Abstract

ARM board provides high speed, better accuracy, good flexibility and low cost solution for development of embedded system. It also supports memory management, process management and peripheral management when ported with OS. When ported with OS it can be used to develop a complex embedded machine. Linux open source distributions provide a good solution for young developers to download, customize, compile and burn Linux distribution OS to their core for free. In this project we have ported OS on ARM board and have designed a data acquisition system.

We have selected Raspberry Pi as our ARM board and ported Debian OS into Raspberry Pi and. We also connected Arduino UNO with Raspberry Pi board and using a serial communication we made an application which continuously monitors the temperature with LM-35 sensor and display this data on Linux terminal and we also plotted that data also.

This is our initial step towards the world of OS and kernel related application. Many further improvements can applied on it like adding more than one sensor , plotting the graph in real time , making a GUI for the entire application related task and so on…

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

Introduction

1.1 Overview :

ARM is 32 bit RISC architecture. It supports a nativefull-fledged 32-bit instruction set. ARM also specifiestwo other instruction sets: a 16-bit compressedRISC set called Thumb, and an 8-bit instruction set forJava byte-codes called Jazelle. The ARM architecture has 37 register. These registers are general purpose registers including a program counter and 6 are status registers. Some of these registers are banked and are hidden except when executing in specific processor modes. These registers such as the stack pointer are automatically switched when entering a different processor mode. This design allows fast processing of interrupts as the handler code does not need to manually switch to a new stack. The ARM processors in both the OMAP and the Integrator support seven modes of operation. All modes except ARM USR have privileges to perform operations that control the MMU and interrupts.

1.2 ARM Processor has following family :

- ARM7
- ARM9
- ARM10
- ARM11
- CORTEX
- XSCALE (from Intel)

1.3 Features of ARM processor :

- RISC architecture
- Low power consumption
- Large register file
- Most instructions are executed in single cycle only.

- ARM has hardware support for JAVA. JAVA code is directly processed by hardware. So power consumption is very much reduced. Jezelle is the name of technology which is responsible for that.
- ARM also has 16 bit instruction core called Thumb. This help to write low density code.
- Advance ARM core also equipped with Floating point unit called as VFP (Vector Floating Point).

1.4 Popular ARM Boards available in market are :

- MK802
- RaspberryPi, Model A&B
- Beagle Board
- Beagle Bone
- Panda Board
- ODROID
- Hackberry

1.5 MK802:

It is a PC-on-a-stick produced by Rikomagic, a Chinese company using AllWinner A1X SoC, based on an ARM architecture, composed of an ARM V7 based Cortex-A8 1 GHz processor, a Mali-400 MP GPU, Wi-Fi 802.11 b/g/n, and a VPU CedarX capable of displaying 1080p video.The thumb sized MK802, which was first brought into market in May 2012, can turn a display with a HDMI or DVI-D port into an Android computer.

1.6 RaspberryPi:

It is a credit-card-sized single-board computer developed in the UK by the RaspberryPi Foundation with the intention of promoting the teaching of basic computer science in schoolsTheRaspberryPi has a Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S 700 MHz processor (The firmware includes a number of "Turbo" modes so that the user can attempt overclocking, up-to 1 GHz, without affecting the warranty),VideoCore IV GPU,and originally shipped with 256 megabytes of RAM, later upgraded to 512MB.It does not include a built-in hard disk or solid-state drive, but uses an SD card for booting and long-term storage.The Foundation's goal is to offer two versions, priced at US$25 and US$35.

1.7 Beagle Board:

It is a low-power open-source hardware single-board computer produced by Texas Instruments in association with Digi-Key. The Beagle Board was also designed with open source software development in mind, and as a way of demonstrating the Texas Instrument's OMAP3530 system-on-a-chip. The board was developed by a small team of engineers as an educational board that could be used in colleges around the world to teach open source hardware and open source software capabilities

1.8 Panda Board:

It is a low-power, low-cost single-board computer development platform based on the Texas Instruments OMAP4430 system on a chip (SoC). The Panda Board ES is a newer version based on the OMAP4460 SoC, with the CPU and GPU running at higher clock rates.

1.9 ODROID:

It is the Ultra-Compact and Ultra-low-cost platform with 1.4 GHz Quad-core CPU and 1Gbyte RAM. This board is the world’s smallest and cost effective US$69 priced Quad-core ARM PC. Board dimension is only 48x52mm (1.9x2.0inch approx.). The new ODROID boards are powered by the Samsung Exynos-4412 and Exynos-4412-Prime chip. Exynos 4412 Prime has 4 cores of 1.7 GHz ARM Cortex-A9 processor and 2Gbyte of Low-Power DDR RAM. New ODROID boards can run the Android and various Linux operating systems.

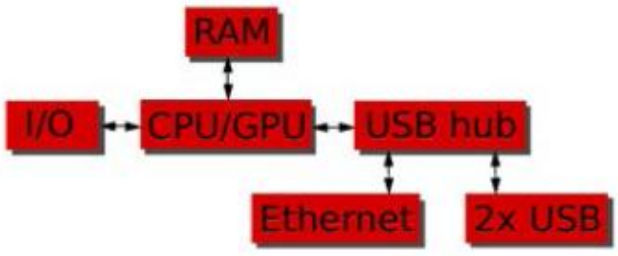
Introduction to Raspberry Pi model B

2.1 Overview :

We are working on RaspberryPi model B it is a credit-card-sized single-board computer developed in the UK by the RaspberryPi Foundation with the intention of promoting the teaching of basic computer science in schools TheRaspberryPi has a Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S 700 MHz processor (The firmware includes a number of "Turbo" modes so that the user can attempt overclocking, up-to 1 GHz, without affecting the warranty),VideoCore IV GPU,and originally shipped with 256 megabytes of RAM, later upgraded to 512MB.It does not include a built-in hard disk or solid-state drive, but uses an SD card for booting and long-term storage.

2.1) RaspberryPi model B

$$\begin{tikzcd} & RAM \arrow[d, leftrightarrow] & & \\ I/O \arrow[r, leftrightarrow] & CPU/GPU \arrow[r, leftrightarrow] & USB\ hub \arrow[d, leftrightarrow] \arrow[dr, leftrightarrow] & \\ & & Ethernet & 2x\ USB \end{tikzcd}$$



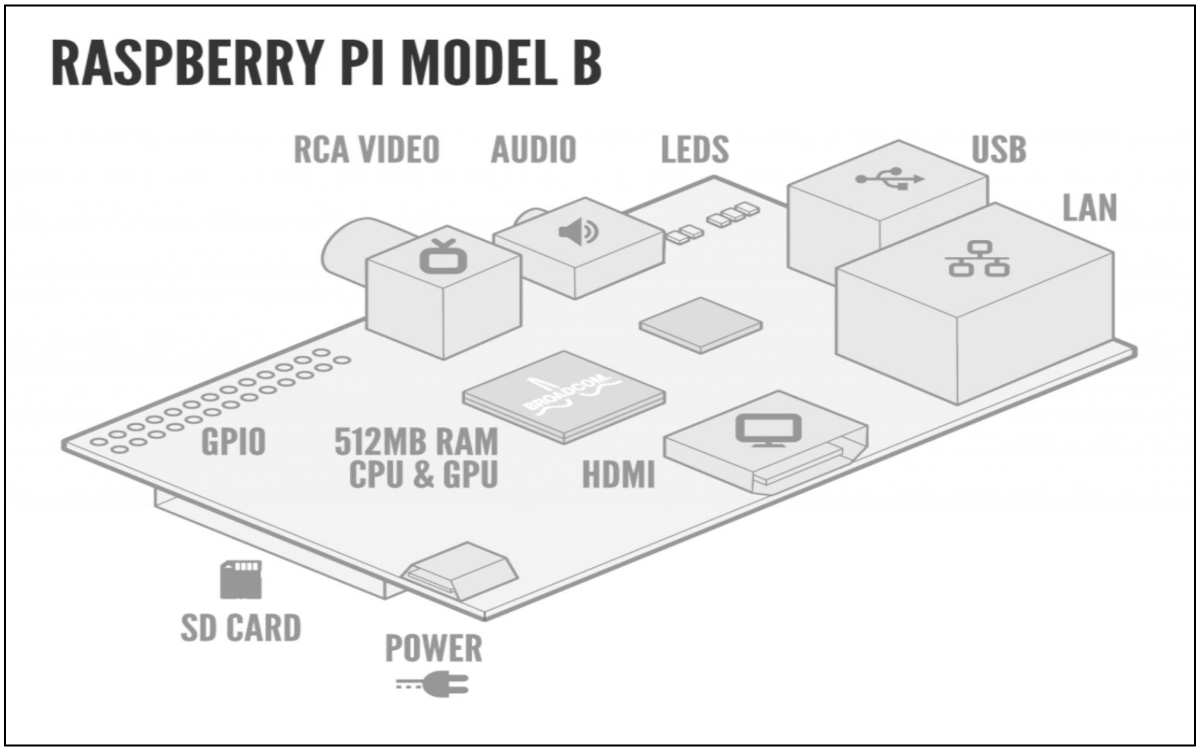
2.2) Block diagram of RaspberryPi model b

The table below gives the detailed information regarding RaspberryPi model B

Product details	Raspberry –Pi model B
Price	US$35.00
SOC	
SOC Type	Broadcom BCM2835
Core Type	ARM1176JZF-S
No. Of Cores	1
GPU	VideoCore IV
CPU Clock	700 MHz
On Board Memory	
RAM	512 MB
Flash	0 MB
Wired Connectivity	
USB Ports	**Yes** 2
Ethernet	**Yes** 10/100M
SATA Ports	**No**
HDMI	**Yes**
Analog Video Out	**Yes** Composite
Analog Audio Out	**Yes**

Analog Audio In	**Yes**
SPI	**Yes**
I2C	**Yes**
GPIO	**Yes**
LCD Panel	**Yes** DSI
Camera	**Yes** DSI
SD/MMC	**Yes** SD Cage
Serial	**Yes** Through Expansion Connector, needs level shifting
Wireless Connectivity (On-Board)	
Wi-Fi	**No**
Bluetooth®	**No**
Dimensions	
Height	3.37 in (85.6 mm)
Width	2.12 in (53.98 mm)
Depth	
Weight	1.58 oz. (45 g)

Table 2.1) Features of raspberry pi model B



2.3) Hardware model of Raspberry Pi

Available OS for porting

3.1 LinuxDistribution :

Linux provide open source distributions for young developers to download, customize, compile and burn OS to their core for free. VariousLinuxdistributions are available for porting on ARM board in market such as

- The RaspberryPi Fedora Remix
- Debian
- Arch Linux ARM
- Raspbian
- Moebius
- OpenELEC
- RISC OS
- PiBangLinux

3.2 The RaspberryPi Fedora Remix:

It is a Linux software distribution for the RaspberryPi computer. It contains software packages from the Fedora Project (specifically, the Fedora ARM secondary architecture project), packages which have been specifically written for or modified for the RaspberryPi, and proprietary software provided by the RaspberryPi Foundation.

3.3 Debian:

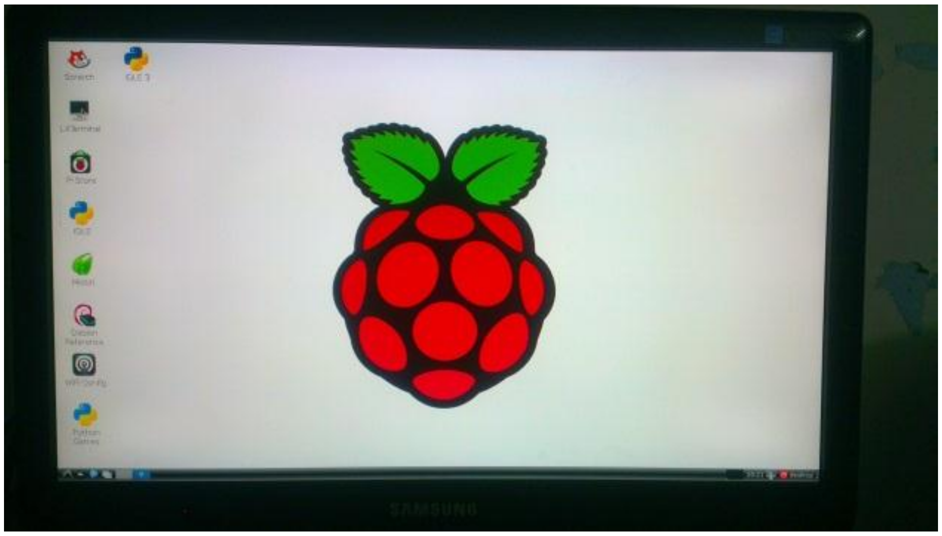
It was the default distribution on the Alpha boards. Boot time depends on width & speed of SD-card. Alpha board boot into Debian prompt (no GUI) was timed taking about 34 seconds. The Debian distro for RaspberryPi is the Cambridge reference file system, which is a fully functional Debian Squeeze installation containing LXDE (desktop) and Midori (browser); development tools; and sample code for accessing the multimedia functionality on the device.

