

# Implementation of a Smart Network Communications Interface for Home and Small Office

Mudgerikar Aditya, More Chaitanya, Palkar Soham and Mokashi Rohit

## ABSTRACT

The Smart Home bug has bitten nearly everyone. In this project, we develop a way to form a communications interface between the different smart devices in your home or small office. This eliminates, from future developers the issues of Compatibility, Network Communication, Protocol management, thereby focusing only on the core functionality. We present to you a base interface which deals with the underlying protocols, communication over different interfaces and compatibility between the different components: namely your Raspberry Pi, Sensors, cameras and other peripherals and your base computers. The developer focuses solely on the functionality of the system without the usual headaches.

## General Terms

Home Automation, Human Interface Devices.

## Keywords

Raspberry Pi 2, Arduino, Debian based Ubuntu Server, GPIO, TCP Hole Punching, IOT, Human Interface Devices, Home Automation..

## 1. INTRODUCTION

There exist various solutions for automation in the market, but none exists with such ease of use that the normal lay home user may be able to use it. With huge costs of deployment, maintenance and close systems, these solutions become inappropriate for home use. In our system we propose a solution which is reliable, cheap, and easy to use and maintain. This is an excellent solution for the avg. home tinkerer who can install self-made DIY solutions to operate his home successfully. With the advent of the multitude of various electronic devices it is imperative, the existence of such a system which can bridge the gap between the communication of these various devices with each other.

In our system, we propose a simple solution to solve all these problems with ease of use the main priority. So let's begin.

Some features of our system include:

- 1) Entire Home automation system
- 2) Wireless Router
- 3) Unlimited Drives (up to 16)
- 4) miniDLNA server
- 5) Active media transcoding for upto 2 clients and a Plex miniDLNA server.

Comparing this to other systems, we get:

- 1) Cost of Home system : 18000
- 2) Wireless Drive: 1 for 9900 per 500 GB
- 3) Limited usability
- 4) Closed upgrade paths
- 5) Security concerns, since such an open source community is not involved.

Some concerns, if they may exist:

- 1) Why USB 2.0 does not matter: Each USB 2.0 has a speed of 480mbps while the fastest protocol: Wireless N has a speed of 150mbps.

## 2. DESIGN COMPONENTS:

### 2.1 Hardware Components

#### 2.1.1 Raspberry Pi 2:

The Raspberry Pi 2 is the core component of our system. It takes commands from the server, manages authentication, and processes the signals to be sent to the relay, based on server's commands. This core circuit provides us with the ability to monitor our home devices, power them on/off and send them commands over a network. This system's extensions include: An NAT Service, a core Firewall, A MiniDLNA Server, an NAS box, a USB Device management service and so on.

#### 2.1.2 Server:

The Server is the center of our system. You do not need a discrete server, but it is a good to have. The server extends your system beyond one room, which helps you deploy multiple Raspberry Pi 2's for extensibility.

#### 2.1.3 Relay:

The Relay actually powers the components. When the Raspberry Pi 2 sends the power on signal through the GPIO, the Relay closes the connection making the circuit to be complete allowing the device to power on.

#### 2.1.4 Gateway:

A Gateway is required so as to provide access to the web.

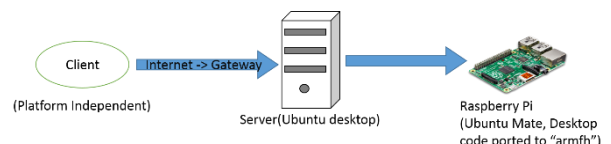


Figure 1: Hardware Components

### 2.2 Software Components:

#### 2.2.1 Ubuntu Mate armfh:

Ubuntu Mate armfh is desktop code modified to run on the Raspberry Pi 2's ARM Cortex-A7 CPU's. It has floating point and hexadecimal operations available. The Ubuntu developer team has managed to successfully port most of the functionality and the libraries, so that programmers can code the same code on the ARM platform. This saves a lot of headaches when developing for the armfh platform.

