



presents

MinnowBoard

A Quick Start Guide



Jayneil Dalal

Scott Garman

Contents

0.1	About the Book	3
0.2	About the Authors	3
0.2.1	Jayneil Dalal	3
0.2.2	Scott Garman	3
1	Getting Started	4
1.1	Contents of the box	4
1.2	Anatomy of a MinnowBoard	5
1.3	Specifications	7
1.4	Preparing the MicroSD Card	7
1.5	Booting Angstrom	8
1.6	Windows Section	12
1.6.1	FTDI Driver	12
1.6.2	Serial Terminal	15
1.6.3	Preparing the microSD card	17
1.7	Mac Section	19
1.7.1	Serial Terminal	19
1.7.2	Preparing the microSD card	23
2	Toggling User LED	25
2.1	User LED on the MinnowBoard	25
2.2	Extra Credit	26
2.3	Steps	28
2.4	Output	29
3	GPIO Programming on the MinnowBoard	30
3.1	GPIO on the MinnowBoard	30
3.2	Selecting the correct LED	32
3.3	Extra Credit	32
3.4	Setup	33
3.5	Steps	33
3.6	Output	34

4	Physical Computing On The MinnowBoard	35
4.1	Pullup Resistors	35
4.1.1	Concept:	35
4.2	Push button	36
4.3	Physical Computing	37
4.4	Setup	37
4.5	Script	38
4.6	Steps	39
4.7	Output	40
4.8	Resources	41

0.1 About the Book

The purpose of this book is to provide a gentle walk through and get beginners quickly started with the MinnowBoard.

Suggestions/feedback about the book are always welcome. Feel free to shoot us an email at contact@minnowboard.org

Readers are more than welcome to contribute chapters to this book.

This work is licensed under the Creative Commons Attribution 3.0 Unported License. To view a copy of this license, visit

<http://creativecommons.org/licenses/by/3.0/>

0.2 About the Authors

0.2.1 Jayneil Dalal

Jayneil Dalal is a Technical Writer who loves to explore different open source technologies and is currently part of the Minnowboard.org project.

0.2.2 Scott Garman

Scott Garman is a Technical Evangelist for MinnowBoard at Intel. His background is in embedded Linux development, and he is a core team member of the Yocto Project. Scott lives in Portland, OR and lives to hack and go on epic bike rides.

Chapter 1

Getting Started

1.1 Contents of the box

When you purchase a MinnowBoard, the following items are included in the box as shown in Figure -1:

1. MinnowBoard
2. 5V Power Adaptor
3. 4GB micro SD card preloaded with Angstrom Linux Distribution(Yocto Project Compatible)
4. USB Cable

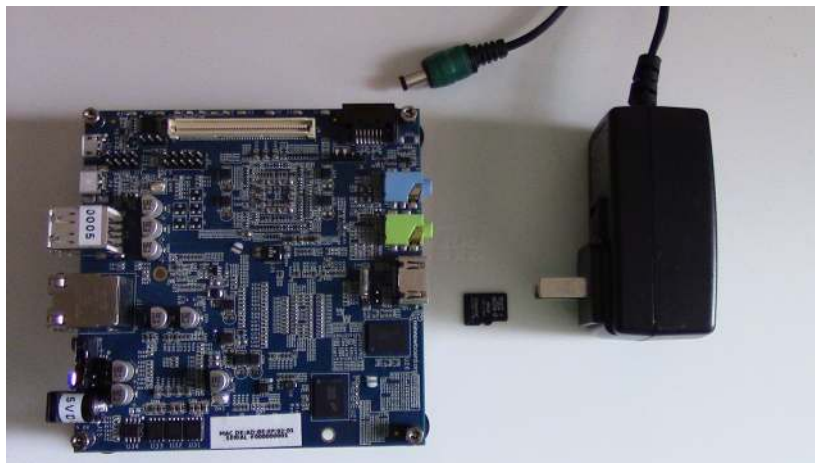


Figure-1: Box contents

1.2 Anatomy of a MinnowBoard

This section outlines the various components on the MinnowBoard. Figure -2 below annotates the important components on the top side of the MinnowBoard.

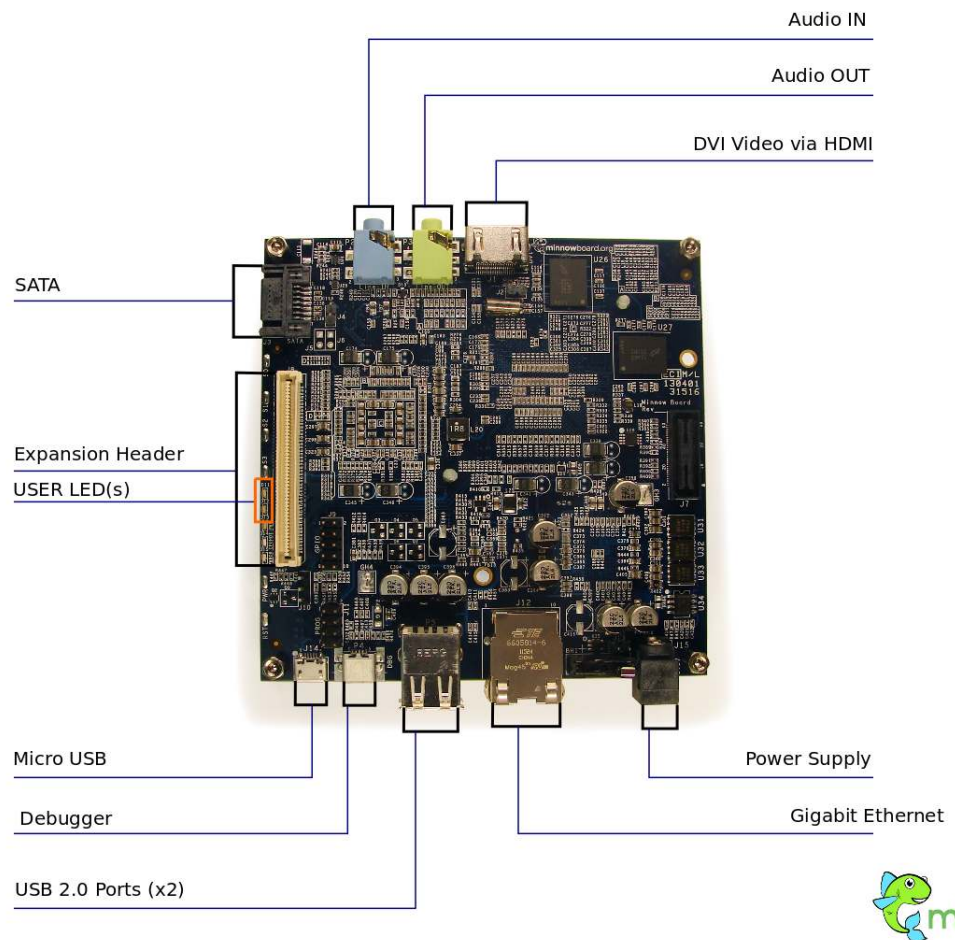


Figure-2: *MinnowBoard top side*

Below is a brief description of the micro USB and debugger ports which are annotated in Figure -2.

- Micro USB: This port can be used to access the contents of the MinnowBoard's microSD card as a removable drive on your desktop computer. Simply boot the MinnowBoard up and connect it to your desktop computer using a micro-USB cable.
- Debugger: This mini-USB port allows you to interact with the serial console of the MinnowBoard, so you can log into it and view debugging messages even when an external monitor and keyboard are not connected to the MinnowBoard. You need to use a terminal emulator on your desktop computer to access the serial console at a baud rate of 115200. This is demonstrated later in this guide in the Booting Angstrom section.

Figure -3 below annotates the important components on the bottom side of the MinnowBoard.