3.4 Arch Linux ARM:

It is based on Arch Linux, which aims for simplicity and full control to the end user. It provides a lightweight base structure that allows you to shape the system to your needs. For this reason, the Arch Linux ARM image for the RaspberryPi does not come with a graphical user interface, though you can easily install one yourself. Please note that the Arch distribution may not be suitable for beginners. Arch Linux ARM is on a rolling-release cycle that can be updated daily through small packages instead of huge updates every few months.

3.5 Raspbian:

RaspberryPi + Debian = Raspbian. A project to create a hard float port of Debian Wheezy (7.x) armhf for the RaspberryPi. The intent of Raspbian is to bring to the RaspberryPi user 10,000s of pre-built Debian packages specifically tuned for optimal performance on the RaspberryPi hardware. The project is still in its early phases, but the major push to rebuild nearly all Debian packages for the RaspberryPi is expected to be completed by early June, 2012 (only several hundred packages remain as of June 1st). After that, efforts will focus on making Raspbian the easiest to use, most stable and best performing Linux distribution available for the RaspberryPi.



3.1) Raspbian ported on raspberry pi

3.6 Moebius:

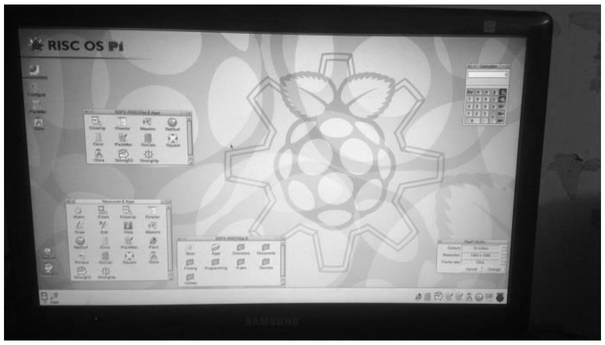
A very compact ARM HF debian based distribution, it fits in a 1 GB SD card, has auto resizing features to better adapt to your SD card size and uses Raspbian huge repositories for installing everything you need. A wise configuration and a small memory footprint are ideal for a headless machine or for interacting with real word I/O devices.

3.7 OpenELEC:

OpenELEC is an embedded operating system built specifically to run XBMC, the open source entertainment media hub. The idea behind OpenELEC is to allow people to use their Home Theatre PC (HTPC) like any other device you might have attached to your TV, like a DVD player or Sky box. Instead of having to manage a full operating system, configure it and install the packages required to turn it into a hybrid media center, OpenELEC is designed to be simple to install, manage and use, making it more like running a set-top box than a full-blown computer.

3.8 RISC OS:

RISC OS is a fast and lightweight computer operating system designed in Cambridge, England by Acorn. First released in 1987, its origins can be traced back to the original team that developed the ARM microprocessor. RISC OS includes BBC BASIC which was primarily conceived to teach programming skills as part of the BBC computer literacy project. RISC OS Open (ROOL) has released the sources. Community members have ported the OS to the Beagle Board and similar hardware



3.2) RISC OS ported on raspberry pi

3.9 PiBangLinux:

PiBangLinux is a Raspbian based distribution. PiBang is insPired by Crunch bangLinux, an i686 and x86_68 Debian based distribution. It comes preconfigured with many helpful scripts and Pipe menus as well as a fork of RasPi-config with increases functions such as support for changing the user and hostname. PiBang is also one of the heavier RaspberryPi distributions boasting a complete package set with favorites such as Abiword, OMXPlayer, GIMP, and VLC all pre-installed.

3.10 Plan 9:

Plan 9 is a distributed operating system originally designed and implemented by Ken Thompson, Rob Pike, Dave Presotto, and Phil Winter bottom @ Bell Labs. It is a lean operating system that has been ported to super computers such as IBM's Blue Gene down to tiny boards such the RaspberryPi.

Introduction to Linux

4.1 What is Linux ?

Linux (also known as *GNU/Linux*) is a computer operating system, like Microsoft Windows or Apple Mac OS. Unlike those two, however, Linux is built with a collaborative development model. The operating system and most of its software are created by volunteers and employees of companies, governments and organisations from all over the world.

The operating system is free to use and everyone has the freedom to contribute to its development. This co-operative development model means that everyone can benefit. Because of this, we like to call it Free Software, or Socially Responsible Software. Closely related is the concept of Open Source Software. Together, Free and Open Source Software is collectively abbreviated as FOSS. This contrasts with the proprietary (or closed source) development model used by some software companies today.

Many of the principles behind FOSS are derived from the axiom of standing on the shoulders of giants, most famously used by Isaac Newton, which has guided scientific and industrial development for hundreds of years. Transparency of the code and development process means that it can be participated in and audited at all levels. Software is just another form of information, and people have the right to have full control over that information. In the same way that you are free to share cooking recipes with your neighbour, you should also have the freedom to share and change software.

Linux has many other benefits, including speed, security and stability. It is renowned for its ability to run well on more modest hardware. Linux comes from the venerable UNIX family of operating systems, and so has been built from the ground-up with Internet-style networking and security in mind. Hence, viruses, worms, spyware and adware are basically a non-issue on Linux.

Linux commands

5.1 Overview :

There are many different commands you can enter to configure and interact with Linux. These commands can be entered at the shell prompt, or command (line) prompt. If you using Linux with a GUI, look for the shell program icon. If you have no GUI installed you should be looking at the prompt, and if you log
in remotely you will be at the prompt. There are two types of commands, shell commands and Linux commands.

5.2 Shell Commands :

Shell commands are part of the shell program. There are several different shells (C shell, bash shell, bourne shell etc) to choose from, and each will have a variation of the shell commands built in. The commands vary between
shells, but each shell is the same across different Linux distros.

5.3 Linux Commands :

Linux commands are not part of the shell. Each one is a seperate executable program, probably written in the C

5.4 programming language :

These executables are stored in various directories set up for binary files, such as /bin and /usr/bin. The locationof these directories can be defined with the $PATH variable so the shells know where to find them. Thesecommands vary between different Linux distrubutions, and remain the same whichever shell you are using.

5.5 The Command Prompt :

What your prompt looks like will depend on the shell you use (bash, borne, cshetc), and the settings you have for that shell. I use a bash shell, and have the prompt set up to look like this......

```
[root@cirrus home]#
```

This shows me I am logged in as root, to the computer called cirrus, and I am in a folder called home.

TIP: To see the full path of the folder 'home', use the pwdcommand.

Your prompt is bound to look different, so for the purpose of this section I will just use the > to represent the
prompt.>
Some linux commands are very simple and need nothing more than the command itself.
For example the command **ls**(list) will simply list the names of the directories and files in the current directory...

```
>ls
files/
images/
index.txt
>
```

In this instance ls shows there are two directories, called files and images, and a text file called index. However you can often add options or parameters, which are usually added to the command after a '-'.

Adding the option -l will give you a long listing which includes the permissions, ownership, size, date/time, and name of the files and directories...

```
>ls -l
drwx------ 2 bob bob 4096 Aug 22 10:31 files/
drwx------ 3 bob bob 4096 Mar 19 11:17 images/
-rwx------ 1 bob bob 284 Mar 18 10:23 index.txt
>
```

5.6 List of important commands :

A :

apt-get : Search for and install software packages (Debian/Ubuntu)

aptitude : Search for and install software packages (Debian/Ubuntu)

awk : Find and Replace text, database sort/validate/index

B :

bash :GNU Bourne-Again SHell

break :Exit from a loop •

bzip2 :Compress or decompress named file(s)

C :

cal:Display a calendar

case :Conditionally perform a command

cd :Change Directory

cfdisk:Partition table manipulator for Linux

chmod:Change access permissions

chown:Change file owner and group

chroot:Run a command with a different root directory

chkconfig:System services (runlevel)

cksum:Print CRC checksum and byte counts

clear :Clear terminal screen

cmp:Compare two files

comm:Compare two sorted files line by line

command :Run a command - ignoring shell functions •

D :

date :Display or change the date & time

dc :Desk Calculator

declare :Declare variables and give them attributes •

df:Display free disk space

diff :Display the differences between two files

diff3 :Show differences among three files

dircolors:Colour setup for `ls'

du :Estimate file space usage

E :

echo :Display message on screen •

eject :Eject removable media

enable :Enable and disable builtin shell commands •

env:Environment variables

ethtool:Ethernet card settings

exec :Execute a command

exit :Exit the shell

F :

false :Do nothing, unsuccessfully

file :Determine file type

find :Search for files that meet a desired criteria format :Format disks or tapes

free :Display memory usage

ftp :File Transfer Protocol

function :Define Function Macros

G :

gawk :Find and Replace text within file(s)

groupdel:Delete a group

groupmod:Modify a group

gzip:Compress or decompress named file(s)

H :

head :Output the first part of file(s)

history :Command History

hostname :Print or set system name

I :

iconv:Convert the character set of a file

id :Print user and group id's

if :Conditionally perform a command

ifconfig:Configure a network interface

ifdown:Stop a network interface

ifup:Start a network interface up

import :Capture an X server screen and save the image to file

install :Copy files and set attributes

K :

kill :Stop a process from running
killall:Kill processes by name

L :

ln:Create a symbolic link to a file
local :Create variables
locate :Find files
logname:Print current login name
logout :Exit a login shell •
lprint:Print a file
ls:List information about file(s)
lsof:List open files

M :

make :Recompile a group of programs
man :Help manual
mkdir:Create new folder(s)
mkfifo:Make FIFOs (named pipes)
mkisofs:Create an hybrid ISO9660/JOLIET/HFS filesystem
mknod:Make block or character special files
mtr:Network diagnostics (traceroute/ping)

O :

open :Open a file in its default application
op :Operator access

P :

passwd:Modify a user password
paste :Merge lines of files
pathchk:Check file name portability
ping :Test a network connection
pkill:Stop processes from running
printf:Format and print data •
ps:Process status
pwd:Print Working Directory

Q :

quota :Display disk usage and limits

R :

read :Read a line from standard input •
readarray:Read from stdin into an array variable •
readonly:Mark variables/functions as readonly
reboot :Reboot the system
rename :Rename files
rm:Remove files

S :

shutdown :Shutdown or restart linux
sleep :Delay for a specified time

slocate:Find files

sort :Sort text files

source :Run commands from a file `.'

ssh:Secure Shell client (remote login program)

su:Substitute user identity

sudo:Execute a command as another user

sync :Synchronize data on disk with memory

T :

tar :Tape ARchiver

time :Measure Program running time

times :User and system times

true :Do nothing, successfully

tsort:Topological sort

tty:Print filename of terminal on stdin

U :

useradd:Create new user account

userdel:Delete a user account

usermod:Modify user account

users :List users currently logged in

V :

vdir:Verbosely list directory contents (`ls -l -b')

vi :Text Editor

W :

wait :Wait for a process to complete •

watch :Execute/display a program periodically

while :Execute commands

write :Send a message to another user

Introduction to vi commands

6.1 What is vi?

The default editor that comes with the UNIX operating system is called vi (**vi**sual editor). [Alternate editors for UNIX environments include pico and emacs, a product of GNU.]

The UNIX vi editor is a full screen editor and has two modes of operation:

1. *Command mode* commands which cause action to be taken on the file, and
2. *Insert mode* in which entered text is inserted into the file.

In the command mode, every character typed is a command that does something to the text file being edited; a character typed in the command mode may even cause the vi editor to enter the insert mode. In the insert mode, every character typed is added to the text in the file; pressing the <Esc> (*Escape*) key turns off the Insert mode.

While there are a number of vi commands, just a handful of these is usually sufficient for beginning vi users. To assist such users, this Web page contains a sampling of basic vi commands. The most basic and useful commands are marked with an asterisk (* or star) in the tables below. With practice, these commands should become automatic.

NOTE:

Both UNIX and vi are **case-sensitive**. Be sure not to use a capital letter in place of a lowercase letter; the results will not be what you expect.

6.2 List of vi commands :

Entering command mode :

[Esc] : Exit editing mode. Keyboard keys now interpreted as commands.

Moving the cursor :

h : (or left arrow key) move the cursor left.

l : (or right arrow key) move the cursor right.

j : (or down arrow key) move the cursor down.

k : (or up arrow key) move the cursor up.

[Ctrl] f : move the cursor one page **f**orward .

[Ctrl] b : move the cursor one page **b**ackward.

^ : move cursor to the first non-white character in the current line.

$: move the cursor to the end of the current line.

G :**g**o to the last line in the file.

*n*G :**g**o to line number *n*.

[Ctrl] G : display the name of the current file and the cursor position in it.

Entering editing mode :

i :**i**nsert new text before the cursor.

a :**a**ppend new text after the cursor.

o : start to edit a new line after the current one.

O : start to edit a new line before the current one.

Replacing characters, lines and words :

r :**r**eplace the current character (does not enter edit mode).
s : enter edit mode and **s**ubstitute the current character by several ones.
cw : enter edit mode and **c**hange the **w**ord after the cursor.
C : enter edit mode and **c**hange the rest of the line after the cursor.

