

CHAPTER-1

INTRODUCTION

1.1. Home Automation

Home automation is the residential extension of building automation. It is automation of the home, housework or household activity. Home automation may include centralized control of lighting, HVAC (heating, ventilation and air conditioning), appliances, security locks of gates and doors and other systems, to provide improved convenience, comfort, energy efficiency and security. Home automation for the elderly and disabled can provide increased quality of life for persons who might otherwise require caregivers or institutional care.

A home automation system integrates electrical devices in a house with each other. The techniques employed in home automation include those in building automation as well as the control of domestic activities, such as home entertainment systems, houseplant and yard watering, pet feeding, changing the ambiance "scenes" for different events (such as dinners or parties), and the use of domestic robots. Devices may be connected through a home network to allow control by a personal computer, and may allow remote access from the internet. Through the integration of information technologies with the home environment, systems and appliances are able to communicate in an integrated manner which results in convenience, energy efficiency, and safety benefits.

Home automation has been a feature of science fiction writing for many years, but has only become practical since the early 20th Century following the widespread introduction of electricity into the home, and the rapid advancement of information technology. Early remote control devices began to emerge in the late 1800s. For example, Nikola Tesla patented an idea for the remote control of vessels and vehicles in 1898. The emergence of electrical home appliances began between 1915 and 1920; the decline in domestic servants meant that households needed cheap, mechanical replacements. Domestic electricity supply, however, was still in its infancy — meaning this luxury was afforded only the more affluent households.

Ideas similar to modern home automation systems originated during the World's Fairs of the 1930s. Fairs in Chicago (1934), New York (1939) and (1964–65), depicted

electrified and automated homes. In 1966 Jim Sutherland, an engineer working for Westinghouse Electric, developed a home automation system called "ECHO IV"; this was a private project and never commercialized. The first "wired homes" were built by American hobbyists during the 1960s, but were limited by the technology of the times. The term "smart house" was first coined by the American Association of House builders in 1984. With the invention of the microcontroller, the cost of electronic control fell rapidly. Remote and intelligent control technologies were adopted by the building services industry and appliance manufacturers.

By the end of the 1990s, "domotics" was commonly used to describe any system in which informatics and telematics were combined to support activities in the home. The phrase is a neologism formed from domus (Latin, meaning house) and informatics, and refers to the application of computer and robot technologies to domestic appliances. The concept "Domotique" was initially introduced in France in the 1980s and was during the 1990's introduced in Spain and Italy as "Domótica", and refers to home automation.

Despite interest in home automation, by the end of the 1990s there was not a widespread uptake, with such systems still considered the domain of hobbyists or the rich. The lack of a single, simplified, protocol and high cost of entry has put off consumers. While there is still much room for growth, according to ABI Research, 1.5 million home automation systems were installed in the US in 2012, and a sharp uptake could see shipments topping over 8 million in 2017. Home automation has greatly increased in popularity over the past several years. One of the greatest advantages of an automated home is the ease with which functionality can be managed on an array of devices: desktop, laptop, tablet or smartphone. Before determining which home automation package is right for you and your family, it is important to become better informed of the features and settings associated with home safety and security systems.

1.1.1. Importance and benefits

When you're not home, nagging little doubts can start to crowd your mind. Did I turn the coffee maker off? Did I set the security alarm? Are the kids doing their homework or watching television? With a smart home, you could quiet all of these worries with a quick glance at your smartphone or tablet. You could connect the devices

and appliances in your home so they can communicate with each other and with you. Any device in your home that uses electricity can be put on your home network and at your command. Whether you give that command by voice, remote control, tablet or smartphone, the home reacts.

Most applications relate to lighting, home security, home theater and entertainment, and thermostat regulation. The idea of a smart home might make you think of George Jetson and his futuristic abode or maybe Bill Gates, who spent more than \$100 million building his smart home. Once a draw for the tech-savvy or the wealthy, smart homes and home automation are becoming more common.

Instead of start-up companies, more established tech organizations are launching new smart home products. Sales of automation systems could grow to around \$9.5 billion by 2015. By 2017, that number could balloon to \$44 billion. Much of this is due to the jaw-dropping success of smartphones and tablet computers. These ultra-portable computers are everywhere, and their constant Internet connections means they can be configured to control myriad other online devices. It's all about the Internet of Things.