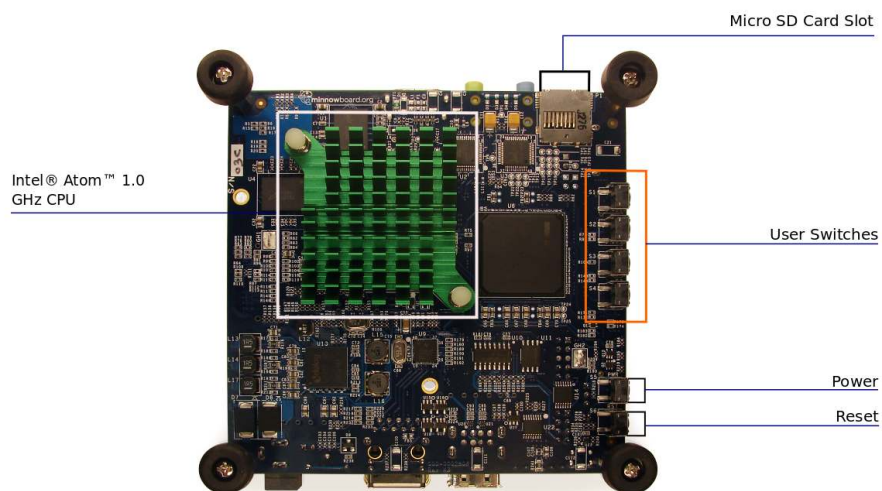


Figure-3: *MinnowBoard bottom side*

1.3 Specifications

For detailed specifications of the MinnowBoard, please visit our website below:

<http://www.minnowboard.org/technical-features/>

1.4 Preparing the MicroSD Card

Please note that this section is completely optional. The MinnowBoard already comes with a microSD card that is preloaded with a working Angstrom Linux image. In case you want to use a newer image or want to program the microSD from scratch, this section covers it all. These steps were created using an Ubuntu 12.10 64-bit system.

Note:- This section is for Linux users only. If you are using Windows, please skip to page 12 . If you are using Mac, please skip to page 19

- First download the latest Angstrom Linux image for MinnowBoard using the following link:

<http://dominion.thruhere.net/koen/angstrom/minnow/production-Angstrom-development-GNOME-image-eglibc-ipk-v2012.12-minnow-2013.07.10.img.xz>

At the time of writing this guide, the latest image available for download was 'Angstrom-development-GNOME-image-eglibc-ipk-v2012.12-minnow-2013.07.10.img.xz'

- Insert the microSD card into a microSD compatible writer in your desktop computer. Now, identify the disk device filename for your microSD card. You can do this by opening the Disk Utility application (sometimes also called Disks) in Ubuntu and clicking on the microSD card entry. See the Device field to determine the raw device name of your microSD card. Note that this device name should specify an entire disk device (e.g, /dev/sde), and not an individual disk partition (e.g, /dev/sde1)
- Now, make sure all possible disk partitions from the microSD card are unmounted by typing the command below into your terminal:

```
$ sudo umount /dev/sdX?*
```

Here 'sdX' stands for the raw device id of the microSD card.

- Now unpack the image to the microSD card by typing the command below in a terminal window:

```
$ xzcat Angstrom-development-GNOME-image-eglibc-ipk-v2012.12-minnow-2013.07.10.img.xz | sudo dd of=/dev/sdX
```


1.5 Booting Angstrom

- First insert the card into the microSD card slot label side up as shown in Figure -4. Push the microSD card into the slot until it clicks into place.

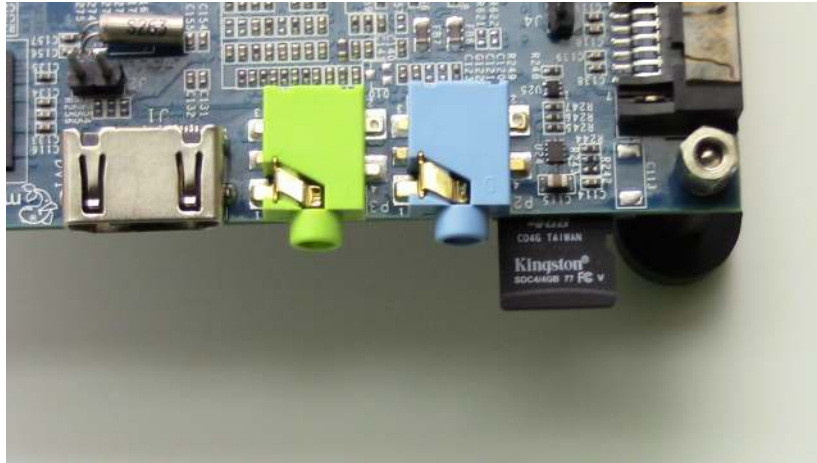


Figure-4: Connecting the micro SD card

- Next, connect an external keyboard and mouse to the MinnowBoard. Then connect a USB mini cable to your computer and the MinnowBoard as shown in Figure -5. This connection will be used to access the serial console of the MinnowBoard.

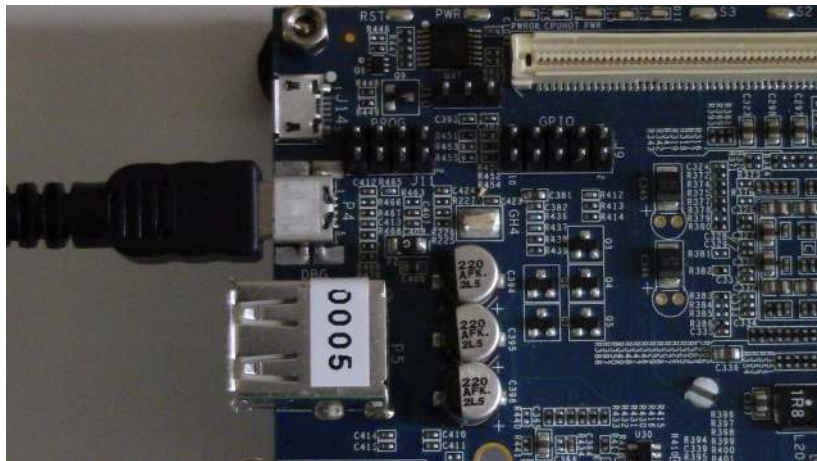


Figure-5: Connecting a USB mini cable for serial console access

- Now power up the MinnowBoard, connect the 5V/2.5A power supply to it as shown in Figure -6:

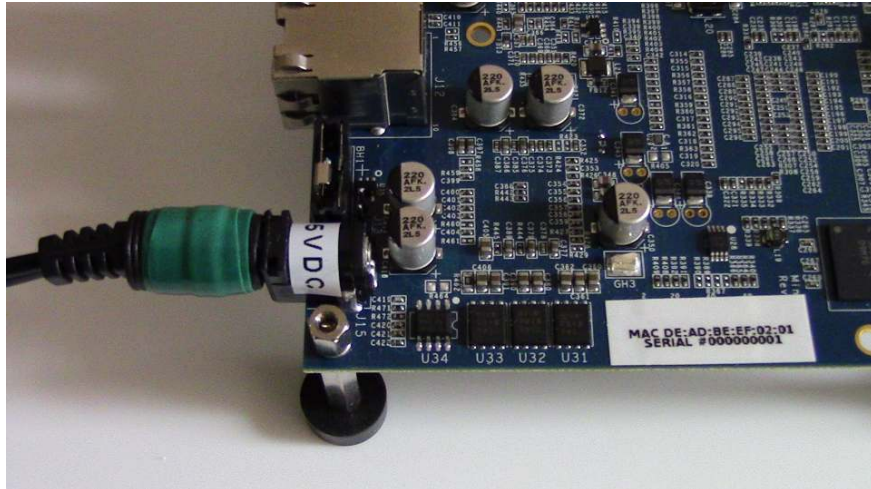


Figure-6: Powering up the MinnowBoard

- Access the MinnowBoard's serial console via a terminal emulator:

```
$ sudo screen /dev/ttyUSB0 115200
```

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

- You should see a similar Angstrom login screen as shown in Figure - 7. The username is 'root' and there is no password - just press the Enter key.

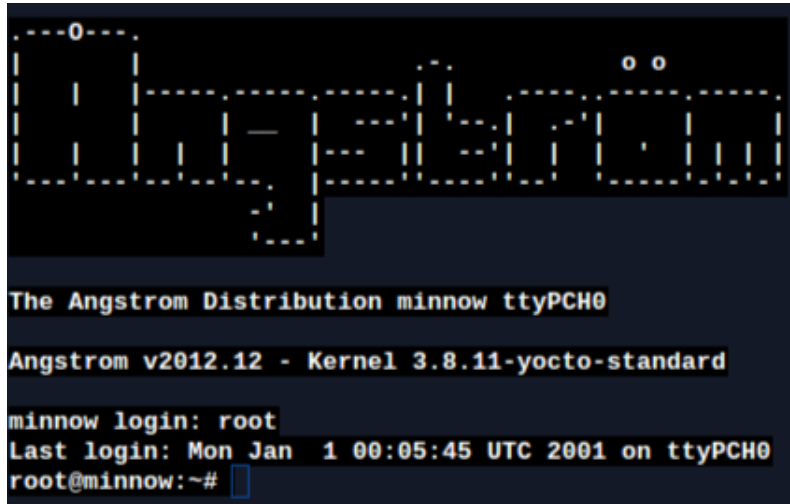


Figure-7: Angstrom

- If you have an external display connected to the MinnowBoard via the HDMI port, you should see output similar to that shown below in Figure - 8



Figure-8: Angstrom

Figures - 9,10 show my setup.

