CLOUD FRAMEWORK FOR ENERGY MANAGEMENT USING AUTOMATED APPLIANCES

A PROJECT REPORT

submitted by

ANISHA S BHAT

KUSHAL MALANI

NITHYA N

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE ENGINEERING

AMRITA SCHOOL OF ENGINEERING, COIMBATORE

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE 641 105

MAY 2013

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF ENGINEERING, COIMBATORE, 641105

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **CLOUD FRAMEWORK FOR ENERGY MANAGEMENT USING AUTOMATED APPLIANCES** submitted by

CB.EN.U4CSE09202 ANISHA S BHAT

CB.EN.U4CSE09227 KUSHAL MALANI

CB.EN.U4CSE09235 NITHYA N

in partial fulfilment of the requirements for the award of the **Degree Bachelo**r **of Technology** in **COMPUTER SCIENCE ENGINEERING** is a bonafide record of the work carried out under our guidance and supervision at Amrita School of Engineering, Coimbatore.

Mr V Ananthanarayanan	**Prof. P.N Kumar**
Project Guide	Chairman
Asst. Professor	Computer Science Engineering
Computer Science Engineering	

This project was evaluated by us on: _______________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We take this opportunity to thank **Br.Abhayamrita Chaitanya**, Chief Operating Officer and **Dr. M.P.Chandrasekaran**, Dean (Engineering) for providing the necessary infrastructure required for the successful completion of the project.

We take this opportunity to thank **Prof. P N Kumar**, Chairperson of the Department of Computer Science Engineering for providing an opportunity to do this project.

We express our sincere thanks and whole hearted gratitude to our Guide **Mr.V.Ananthanarayanan**, Assistant Professor of the Department of Computer Science and Engineering who has been a pillar behind every step in this project.

It would be unfair not thanking our beloved friend, **Showndharyaa M,** for helping us in this project.

We also extend our thanks to all the faculty members and non-teaching staff of our college and our friends whose timely help and cooperation is worth mentioning.

TABLE OF CONTENTS

ABSTRACT

Home automation involves introducing a degree of computerized or automatic control to certain electrical and electronic systems in a building. These include lighting, temperature control, security systems, garage doors, etc. A hardware system is installed to monitor and control the various appliances. The system would control the appliances based on its configuration. For example, it could automatically turn on the lights at a specified time in the evening, or it could measure the ambient light using a hardware sensor and turn on the lights when it grows dark. It can also allow a person to control appliances from a remote location, such as over the internet. For example, one could turn on the air conditioning from the office, before leaving for home.

This project demonstrates a simple home automation system that allows the user to control it with a wireless device such as a Wi-Fi enabled laptop or a high end mobile phone (smart phone). The system allows the user to control each of the lights, fans and power plug points individually. The user can switch On/Off the appliances both locally (i.e. from anywhere in the house using Wi-Fi) and globally (i.e. from anywhere in the world using a mobile phone/laptop and with the help of Internet).

LIST OF ABBREVIATIONS

UWB	:	Ultra Wideband
CPU	:	Central Processing Unit
PDA	:	Personal Digital Assistant
GUI	:	Graphical User Interface
GSM	:	Global System for Mobile Communications
HGW	:	Home Gateway
RSS	:	Recall Support System
SMS	:	Short Messaging Service
GPRS	:	General Packet Radio Service
EDGE	:	Enhanced Data for Global Evolution
IR	:	Infrared rays
GAE	:	Google App Engine
Rest API	:	Representational State Transfer Application Interface
PaaS	:	Platform as a Service
SoC	:	Socket on a Chip
HTTP	:	Hyper Text Transfer Protocol
SSH	:	Secure Shell
WPAN	:	Wireless Personal Area Network
AVR	:	Advanced Virtual RISC
RF	:	Radio Frequency
LAN	:	Local Area Network
SQL	:	Structured Query Language
SOAP	:	Simple Object Access Protocol
XML	:	Extensible Mark-up Language
IP	:	Internet Protocol
TCP	:	Transmission Control Protocol

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 PROBLEM DEFINITION

To develop a cloud framework to automate the electrical appliances at home, shopping malls, theatres, etc. from anywhere in the world using internet

1.2 BACKGROUND

Everything in the world now is connected to each other. This is the information age. There are innumerable devices with varied architecture and design that are communicating to one another. The amount of data that is being processed and transmitted is something which cannot be contained. Such is the power of the Internet.

How would it be if, appliances that you use in your daily life is also connected to one another, how easy would it be to control them, how easy would it be to communicate with them. The realization of such an idea is made possible with advancement in technology.

There are lots of protocols which dictate, how electronic devices should communicate with each other. Finding which protocol suits best, requires remarkably good analysis and modification in the implementation of the protocol on the devices which we want to communicate.

1.3 MOTIVATION

The motivation behind the goal to control home appliances remotely is simple and its four fold:

- It helps you to monitor what is happening inside your home, at any time. The necessity for you to be present inside your home is eliminated. The main use case here is the security that is being provided.
- It helps you to monitor the use of different equipment's and calculate the total energy being spent. The main use-case here is to manage the energy consumption.
- Managing which appliances should be used by who can help prevent catastrophes. The main use case is to provide restriction on the use of certain equipment's (Baby proofing systems)
- The Most important use case is the ease of use that the technology provides. People could control the appliances with a push of the button from their phone or any device linked to the network.

Our project proposes a system that would help achieve all the above requirements using Raspberry Pi arm processor, ZigBee protocols, and Google App Engine. We have used a microprocessor because it consumes less power compared to the Computers. ZigBee has a maximum communication speed of 250kbps. But ZigBee is inexpensive and requires very low power compared with two other families, Bluetooth and UWB.

ZigBee may not be used for communications that require high speed such as for multimedia transmission of voice, video or large data. ZigBee can transmit data over 127 characters (127 bytes) only. ZigBee has a 250 kbps transmission speed and can reduce the burden on host CPUs. When the Bluetooth requires a 32-bit microcomputer, the ZigBee requires only a microcomputer of 4 or 8 bits. In addition, the Bluetooth system initialization takes tens of seconds, while ZigBee requires only 30ms. This is very suitable for sensor devices that require time-speed operation like ON/OFF.

Google App Engine is used as cloud is an emerging technology these days. Moreover cloud offers more security to the data being stored. Cloud has its main role when there are a lot of appliances to be automated and when all their data has to be stored securely. Since the project is based on building a framework, basing all the data on Cloud would be considered as a best method.

Chapter 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE REVIEW

Smart home is a home equipped with special facilities to enable occupants to control or program an array of automated home electronic devices. In this chapter, the different methodologies used in automating home appliances are given.

Automating electrical appliances at home involves a lot of hardware support. Often there is confusion as to which hardware can be considered more efficient for this purpose. In [1], the authors brief on their idea of automating the appliances. Their basic goal was to reduce the consumption of power. They have discussed different hardware techniques for power monitoring, power management, remote power controlling at home, transmission side and also discussed the suitability of ZigBee for required communication link. The authors have identified a way to power on/off the appliances through the internet, PDA using Graphical User Interface (GUI) and through GSM cellular mobile phones.

In [2], the authors address a new smart home control system based on sensor networks to make home networks more intelligent and automatic. Their proposed system can do turn lights off or turn them down based on factors such as occupancy, available daylight and time of day. They have addressed their work using IEEE 802.15.4 and ZigBee and have thereby presented a design of a multi-sensing, heating and the air-conditioning system: a sensor network-based smart light control system for smart home and energy control production.

In [3], the author proposes a new home network system architecture where household appliances and network services work together by moving management functions of Home Gateway (HGW) onto network cloud. In detail, device detection, network routing and security functions are run on HGW and device management, service management and multi-protocol compatibility functions are run on the network cloud. Additionally, security function is added on the network cloud part. The authors have implemented and evaluated a new architecture with Recall Support System (RSS) which supplies services of consumer electronics recall. RSS is a system to detect a recalled household appliance. RSS is a service which needs non-real-time property. Hence the research still has a challenge to design and evaluate the services that need real-time property in proposed system architecture.

In [4], the authors introduce a new mechanism so that the ordinary services of the mobile phones can be leveraged to communicate with and control the home appliances and to make our homes a really smart one. Smart homes become smarter if the controlling can be done from any remote place. Their main focus also was to control the appliances from any remote area. They came to the conclusion of using a mobile phone over internet because mobile phones provide security with the help of strict traffic control whereas internet is a place crowded with various types of traffic. Mobile telephony offers voice and data transfer. It uses SMS, GPRS, and EDGE for the data transfers. The authors have used X10 for the controlling part of the appliances due to its wide availability. X10 comes as a package which has one control module and other appliance modules categorised by their classes such as lights, fans etc.

In [5], the authors proposed a remote appliance control system using a cell phone. The control system was designed to control the units at remote location, using web-enabled interface (GPRS). The authors focussed on the Infrared as a remote appliance control. Though it is extendible to Bluetooth interface, the authors choose not to use them as most of the home appliances are not Bluetooth enabled today (e.g. TV, Refrigerator, Music player, etc.). The user should send an authentication code along with the desired function to his home control system via GPRS. Upon being properly authenticated, the control unit would relay the command through IR port to home appliances. The system controls the home appliances the mobile device and sees the state of home appliance via internet.

Chapter 3

SYSTEM REQUIREMENTS

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 SOFTWARE REQUIREMENTS

- The Linux OS should have python installed in the terminal
- wxPython and wxWidgets
- pySerial should be installed for serial communication
- Google App Engine(python) and Google Data Store
- vncviewer for remote desktop connection
- PUTTY for ssh connection to Raspberry Pi
- Apache Web Server on Raspberry Pi for REST Api Connectivity

3.2 HARDWARE REQUIREMENTS

- Raspberry Pi ARM processor with Debian Wheezy as OS(on a SD Class 4 card)
- Wi-Fi enabled laptop/desktop with Linux OS
- Microprocessor (Number of microprocessor = Number of switch boards)
- ZigBee transceiver (Number of ZigBee = Number of microprocessors + 1)
- Netgear Wi-Fi Dongle
- A USB to Serial Converter
- Power Supply to ZigBee which is connected to the ARM board (~5V)
- Relay Board with number of relays as per requirement
- Samsung Mobile Charger or USB cable to power Raspberry Pi

3.3 DEVELOPMENT TOOLS

A number of different programming tools and languages were used for the development of the various software components of the home automation system. Since the system involves components that run on different platforms such as Laptops, Raspberry Pi board, Google App Engine Server (Cloud) the most suitable language for each platform was chosen.

3.3.1 LINUX OPERATING SYSTEM

We have done all our work on Linux Based Systems(Ubuntu) since Python was programming language used to program devices and it works perfectly well with Linux based Operating Systems.

3.3.2 GOOGLE APP ENGINE (CLOUD)

Google App Engine (often referred to as GAE or simply App Engine) is a platform as a service (PaaS) cloud computing platform for developing and hosting web applications in Google-managed data centers. Applications are sandboxed and run across multiple servers. App Engine offers automatic scaling for web applications- as the number of requests increases for an application, App Engine automatically allocates more resources for the web application to handle the additional demand.

Google App Engine is free up to a certain level of consumed resources. Fees are charged for additional storage, bandwidth, or instance hours required by the application. App Engine is designed in such a way that it can sustain multiple data center outages without any downtime. Compared to other scalable hosting services, App Engine provides more infrastructures to make it easy to write scalable applications, but can only run a limited range of applications designed for that infrastructure.

3.3.3 GOOGLE DATASTORE

Datastore is for storing data on the Google's Cloud App Server. This comes as a bundled package with Google cloud server and different databases can be created by creating different models in the python script.

3.3.4 RASPBERRY PI (ARM PROCESSOR)

The Raspberry Pi is a credit-card-sized single-board computer which has a Broadcom BCM2835 system on a chip (SoC) which includes an ARM1176JZF-S 700 MHz processor.

It does not include a built-in hard disk or solid-state drive, but uses an SD card for booting and long-term storage. It has two models called as Model A and Model B. Model A has one USB port and no Ethernet controller, and Model B has two USB ports and a 10/100 Ethernet controller.

We use Raspberry Pi (Model B), an Arm 11 processor that helps in controlling appliances in home both locally (via Wi-Fi connectivity) and globally(via http rest API connectivity). We installed Debian "Wheezy" operating system on a class 4 SD card and we can connect it wirelessly using a Wi-Fi dongle and operate locally. To store data, we have to partition the SD card.

3.3.5 APACHE WEB SERVER

Apache Web Server is installed on Raspberry Pi which helps to make an http connection from the Google App Server.

3.3.6 PUTTY SSH

PUTTY is a free and open source terminal emulator application which can act as a client for the SSH, Telnet, rlogin, and raw TCP computing protocols and as a serial console client. This helps in connecting to Raspberry Pi by using the board's static IP.

3.3.7 VNCSERVER

Vncserver is a remote display daemon that allows users to run totally parallel sessions on a machine which can be accessed from anywhere. All applications running under the server continue to run, even when the user disconnects. Vncserver runs on multiple different ports.

3.3.8 VNCVIEWER

The VNC client (or viewer) is the program that watches, controls, and interacts with the server. The client controls the server. User needs to enter the board static IP address and the server's port number and the password to control the server remotely.

3.3.9 wxPython and wxWidgets

wxPython is a wrapper for the cross-platform GUI API (often referred to as a 'toolkit') wxWidgets (which is written in C++) for the Python programming language. It is implemented as a Python extension module (native code). In other words, wxWidgets helps in designing the GUI whereas wxPython helps in programming the application.

3.3.10 ZIGBEE TRANSCEIVERS

ZigBee is a specification for a suite of high level communication protocols used to create personal area networks built from small, low-power digital radios. ZigBee is based on an IEEE 802 standard. Though low-powered, ZigBee devices often transmit data over longer distances by passing data through intermediate devices to reach more distant ones, creating a mesh network.