Home automation is "The Internet of Things". The way that all of our devices and appliances will be networked together to provide us with a seamless control over all aspects of our home and more. Home automation has been around from many decades in terms of lighting and simple appliance control, and only recently has technology caught up for the idea of the interconnected world, allowing full control of your home from anywhere, to become a reality.

With home automation, you dictate how a device should react, when it should react, and why it should react. Home automation can also alert you to events that you might want to know about right-away while you are gone like water leaks and unexpected access to your home, or any part of it. At any time, you can grab your iPhone, Android device or other remote control and change the settings in your house as desired.

Home automation systems are composed of hardware, communication and electronic interfaces that work to integrate electrical devices with one another. Domestic activities can then be regulated with the touch of a button. From any remote

location, users can adjust the controls on home entertainment systems, limit the amount of sunlight given to houseplants, or change the temperatures in certain rooms. Home automation software is often connected through computer networks so that users can adjust settings on their personal devices.

The three main elements of a home security system are sensors, controllers and actuators. Sensors can monitor changes in daylight, temperature or motion detection; home automation systems can then adjust settings to the preferred levels of a user. Controllers refer to the devices—personal computers, tablets or smartphones—used to send and receive messages about the status of automated features in users' homes. Actuators may be light switches, motors or motorized valves that control a mechanism or function of a home automation system.

The household activities are automated by the development of special appliances such as water heaters to reduce the time taken to boil water for bathing and automatic washing machines to reduce manual labour of washing clothes. In developed countries, homes are wired for electrical power, doorbell, TV outlets, and telephones. The different application includes when a person enters the room, the light turns on. In advanced technology, the room can sense the presence of the person and who the person is. In the case of a smoke detector when fire or smoke is detected, the lights in the entire house begin to blink to alert the resident to the probable fire. In case of a home theatre, the home automation system can avoid distraction and lock the audio and video components and can also make an announcement. The home automation system can also dial up the house owner on their mobile phone to alert them or call any alarm monitoring company.

Home automation refers to the use of computer and information technology to control home appliances and features (such as windows or lighting). Systems can range from simple remote control of lighting through to complex computer/micro-controller based networks with varying degrees of intelligence and automation. Home automation is adopted for reasons of ease, security and energy efficiency. In modern construction in industrialized nations, most homes have been wired for electrical power, telephones, TV outlets (cable or antenna), and a doorbell. Many household tasks were automated by the development of specialized automated appliances. For instance, automatic washing machines were developed to reduce the manual labor of cleaning clothes, and

water heaters reduced the labor necessary for bathing. The use of gaseous or liquid fuels, and later the use of electricity enabled increased automation in heating, reducing the labor necessary to manually refuel heaters and stoves. Development of thermostats allowed more automated control of heating, and later cooling.

One of the greatest advantages of home automation systems is that users can protect against break-ins and fires, while enjoying automations for lights, temperature, and more. The automation of features in one's home helps to promote security, comfort, energy efficiency, and convenience. Another benefit of home automation systems is the amount of labor, time, energy and materials that is saved. Home automation systems are becoming more and more affordable. Not only are prices decreasing, but operating systems are also become less complex so that users can readily master all the controls associated with their safety and security devices. Home automation commands can now be given through smartphones, tablets, and televisions, in addition to computers.

As the number of controllable devices in the home rises, interconnection and communication becomes a useful and desirable feature. For example, a furnace can send an alert message when it needs cleaning or a refrigerator when it needs service. If no one is supposed to be home and the alarm system is set, the home automation system could call the owner, or the neighbors, or an emergency number if an intruder is detected. In simple installations, automation may be as straightforward as turning on the lights when a person enters the room.

In advanced installations, rooms can sense not only the presence of a person inside but know who that person is and perhaps set appropriate lighting, temperature, music levels or television channels, taking into account the day of the week, the time of day, and other factors. Other automated tasks may include reduced setting of the heating or air conditioning when the house is unoccupied, and restoring the normal setting when an occupant is about to return.

The real hands-on control comes in when you start interacting with the home automation system from your remote app. In addition to arming and disarming your security system, you can reprogram the scheduling, lock and unlock doors, reset the thermostat and adjust the lights all from your phone, from anywhere in the world. As manufacturers are creating more and more "smart" devices and appliances all the time,

the possibilities for home automation are virtually limitless. Until fairly recently, automated central control of building-wide systems was found only in larger commercial buildings and expensive homes. Typically involving only lighting, heating and cooling systems, building automation rarely provided more than basic control, monitoring and scheduling functions and was accessible only from specific control points within the building itself.