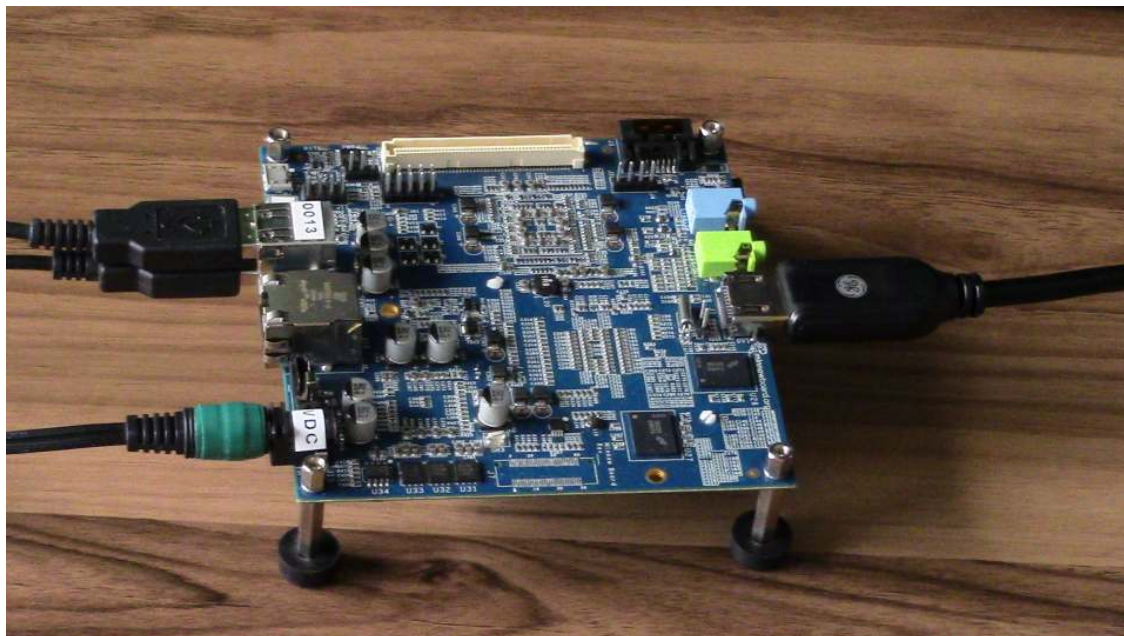


Figure-9: My Setup



Figure-10: My Setup

1.6 Windows Section

This section is for those who are using a Windows machine to interact with the MinnowBoard.

1.6.1 FTDI Driver

- Please go to the FTDI Driver(it enables communication between the development board and computer over USB) website below:

<http://www.ftdichip.com/Drivers/VCP.htm>

- Now click on the highlighted "setup executable" link under the 'Category' section as shown in Figure - 11 to download the driver.

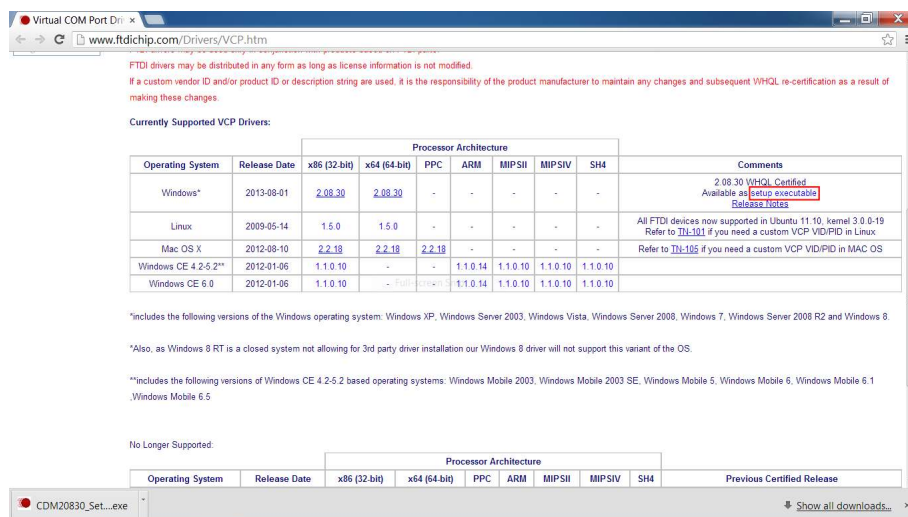


Figure-11: Download FTDI Driver

- Double click on the downloaded file to start the installation process. Then click on the 'Extract' button as shown in Figure - 12 below:

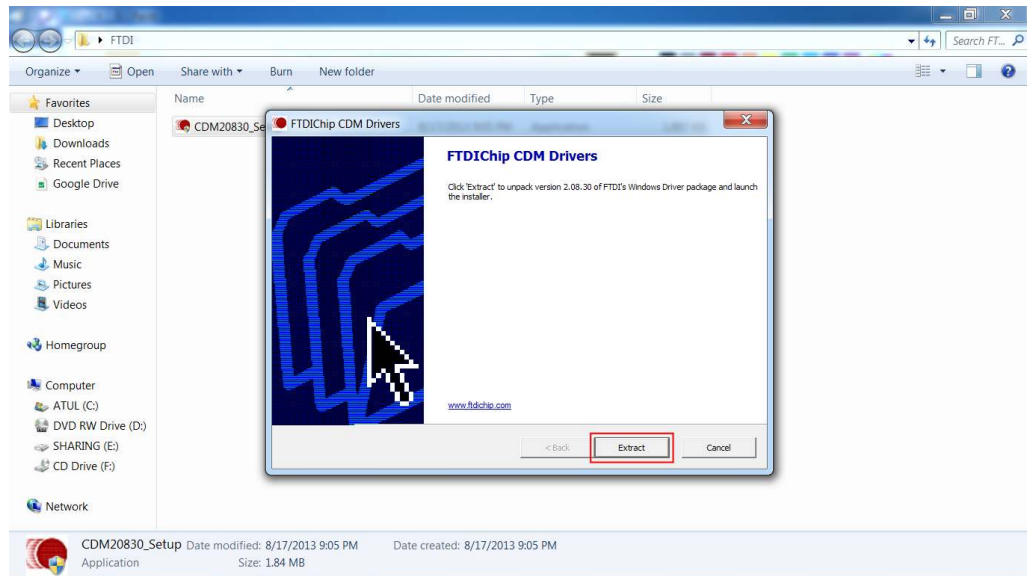


Figure-12: Extract the driver

- You will now be greeted with the driver installation wizard. Click on the 'Next' button and proceed forward as shown in Figure - 13.



Figure-13: Device Driver Installation Wizard

- If the installation was successful, you should be greeted by a screen similar to the one shown in Figure - 14. Click 'Finish' to exit the installation.

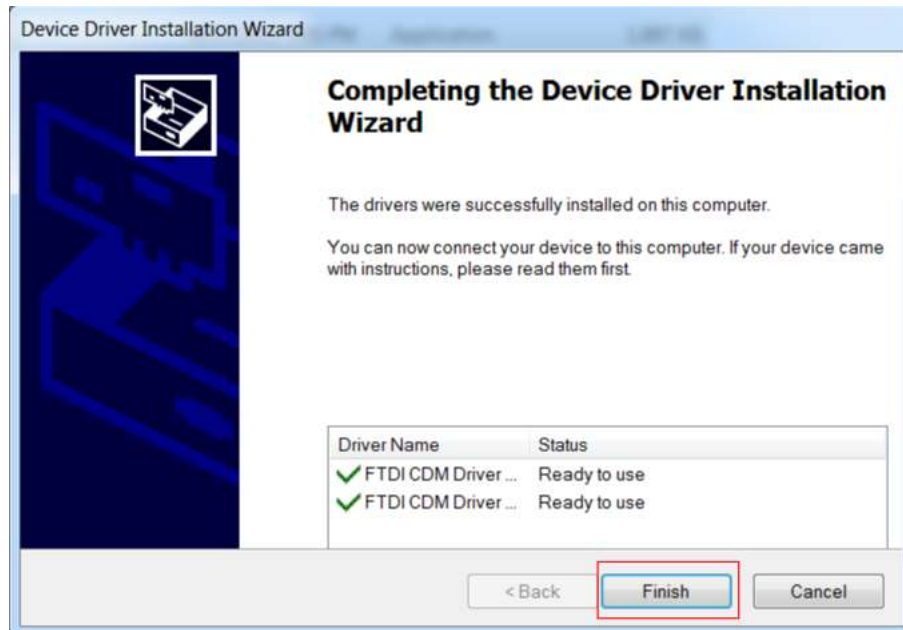


Figure-14: FTDI Driver installation successful

1.6.2 Serial Terminal

We will be using Teraterm as our terminal application. Please follow the steps below to set up Teraterm:

- Download the latest version of Teraterm using the link below:

<http://en.sourceforge.jp/projects/ttssh2/releases/>

At the time of writing this guide, the latest version was 'teraterm-4.78.exe'

- Install the Teraterm software by double clicking on the exe file downloaded earlier. If it displays a security warning, accept it and click on 'Run' as shown in Figure - 15 below:

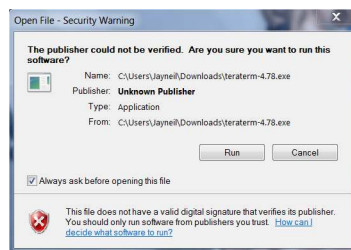


Figure-15: Teraterm installation

- Now just follow the on-screen instructions to install the software. Select the 'standard installation' as shown in Figure - 16 below:

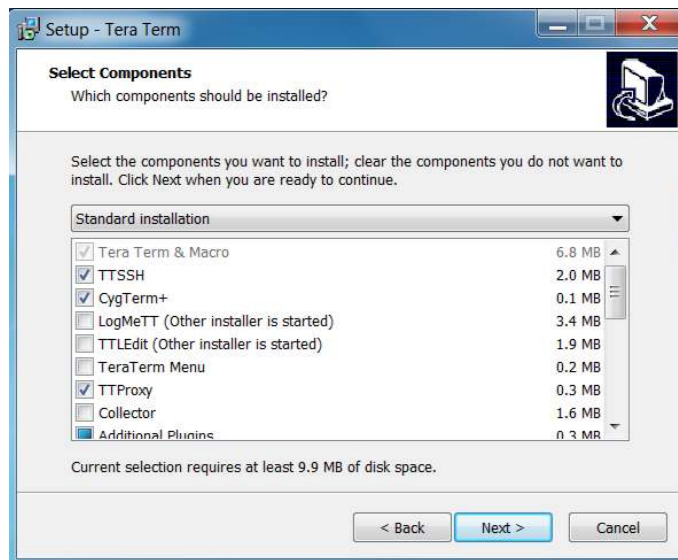


Figure-16: Perform Teraterm full installation

- Launch TeraTerm. Then go to Setup>Serial port and change the settings as shown in Figure - 17 below:

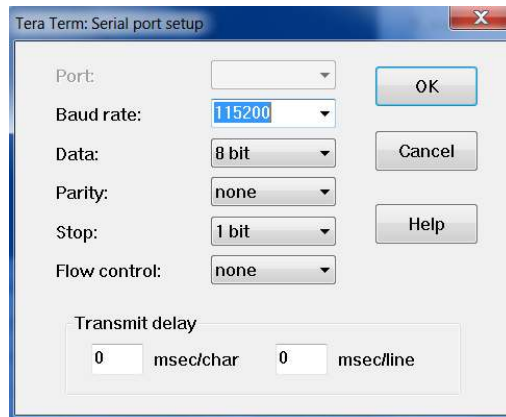


Figure-17: TeraTerm Serial Port setup

- Then go to File>New connections and select the Serial option and chose the appropriate COM port as shown in Figure - 18 below



Figure-18: Setting up a new connection

- Then connect the required accessories as described earlier in this guide and power up the MinnowBoard. If all went well, you should see a login screen in TeraTerm as shown in Figure - 19 below:

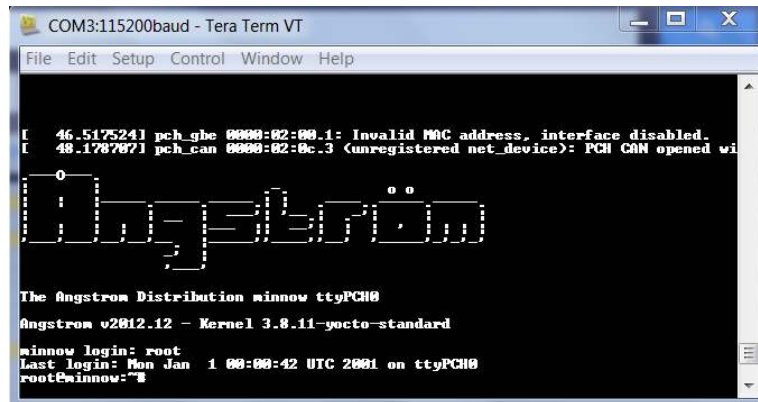


Figure-19: Angstrom login

1.6.3 Preparing the microSD card

- First, download the latest Angstrom Linux image from the link below:

<http://dominion.thruhere.net/koen/angstrom/minnow/production-Angstrom-development-GNOME-image-eglbc-ipk-v2012.12-minnow-2013.07.10.img.xz>

- Next, download the 7-zip utility from the link below and install it:

<http://www.7-zip.org/download.html>

- Decompress the previously downloaded Angstrom image using 7-zip as shown in Figure - 20 below. The resulting file should have an extension '.img'.

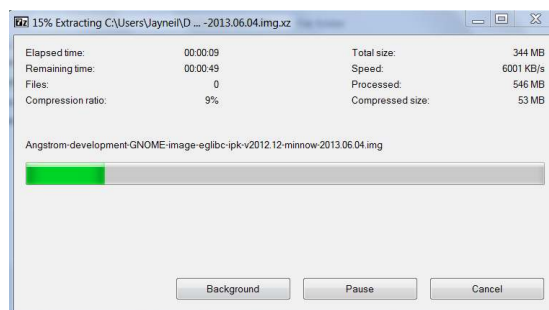


Figure-20: Decompressing the image

- Image Writer for Windows is needed to write the .img file to a microSD card. Download and install Image Writer using the link below: :

<http://sourceforge.net/projects/win32diskimager/files/latest/download>

- Insert the microSD card into the computer using an appropriate adapter. Next, launch the image writer software we just installed earlier. Select the Angstrom image and write it to the microSD card as shown in Figure - 21 below.

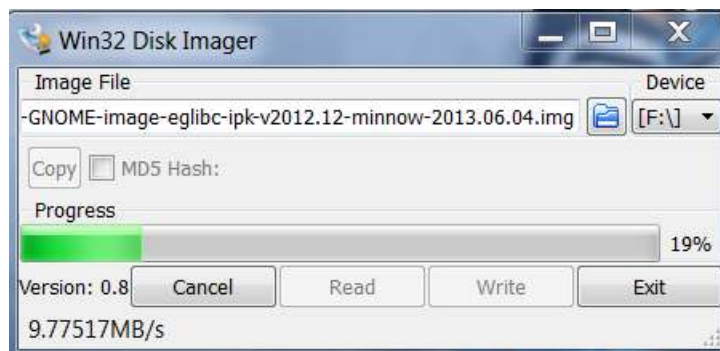


Figure-21: Writing the image to microSD card

1.7 Mac Section

This section is for those who are using OS X to interact with the MinnowBoard.

1.7.1 Serial Terminal

We will be using the 'screen' command which is by default a part of the bash shell on OS X. To use screen command, the FTDI driver(it enables communication between the development board and computer over USB) has to be installed. Please follow the steps below to install it:

- Download the latest ftdi driver for Mac OS X from the link below and make sure that you select the correct file corresponding to your computer architecture(32/64 bit).
<http://www.ftdichip.com/Drivers/VCP.htm>
- Double click on the downloaded file to start the installation procedure.
- You will have two files inside the downloaded driver package as shown below in Figure - 22. The first file, ending with 10_3 is specific for Mac OS X version 10.3 . The second file is for all other versions, newer than 10.3 .

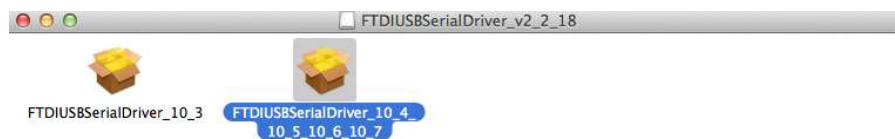


Figure-22: FTDI driver

- Based on your Mac OS X version, select the appropriate and double click it to proceed further.

- You will now be greeted with an ftdi installer screen as shown in Figure - 23.



Figure-23: FTDI installer screen

- Proceed forward by pressing the 'continue' button till you reach the "installation type" screen as shown below in Figure -24.

