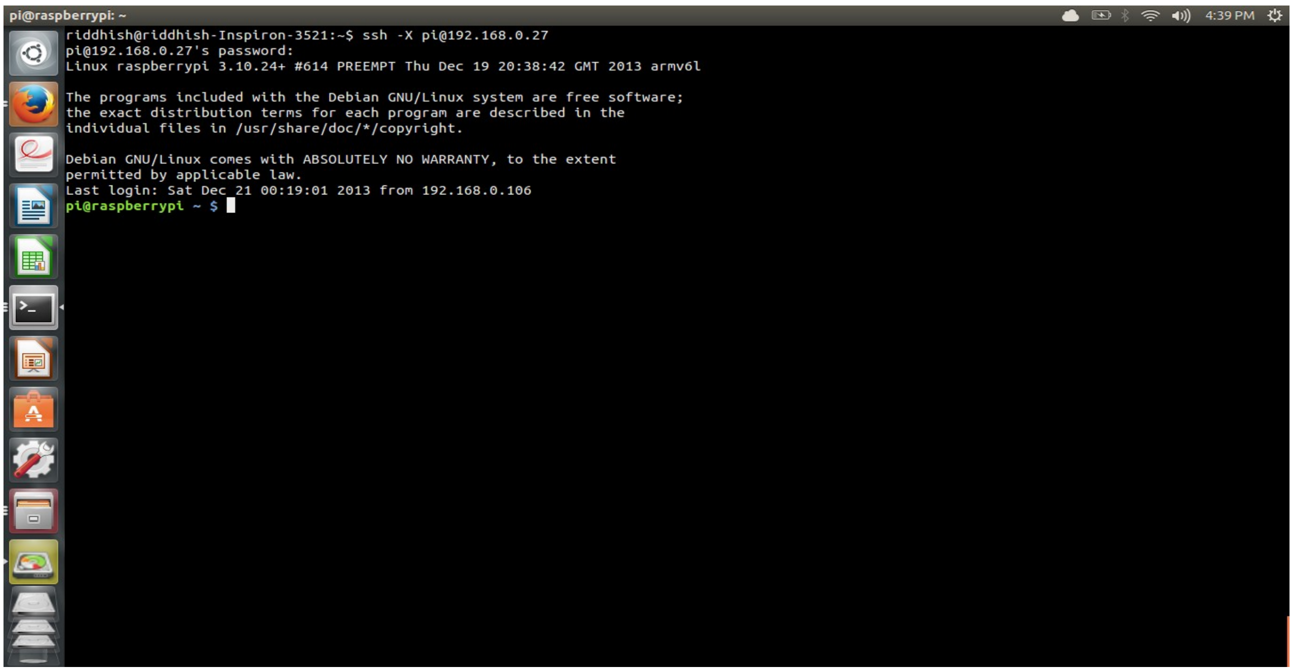
Setting up wiringPi and coding our first R-pi program

Hi again, I hope you all have followed the instructions I provided for getting your R-pi up and running and you have a terminal window like this.