Home automation is a step toward what is referred to as the "Internet of Things," in which everything has an assigned IP address, and can be monitored and accessed remotely. The first and most obvious beneficiaries of this approach are "smart" devices and appliances that can be connected to a local area network, via Ethernet or Wi-Fi. Although the day is still far off when you'll be able to use your mobile browser to track down a lost sock, home networks are capable of including an increasing number of devices and systems.

Automation is, unsurprisingly, one of the two main characteristics of home automation. Automation refers to the ability to program and schedule events for the devices on the network. The programming may include time-related commands, such as having your lights turn on or off at specific times each day. It can also include non-scheduled events, such as turning on all the lights in your home when your security system alarm is triggered.

Once you start to understand the possibilities of home automation scheduling, you can come up with any number of useful and creative solutions to make your life better. Plug your motorized blinds into a "smart" outlet and program it to close at noon each day. Program your home automation system to unlock the front door for them, and lock it up again when they're done. The other main characteristic of cutting-edge home automation is remote monitoring and access. While a limited amount of one-way remote monitoring has been possible for some time, it's only since the rise in smartphones and tablets that we've had the ability to truly connect to our home networks while we're away. With the right home automation system, you can use any Internet-connected device to view and control the system itself and any attached devices.

Monitoring apps can provide a wealth of information about your home, from the status of the current moment to a detailed history of what has happened up to now.

You can check your security system's status, whether the lights are on, whether the doors are locked, what the current temperature of your home is and much more. With cameras as part of your home automation system, you can even pull up real-time video feeds and literally see what's going on in your home while you're away. Even simple notifications can be used to perform many important tasks.

You can program your system to send you a text message or email whenever your security system registers a potential problem, from severe weather alerts to motion detector warnings to fire alarms. You can also get notified for more mundane events, such as programming your "smart" front door lock to let you know when your child returns home from school.

Home automation can also provide a remote interface to home appliances or the automation system itself, to provide control and monitoring on a smartphone or web browser. An example of remote monitoring in home automation could be triggered when a smoke detector detects a fire or smoke condition, causing all lights in the house to blink to alert any occupants of the house to the possible emergency. If the house is equipped with a home theater, a home automation system can shut down all audio and video components to avoid distractions, or make an audible announcement. The system could also call the home owner on their mobile phone to alert them, or call the fire department or alarm monitoring company.

