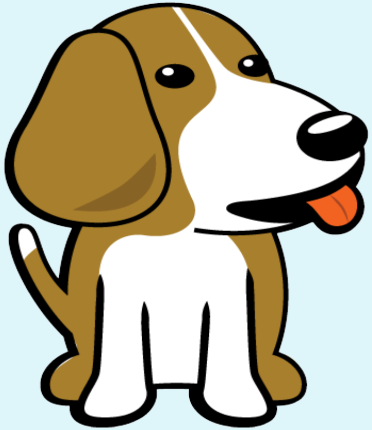
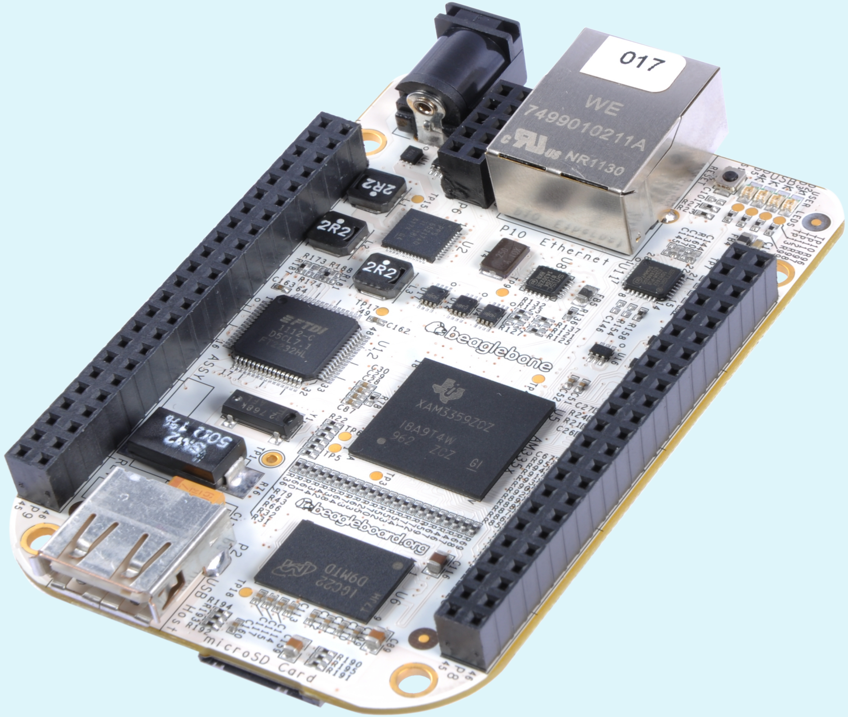
Embedded Linux Conference 2013

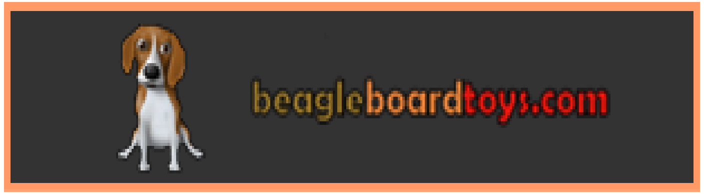




2-20-13

Beaglebone – Hands on Tutorial

Sponsors



Speaker

Jayneil Dalal is a FOSS advocate who loves to explore different open source technologies and has been a key member of the PandaBoard.org project at Texas Instruments. He has previously presented at Linuxcon North America 2012, Drodicon 2012 in Berlin, Southeast Linuxfest 2012, Indiana Linuxfest 2012, Northwest Linuxfest 2012, Scipy 2011 and Opensource bridge 2012.

Agenda

- **Beaglebone Overview**

- **Tutorial -1**
  Blinking the user LED on the Beaglebone

- **Tutorial -2**
  GPIO Programming on the Beaglebone

- **Tutorial -3**
  Physical computing on the Beaglebone

- **Q&A**

Tutorial Resources

Please download the tutorial guides from the link below:

http://elinux.org/Beaglebone_Guides

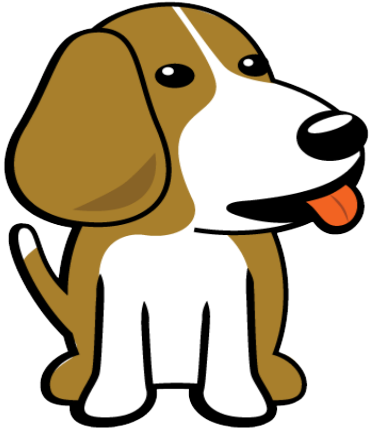
Please download the Beaglebone System Reference Manual from the link below:

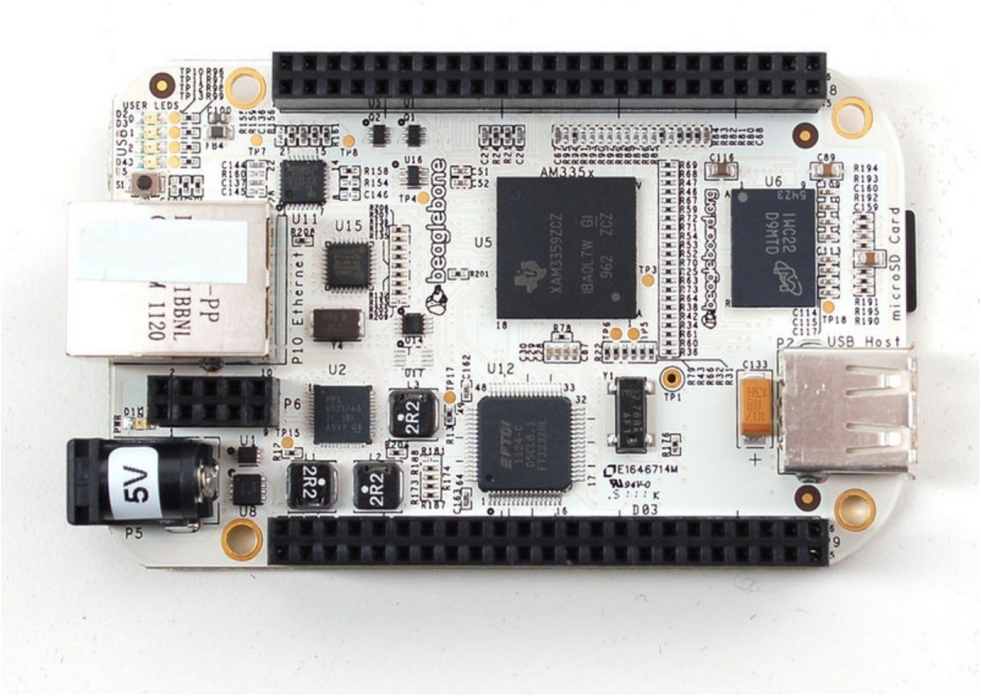
http://beagleboard.org/static/BONESRM_latest.pdf

Please download the AM335x Technical Reference Manual from the link below:

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

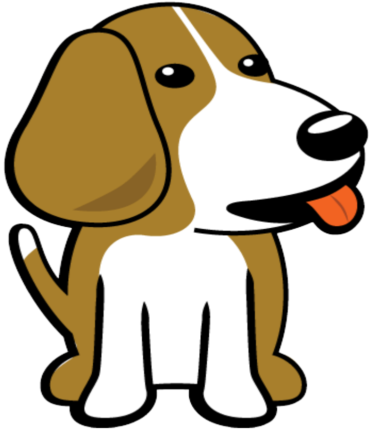
Beaglebone Overview





About me

- **Who am I?**
  I am a low-cost credit-card-sized Linux computer that connects with the Internet and runs software such as Android and Ubuntu
- **How much do I cost?**
  I cost $89
- **Where to buy me?**
  http://beagleboard.org/buy
- **Want to contact me?**
  #beagle [IRC]
  groups.google.com/group/beagleboard



I am so ripped!