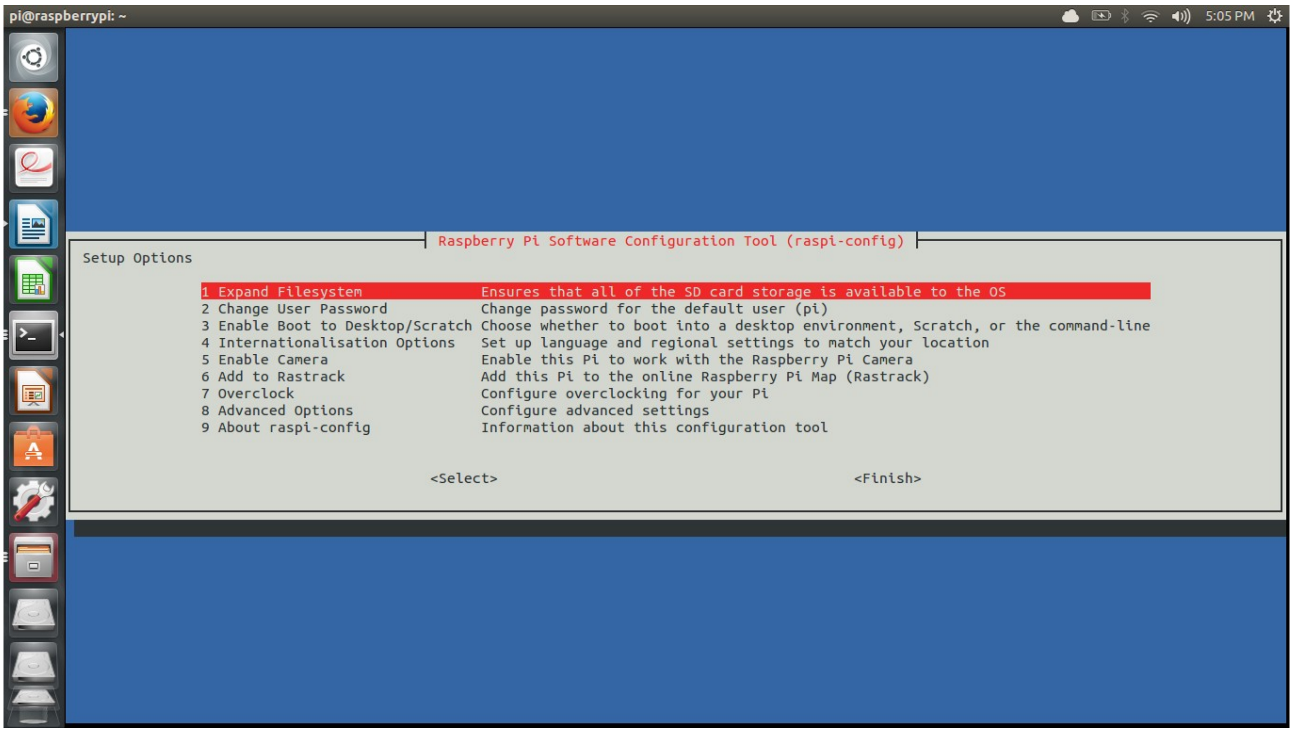


```
riddhish@riddhish-Inspiron-3521:~$ ssh -X pi@192.168.0.27
pi@192.168.0.27's password:
Linux raspberrypi 3.10.24+ #614 PREEMPT Thu Dec 19 20:38:42 GMT 2013 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Dec 21 00:19:01 2013 from 192.168.0.106
pi@raspberrypi ~ $
```

Now, first thing you need to do is config the r-pi, type sudo raspi-config in the terminal above, you will get a window like.



```
                    Raspberry Pi Software Configuration Tool (raspi-config)
Setup Options

         1 Expand Filesystem                Ensures that all of the SD card storage is available to the OS
         2 Change User Password             Change password for the default user (pi)
         3 Enable Boot to Desktop/Scratch   Choose whether to boot into a desktop environment, Scratch, or the command-line
         4 Internationalisation Options     Set up language and regional settings to match your location
         5 Enable Camera                    Enable this Pi to work with the Raspberry Pi Camera
         6 Add to Rastrack                  Add this Pi to the online Raspberry Pi Map (Rastrack)
         7 Overclock                        Configure overclocking for your Pi
         8 Advanced Options                 Configure advanced settings
         9 About raspi-config               Information about this configuration tool

                         <Select>                                   <Finish>
```

Click on Expand filesystem and let it finish the job. You can do all sort of stuff like change your password for starters.

Now lets get to coding, shall we?

What you need to understand is, R-pi is an embedded Linux platform which can be programmed to give some physical output, but before you do that you need to have a library for coding in C and can easily access the firmware file discripters as your GPIO/ADC/USART etc pins. For this we are going to use wiringPi.

Click on this URL: (it should open in a new page)