Copying and pasting :

yy : copy (**y**ank) the current line to the copy/paste buffer.
p :**p**aste the copy/paste buffer after the current line.
P :**P**aste the copy/paste buffer before the current line.

Deleting characters, words and lines :

All deleted characters, words and lines are copied to the copy/paste buffer.

x : delete the character at the cursor location.
dw :**d**elete the current **w**ord.
D :**d**elete the remainder of the line after the cursor.
dd :**d**elete the current line.

Repeating commands :

. repeat the last insertion, replacement or delete command.

Looking for strings :

/*string :*find the first occurrence of *string* after the cursor.

?*string :*find the first occurrence of *string* before the cursor.

n : find the **n**ext occurrence in the last search.

Replacing strings :

Can also be done manually, searching and replacing once, and then using

n : (next occurrence) and . (repeat last edit).

n,*p*s/*str1*/*str2*/g : between line numbers *n* and *p*, **s**ubstitute all (**g**: global) occurrences of *str1* by *str2*.

1,$s/*str1*/*str2*/g : in the whole file ($: last line), **s**ubstitute all occurrences of *str1* by *str2*.

Applying a command several times – Examples :

5j : move the cursor 5 lines down.

30dd :**d**elete 30 lines.

4cw :**c**hange 4 **w**ords from the cursor.

1G :**g**o to the first line in the file.

Misc :

[Ctrl] l : redraw the screen.

J :**j**oin the current line with the next one

Exiting and saving :

ZZ : save current file and exit vi.

:w : **w**rite (save) to the current file.

:w : *file* **w**rite (save) to the *file* file.

:q! : **q**uit vi without saving changes.

Introduction to python

7.1 Overview :

Python is a dynamic, interpreted language. Source code does not declare the types of variables or parameters or methods. This makes the code short and flexible, and you lose the compile-time type checking in the source code. Python tracks the types of all values at runtime and flags code that does not make sense as it runs.

An excellent way to see how Python code works is to run the Python interpreter and type code right into it. If you ever have a question like "what happens if I add an int to a list?" ... just typing it into the Python interpreter is fast way to see what happens. Python code does not declare the types of variables -- just assign to them and go. Python raises a runtime error if the code tries to read from a variable that has not been given a value. Like C++ and Java, Python is case sensitive so "a" and "A" are different variables. The end of a line marks the end of a statement, so unlike C++ and Java, Python does not require a semicolon at the end of each statement. You can include semicolons at the end of Python statements (perhaps just out of habit), but it's not the best style. Comments begin with a '#' and extend to the end of the line.

7.2 Python program :

Python source files use the ".py" extension. With a Python program in file hello.py, the easiest way to run it is with the shell command "python hello.py Alice" -- loading and running the code in hello.py, passing it the command line argument "Alice".

Here's a very simple Python hello.py program (notice that blocks of code are delimited strictly using indentation rather than curly braces -- more on this later!):

```
#!/usr/bin/python
# import modules used here -- sys is a very standard one
```

```
import sys

# Gather our code in a main() function
def main():
    print 'Hello there', sys.argv[1]
    # Command line args are in sys.argv[1], sys.argv[2] ...
    # sys.argv[0] is the script name itself and can be ignored

# Standard boilerplate to call the main() function to begin
# the program.
if __name__ == '__main__':
    main()
```

Running this program from the command line looks like :

```
$ python hello.py Guido

Hello there Guido

$ ./hello.py Alice    # without needing 'python' first (Unix)

Hello there Alice
```

7.3 Modules and imports :

One file of Python code is called a *module*. The file "binky.py" is also known as the module "binky". A module essentially contains variable definitions like, "x = 6" and "deffoo()". Suppose the file "binky.py" contains a "deffoo()". The fully qualified name of that foo function is "binky.foo". In this way, various Python modules can name their functions and variables whatever they want, and the variable names won't conflict -- module1.foo is different from module2.foo.

For example, we have the standard "sys" module that contains some standard system facilities, like the argv list, and exit() function. With the statement "import sys" you can can then access the definitions in the sys module and makes them available by their fully-qualified name, e.g. sys.exit().

```
import sys

  # Now can refer to sys.xxx facilities
  sys.exit(0)
```

There is another import form that looks like this: "from sys import argv, exit". That makes argv and exit() available by their short names; however, we recommend the original form with the fully-qualified names because it's a lot easier to determine where a function or attribute came from.

There are many modules and packages which are bundled with a standard installation of the Python interpreter, so you don't have do anything extra to use them. These are collectively known as the "Python Standard Library." Commonly used modules/packages include:

- sys -- access to exit(), argv, stdin, stdout, ...
- re -- regular expressions
- os -- operating system interface, file system

Arch Linux

8.1 Overview :

Arch Linux is aLinuxopen source distribution. Arch Linux defines simplicity as without unnecessary additions, modifications, or complications, and provides a lightweight UNIX-like base structure that allows an individual user to shape the system according to their own needs. In short: an elegant, minimalist approach. A lightweight base structure built with high programming standards will tend to have lower system resource demands. The base system is devoid of all clutter that may obscure important parts of the system, or make access to them difficult or convoluted. It has a streamlined set of succinctly commented, clean configuration files that are arranged for quick access and editing, with no cumbersome graphical configuration tools to hide possibilities from the user. An Arch Linux system is therefore readily configurable to the very last detail. It has enormous customization potential.it is low cost, highly customizable, light weight operating system

8.2 Porting Arch Linux:

Porting Arch Linux require following simple steps :

1. First, install the package dependencies, git and the cross-comPilationToolchain
2. Create a symlink for the cross comPiler
3. Make a directory for the sources and tools, then clone them with git
4. Generate the .config file from the pre-packaged RaspberryPi one
5. run make menuconfig
6. create a directory for the modules
7. comPile and ‘install’ the loadable modules to the temp directory
8. create a kernel.img in the current directory
9. Plug in the SD card that you wish to install the new kernel on. Delete the existing kernel.img and replace it with the new one, substituting “boot-partition-uuid” with the identifier of the portion as it is mounted in Ubuntu.

8.3 Terminal Step by step code :

Below is the code as per the steps we mentioned above

Step1:

```
Sudo apt-get install git-core gcc-4.6-arm-Linux-gnueabi
```

Step2:

```
sudoln -s /usr/bin/arm-Linux-gnueabi-gcc-4.6 /usr/bin/arm-Linux-gnueabi-gcc
```

Step3:

```
mkdirRaspberryPi

cdRaspberryPi

git clone https://github.com/RaspberryPi/tools.git

git clone https://github.com/RaspberryPi/Linux.git

cdLinux
```

Step4:

```
make ARCH=arm CROSS_COMPILE=/usr/bin/arm-Linux-gnueabi-
bcmrPi_cutdown_defconfig
```

Step5:

```
make ARCH=arm CROSS_COMPILE=/usr/bin/arm-Linux-gnueabi- menuconfig
```

Step6:

```
mkdir ../modules
```

Step7:

```
makemodules_install ARCH=arm CROSS_COMPILE=/usr/bin/arm-Linux-gnueabi- INSTALL_MOD_PATH=../modules/
```

Step8:

```
cd ../tools/mkimage/
./imagetool-uncompressed.py ../../Linux/arch/arm/boot/Image
```

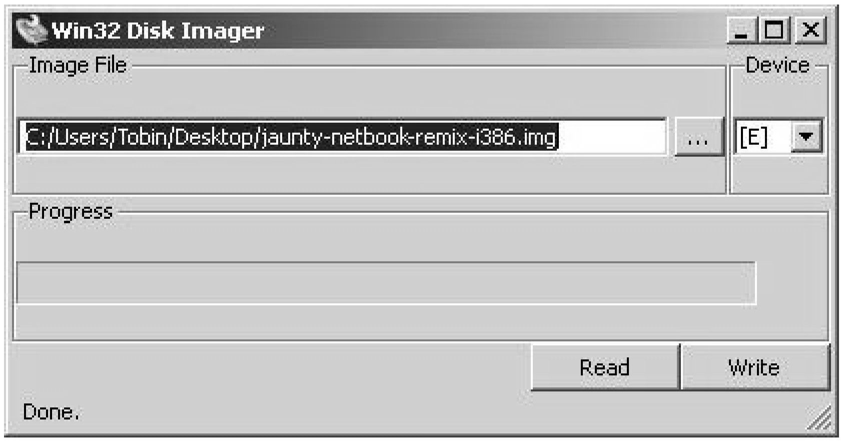
Step9:

```
sudorm /media/boot-partition-uuid/kernel.img
sudo mv kernel.img /media/boot-partition-uuid/
```

Now the RaspberryPi is ready to get started…..

Writing an SD card image

9.1 For Windows :



C:/Users/Tobin/Desktop/jaunty-netbook-remix-i386.img

[E]

Done.

9.1) Graphical Interface for windows

Steps :

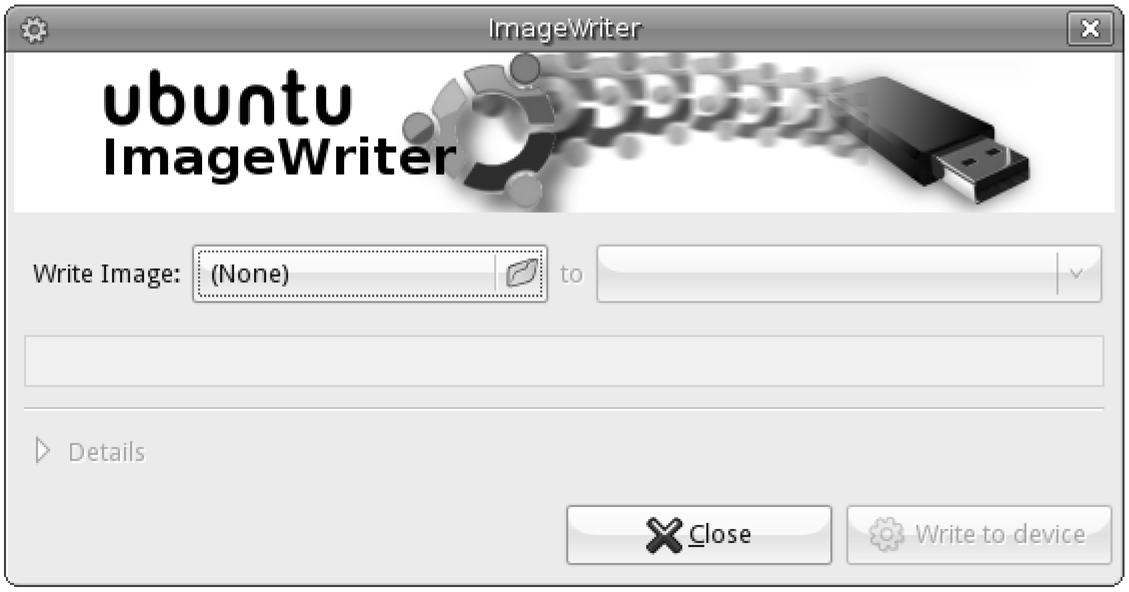
1. Download the desired .img file
2. Download Disk Imager from https://launchpad.net/win32-image-writer/+download
3. Insert your flash media
4. Note the drive letter assigned to your flash media
5. Start Disk Imager
6. Select the downloaded file and target device, and click "Write"
7. Remove your flash media when the operation is complete

Command Line Interface :

1. Download the desired .img file
2. Download flashnul from http://shounen.ru/soft/flashnul
3. Attach your USB drive
4. Run flashnul -p
5. Note the physical device number for the USB drive
6. Run flashnul <number obtained in prior step> -L \path\to\downloaded.img

7. Answer "yes" if the selected destination device is correct
8. Remove your USB drive when the command completes

9.2 Ubuntu :



9.2) Graphical Interface for Linux

Steps :

1. Download the desired .img file
2. Install the usb-imagewriter package
    1. If your release does not include this, download it from Oliver's PPA
    2. If imagewriter fails to launch, you may need to install python glade2 support. Install the python-glade2 package or Run sudo apt-get install python-glade2
    3. If your release does not include it and you are running 9.04 Jaunty Jackalope then run this command from the console:
        1. sudo apt-get install usb-imagewriter
3. Open Applications -> Accessories -> Image Writer
    1. KDE users will find this in Applications -> Utilities -> Image Writer
    2. from the command line, from the console:
        1. sudo imagewriter

   3. on some usb-imagewriter versions (console command: imagewriter) the application fails to write if the image path contains blank spaces, exiting with "IndexError: list index out of range".
4. Insert your flash media
5. Select the downloaded file and flash device, and click "Write to Device"
6. Remove your device when the operation is complete

Command Line Interface :

Be very careful about which /dev device you write to. If your machine is booted up off of disk /dev/sda, and your usb stick is on /dev/sdc, and you accidentally write to /dev/sda instead of /dev/sdc, your filesystem will be irreparably damaged and you will lose all of your files.