#### 2.2.2 Ubuntu:

Ubuntu is a Linux operating system. It is based on Debian. Ubuntu contains the .deb files. Ubuntu Touch is used by personal computers, tablets, etc. Ubuntu is very much secure operating system. In our system we have used the Ubuntu as

server. It can run on most popular architectures such as ARM based architecture.

### 2.2.3 Lex and Yacc:

Lex and Yacc are used for compiling a language. For compiling a language there are total 7 steps. Among those 7 steps Lexical Analysis and Syntax Analysis are the first 2 steps which includes Lex and Yacc respectively.

### 2.2.4 A Server Side Scripting Language:

The Server is essential in managing all the connections. It aggregates the data. Sends it to the required Server, and parses all the incoming connections. It is an integral part of our project. [1]

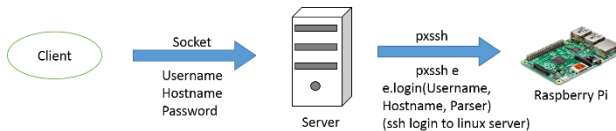


Figure 2: Software Components

## 3. DESIGN IMPLEMENTATION:

### 3.1 Preparing the Raspberry Pi 2:

#### 3.1.1 Software Required:

- **MiniDLNA:**  
MiniDLNA is a lightweight server software which is compliant with DLNA/UPnP clients. This daemon serves media files like music, pictures, and video, etc. to the requesting clients on given network. MiniDLNA has a rich set of configuration options which can easily be accessed by editing a text file.
- **Samba:**  
Samba is a network configuration software. It is based on the Windows network share protocol. Its main purpose lies in File System conversion between incompatible protocols for network shares between two separate OS's on different hardware over different communication protocols.
- **GPIO Headers (For Include files):**

3V3	1	2	5V
GPIO 2	3	4	5V
GPIO 3	5	6	GND
GPIO 4	7	8	GPIO 14
GND	9	10	GPIO 15
GPIO 17	11	12	GPIO 18
GPIO 27	13	14	GND
GPIO 22	15	16	GPIO 23
3V3	17	18	GPIO 24
GPIO 10	19	20	GND
GPIO 9	21	22	GPIO 25
GPIO 11	23	24	GPIO 8
GND	25	26	GPIO 7
ID_SD	27	28	ID_SC
GPIO 5	29	30	GND
GPIO 6	31	32	GPIO 12
GPIO 13	33	34	GND
GPIO 19	35	36	GPIO 16
GPIO 26	37	38	GPIO 20
GND	39	40	GPIO 21

- The GPIO (General Purpose Input/Output) is a set of generic pins on the circuit board of Raspberry Pi controlled by a micro controller whose functions can be accessed by the C/C++ header file "GPIO.h".
- This avoids the need of separate digital circuitry to provide for digital Input/Output lines to emulate given sample macro circuitry.
- The GPIO pins are controlled by the micro controller Broadcom BCM2835 SoC which provides us a set of 26 GPIO pins for use with computer components.

### Figure 3: GPIO set for the Raspberry Pi used

#### 3.1.2 Setting it up:

Installation of the components:

- **MiniDLNA:**  
sudo apt-get install miniDLNA
- **Samba:**  
sudo apt-get install samba smbfs

## 3.2 Preparing the Server:

#### 3.2.1 Software Required:

- **Ubuntu Server:**  
Ubuntu server is the server edition of Ubuntu. It uses the same APT repositories as desktop version of Ubuntu. The main difference between them being, that the server edition does not include GUI environment of Ubuntu in its default installation.
- **PHP:**  
PHP is a server side scripting language which stands for Hypertext Pre-Processor, earlier called as personal homepage. In this system we use PHP above the Apache server for HTML page access as well as in CLI.
- **Apache 2:**  
In our system we use an Apache server to enable hosting of PHP code for server side computations. The PHP code is the backbone of our system to which Apache enables access to all the platform independent libraries for efficient server execution.
- **SSH client:**  
SSH is an acronym for Secure Shell. SSH is used to login to our Raspberry Pi so that the end client need not worry about security until the end server is hacked.