ZigBee is used in applications that require a low data rate, long battery life, and secure networking. The technology defined by the ZigBee specification is intended to be simpler and less expensive than other WPANs, such as Bluetooth or Wi-Fi. ZigBee networks are secured by 128 bit encryption keys.

3.3.11 RELAYS

A relay is an electrically operated switch. Relays are used where it is necessary to control a circuit by a low-power signal (with complete electrical isolation between control and controlled circuits), or where several circuits must be controlled by one signal.

3.3.12 MICROCONTROLLER (ARDUINO)

Arduino is a single-board microcontroller designed to make the process of using electronics in multidisciplinary projects more accessible. The hardware consists of a simple open source hardware board designed around an 8-bit Atmel AVR microcontroller, though a new model has been designed around a 32-bit Atmel ARM. The software consists of a standard programming language compiler and a boot loader that executes on the microcontroller.

Chapter 4

SYSTEM ANALYSIS

CHAPTER 4

SYSTEM ANALYSIS

4.1 PROBLEM STATEMENT

With advancement of technology things are becoming simpler and easier for us. Automation is the use of control systems and information technologies to reduce the need for human work in the production of goods and services. In the scope of industrialization, automation is a step beyond mechanization. Whereas mechanization provides human operators with machinery to assist them with the muscular requirements of work, automation greatly decreases the need for human sensory and mental requirements as well. Automation plays a very important role in the world economy and in daily experience.

Home automation is the control of any or all electrical devices in our home, whether we are there or away. It is one of the most exciting developments in technology for the home that has come along in decades. There are hundreds of products available today that allow us to control over the devices automatically, either by remote control; or even by voice command.

4.2 NEED FOR AUTOMATION

Earlier, we looked into the face of future when we talked about automated devices, which could do anything on instigation of a controller, but today it has become a reality. An automated device can replace good amount of human working force, moreover humans are more prone to errors and in intensive conditions the probability of error increases whereas, an automated device can work with diligence, versatility and with almost zero error.

4.3 EXISTING SYSTEMS

There are different methods in which one can automate appliances at home or office, which are explained in brief below:

4.3.1 USING MICROCONTROLLER WITH BLUETOOTH

This construction has three important parts:

- An electrical circuit with relays to control the connected device
- A microcontroller board with Bluetooth module
- A PC to send commands to the microcontroller over Bluetooth

This project was implemented using a Dwengo board and a Bluetooth module. This board can be easily connected to any computer using a serial cable, USB or a Bluetooth module. On the PC side, tools such as Tera Term Pro or Minicom has been used to send instructions over Bluetooth (or any other serial connection) using an operating system. The program code has been written using Java Programming Language.

4.3.2 USING RF CONTROLLED REMOTE

This project was implemented to automate home appliances using RF controlled remote. The conventional wall switches located in different parts of the house makes it difficult for the user to go near them to operate. RF itself has become synonymous with wireless and high frequency signals, describing anything from AM radio to computer local area networks (LANs) at 2.4 GHz. An RF signal is an electromagnetic wave that communication systems use to transport information through air from one point to another. To achieve the remote controlled home automation system with RF technology, an RF remote is interfaced to the microcontroller on the transmitter side which sends ON/OFF commands to the receiver where loads are connected. By operating the specified remote switch on the transmitter, the loads can be turned ON/OFF remotely through wireless technology. The microcontroller used is of 8051 family.

4.3.3 USING ARDUINO AND ZIGBEE

In this project, each smart device has a ZigBee Interface to talk to their Gateway. The Gateway of the smart building will have Small Energy Meter integrated with ZigBee to talk to the Smart devices and Wi-Fi to talk to the Smart Energy Management System. It aggregates data from buildings and transmits it to the Control System and sends commands for binary ON/OFF activation schemes of the various Smart Appliances based on the commands issued by the Smart Energy Management System. The server in the power house collects information from all the gateways. The server is enabled with Wi-Fi or the GSM networks based on the requirement. The server examines the current status and runs a scheduling algorithm and sends commands to the gateway nodes that are deployed all over.

4.3.4 TOUCH SCREEN BASED CONTROL

Here, a touch panel is interfaced to the microcontroller on the transmitter side which sends ON/OFF commands to the receiver where loads are connected. By touching the specified portion on the touch screen panel, the loads can be turned ON/OFF remotely through wireless technology. The microcontroller used here is of 8051 family. The loads are interfaced to the microcontroller using opto-isolators and triacs.

GSM modem has also been interfaced to the control unit. Using GSM modem, the user can control home appliances by sending an SMS. Advantage of this technology is that there is no range limitation when compared to the RF technology.

4.4 PROPOSED SYSTEM

The proposed system architecture can be divided into 8 modules as depicted in the following figures. A brief description of each module is given below.

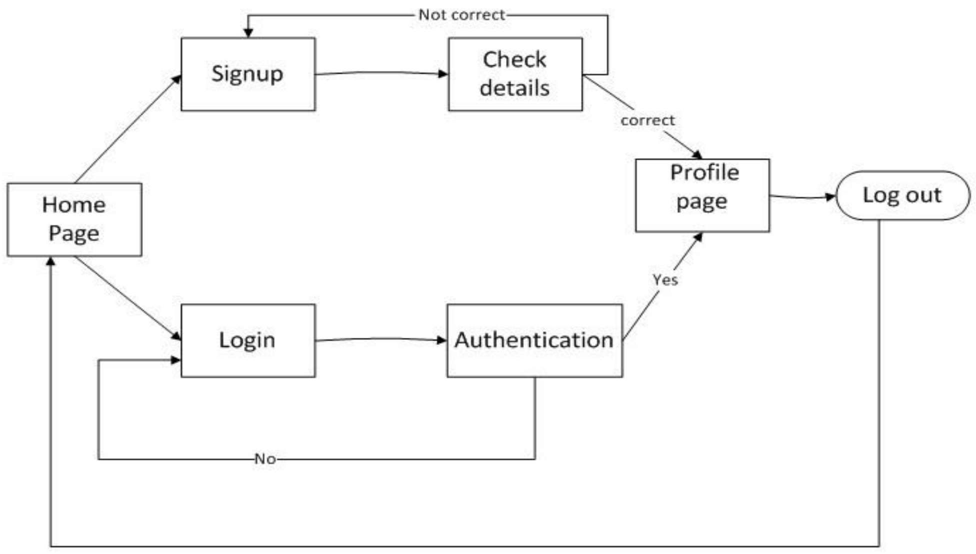


Fig 4.1

The user/house owner should have a user account to access the appliances at his house. Fig 4.1 shows the module of creating a user profile.

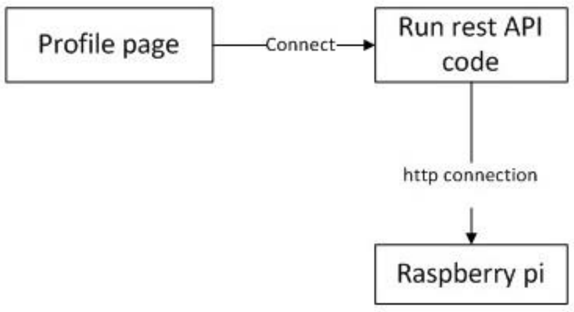


Fig 4.2

The user profile created using Google App Engine will be connected to the Raspberry Pi using RESTful API. This module is illustrated in Fig 4.2

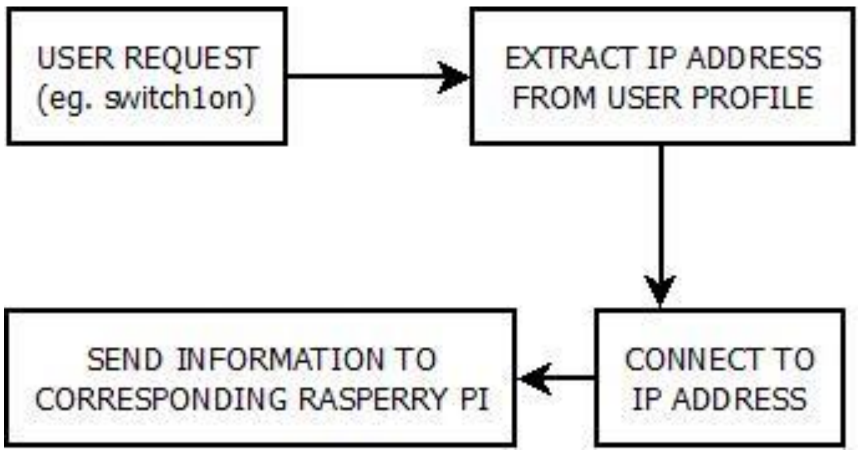

USER REQUEST (eg. switch1on)
EXTRACT IP ADDRESS FROM USER PROFILE
CONNECT TO IP ADDRESS
SEND INFORMATION TO CORRESPONDING RASPERRY PI

Fig 4.3

When the user sends a request, the IP address of the user is fetched from the user profile. Using this IP address, the application will be able to send the request to the Raspberry Pi installed at the user's home.

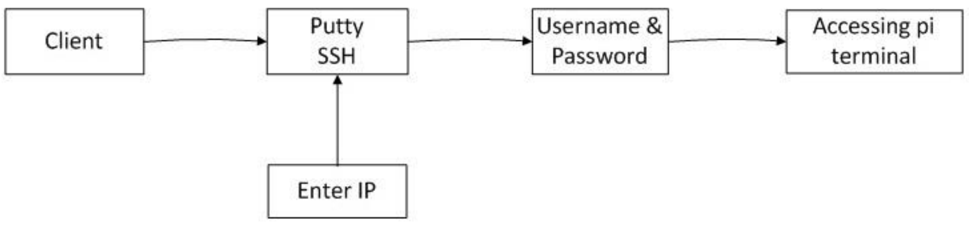


Fig 4.4

The client can access the Raspberry Pi terminal using an SSH connectivity. The user will be prompted to give the username and password. On authentication, the user will be able to access the Pi board.

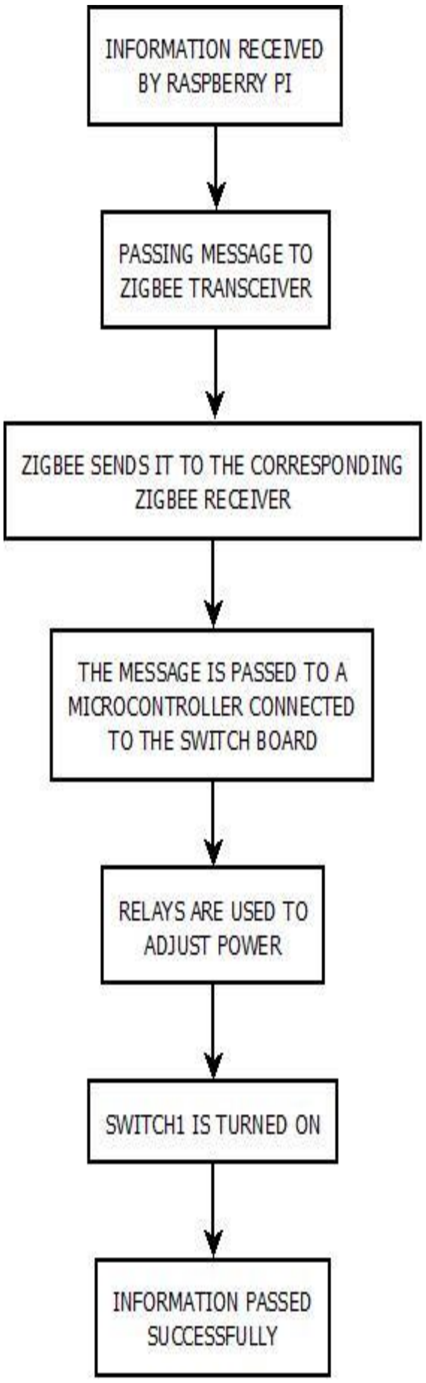

INFORMATION RECEIVED BY RASPBERRY PI
PASSING MESSAGE TO ZIGBEE TRANSCEIVER
ZIGBEE SENDS IT TO THE CORRESPONDING ZIGBEE RECEIVER
THE MESSAGE IS PASSED TO A MICROCONTROLLER CONNECTED TO THE SWITCH BOARD
RELAYS ARE USED TO ADJUST POWER
SWITCH1 IS TURNED ON
INFORMATION PASSED SUCCESSFULLY

Fig 4.5

The information received by the Raspberry Pi is passed on to the ZigBee transceiver. Serial communication between ZigBee's takes place. The other ZigBee gets the information and passes on to the Microcontroller which then passes it to the Switches through the Relay.