1. Download the desired .img file
2. Open a terminal and insert your flash media
3. Look at the output of sudo dmesg | tail -20 to determine the device node assigned to your flash media (ignore the device number; e.g. /dev/sdc, not sdc1)
   1. Example output of dmesg, where the device node is 'sdc':
   2. [5046.396364] usb-storage: device scan complete
   3. [5046.397075] scsi 10:0:0:0: Direct-
      Access USB Flash Memory 1.00 PQ: 0 ANSI: 0 CCS
   4. [5047.068761] sd 10:0:0:0: [sdc] Mode Sense: 23 00 00 00
   5. [5047.068769] sd 10:0:0:0: [sdc] Assuming drive cache: write through
   6. [5047.075021] sdc: sdc1
   7. [5047.076459] sd 10:0:0:0: [sdc] Attached SCSI removable disk
4. Run sudo umount /dev/devicenode
5. Run sudo dd if=/path/to/downloaded.img of=/dev/devicenode bs=1M
6. Remove your flash media when the command completes (you may need to wait a few extra seconds for it to finish)

9.3 Mac OS X :

Command Line Interface :

1. Download the desired .img file
2. Open a Terminal (in /Applications/Utilities/)
3. Run diskutil list to get the current list of devices
4. Insert your flash media
5. Run diskutil list again and determine the device node assigned to your flash media (e.g. /dev/disk2)
6. Run diskutil unmountDisk /dev/disk*N* (replace *N* with the disk number from the last command; in the previous example, *N* would be2)
7. Execute sudo dd if=/path/to/downloaded.img of=/dev/rdisk*N* bs=1m (replace /path/ to/downloaded.img with the path where the image file is located; for example, ./ubuntu.img, /dev/rdiskN is faster than /dev/diskN). If you see the errordd: Invalid number `1m', you are using GNU dd. Use the same command but replace bs=1m with bs=1M.
8. Run diskutil eject /dev/disk*N* and remove your flash media when the command completes

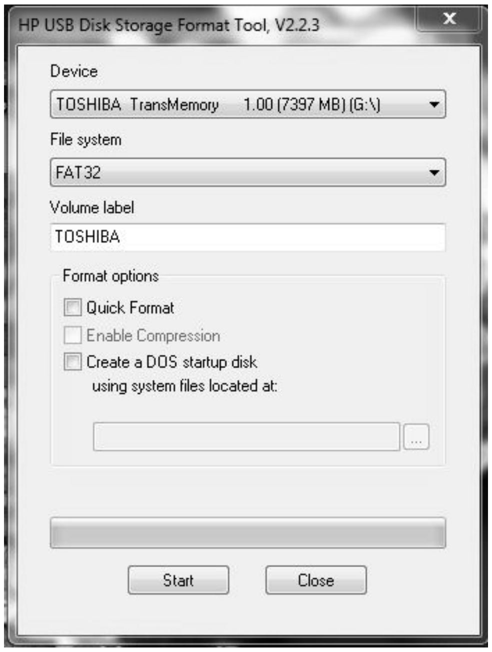
Formatting SD card with FAT32

10.1 Overview :

We are using “HP USB Disk Storage Format Tool” for formatting our SD card after porting OS into it. Once you port the image of OS the original size of SD card will reduced to some MBs because the OS has blocked some part of memory and so that part cannot be accessed by user before formatting memory card. This utility will format any USB flash drive, with your choice of FAT, FAT32, or NTFS partition types.Optionally you can also make the disk BOOTABLE by specifying a file location.

FIXES:

- Allows creation of a FAT32 volume larger than 32 GB.
- Fixes installation issue where installation process stopped after the earlier version of software was uninstalled and the new software was not automatically installed. The installation process now restarts automatically to install the new software after uninstalling the older version.


HP USB Disk Storage Format Tool, V2.2.3

Device: TOSHIBA TransMemory 1.00 (7397 MB) (G:\)

File system: FAT32

Volume label: TOSHIBA

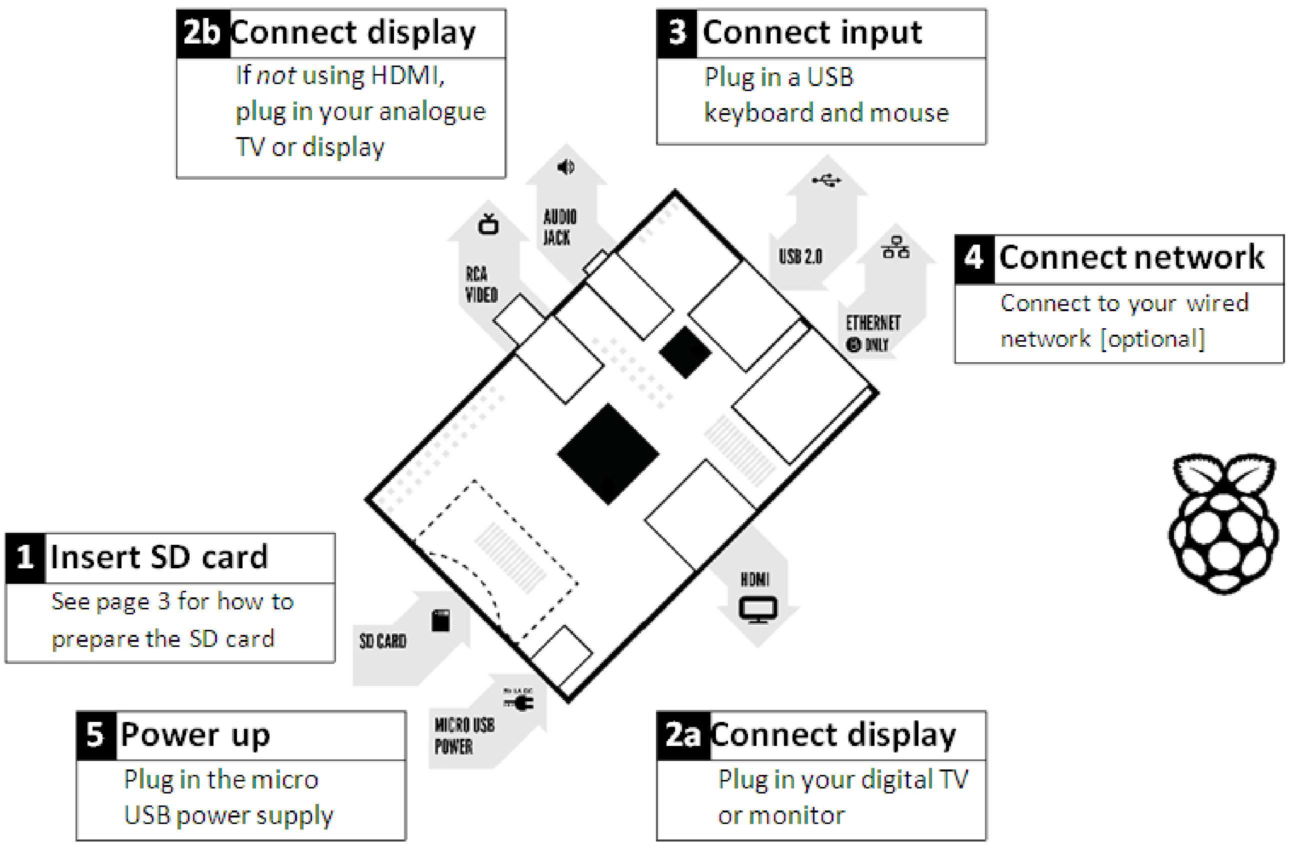
Format options
- [ ] Quick Format
- [ ] Enable Compression
- [ ] Create a DOS startup disk using system files located at:

10.1) GUI of HP USB SD card Format Tool

RaspberryPi setup

11.1 Overview :

There are many ways to setup the RaspberryPi, depending on your needs and the peripherals you wish to use. The set up process can be completed in five basic steps as shown.



2b Connect display
If *not* using HDMI, plug in your analogue TV or display
3 Connect input
Plug in a USB keyboard and mouse
AUDIO JACK
RCA VIDEO
USB 2.0
ETHERNET
B ONLY
4 Connect network
Connect to your wired network [optional]
1 Insert SD card
See page 3 for how to prepare the SD card
SD CARD
HDMI
MICRO USB POWER
5 Power up
Plug in the micro USB power supply
2a Connect display
Plug in your digital TV or monitor

11.1) Hardware features of Raspberry Pi

11.2 Basic Equipment required for general Setup :

1. Micro USB Charger (rated at 5V 700mA minimum), plus micro USB cable if needed.
2. SD Card (2Gb up to SDHC 32Gb), plus suitable card reader.
3. HDMI Cable/RCA cable for display
4. Compatible Mouse/Keyboard
5. Powered USB Hub (required for high powered USB devices)
6. Network Cable (Ethernet)

11.3 Quick Start :

- **Unzip the file that you just compiled in Ubuntu:**

  a) Right click on the file and choose .Extract all..

  b) Follow the instructions.you will end up with a file ending in .img

  This .img file can only be written to your SD card by special disk imaging software, so.

- **Download the Win32DiskImager software:**

  a) Download win32diskimager-binary.zip (currently version 0.6)

  b) Unzip it in the same way you did the Raspbian .zip file

  c) You now have a new folder called win32diskimager-binary

  You are now ready to write the Raspbian image to your SD card.

- **Writing Raspbian to the SD card:**

  a) Plug your SD card into your PC

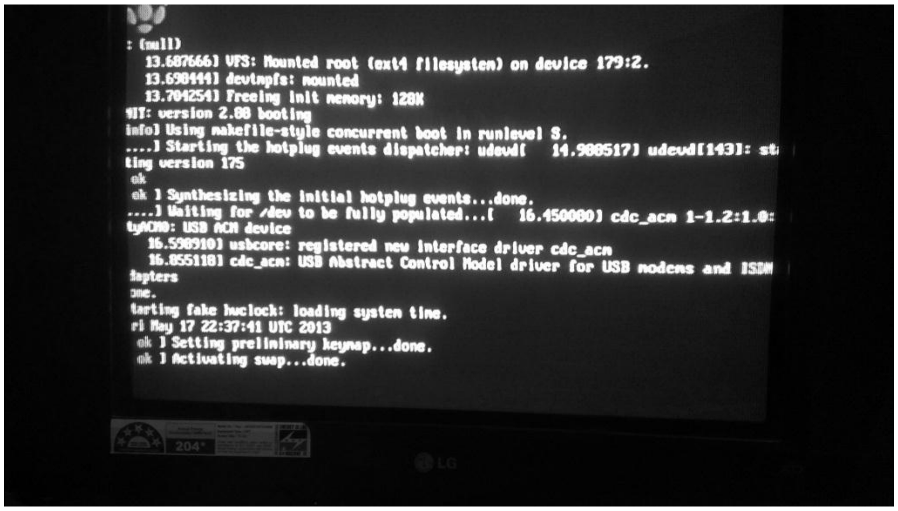
  b) In the folder you made in step 3(b), run the file named Win32DiskImager.exe

If the SD card you are using isn't found automatically then click on the drop down box and select it. Then click Write. After a few minutes you will have an SD card that you can use in your Raspberry Pi.

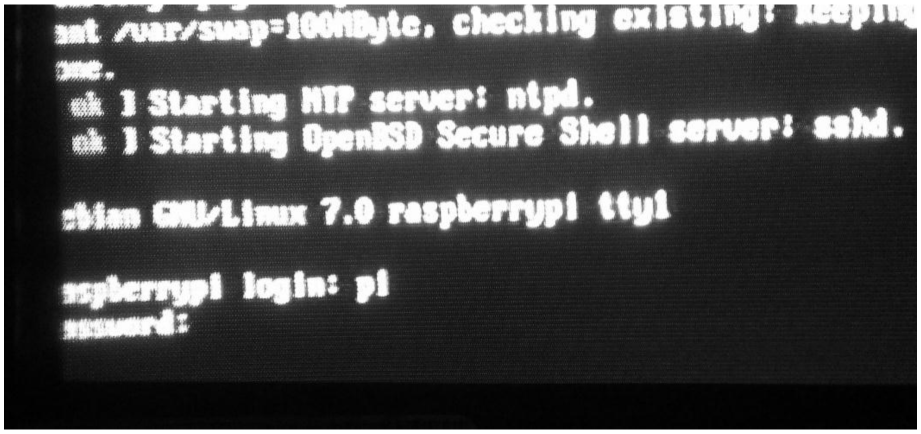
- **Booting for the first time:**

a) at Raspberrypi login screen type “pi” and you will be asked for password which is "raspberry".

b) When you see the prompt: pi@raspberry ~ $ type "startx" for a graphical environment



```
: (null)
   13.687666] VFS: Mounted root (ext4 filesystem) on device 179:2.
   13.690444] devtmpfs: mounted
   13.704254] Freeing init memory: 128K
NIT: version 2.88 booting
info] Using makefile-style concurrent boot in runlevel S.
....] Starting the hotplug events dispatcher: udevd[   14.980517] udevd[143]: st
ting version 175
 ok
 ok ] Synthesizing the initial hotplug events...done.
....] Waiting for /dev to be fully populated...[   16.450080] cdc_acm 1-1.2:1.0:
tyACM0: USB ACM device
   16.598910] usbcore: registered new interface driver cdc_acm
   16.855118] cdc_acm: USB Abstract Control Model driver for USB modems and ISDN
dapters
one.
tarting fake hwclock: loading system time.
ri May 17 22:37:41 UTC 2013
 ok ] Setting preliminary keymap...done.
 ok ] Activating swap...done.
```



```
mt /var/swap=100MByte, checking existing: keeping
one.
 ok ] Starting NTP server: ntpd.
 ok ] Starting OpenBSD Secure Shell server: sshd.

ebian GNU/Linux 7.0 raspberrypi tty1

aspberrypi login: pi
assword:
```

11.2) Screenshot of booting and login

Getting Network Acess

12.1 Overview :

For connecting RaspberryPi to network you have 3 options

1)Ethernet

2)USBtethering from your smart phone

12.2 Using Ethernet:

Connect the Ethernet Cable and open the browser and set the IP Address and proxy and get connected.