#### 3.2.2 Setting it up:

- **Ubuntu server:**  
sudo apt-get install tasksel  
Download the image from <http://cdimages.ubuntu.com>
- **PHP:**  
sudo apt-get install php5 libapache2-mod-php5
- **Apache:**  
sudo apt-get install apache2
- **SSH:**  
apt-get install openssh

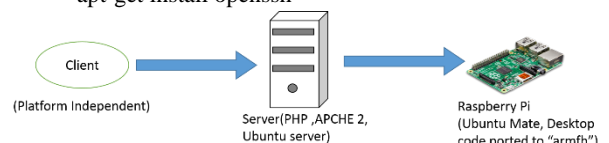


Figure 4: Software Stack

### 3.3 Preparing the Connections:

#### 3.3.1 Hardware Setup Required

- **Ethernet cable:**  
Ethernet cable is mostly popular for connection wired network. It connects devices on LAN like PCs, routers, switches.
- **Switch:**  
To receive, process and forward data to its particular destination device switches are used. It is a computer networking device. Above three processes are done by using packet switching.
- **Power Source:**
  - For Raspberry Pi: a 5V 2.5A Power supply
  - For GPIO: A 5V 2.5 A Power supply
  - For Server (If required): A 450W Power supply.

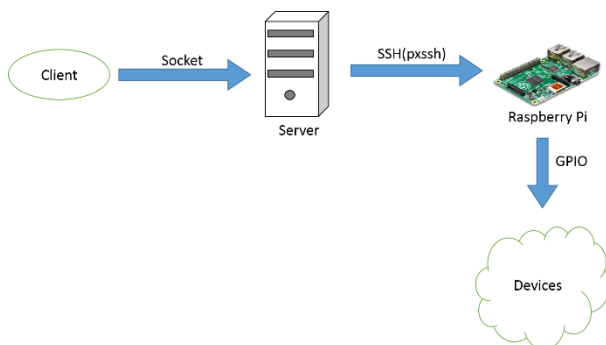


Figure 5: Hardware Stack

### 3.4 Setting it all together:

#### 3.4.1 Hardware Setup Required:

- Raspberry Pi 2
- Server: Any Machine running a minimal OS, for running a server.
- Relay
- Male to Female cables
- Breadboard.
- AC Power Source
- An Extension Board with Switches.

#### 3.4.2 Setting it up:

To set up the system,

- We power on the Raspberry Pi and the Server.
- Test the ssh connectivity and User`s permissions on the Raspberry Pi.
- Connect the Server to the Router.
- Enable DMZ Host on the router: To make the incoming requests point to the server.
- Connect the devices to the Relay.
- Check the connection of each device using an Arduino. (Very Important!)

- Connect the 8-channel/6channel/4channel relay to the Raspberry Pi 2.
- Test the code from the Raspberry Pi 2. (Before connecting the server).
- Test the connections from the server side: By sending random Device Power ON Requests as required from the server.
- Then test the connection from the client side.
- Then check the connection with multiple Raspberry Pi`s powered ON and Connected to the network, with their multiple relays.
- The test from the client side from the GUI.
- Regressively test the system in various connections like: Internet, Local Wi-Fi Hotspot, LAN, Local Network, extending to a few cities, etc.
- After the System is working, declare it as complete and test it at the actual client devices: Like web GUI, Admin Pages, and the Android App (If necessary). [2]

### 3.5 Testing the system:

#### 3.5.1 Testing Methodology:

- **Regression testing:**  
After the addition of each feature, the system was tested manually by software testers and programmers alike, to find bugs in the software.
- **Penetration Testing/Security Testing:**  
The system was tested against various security flaws, none of which were enough to compromise the system. The Penetration testing was done by Hackers and programmers alike.
- **Stress Testing:**  
The system responded well to various scenarios of differing load, well adhering to the protocols and standards of security.  
The Testing performed on the system revealed that the packages and the code was efficient to run and that the custom Ubuntu Distribution has done no defect to the overall maintainability of the system under load.  
The System performed well under the Stress testing.