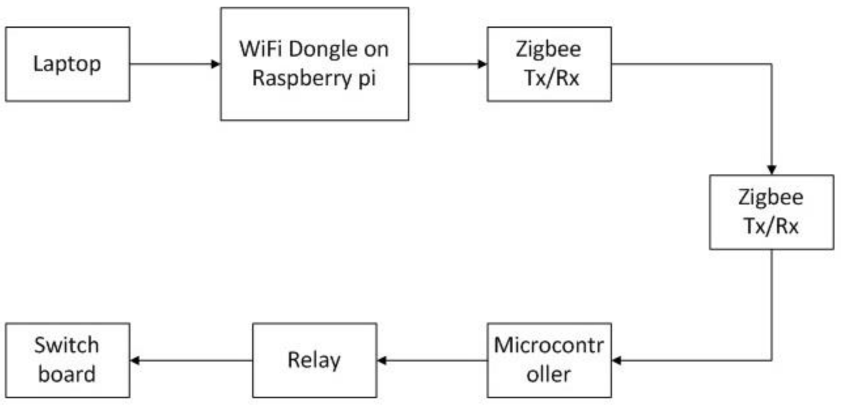


Fig 4.6

To locally connect and control the appliances at home, the laptop/mobile can be connected to the Wi-Fi Dongle on the Raspberry Pi. And the information gets passed from there on. This is represented in the Fig 4.6

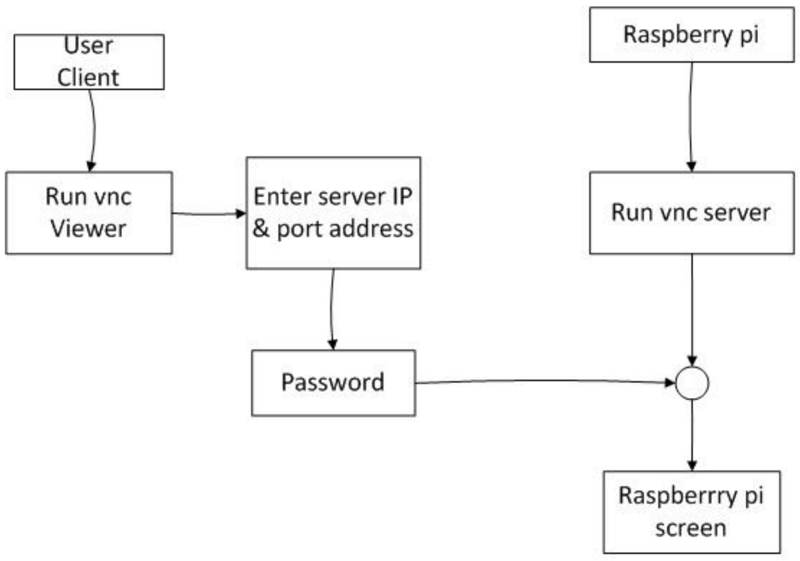


Fig 4.7

To control Raspberry Pi from the Desktop interface, we can have a remote desktop connection using vnc by running vnc server on Raspberry Pi and vncviewer on the user's laptop.

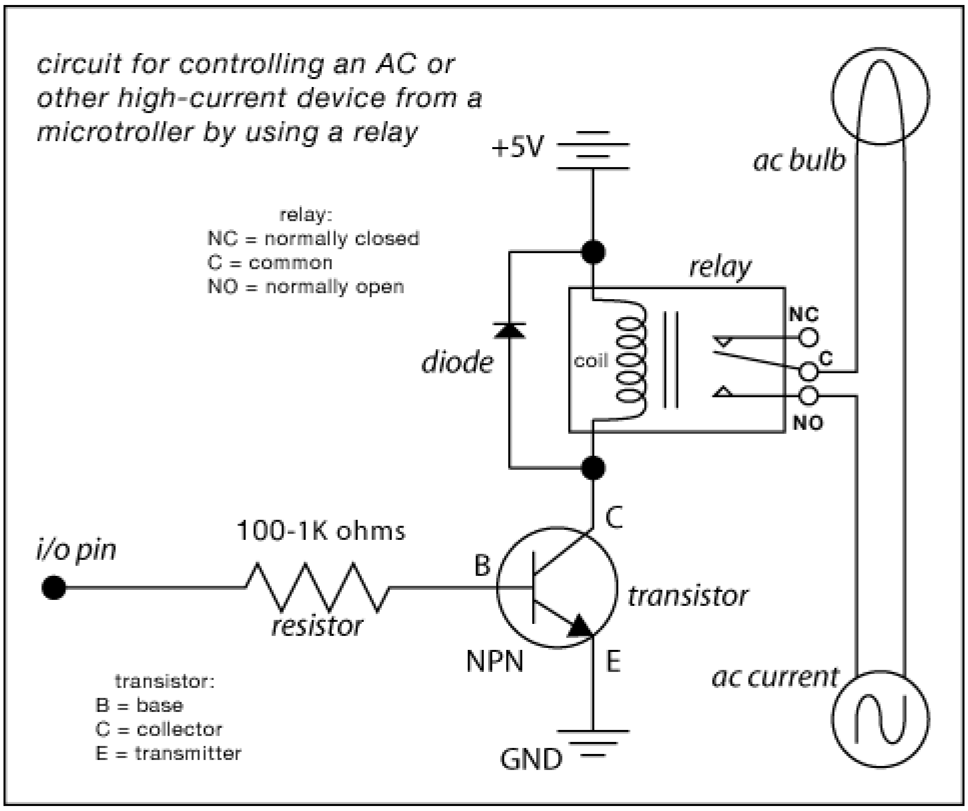


Fig 4.8

Relays are circuits for controlling the AC from a microcontroller and convert them to a power which can actuate the appliance. The concept of Relays will be explained in detail in the next chapter. The Fig 4.8 shows the circuit diagram of a relay.

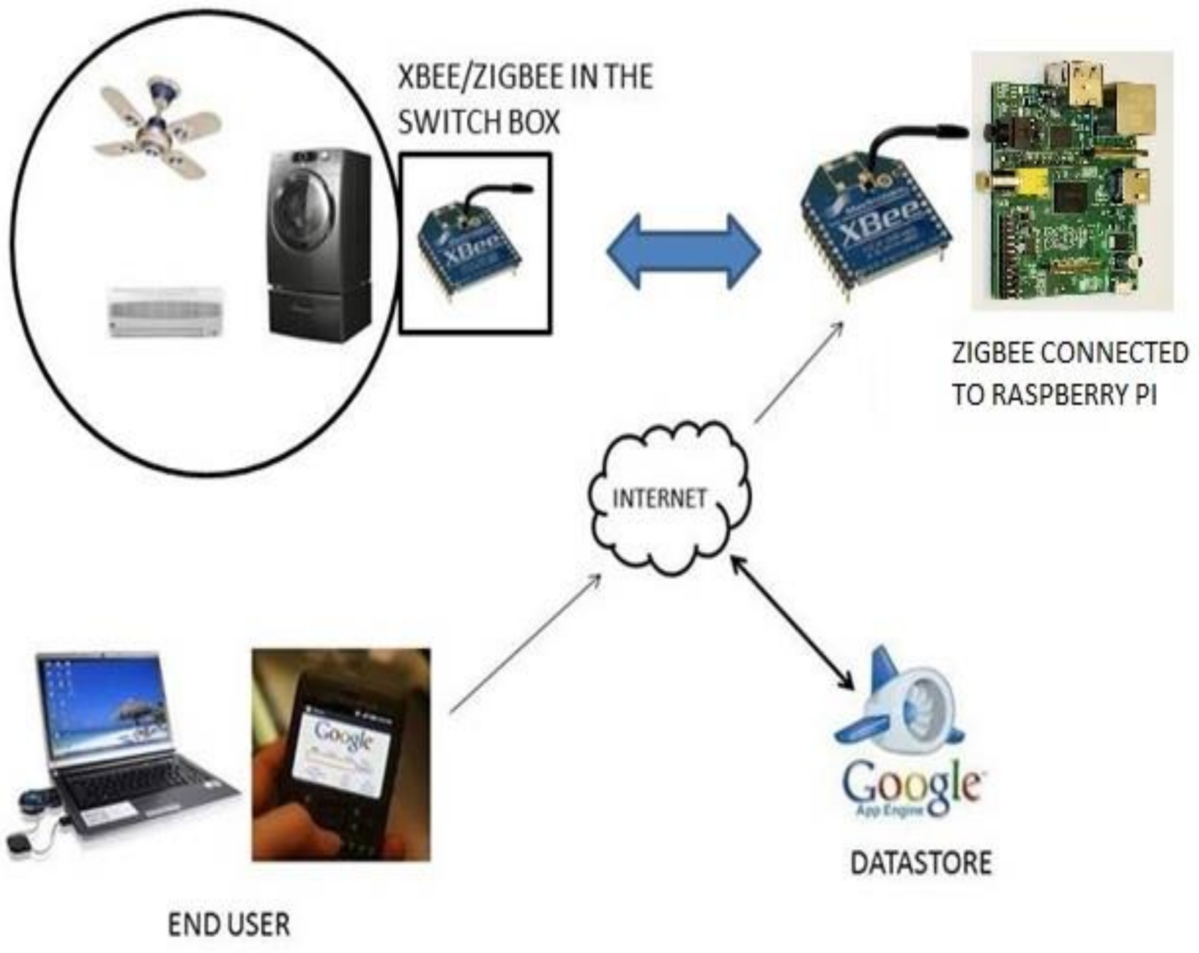
Chapter 5

SYSTEM DESIGN

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE



XBEE/ZIGBEE IN THE SWITCH BOX
ZIGBEE CONNECTED TO RASPBERRY PI
INTERNET
END USER
Google App Engine
DATASTORE

Fig 5.1

5.2 DATA FLOW DIAGRAM

5.2.1 LEVEL 0

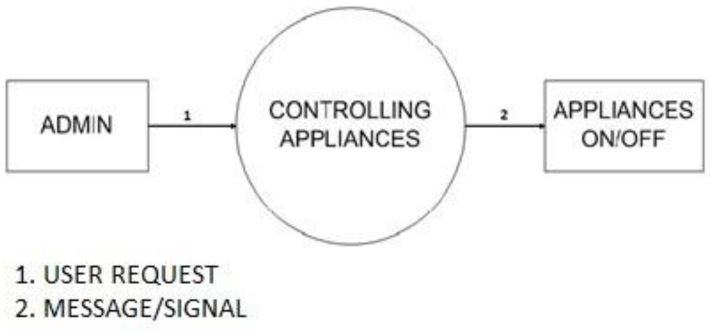


Fig 5.2

5.2.2 LEVEL 1

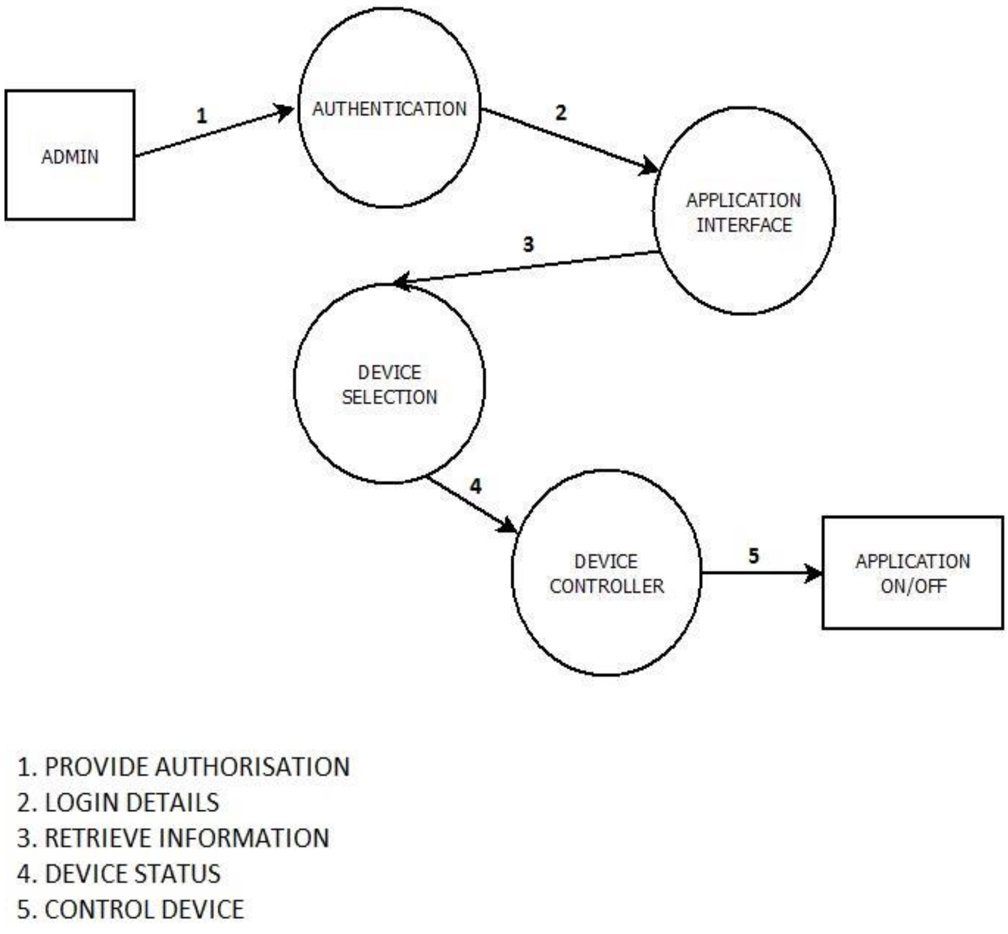
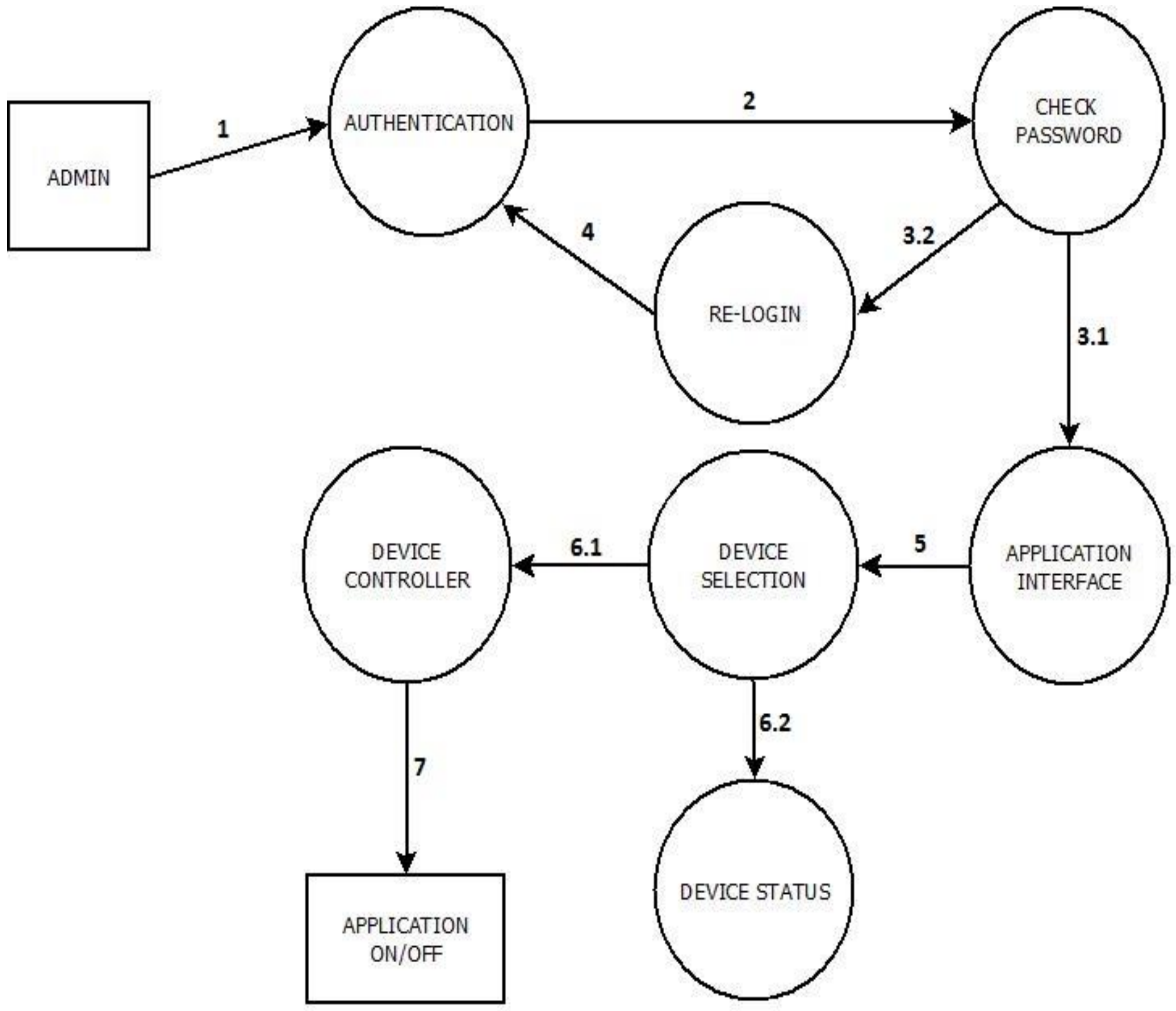