Processor

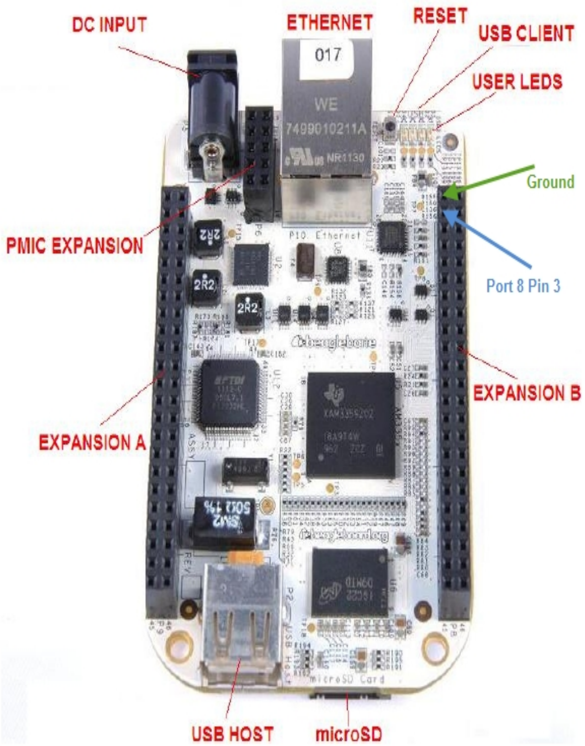
- 720MHz super-scalar ARM Cortex-A8 (armv7a)
- 3D graphics accelerator ARM Cortex-M3 for power management
- 2x Programmable Realtime Unit 32-bit RISC CPUs

Connectivity

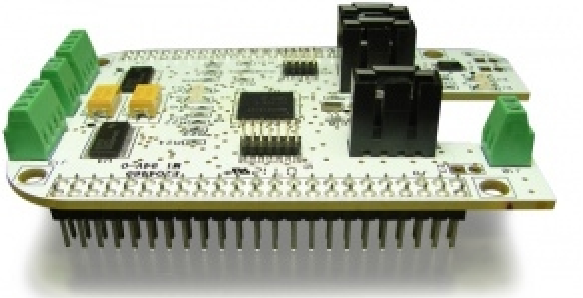
- USB client: power, debug and device
- USB host
- Ethernet
- 2x 46 pin headers 2x I2C, 5x UART, I2S, SPI, CAN, 66x 3.3V GPIO, 7x ADC