#### 3.5.2 Scenarios and Components:

The System was tested under various scenarios and differing load:

The system performed well under being pushed to its max limits, and did not fail in any of the tests.

The scenarios considered were:

- **Only a single Raspberry Pi with no Server:**
  - In case of a small room, no server is required.
  - The Raspberry Pi handles the load well, without the need of a special server, thereby omitting Electricity usage and General Maintenance. [3]
- **A Single Raspberry Pi with no Server communicating with an App:**
  - In case of a single Raspberry Pi 2 with no server communicating with an App directly, there is no

need for an end to end server side scripting language like PHP. This system can be run directly under a C++ program with an Additional code for a Socket Connection.

## 4. RESULTS AND ANALYSIS

### 4.1 Results:

After a Successful implementation of the system, we get the following results:

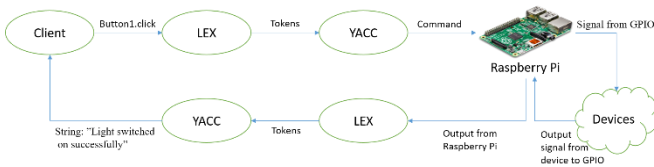


Figure 6: Command Flow

The Flow of the system is as follows:

- The User clicks a button to perform an action.
- The data is sent to the Server over the Connection.
- The Server sends the Data to Lex for Tokenizing.
- The Tokens are sent to YACC for Parsing and generating Command strings.
- The generated Command Strings are sent to the Raspberry Pi for Execution.
- The Raspberry Pi sends a signal to the GPIO Microcontroller for outputting a signal to the Relay.
- A Signal is sent from the GPIO to the Respective Header on the Relay.
- The Relay receives the signal, and depending on whether it is active Low or Active High, it switches ON/OFF the current.
- On receiving the current, the device turns ON.
- If the device s capable, it sends a signal back to the Raspberry Pi via the GPIO or any other Headers: (Like USB Port or Ethernet).
- The Raspberry Pi sends this signal back to the server.
- The server then acts on the command (on whether parsing is required), and sends the output back to the Client.

Alternatively: without LEX and YACC:

- The Client sends regular signals to the server.
- But, the server (PHP), instead of Parsing, directly sends the command to the Raspberry Pi for execution, based on the if-else conditions, or the loops, in C/C++/PHP/Java.
- The Raspberry Pi then executes this command, and if the device returns an error, sends it to the server.
- The server, on receiving it, parses it based on if-loop conditions, and sends the output back to the client.

## 4.2 Analysis of speed:

### 4.2.1 Latency

Latency is nothing but the response time for particular transforming devices. That is after getting request from any device, the current device has a particular response time and this response time is known as Latency. In our system there are various devices that have their particular latency as shown in the table:

Table 1. Latency of various components

Sr. No.	Device	Latency(in ms)
1	Client Side:	
	1.1 Client Side Architecture	~ 10ms
	1.2 Cell Tower	~ 110ms
	1.3 Client ISP (3G,4G)	10ms~100ms
2	Internet Route:	10ms ~ 100ms ~ 10s
3	Physical Layer:	
	3.1 LAN Wire (CAT 5e)	1ms(per 15meters)
4	Server Side:	
	4.1 Router/Gateway	2ms
	4.2 Main Server(UBUNTU Core)	2ms response time + process(~ 5ms)
	4.3 Raspberry Pi	2ms response time + process =~ 5ms
	4.4 GPIO + Relay	response time =350 ms

#### • Client Side:

There are basic sub points in it, first one is Client Side Architecture, next is the Cell Tower and Client Internet Service Provider (ISP). In client side there is basically mobile like device so that anyone can operate the electrical appliances through that device. It sends request to the cell tower and then to client ISP which is also called as 3G provider. There is latest version of it is the 4G provider which is actually faster than 3G provider.

