ABSTRACT

Notice Board is primary thing in any institution or public utility places like bus stations, railway stations, colleges, malls, etc. But sticking various notices day to day is a difficult process. A separate person is required to take care of this notices display. This project is about advanced wireless notice board. The project is built around ARM controller raspberry-pi which is heart of the system**.**

The project aims at designing a LCD Monitor based message display controlled from an Android mobile phone. The proposed system makes use of wireless technology to communicate from Android phone to Raspberry Pi (ARM7) display board. The system has a provision for giving message through text.

Automation is the most frequently spelled term in the field of electronics. The hunger for automation brought many revolutions in the existing technologies. This project makes use of an onboard computer, which is commonly termed as **Raspberry Pi** processor. It acts as heart of the project. This onboard computer can efficiently communicate with the output and input modules which are being used. The **Raspberry Pi** is a credit-card-sized single-board computer developed in the UK by the Raspberry Pi Foundation.

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Android boasts a healthy array of connectivity options, including Wi-Fi, Bluetooth, and wireless data over a cellular connection (for example, GPRS, EDGE (Enhanced Data rates for GSM Evolution), and 3G). Android provides access to a wide range of useful libraries and tools that can be used to build rich applications. In addition, Android includes a full set of tools that have been built from the ground up alongside the platform providing developers with high productivity and deep insight into their applications.

CHAPTER 1

1.1 INRODUCTION

In this world everyone needs a comfort living life. Man has researched different technology for his sake of life. In today's world of connectedness, people are becoming accustomed to easy access to information. Whether it's through the internet or television, people want to be informed and up-to-date with the latest events happening around the world. Wired network connection such as Ethernet has many limitations depending on the need and type of connection. Now a day's people prefer wireless connection because they can interact with people easily and it require less time. The main objective of this project is to develop a wireless notice board that display message sent from the user (phone) and to design a simple, easy to install, user friendly system, which can receive and display notice in a particular manner With respect to date and time which will help the user to easily keep the track of notice board every day and each time he uses the system. Wi-Fi is the wireless technology used.

1.2 Need for Electronic Notice Board

Notice Board is primary thing in any institution / organization or public utility places like bus stations, railway stations and parks. But sticking various notices day-to-day is a difficult process. A separate person is required to take care of this notices display. This project deals about an advanced hi-tech wireless notice board. A authenticated person can send message from a remote place which is visible on the LCD/LED Monitor.

1.3 PROBLEM STATEMENT

The project is about displaying data sent from mobile phone over a network to remote server. The server has to receive messages received from client process it and displays it on LCD/LED display. The text data is sent in real time is to continuously monitored and validated before it is displayed on the monitor. The entire system shall be a stand lone system with no human intervention.

1.4 PROJECT OBJECTIVE

The main objective is design an automatic, self enabled highly reliable electronic notice board. A display connected to a server system should continuously listen for the incoming calls from client or user process it and display it on LCD screen. Message displayed should be updated every time the user sends new data. Only authenticated people should be able to access the server. User should get an update every time the data is displayed on the monitor

1.5 Problem Solving Approach

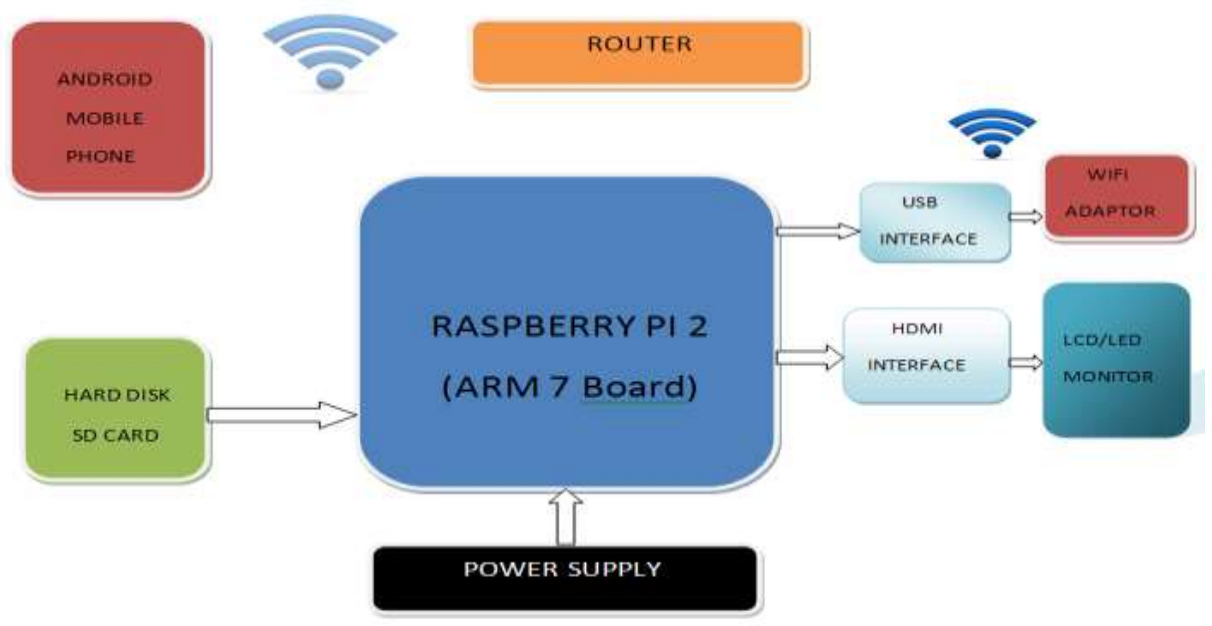
- To establish a TCP/IP client server wireless connection.
- Configure display side setup as Server.
- Configure user side setup as Client
- Use Sockets to send and receive data over wireless network.
- To display real time data on LCD Screen.
- Use Raspberry Pi as server.
- Use Android mobile phone as Server.
- Attach LCD Screen using Raspberry Pi HDMI

CHAPTER 2

2.1 Overview

The project deals with displaying text messages sent by the user from a remote place. A sever is fixed by setting up a local server on Raspberry Pi. A LCD display is attached to Raspberry Pi using HDMI interface. The server continuously listens for a incoming message from a client which is an Android Based Smart phone with a dedicated application running on android OS. The user needs to enter sever IP address and port no in order to connect with the server. Once the connection is established between the clinet and server they can send messages to each other over TCP sockets. The server receives the message stores it in a temporary buffer. A secured script reads the messages stored in the buffer and displays it on the LCD monitor using HDMI interface.

2.2 Basic Block Diagram:



2.3 LIST OF COMPONENTS:

2.3.3 Hardware Used

- Raspberry Pi 2 (ARM7 Based Control Board)
- 5V 2A AC-DC adaptor - to power Raspberry Pi via micro USB
- Micro SD Card (At least 8GB) – a micro SD Card is used to store OS and other important files
- Wi-Fi adapter – A high speed USB 2.0 WIFI support 2.4G following IEEE 802.11n standard.
- LCD/LED screen with HDMI support.
- HDMI cable – to interface Raspberry Pi with LCD.
- Android based Smart Phone

2.4 Software Used

1. **Raspbian OS**

Raspbian is a free operating system based on Debian optimized for the Raspberry Pi hardware. An operating system is the set of basic programs and utilities that make your Raspberry Pi run. However, Raspbian provides more than a pure OS: it comes with over 35,000 packages, pre-compiled software bundled in a nice format for easy installation on your Raspberry Pi.

2. **Apache based web server**

   The **Apache Software Foundation** (**ASF**) is an American non-profit corporation (classified as 501(c)(3) in the United States) to support Apache software projects, including the Apache HTTP Server.

3. **Openvg**

OpenVG is an API designed for hardware-accelerated 2D vector graphics. Its primary platforms are mobile phones, gaming & media consoles and consumer electronic devices. It was designed to help manufacturers create more attractive user interfaces by offloading computationally intensive graphics processing from the CPU onto a GPU to save energy. OpenVG is well suited to accelerating Flash and mobile profile of SVG sequences. The OpenGL ES library provides similar functionality for 3D graphics. OpenVG is managed by the non-profit technology consortium Khronos Group.

CHAPTER 3

LITERATURE SURVEY

3.1 TCP/IP Sockets

The **Transmission Control Protocol** (**TCP**) is a core protocol of the Internet protocol suite. It originated in the initial network implementation in which it complemented the Internet Protocol (IP). Therefore, the entire suite is commonly referred to as *TCP/IP*. TCP provides reliable, ordered, and error-checked delivery of a stream of octets between applications running on hosts communicating over an IP network. Major Internet applications such as the World Wide Web, email, remote administration and file transfer rely on TCP. Applications that do not require reliable data stream service may use the User Datagram Protocol (UDP), which provides a connectionless datagram service that emphasizes reduced latency over reliability.

TCP abstracts the application's communication from the underlying networking details.

TCP is utilized extensively by many popular applications carried on the Internet, including the World Wide Web (WWW), E-mail, File Transfer Protocol, Secure Shell, peer-to-peer file sharing, and many streaming media applications.

3.1.1 TCP/IP Connection Establishment and termination

Process when transmitting device establishes a connection-oriented session with remote peer is called a **three-way handshake**. As the result end-to-end virtual (logical) circuit is created where flow controls and acknowledgment for reliable delivery is used. TCP has several message types used in connection establishment and termination process

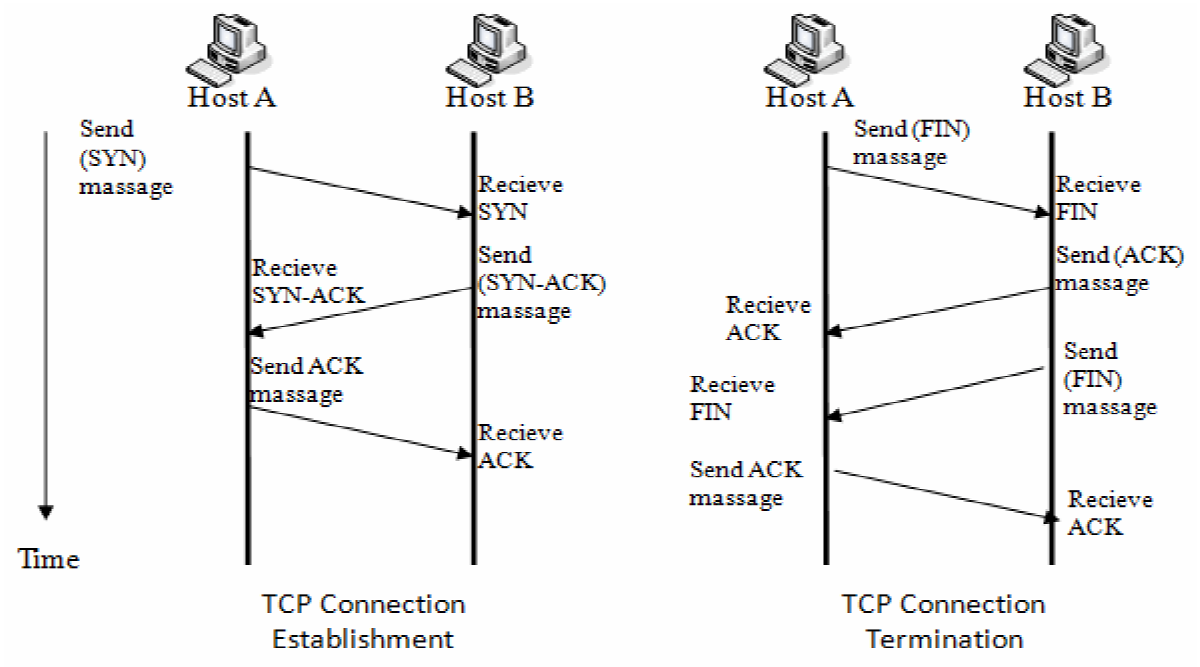


Figure 2.1. TCP session establishment and termination

Host A can be considered as server side setup whereas Host B is client side setup

Connection establishment steps:

- The host A who needs to initialize a connection sends out a SYN (Synchronize) packet with proposed initial sequence number to the destination host B.
- When the host B receives SYN message, it returns a packet with both SYN and ACK fags set in the TCP header (SYN-ACK).
- When the host A receives the SYN-ACK, it sends back ACK (Acknowledgment) macket
- Host B receives ACK and at this stage the connection is ESTABLISHED.

Connection-oriented protocol services are often sending acknowledgments (ACKs) after successful delivery. After packet with data is transmitted, sender waits acknowledgement from receiver. If time expires and sender did not receive ACK, packet is retransmitted.

Steps for connection Termination

When the data transmission is complete and the host wants to terminate the connection, termination process is initiated. Unlike TCP Connection establishment, which uses three-way handshake, connection termination uses four-way massages. Connection is terminated when both sides have finished the shut down procedure by sending a FIN and receiving an ACK.

- The host A, who needs to terminate the connection, sends a special message with the FIN (finish) flag, indicating that it has finished sending the data.
- The host B, who receives the FIN segment, does not terminate the connection but enters into a "passive close" (CLOSE_WAIT) state and sends the ACK for the FIN back to the host A. Now the host B enters into LAST_ACK state. At this point host B will no longer accept data from host A, but can continue transmit data to host A. If host B does not have any data to transmit to the host A it will also terminate the connection by sending FIN segment.
- When the host A receives the last ACK from the host B, it enters into a (TIME_WAIT) state, and sends an ACK back to the host B.
- Host B gets the ACK from the host A and closes the connection.

3.3.1What is a socket

Sockets are communication points on the same or different computers to exchange data. Sockets are supported by Unix, Windows, Mac, and many other operating systems. The tutorial provides a strong foundation by covering basic topics such as network addresses, host names, architecture, ports and services before moving into network address functions and explaining how to write client/server codes using sockets.

Sockets allow communication between two different processes on the same or different machines. To be more precise, it's a way to talk to other computers using standard Unix file descriptors. In Unix, every I/O action is done by writing or reading a file descriptor. A file descriptor is just an integer associated with an open file and it can be a network connection, a text file, a terminal, or something else.

To a programmer, a socket looks and behaves much like a low-level file descriptor. This is because commands such as read() and write() work with sockets in the same way they do with files and pipes.

Sockets were first introduced in 2.1BSD and subsequently refined into their current form with 4.2BSD. The sockets feature is now available with most current UNIX system releases.

3.1.2 Where are sockets used

A Unix Socket is used in a client-server application framework. A server is a process that performs some functions on request from a client. Most of the application-level protocols like FTP, SMTP, and POP3 make use of sockets to establish connection between client and server and then for exchanging data.

3.1.3 Types of sockets

There are four types of sockets available to the users. The first two are most commonly used and the last two are rarely used.

Processes are presumed to communicate only between sockets of the same type but there is no restriction that prevents communication between sockets of different types.