For excess internet through terminal ,type the following for setting IP

exporthttp_proxy=http://proxy IP address:port

ifconfig eth0 10.10.17.23

here**10.10.17.23**is the Proxy IP we used

nowPing to check the internet connection.

12.3 USB tethering:

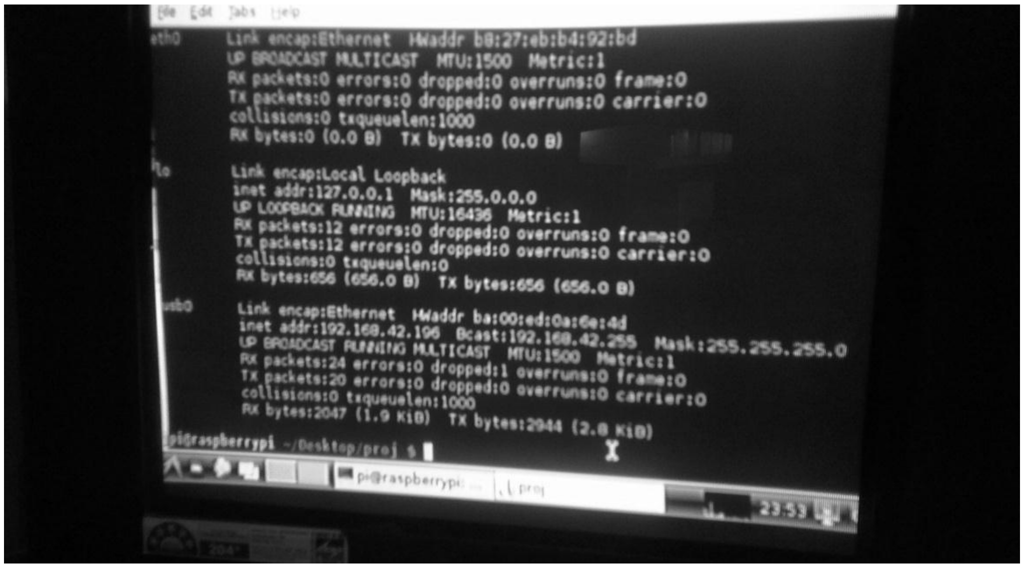
Connect your Phone and RaspberryPi via USB cable and enable USB tethering in your device. Then open terminal and follow the steps given below for configuring your PI

1) Get root access by typing “sudo bash”
2) Go to the directory etc/network
3) Open and file interfaces with vi command and add following lines in file

```
auto eth0
iface eth0 inetdhcp
iface usb0 inetdhcp
```

4) Save the file and type the following command in terminal for configure the changes, “ifup usb0”
5) Now for checking whether your Pi have been configured or not, type following

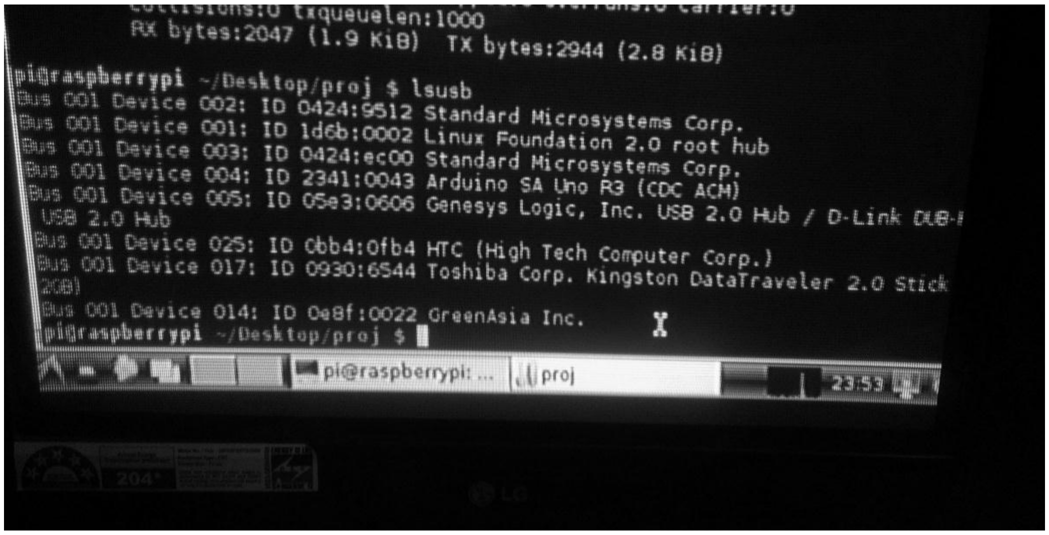
Ifconfig :



```
eth0      Link encap:Ethernet  HWaddr b8:27:eb:b4:92:bd
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:12 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:656 (656.0 B)  TX bytes:656 (656.0 B)

usb0      Link encap:Ethernet  HWaddr ba:00:ed:0a:6e:4d
          inet addr:192.168.42.196  Bcast:192.168.42.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:24 errors:0 dropped:1 overruns:0 frame:0
          TX packets:20 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2047 (1.9 KiB)  TX bytes:2944 (2.8 KiB)

pi@raspberrypi ~/Desktop/proj $
```

12.1) Screenshot of ifconfig

Lsusb :



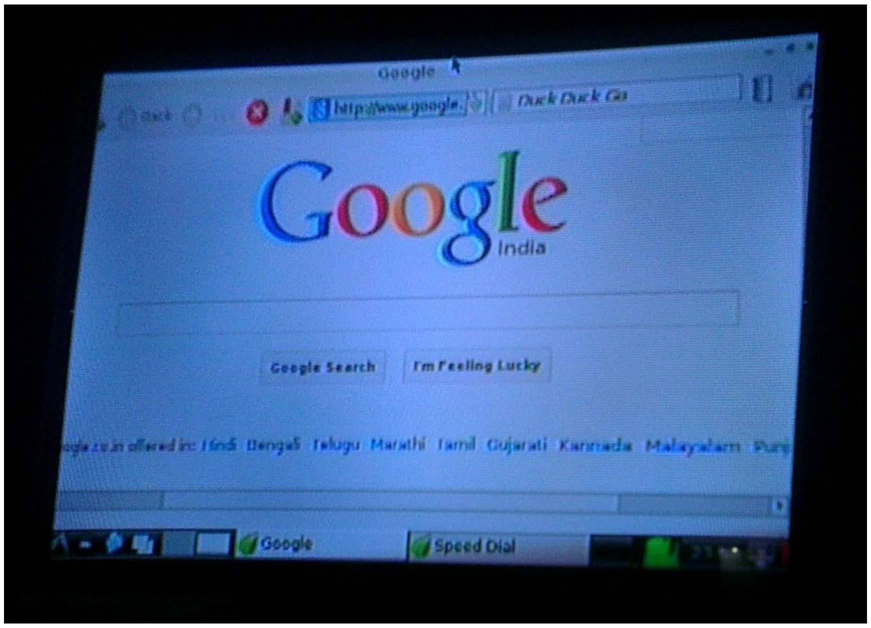
```
          collisions:0 txqueuelen:1000
          RX bytes:2047 (1.9 KiB)  TX bytes:2944 (2.8 KiB)

pi@raspberrypi ~/Desktop/proj $ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 2341:0043 Arduino SA Uno R3 (CDC ACM)
Bus 001 Device 005: ID 05e3:0606 Genesys Logic, Inc. USB 2.0 Hub / D-Link DUB-
 USB 2.0 Hub
Bus 001 Device 025: ID 0bb4:0fb4 HTC (High Tech Computer Corp.)
Bus 001 Device 017: ID 0930:6544 Toshiba Corp. Kingston DataTraveler 2.0 Stick
2GB)
Bus 001 Device 014: ID 0e8f:0022 GreenAsia Inc.
pi@raspberrypi ~/Desktop/proj $
```

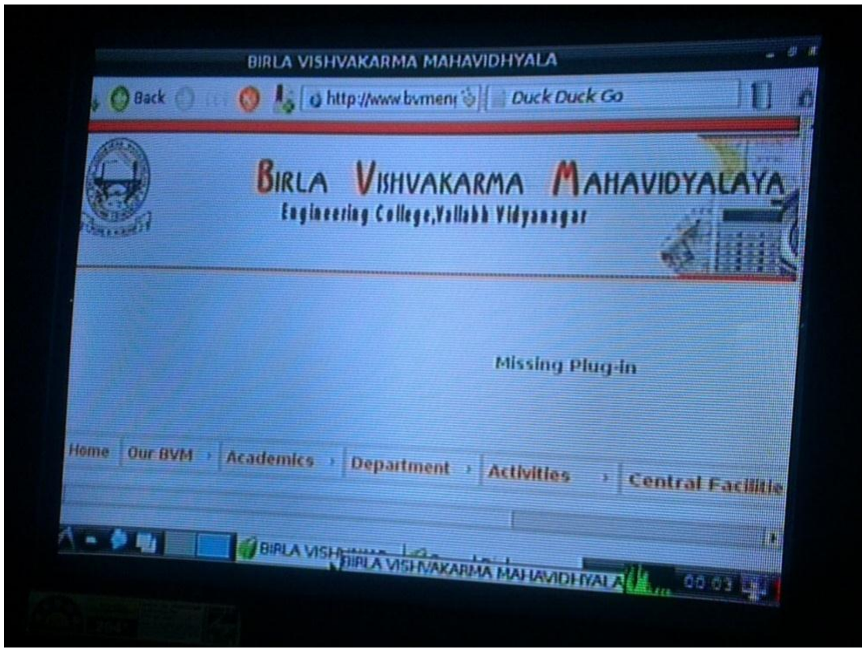
12.2) Screenshot of lsusb

Accessing internet in Raspberry Pi

13.1 Overview :

We have dedicated browser in Raspberry Pi named “Midori”. So we can access internet through it using any of two method either with Ethernet or tethering.