#### • Internet Route:

There are multiple hops over the internet which have latency between 10ms to 10s. These can be discovered using the “traceroute” command.

```
Administrator: Command Prompt
C:\WINDOWS\system32\tracert google.co.in
Tracing route to google.co.in [216.58.203.131]
over a maximum of 30 hops:
  0  0 ms  0 ms  0 ms  127.0.0.1
  1  1 ms  *  51 ms  ASH.mshome.net [192.168.137.1]
  2  *  *  *  Request timed out.
  3  1 ms  1 ms  1 ms  172.25.27.1
  4  90 ms  2 ms  1 ms  14.140.40.118.static-pune.vsnl.net.in [14.140.40.118]
  5  *  *  *  Request timed out.
  6  85 ms  4 ms  3 ms  121.244.33.242.static-pune.vsnl.net.in [121.244.33.242]
  7  123 ms  27 ms  26 ms  172.29.253.33
  8  78 ms  27 ms  26 ms  172.31.107.28
  9  *  *  *  Request timed out.
 10  *  *  *  Request timed out.
 11  27 ms  28 ms  29 ms  115.240.1.50
 12  80 ms  31 ms  *  72.14.233.204
 13  59 ms  26 ms  25 ms  245.237.50.170
 14  143 ms  35 ms  27 ms  64.133.174.161
 15  122 ms  26 ms  25 ms  216.239.58.113
 16  122 ms  24 ms  24 ms  bomb0210-in-73.1e180.net [216.58.203.131]
Trace complete.
```

Figure 7: Tracert command

#### • Physical Layer:

For physical layer basic thing we use is the LAN Wire. There are three types of LAN wires:

- 1) CAT 5, 2) CAT 5e, 3) CAT 6, where CAT 5 is used for the transformation data up to 10Mbps, CAT 5e is for 100Mbps and CAT 6 is for 1000Mbps. The Latency for this wire is measured per meter.

#### • Server Side:

The server contains the router, main server (UBUNTU core), Raspberry Pi (RPI), relay. In sever it has

response time and process time for calculating latency. There is particular latency for each every device in server. Here, router, Raspberry Pi and relay are the actual devices whereas main server (core UBUNTU) is an operating system.

A Graphical Representation can be shown as follows:

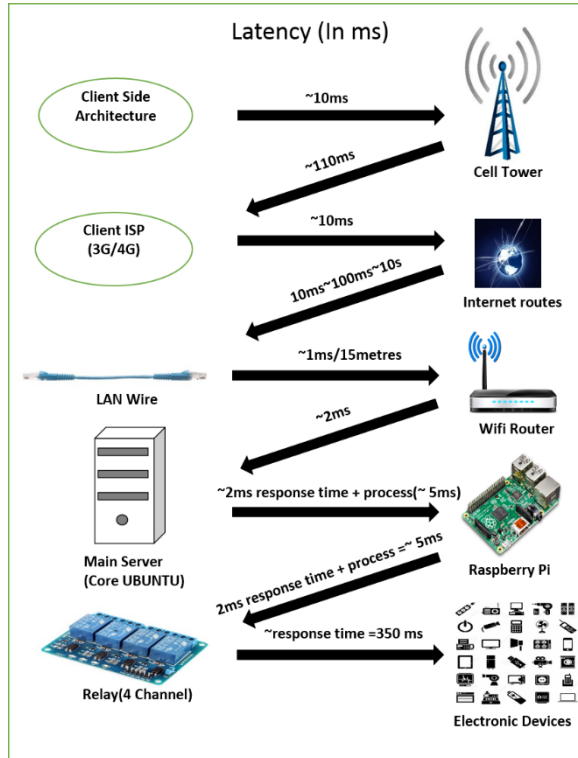


Figure 8: Latency diagram for various components

#### 4.2.2 Speed of Various Components:

We tested the speed of various components by using commonly used free benchmarks. [4]

The results and details are as below:

- **Ethernet speed:** 100mbps/1GBps
- **Ethernet Switch:** 100mbps/1GBps per connection
- **Router:** 255\*255 simultaneous connections each at 100mbps
- **Server:** Core ix processor 1.9GHz~5GHz
- **Raspberry Pi:** 900MHz quad-core ARM Cortex-A7 CPU, 1GB RAM, GPU: VideoCore IV @ @ 400 MHz

### 4.3 Analysis of Bottlenecks:

#### 4.3.1 Possible Bottlenecks:

- **Raspberry Pi 2 Ethernet connection:** Offers a maximum of 100mbps connection. This bottlenecks all other components, such as the multiple drives attached, the miniDLNA Service, the NAS Service and the Wireless router service, since the max net speed is restricted up to a 100mbps or (10MBps in real time).

- **Raspberry Pi 2 power:**

The Raspberry Pi 2 supplies a limited power to its USB Devices, limiting the capacity of the drives attached to it. If we increase the power limit, the devices and the components on the Raspberry Pi 2 would burn out.

#### 4.3.2 Bypassing the Possible Bottlenecks:

- **Raspberry Pi 2 Ethernet connection:**

Using a USB to Ethernet adapter would give us a theoretical speed of 480mbps (60MBps), which would be enough for our daily needs.

- **Raspberry Pi 2 power:**

Supplying additional power to the USB Devices (e.g.: Commercial Devices such as a USB hub which has external Power source). Care must be taken that the external power source does not have latency, which would possibly cause burn out in the devices.

## 5. Enhancements:

### 5.1 Performance Enhancements:

#### 5.1.1 Use of a Solid Storage Device:

- Since network speeds are low, low access time is of highest priority.
- Along with low power usage, SSD's have much lower access time as compared to traditional Mechanical HDD's.

#### 5.1.2 Use of Class 10 Memory Card or an SSD to Store the OS:

- A Class 10 Memory card greatly improves the Access Speed of the OS. This improves Boot up time.

### 5.2 Longevity Enhancements:

#### 5.2.1 Replaced all Mechanical Drives with Solid State Storage:

- Despite their low price tag, HDDs prove to be expensive in the long term, due to their high speed running mechanical parts, which wear out over time.
- This wear and tear causes HDDs to lose performance over time. To reduce this we use SSD's instead of hard Disk Drives, which have no mechanical storage, and therefore, cause no wear and tear that is noticeable to cause instant failure.
- SSD's generally fail with a lot of warning signs in their SMART data.

#### 5.2.2 Check all the SMART Data Regularly:

- All Storage level deices have a SMART attribute which stores all the reliability information.

#### 5.2.3 Use of Trusted PSU for Dependable Power Supply:

- Major causes of failure of components in this system include:
  - HDD Mechanical Faults.
  - Electrostatic interference.

- Electrical spikes
- Of these, we can minimize the electrical spikes, by using a dependable Power supply.

- A dependable Power supply also reduces the chances of electronic spikes that occur internally during the breaking and closing of a connection, which is common in electronic circuits.