- **Stream Sockets** − Delivery in a networked environment is guaranteed. If you send through the stream socket three items "A, B, C", they will arrive in the same order − "A, B, C". These sockets use TCP (Transmission Control Protocol) for data transmission. If delivery is impossible, the sender receives an error indicator. Data records do not have any boundaries.
- **Datagram Sockets** − Delivery in a networked environment is not guaranteed. They're connectionless because you don't need to have an open connection as in Stream Sockets − you build a packet with the destination information and send it out. They use UDP (User Datagram Protocol).
- **Raw Sockets** − These provide users access to the underlying communication protocols, which support socket abstractions. These sockets are normally datagram oriented, though their exact characteristics are dependent on the interface provided by the protocol. Raw sockets are not intended for the general user; they have been provided mainly for those interested in developing new communication protocols, or for gaining access to some of the more cryptic facilities of an existing protocol.
- **Sequenced Packet Sockets** − They are similar to a stream socket, with the exception that record boundaries are preserved. This interface is provided only as a part of the Network Systems (NS) socket abstraction, and is very important in most serious NS applications. Sequenced-packet sockets allow the user to manipulate the Sequence Packet Protocol (SPP) or Internet Datagram Protocol (IDP) headers on a packet or a group of packets, either by writing a prototype header along with whatever data is to be sent, or by specifying a default header to be used with all outgoing data, and allows the user to receive the headers on incoming packets.

3.1.4 Client Process

This is the process, which typically makes a request for information. After getting the response, this process may terminate or may do some other processing.

Example, Internet Browser works as a client application, which sends a request to the Web Server to get one HTML webpage.

3.1.5 Server Process

This is the process which takes a request from the clients. After getting a request from the client, this process will perform the required processing, gather the requested information, and send it to the requestor client. Once done, it becomes ready to serve another client. Server processes are always alert and ready to serve incoming requests.

Example − Web Server keeps waiting for requests from Internet Browsers and as soon as it gets any request from a browser, it picks up a requested HTML page and sends it back to that Browser.

Note that the client needs to know the address of the server, but the server does not need to know the address or even the existence of the client prior to the connection being established. Once a connection is established, both sides can send and receive information.

3.1.6 Structure used in sockets

Various structures are used in Unix Socket Programming to hold information about the address and port, and other information. Most socket functions require a pointer to a socket address structure as an argument. Structures defined in this chapter are related to Internet Protocol Family.

sockaddr

The first structure is *sockaddr* that holds the socket information −

```
struct sockaddr {
   unsigned short   sa_family;
   char             sa_data[14];
};
```

This is a generic socket address structure, which will be passed in most of the socket function calls. The following table provides a description of the member fields −

Attribute	Values	Description
sa_family	AF_INET AF_UNIX AF_NS AF_IMPLINK	It represents an address family. In most of the Internet-based applications, we use AF_INET.

sa_data	Protocol-specific Address	The content of the 14 bytes of protocol specific address are interpreted according to the type of address. For the Internet family, we will use port number IP address, which is represented by *sockaddr_in* structure defined below.

sockaddr in

The second structure that helps you to reference to the socket's elements is as follows −

```
struct sockaddr_in {
   short int            sin_family;
   unsigned short int   sin_port;
   struct in_addr       sin_addr;
   unsigned char        sin_zero[8];
};
```

Attribute	Values	Description
sa_family	AF_INET AF_UNIX AF_NS AF_IMPLINK	It represents an address family. In most of the Internet-based applications, we use AF_INET.
sin_port	Service Port	A 16-bit port number in Network Byte Order.
sin_addr	IP Address	A 32-bit IP address in Network Byte Order.
sin_zero	Not Used	You just set this value to NULL as this is not being used.

3.1.7 Steps to create TCP Server

- Create a socket with the **socket()** system call.
- Bind the socket to an address using the **bind()** system call. For a server socket on the Internet, an address consists of a port number on the host machine.
- Listen for connections with the **listen()** system call.
- Accept a connection with the **accept()** system call. This call typically blocks until a client connects with the server.
- Send and receive data using the **read()** and **write()** system calls.

3.1.8 Steps to create TCP Client

To make a process a TCP client, you need to follow the steps given below &minus ;

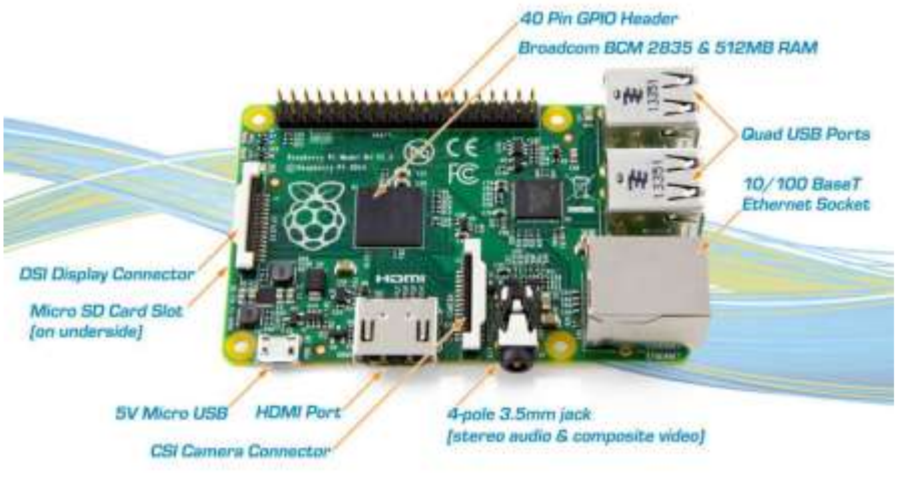
- Create a socket with the **socket()** system call.
- Connect the socket to the address of the server using the **connect()** system call.
- Send and receive data. There are a number of ways to do this, but the simplest way is to use the **read()** and **write()** system calls.

Now let us put these steps in the form of source code. Put this code into the file **client.c** and compile it with **gcc** compiler.

3.2 RASPBEERY PI 2

Raspberry Pi can be configured TCP server to listen for incoming calls.

The **Raspberry Pi** is a series of credit card–sized single-board computers developed in England, United Kingdom by the Raspberry Pi Foundation with the intent to promote the teaching of basic computer science in schools and developing countries.


40 Pin GPIO Header
Broadcom BCM 2835 & 512MB RAM
Quad USB Ports
10/100 BaseT Ethernet Socket
DSI Display Connector
Micro SD Card Slot [on underside]
5V Micro USB
HDMI Port
CSI Camera Connector
4-pole 3.5mm jack [stereo audio & composite video]

Raspberry Pi and Android phone talks to each other via router. Router allots a IP address to Raspberry Pi and TCP IP Server is established which continuously listens for incoming client connection.

Android Phone connects to server IP address and port and when the connection is established the data is transmitted from client to server.

The messages received by server is then displayed on LCD monitor interfaced to Raspberry Pi over HDMI

SD card consist of OS (Raspbian) and also used to store other data.

A wifi adaptor is used which is connected to raspberry Pi over USB.

Features of Raspberry Pi2

- Broadcom BCM2836 Arm7 Quad Core Processor powered Single Board Computer running at 900MHz
- 1GB RAM which can run bigger and powerful program.
- 40pin extended GPIO.
- 4 x USB 2 ports
- 4 pole Stereo output and Composite video port.
- Full size HDMI- stream and watch HD videos.
- DSI display port for connecting the Raspberry Pi touch screen display.
- Micro SD port for loading your operating system and storing data
- Micro USB power source.
- Supports Raspbian (Debian based) Operating System.
- 10/100 Ethernet Port to quickly connect the Raspberry Pi to the Internet

3.4 Android Phone

In this Project android phone is used as client which sends text data to server from a remote place using TCP/IP Sockets

Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics.The android OS supports TCP/IP socket, it can act as a TCP server as well as TCP client.

3.5 CHROMIUM BROWSER

Chromium browser is used as to display web pages. The web page is to be displayed in kiosk mode. he pi would display a full screen webpage with session details taken from our database. In the event someone power cycled the Pi we wanted it to boot straight into this full screen state and straight onto the webpage we'd setup.

3.6 Android Studio

Android studio is used to integrated development (IDE) for android platform development. The IDE is developed by google. IDE is used to develop application used to configure as a TCP client. The google has provided standard API to communicate over TCP/IP.

3.7 Gaps in Literature

The communication between server and client is obtained using Ethernet connection in order to achieve wireless connection proper configuration is to be achieved. The message sent by the user is displayed in kiosk mode. The client is to be setup on android based phone. Hands on experience in android studio is required in order to develop application which can talk to server over TCP client.

A web page which reads message sent from client is to be opened in a web browser in kiosk mode which is some of the minor reviews which are to be understood and implemented.

3.8 Methods and Methodology

1. Client : User Mobile (Android Powered Mobile Phone)
2. Server : Raspberry Pi
3. Raspberry Pi connects to Router using a Wi-Fi adaptor
4. User enters SSID (Router Name) and Password of Router.
5. Router assigns IP address to Raspberry Pi.
6. TCP Server is created on Raspberry Pi which listens for incoming calls
7. A TCP client is created on Android which connects to TCP server.
8. When a connection is established the client sends message to server.
9. The message sent to by the client is stored in a text file on raspberry pi hard disk (SD card).
10. The text file is read by another Program which is displays the text on LCD/LED monitor connected on HDMI Interface.

CHAPTER 4

INSTALLING OS IN RASPBERRY PI

In this chapter we will discuss about installing an operating system on Raspberry Pi. Raspberry Pi supports more than 15 OS, Raspbian OS is the most popular application therefore we will are using Raspbian OS in our Project.

4.1 Raspbian

Raspbian is a free operating system based on Debian optimized for the Raspberry Pi hardware. An operating system is the set of basic programs and utilities that make your Raspberry Pi run. However, Raspbian provides more than a pure OS: it comes with over 35,000 packages, pre-compiled software bundled in a nice format for easy installation on your Raspberry Pi.

The initial build of over 35,000 Raspbian packages, optimized for best performance on the Raspberry Pi, was completed in June of 2012. However, Raspbian is still under active development with an emphasis on improving the stability and performance of as many Debian packages as possible.

4.2 Required Tools

Following are the bare minimum required to install Raspbian OS on raspberry Pi

- An SD card 4GB
- A PC with Windows operating System.
- The Raspbian Wheezy .IMG file.
- Raspberry Pi 2

4.3 Downloading Raspbian

Raspbian Wheezy .IMG file can be downloaded from the official Raspberry Pi website for free of cost. Visit the following link: https://www.raspberrypi.org/downloads/

Under the header "Raspbian 'wheezy'", download either the torrent or direct download. The torrent has the potential to be faster, but some firewalls may block the required ports and you may have to use the direct download instead.

Once you have the ZIP file downloaded to your computer, unarchive it. There will be a single .img file inside. This is the disk image you will flash to the Raspberry Pi's SD card. To install Raspbian, you will need an SD card that has 2 GB of space or more– this cheap 16 GB Class 10 SD card works great on the Raspberry Pi, and gives you plenty of room to add media and other programs once Raspbian is installed.

4.3 Flashing SD Card in Windows

The recommended method for flashing an SD for use in a Raspberry Pi is a program called Win32DiskImager. The latest version can be found at https://launchpad.net/win32-image-writer/+download . Personally I recommend version 0.5, since the latest version (0.6) has a bug that may cause your entire hard drive to be flashed instead of the SD card.

Once you've downloaded the Win32DiskImager application and extracted the ZIP file, download the Raspbian distribution. Once the ZIP file downloads, extract the .img from the .zip.

In Win32DiskImager, ensure you select the correct drive letter for your SD card. In my case, the SD card was drive F:/. Yours may be different, so check in Windows Explorer to make sure you have the correct letter. Do not choose C:\, since that is your main hard drive.

Also, select the .img file you extracted from the Raspbian distribution above using the file picker. Once you have made sure you have the correct .img file and drive letter for your SD card, click "Write" (not read) to flash the SD card. This will take less than five minutes on average and you can see the current progress in the Win32DiskImager window. Once the flash completes, you can exit the program.

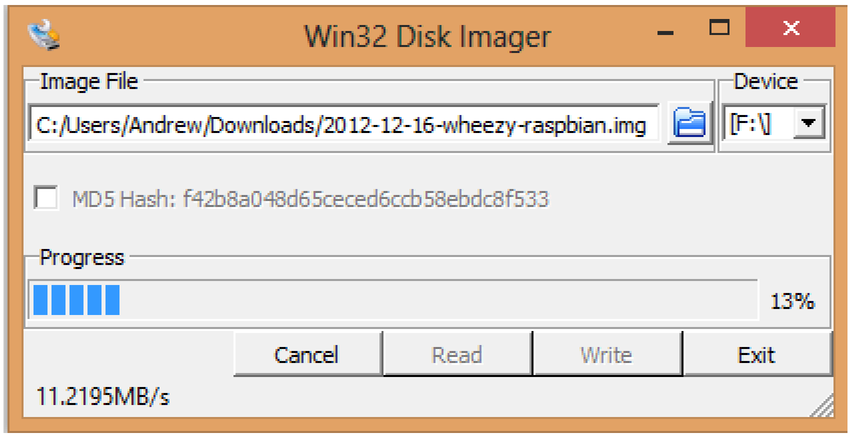

Win32 Disk Imager

Image File: C:/Users/Andrew/Downloads/2012-12-16-wheezy-raspbian.img

Device: [F:\]

MD5 Hash: f42b8a048d65ceced6ccb58ebdc8f533

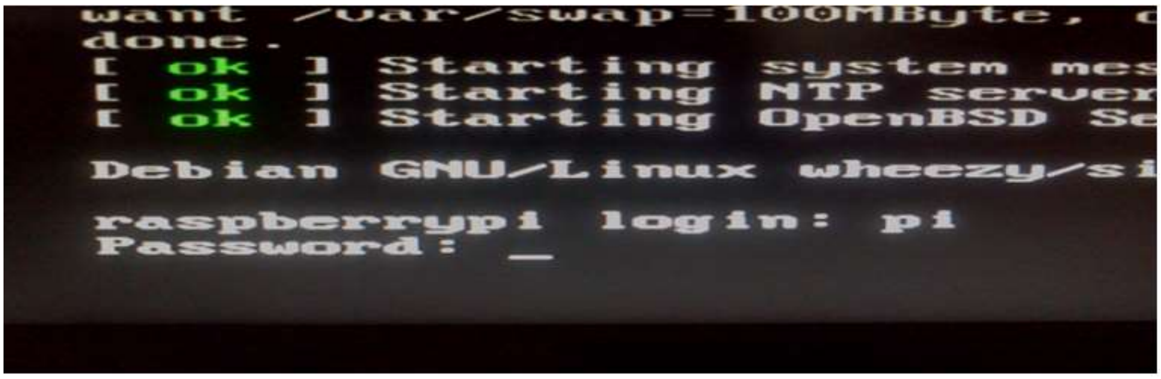
Progress: 13%

11.2195MB/s

4.4 Setting up Raspberry Pi

Once we have flashed the disk image using the methods above, place the SD card into your Raspberry Pi, plug in the HDMI monitor, any keyboards and mice, and then the power cable. Your Raspberry Pi should now begin to boot and you should be able to see Raspbian on your screen.