Software

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

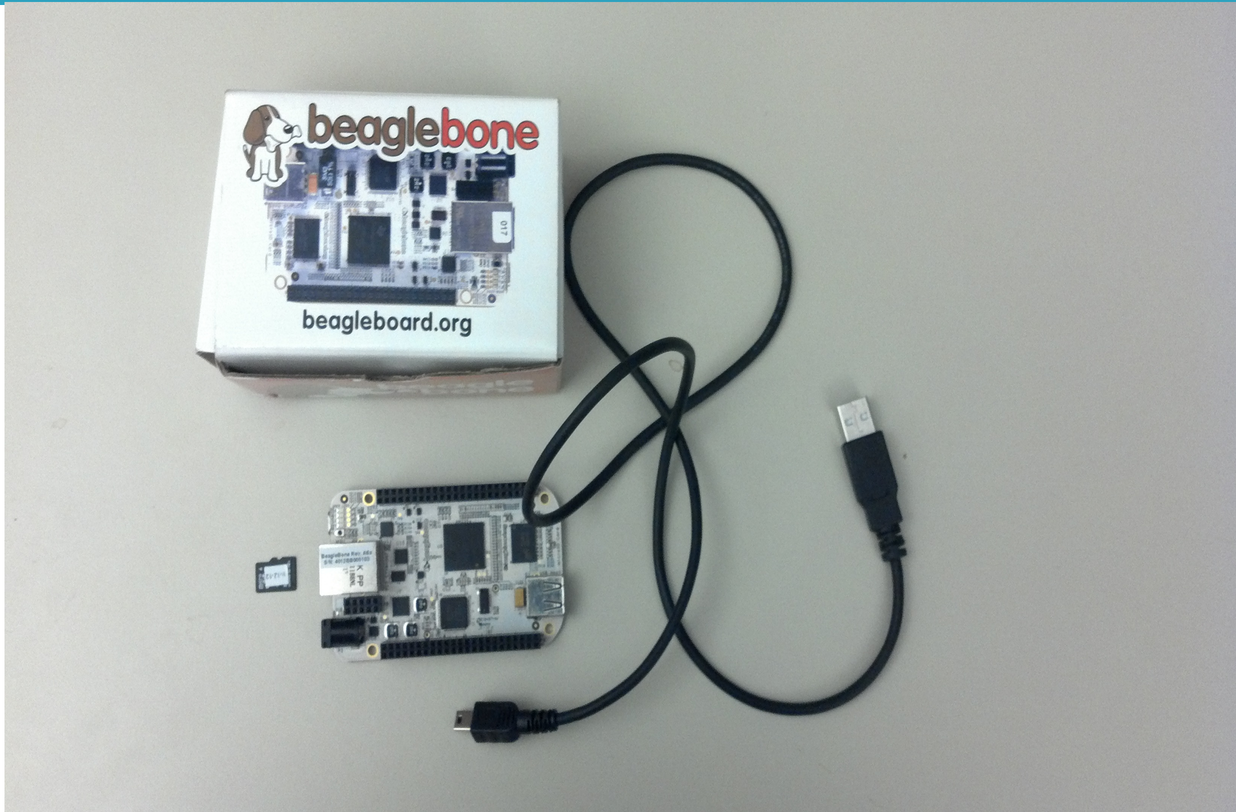


Capes



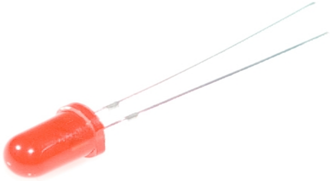
Box Contents

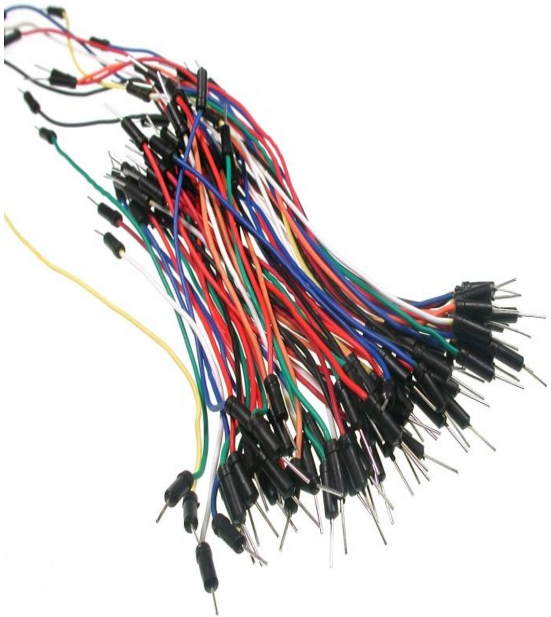
- Beaglebone
- 4GB sdcard
- USB Cable



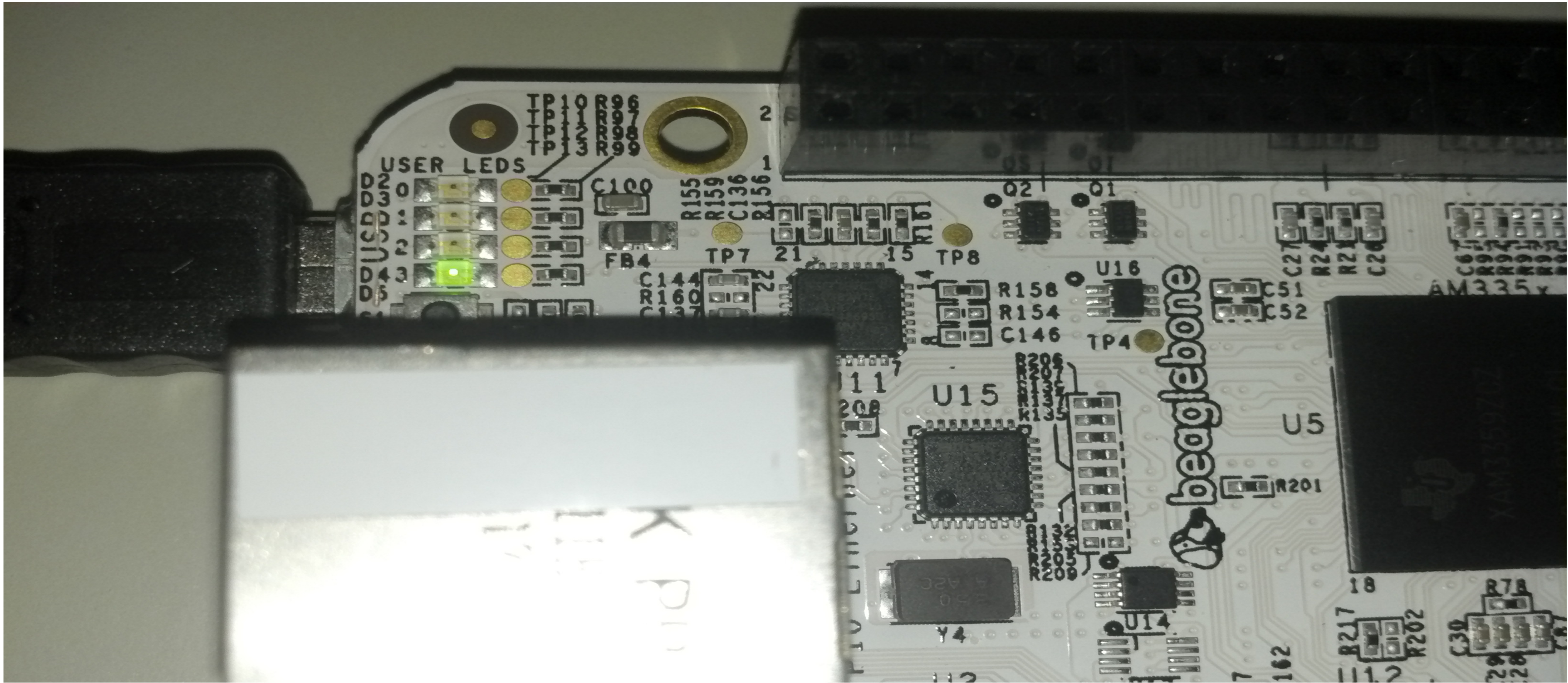
Tutorial Accessories

- Breadboard(x1)
- Hookupwires(x4)
- LED(x1)
- Resistor(x1)
- Pushbutton(x1)





Tutorial – 1: Blinking user LED



Preparing the sd card(optional)

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 making these slides, 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.

Powering up the Beaglebone

- To power up the Beaglebone, connect it to the computer via the usb cable.

- Eject the Beaglebone. Upon every boot, the Beaglebone is the “storage mode” by default. Hence, this step is done to switch it to “network mode”.

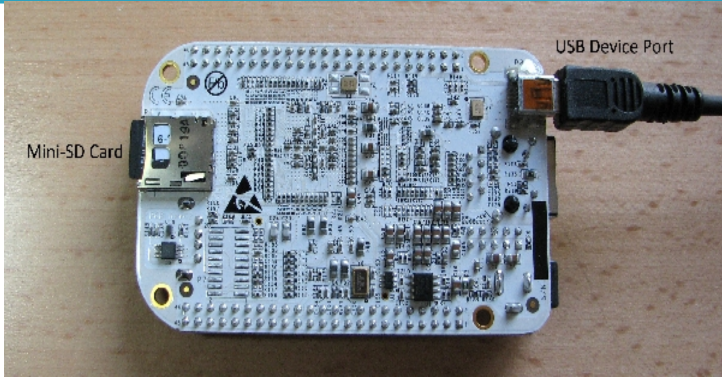
- Access the beaglebone via the terminal:

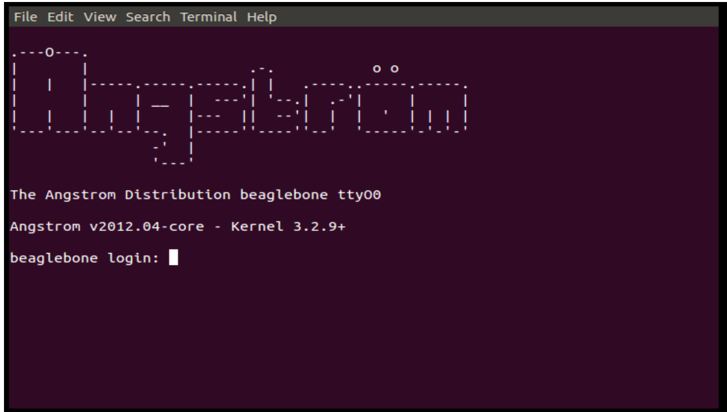
  ```
  $ screen /dev/ttyUSB1 115200
  ```

  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:~#
  ```





```
The Angstrom Distribution beaglebone tty00

Angstrom v2012.04-core - Kernel 3.2.9+

beaglebone login:
```

GPIO support in the kernel

The kernel in the stock Angstrom image on the sd card has GPIO support. But in any case to check, follow the steps below:

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

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