Fig 5.3

5.2.3 LEVEL 2



1. PROVIDE AUTHORISATION
2. LOGIN DETAILS

3.1 PASSWORD CORRECT

3.2 PASSWORD INCORRECT

4. ENTER CORRECT LOGIN DETAILS
5. APPLIANCE DETAILS

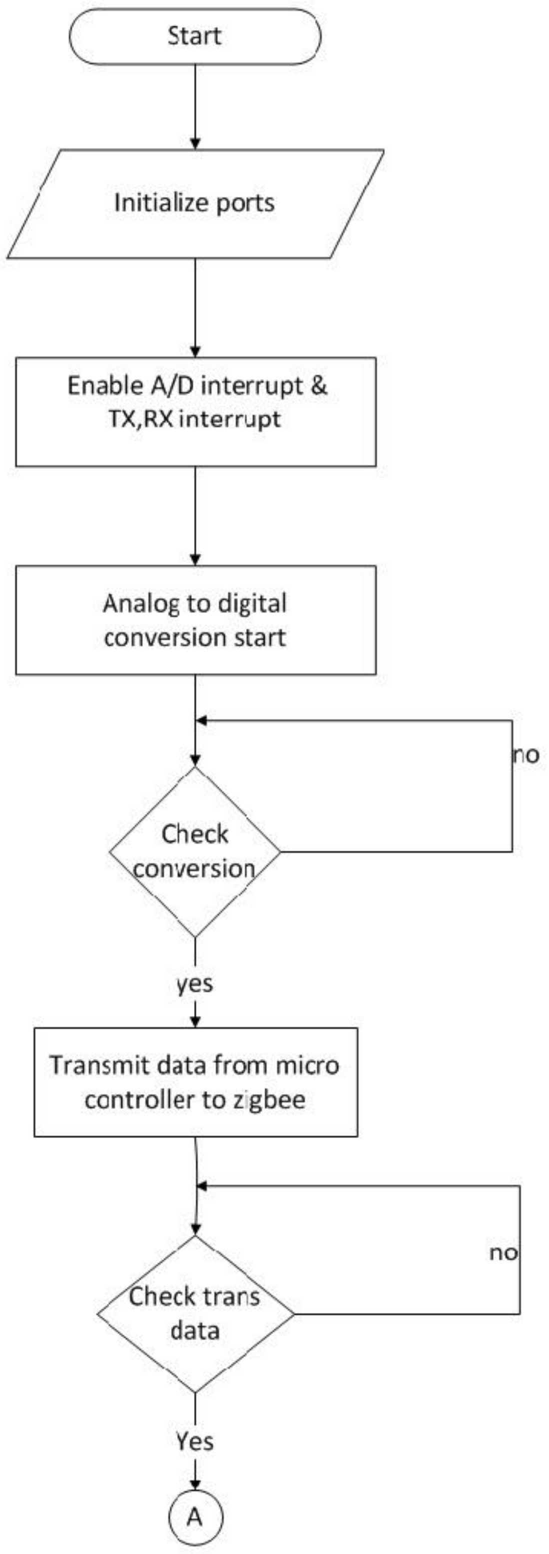
6.1 SELECT APPLIANCE

6.2 REQUEST FOR DEVICE STATUS

7. CONTROL APPLIANCES

Fig 5.4

5.3 EMBEDDED FLOW CONTROL



Start
Initialize ports
Enable A/D interrupt & TX,RX interrupt
Analog to digital conversion start
no
Check conversion
yes
Transmit data from micro controller to zigbee
no
Check trans data
Yes
A

Fig 5.5

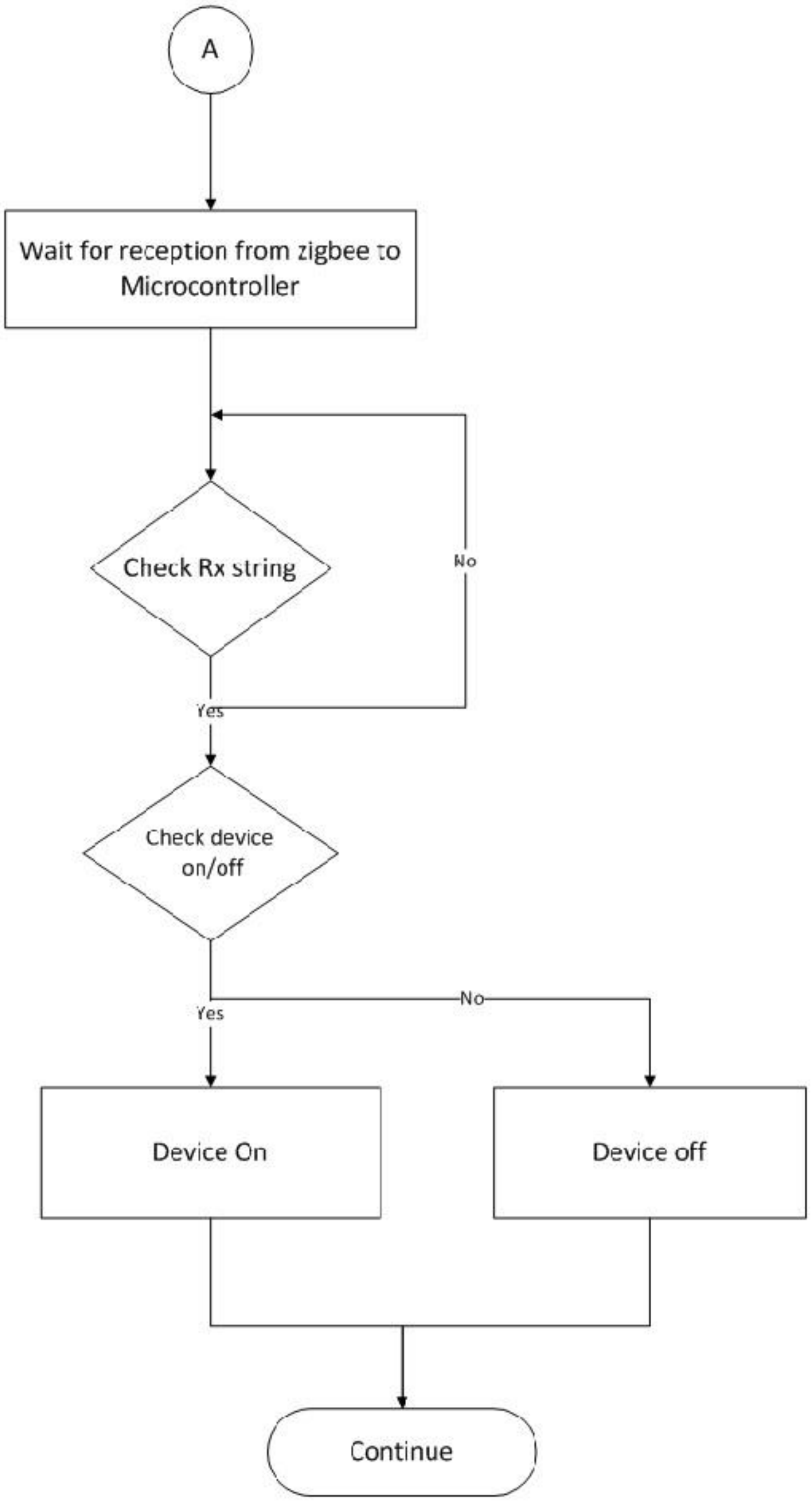

A
Wait for reception from zigbee to Microcontroller
Check Rx string
No
Yes
Check device on/off
Yes
No
Device On
Device off
Continue

Fig 5.6

5.4 USE CASE DIAGRAM

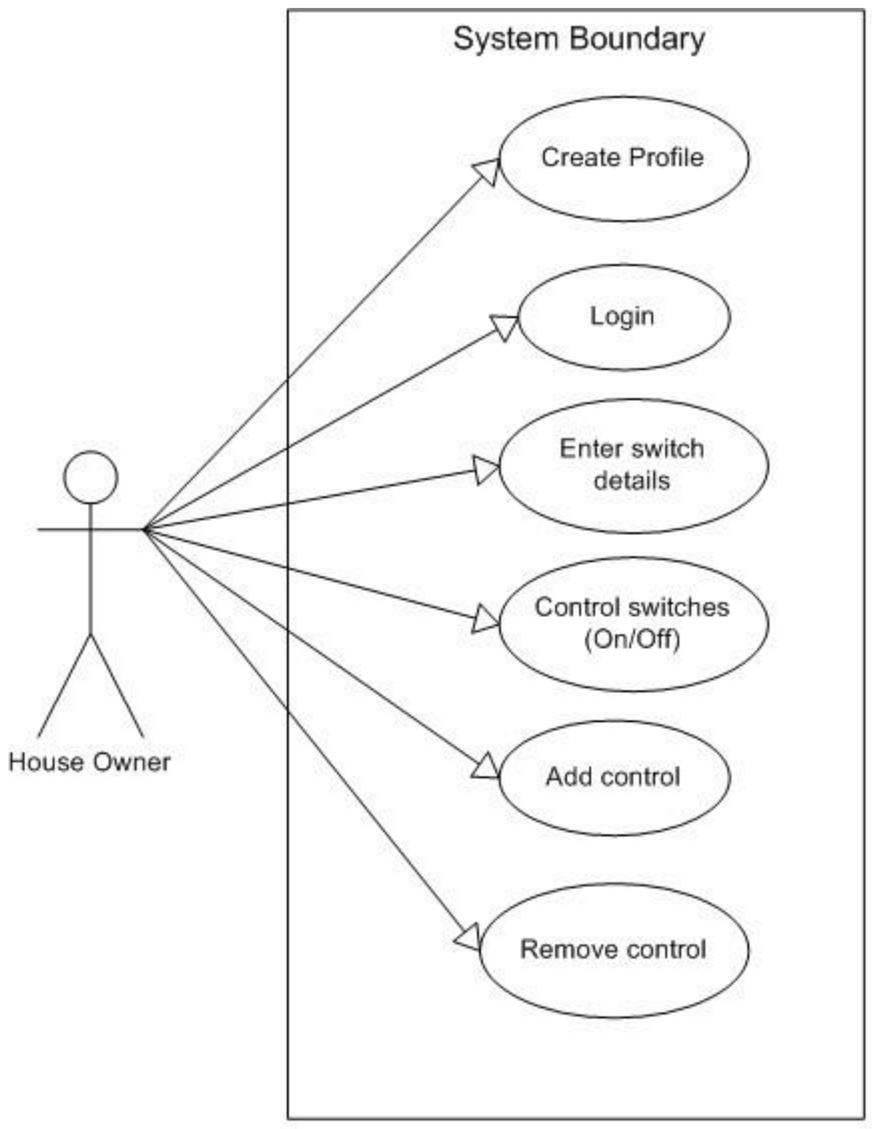


Fig 5.7

Chapter 6

IMPLEMENTATION DETAILS

CHAPTER 6

IMPLEMENTATION DETAILS

6.1 INPUT CONSIDERATION

User Signs up and logs in from the interface and connects to the Raspberry Pi board at home via Google app engine. She/he can control the switches to save unwanted wastage of energy.

6.2 OUTPUT CONSIDERATION

On receiving an operation (On/Off) message, the Raspberry Pi board sends signals to the ZigBee attached to the microcontroller, which in turn communicates with the Relays attached to the switch board. This communication controls the ON/OFF of switches.