Following screen appears on the LCD screen


```
want /var/swap=100MByte, c
done.
[ ok ] Starting system mes
[ ok ] Starting NTP server
[ ok ] Starting OpenBSD Se

Debian GNU/Linux wheezy/si

raspberrypi login: pi
Password: _
```

The Raspberry Pi boots and asks for a user name and password which is pi and raspberry respectively by default.

4.5 Wi-Fi on Raspberry Pi

Raspberry pi doesn't have a inbuilt Wi-Fi therefore we have to connect a external Wi-Fi to our pi to access Raspberry pi wireless. For our project we have used TL-Wn725n Wi-Fi adapter.

4.5.1 TL-WN725n

Hardware Features

Interface	USB 2.0
Dimensions	0.73x0.59x0.28in.(18.6x15x7.1mm)
Antenna	Internal antenna
LED	Status
Weight	2.2grams

Wireless Features

Wireless Standards	IEEE 802.11b, IEEE 802.11g, IEEE 802.11n
Frequency	2.400-2.4835GHz
Signal Rate	11b: Up to 11Mbps (dynamic) 11g: Up to 54Mbps (dynamic) 11n: Up to 150Mbps (dynamic)
Reception Sensitivity	130M: -68dBm@10% PER 108M: -68dBm@10% PER 54M: -68dBm@10% PER

	11M: -85dBm@8% PER 6M: -88dBm@10% PER 1M: -90dBm@8% PER
Transmit Power	<20dBm
Wireless Modes	Ad-Hoc / Infrastructure mode
Wireless Security	Supports 64/128 WEP, WPA/WPA2, WPA-PSK/WPA2-PSK (TKIP/AES), supports IEEE 802.1X
Modulation Technology	DBPSK, DQPSK, CCK, OFDM, 16-QAM, 64-QAM

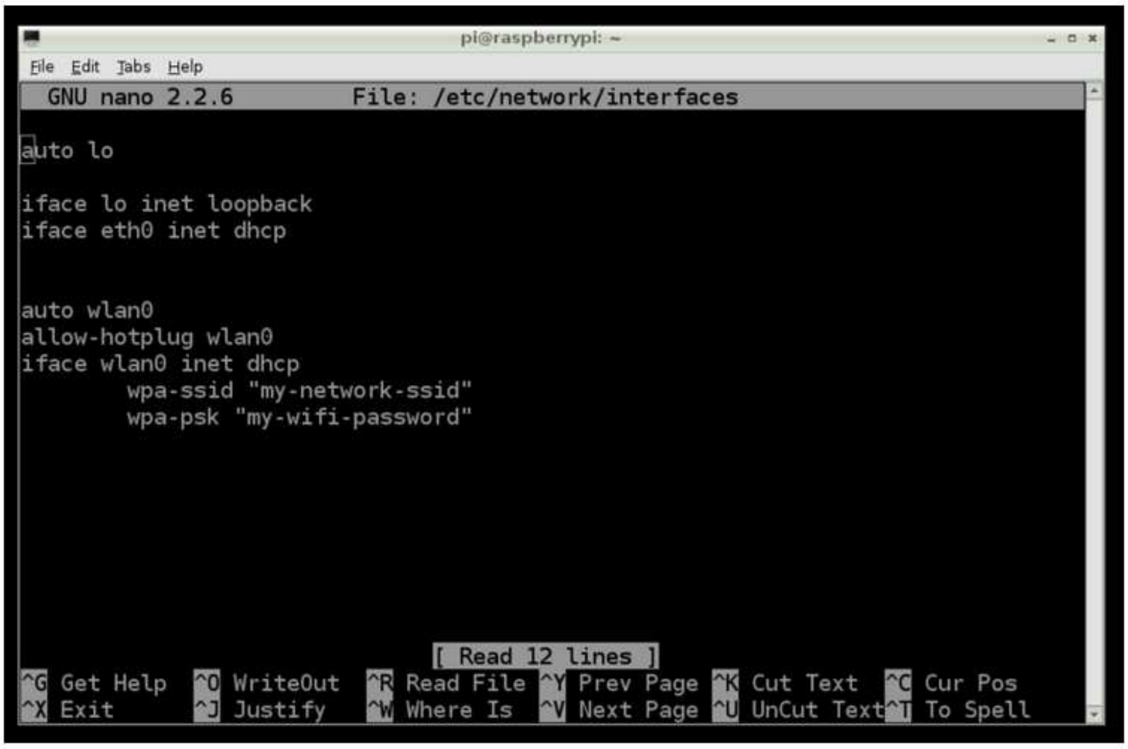
4.5.2 Setting up Wi-Fi through command line

Open a LX terminal after the pi boots up and plug in the Wi-Fi adapter in to one of the USB port of Raspberry Pi

Type the following command on the terminal

sudo nano /etc/network/interfaces

This opens the editor screen of the Wi-Fi configuration file you need to change



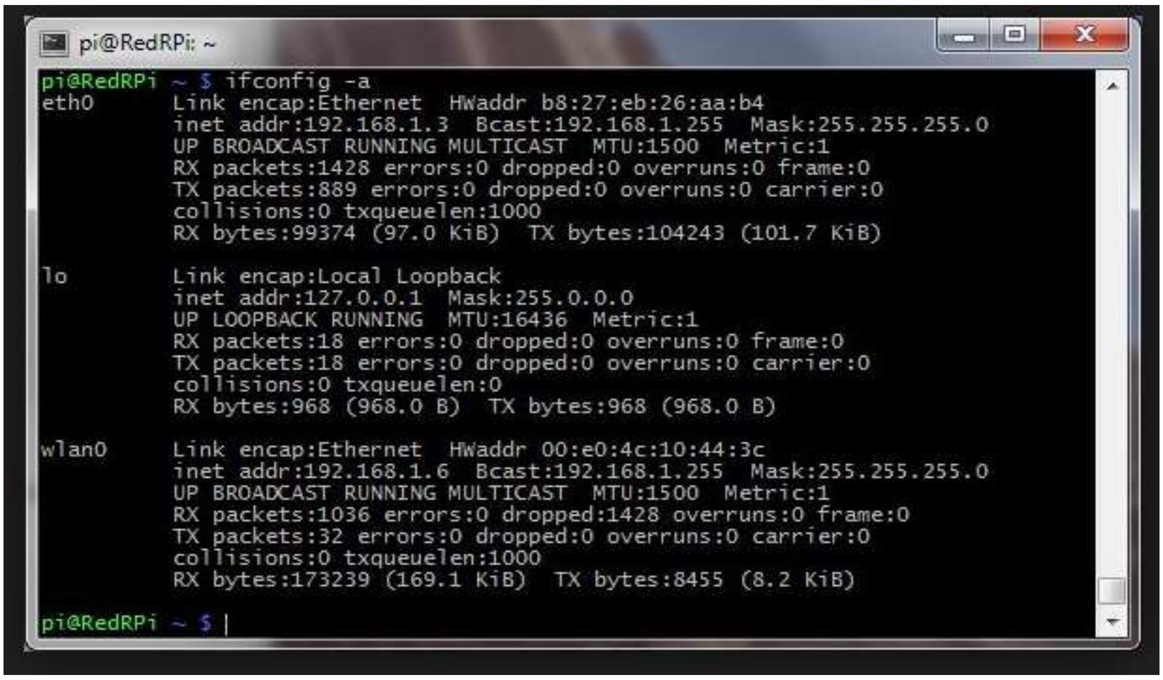
```
  GNU nano 2.2.6           File: /etc/network/interfaces

auto lo

iface lo inet loopback
iface eth0 inet dhcp


auto wlan0
allow-hotplug wlan0
iface wlan0 inet dhcp
        wpa-ssid "my-network-ssid"
        wpa-psk "my-wifi-password"






                           [ Read 12 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
```

Now enter Wi-Fi ssid and password of your router under the doule quotes wpa-ssid and wpa-psk respectively. Once the file is edited presss ctrl-X to sve and reboot pi to connect pi to your router.

To check whether pi is connected to router through Wi-Fi type the following command on LXterminal

Ifconfig

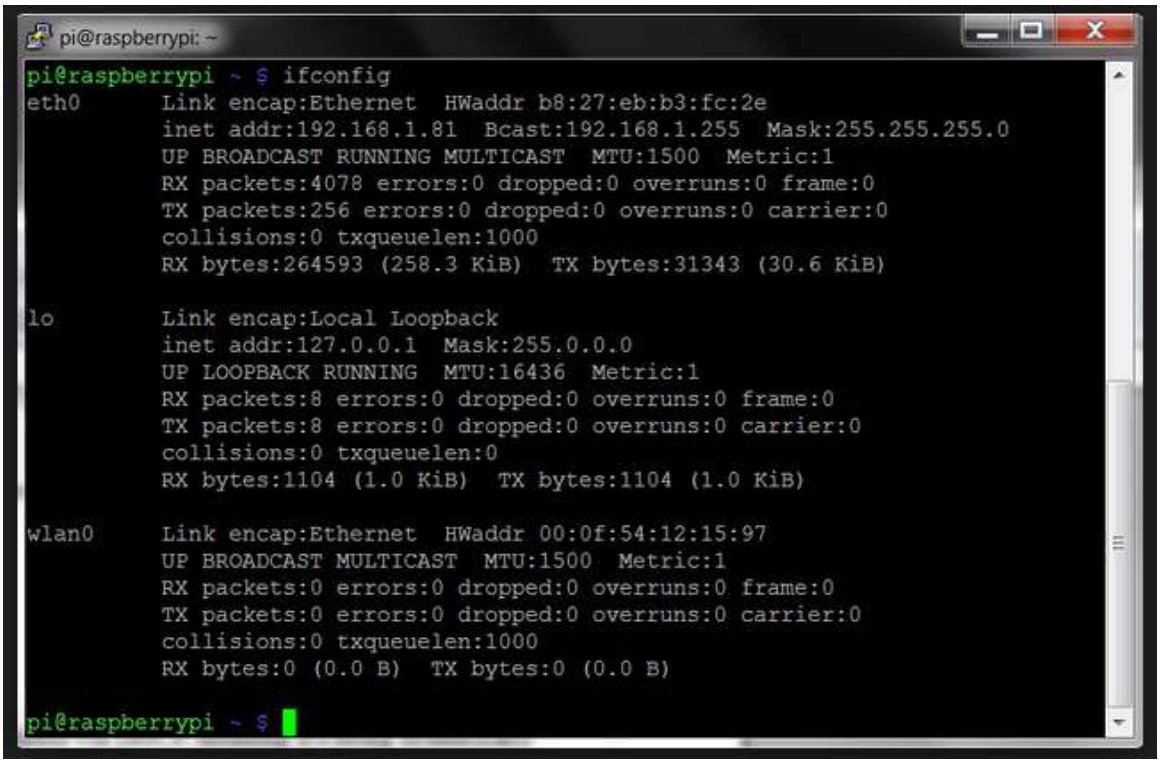
Hit enter and you will see a the screen shown below



```
pi@RedRPi ~ $ ifconfig -a
eth0      Link encap:Ethernet  HWaddr b8:27:eb:26:aa:b4
          inet addr:192.168.1.3  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1428 errors:0 dropped:0 overruns:0 frame:0
          TX packets:889 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:99374 (97.0 KiB)  TX bytes:104243 (101.7 KiB)

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:18 errors:0 dropped:0 overruns:0 frame:0
          TX packets:18 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:968 (968.0 B)  TX bytes:968 (968.0 B)

wlan0     Link encap:Ethernet  HWaddr 00:e0:4c:10:44:3c
          inet addr:192.168.1.6  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1036 errors:0 dropped:1428 overruns:0 frame:0
          TX packets:32 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:173239 (169.1 KiB)  TX bytes:8455 (8.2 KiB)

pi@RedRPi ~ $ |
```

Now if the connection is successfully established you will see a ip address under wlan0 i.e inet addr: ip_addr of pi allotted by the router.

If the connection is not established between your pi and router through Wi-Fi then you will see a screen as shown below


pi@raspberrypi: ~

```
pi@raspberrypi ~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:b3:fc:2e
          inet addr:192.168.1.81  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4078 errors:0 dropped:0 overruns:0 frame:0
          TX packets:256 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:264593 (258.3 KiB)  TX bytes:31343 (30.6 KiB)

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:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1104 (1.0 KiB)  TX bytes:1104 (1.0 KiB)

wlan0     Link encap:Ethernet  HWaddr 00:0f:54:12:15:97
          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)

pi@raspberrypi ~ $
```

CHAPTER 5

SETTING LAMP SERVER

5.1 LAMP SERVER (Linux, Apache, MySQL, PHP)

LAMP is an open source Web development platform that uses Linux as the operating system, Apache as the Web server, MySQL as the relational database management system and PHP as the object-oriented scripting language. The LAMP components are largely interchangeable and not limited to the original selection. As a solution stack, LAMP is suitable for building dynamic web sites and web applications.

5.1.1 Linux

Linux is a Unix-like computer operating system assembled under the model of free and open source software development and distribution. Most Linux distributions, as collections of software based around the Linux kernel and often around a package management system, provide complete LAMP setups through their packages. According to W3Techs in October 2013, 58.5% of web server market share was shared between Debian and Ubuntu, while RHEL, Fedora and centOS together shared 37.3%. In this project we will use Raspbian OS which is a flavor of linux suitable for Raspberry Pi.

5.1.2 Apache

The role of LAMP's web server has been traditionally supplied by Apache, and has since included other web servers such as Nginx.

The Apache HTTP Server has been the most popular web server on the public Internet. In June 2013, Netcraft estimated that Apache served 54.2% of all active websites and 53.3% of the top servers across all domains. In June 2014, Apache was estimated to serve 52.27% of all active websites, followed by nginx with 14.36%.

Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. Released under the Apache License, Apache is open-source software. A wide variety of features are supported, and many of them are implemented as compiled modules which extend the core functionality of Apache. These can range from server-side programming language support to authentication schemes.