## 6. SUMMARY

### 6.1 End Architecture:

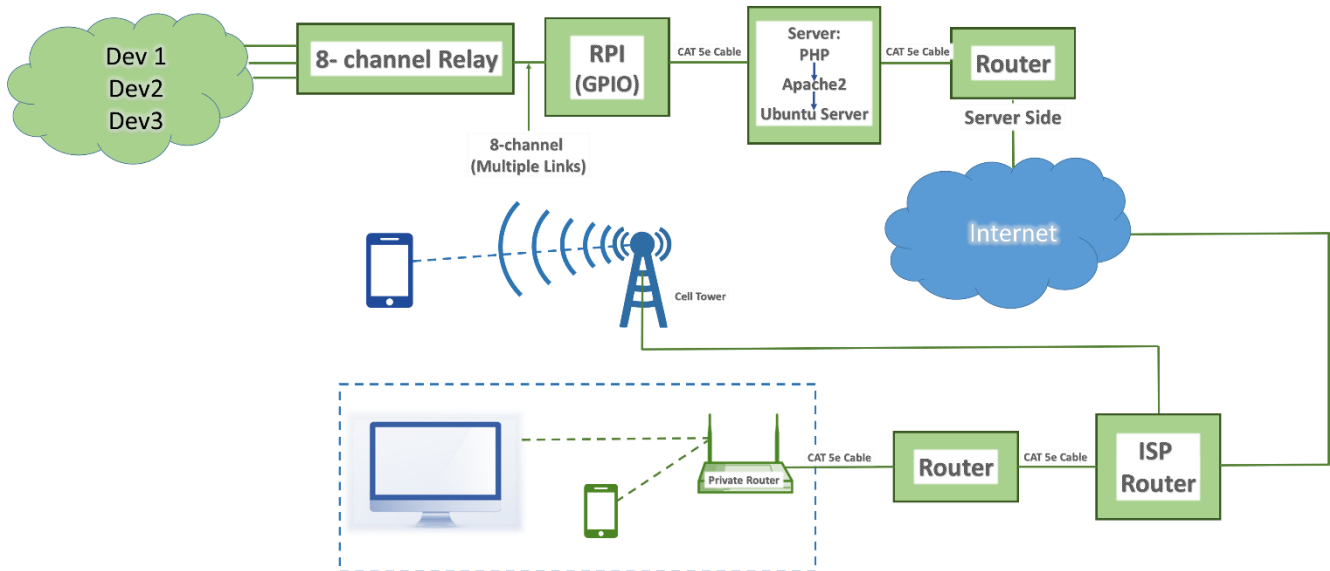


Figure 9: The End Architecture of the system [5] [6] [7]

#### The End Architecture of the system includes:

- Client Side ISP
- Client side Cell Tower:  
For Connection when Client is roaming.
- A Possible CGNAT: Due to the depletion of the IPv4 Addresses, many ISP's are implementing the NAT feature, which saves cost of multiple IP addresses in IPv4 format, but in CGNAT, may cause IPv6 connectivity to be broken (and also resulting in a huge latency).
- Server Side Router: With DMZ Host Enabled: This for directly forwarding incoming requests to the Server, for direct access to the Apache Server.
- The Server: Running PHP code on Apache Server on The Ubuntu Server Desktop OS, by the Ubuntu developers.
- The Raspberry Pi 2: Possibly running its own custom OS, or the Desktop Ubuntu Mate code ported to the MATE Platform for the armfh.
- 8- Channel relay.

## 7. CONCLUSION

Automation of home appliances has taken further step to become popular and being widely adopted and here fits the Raspberry Pi with which we want to customize and configure control of the devices with cheap tools. For Raspberry Pi system there are many option can explore. Automation of home appliances use raspberry pi controller with the help of android interface. It uses WLAN and connects controller to android interface. LAN is limited within the range of Wi-Fi is the limitation of this system. Our proposed wireless based home automation system reduces installation cost and effort, and enhance system flexibility and scalability.

We have designed a very basic application for automation of home appliance with the help of Raspberry Pi through the specific commands for specific operation. The system described allows for a user to control appliances by sending appropriate codes to the system. This could be very difficult if the number of devices increases as the user must either remember or have a catalogue of all the codes for each appliance on him/her all the time. The proposed system effectively removes this obstacle with the help of a custom designed web page that has user friendly button interface to control the appliance. A novel architecture for a home automation system is proposed using the raspberry pi board, Weaved services and electromagnetic relay. Raspberry pi micro-controller board is used to control the switches of appliances through internet. A website has been created using weaved services for acquiring user input. As the GPIO pin of the Raspberry pi activates, relay closes the switch which controls the appliance. The implemented home automation system provides an efficient, flexible and also comfortable user interface for controlling electric appliances from remote location.