6.3 GOOGLE APP ENGINE

Google App Engine runs web applications on Google's infrastructure. App Engine applications are easy to build, easy to maintain, and easy to scale as your traffic and data storage needs grow. With App Engine, there are no servers to maintain: Just upload the application, and it's ready to serve the users.

Google App Engine makes it easy to build an application that runs reliably, even under heavy load and with large amounts of data. App Engine includes the following features:

- Dynamic web serving, with full support for common web technologies
- Persistent storage with queries, sorting and transactions
- Automatic scaling and load balancing

- APIs for authenticating users and sending email using Google Accounts
- A fully featured local development environment that simulates Google App Engine on your computer
- Task queues for performing work outside of the scope of a web request
- Scheduled tasks for triggering events at specified times and regular intervals

Google app engine provides support for the application, so that it can be run in the Python environment (Python 2.7). Each environment provides standard protocols and common technologies for web application development.

The next important part is the Datastore. App Engine provides a distributed NoSQL data storage service that features a query engine and transactions. Just as the distributed web server grows with your traffic, the distributed datastore grows with your data. You have the choice between two different data storage options differentiated by their availability and consistency guarantees.

The App Engine datastore is not like a traditional relational database. Data objects, or "entities," have a kind and a set of properties. Queries can retrieve entities of a given kind filtered and sorted by the values of the properties. Property values can be of any of the supported property value types.

Datastore entities are "schemaless." The structure of data entities is provided by and enforced by the application code. The Python datastore interface includes features for applying and enforcing structure within the app. The app can also access the datastore directly to apply as much or as little structure as it needs.

6.4 REST API (Representational State Transfer)

A RESTful web API (also called a RESTful web service) is a web API implemented using HTTP and the principles of REST. It is a collection of resources, with four defined aspects:

- Base URI for the web API, such as http://example.com/resources/
- Internet media type of the data supported by the web API. This is often JSON but can be any other valid Internet media type provided that it is a valid hypertext standard.
- Set of operations supported by the web API using HTTP methods (e.g., GET, PUT, POST, or DELETE).
- The PUT and DELETE methods are idempotent methods. The GET method is a safe method (or nullipotent), meaning that calling it produces no side-effects.
- Hypertext driven.

6.4.1 ARCHITECTURE

REST-style architectures consist of clients and servers. Clients initiate requests to servers; servers process requests and return appropriate responses. Requests and responses are built around the transfer of representations of resources. A resource can be essentially any coherent and meaningful concept that may be addressed. A representation of a resource is typically a document that captures the current or intended state of a resource.

REST facilitates the transaction between web servers by allowing loose coupling between different services.

6.4.2 BENEFITS OF USING REST

REST is less strongly typed than its counterpart, SOAP. The REST language uses nouns and verbs, and has an emphasis on readability. Unlike SOAP, REST does not require XML parsing and does not require a message header to and from a service provider. This ultimately uses less bandwidth. REST error-handling also differs from that used by SOAP.

6.4.3 KEY GOALS OF REST

- Scalability of component interactions
- Generality of interfaces
- Independent deployment of components
- Intermediary components to reduce latency, enforce security and encapsulate legacy systems.

REST has been applied to describe the desired web architecture, to help identify existing problems, to compare alternative solutions, and to ensure that protocol extensions would not violate the core constraints that make the Web successful.

6.5 PROCEDURE

6.5.1 SET UP

User Signs up using his/her raspberry pi's static IP. All these details are stored in the Google Data store on the Google Cloud App Server. From the Profile page, when a User presses the connect button, the app establishes a connection to the Raspberry pi Board. For the connection we can use a simple TCP Socket connection.

Raspberry Pi has MYSQL installed on it thus we create a switches database here. Whenever a User request for the status of the switches the Pi gets the information from the Relays on the ZigBee Network and Sends it back to the interface via HTTP connection. A ZigBee Trans-receiver (Both sends and receives information) is attached to the Raspberry Pi and the microcontroller for converting messages in signal form.

User can connect to the Raspberry Pi locally (without the requirement of Internet) by connecting to the attached Wi-Fi Dongle (helps for wireless connectivity) on Raspberry pi and making an SSH (using Putty) for terminal or Remote Desktop Connection (using vnc) for desktop interface to control switches.

6.5.2 ISSUE FACED

The issue is Google App Engine allows inbound socket connection (where the App Engine is made as a client) only for paid users and hence this wouldn’t be possible for connectivity

6.5.3 SOLUTION (An intelligent one!)

We have made Raspberry Pi as an HTTP Web Server by installing Apache Server on it. Thus by using a REST (Representative State Transfer) API code on Google App Engine we can make an HTTP connection to the Raspberry Pi. All messages for controlling the switches are sent on this HTTP connection. A set of operations are supported by the web API using HTTP methods (e.g., GET, PUT, POST, or DELETE).

Resource	GET	PUT	POST	DELETE
Collection URI, such ashttp://example.com/resources/	**List** the URIs and perhaps other details of the collection's members.	**Replace** the entire collection with another collection	**Create** a new entry in the collection	**Delete** the entire collection
Element URI, such ashttp://example.com/resources/item17	**Retrieve** a representation of the addressed member of the collection, expressed in an appropriate Internet media type.	**Replace** the addressed member of the collection, or if it doesn't exist, **create** it.	Not generally used. Treat the addressed member as a collection in its own.	**Delete** the addressed member of the collection

Table 6.1

Chapter 7

TESTING

CHAPTER 7

TESTING

Each of the modules was tested using both black box and white box testing techniques. Black box testing ensures the correctness of the system's output, given various possible inputs. White box testing involves providing calculated inputs so that the internal structure of the system can be verified to be error free. For example, white box testing of a software module involves providing different inputs that test all possible code paths within the program.

The modules were designed keeping in mind all possible inputs and any erroneous inputs as well. Any exceptions that occur are handled appropriately without compromising the stability of the system.

Different issues, both foreseen and unforeseen, were encountered during the design process. These issues and the measures that were adopted to overcome them are described in this chapter.

7.1 ZIGBEE – ZIGBEE COMMUNICATION

Three switch boxes were configured with relays and ZigBee's and were placed in corners of a large room. The Raspberry Pi and the ZigBee transmitter are placed almost in the centre of the room. It was found that certain strings were not being received by the ZigBee's in the switch boxes. This could be because of various factors like distance, obstacles, and other disturbances. In our case, Distance was found to be the main factor. The Switch boxes were later moved a little closer to the transmitter and the communication was found to be perfect. Also, it was found that the distance between the transmitter and receiver ZigBee should not be more than 15 m approximately.

7.2 Wi-Fi CONNECTIVITY

The Wi-Fi Dongle is connected to the Raspberry Pi on one of the USB ports. A Wi-Fi session is made active using this. Any user can connect to this Wi-Fi using a laptop. When trying to automate the appliances locally, the laptop is connected to the Raspberry Pi using the Wi-Fi. Once it is connected, the home user will be able to switch On/Off the devices. While testing, the Wi-Fi signal was found to be very strong and we were able to access it from anywhere in that same floor (i.e.) the distance was approximately found to be <100 m.

7.3 GLOBAL CONNECTIVITY

Global connectivity is when we try to connect to our Raspberry Pi from anywhere in the world. This could be done if both the user and the board have internet connectivity. The GUI is designed using python and the user details are stored in Google Data Store, which is a popular functionality of Google App Engine. Socket Programming in python was done in order to enable global connectivity. But recently, App Engine has blocked this feature to free customers. So, one has to be a paid customer to get access to this service. Hence we had come up with another solution of using http connection. This could be done using REST Api.

7.4 HARDWARE MODULE

7.4.1 QUALITY OF MAINS SUPPLY

The electromagnetic relays require a stable +12V DC supply in order to function reliably. Initially a transformer with a rectifier was used to provide a +12V DC supply from the mains. However the quality of the mains supply is unreliable because at times the supply is at a low voltage instead of 220V. This was causing the transformer to output only 9-10V and some of the relays would not activate. To compensate for this, a transformer with a higher output voltage of +16V is used. The transformer output is connected to a 7812 positive voltage regulator IC to produce a +12V DC for the relays. Similarly the ZigBee would require only a +5V DC and a similar circuit is used to provide the same.

7.5 UNIT TESTING

The following diagram shows the flow when a user chooses to switch on switch 1.

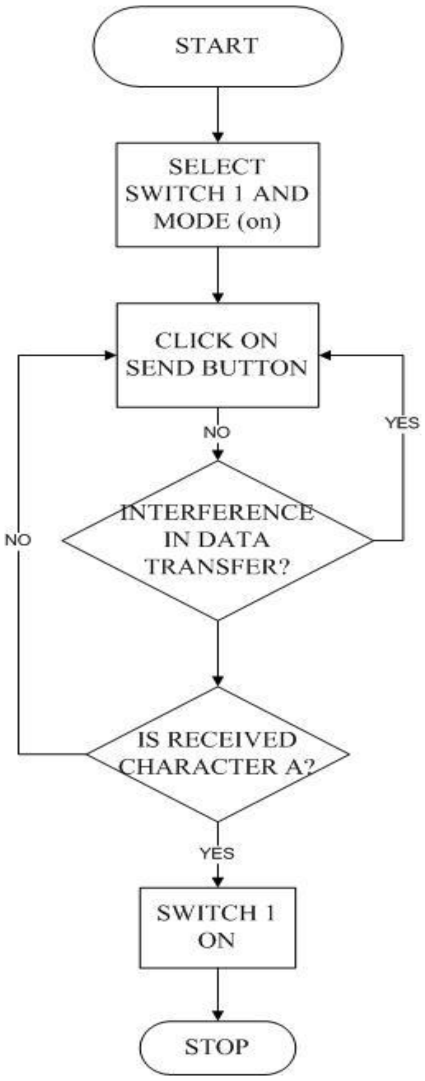


Fig 7.1

7.5.1 PATHS

Interference: (YES)

Start → Select Switch 1 and ON → Click on Send button → Interference in data transfer → click on send again until it gets delivered.

Interference: (NO) and Character received is ‘A’: (NO)

Start → Select Switch 1 and ON → Click on Send button → Interference in data transfer → Is character received ‘A’→ Click on send button again.

Interference: (NO) and Character received is ‘A’: (YES)

Start → Select Switch 1 and ON → Click on Send button → Interference in data transfer → Is character received ‘A’→ Switch 1 is turned ON → Stop.

7.5.2 DATA FLOW TESTING

` The values of each variable is checked for consistency, proper format along with flow of control and found to be correct.

7.6 INTEGRATION TESTING

All the modules that had been subject to unit testing are integrated to ensure that each module works properly in coordination with others. The modules were tested at each level of integration.

7.7 SYSTEM TESTING

System Testing is the process of checking if the developed software is working according to the original objectives and requirements.

7.8 VALIDATION

Validation testing succeeds when software functions in a manner that can be reasonably expected by the customer. The project was given to third parties and they validated the project conforming to the requirements.

Chapter 8

CONCLUSION

CHAPTER 8

CONCLUSION

8.1 CONCLUSION

The home automation system has been experimentally proven to work satisfactorily by connecting sample appliances to it and the appliances were successfully controlled from a wireless mobile device (laptop).

Thus a low-cost home automation was successfully designed, implemented and tested.

8.2 FUTURE ENHANCEMENTS

A useful feature would be to add support for remotely controlling an appliance using a smart phone. Almost everyone has a smart phone these days. An android application can be created which gives the user the facility to login through his phone and access his appliances.

Another idea is to control the appliances using voice commands. A voice recognition system can be developed through which one can read out the commands loud and the corresponding actions will be performed.

Also sensors can be installed on the appliances which will help us automate them more accurately. For e.g. the room temperature can be detected using the sensor which will prompt the user to switch on the AC if the temperature is high.