```
CONFIG_ARCH_REQUIRE_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
```

User LED(s)

There are four user LED(s) on the Beaglebone. The user LED(s) are accessible from user space on the file system at this location:

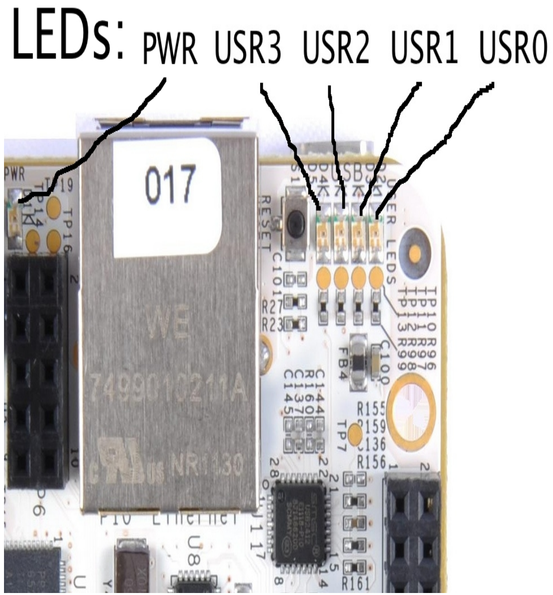
/sys/class/leds/

There is one directory per user LED, named as shown below:

/sys/class/leds/beaglebone::usr0/
/sys/class/leds/beaglebone::usr1/
/sys/class/leds/beaglebone::usr2/
/sys/class/leds/beaglebone::usr3/

Inside each one of those directories, there is a file named "brightness". If you write a "1" or a "0" to this file, then you can control the status of that led, i.e. , toggle it ON or OFF respectively.

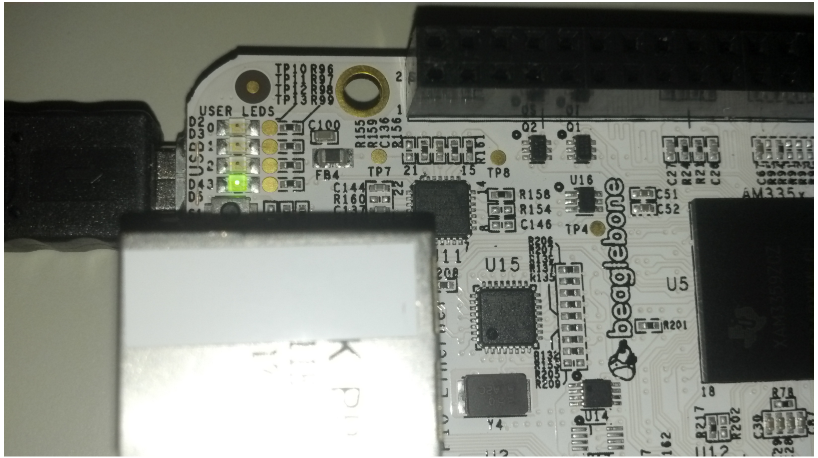
Note:- Since, User LED 0 is already in use to indicate Ethernet activity, you should use the remaining LED(s) for your projects.



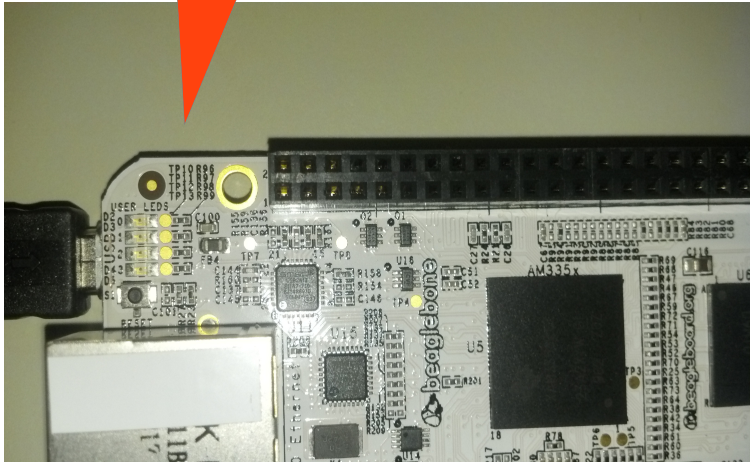
Lets Blink that LED!

Write the following commands in your terminal(First one is for turning ON and latter for OFF):

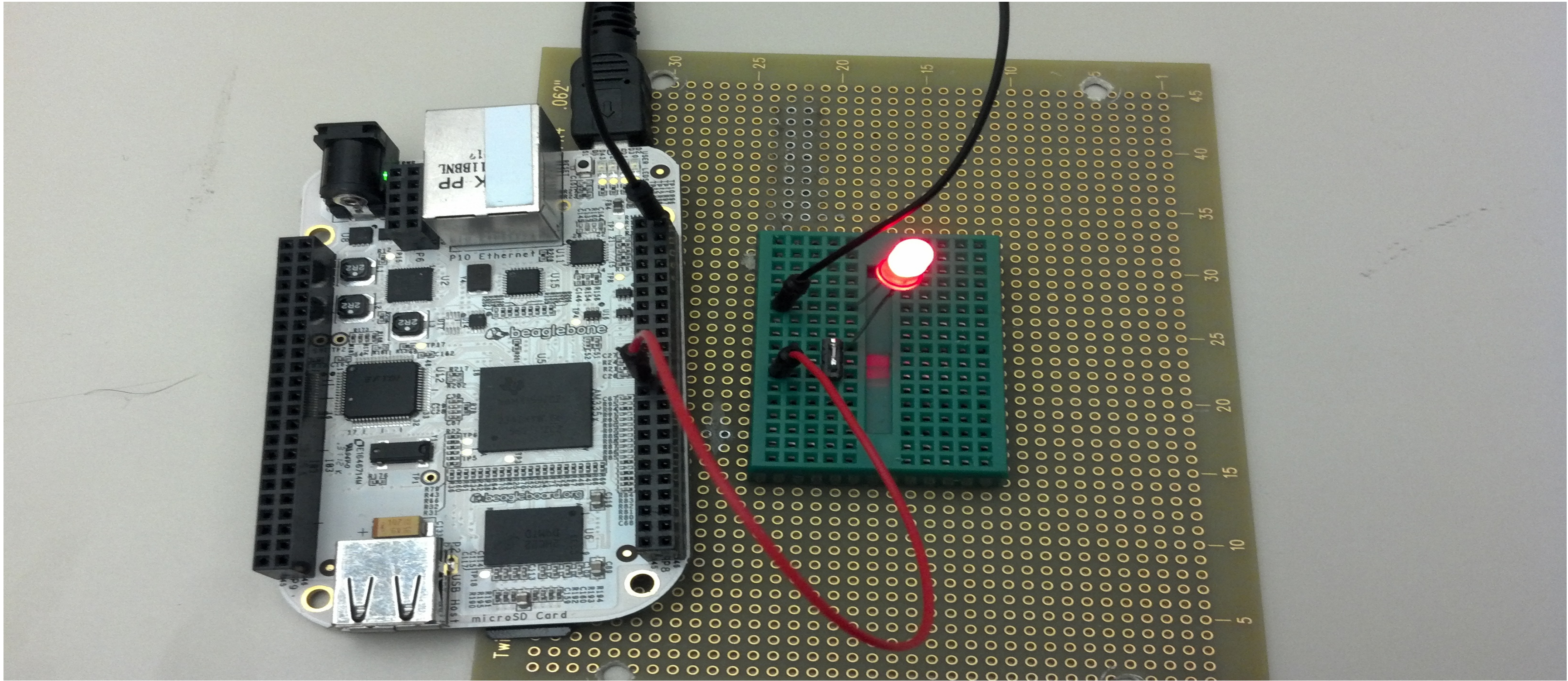
```
echo 1 > /sys/class/leds/beaglebone::usr3/brightness
echo 0 > /sys/class/leds/beaglebone::usr3/brightness
```



LED OFF



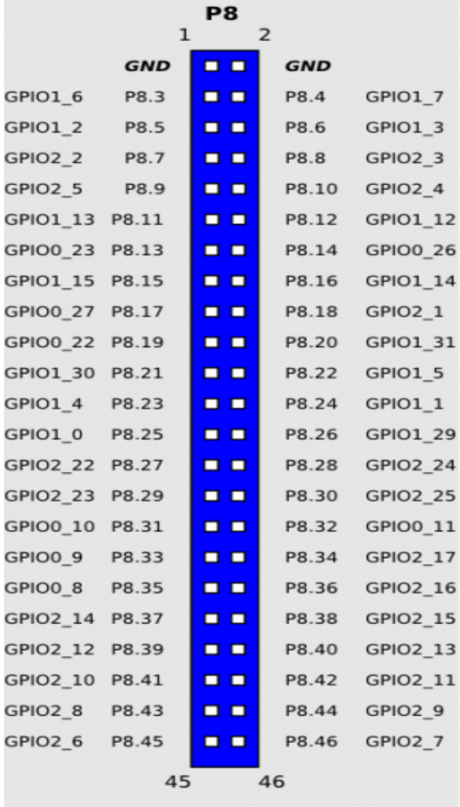
Tutorial – 2: GPIO Programming



Expansion headers

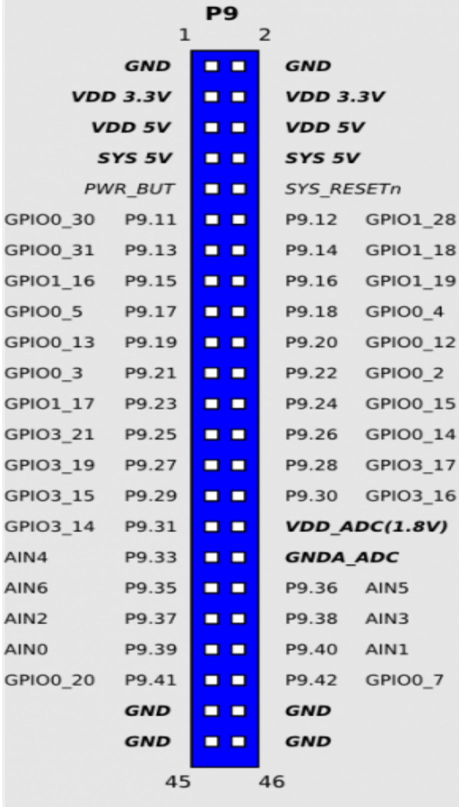
The expansion headers on the beaglebone are comprised of two 46 pin connectors which are P8 and P9. All signals on the expansion headers are 3.3V unless otherwise indicated.

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:



P8

		1	2		
	GND			***GND***	
GPIO1_6	P8.3			P8.4	GPIO1_7
GPIO1_2	P8.5			P8.6	GPIO1_3
GPIO2_2	P8.7			P8.8	GPIO2_3
GPIO2_5	P8.9			P8.10	GPIO2_4
GPIO1_13	P8.11			P8.12	GPIO1_12
GPIO0_23	P8.13			P8.14	GPIO0_26