5.1.3 PHP

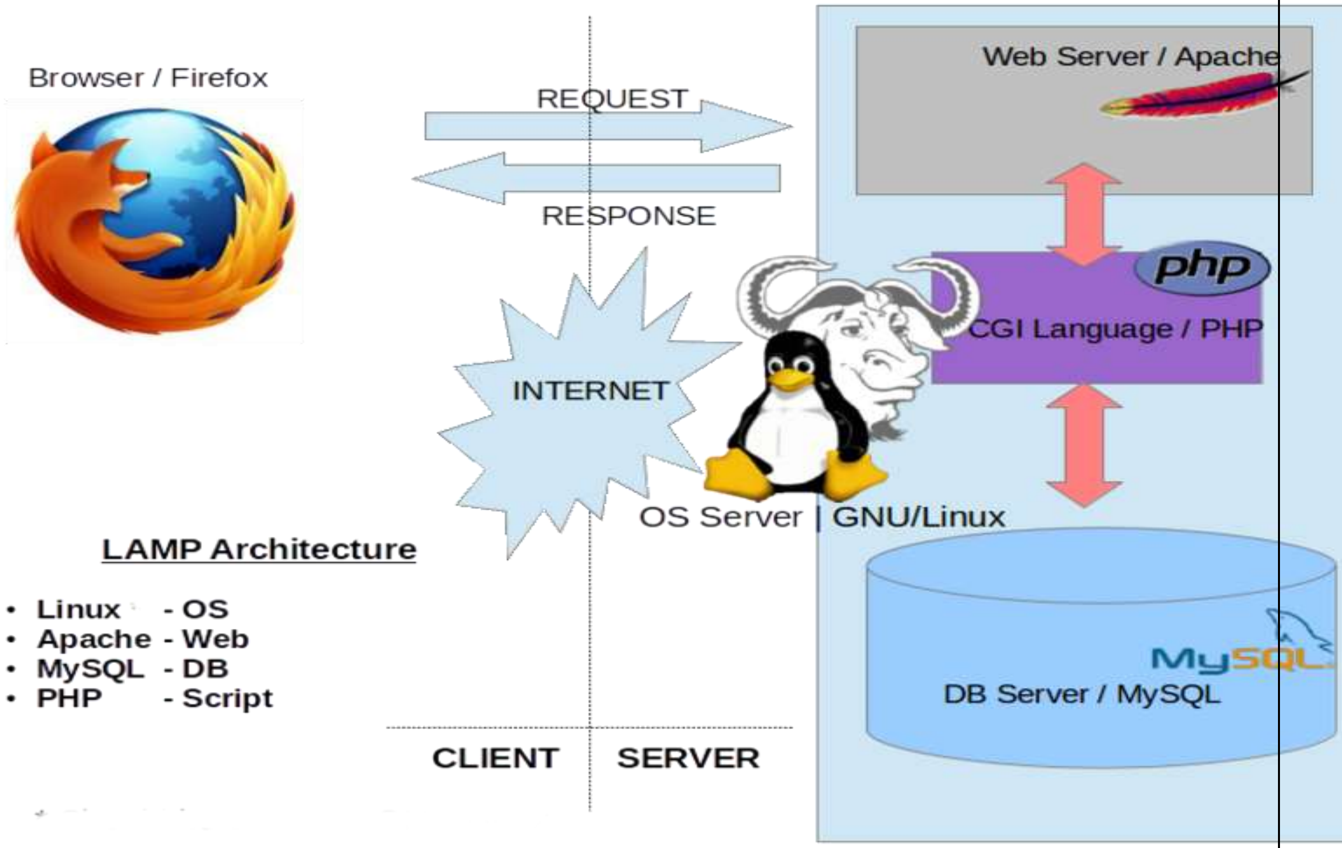
PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. Originally created by Rasmus Lerdorf in 1994, the PHP reference implementation is now produced by The PHP Group.PHP originally stood for Personal Home Page,[3] but it now stands for the recursive backronym PHP: Hypertext Preprocessor.

PHP code may be embedded into HTML code, or it can be used in combination with various web template systems, web content management system and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications.

The standard PHP interpreter, powered by the Zend Engine, is free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers on almost every operating system and platform, free of charge.

The PHP language evolved without a written formal specification or standard until 2014, leaving the canonical PHP interpreter as a de facto standard. Since 2014 work has gone on to create a formal PHP specification.

5.1.4 High Level overview of LAMP



Note:

MySQL is beyond the scope of this project.

5.2 Installing Apache Web server on Raspbian

The Apache web server is currently the most popular web server in the world, which makes it a great default choice for hosting a website. Raspbian OS is based on Linux OS therefore we can follow the steps.

We can install Apache easily using Linux package manager, apt. A package manager allows us to install most software pain-free from a repository maintained by Linux. You can learn more about how to use apt here.

Open terminal and type the command shown below

```
sudo apt-get install apache2
```

Since we are using a `sudo` command, these operations get executed with root privileges. It will ask you for your regular user's password to verify your intentions.

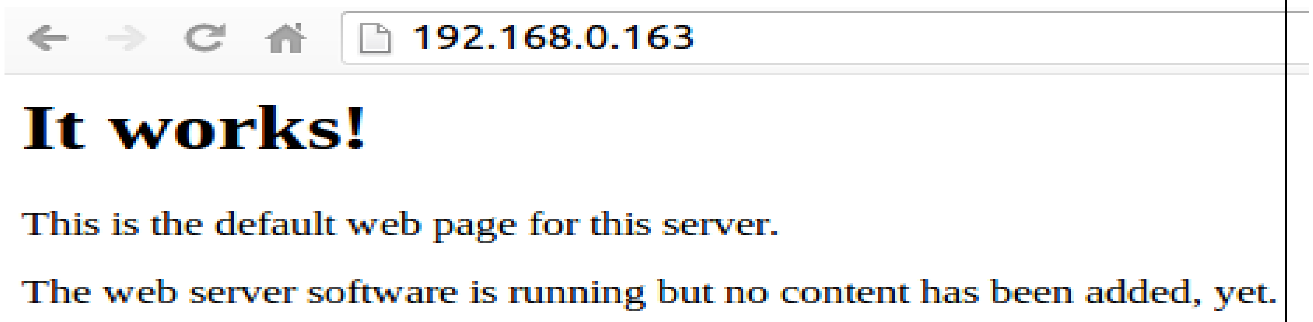
You can do a spot check right away to verify that everything went as planned by visiting your server's public IP address in your web browser (see the note under the next heading to find out what your public IP address is if you do not have this information already):

```
http://your_server_IP_address
```

or

http://localhost

you should see the following on web bowser:


192.168.0.163

It works!

This is the default web page for this server.

The web server software is running but no content has been added, yet.

5.2.1 Changing the default web page

This default web page is just a HTML file on the filesystem. It is located at /var/www/html/index.html.

Navigate to this directory in the Terminal and have a look at what's inside:

```
cd /var/www/html
ls –al
```

5.3 Installing PHP

PHP is a preprocessor; it's code that runs when the server receives a request for a web page. It runs, works out what needs to be shown on the page, then sends that page to the browser. Unlike static HTML, PHP can show different content under different circumstances. Other languages are capable of this, but since WordPress is written in PHP, that's what we need to use this time. PHP is a very popular language on the web; large projects like Facebook and Wikipedia are written in PHP.

Install the PHP and Apache packages with the following command:

```
sudo apt-get install php5 libapache2-mod-php5 -y
```

5.3.1 TEST PHP

Create the file index.php:

```
sudo nano index.php
```

Put some PHP content in it:

```
<?php echo "hello world"; ?>
```

Now save the file. Next delete index.html because it takes precendence over index.php:

```
sudo rm index.html
```

Refresh your browser. You should see "hello world". This is not dynamic but it is still served by PHP. If you see the raw PHP above instead of "hello world", reload and restart Apache like so:

```
sudo /etc/init.d/apache2 reload
sudo /etc/init.d/apache2 restart
```

Otherwise try something dynamic, for example:

```
<?php echo date('Y-m-d H:i:s'); ?>
```

Or show your PHP info:

```
<?php phpinfo(); ?>
```

CHAPTER 6

CHROMIUM BROWSER IN KIOSK MODE

In our project we want Raspberry pi to be hidden behind the screen with just a network cable and a HDMI cable coming out of it. The HDMI is connected to LCD screen to view text messages sent by Android phone. In this chapter we will discuss various steps followed to configure chromium browser in kiosk mode.

6.1 Purpose

The purpose is to open a browser and display a web page present in our browser to be displayed in a full screen mode. In the event someone power cycled the Pi we wanted it to boot straight into this full screen state and straight onto the webpage we'd setup. We would need to setup SSH to allow us remote access to the Pi too to avoid the need for a mouse and keyboard at any point.

6.2 Setting up raspberry Pi

Once we have a SD card with Raspbian installed on it. Insert the Sd card in Raspberry Pi power it up and lets the Pi boots up

Once this is done boot your Raspberry Pi and open up a terminal window by navigating to Start -> Application -> LXTerminal in GUI mode.

Type in

```
sudo apt-get update && sudo apt-get upgrade -y
```

These commands will update the package list and upgrade any packages on your device.

Next we install the chromium browser, x11 server utilities and unclutter(removes the cursor from the screen).

To do this type in

```
sudo apt-get install chromium x11-xserver-utils unclutter
```

The chromium browser includes a kiosk mode which displays the browser full screen without any taskbars or icons which works perfectly for a kiosk style screen.

Now that we have everything we need we can setup SSH to allow us remote access to the unit and get rid of the keyboard and mouse.

6.3 Direct SSH into Raspberry Pi in Headless mode

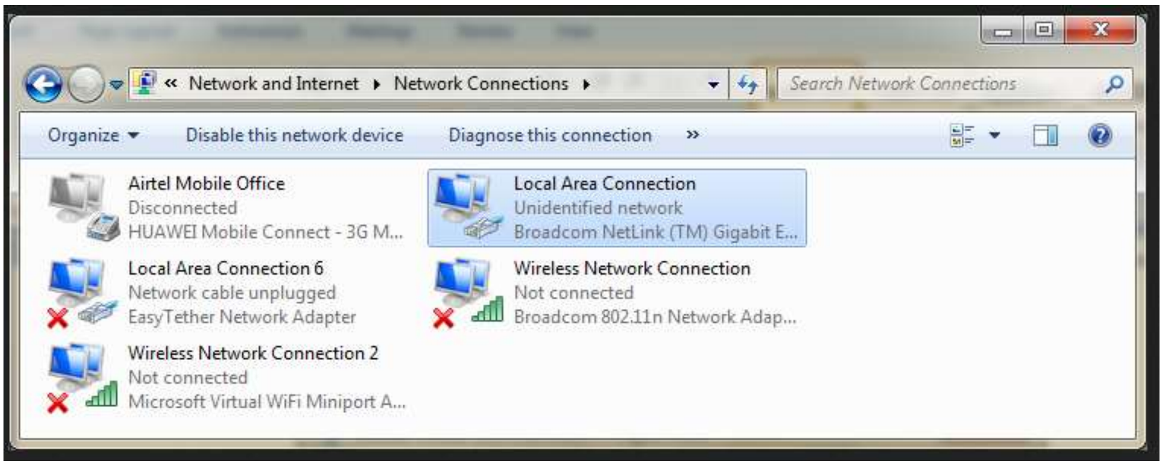
First we want to setup a static IP address for our Pi on the network. Make sure your pi is connected to your network and in a terminal window type

```
ifconfig
```

Note down the gateway and broadcast address and any other address settings you think you may need for your current connection.

Following are the steps to open raspberry pi in Headless mode

- While the Raspberry Pi is switched off, connect one side of the Ethernet cable to Raspberry Pi and other side to the RJ45 jack of the PC/Laptop
- Open LAN properties and make sure that IPV4 properties are set to Obtain IP address automatically as shown below:


« Network and Internet ▸ Network Connections ▸
Search Network Connections
Organize ▾
Disable this network device
Diagnose this connection
»

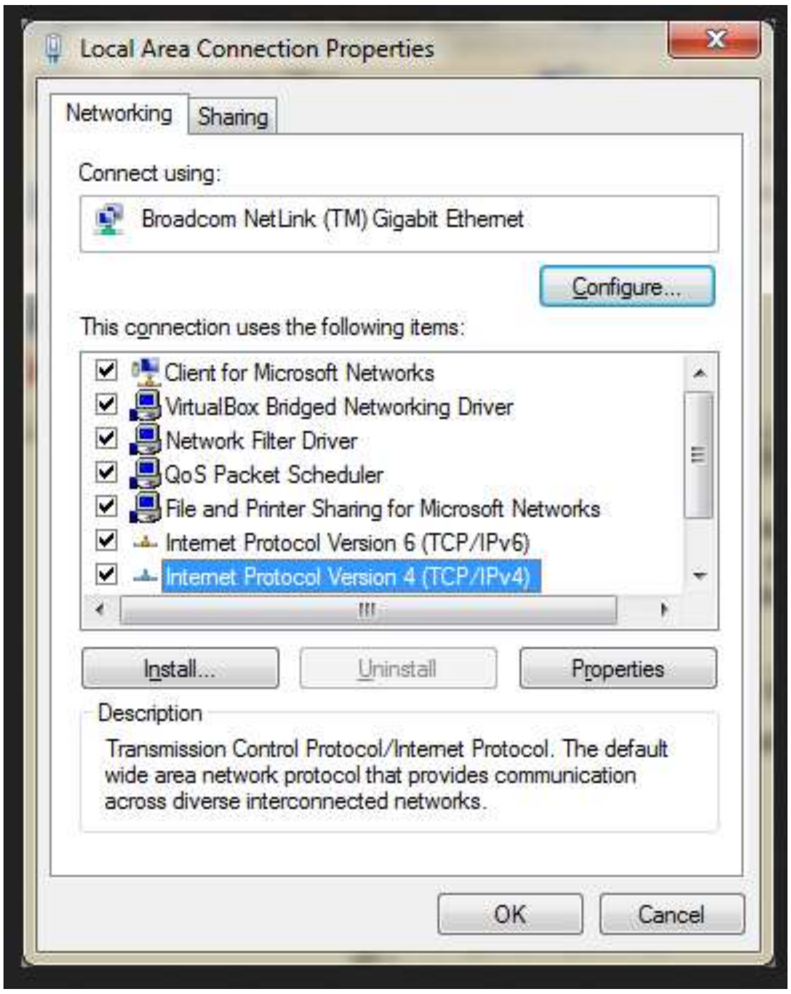
Airtel Mobile Office
Disconnected
HUAWEI Mobile Connect - 3G M...

Local Area Connection
Unidentified network
Broadcom NetLink (TM) Gigabit E...

Local Area Connection 6
Network cable unplugged
EasyTether Network Adapter

Wireless Network Connection
Not connected
Broadcom 802.11n Network Adap...

Wireless Network Connection 2
Not connected
Microsoft Virtual WiFi Miniport A...



Local Area Connection Properties

Networking | Sharing

Connect using:

Broadcom NetLink (TM) Gigabit Ethernet

Configure...