Chapter 9

REFERENCES

CHAPTER 9

REFERENCES

[1] N. Javaid, A. Sharif, A. Mahmood, S. Ahmed, U. Qasim, Z. A. Khan, “Monitoring and Controlling Power using Zigbee Communications”, arXiv-Cornell University, Vol.1, No.1, August 2012

[2] Dae-Man Han and Jae-Hyun Lim, “Smart Home Energy Management System using IEEE 802.15.4 and Zigbee”, IEEE Transactions on Consumer Electronics, Vol.56, No.3, August 2010

[3] Katsuya Suzuki and Masahiro Inoue, “Home network with Cloud Computing for Home Management”, IEEE 15th International Symposium on Consumer Electronics, June 2011

[4] Rifat Shahriyar, Enamul Hoque, S.M. Sohan, Iftekhar Naim, Md. Mostafa Akbar, Masud Karim Khan, "Remote Controlling of Home Appliances using Mobile Telephony“, International Journal of Smart Home, Vol. 2, No. 3, July 2008

[5] Sachin Sharma, Gaurav Chitranshi, Basanta Mahato, Atul Kumar Srivastava, “ Control of Home-Appliances through IR interface using web (GPRS) enabled Mobile Phones “, International Journal of Advanced Engineering Sciences and Technologies, Vol.6, No. 2, 242-245, 2011

APPENDIX 1

CODING

A1.1 PYTHON SCRIPT

```
#!/usr/bin/env python
import os
import logging
import wsgiref.handlers
import re
#import socket
#import sys
#from socket import *
from google.appengine.ext import webapp
from google.appengine.ext.webapp import template
from util.sessions import Session
from google.appengine.ext import db

class User(db.Model):
  name = db.StringProperty()
  ipaddress = db.StringProperty()
  emailaddress = db.StringProperty()
  mobilenumber = db.StringProperty()
  password = db.StringProperty()
  reenterpassword = db.StringProperty()
  """day = db.StringProperty()
  months = db.StringProperty()
  year = db.StringProperty()
  gender = db.StringProperty()
  security = db.StringProperty()
```

```
 answer = db.StringProperty() """

def doRender(handler, tname = 'index.htm', values = { }):
 temp = os.path.join(
     os.path.dirname(__file__),
     'templates/' + tname)
 if not os.path.isfile(temp):
   return False

 newval = dict(values)
 newval['path'] = handler.request.path
 handler.session = Session()
 if 'username' in handler.session:
    newval['username'] = handler.session['username']

 outstr = template.render(temp, newval)
 handler.response.out.write(outstr)
 return True

class LoginHandler(webapp.RequestHandler):

 def get(self):
   doRender(self, 'loginscreen.htm')

 def post(self):
   self.session = Session()
   name = self.request.get('name')
   emailaddress = self.request.get('emailaddress')
   password = self.request.get('password')
   logging.info('Checking emailaddress='+emailaddress+' password='+password)

   self.session.delete_item('username')
```

```
    self.session.delete_item('userkey')

    if name == '' or password == '' or emailaddress == '':
      doRender(
          self,
          'loginscreen.htm',
          {'error' : 'Please fill all the Fields'} )
      return

    que = db.Query(User)
    que = que.filter('name = ', name)
    que = que.filter('emailaddress =',emailaddress)
    que = que.filter('password = ',password)

    results = que.fetch(limit=1)

    if len(results) > 0 :
      user = results[0]
      self.session['userkey'] = user.key()
      self.session['username'] = name
      doRender(self,'profile.htm',{ } )

    else :
      doRender(
          self,
          'loginscreen.htm',
          {'error' : 'All Details Need to Match!! '} )
      return

class SignupHandler(webapp.RequestHandler):

 def get(self):
    doRender(self,'applyscreen.htm')
  def post(self):
    self.session = Session()
    name = self.request.get('name')
    ipaddress =  self.request.get('ipaddress')
    emailaddress = self.request.get('emailaddress')
    mobilenumber = self.request.get('mobilenumber')
    password = self.request.get('password')
    reenterpassword = self.request.get('reenterpassword')
    """day = self.request.get('day')
    months= self.request.get('months')
    year = self.request.get('year')
    gender = self.request.get('gender')
    security = self.request.get('security')
    answer = self.request.get('answer') """
    logging.info('Adding email address='+emailaddress)

    if name == '' or ipaddress =='' or emailaddress == '' or mobilenumber == '' or
password == '' or reenterpassword == '' :

      doRender(
              self,
              'applyscreen.htm',
              {'error' : 'Please fill in all fields'} )
      return

    if password != reenterpassword :
      doRender(
          self,
          'applyscreen.htm',
              {'error' : 'Check the Passwords'} )
      return
    if len(emailaddress) < 7 :
      doRender(
          self,
          'applyscreen.htm',
          {'error' : 'Incorrect email address format'} )
      return

    que = db.Query(User).filter('emailaddress =',emailaddress)
    results = que.fetch(limit=1)

    if len(results) > 0 :
      doRender(
          self,
              'applyscreen.htm',
              {'error' : 'Account Already Exists'} )
      return

    newuser = User(name=name, ipaddress=ipaddress, emailaddress=emailaddress,
mobilenumber=mobilenumber, password=password,
reenterpassword=reenterpassword );

    pkey = newuser.put();
    self.session['username'] = name
    self.session['userkey'] = pkey
    doRender(self,'profile.htm',{ })

class LogoutHandler(webapp.RequestHandler):

  def get(self):
    self.session = Session()
    self.session.delete_item('username')
    self.session.delete_item('userkey')
    doRender(self, 'index.htm')

class AboutHandler(webapp.RequestHandler):

  def get(self):
    if doRender(self,self.request.path) :
      return
    doRender(self,'about.htm')

class ProfileHandler(webapp.RequestHandler):
  def get(self):
    if doRender(self,self.request.path) :
      return
    doRender(self,'profile.htm')

  def post(self):
    """
    self.session = Session()
    HOST, PORT = "192.168.1.5", 9999
    data = " ".join(sys.argv[1:])

    sock =socket(socket.AF_INET, socket.SOCK_STREAM)


    try:
        sock.connect((HOST, PORT))
        sock.sendall(data + "\n")
        received = sock.recv(1024)
    finally:
        sock.close()

    print "Sent:     {}".format(data)
    print "Received: {}".format(received)  """

class MainHandler(webapp.RequestHandler):

  def get(self):
    if doRender(self,self.request.path) :
      return
    doRender(self,'index.htm')

def main():
  application = webapp.WSGIApplication([
     ('/login', LoginHandler),
     ('/apply', SignupHandler),
     ('/logout', LogoutHandler),
     ('/about', AboutHandler),
     ('/profile',ProfileHandler),
     ('/.*', MainHandler)],
     debug=True)
  wsgiref.handlers.CGIHandler().run(application)

if __name__ == '__main__':
  main()
```

A1.2 MICROCONTROLLER WITH ZIGBEE (1)

```
int incomingByte = 0; // for incoming serial data
int lamp1=7;
int lamp2=6;
int lamp3=5;
void setup() {
            Serial.begin(9600);     // opens serial port, sets data rate to 9600 bps
```

```
    pinMode(lamp1,OUTPUT);
    pinMode(lamp2,OUTPUT);
      pinMode(lamp3,OUTPUT);
}

void loop() {

      // send data only when you receive data:
      if (Serial.available() > 0) {
            // read the incoming byte:
            incomingByte = Serial.read();

            // say what you got:
            //Serial.print("I received: ");
            //Serial.println(incomingByte, DEC);

        if(incomingByte =='A'){digitalWrite(lamp1,HIGH);}
        if(incomingByte =='B'){digitalWrite(lamp1,LOW);}
        if(incomingByte =='C'){digitalWrite(lamp2,HIGH);}
        if(incomingByte =='D'){digitalWrite(lamp2,LOW);}
        if(incomingByte =='E'){digitalWrite(lamp3,HIGH);}
        if(incomingByte =='F'){digitalWrite(lamp3,LOW);}
      }
}
```

A1.3 MICROCONTROLLER WITH ZIGBEE (2)

```
int incomingByte = 0; // for incoming serial data
int lamp1=7;
int lamp2=6;
int lamp3=5;
void setup() {
        Serial.begin(9600);     // opens serial port, sets data rate to 9600 bps
     pinMode(lamp1,OUTPUT);
     pinMode(lamp2,OUTPUT);
     pinMode(lamp3,OUTPUT);
}

void loop() {

        // send data only when you receive data:
        if (Serial.available() > 0) {
                // read the incoming byte:
                incomingByte = Serial.read();

                // say what you got:
                //Serial.print("I received: ");
                //Serial.println(incomingByte, DEC);

          if(incomingByte =='G'){digitalWrite(lamp1,HIGH);}
          if(incomingByte =='H'){digitalWrite(lamp1,LOW);}
          if(incomingByte =='I'){digitalWrite(lamp2,HIGH);}
          if(incomingByte =='J'){digitalWrite(lamp2,LOW);}
          if(incomingByte =='K'){digitalWrite(lamp3,HIGH);}
          if(incomingByte =='L'){digitalWrite(lamp3,LOW);}
        }
}
```

A1.4 MICROCONTROLLER WITH ZIGBEE (3)

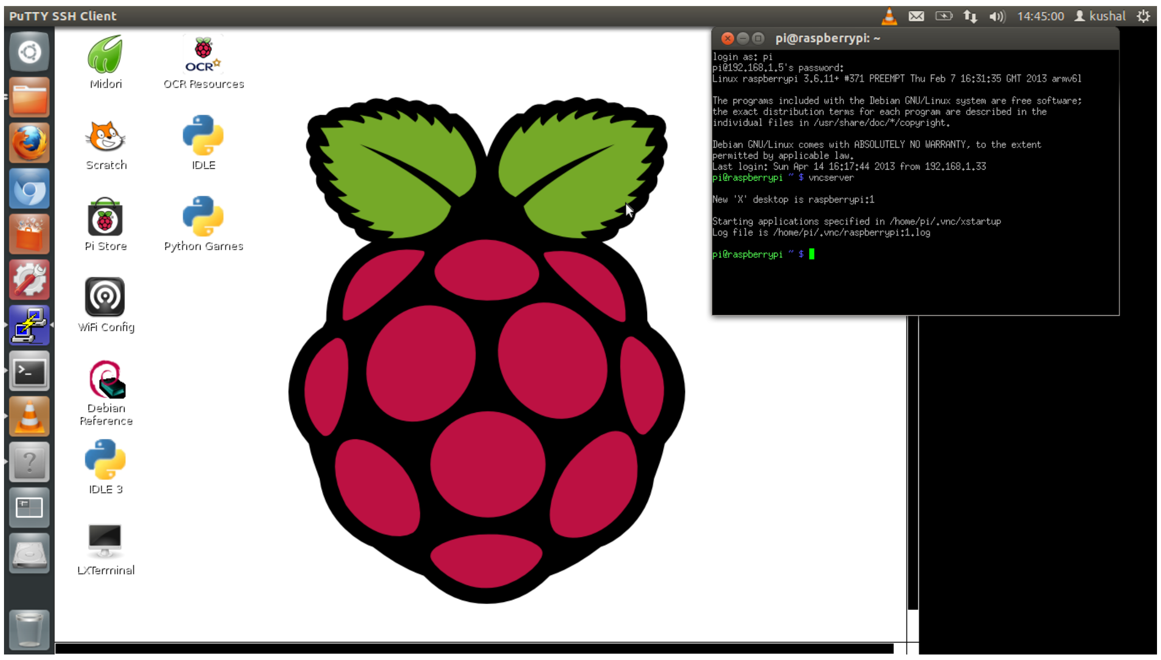
```
int incomingByte = 0; // for incoming serial data
int lamp1=4;
int lamp2=3;
int lamp3=2;
void setup() {
        Serial.begin(9600);    // opens serial port, sets data rate to 9600 bps
     pinMode(lamp1,OUTPUT);
     pinMode(lamp2,OUTPUT);
     pinMode(lamp3,OUTPUT);
}

void loop() {

        // send data only when you receive data:
        if (Serial.available() > 0) {
                // read the incoming byte:
                incomingByte = Serial.read();

                // say what you got:
                //Serial.print("I received: ");
                //Serial.println(incomingByte, DEC);

           if(incomingByte =='M'){digitalWrite(lamp1,HIGH);}
           if(incomingByte =='N'){digitalWrite(lamp1,LOW);}
           if(incomingByte =='O'){digitalWrite(lamp2,HIGH);}
           if(incomingByte =='P'){digitalWrite(lamp2,LOW);}
           if(incomingByte =='Q'){digitalWrite(lamp3,HIGH);}
           if(incomingByte =='R'){digitalWrite(lamp3,LOW);}
        }
}
```

A1.5 DESKTOP APPLICATION

```
#!/usr/bin/python

import wx
import serial

class MyDialog(wx.Dialog):
    def __init__(self, parent, id, title):
        wx.Dialog.__init__(self, None, wx.ID_ANY,"LazyBones")

            #pan=wx.Panel(self,-1)
            pic=wx.StaticBitmap(self, -1)
            pic.SetBitmap(wx.Bitmap("/home/kushal/Desktop/header.jpeg"))

            self.Show()

            wx.StaticText(self, -1, 'Operation:',(10, 110))
        self.rb1 = wx.RadioButton(self, -1, 'ON', (95, 130), style=wx.RB_GROUP)
        self.rb2 = wx.RadioButton(self, -1, 'OFF', (95, 150))

        wx.StaticText(self, -1, 'Switch: ', (10, 210))
        self.spct = wx.SpinCtrl(self, -1 ,'1', (95, 210) , (80, -1), min=1, max=9)
            button = wx.Button(self,1,'Send',(10,280),(75, -1))
            button.Bind(wx.EVT_BUTTON, self.onButton)

            wx.StaticText(self, -1, 'Testing :  Switches 3, 6 and 9 are Power Plugs \n
\n \t \t    Switches 2, 5 and 8 are 100 watt bulbs \n \n \t \t    Switches 1, 4 and 7 are
Zero watt Bulbs', (10, 330))
        self.Bind(wx.EVT_BUTTON, self.OnClose, id=1)
    self.Centre()
    self.ShowModal()
    self.Destroy()

def onButton(self,event):
        print "Button Pressed!!",self.rb1.GetValue(),self.spct.GetValue()
        ser = serial.Serial('/dev/ttyUSB0', 9600 ,timeout = 2)
        line = ser.readline()
        if (self.rb1.GetValue() == True) and (self.spct.GetValue() == 1):
                ser.write('A')
        elif (self.rb1.GetValue() == False) and (self.spct.GetValue() == 1):
          ser.write('B')
        elif (self.rb1.GetValue() == True) and (self.spct.GetValue() == 2):
          ser.write('C')
        elif (self.rb1.GetValue() == False) and (self.spct.GetValue() == 2):
          ser.write('D')
        elif (self.rb1.GetValue() == True) and (self.spct.GetValue() == 3):
          ser.write('E')
        elif (self.rb1.GetValue() == False) and (self.spct.GetValue() == 3):
          ser.write('F')
        elif (self.rb1.GetValue() == True) and (self.spct.GetValue() == 4):
          ser.write('G')
        elif (self.rb1.GetValue() == False) and (self.spct.GetValue() == 4):
          ser.write('H')
        elif (self.rb1.GetValue() == True) and (self.spct.GetValue() == 5):
          ser.write('I')
        elif (self.rb1.GetValue() == False) and (self.spct.GetValue() == 5):
          ser.write('J')
        elif (self.rb1.GetValue() == True) and (self.spct.GetValue() == 6):
          ser.write('K')
        elif (self.rb1.GetValue() == False) and (self.spct.GetValue() == 6):
                ser.write('L')
            elif (self.rb1.GetValue() == True) and (self.spct.GetValue() == 7):
                ser.write('M')
            elif (self.rb1.GetValue() == False) and (self.spct.GetValue() == 7):
                ser.write('N')
            elif (self.rb1.GetValue() == True) and (self.spct.GetValue() == 8):
                ser.write('O')
            elif (self.rb1.GetValue() == False) and (self.spct.GetValue() == 8):
                ser.write('P')
            elif (self.rb1.GetValue() == True) and (self.spct.GetValue() == 9):
                ser.write('Q')
            elif (self.rb1.GetValue() == False) and (self.spct.GetValue() == 9):
                ser.write('R')
            else :
                    print 'error'

    def OnClose(self, event):
        self.Close()
            ser.close()

app = wx.App(0)
MyDialog(None, -1, 'sample.py')
app.MainLoop()
```