GPIO1_15	P8.15			P8.16	GPIO1_14
GPIO0_27	P8.17			P8.18	GPIO2_1
GPIO0_22	P8.19			P8.20	GPIO1_31
GPIO1_30	P8.21			P8.22	GPIO1_5
GPIO1_4	P8.23			P8.24	GPIO1_1
GPIO1_0	P8.25			P8.26	GPIO1_29
GPIO2_22	P8.27			P8.28	GPIO2_24
GPIO2_23	P8.29			P8.30	GPIO2_25
GPIO0_10	P8.31			P8.32	GPIO0_11
GPIO0_9	P8.33			P8.34	GPIO2_17
GPIO0_8	P8.35			P8.36	GPIO2_16
GPIO2_14	P8.37			P8.38	GPIO2_15
GPIO2_12	P8.39			P8.40	GPIO2_13
GPIO2_10	P8.41			P8.42	GPIO2_11
GPIO2_8	P8.43			P8.44	GPIO2_9
GPIO2_6	P8.45			P8.46	GPIO2_7
		45	46		



P9

		1	2		
	GND			***GND***	
	VDD 3.3V			***VDD 3.3V***	
	VDD 5V			***VDD 5V***	
	SYS 5V			***SYS 5V***	
	PWR_BUT			*SYS_RESETn*	
GPIO0_30	P9.11			P9.12	GPIO1_28
GPIO0_31	P9.13			P9.14	GPIO1_18
GPIO1_16	P9.15			P9.16	GPIO1_19
GPIO0_5	P9.17			P9.18	GPIO0_4
GPIO0_13	P9.19			P9.20	GPIO0_12
GPIO0_3	P9.21			P9.22	GPIO0_2
GPIO1_17	P9.23			P9.24	GPIO0_15
GPIO3_21	P9.25			P9.26	GPIO0_14
GPIO3_19	P9.27			P9.28	GPIO3_17
GPIO3_15	P9.29			P9.30	GPIO3_16
GPIO3_14	P9.31			***VDD_ADC(1.8V)***	
AIN4	P9.33			***GNDA_ADC***	
AIN6	P9.35			P9.36	AIN5
AIN2	P9.37			P9.38	AIN3
AIN0	P9.39			P9.40	AIN1
GPIO0_20	P9.41			P9.42	GPIO0_7
	GND			***GND***	
	GND			***GND***	
		45	46		

GPIO

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:

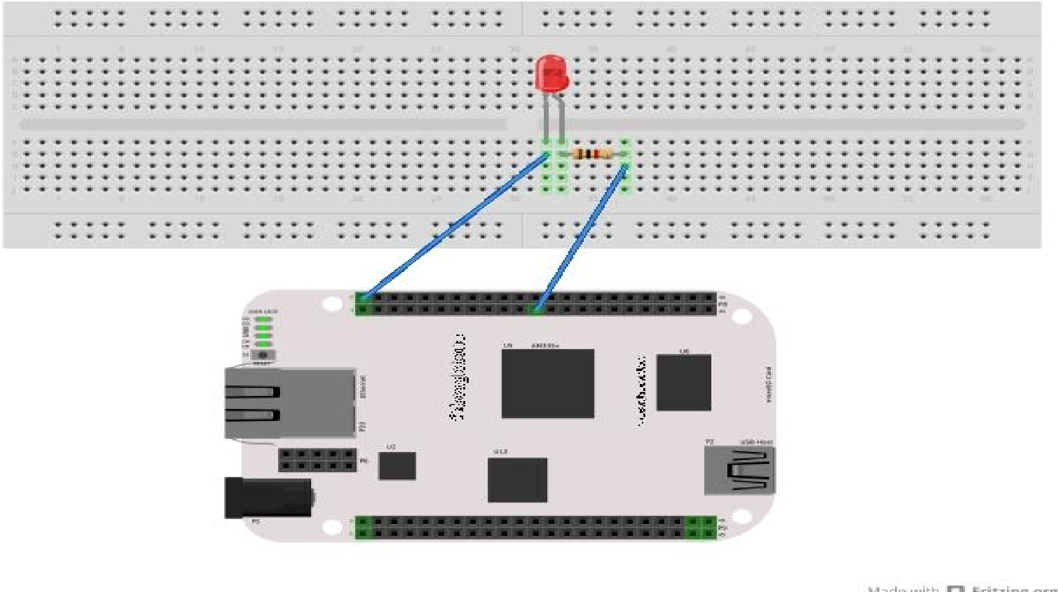
Reference number = Mx32 + Y

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.

Connection Diagram

Only the pins on the P8 expansion header are used in this case. Connect the anode of the LED to the 110 ohms resistor which in turn is connected to pin 23(GPIO) and connect the cathode of the LED to pin 2(GND) .



Light up the LED

Make sure that the pin you are using is configured to be in the gpio mode. So, run the following command in the terminal:

```
echo 7 > /sys/kernel/debug/omap_mux/gpmc_ad4
```

Export the pin

```
echo 36 > /sys/class/gpio/export
```

Since its a GPIO, we need to configure it in the output mode. Write the following commands in your terminal

```
echo out > /sys/class/gpio/gpio32/direction
```

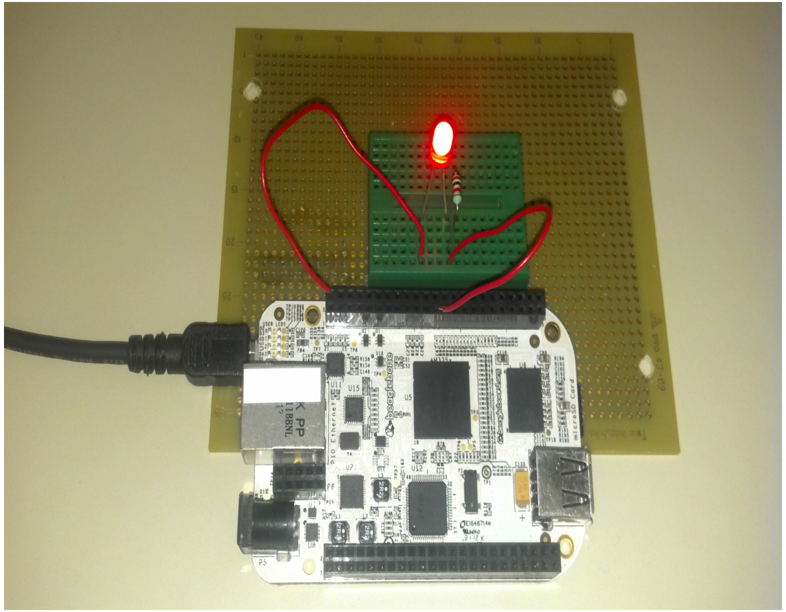
Now, let us toggle the LED by typing the following commands in terminal (First one is for turning ON and latter for OFF):

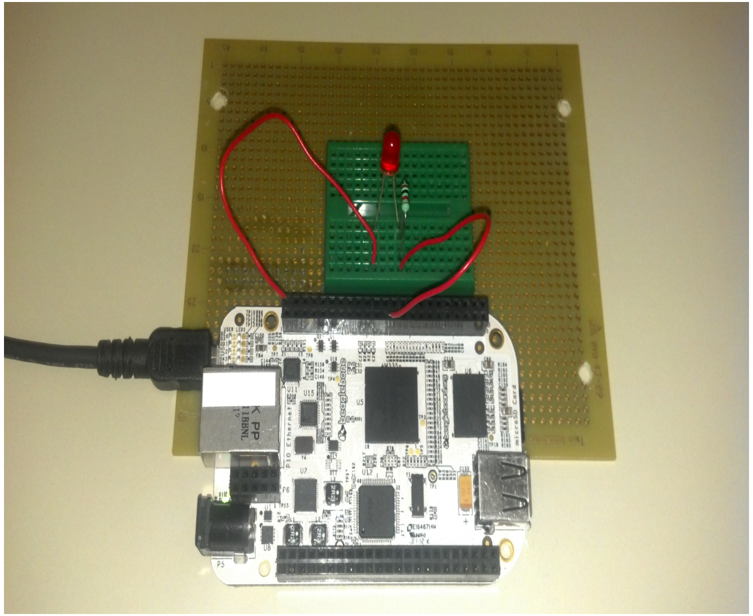
```
echo 1 > /sys/class/gpio/gpio32/value
echo 0 > /sys/class/gpio/gpio32/value
```

Unexport the pin once finished

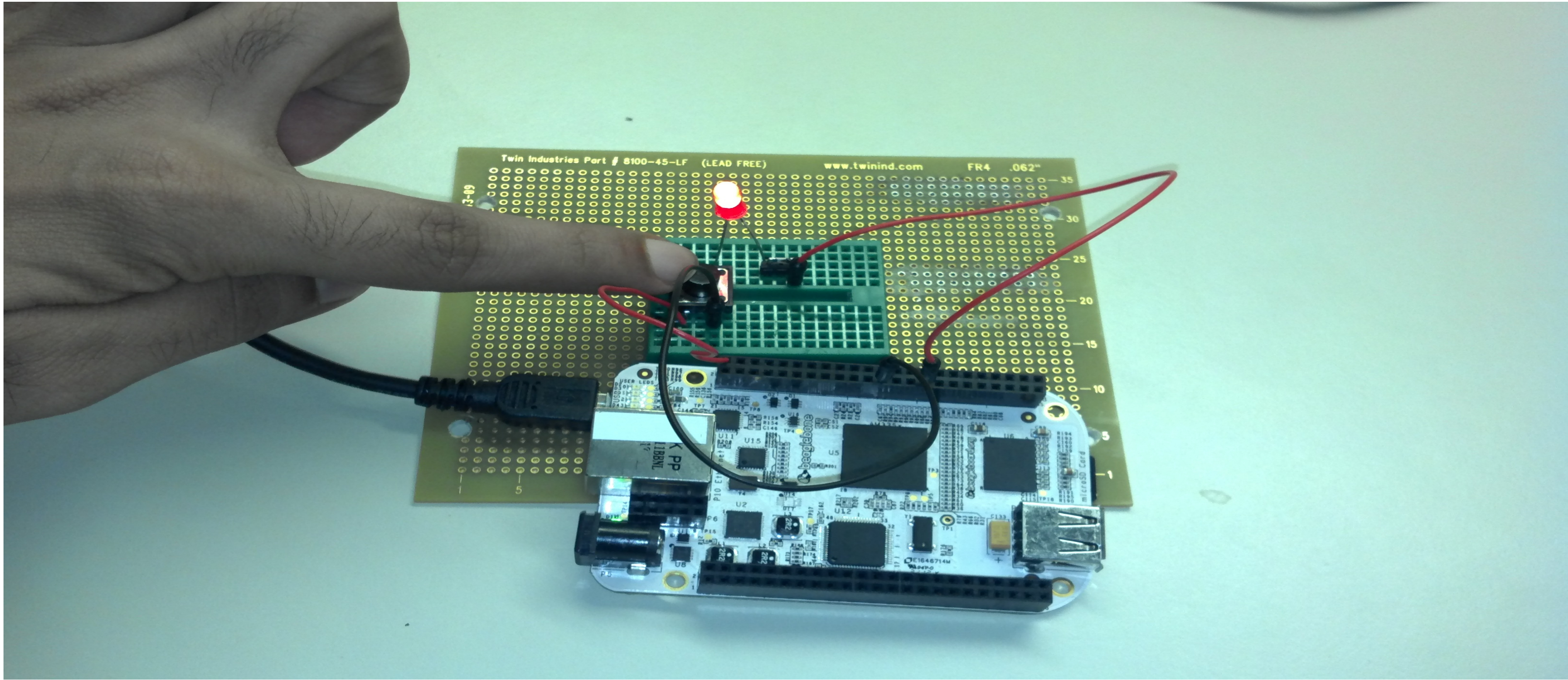
```
echo 36 > /sys/class/gpio/unexport
```

Output

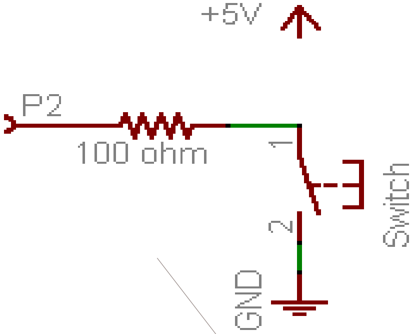




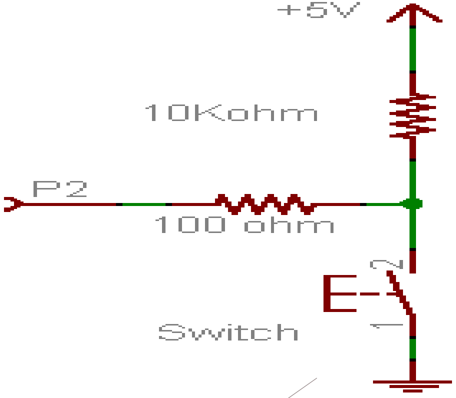
Tutorial – 3: Physical Computing



Pullup Resistors

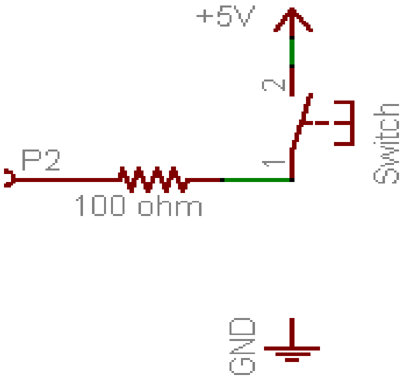


Circuit with no pullup resistor