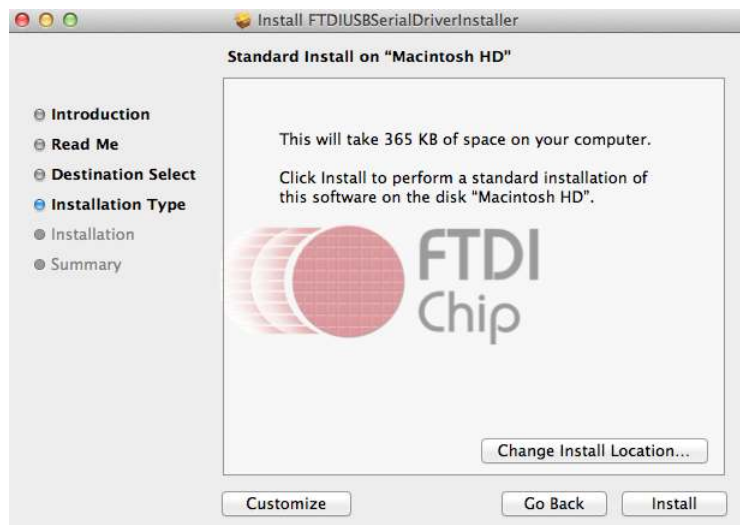


Figure-24: FTDI driver installation

- Now press the 'install' button and wait for it to finish as shown in Figure - 25.

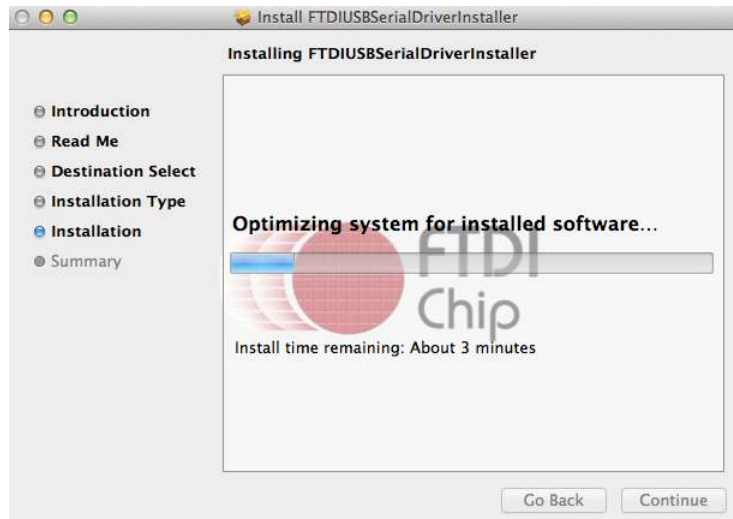


Figure-25: FTDI driver installation in progress

- If the installation was successful, you should be greeted by a screen similar to the one shown in Figure - 26

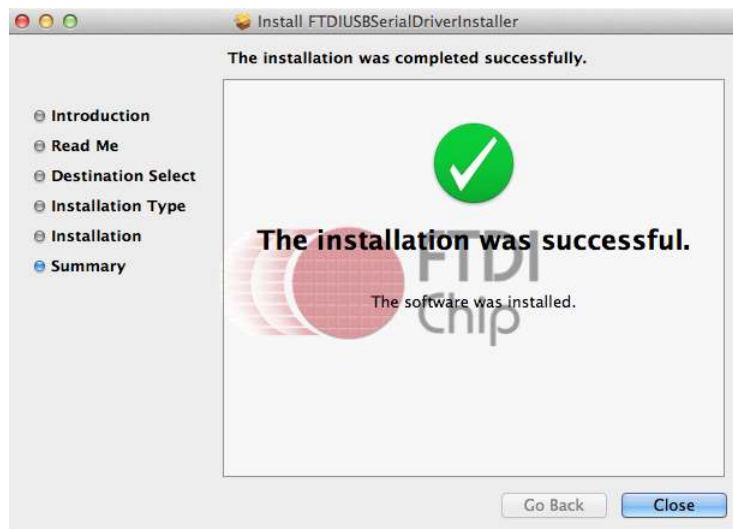


Figure-26: FTDI driver installation successful

- Then connect the required accessories as described earlier in this guide and power up the MinnowBoard.
- If everything went well, you will see new entries in the /dev directory:
/dev/cu.usbserial-xxxxxxx
/dev/tty.usbserial-xxxxxxx

Here, xxxxxxxx is either the device's serial number or, for unserialized devices, a location string that depends on which USB port your device is connected to. /dev can be accessed through the Terminal application. The Terminal application can be launched by selecting Go > Applications > Utilities > Terminal. Now fire up the terminal in Mac OS X and type the following command in it:

```
$ cd /dev  
$ ls-l
```

- Access the MinnowBoard's serial console via a terminal emulator and you should be greeted by the Angstrom screen as shown in Figure - 27:

```
$ sudo screen /dev/tty.usbserial-xxxxxxx 115200
```

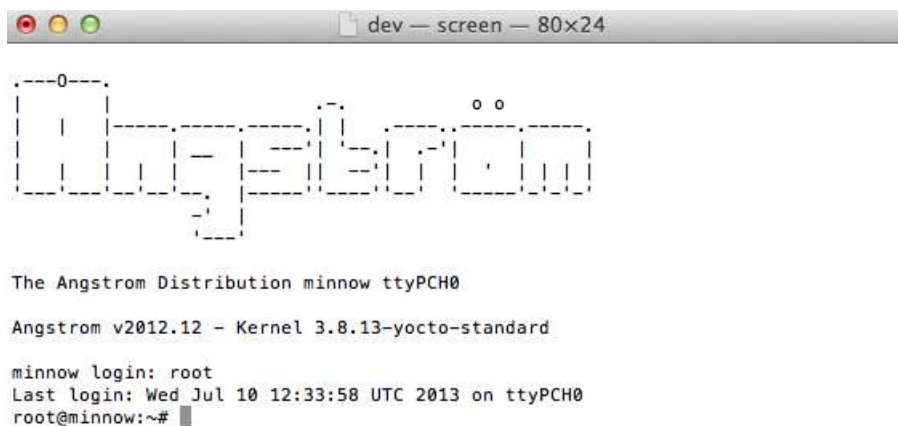


Figure-27: Angstrom

1.7.2 Preparing the microSD card

- First, download the latest Angstrom Linux image from the link below:

<http://dominion.thruhere.net/koen/angstrom/minnow/production-Angstrom-development-GNOME-image-eglibc-ipk-v2012.12-minnow-2013.07.10.img.xz>

- Next, install the “The Unarchiver” application(it is free) from the app store on Mac OS X as shown in Figure - 28



Figure-28: The Unarchiver

- Then unpack the downloaded angstrom image using the unarchiver application as shown below in Figure -29



Figure-29: Unpacking the Angstrom image

- Insert the microSD card into a microSD compatible writer in your computer. Now, identify the disk device filename for your microSD card. You can do this by typing the command below, before and after you insert the microSD card and comparing the outputs to see the newly added device:

```
$ df-h
```

Note:- If you notice closely, the microSD card entry will be in the form of /dev/diskNsX where 'N' and 'X' are numbers. diskNsX is basically the device id of your microSD card. So, to obtain the raw device id from this, remove the 's' and 'X' from the device id. The raw device id should be in the form of diskN .

- Now, make sure all possible disk partitions from the microSD card are unmounted by typing the command below into your terminal:

```
$ sudo umount /dev/diskN?*
```

Here 'diskN' stands for the raw device id of the microSD card.

- Now unpack the image to the microSD card by typing the command below in a terminal window:

```
$ sudo dd bs=1m if=Angstrom-development-GNOME-image-eglibc-ipk-  
v2012.12-minnow-2013.07.10.img of=/dev/rdiskN
```

Note:- In the above command we subtly replaced "diskN" with "rdiskN" because in the latter case, you are writing to a buffered device and hence the process becomes much faster!