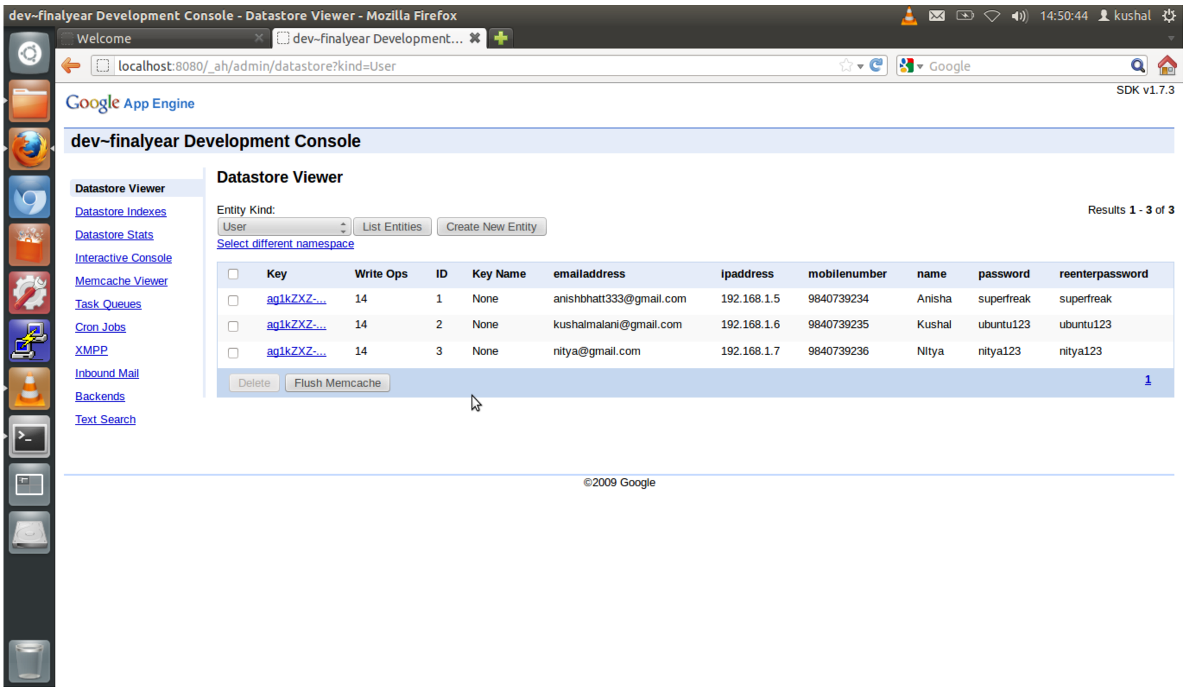
APPENDIX 2

SCREENSHOTS



```
login as: pi
pi@192.168.1.5's password:
Linux raspberrypi 3.6.11+ #371 PREEMPT Thu Feb 7 16:31:35 GMT 2013 armv6l

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sun Apr 14 16:17:44 2013 from 192.168.1.33
pi@raspberrypi ~ $ vncserver

New 'X' desktop is raspberrypi:1

Starting applications specified in /home/pi/.vnc/xstartup
Log file is /home/pi/.vnc/raspberrypi:1.log

pi@raspberrypi ~ $
```

Fig A2.1 Remote Desktop Connection



dev~finalyear Development Console

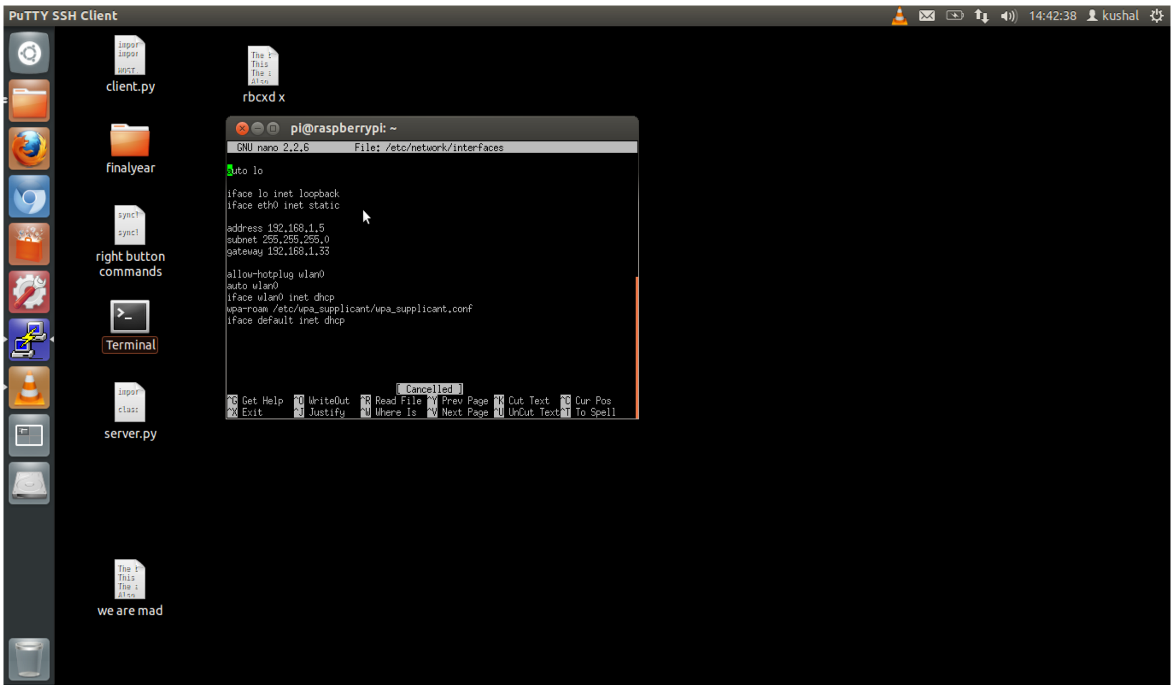
Datastore Viewer

Entity Kind: User

Results 1 - 3 of 3

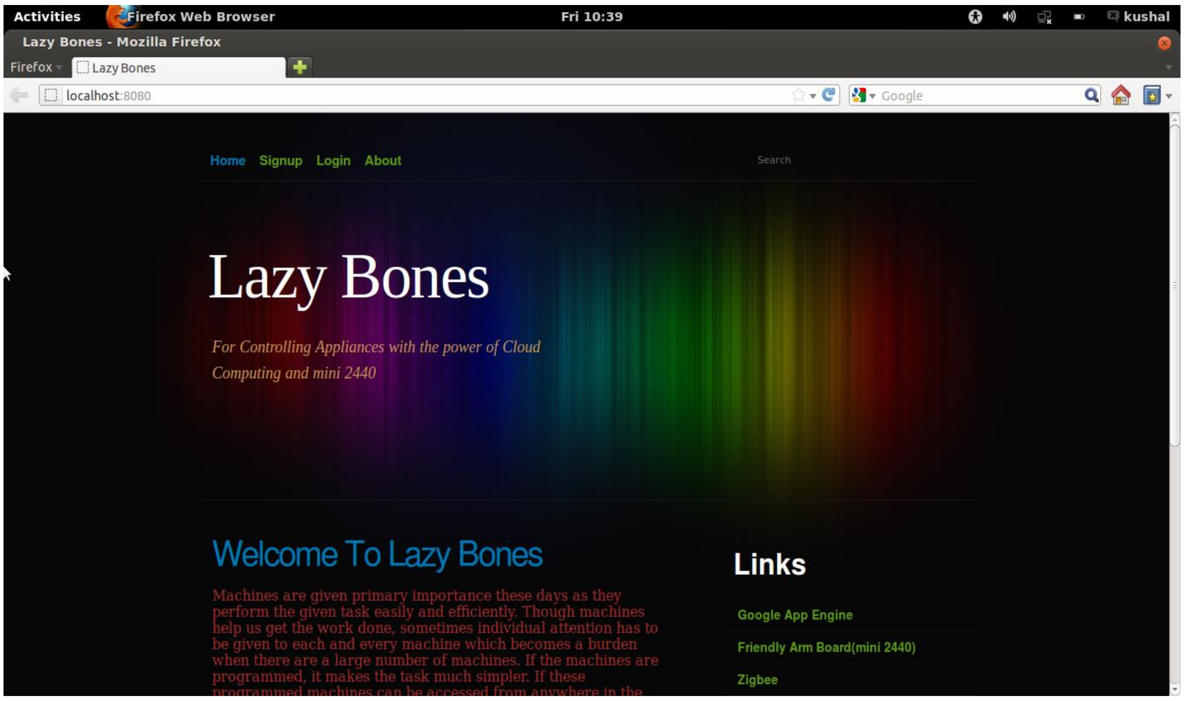
	Key	Write Ops	ID	Key Name	emailaddress	ipaddress	mobilenumber	name	password	reenterpassword
☐	ag1kZXZ-...	14	1	None	anishbhatt333@gmail.com	192.168.1.5	9840739234	Anisha	superfreak	superfreak
☐	ag1kZXZ-...	14	2	None	kushalmalani@gmail.com	192.168.1.6	9840739235	Kushal	ubuntu123	ubuntu123
☐	ag1kZXZ-...	14	3	None	nitya@gmail.com	192.168.1.7	9840739236	NItya	nitya123	nitya123

Fig A2.2 Google Data Store



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

auto lo

iface lo inet loopback
iface eth0 inet static

address 192.168.1.5
subnet 255.255.255.0
gateway 192.168.1.33

allow-hotplug wlan0
auto wlan0
iface wlan0 inet dhcp
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp



                           [ Cancelled ]
^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
```

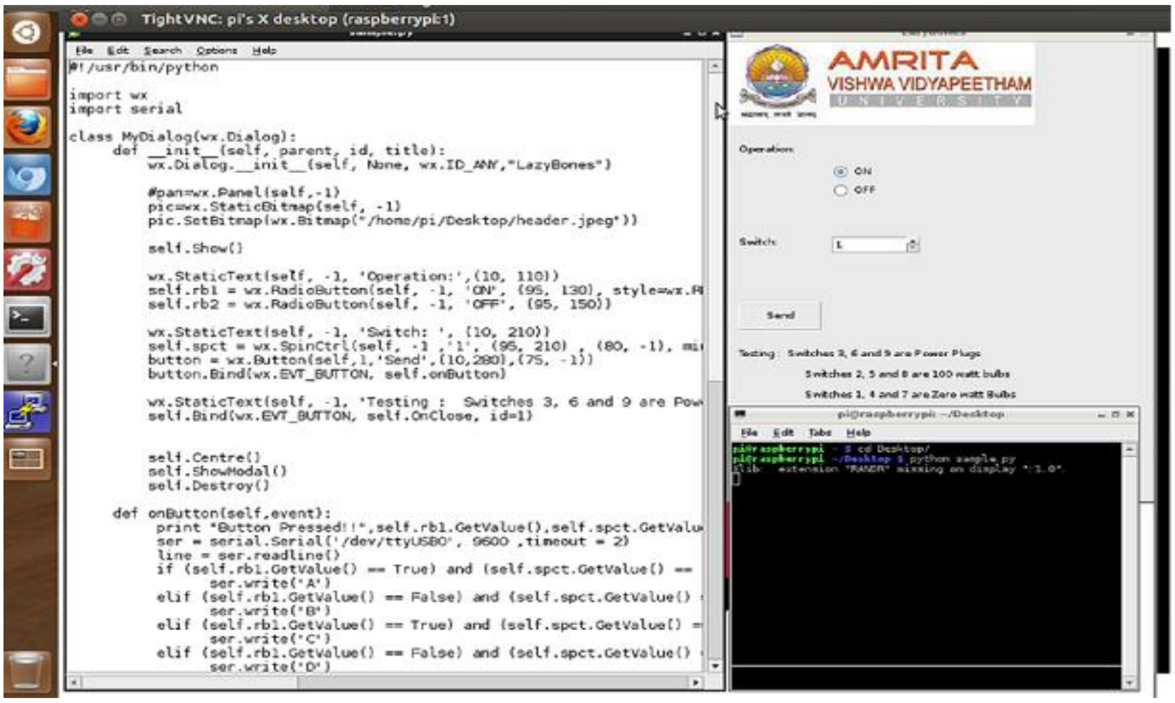
Fig A2.3 Setting Static IP



Welcome To Lazy Bones

Machines are given primary importance these days as they perform the given task easily and efficiently. Though machines help us get the work done, sometimes individual attention has to be given to each and every machine which becomes a burden when there are a large number of machines. If the machines are programmed, it makes the task much simpler. If these programmed machines can be accessed from anywhere in the

Fig A2.4 Website for Global Connectivity


TightVNC: pi's X desktop (raspberrypi:1)
File Edit Search Options Help
AMRITA
VISHWA VIDYAPEETHAM
UNIVERSITY
Send
File Edit Tabs Help

```
#!/usr/bin/python

import wx
import serial

class MyDialog(wx.Dialog):
      def __init__(self, parent, id, title):
            wx.Dialog.__init__(self, None, wx.ID_ANY,"LazyBones")