This connection uses the following items:

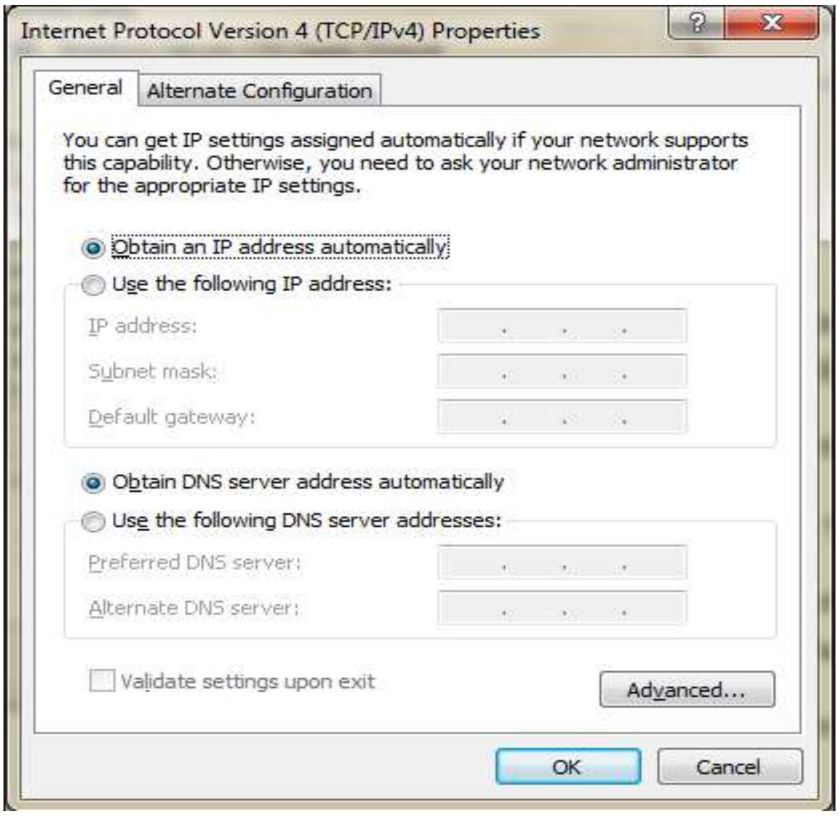
- [✓] Client for Microsoft Networks
- [✓] VirtualBox Bridged Networking Driver
- [✓] Network Filter Driver
- [✓] QoS Packet Scheduler
- [✓] File and Printer Sharing for Microsoft Networks
- [✓] Internet Protocol Version 6 (TCP/IPv6)
- [✓] Internet Protocol Version 4 (TCP/IPv4)

Install... | Uninstall | Properties

Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

OK | Cancel



Internet Protocol Version 4 (TCP/IPv4) Properties

General | Alternate Configuration

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

● Obtain an IP address automatically

○ Use the following IP address:

IP address: . . .

Subnet mask: . . .

Default gateway: . . .

● Obtain DNS server address automatically

○ Use the following DNS server addresses:

Preferred DNS server: . . .

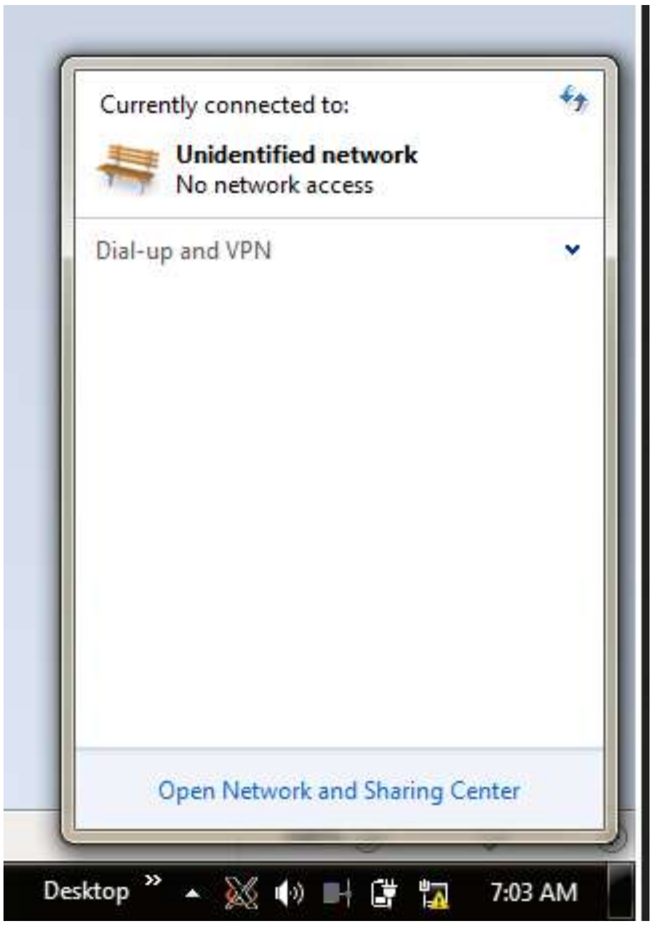
Alternate DNS server: . . .

☐ Validate settings upon exit

Advanced...

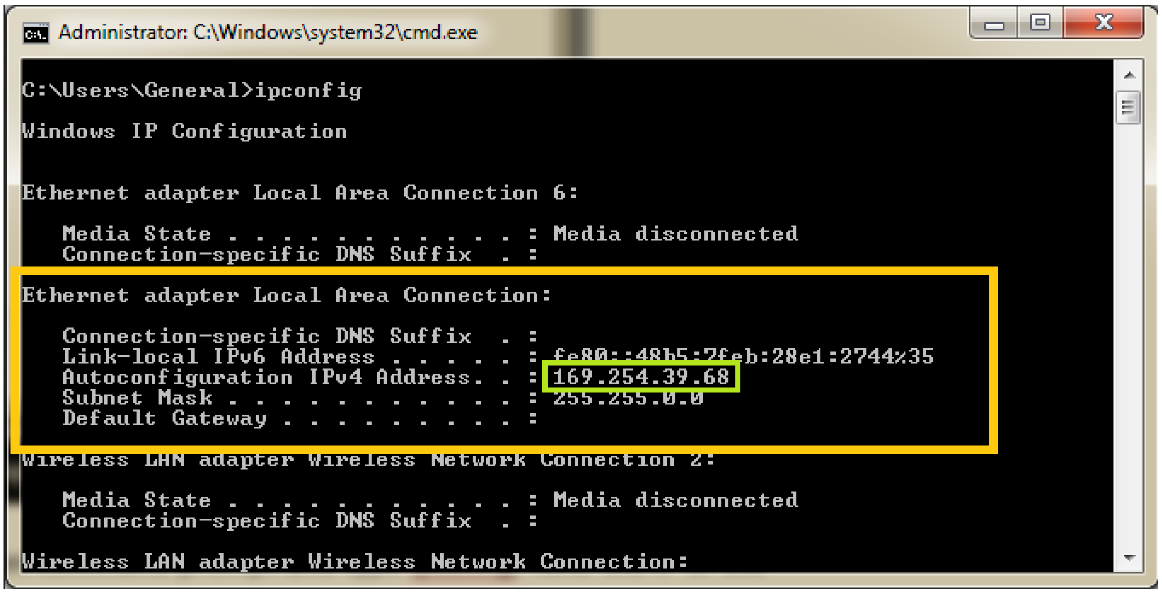
OK | Cancel

- We now need to determine the IP of our PC/Laptop when it's connected to the Raspberry Pi.
- Now power on the Raspberry Pi while making sure that the network cable is connected on both ends
- Wait for a min or two. You'll notice that the PC/Laptop will scan and then show a small warning indicating the presence of an unidentified network.



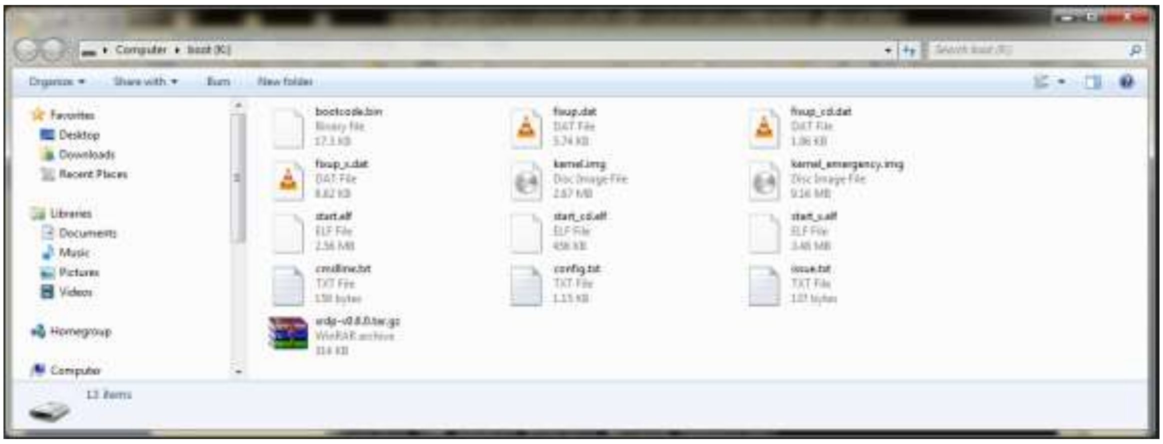
Currently connected to:
Unidentified network
No network access

- Now, open command prompt and type **ipconfig**. Note the IP of the Ethernet Adapter Local Area Connection as shown below:



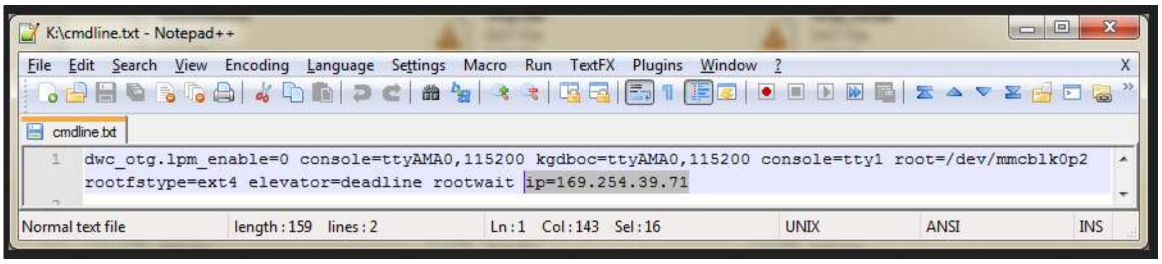
```
C:\Users\General>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection 6:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::48b5:7feb:28e1:2744%35
   Autoconfiguration IPv4 Address. . : 169.254.39.68
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Wireless Network Connection 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wireless Network Connection:
```

- Now, power off the Raspberry Pi and take out the SD Card. Plug in the SD card into a card reader and open it. You should see a couple of files



- Open the **cmdline.txt** file and append this to the end of it:

ip=169.254.39.71


K:\cmdline.txt - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
cmdline.txt

```
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2
rootfstype=ext4 elevator=deadline rootwait ip=169.254.39.71
```

Normal text file
length : 159 lines : 2
Ln : 1 Col : 143 Sel : 16
UNIX
ANSI
INS

We have to change the IP accordingly and assign a unique value (while making sure that we don't go beyond the subnet mask). Preferably, change only the last parameter

For example, if your LAN's ip is **169.254.0.1**, we recommend using **169.254.0.3** as the IP for Raspberry Pi in **cmdline.txt**

- Save the **cmdline.txt** file without making any other changes
- Plug this back into the Raspberry Pi and with the Ethernet cables connected, power on the Raspberry Pi.
- Wait a couple of minutes while the Raspberry Pi tries to establish a local network connection with our PC/Laptop.
- Once you see the network warning message as shown above, ping the Raspberry Pi to see if it's live on the network as follows

Before you go neck-deep into this post, please check if this post is applicable for you:

What you want to do:

1. Access Raspberry Pi's desktop on Laptop/PC screen

What you have:

1. Raspberry Pi A+/B/B+/Pi 2 (If you've a Pi Zero, please try and let us know if this works on a Zero. Not tested with a Pi 3 either) running the latest Raspbian Wheezy (I've not yet tried it with recently released Raspbian Jessie based off Debian 8. If you've been successful with this latest release, please post in the comment and I'll update it here. Thanks.)
2. Laptop (Windows 7 or XP with Internet connection, required to download one setup file)
3. A working SSH connection with Raspberry Pi (direct access, explained below)

What you don't have (and not required):

1. Display Device for the Raspberry Pi (HDMI enabled display unit / Old TV)
2. Internet connection for the Raspberry Pi (LAN or WiFi)

Extra Stuff you need:

1. Standard Network Cable (Cat 5, Standard or Crossover)
2. PutTTY Software (will be used to SSH into the Pi)
3. SD Card Writer

There are hundreds of tutorials sprawling across the web that show you how to remote access your Raspberry Pi's desktop from your laptop/PC. This will come handy if you don't have (or don't want) a display unit to see your Pi or want to use your Laptop/PC's screen and keyboard/mouse itself. This mode of accessing the terminal and/or desktop of the Raspberry Pi without connecting it to any display unit (and not connecting any keyboard/mouse to the Pi) is usually termed as Headless mode.

In most of the remote desktop tutorials, you'd need to initially install TightVNC Server (software that streams the Desktop's GUI to any VNC Client) on the Raspberry Pi and then install any VNC viewer on the Laptop/PC. Now, installing TightVNC server on the Raspberry Pi requires it to be connected to the internet. And here's my problem: I don't have a Router or Internet cable in my home. I essentially use a 3G dongle for all my internet needs.

We even tried installing XRDP onto Raspberry Pi (by copying the tar and running it on the Pi) but this failed as it tried downloading the missing dependencies.

We tried couple of methods to share our Laptop's internet connection with the Raspberry Pi but they failed. Desperate still to access my Raspberry Pi's desktop, we stumbled upon this setup which finally worked.

Before we proceed with remotely accessing the Raspberry Pi's desktop, we first need to be able to SSH into it (access it's shell, akin to command prompt in windows OS). The procedure for SSHing into the Raspberry Pi without the need of any Display Device for Raspberry Pi is explained below.

Direct SSH into Raspberry Pi in Headless Mode

In this tutorial, we'll quickly setup our Raspberry Pi connected locally to our PC/Laptop using a standard Cat 5 Ethernet cable (crossover not required).

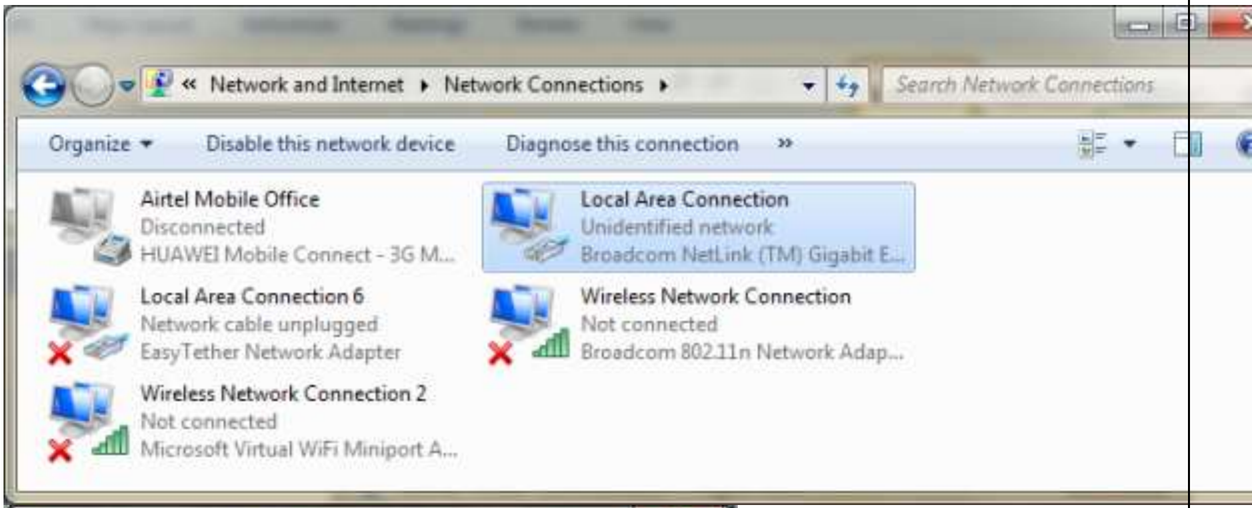
Edit: 1st Jan, 2015:

Edit: 11 September, 2015: Can someone please confirm if this works on Windows 10? Thanks.