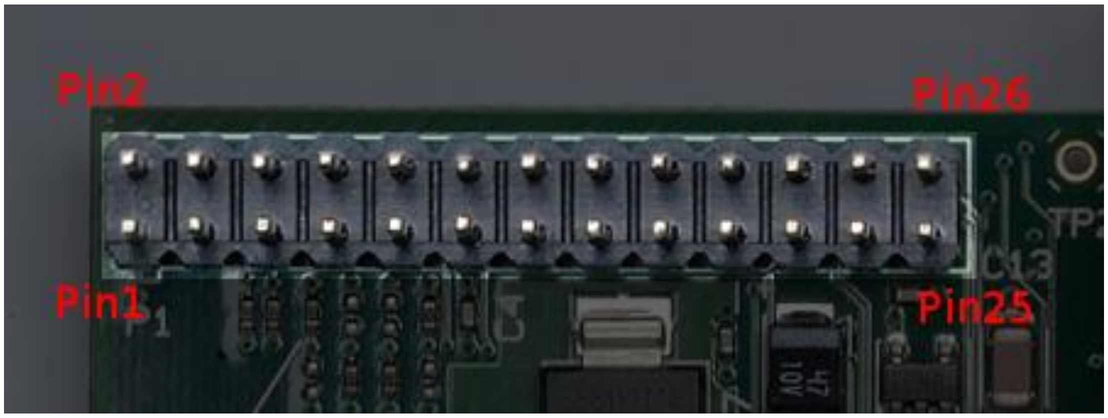


13.1) Screenshot of various sites

Introduction to GPIO of RaspberryPi

14.1 Overview :

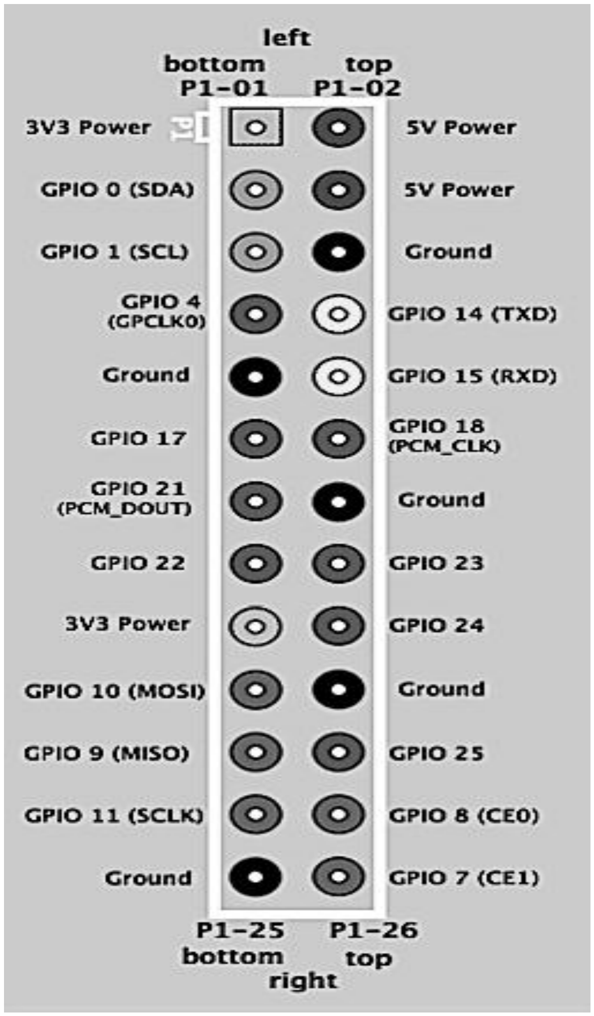
General Purpose Input/Output (a.k.a. GPIO) is a generic Pin on a chip whose behavior (including whether it is an input or output Pin) can be controlled (programmed) through software.RaspberryPi has broadcoms BC2835 as core which is a ARM1176JZF-S processor which supports 54 general purpose Input /Output Pin. All GPIO Pins have at least two alternative functions within BCM. RaspberryPi has pre-configured few Pins as USB, Ethernet and HDMI ports and rest 26 Pin are provided for general purpose use. The production RaspberryPi board has a 26-Pin 2.54 mm (100 mil)expansion header, marked as P1, arranged in a 2x13 strip. They provide 8 GPIO Pins plus access to I²C, SPI, UART, as well as +3.3 V, +5 V and GND supply lines. Pin one is the Pin in the first column and on the bottom row.



14.1) Image of 2x13 strip provided to use as GPIO

GPIO voltage levels are 3.3 V and are not 5 V tolerant. There is no over-voltage protection on the board. All the GPIO Pins can be reconfigured to provide alternate functions, SPI, PWM, I²C and so. At reset only Pins GPIO 14 & 15 are assigned to the alternate function UART, these two can be switched back to GPIO to provide a total of 17 GPIO Pins. Each GPIO can interrupt, high/low/rise/fall/change.

A detail image of GPIO header is given below



14.2) Pin description of GPIO

14.2 Power Pins:

The maximum permitted current draw from the 3.3 V Pins is 50 mA.
Maximum permitted current draw from the 5 V Pin is the USB input current (usually 1 A) minus any current draw from the rest of the board.

- Model B: 1000 mA - 700 mA -> max current draw: 300 mA

Be very careful with the 5 V Pins P1-02 and P1-04, because if you short 5 V to any other P1 Pin you may permanently damage your RasPi. Before probing P1, it's a good idea to strip short Pieces of insulation off a wire and push them over the 5 V Pins so you don't accidentally short them with a probe.

The other chipset GPIO Pins accessible on the PCB but are in use are:

- GPIO16 drives status LED D5 (usually SD card access indicator)
- GPIO28-31 are used by the board ID and are connected to resistors R3 to R10 (only on Rev1.0 boards).
- GPIO40 and 45 are used by analogue audio and support PWM. They connect to the analogue audio circuitry via R21 and R27 respectively.
- GPIO46 is HDMI hotplug detect (goes to Pin 6 of IC1).
- GPIO47 to 53 are used by the SD card interface. In particular, GPIO47 is SD card detect (this would seem to be a good candidate for re-use). GPIO47 is connected to the SD card interface card detect switch; GPIO48 to 53 are connected to the SD card interface via resistors R45 to R50.

Extending the size of memory card (menu config)

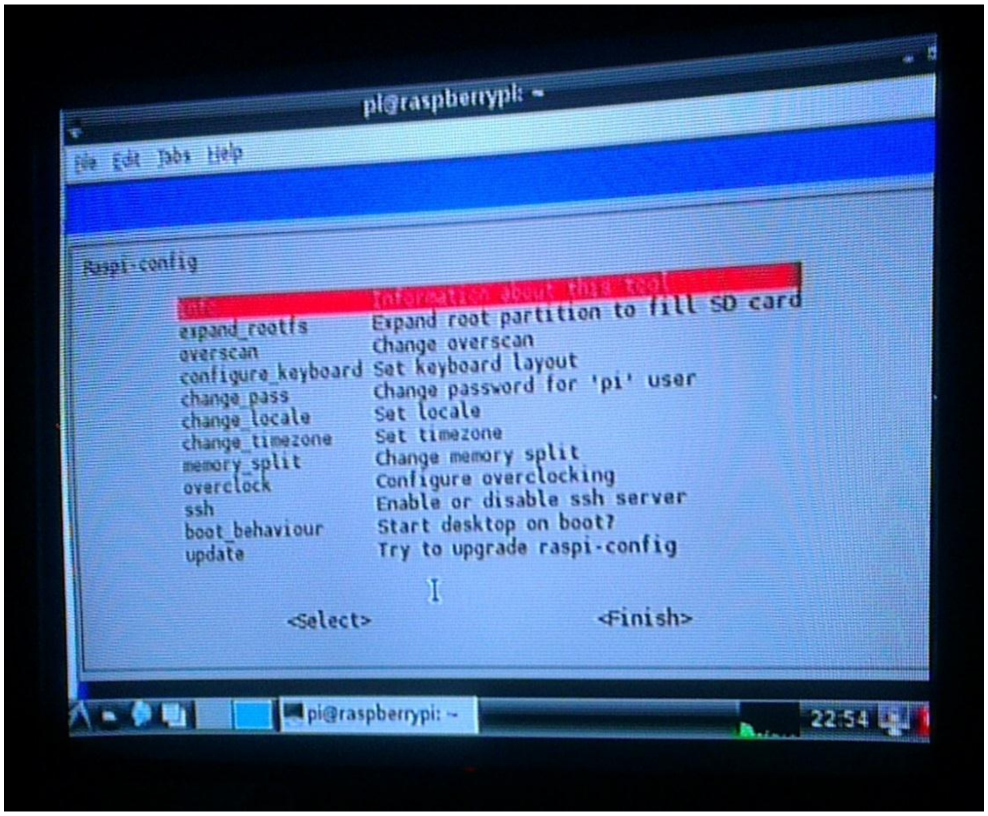
15.1 Overview :

The default size allowed by Raspbian OS is 2 GB only , no matter whatever the size of memory card. So if we want to extend the size of it then we have to do it through “menu config”.

For doing it follow this :

Go to terminal >type this :sudoraspi-config >select this : expand_rootfs>select yes , then restart Raspberry Pi

Now you have full accessibility to your memory card. The full size of memory card will now available for storage of data.



```
Raspi-config

    info                Information about this tool
    expand_rootfs       Expand root partition to fill SD card
    overscan            Change overscan
    configure_keyboard  Set keyboard layout
    change_pass         Change password for 'pi' user
    change_locale       Set locale
    change_timezone     Set timezone
    memory_split        Change memory split
    overclock           Configure overclocking
    ssh                 Enable or disable ssh server
    boot_behaviour      Start desktop on boot?
    update              Try to upgrade raspi-config

           <Select>                    <Finish>
```

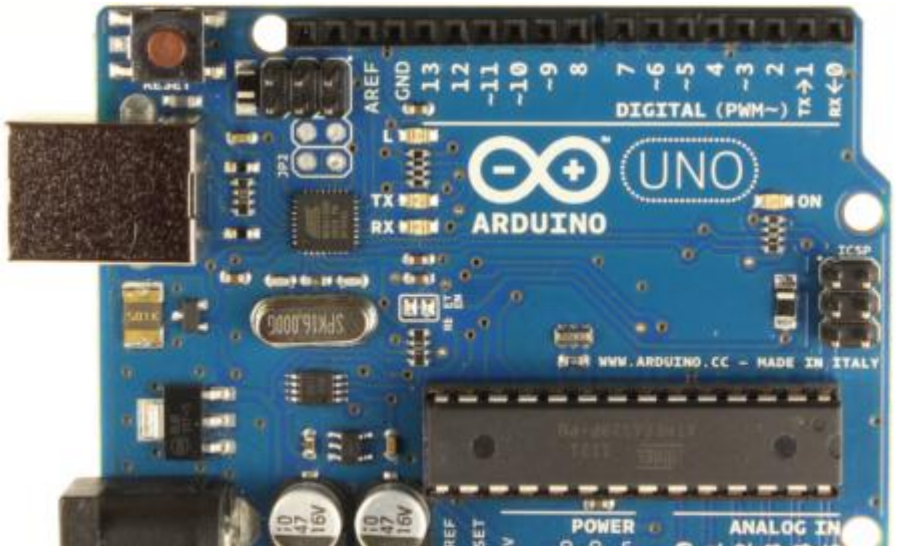
15.1) Screenshot of expand_rootfs (menu config)

Arduino UNO

16.1 Overview :

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software running on a computer (e.g. Flash, Processing, MaxMSP).



16.1) Arduino UNO board

16.2 Key features of the Arduino Uno includes:

- An open source design. The advantage of it being open source is that it has a large communityof people using and troubleshooting it. This makes it easy to find someone to help you debug your projects.

- An easy USB interface. The chip on the board plugs straight into your USB port and registers on your computer as a virtual serial port. This allows you to interface with it as through it were a serial device. The benefit of this setup is that serial communication is an extremely easy (and time-tested) protocol, and USB makes connecting it to modern computers really convenient.

- Very convenient power management and built-in voltage regulation. You can connect an external power source of up to 12v and it will regulate it to both 5v and 3.3v. It also can be powered directly off of a USB port without any external power.

- An easy-to-find, and dirt cheap, microcontroller "brain." The ATmega328 chip retails for about $2.88 on Digikey. It has countless number of nice hardware features like timers, PWM Pins, external and internal interrupts, and multiple sleep modes.

- A 16mhz clock. This makes it not the speediest microcontroller around, but fast enough for most applications.

- 32 KB of flash memory for storing your code.

- 13 digital Pins and 6 analog Pins. These Pins allow you to connect external hardware to your Arduino. These Pins are key for extending the computing capability of the Arduino into the real world. Simply plug your devices and sensors into the sockets that correspond to each of these Pins and you are good to go.

- An ICSP connector for bypassing the USB port and interfacing the Arduino directly as a serial device. This port is necessary to re-bootload your chip if it corrupts and can no longer talk to your computer.

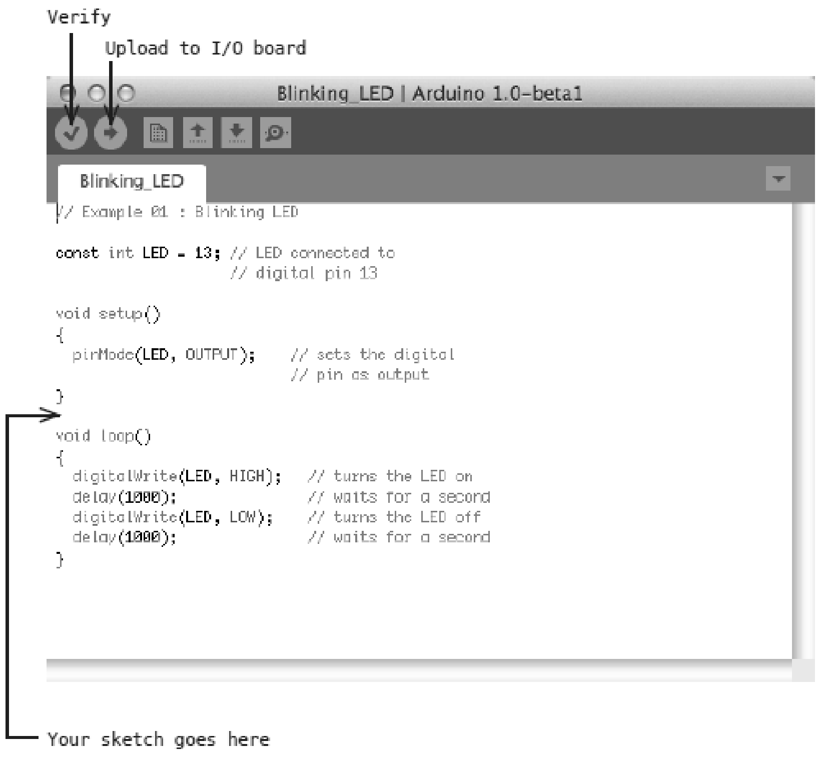
- An on-board LED attached to digital Pin 13 for fast an easy debugging of code.