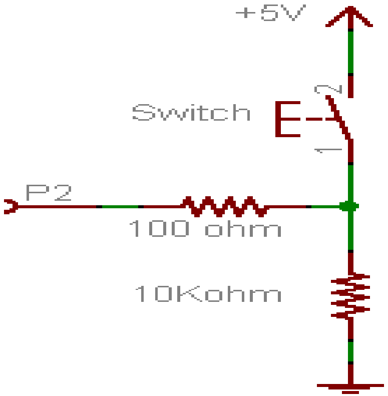


Circuit with pullup resistor

Pulldown Resistors

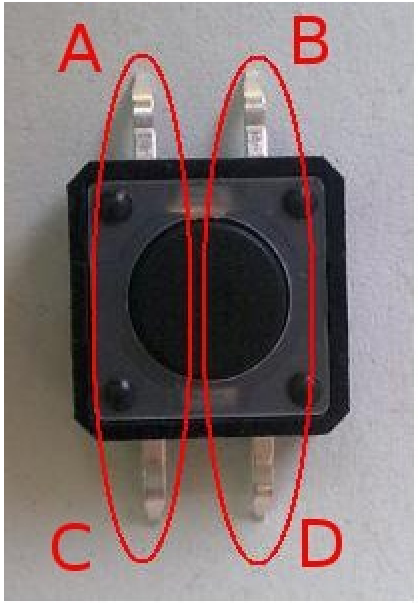


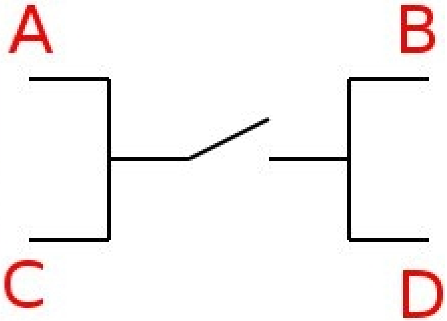
Circuit with no pulldown resistor



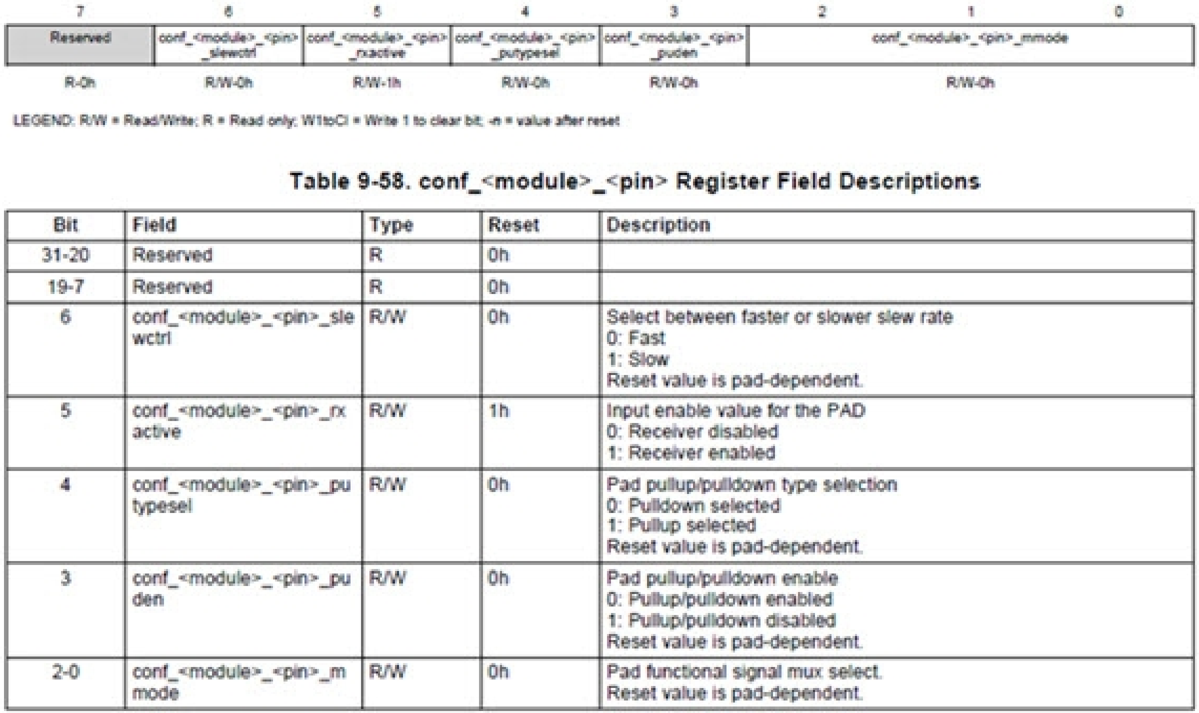
Circuit with pulldown resistor

Push button





Pin Mux Register



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.

Changing mode of the pin

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 belowname:

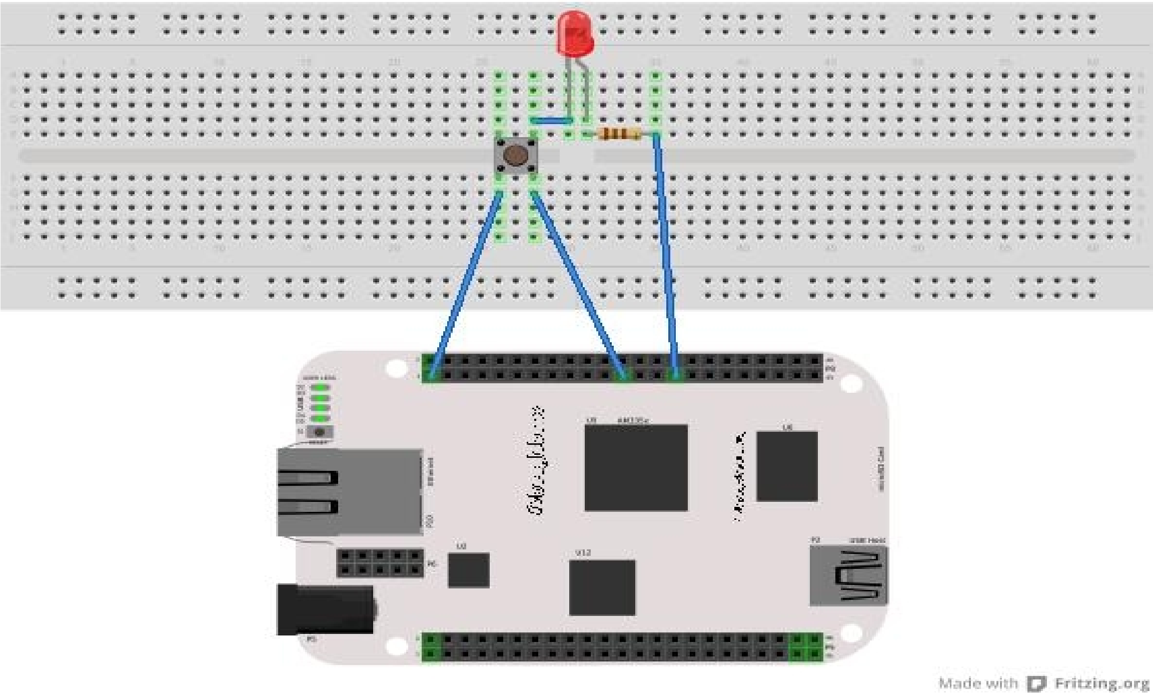
```
gpmc_ad4.gpio1_4 (0x44e10810/0x810 = 0x0027), b NA, t NA
mode: OMAP_PIN_INPUT_PULLDOWN | OMAP_MUX_MODE7
signals: gpmc_ad4 | mmc1_dat4 | NA | NA | NA | NA | NA | gpio1_4
```

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.

Mode-1: Using pullup resistor

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.



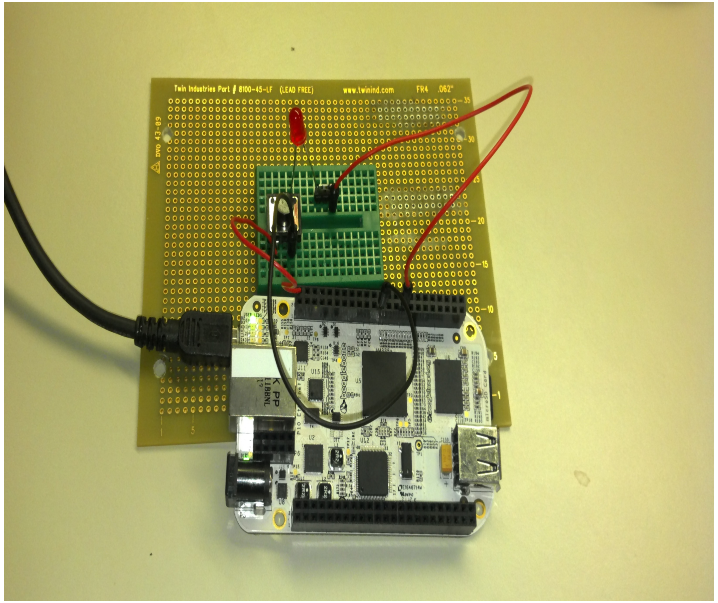
Code