I compiled a quick video illustrating the following steps so that you can configure it more easily.

Please follow the below steps to have a working SSH connection:

1. Install and burn the latest Raspbian Wheezy OS onto the SD card for the Raspberry Pi
2. While the Raspberry Pi is switched off, connect one side of the Ethernet cable to Raspberry Pi and other side to the RJ45 jack of the PC/Laptop
3. Open LAN properties and make sure that IPV4 properties are set to Obtain IP address automatically as shown below:


« Network and Internet ▸ Network Connections ▸
Search Network Connections
Organize ▾
Disable this network device
Diagnose this connection
»

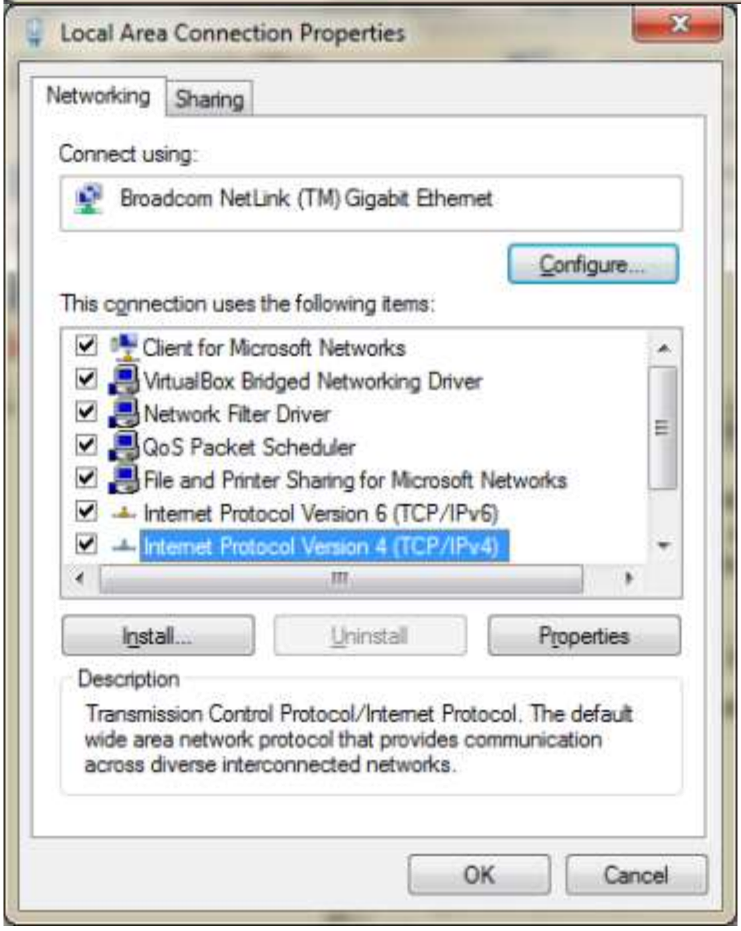
Airtel Mobile Office
Disconnected
HUAWEI Mobile Connect - 3G M...

Local Area Connection
Unidentified network
Broadcom NetLink (TM) Gigabit E...

Local Area Connection 6
Network cable unplugged
EasyTether Network Adapter

Wireless Network Connection
Not connected
Broadcom 802.11n Network Adap...

Wireless Network Connection 2
Not connected
Microsoft Virtual WiFi Miniport A...

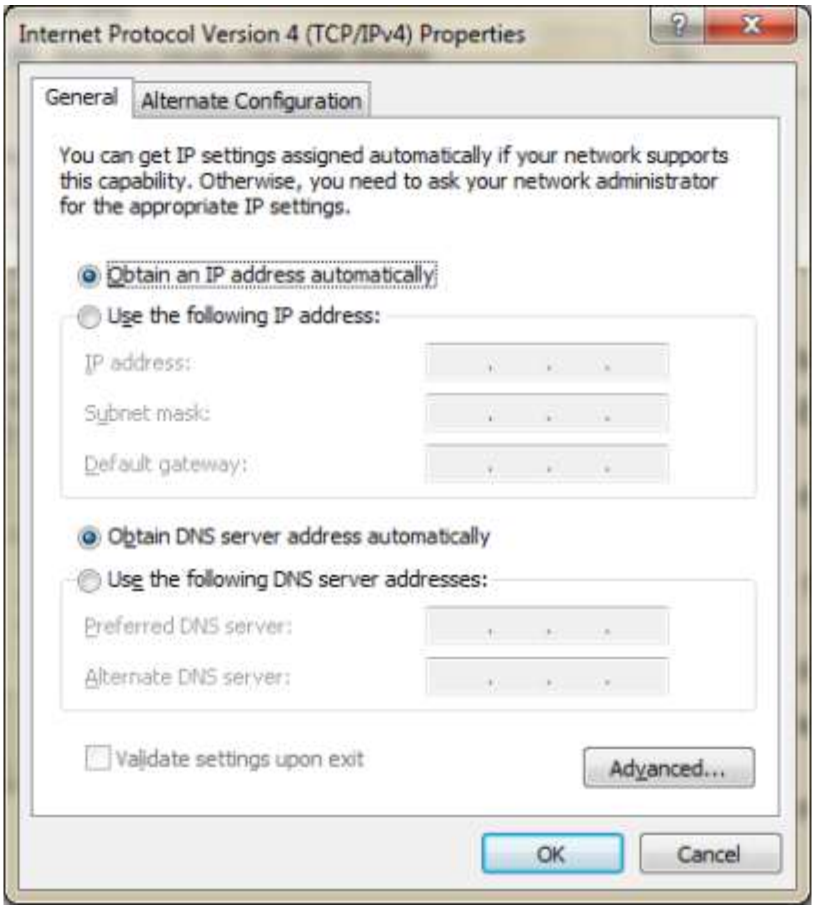


Connect using:
Broadcom NetLink (TM) Gigabit Ethernet

This connection uses the following items:

- ☑ Client for Microsoft Networks
- ☑ VirtualBox Bridged Networking Driver
- ☑ Network Filter Driver
- ☑ QoS Packet Scheduler
- ☑ File and Printer Sharing for Microsoft Networks
- ☑ Internet Protocol Version 6 (TCP/IPv6)
- ☑ Internet Protocol Version 4 (TCP/IPv4)

Description
Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.



Internet Protocol Version 4 (TCP/IPv4) Properties

General | Alternate Configuration

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

[✓ Obtain an IP address automatically]

Use the following IP address:

IP address: . . .

Subnet mask: . . .

Default gateway: . . .

[✓ Obtain DNS server address automatically]

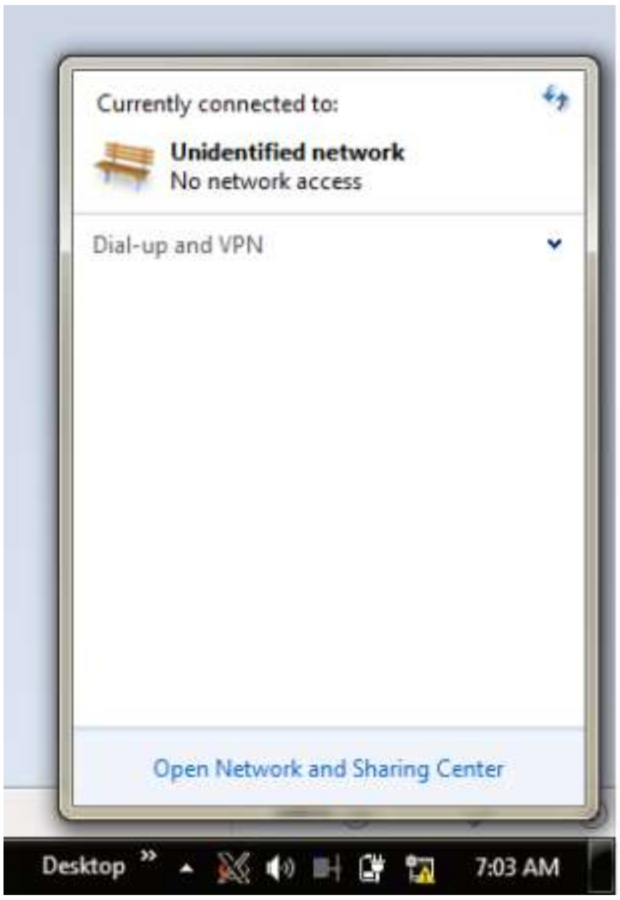
Use the following DNS server addresses:

Preferred DNS server: . . .

Alternate DNS server: . . .

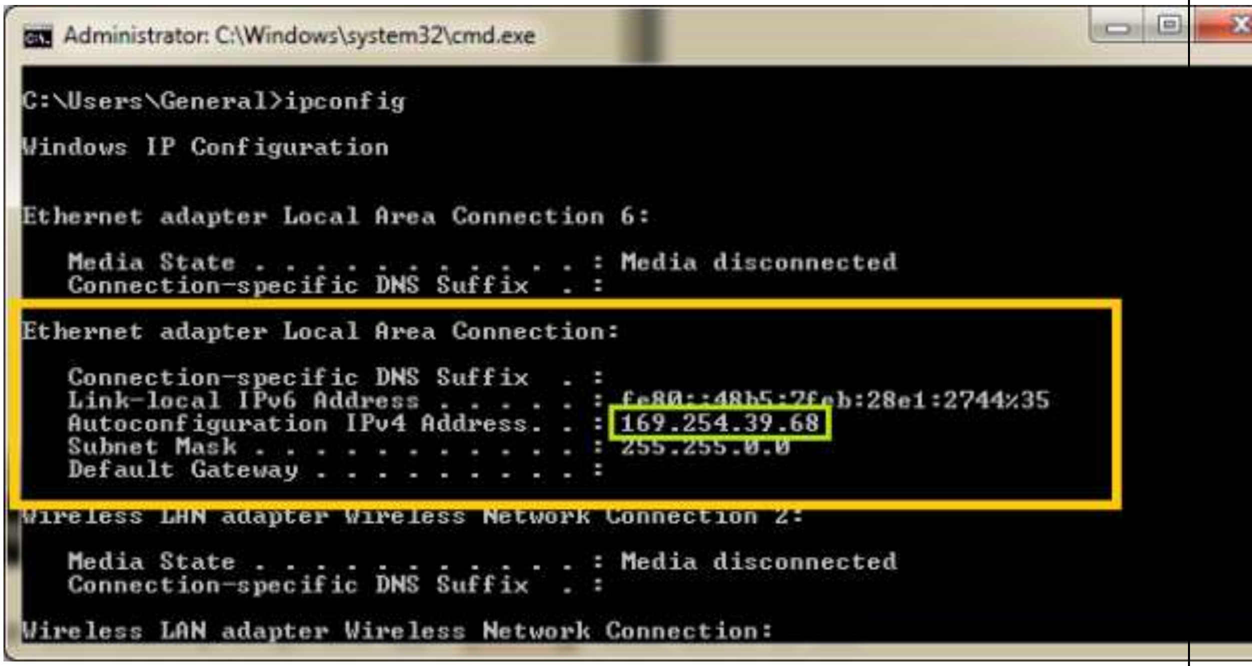
☐ Validate settings upon exit

4. We now need to determine the IP of our PC/Laptop when it's connected to the Raspberry Pi
5. Now power on the Raspberry Pi while making sure that the network cable is connected on both ends
6. Wait for a min or two. You'll notice that the PC/Laptop will scan and then show a small warning indicating the presence of an unidentified network



7.

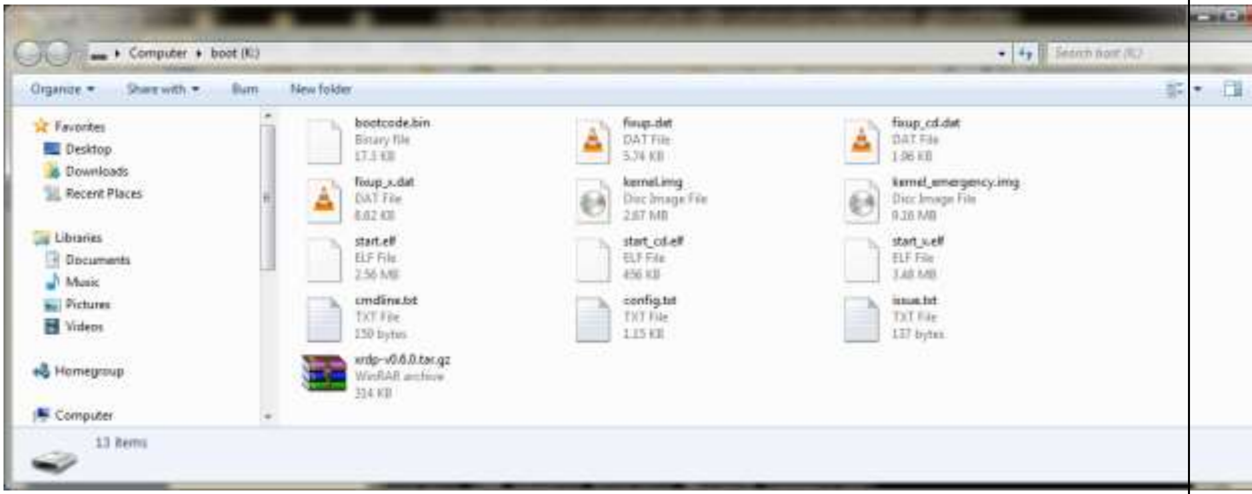
7. Now, open command prompt and type **ipconfig**. Note the IP of the Ethernet Adapter Local Area Connection as shown below:



```
Administrator: C:\Windows\system32\cmd.exe

C:\Users\General>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection 6:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::48b5:7feb:28e1:2744%35
   Autoconfiguration IPv4 Address. . : 169.254.39.68
   Subnet Mask . . . . . . . . . . . : 255.255.0.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Wireless Network Connection 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wireless Network Connection:
```

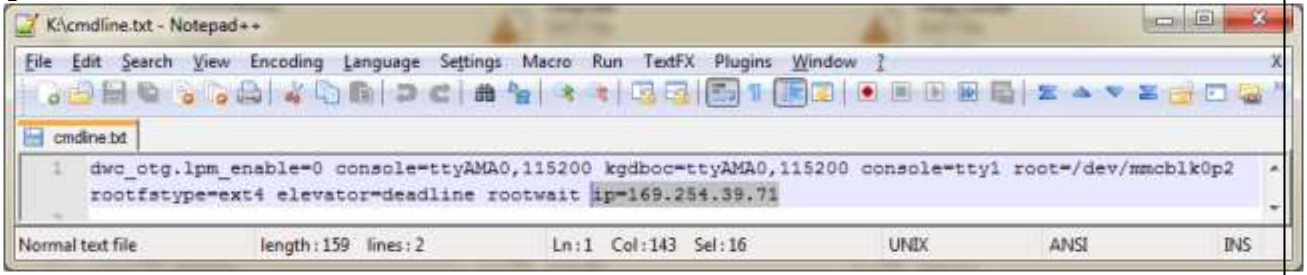
In our example, its **169.254.39.68**

8. Now, power off the Raspberry Pi and take out the SD Card. Plug in the SD card into a card reader and open it. You should see a couple of files