            #pan=wx.Panel(self,-1)
            pic=wx.StaticBitmap(self, -1)
            pic.SetBitmap(wx.Bitmap("/home/pi/Desktop/header.jpeg"))

            self.Show()

            wx.StaticText(self, -1, 'Operation:',(10, 110))
            self.rb1 = wx.RadioButton(self, -1, 'ON', (95, 130), style=wx.R
            self.rb2 = wx.RadioButton(self, -1, 'OFF', (95, 150))

            wx.StaticText(self, -1, 'Switch: ', (10, 210))
            self.spct = wx.SpinCtrl(self, -1 ,'1', (95, 210) , (80, -1), mi
            button = wx.Button(self,1,'Send',(10,280),(75, -1))
            button.Bind(wx.EVT_BUTTON, self.onButton)

            wx.StaticText(self, -1, 'Testing :  Switches 3, 6 and 9 are Pow
            self.Bind(wx.EVT_BUTTON, self.OnClose, id=1)


            self.Centre()
            self.ShowModal()
            self.Destroy()

      def onButton(self,event):
             print "Button Pressed!!",self.rb1.GetValue(),self.spct.GetValu
             ser = serial.Serial('/dev/ttyUSB0', 9600 ,timeout = 2)
             line = ser.readline()
             if (self.rb1.GetValue() == True) and (self.spct.GetValue() ==
                    ser.write('A')
             elif (self.rb1.GetValue() == False) and (self.spct.GetValue()
                    ser.write('B')
             elif (self.rb1.GetValue() == True) and (self.spct.GetValue() =
                    ser.write('C')
             elif (self.rb1.GetValue() == False) and (self.spct.GetValue()
                    ser.write('D')
```

Operation: [✓ ON] OFF

Switch: 1

Testing : Switches 3, 6 and 9 are Power Plugs
Switches 2, 5 and 8 are 100 watt bulbs
Switches 1, 4 and 7 are Zero watt Bulbs

```
pi@raspberrypi ~ $ cd Desktop/
pi@raspberrypi ~/Desktop $ python sample.py
Xlib:  extension "RANDR" missing on display ":1.0".
```

Fig A2.5 Local connectivity

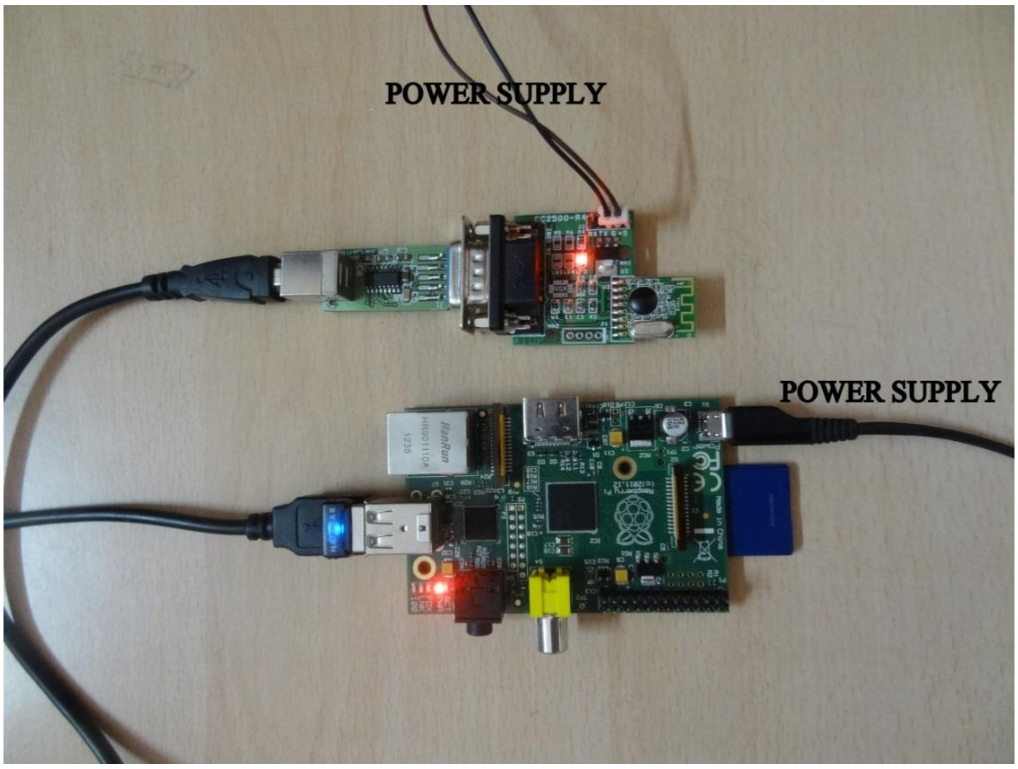


Fig A2.6 Raspberry Pi connected to Zigbee

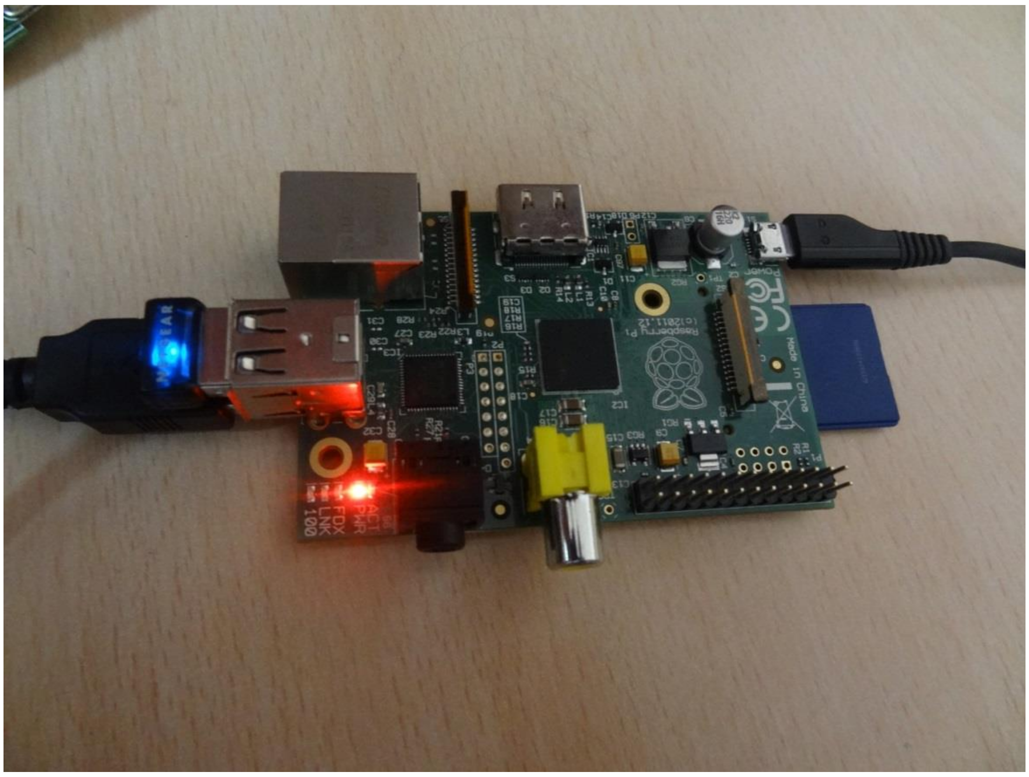


Fig A2.7 Raspberry Pi with a Netgear Wi-Fi Dongle

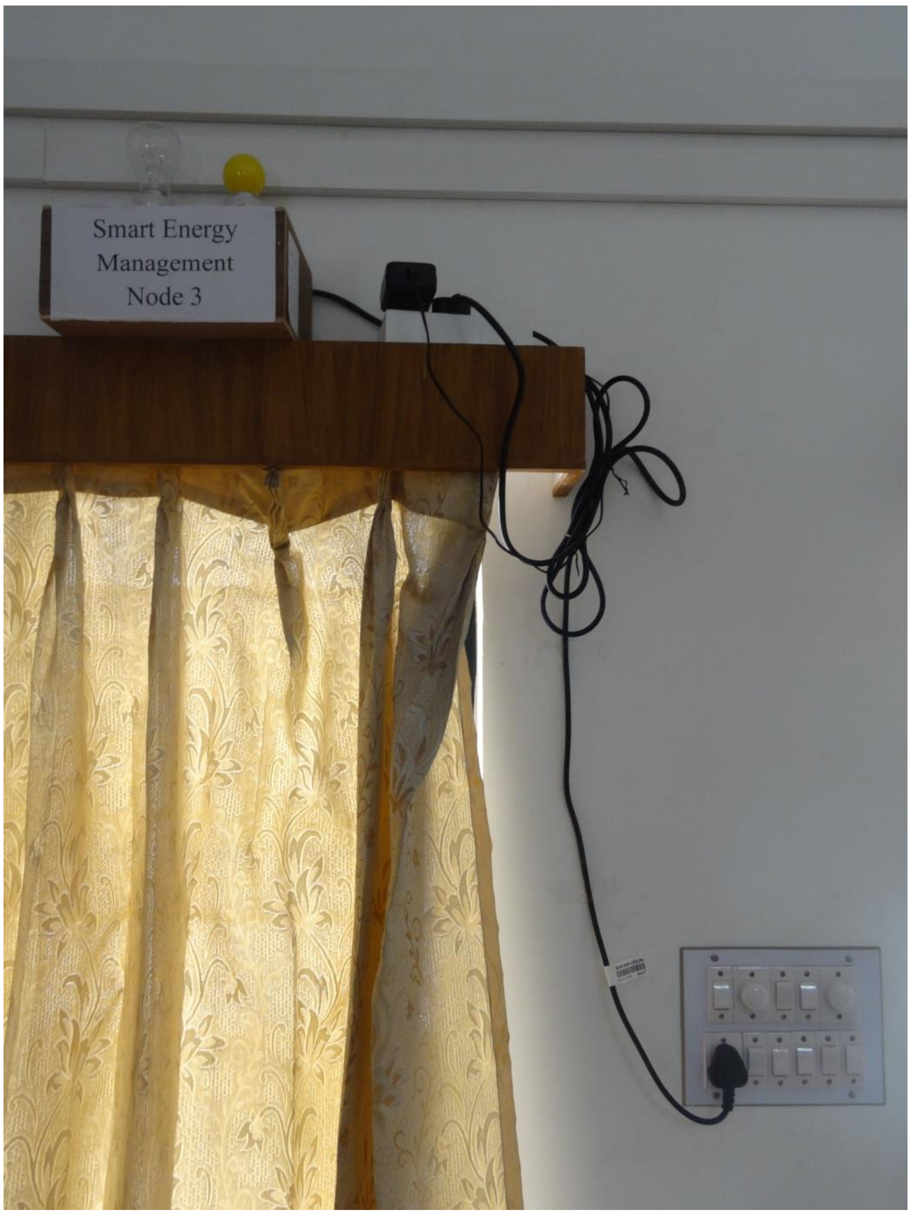


Fig A2.8 Switch board connected to a microcontroller and ZigBee

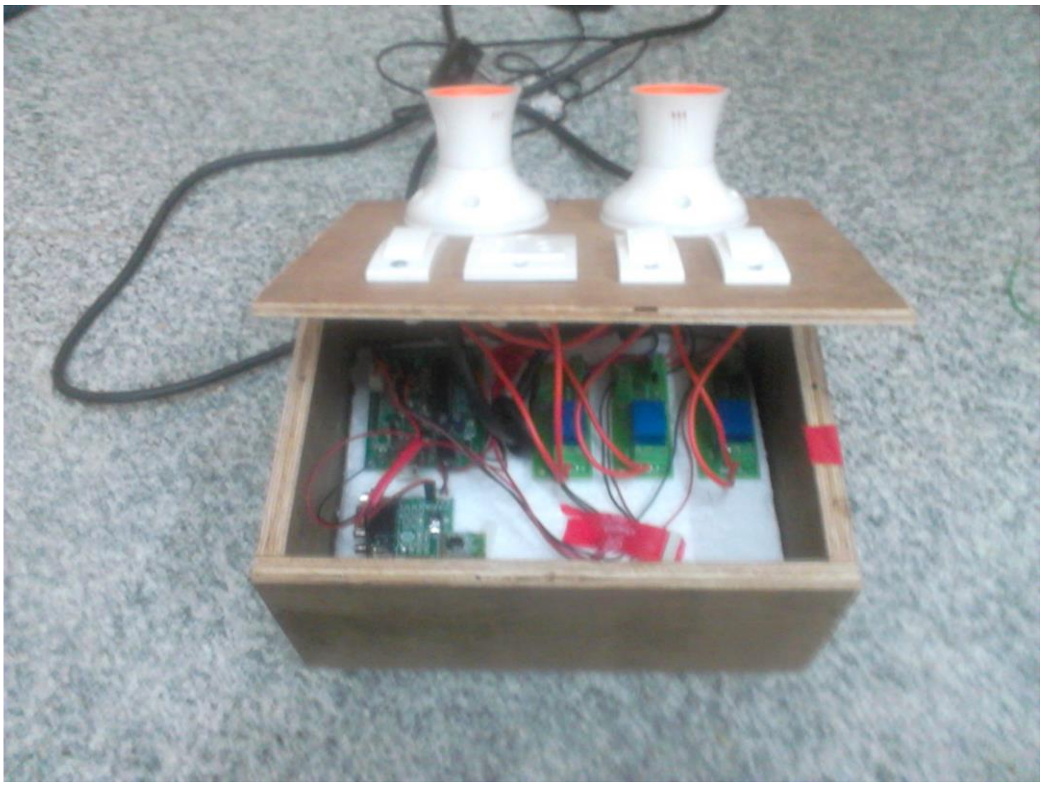


Fig A2.9 Switch board with Relay Circuit