Chapter 2

Toggling User LED

2.1 User LED on the MinnowBoard

There are a total of five LED(s) on the MinnowBoard - D11, D12, D13, D14, D15 as shown in Figure -30. Their functions have been listed in the table below:

Sr. No.	LED	Function
1.	D11	Heartbeat/USER LED*
2.	D12	microSD card activity/USER LED*
3.	D13	PWR
4.	D14	CPUHOT
5.	D15	PWROK

*You need to change the trigger for it to function as USER LED

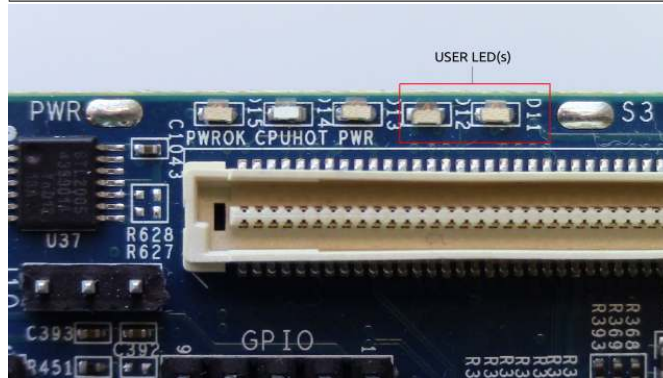


Figure-30: *LED(s) on the MinnowBoard*

Now, let us understand these functions in a bit more detail:

1. Heartbeat: The Heartbeat LED blinks at a regular rate to indicate that the hardware is operating properly, and that the CPU hasn't locked up. This is useful in embedded systems where you may not always have a board connected to a display to confirm its operation.
2. microSD Card Activity: This LED blinks when there is disk activity happening on the microSD card port.
3. CPUHOT - When this LED is on, it indicates that the CPU temperature is abnormally high. This may suggest that the MinnowBoard is running in too hot of an environment, or that there is a problem with the CPU heatsink attachment. The MinnowBoard will shut off in the event that the CPU temperature reaches a level that could possibly damage itself, and CPUHOT represents a warning temperature before this state is reached.
4. PWROK - This LED indicates that the power source is operating within required tolerances for the MinnowBoard.

So, there are two USER LED(s) on the MinnowBoard - D11 and D12 but by default they cannot be used as USER LED(s).

The user LED(s) are accessible via the user space in Linux at the location below on the filesystem:

```
/sys/class/leds
```

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

```
/sys/class/leds/minnow_led0  
/sys/class/leds/minnow_led1
```

Here, led0 is D11, while led1 is D12. 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.

2.2 Extra Credit

This is an optional section which you can read for further understanding. In a nutshell, we are trying to access the on board USER LED(s) via userspace¹ in Linux. To be more precise we are using the sysfs² interface. sysfs is a virtual filesystem which translates the hardware devices and busses attached to the system(board in our case) into a file system hierarchy that can be accessed from userspace. sysfs is generated by the kernel and always mounted at /sys. As

¹<http://elinux.org/images/4/4f/02-linux-quick-start.pdf>

²<http://www.makelinux.net/books/lkd2/ch17lev1sec8>

discussed earlier, trigger for D11 and D12 LEDs has to be changed to make them function as USER LEDs. I would describe 'trigger' as the API used to link a LED to an 'event' in kernel space. Here, 'event' could be microSD card or Ethernet activity, heartbeat, power etc. To understand this better, take LED D11 for example. The trigger for D11 has been set by default to heartbeat. So, to use it as an USER LED, we have to change the trigger for it to 'none' as shown in Figure - 31 below:

```
root@minnow:/sys/class/leds/minnow_led0# ls
brightness device max_brightness power subsystem trigger uevent
root@minnow:/sys/class/leds/minnow_led0# cat trigger
none mmc0 mmc1 timer oneshot [heartbeat] backlight gpio cpu0 cpu1 default-on
root@minnow:/sys/class/leds/minnow_led0# echo none > trigger
```

Figure-31: Change LED Trigger

2.3 Steps

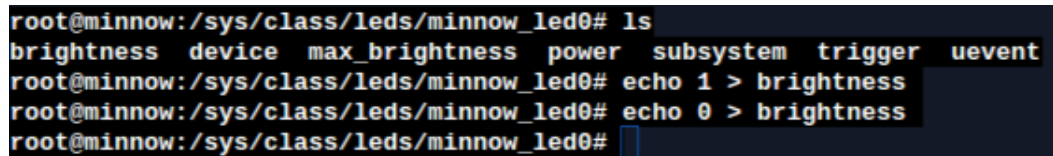
- Now let us control that LED!

To use D11 as USER LED, we need to change its default trigger. Type the following command in your terminal to accomplish this:

```
echo none > /sys/class/leds/minnow_led0/trigger
```

Now we are ready to toggle the LED. Type the following commands in your terminal as shown in Figure - 32 (First one is for turning ON and latter for OFF):

```
echo 1 > /sys/class/leds/minnow_led0/brightness  
echo 0 > /sys/class/leds/minnow_led0/brightness
```



```
root@minnow:/sys/class/leds/minnow_led0# ls  
brightness device max_brightness power subsystem trigger uevent  
root@minnow:/sys/class/leds/minnow_led0# echo 1 > brightness  
root@minnow:/sys/class/leds/minnow_led0# echo 0 > brightness  
root@minnow:/sys/class/leds/minnow_led0#
```

Figure-32: Toggle the LED

2.4 Output

You should see the LED change state as shown below:

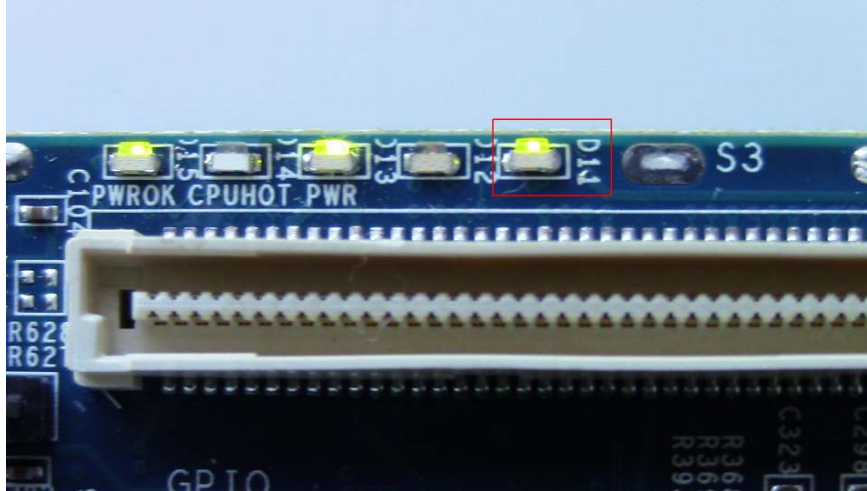


Figure-33: Led ON

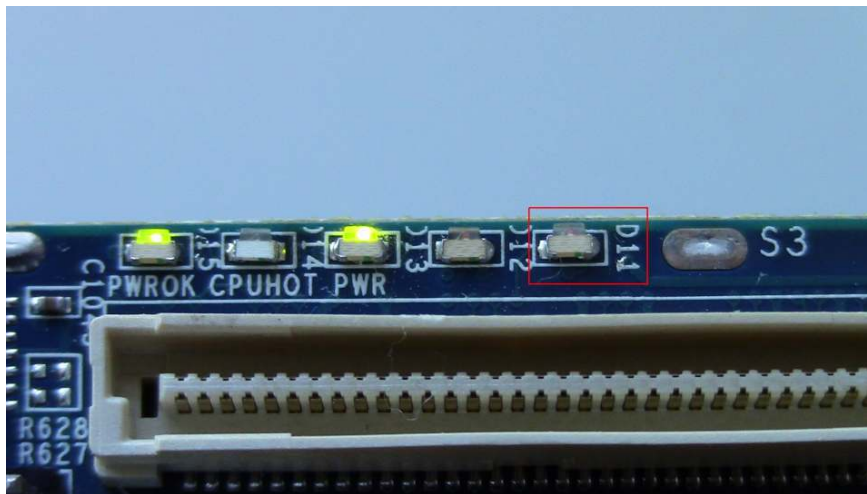


Figure-34: LED OFF

Chapter 3

GPIO Programming on the MinnowBoard

3.1 GPIO on the MinnowBoard

The MinnowBoard provides a variety of GPIO(General Purpose Input Output), some with a dedicated purpose. For the purpose of this guide, I will be concentrating only on GPIO(s) which are on the J9 expansion header as shown in Figure - 35. Please refer the table below to find out how are the GPIO(s) actually referenced in the kernel and their default modes as well as values. Unless otherwise stated the GPIO(s) on J9 expansion header are rated at 3.3V and can source/sink a maximum of 10mA. Also, it can be seen from the table that the GPIO(s) are by default set to be used in INPUT mode and have PULL-UP resistors enabled.