- And last, but not least, a button to reset the program on the chip.

16.3 Arduino UNO IDE :

Arduino has dedicated IDE (integrated Development Environment) which uses some low level language which is easy to understand as well as write and IDE converts this language into C/C++ for execution of code.

Each program written into Arduino IDE is called “sketch” . here below we have shown the sample sketch for the LED blinking code.



```
// Example 01 : Blinking LED

const int LED = 13; // LED connected to
                    // digital pin 13

void setup()
{
  pinMode(LED, OUTPUT);    // sets the digital
                           // pin as output
}

void loop()
{
  digitalWrite(LED, HIGH);   // turns the LED on
  delay(1000);               // waits for a second
  digitalWrite(LED, LOW);    // turns the LED off
  delay(1000);               // waits for a second
}
```

16.2) Arduino UNO sketch

Verify :

This button is used for compile the written sketch.

Upload :

This button is used for uploading the code into connected board. Before uploading the code into board the IDE first compiles it and checks for and error.

The sketch has two main body

1) setup

2)loop

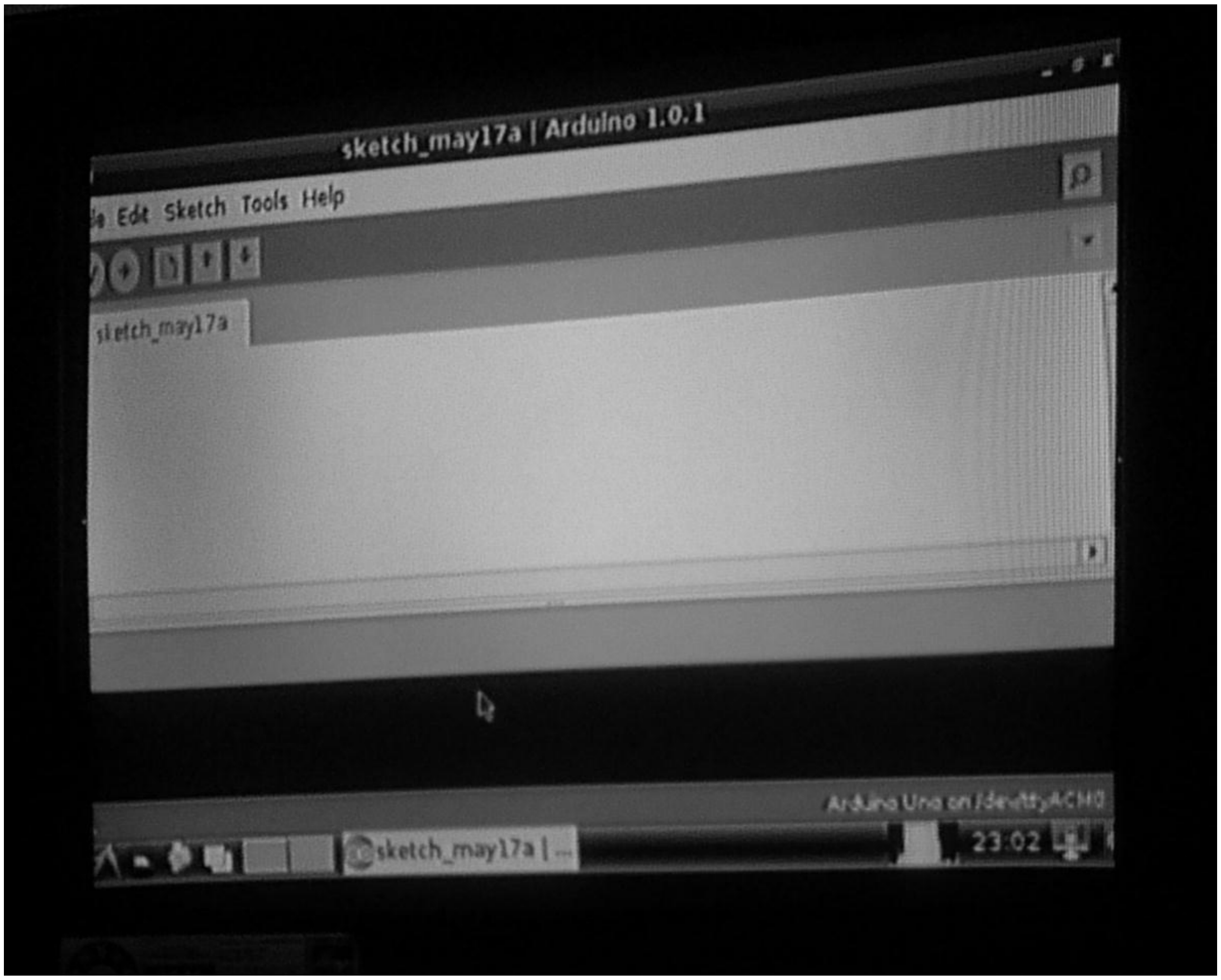
Setup :

In this portion we assigns the pins to various names and defines that either that pin should be input of output. In short we make a setup for the pins.

Loop :

In this portion we write that part of code that should be repeated continuously. So this part does the work of "while 1".

16.4 Arduino IDE in Raspbian :



16.3) Screenshot of Arduino IDE

Making Arduino and RaspberryPi talk

17.1) Connection of Arduino and Raspberry Pi

17.1 Overview :

There is a Python library for serial communications called 'pySerial' which allows RaspberryPi to serially communicate with Arduino using USB.

Step 1:

Now load the scratch for serial transmission on Ardiuno and connect it with RaspberryPi using USB

Step 2 :

Power up RaspberryPi and login

Step 3 :

Browse and download **pyserial-2.5.tar.gz (106.3 kB)** and save it somewhere convenient. I saved it to the 'other' folder on the Desktop.

Step 4 :

This is a gziped tar file. Which needs unzipping and untaring. To unzip it open a Terminal, which you will find from the 'start menu' under 'accessories'. Now paste the following commands into it.

```
cd /home/Pi/Desktop/other
gunzip pyserial-2.5.tar.gz
tar - xvf pyserial-2.5.tar
```

Step 5 :

Install pySerial, by typing these lines in your terminal window:

```
cd pyserial-2.5
sudo python setup.py install
```

Step 6 :

Run Python 2. You will find this from the menu under Programming - Use Python 2

Step 7 :

Now we just need to write following Python code in python 2 to access the Serial port :

```
import serial

ser=serial.Serial('/dev/ttyACM0',9600)

while 1:

        ser.readline()
```

This will make Pi ready to read the data transmitted serially by Ardiuno and The data will be shown on Screen as soon as you type last command.

Note :

Please make sure that baudrate of both the device is same.for checking the baudrate of RaspberryPi go to file serial.py and compare the baudrate mentioned there with your baudrate if it id same it is good if not then change it with 9600 standard baubrate otherwise your evice will not talk.

LM-35 temperature sensor

18.1 General Description :

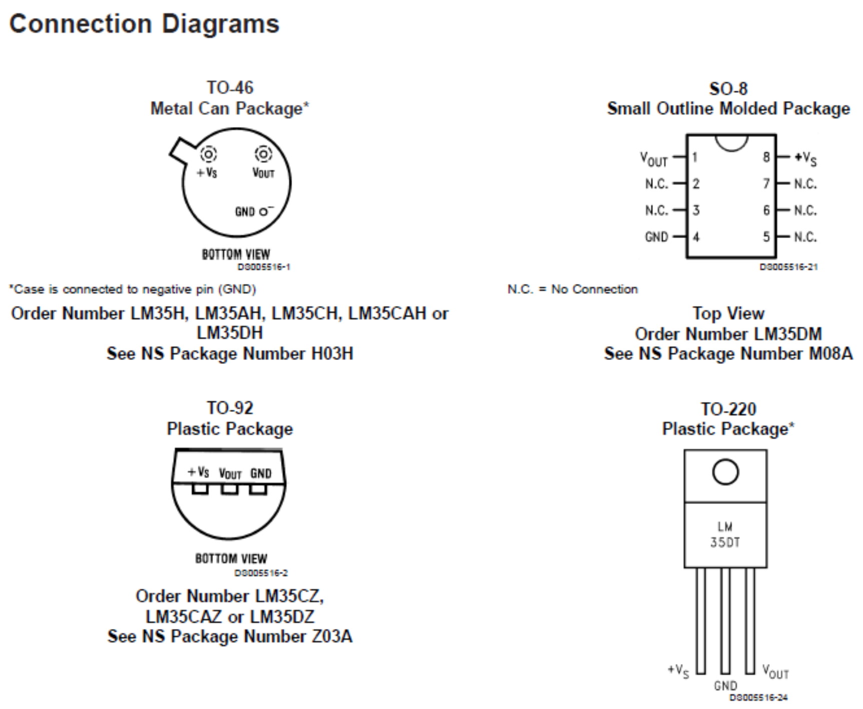
The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in
° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling.

The LM35 does not require any external calibration or trimming to provide typical accuracies of ±1⁄4°C
at room temperature and ±3⁄4°C over a full −55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make
interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a −55° to +150°C temperature range, while the LM35C is rated for a −40° to +110°C range (−10° with improved accuracy).

The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

18.2 Features :

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full −55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±1⁄4°C typical
- Low impedance output, 0.1 W for 1 mA load


Connection Diagrams

TO-46
Metal Can Package*

*Case is connected to negative pin (GND)

Order Number LM35H, LM35AH, LM35CH, LM35CAH or LM35DH
See NS Package Number H03H

SO-8
Small Outline Molded Package

N.C. = No Connection

Top View
Order Number LM35DM
See NS Package Number M08A

TO-92
Plastic Package

Order Number LM35CZ, LM35CAZ or LM35DZ
See NS Package Number Z03A

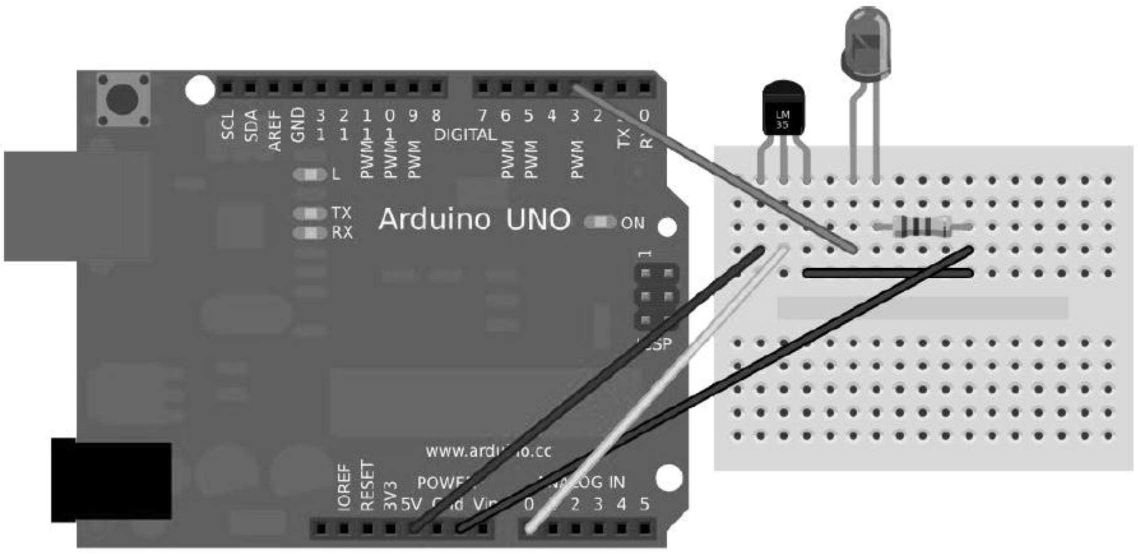
TO-220
Plastic Package*

18.1) Pin diagram of various LM-35 sensors

Temperature sensing using RaspberryPi and Ardiuno

19.1 Overview :

For this connect RaspberryPi and Ardiuno using USB as mentioned earlier. Connect a temperature sensor with analog input of Ardiuno and load the following scratch code in Ardiuno



19.1) Connection of LM-35 to Arduino

19.2 Ardiunosratch code:

```
        constintledPin = 13;

        float tempC;

        int reading;

        inttempPin = 0;

void setup()
```

```
            {
            PinMode(ledPin, OUTPUT);
            Serial.begin(9600);
            analogReference(INTERNAL);}
void loop()
            {
              reading = analogRead(tempPin);
            tempC = reading / 9.31;
            Serial.println(tempC);
              if (Serial.available())
              {
                 flash(Serial.read() - '0');
              }
              delay(1000);
            }
void flash(int n)
            {
              for (int i = 0; i < n; i++)
              {
```

```
        digitalWrite(ledPin, HIGH);

          delay(100);

        digitalWrite(ledPin, LOW);

          delay(100);

}
```

And now follow the step 6 and step 7 mentioned in previous chapter

Data acquisition application

20.1 Overview :

Finally we made an application which continuously senses temperature using LM-35 temperature sensor and displays the output. Meanwhile we also plotted the previously generated temperature readings.