https://git.drogon.net/?p=wiringPi;a=summary

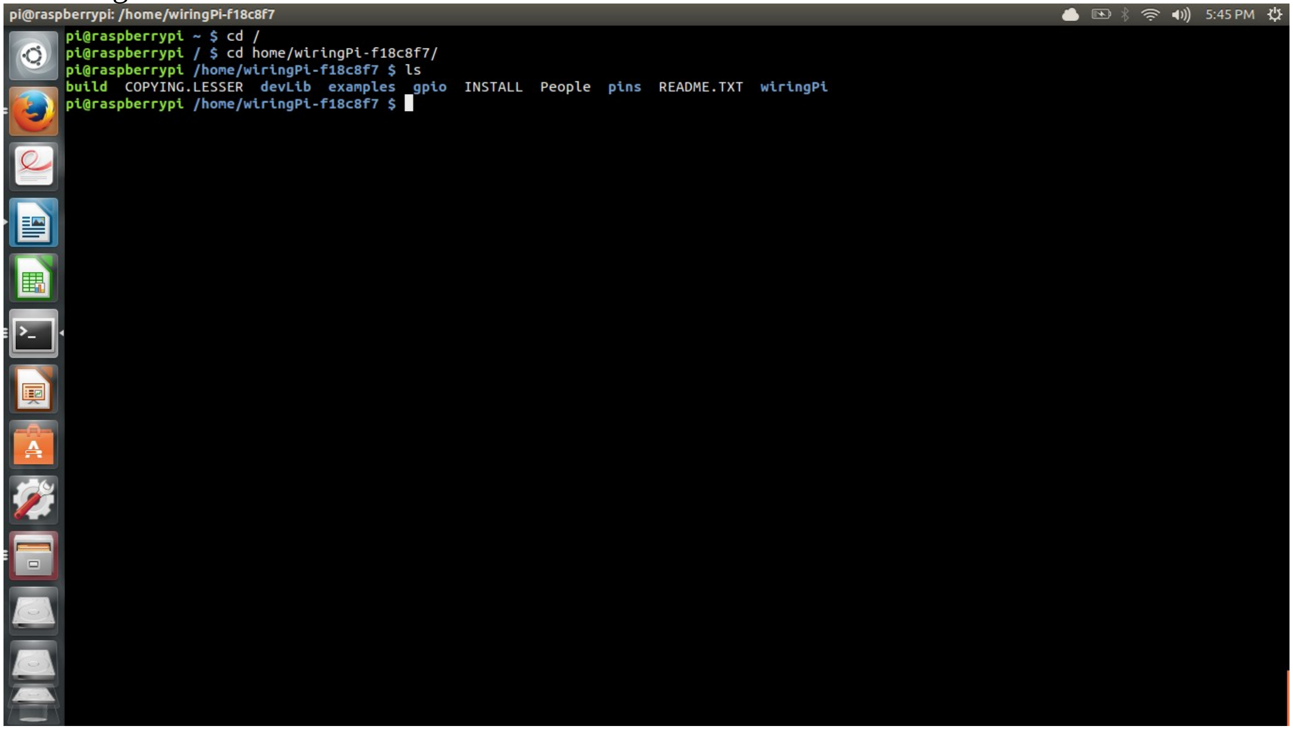
Then look for the link marked **snapshot** at the right-hand side. You want to click on the top one.

This will download a tar.gz file with a name *like* wiringPi-98bcb20.tar.gz. Note that the numbers and letters after **wiringPi** (98bcb20 in this case) will probably be different – they're a unique identifier for each release.

Extract this and copy the folder into the home folder of your Raspbian loaded SD card, you may need to use sudo nautilus.