Sr. No.	GPIO	Reference number in the Kernel*	Default Mode	Default Value
1.	1	N.A.	N.A.	3.3V
2.	2	N.A.	N.A.	GND
3.	3	244	Input	HIGH
4.	4	245	Input	HIGH
5.	5	246	Input	HIGH
6.	6	247	Input	HIGH
7.	7	248	Input	HIGH
8.	8	249	Input	HIGH
9.	9	250	Input	HIGH
10.	10	251	Input	HIGH
*For example GPIO-3 on J9 header will be referenced as gpio-244 in kernel				

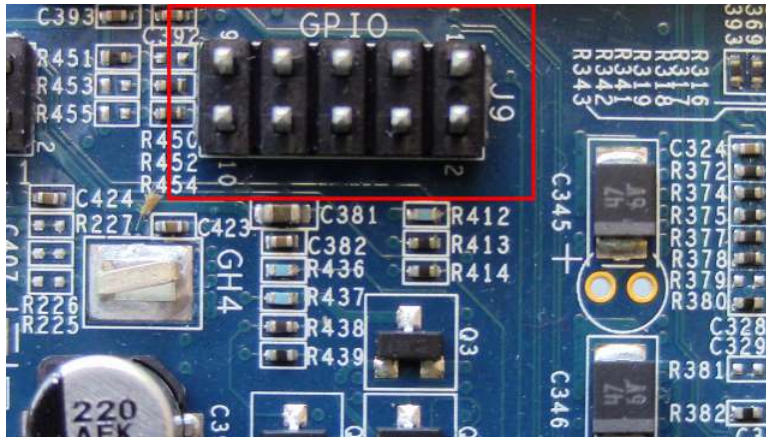


Figure-35: *GPIO on the MinnowBoard*

As you can see from Figure -35, there are a total of ten pins on the J9 header out of which 8 pins can be used as GPIO. The remaining two pins are GND and 3.3V as mentioned in the table.

The GPIO(s) are accessible via the user space in Linux at the location below on the filesystem:

```
/sys/class/gpio
```

There is one directory per GPIO, named as shown below(as an example only two GPIO(s) are shown here):

```
/sys/class/gpio/gpio245
/sys/class/gpio/gpio246
```

Inside each one of those directories, there are files named "direction" and "value" as shown in Figure - 36. The former is for configuring the mode of GPIO as input("in") or output("out") while the latter is for the value('1' for HIGH and '0' for LOW) if used in output mode.

```
root@minnow:/sys/class/gpio/gpio244# ls
active_low device direction edge power subsystem uevent value
```

Figure-36: GPIO attributes

3.2 Selecting the correct LED

- To make sure that you do not damage the GPIO pins on the Minnow-Board, please use a LED whose rating should not exceed 3.3V/10mA:

<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 link below to select an appropriate current limiting resistor:

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

3.3 Extra Credit

This is an optional section which you can read for further understanding. In a nutshell, we are trying to access the on board GPIO(s) via the userspace¹ in Linux. To be more precise we are trying to use the sysfs² interface. sysfs is a virtual filesystem which enumerates the devices and busses attached to the system(board in our case) into a file system hierarchy that can be accessed from userspace. It's generated by the kernel and always mounted at /sys. The GPIO sysfs interface allows users to manipulate any GPIO from userspace. Since it uses gpiolib, it is able to dynamically utilize all GPIOs on the system. The preloaded Angstrom image on the MinnowBoard already has all the GPIO on J9 header exported to the userspace by default.

¹<http://elinux.org/images/4/4f/02-linux-quick-start.pdf>

²<http://www.makelinux.net/books/lkd2/ch17lev1sec8>

3.4 Setup

Please refer the Figure - 37 below to see the connections:

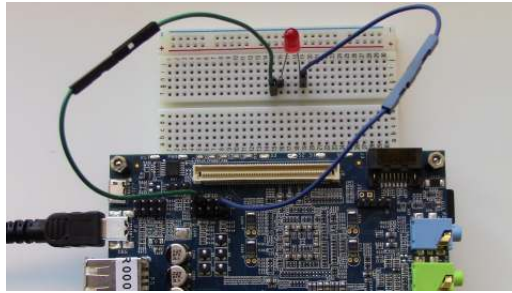


Figure-37: Connection Diagram

Only the pins on the J9 expansion header are used in this case. Connect the anode of the LED to the resistor(I am using a 330 ohms resistor) which in turn is connected to pin 3(GPIO) and connect the cathode of the LED to pin 2(GND) . You can purchase the male-female type hookup/jumper wires used for connections from the link below:

<https://www.sparkfun.com/products/9140>

3.5 Steps

- Now let us toggle that LED!

GPIO - 3 is referenced as 244 in the kernel. Since, it is set to be used in INPUT mode by default, we need to change its mode to OUTPUT. Type the following command in your terminal to accomplish this:

```
echo out > /sys/class/gpio/gpio244/direction
```

Now we are ready to toggle the LED. Type the following commands in your terminal as shown in Figure - 38(First one is for turning ON and latter for OFF):

```
echo 1 > /sys/class/gpio/gpio244/value
echo 0 > /sys/class/gpio/gpio244/value
```

```
root@minnow:/sys/class/gpio/gpio244# echo out > direction
root@minnow:/sys/class/gpio/gpio244# echo 1 > value
root@minnow:/sys/class/gpio/gpio244# echo 0 > value
```

Figure-38: Toggle the LED

3.6 Output

You should get an output as shown below:

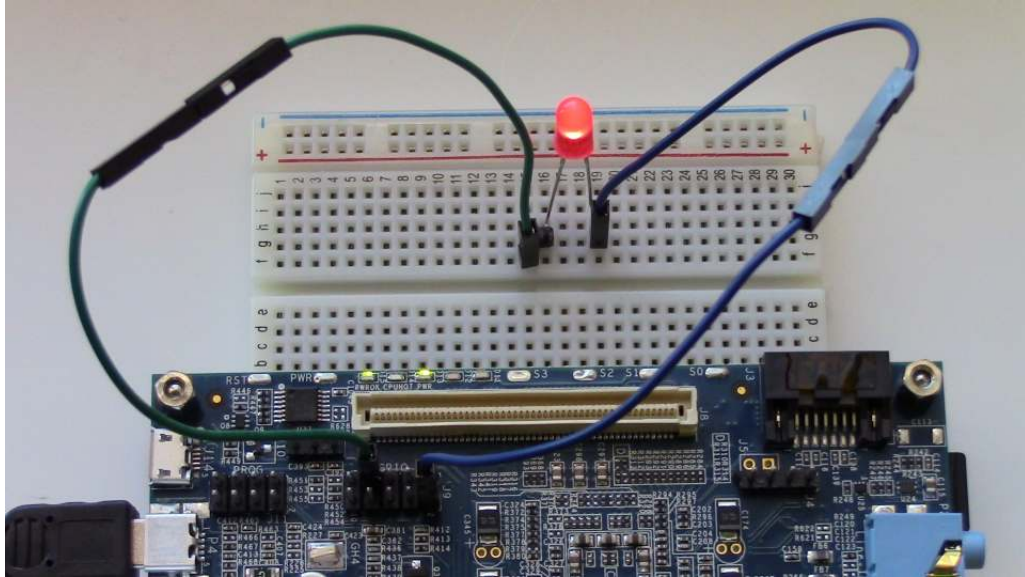


Figure-39: Led ON

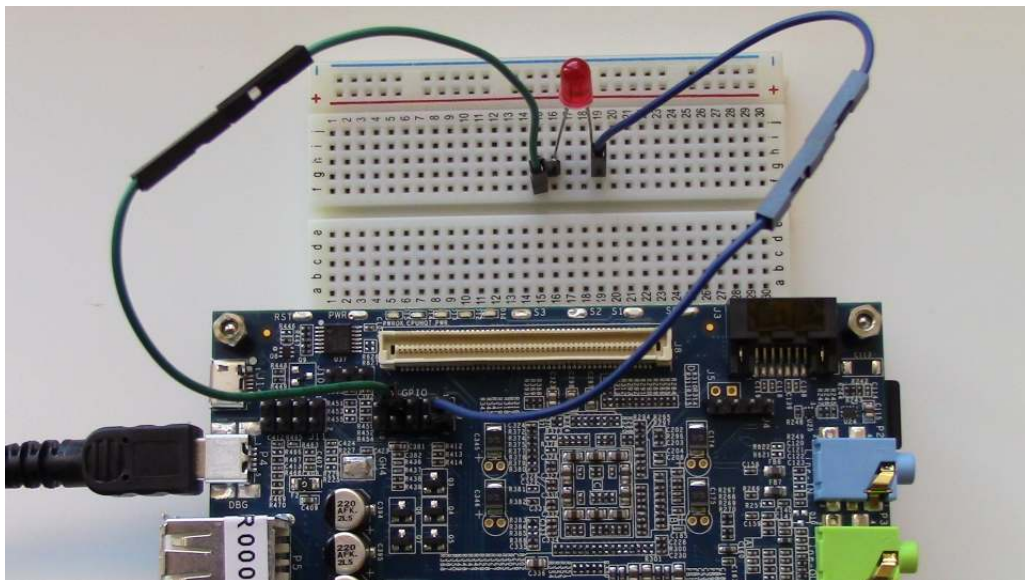


Figure-40: LED OFF

Chapter 4

Physical Computing On The MinnowBoard

4.1 Pullup Resistors

4.1.1 Concept:

So, you want to use the GPIO on the MinnowBoard 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 - 41 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.