9. Open the **cmdline.txt** file and append this to the end of it:

ip=169.254.39.71



```
dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1 root=/dev/mmcblk0p2
rootfstype=ext4 elevator=deadline rootwait ip=169.254.39.71
```

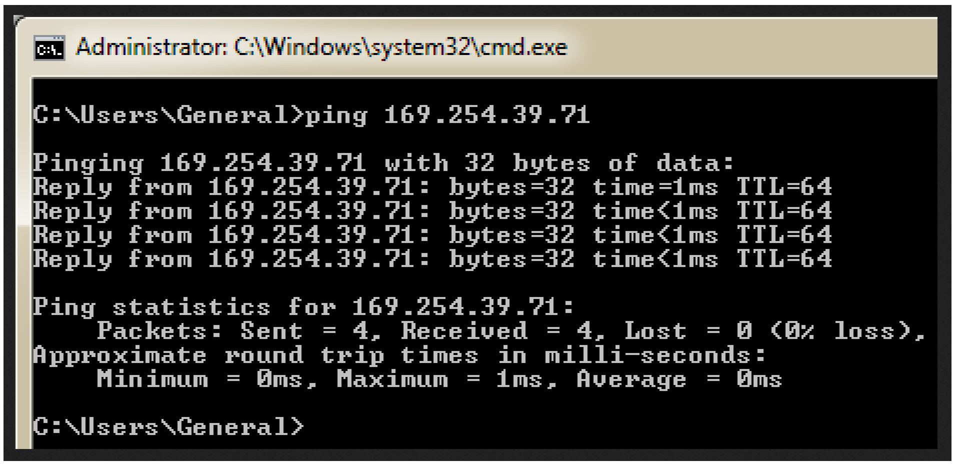
Please change the IP accordingly and assign a unique value (while making sure you don't go beyond the subnet mask). Preferably, change only the last parameter

For example, if your LAN's ip is **169.254.0.1**, we recommend using **169.254.0.3** as the IP for Raspberry Pi in **cmdline.txt**

10. Save the **cmdline.txt** file without making any other changes
11. Plug this back into the Raspberry Pi and with the Ethernet cables connected, power on the Raspberry Pi
12. Wait a couple of minutes while the Raspberry Pi tries to establish a local network connection with our PC/Laptop
13. Once you see the network warning message as shown above, ping the Raspberry Pi to see if it's live on the network as follows
    1. Open CMD prompt
    2. Type **ping ipaddress of raspberry-pi**

Ex: **ping 169.254.39.71**

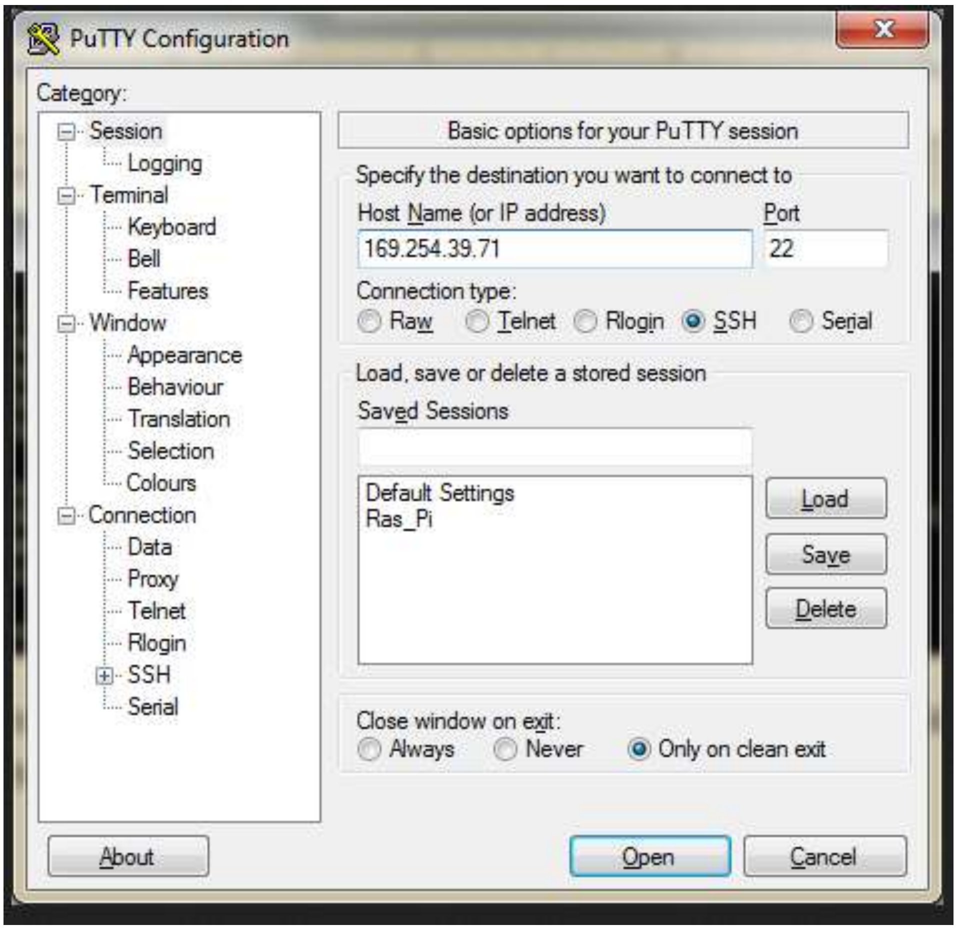
If all went good, we will see the Pi responding back to the ping requests as shown below:



Administrator: C:\Windows\system32\cmd.exe

```
C:\Users\General>ping 169.254.39.71

Pinging 169.254.39.71 with 32 bytes of data:
Reply from 169.254.39.71: bytes=32 time=1ms TTL=64
Reply from 169.254.39.71: bytes=32 time<1ms TTL=64
Reply from 169.254.39.71: bytes=32 time<1ms TTL=64
Reply from 169.254.39.71: bytes=32 time<1ms TTL=64

Ping statistics for 169.254.39.71:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Users\General>
```

- Now its time to SSH into the R Pi. Open PuTTY client and type in the IP address of the Raspberry Pi (169.254.39.71) and hit Open



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 169.254.39.71

Port: 22

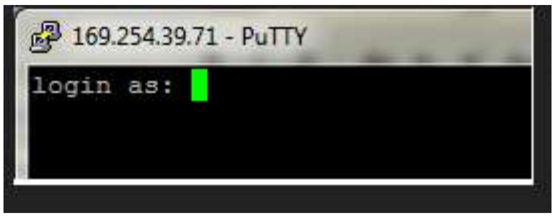
Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions: Default Settings, Ras_Pi

Close window on exit: Always Never [✓ Only on clean exit]

We get a certificate trust warning message (if SSHing for the first time) which we shall accept. Once done, we'll see the login terminal:

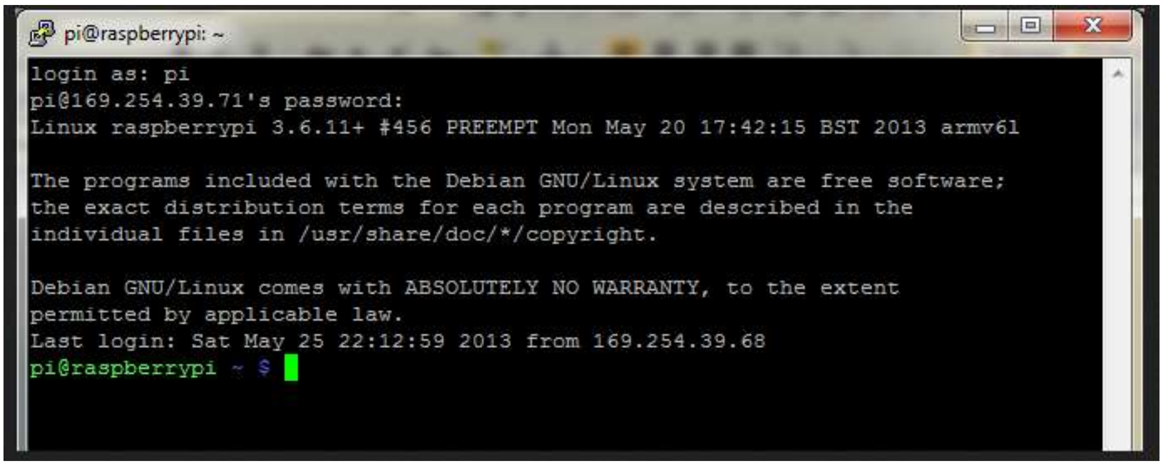


```
169.254.39.71 - PuTTY
login as:
```

Enter pi as the user name and password is raspberry

After entering the credential hit enter to SSH in to raspberry Pi

If we are successful then we would see the screen below on our compueter screen



```
pi@raspberrypi: ~
login as: pi
pi@169.254.39.71's password:
Linux raspberrypi 3.6.11+ #456 PREEMPT Mon May 20 17:42:15 BST 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 May 25 22:12:59 2013 from 169.254.39.68
pi@raspberrypi ~ $
```

6.3 Setting up Kiosk Mode

To start displaying anything on the screen we need to go through a few steps to setup and disable a few settings.

Firstly we should disable the screensaver and any energy saving settings as we don't want our screen to go to sleep at all when it's in use, wouldn't be very useful if it went blank every 5 minutes.

While connected to your pi over SSH type

sudo nano /etc/xdg/lxsession/LXDE/autostart

autostart file is a file that runs when your pi boots.
To disable the screensaver add a # to the beginning of the line, this comments the line out.

@xscreensaver -no-splash

Next add these lines underneath the screensaver line

```
@xset s off
@xset -dpms
@xset s noblank
```

This disables power management settings and stops the screen blanking after a period of inactivity.

Now that is done we should prevent any error messages displaying on the screen in the instance that someone accidentally power cycles the pi without going through the shutdown procedure. To do this we add the following line underneath the lines you just added.

```
@chromium --noerrdialogs --kiosk http://localhost/FileRead.php --incognito
```

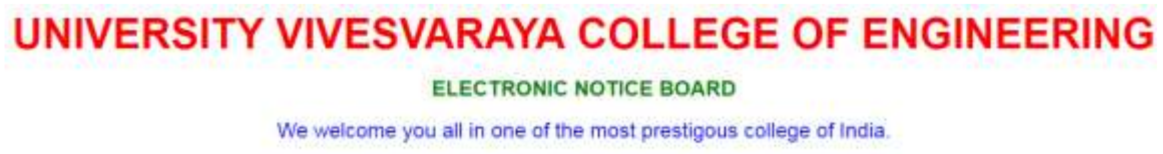
The url http:localhost//FileRead.php is a file which is to be opened in kiosk mode to display

Hit ctrl-O and then ctrl-X again to write out and exit the file and now type.

Reboot the system to apply changes made to system

```
sudo reboot
```

Keep an eye on the display from your Pi now and once the unit boots again it should automatically load chromium in kiosk mode and display the page you'd chosen.



UNIVERSITY VIVESVARAYA COLLEGE OF ENGINEERING

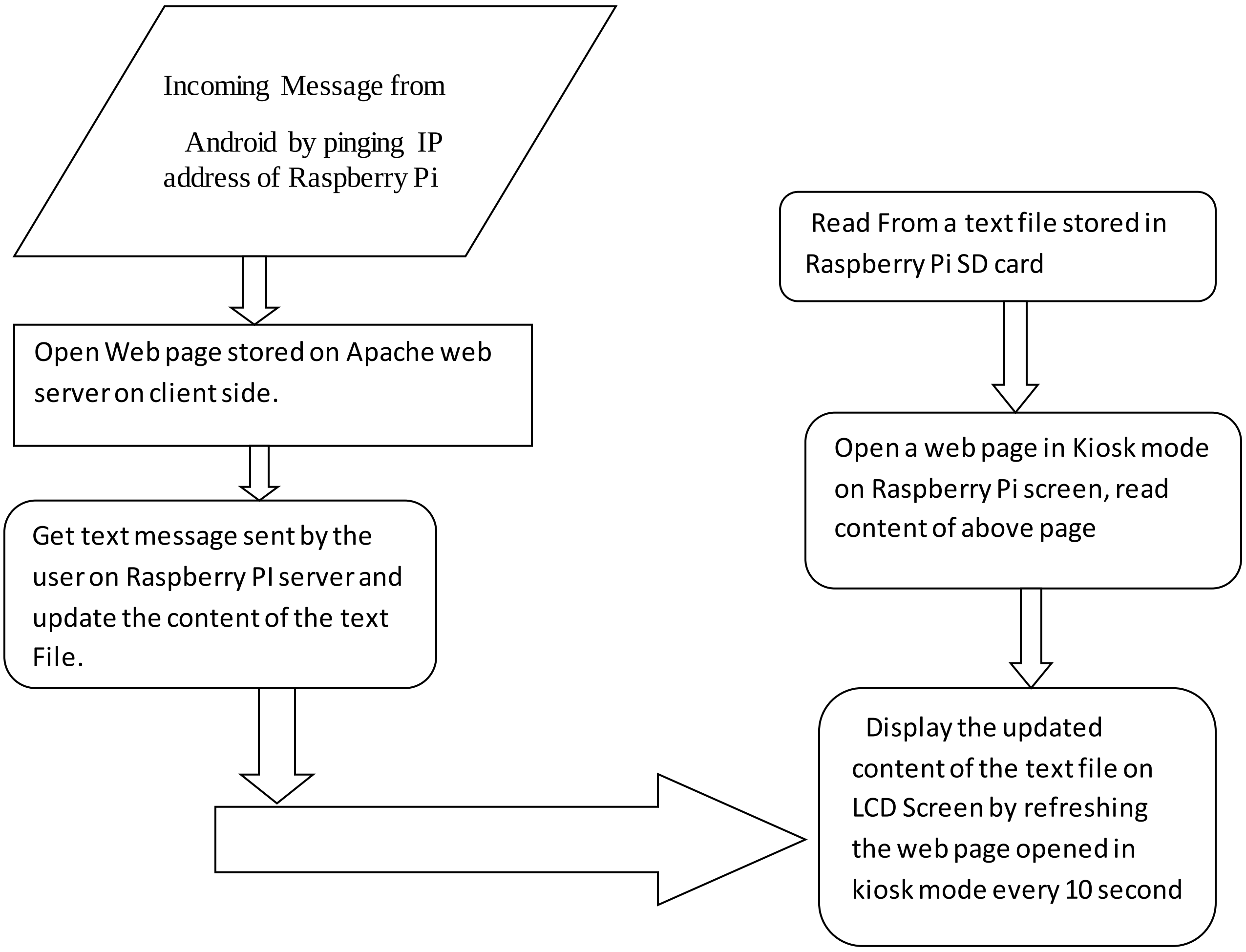
ELECTRONIC NOTICE BOARD

We welcome you all in one of the most prestigous college of India.