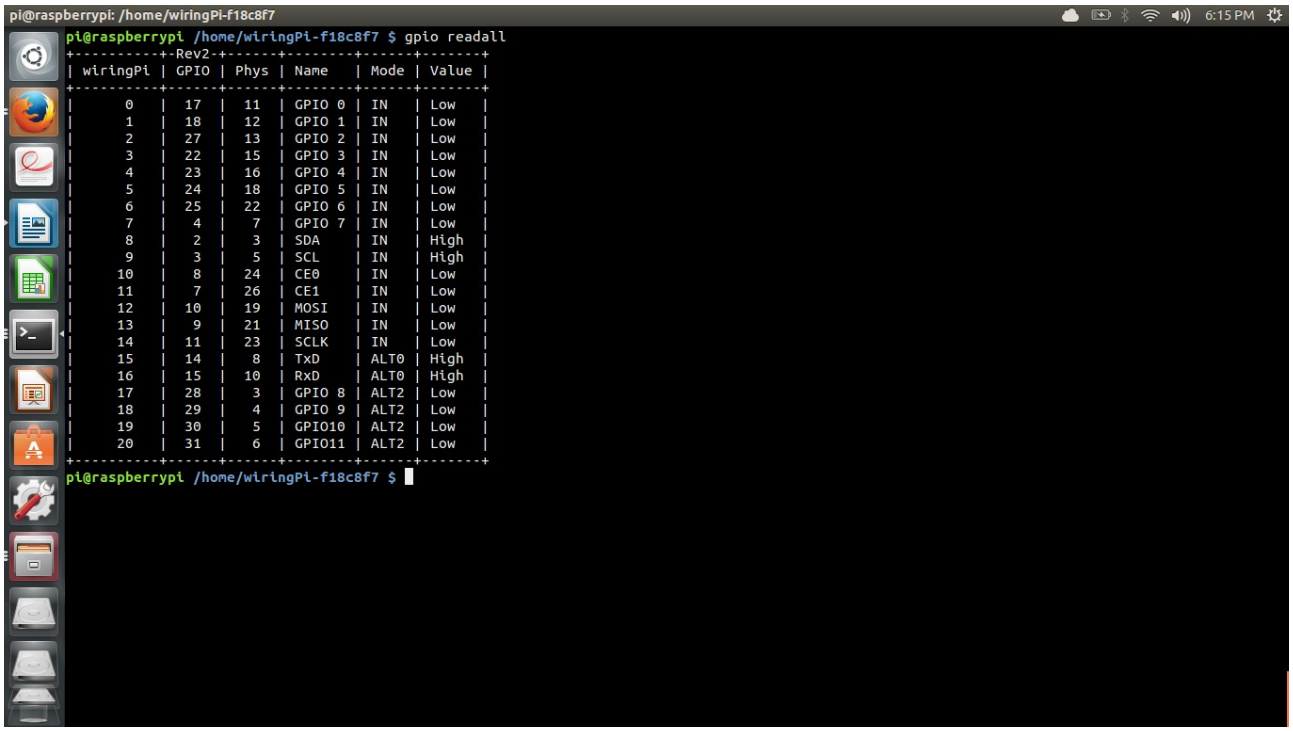
Reinsert your SD card and start up the r-pi.

Navigate to home and open up the wiringPi folder you had copied to it. You will have to execute the following commands to reach there.



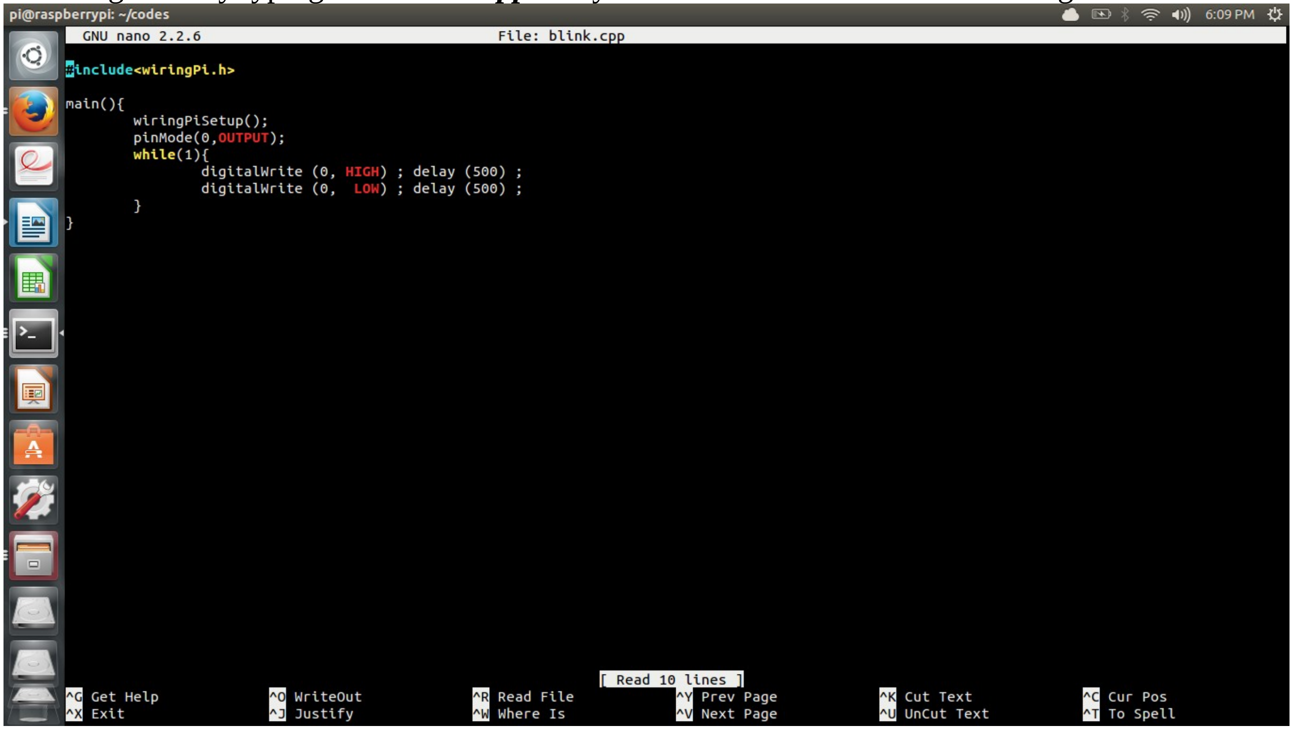
```
pi@raspberrypi ~ $ cd /
pi@raspberrypi / $ cd home/wiringPi-f18c8f7/
pi@raspberrypi /home/wiringPi-f18c8f7 $ ls
build  COPYING.LESSER  devLib  examples  gpio  INSTALL  People  pins  README.TXT  wiringPi
pi@raspberrypi /home/wiringPi-f18c8f7 $
```