## 8. SOCIOECONOMIC IMPACT

The home automation increases efficiency of daily usage home appliances. Their functionality utilized completely. We are mainly concerned with the small offices and homes. They have limited appliances but usability can result in safety, convenience, increased awareness. Home automation also helps to keep look at sensitive areas in home. Using home automation for appliances for example, ACs can be started before you reach home. This can save the time in case of all daily used appliances. In case of small offices only some computers can be kept on and we can check their status from remote place. User can use this automation setup in affordable price. Gaining more internet speed makes homes and small

offices smarter. The scalability of the setup is marginally higher and devices can be connected disconnected as necessary. You also make sure that you are using only resources you need. Raspberry Pi proves to be a smart, economic and efficient platform for implementing the home automation. Its running cost is ~₹130 per year it can be redeemed from. In following way we can calculate average power usage of Raspberry pi 2 B:  $1.8A * 5V = 9W$ . Yearly usage: ₹  $((9W * 10Hrs * 365) / kWh) * 4Rs / Unit = 130Rs$ . Easy implementation and efficient use. The merit of code is that it's generic and flexible. [8] [9] [10]

## 9. ACKNOWLEDGMENTS

With due respect and gratitude we take the opportunity to thank all those who have helped us directly and indirectly. We convey our sincere thanks to Dr. P. N. Mahalle, HoD, Computer Dept. and Prof. A. M. Pawar for their help in selecting the seminar topic and support. Our guide Prof. A. M. Pawar has provided us with immense support and guidance for the same. She has always encouraged us and given us the motivation to move ahead. She has put in a lot of time and effort in this project along with us and given us a lot of confidence. We wish to extend a very big thanks to her for the same. We wish to thank her for the same. Also we wish to thank all the other people who in any smallest way have helped us in the successful completion of this project.

## 10. REFERENCES

- [1] AI-Ali A. R. and AI-Rousan M., "Java-based home automation system", IEEE Transactions on Consumer Electronics, vol. 50, no. 2, pp. 498- 504, 2004.
- [2] Md. Nasimuzzaman Chowdhury, Md. Shiblee Nooman, Srijon Sarker, "Access Control of Door and Home Security by Raspberry Pi Through Internet", International Journal of Scientific & Engineering Research, Volume 4, Issue 11, November-2013 ISSN 2229-5518.
- [3] Kushiro N., Suzuki S., Nakata M., Takahara H. and Inoue M., "Integrated home gateway controller for home energy management system", IEEE International Conference on Consumer Electronics, pp. 386-387, 2003.
- [4] Vladimir Vujović and Mirjana Maksimović, "Raspberry Pi as a Wireless Sensor Node Performances and Constraints".
- [5] Markus Jung, Jurgen Weidinger, Wolfgang Kastner, Alex Olivieri, "Building automation and smart cities: An integration approach based on a service-oriented architecture", The IoT6 project is supported by funding under the Seventh Research Framework Program of the European Union, with the grant agreement FP7-ICT-2011-7-288445.
- [6] Sarthak Jain, Anant Vaibhav, Lovely Goyal, "Raspberry Pi based Interactive Home Automation System through E-mail", 2014 International Conference on Reliability, Optimization and Information Technology ICROIT 2014, India, Feb 6-8 2014.
- [7] Saito T., Tomoda I., Takabatake Y., Ami J. and Teramoto K., "Home Gateway Architecture And Its Implementation", IEEE International Conference on Consumer Electronics, pp. 194-195, 2000.
- [8] Ali M., Vlaskamp J. H. A, Eddiny N. N., Falconer B. and Oramc, "Technical Development and Socioeconomic Implications of the Raspberry Pi as a Learning Tool in Developing Countries", 5th Computer Science and Electronic Engineering Conference (CEEC), pp. 103-108, 2013.
- [9] P. Vigneswari, V. Indhu, R. R. Narmatha, A. Sathinisha and J. M. Subashini, "Automated Security System using Surveillance", International Journal of Current Engineering and Technology E-ISSN 2277 – 4106, P-ISSN 2347 – 5161.
- [10] Cheah Wai Zhao, Jayanand Jegatheesan, Son Chee Loon, "Exploring IOT Application Using Raspberry Pi", International Journal of Computer Networks and Applications Volume 2, Issue 1, January - February (2015).