We have divided the overall task into 3 phases :

1 : First upload the above Arduino code for temperature sensing using Arduino IDE into the Arduino UNO board.

2 : Now make a python file named “dataLogger.py” and add following lines into it

```
“ #!/usr/bin/python

'''

By Shivam Chudasama - may 2013

This program reads data coming from the serial port and saves that data to a text file. It expects data in the format:

"temperature_reading"

It assumes that the Arduino shows up in /dev/ttyACM0 on the Raspberry Pi which should happen if you're using Debian.

'''

import serial

ser = serial.Serial('/dev/ttyACM0',9600)

try:
	while 1:

		line=ser.readline().rstrip()

		temp2=line

		print("%s"%(temp2))

		f=open('tempLog.dat','a')

		print>>f,("%s"%(temp2))

		f.close()

exceptKeyboardInterrupt:

	print "\ndone"

#finish ”
```

3 : Now for plotting the graph using gnuplot we have to first install gnuplot using terminal by following command :

```
Sudo apt-get installgnuplot
```

Now after installing gnuplot we have to again make one new file named “plotData.plt” and add following lines into it :

```
“ reset

set title "temp sensor"

set term png

setxtics rotate by -45

setxlabel "time (s)"
```

setylabel "Temperature (C)"

setyticsnomirror

set grid

set output "data.png"

set key left

plot "tempLog.dat" title "" ”

Now after doing all this we have to do following steps :

1 :

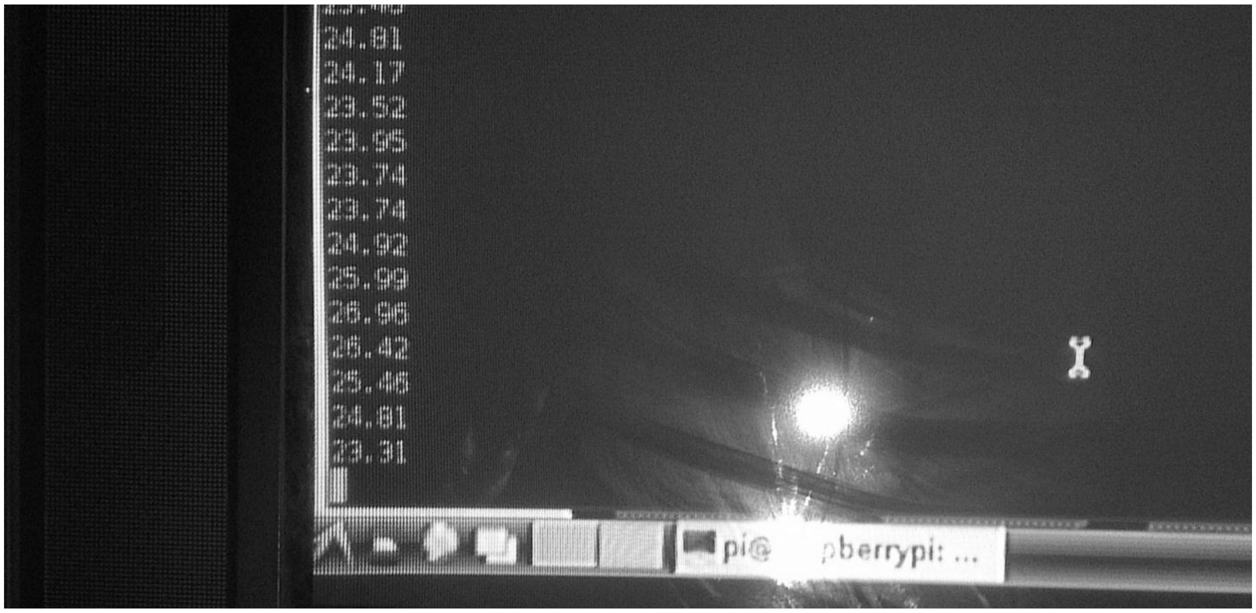
Go to that particular file location where you have saved dataLogger.py &plotData.plt files

2 :

Now type following command

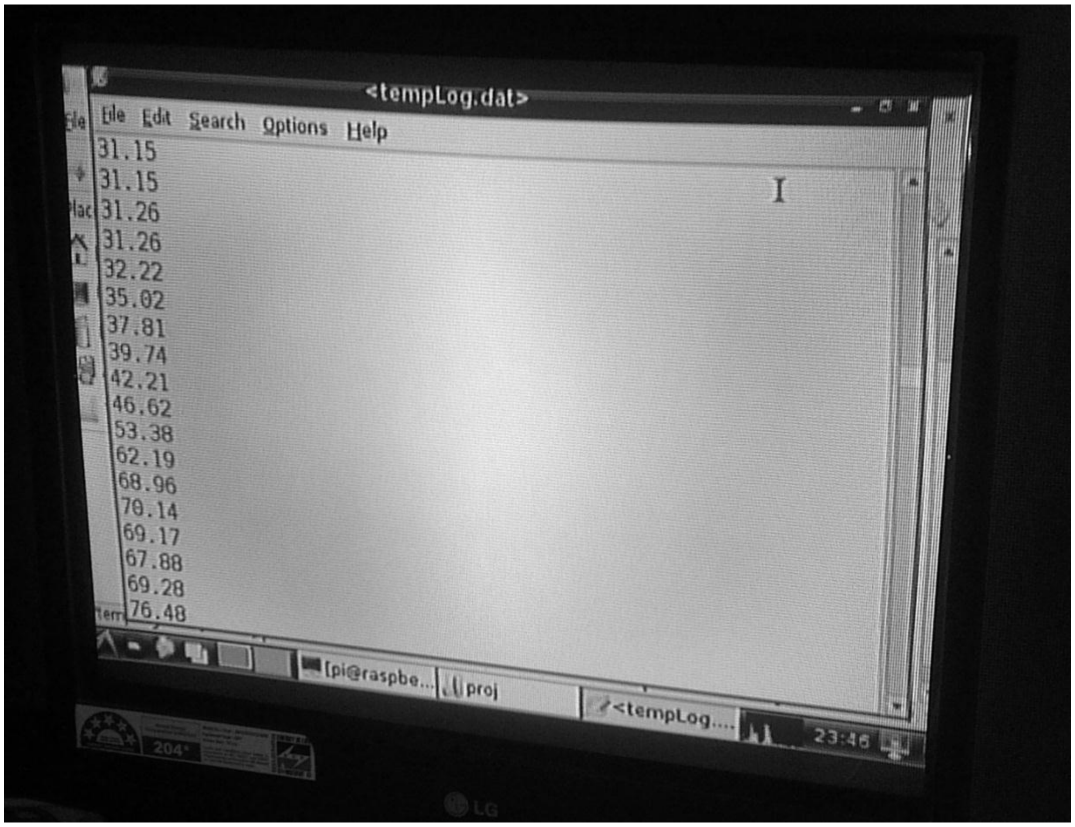
Sudo python dataLogger.py

This code will continuously shows the temperature in terminal and at the same time that data will stored into the file named “tempLog.dat”. The temperature is stored with the rate of one entry per second and listed all the sensed temperature into the “tempLog” until we give keyboard interrupt.



```
24.81
24.17
23.52
23.95
23.74
23.74
24.92
25.99
26.96
26.42
25.46
24.81
23.31
```

20.1) Screenshot of temperature sensing in terminal



```
31.15
31.15
31.26
31.26
32.22
35.02
37.81
39.74
42.21
46.62
53.38
62.19
68.96
70.14
69.17
67.88
69.28
76.48
```

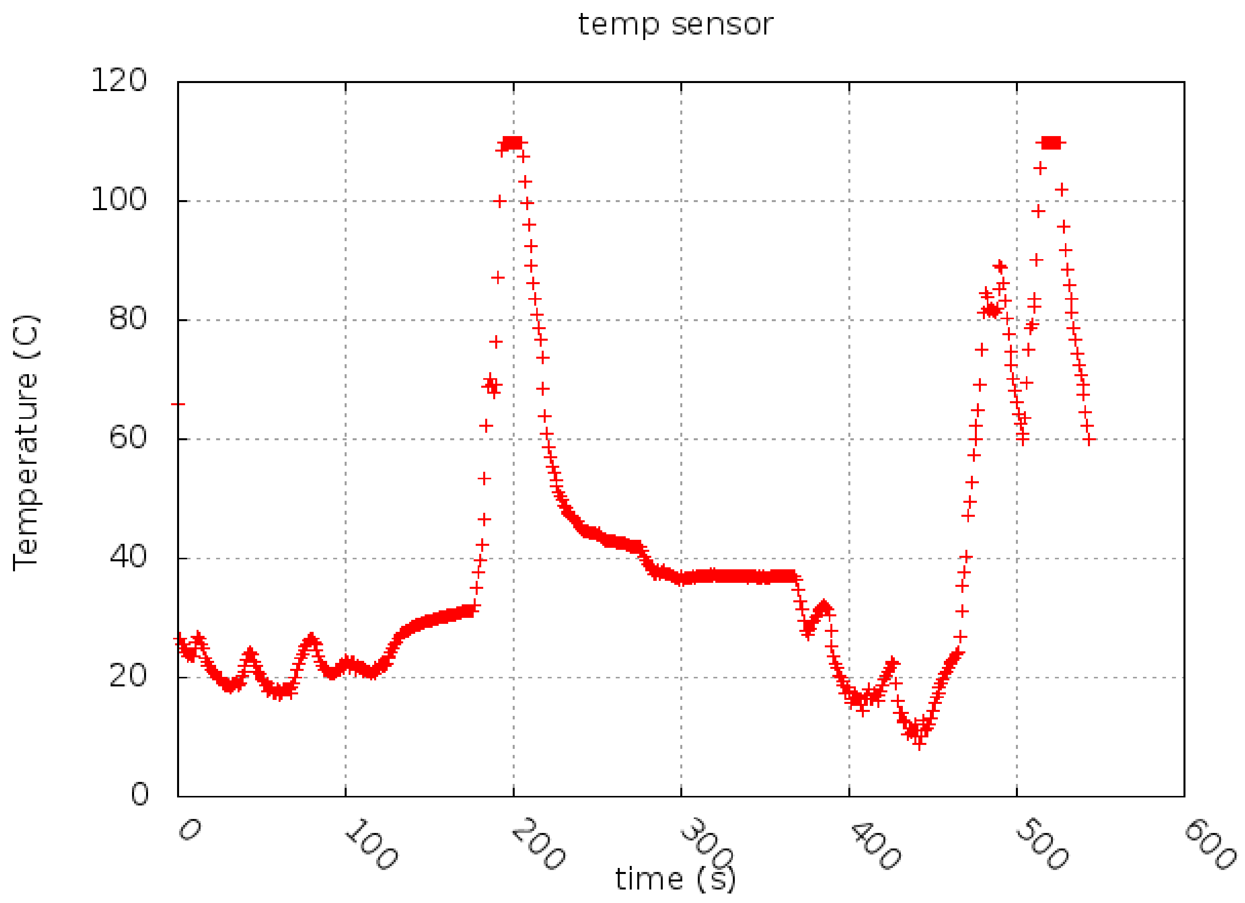
20.2) Screenshotof tempLog.dat file

3 :

Now type following command

SudognuplotplotData.plt

This code will generate the plot of collected temperature data which are stored into the “tempLog.dat” file and generates the PNG file named “data.png” and in this file the image of graph plotted is shown.



20.3) Plot generated by gnuplot

Conclusion

This report explains the design and development of the Data acquisition application using debian operating system. Here we have studied various OS and ported the OS to Raspberry pi Board model B and then we have made Ardiuno UNO board and Raspberry pi board talk serially and exchange data. We have connected LM35 temperature sensor with Ardiuno which reads the temperature from environment and serially provide it to Raspberry pi. At pi we are generating a log file and plotting the graph using a python programand displaying this data on Linux terminal. The combination of Raspberry Pi and Ardiuno have made the connections easy and have simplified the entire task with high speed, better accuracy and good flexibility. Besides as Raspberry pi is new in embedded market Many further improvements can applied on it like adding more than one sensor , plotting the graph in real time .

Bibliography

Books:

1. Magpi Issues (1 to 12)
2. Getting started with Arduino
3. Getting started with Raspberry Pi

Websites:

1. http://elinux.org/RPi_Beginners
2. http://elinux.org/RPi_Education
3. http://elinux.org/RPi_Hardware
4. http://elinux.org/Rpi_Low-level_peripherals
5. http://elinux.org/RPi_Expansion_Boards
6. http://squirrelhosting.co.uk/hosting-blog/hosting-blog-info.php?id=22
7. https://github.com/raspberrypi
8. https://github.com/raspberrypi/linux
9. http://www.cl.cam.ac.uk/freshers/raspberrypi/tutorials/os/
10. http://www.osehra.org/blog/cross-compiling-raspberry-pi
11. http://code.google.com/p/usb-serial-for-android/

Research papers:

1. Porting Linux kernel to an ARM based development board(ISSN:2248-9622)
2. Porting Android to Arm for wireless Data Encryption and Decryption (ISSN 2250-2459)