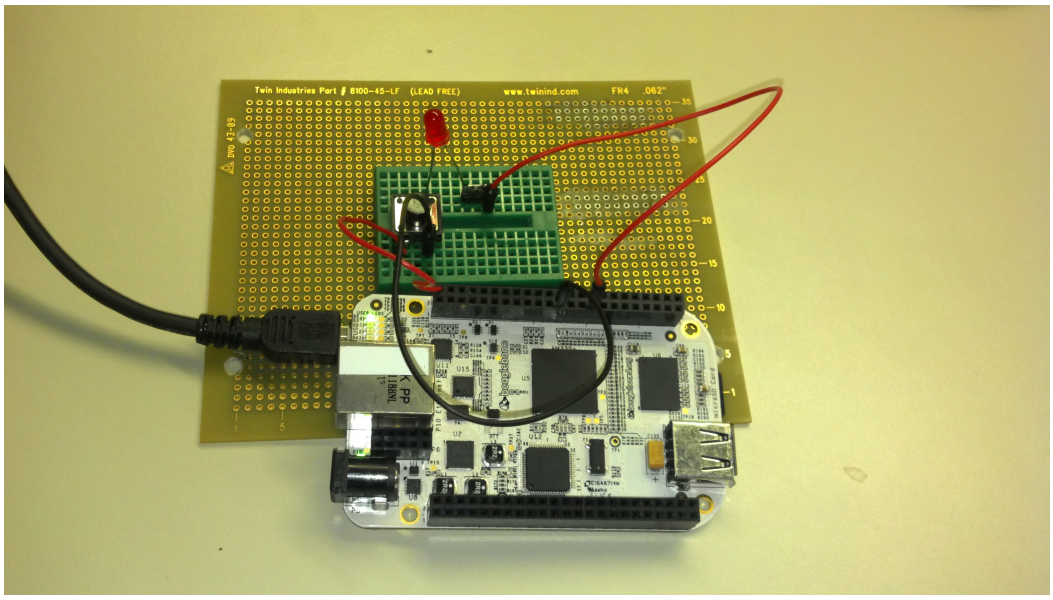


Figure-10: Led OFF

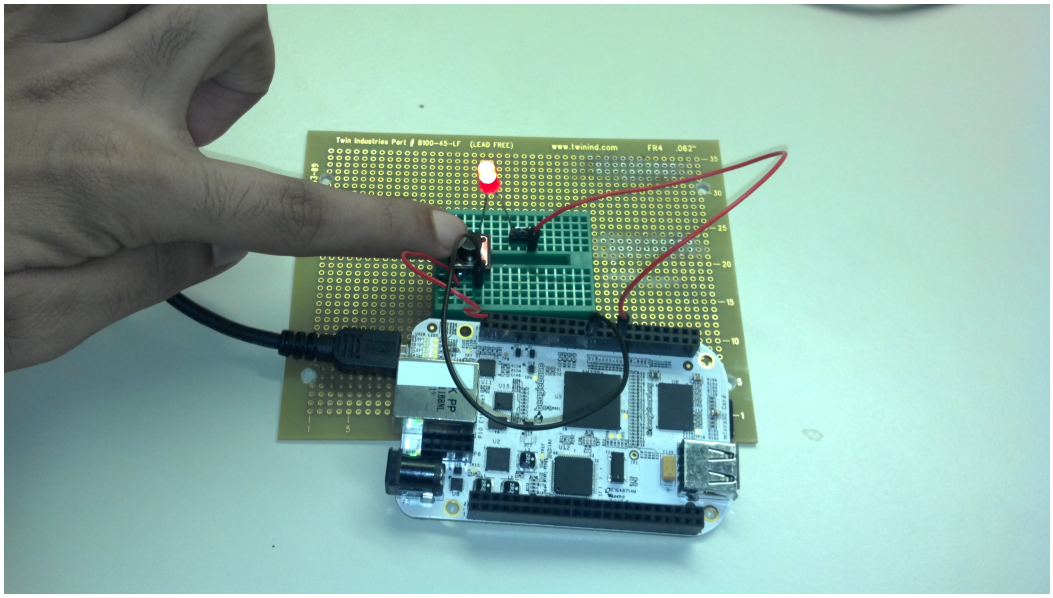


Figure-11: Led ON

Mode -2 : GPIO pin configured to use internal pull down resistor

Setup

Please refer the figure below to see the connections:

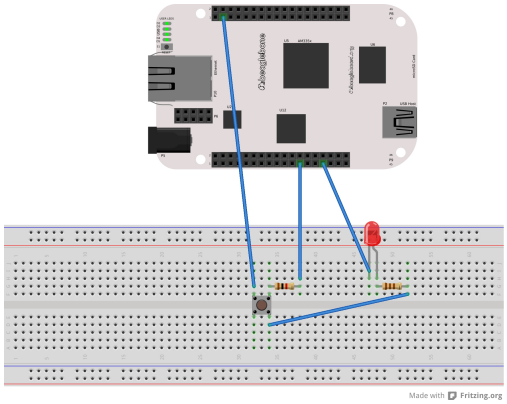


Figure-12: Connection Diagram

Connect the pin-2(VDD or 3.3V) on P9 expansion header to leg-1 of the push button switch. Now, connect pin-25 on P8 expansion header to leg-2(which is not connected by default to leg-1) of the push button via a 1000 ohms resistor. The value of resistor chosen here is high as we want to restrict high amount of current flowing to the gpio pin and damaging it. So, in this case,the current will be reduced to 3.3mA which is below the general 8mA rating of the beaglebone. Connect pin-29 on P8 expansion header to the cathode of the LED. Connect the leg-4(which is connected to leg-2 by default) of the push button to the anode of the LED via a 110 ohms resistor.

Script

```
#!/bin/bash
#Open the GPIO port
#Pin no. 23 aka input pin
echo 7 > /sys/kernel/debug/omap_mux/gpmc_ad4
echo 36 > /sys/class/gpio/export
echo "in" > /sys/class/gpio/gpio36/direction
#Pin no. 29 aka output pin
echo 7 > /sys/kernel/debug/omap_mux/lcd_hysnc
echo 87 > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio87/direction
# Read forever
while :
do
# Read value of the GPIO pin
THIS_VALUE=‘cat /sys/class/gpio/gpio36/value‘
if [ "$THIS_VALUE" = "1" ]
then
echo 0 > /sys/class/gpio/gpio87/value
fi
done
```

In the above script, '#' is used for comments(except for first line). Write the above script in your favorite text editor, save it(make sure to add the '.sh' extension at the end) and place it in on the sd card before you boot the beaglebone. Or you can use a text editor like 'nano' on the beaglebone and write the above script. In any case make sure the script is executable by typing the following command in the terminal:

```
$ chmod a+x <script_name>
```

In my case, I named the script 'gpiobutton(pulldown).sh'

Steps

- Make sure that your kernel supports GPIO by typing the following commands in the terminal:

```
$ grep GPIOLIB /boot/config-‘uname -r‘
```

The output after running the above command should be as shown below:

```
CONFIG_ARCH_REQUIRE_GPIOLIB=y
CONFIG_GPIOLIB=y
```

Now run the following command in the terminal:

```
$ grep GPIO_SYSFS /boot/config-‘uname -r‘
```

The output after running the above command should be as shown below:

```
CONFIG_GPIO_SYSFS=y
```

- So, now run the script created earlier by typing the following command in the terminal:

```
$ ./gpiobutton(pulldown).sh
```

You should get the output as shown below:

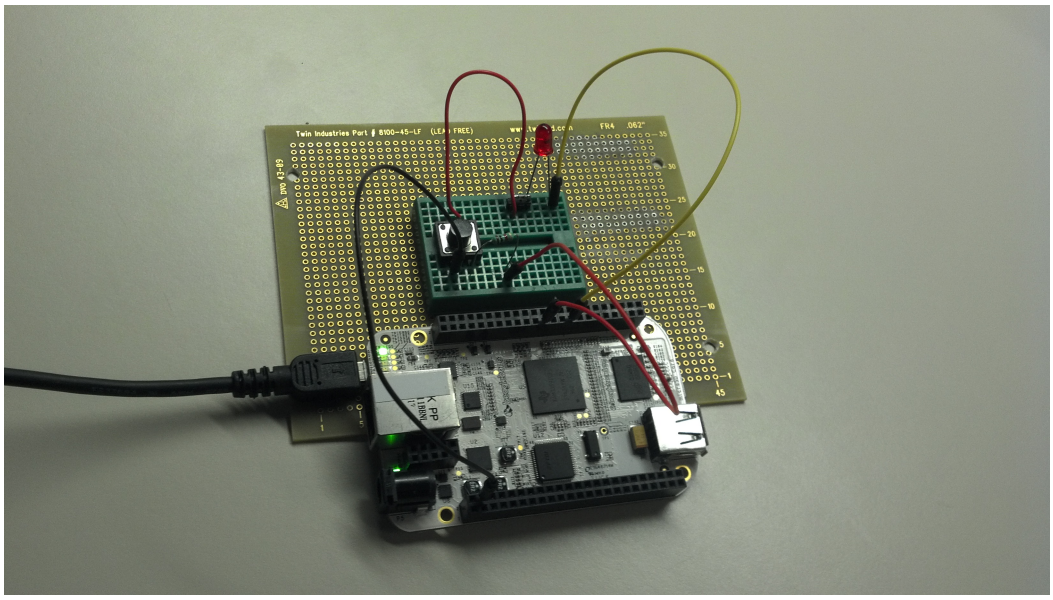


Figure-13: Led OFF

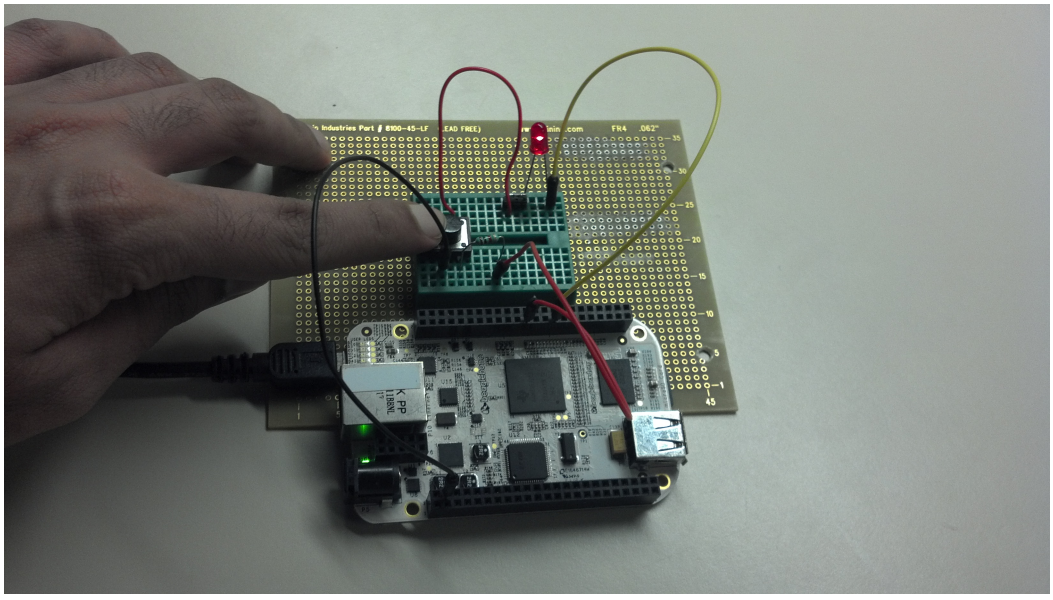


Figure-14: Led ON