CHAPTER 6

SOFTWARE ARCHITECTURE IMPLEMENTATION

6.1 FLOWCHART


Incoming Message from Android by pinging IP address of Raspberry Pi
Open Web page stored on Apache web server on client side.
Get text message sent by the user on Raspberry PI server and update the content of the text File.
Read From a text file stored in Raspberry Pi SD card
Open a web page in Kiosk mode on Raspberry Pi screen, read content of above page
Display the updated content of the text file on LCD Screen by refreshing the web page opened in kiosk mode every 10 second

6.2 Client Side Code

The client side code is stored on raspberry pi apache we server in the following directory

/var/www/

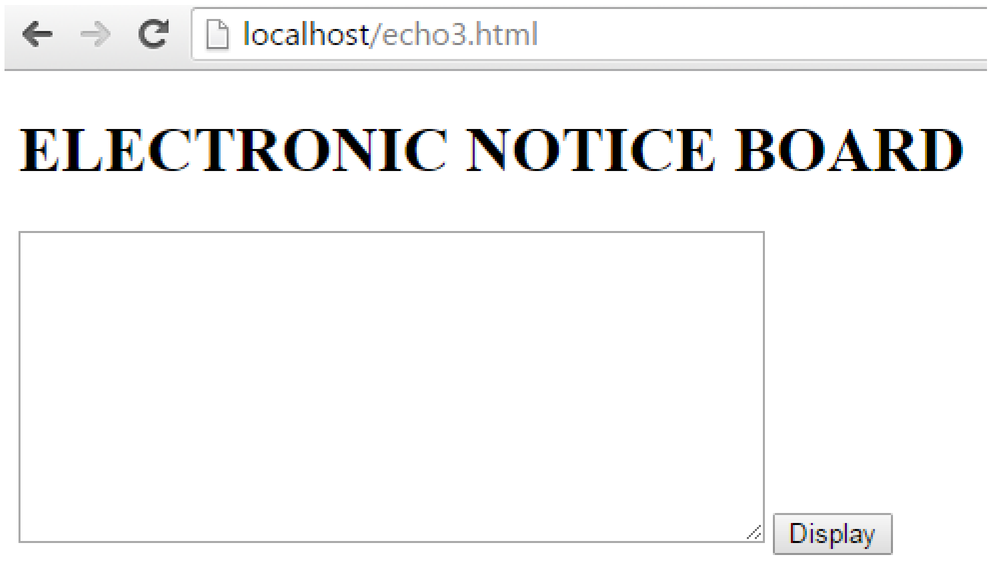
The file name is echo3.html

Following are the contents of the file

```
<!DOCTYPE HTML>
        <html>
        <head>
        <title>Write to a text file</title>
        </head>
        <body>
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-
scale=1">
      <h1>ELECTRONIC NOTICE BOARD</h1>
   <form action="myfile.php" method='post'>
        <textarea rows="10" cols="50" name='textblock'></textarea>
        <input type='submit' value='Display'>
        </form>
        </body>
        </html>
```

This page opens with a heading ELECTRONIC NOTICE BOARD and opens up a text box in which user can enter whatever text which he/she wants to display on the screen and as soon as he hits display button a internal php page name as myfile.php opens a file and stores the user enterd text in file on Raspberry Pi. The content of myfile.php are as the following

```
<?php
        // Open the text file
        $f = fopen("web.txt", "w");
        // Write text in to the file
        fwrite($f, $_POST["textblock"]);
    fwrite($f,"\r\n");
        // Close the text file
        fclose($f);
        // Open file for reading, and read the line
        $f = fopen("web.txt", "r");
        // Read text
        //echo fgets($f);
        fclose($f)
        ?>
```



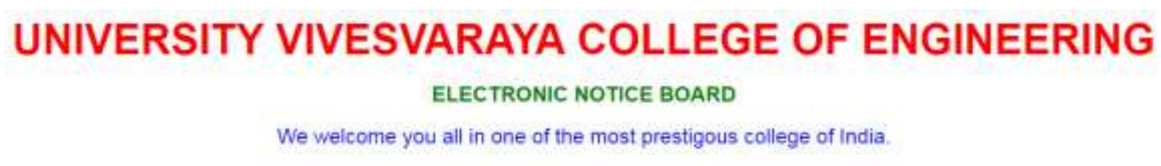
6.3 Server Side Code

When Raspberry Pi opens it open a web page in chromium in kiosk mode which displays the user sent messages from android phone to LCD screen. Following are the contents of that page

```
<?php

echo "<h1 align='center'> <font color=red font face='arial' size='10pt'</font>
UNIVERSITY VISVESVARAYA COLLEGE OF ENGINEERING </h1>";

echo "<h2 align='center'> <font color=green font face='arial'
size='5pt'</font>ELECTRONIC NOTICE BOARD</h2>";

$myfile = fopen("web.txt", "r") or die("Unable to open file!");

$data = fread($myfile,filesize("web.txt"));

fclose($myfile)

echo "<p align='center'> <font color=blue size='5pt'>$data </font> </p>";

header("refresh:10");

?>
```

This Web Page Displays the title as the name of our college and sub heading as Electronic Notice Board and it open a text file in which the text sent from the user from android phone are stored and displays it on the Screen. The Page is Programmed to automatically refreshes every 5 second to update the most recent message.

The Page looks like this with user sent messages are displayed in blue color font



UNIVERSITY VIVESVARAYA COLLEGE OF ENGINEERING

ELECTRONIC NOTICE BOARD

We welcome you all in one of the most prestigous college of India.

CHAPTER 7

ANDROID APPLICATION DEVELOPMENT

We have developed an android app using android studio intellij IDE. The application presents user with a text box where user needs to enter the IP address allotted to pi by router (please refer sec 4.5 on connecting Wi-Fi to router). After entering the correct ip address when the user hits enter button a page will be opened with a textbox and button. In the text box enter the text which is to be displayed on LCD Screen

Following is a screen shot of our application developed

7.1 Android Studio Overview

Android Studio is the official IDE for Android app development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

- A flexible Gradle-based build system
- Build variants and multiple APK file generation
- Code templates to help you build common app features
- A rich layout editor with support for drag and drop theme editing
- Lint tools to catch performance, usability, version compatibility, and other problems
- Code shrinking with ProGuard and resource shrinking with Gradle
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

This page provides an introduction to basic Android Studio features. For more detailed guides to using Android Studio, start by browsing pages in the Workflow section.

7.2 Project Structure

Each project in Android Studio contains one or more modules with source code files and resource files. Different types of modules include:

Android app modules

Test modules

Library modules

App Engine modules

By default, Android Studio displays your project files in the Android project view, as shown in figure 1. This view is organized by modules to provide quick access to the key source files of your project.

All the build files are visible at the top level under Gradle Scripts and each app module contains the following three elements:

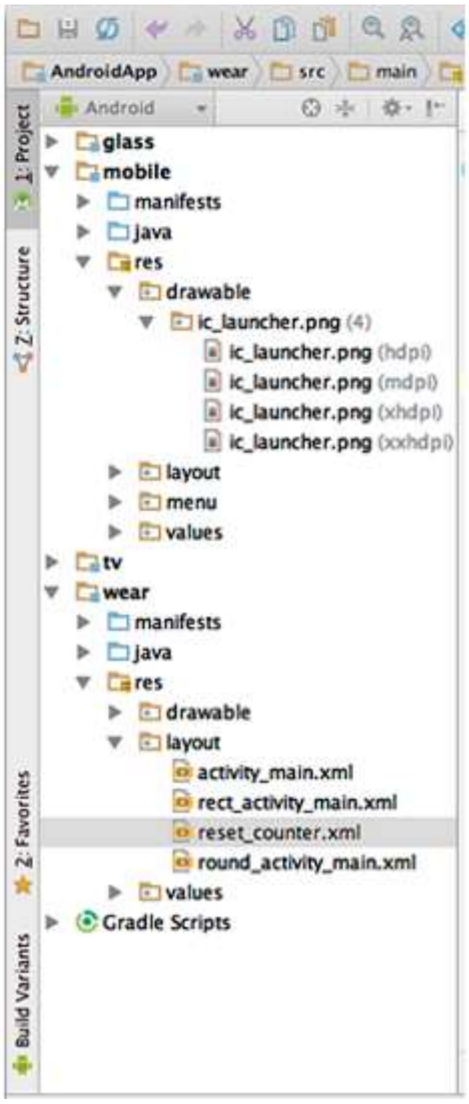
manifests: Manifest files.

java: Source code files.

res: Resource files.

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select Project from the Project drop-down (in figure 1, it's showing as Android).

You can also customize the view of the project files to focus on specific aspects of your app development. For example, selecting the Problems view of your project displays links to the source files containing any recognized coding and syntax errors, such as missing an XML element closing tag in a layout file.



7.3 Main Activity File

The main activity code is a Java file **MainActivity.java**. This is the actual application file which ultimately gets converted to a Dalvik executable and runs your application.

This file contains the following contents

```
package com.example.user.webviewapplication;


import android.support.v7.app.ActionBarActivity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.webkit.WebView;

import android.widget.Button;

import android.widget.EditText;

import android.view.View;

import android.webkit.WebViewClient;

import android.widget.Toast;


import java.io.FileInputStream;

import java.io.FileOutputStream;


public class MainActivity extends ActionBarActivity {

Button b1;

EditText ed1;


private WebView wv1;

@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

b1 = (Button) findViewById(R.id.button);
ed1 = (EditText) findViewById(R.id.editText);

wv1 = (WebView) findViewById(R.id.webView);
wv1.setWebViewClient(new MyBrowser());

b1.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
String url = "http://"+ed1.getText().toString()+"/echo3.html";

wv1.getSettings().setLoadsImagesAutomatically(true);
wv1.getSettings().setJavaScriptEnabled(true);
wv1.setScrollBarStyle(View.SCROLLBARS_INSIDE_OVERLAY);
wv1.loadUrl(url);
}
});
}
private class MyBrowser extends WebViewClient {
@Override
public boolean shouldOverrideUrlLoading(WebView view, String url) {
view.loadUrl(url);
return true;
}

}



@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.menu_main, menu);

return true;

}


@Override

public boolean onOptionsItemSelected(MenuItem item) {

// Handle action bar item clicks here. The action bar will

// automatically handle clicks on the Home/Up button, so long

// as you specify a parent activity in AndroidManifest.xml.

int id = item.getItemId();


//noinspection SimplifiableIfStatement

if (id == R.id.action_settings) {

return true;

}


return super.onOptionsItemSelected(item);

}

}
```

Here, *R.layout.activity_main* refers to the *activity_main.xml* file located in the*res/layout* folder. The *onCreate()* method is one of many methods that are figured when an activity is loaded.

7.3 Manifest File

Whatever component you develop as a part of your application, you must declare all its components in a *manifest.xml* which resides at the root of the application project directory. This file works as an interface between Android OS and your application, so if you do not declare your component in this file, then it will not be considered by the OS. This file contents are shown below

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.sairamkrishna.myapplication" >
<uses-permission android:name="android.permission.INTERNET" />
<application
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >

<activity
android:name=".MainActivity"
android:label="@string/app_name" >

<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

```
                                   </activity>


   </application>

</manifest>
```

7.4 The Layout File

The activity_main.xml is a layout file available in res/layout directory, that is referenced by your application when building its interface. You will modify this file very frequently to change the layout of your application. For your "Hello World!" application, this file will have following content related to default layout

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"

android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"

android:paddingRight="@dimen/activity_horizontal_margin"

android:paddingTop="@dimen/activity_vertical_margin"

android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity">


<TextView

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:text="UNIVERSITY VISVESVARAYA COLLEGE OF ENGINEERING "

android:textSize="50px"


android:textAlignment="center"


android:id="@+id/textView"

android:layout_alignParentTop="true"
```

```
android:layout_alignParentEnd="true"  />


<WebView
android:layout_width="match_parent"
android:layout_height="match_parent"
android:id="@+id/webView"
android:layout_below="@+id/button"
android:layout_alignParentLeft="true"
android:layout_alignParentStart="true"
android:layout_alignParentRight="true"
android:layout_alignParentEnd="true"
android:layout_alignParentBottom="true"  />


<EditText
android:layout_width="match_parent"
android:layout_height="50dp"
android:id="@+id/editText"
android:hint="Enter  IP  Address"
android:focusable="true"
android:textColorHighlight="#ff7eff15"


android:layout_below="@+id/textView"
android:layout_alignEnd="@+id/textView"  />
<Button
android:layout_width="match_parent"
android:layout_height="50dp"
android:text="Enter"
```

```
android:id="@+id/button"


android:layout_below="@+id/editText"
android:layout_alignEnd="@+id/textView"  />
</RelativeLayout>
```

Copy all these file after opening Android Studio and run the program. The apk file will be generated in the following location once the app-module is successfully built.

C:\Users\user\Android Studio Projects\WebViewApplication\app\build\outputs\apk

CHAPTER 8

CONCLUSION

Electronic notice board using Wi-Fi is a collaboration of Software and Hardware through which most of the complexity reduces, even systems size and cost also reduced. This system is very efficient as anyone can send the message from remote place without any human intervention. The android application developed in this project makes the user experience great as it is very simple and easy to use. The raspberry pi automatically boots and displays the screen which avoids any configuration when there is power cut or raspberry is recycled by mistake.

Scope of future work

- Highly advanced and easy to use Android application.
- Automatic text adjustment feature of the text size of the messages displayed.
- Access Raspberry pi Apache Web server globally.
- Automatic scrolling of the pages if the total size of the page is very large.
- Features to upload images on raspberry pi server.
- Feature to store last entered messages and displaying it.
- Power Raspberry pi using a solar panel in order to save power.
- Display multiple pages with a particular delay.

References and Bibliography

- https://www.raspberrypi.org
- https://www.raspberrypi.org/downloads/
- https://www.danpurdy.co.uk/web-development/raspberry-pi-kiosk-screen-tutorial/
- https://anwaarullah.wordpress.com/2013/07/16/direct-access-raspberry-pi-shell-and-desktop/
- http://www.elecfreaks.com/wiki/index.php?title=EFCom_GPRS/GSM_Shield
- http://stackoverflow.com/questions/4754387/php-how-do-i-display-the-contents-of-a-textfile-on-my-page
- https://www.raspberrypi.org/forums/viewtopic.php?f=43&t=56209&hilit=%2bphp
- https://github.com/ajstarks/openvg
- https://www.raspberrypi.org/forums/viewtopic.php?t=52613&p=403805
- http://raspberrypi.stackexchange.com/questions/1719/x11-connection-rejected-because-of-wrong-authentication
- http://www.w3resource.com/php/echo-print/echo.php
-