```
#!/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
```

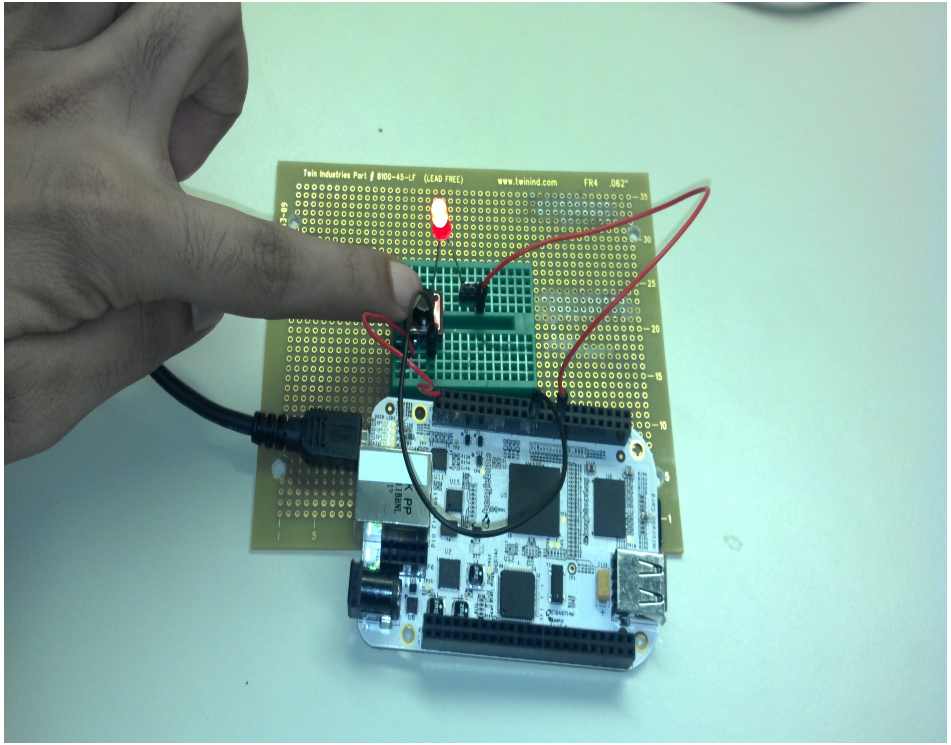
In the script, '#' is used for comments(except for first line). Write the above script in your favorite text editor, save it(make sure to add the '.sh' extension at the end) and place it in on the sd card before you boot the beaglebone. Or you can use a text editor like 'nano' on the beaglebone and write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

Output



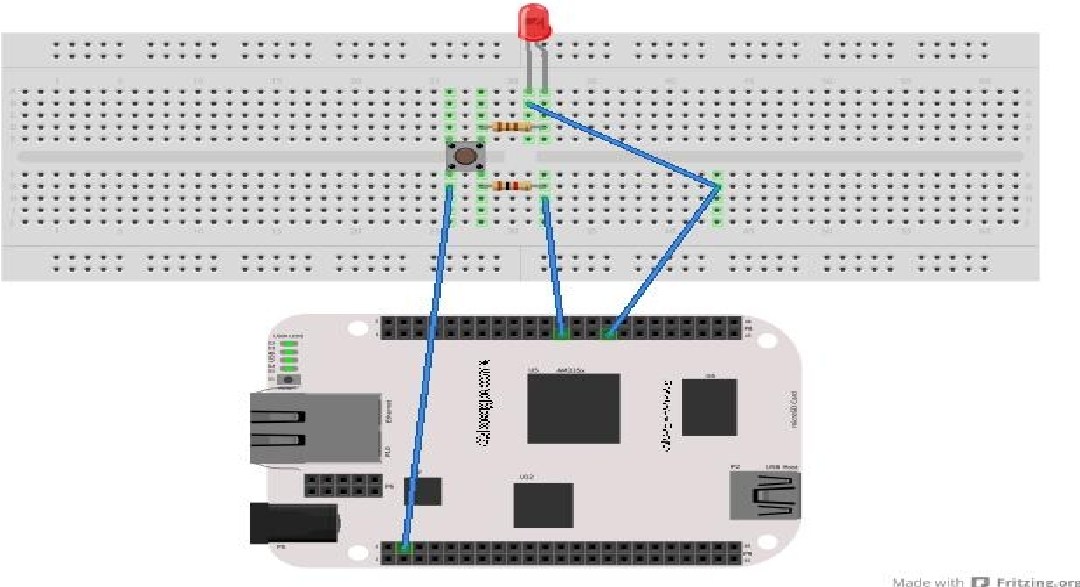
Twin Industries Part # 8100-45-LF (LEAD FREE)
www.twinind.com
FR4 .062"



Twin Industries Part # 8100-45-LF (LEAD FREE)
www.twinind.com
FR4 .062"

Mode-2: Using pulldown resistor

- Connect the pin-4(VDD or 3.3V) on P9 expansion header to leg-1 of the push button switch.
- 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.



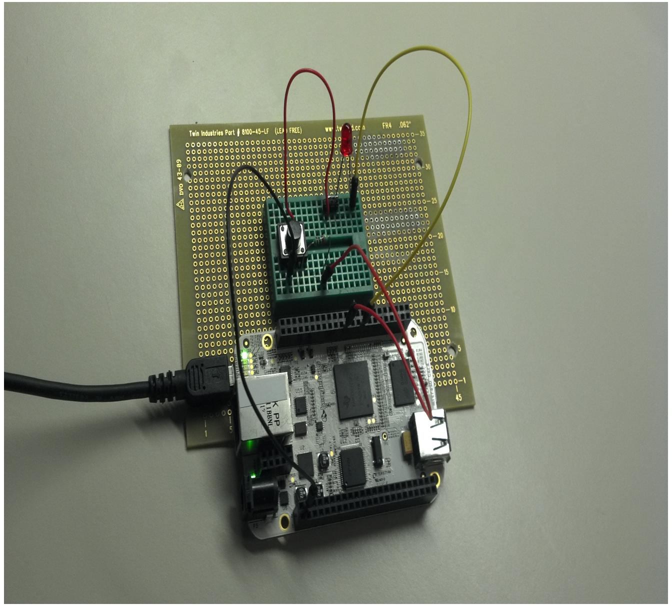
Code

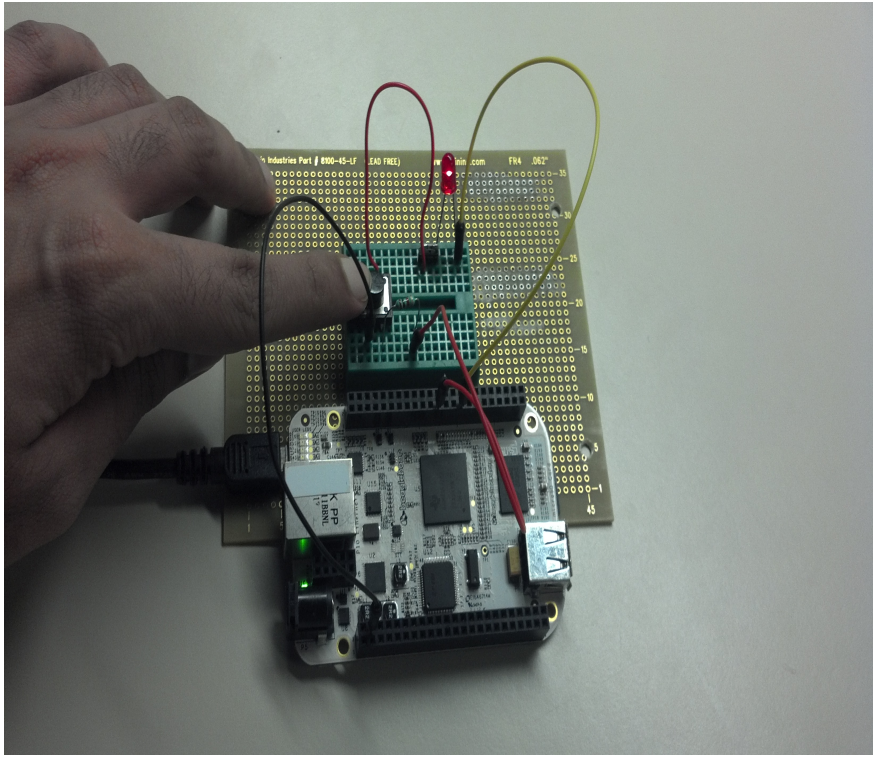
```
#!/bin/bash
#Open the GPIO port
#Pin no. 23 aka input pin
echo 7 > /sys/kernel/debug/omap_mux/gpmc_ad4
echo 36 > /sys/class/gpio/export
echo "in" > /sys/class/gpio/gpio36/direction
#Pin no. 29 aka output pin
echo 7 > /sys/kernel/debug/omap_mux/lcd_hysnc
echo 87 > /sys/class/gpio/export
echo "out" > /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 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>
```

Output


Twin Industries Part # 8100-45-LF (LEAD FREE)
FR4 .062"


Industries Part # 8100-45-LF (LEAD FREE)
FR4 .062"

Acknowledgements

I would like to thank the following people for their help and support:

- David Anders
- Jason Kridner
- Nishanth Menon

Contact

jayneil.dalal@gmail.com

http://elinux.org/Jayneil_Dalal

TRIVIA:

How many Beagle(s) are there in the presentation???