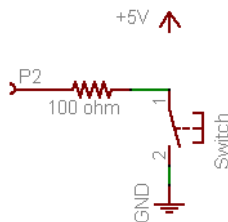


Figure-41: No pullup resistor

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 - 42 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. MinnowBoard has internal pullup resistors.

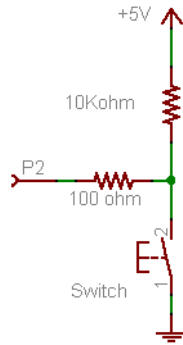


Figure-42: Pullup resistor connected

4.2 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 - 44 below, legs are always connected in groups of two. When the pushbutton is pressed all the 4 legs are connected.



Figure-43: Push Button
ton

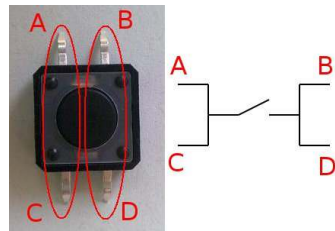


Figure-44: Working of push but-

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

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

4.3 Physical Computing

Physical computing, in the broadest sense, means building interactive physical systems by the use of software and hardware that can sense and respond to the outside world. In our case we are trying to take input from the surrounding via the push button and accordingly causing some change in the outside world by turning ON the LED when the push button is pressed.

4.4 Setup

Please refer Figure - 45 below to see the connections:

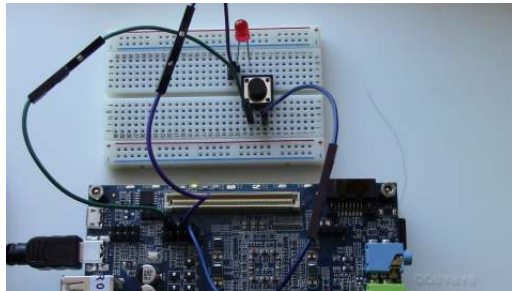


Figure-45: Connection Diagram

Only the pins on the J9 expansion header are used in this case. Connect Pin-2(GND) to leg-1 of the push button. Then connect Pin-7(GPIO) to leg-2(which is not connected to the previous leg-1) of the push button. After that connect the cathode of the LED to leg-4(which is connected by default to leg-4) of the push button. Finally, connect the anode of the LED to a 330 ohm resistor and the other end of the resistor to Pin-3(GPIO). You can purchase the male-female type hookup/jumper wires used for connections from the link below:

<https://www.sparkfun.com/products/9140>

4.5 Script

```
#!/bin/bash
#Use Pin-3 as output pin
echo out > /sys/class/gpio/gpio244/direction
#Use Pin-7 as input pin
echo in > /sys/class/gpio/gpio248/direction
# Read forever
while :
do
# Read value of Pin-7
THIS_VALUE=$(cat /sys/class/gpio/gpio248/value)
if [ "$THIS_VALUE" = "0" ]
then
echo 1 > /sys/class/gpio/gpio244/value
else
echo 0 > /sys/class/gpio/gpio244/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 microSD card before you boot the MinnowBoard. Or you can use a text editor like 'vi' on the MinnowBoard 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 'physicalcomputing.sh'

4.6 Steps

- So, now run the script created earlier by typing the following command in the terminal as shown in Figure - 46:

```
$ ./physicalcomputing.sh
```

A terminal window with a dark background and light-colored text. The prompt is 'root@minnow:~/Desktop#'. The first command entered is 'ls', followed by a space and the filename 'physicalcomputing.sh'. The second command entered is './physicalcomputing.sh', followed by a space and a blue cursor block.

```
root@minnow:~/Desktop# ls  
physicalcomputing.sh  
root@minnow:~/Desktop# ./physicalcomputing.sh
```

Figure-46: Running the script

Please see the next page for the output.

4.7 Output

You should get an output as shown below:

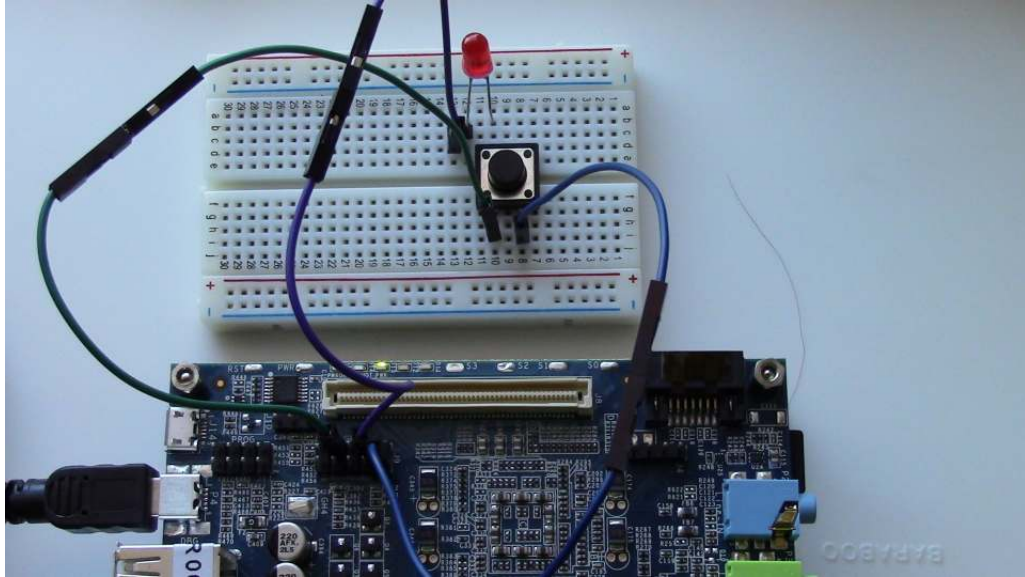


Figure-47: Led OFF

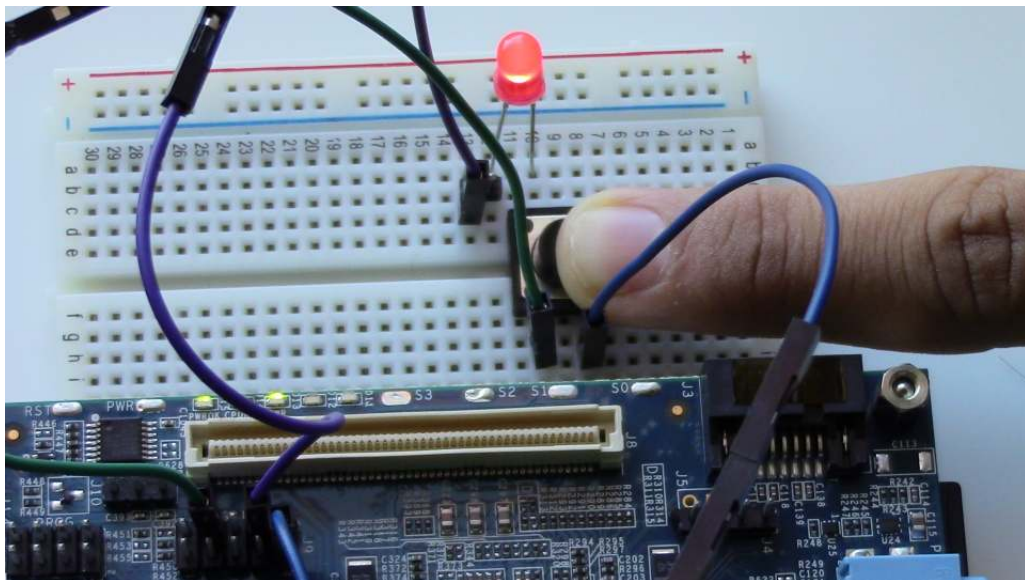


Figure-48: LED ON

4.8 Resources

Please refer the websites below for more videos, tutorials, and projects you can do with your MinnowBoard:

1. <http://www.minnowboard.org/>
2. <http://www.elinux.org/Minnowboard>