Now you see an executable file name build. Execute it with root permissions by writing ***sudo ./build.*** This will take couple of minutes to install.
Type gpio readall and you will see the pin map of R-pi like this,



```
pi@raspberrypi /home/wiringPi-f18c8f7 $ gpio readall
+----------+-Rev2-+------+--------+------+-------+
| wiringPi | GPIO | Phys | Name   | Mode | Value |
+----------+------+------+--------+------+-------+
|      0   |  17  |  11  | GPIO 0 | IN   | Low   |
|      1   |  18  |  12  | GPIO 1 | IN   | Low   |
|      2   |  27  |  13  | GPIO 2 | IN   | Low   |
|      3   |  22  |  15  | GPIO 3 | IN   | Low   |
|      4   |  23  |  16  | GPIO 4 | IN   | Low   |
|      5   |  24  |  18  | GPIO 5 | IN   | Low   |
|      6   |  25  |  22  | GPIO 6 | IN   | Low   |
|      7   |   4  |   7  | GPIO 7 | IN   | Low   |
|      8   |   2  |   3  | SDA    | IN   | High  |
|      9   |   3  |   5  | SCL    | IN   | High  |
|     10   |   8  |  24  | CE0    | IN   | Low   |
|     11   |   7  |  26  | CE1    | IN   | Low   |
|     12   |  10  |  19  | MOSI   | IN   | Low   |
|     13   |   9  |  21  | MISO   | IN   | Low   |
|     14   |  11  |  23  | SCLK   | IN   | Low   |
|     15   |  14  |   8  | TxD    | ALT0 | High  |
|     16   |  15  |  10  | RxD    | ALT0 | High  |
|     17   |  28  |   3  | GPIO 8 | ALT2 | Low   |
|     18   |  29  |   4  | GPIO 9 | ALT2 | Low   |
|     19   |  30  |   5  | GPIO10 | ALT2 | Low   |
|     20   |  31  |   6  | GPIO11 | ALT2 | Low   |
+----------+------+------+--------+------+-------+
pi@raspberrypi /home/wiringPi-f18c8f7 $
```

Now we can code. Go navigate to the base and type ***mkdir codes***. Go in codes and open up a cpp file using nano by typing ***nano blink.cpp*** and you can code in it. It looks something like this.



```
  GNU nano 2.2.6                        File: blink.cpp

#include<wiringPi.h>

main(){
        wiringPiSetup();
        pinMode(0,OUTPUT);
        while(1){
                digitalWrite (0, HIGH) ; delay (500) ;
                digitalWrite (0,  LOW) ; delay (500) ;
        }
}

                                   [ Read 10 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
```

You will quickly learn to use nano. :P. Now, that you have written your first code, lets compile it. Press ctrl+o and enter to save the code and the ctrl+x to exit nano. The code itself is pretty easy to understand especially if you are familiar with the Arduino. Now let's compile it by typing.
gcc -o blink blink.cpp -lwiringPi
and execute the code by ***sudo ./blink***

And you just wrote your first code in r-pi, check the blink output in r-pi by connecting the proper LED to the pin in the code. Go browse the reference section of http://wiringpi.com/, it will definitely help

Author : Riddhish Bhalodia   (sophomore EE at IITB)