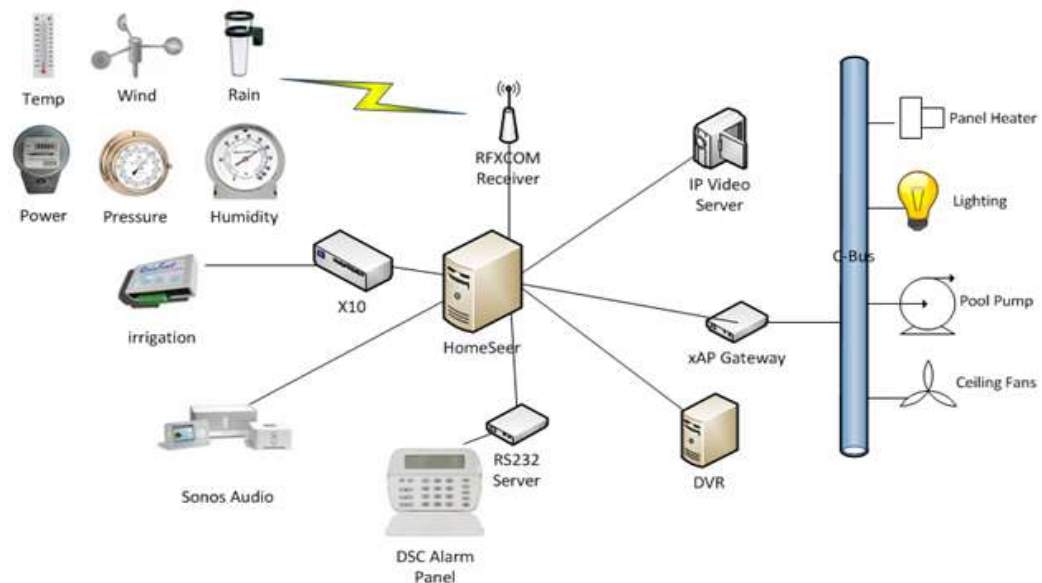


Fig. 1.1: Connection diagram

1.1.2. System Elements

Today's home automation systems are more likely to distribute programming and monitoring control between a dedicated device in the home, like the control panel of a security system, and a user-friendly app interface that can be accessed via an Internet-enabled PC, smartphone or tablet. Manufacturers have produced a wide variety of "smart" devices, many of which are full of innovative features but few of which offer the kind of integration needed to be part of a complete home automation system. Much of the problem has been that each manufacturer has a different idea of how these devices should be connected and controlled. So while you may have a "smart" TV, washing machine, refrigerator, thermostat, coffee maker or any of the other Internet-ready household devices on the market, the end result is usually a separate control scheme for each device.

In the near future, home automation may be standardized to let us truly take advantage of all of these additional possibilities. For the time being, the home security providers that specialize in home automation have focused on the most critical and useful parts of a connected home. At a basic level, this means the doors and windows and environmental devices (thermostat, smoke detectors, temperature, humidity, fire and carbon dioxide sensors) that keep you safe and comfortable. For additional real-time security, convenience and control, home automation systems from security providers should also include options for video cameras. With the best systems, you'll also be able to include lights and individual electrical outlets into your home automation package.

One clear advantage of home automation is the unmatched potential for energy savings, and therefore cost savings. Your thermostat is already "smart" in the sense that it uses a temperature threshold to govern the home's heating and cooling system. In most cases, thermostats can also be programmed with different target temperatures in order to keep energy usage at a minimum during the hours when you're least likely to benefit from the heating and cooling.

At the most basic level, home automation extends that scheduled programmability to lighting, so that you can suit your energy usage to your usual daily schedule. With more flexible home automation systems, electrical outlets or even

individual devices can also be automatically powered down during hours of the day when they're not needed. As with isolated devices like thermostats and sprinkler systems, the scheduling can be further broken down to distinguish between weekends and even seasons of the year, in some cases.

Set schedules are helpful, but many of us keep different hours from day to day. Energy costs can be even further reduced by programming "macros" into the system and controlling it remotely whenever needed. In other words, you could set up a "coming home" event that turns on lights and heating as you're driving home after work, for example, and activate it all with one tap on your smartphone. An opposite "leaving home" event could save you from wasting energy on forgotten lights and appliances once you've left for the day.

Elements of a home automation system include; sensors (such as temperature, daylight, or motion detection); controllers (such as a general-purpose personal computer or a dedicated automation controller); actuators, (such as motorized valves, light switches and motors); buses (wired or wireless); and interfaces (human-machine and / or machine-to-machine). One or more human-machine and/or machine-to-machine, interface devices are required, so that the residents of the home can interact with the system for monitoring and control; this may be a specialized terminal or, increasingly, may be an application running on a smart phone or tablet computer.

Devices may communicate over dedicated wiring, or over a wired network, or wirelessly using one or more protocols. Building automation networks developed for institutional or commercial buildings may be adapted to control in individual residences. A centralized controller can be used, or multiple intelligent devices can be distributed around the home.

1.1.3. Tasks

1.1.3.1. HVAC

Heating, ventilation and air conditioning (HVAC) systems can include temperature and humidity control, including fresh air heating and natural cooling. An Internet-controlled thermostat allows the homeowner to control the building's heating and air conditioning systems remotely. The system may automatically open and close windows to cool the house.

1.1.3.2. Lighting

Lighting control systems can be used to control household electric lights. Lights can be controlled on a time cycle, or arranged to automatically go out when a room is unoccupied. Electronically controlled lamps can be controlled for brightness or color to provide different light levels for different tasks. Lighting can be controlled remotely by a wireless control or over the Internet. Natural lighting can be used to automatically control window shades and draperies to make best use of natural light.

1.1.3.3. Audio-visual

This category includes audio and video switching and distribution. Multiple audio or video sources can be selected and distributed to one or more rooms and can be linked with lighting and blinds to provide mood settings.

1.1.3.4. Shading

Automatic control of blinds and curtains can be used for:

- Presence simulation
- Privacy
- Temperature control
- Brightness control
- Glare control
- Security (in case of shutters)

1.1.3.5. Security

A household security systems integrated with a home automation system can provide additional services such as remote surveillance of security cameras over the Internet or central locking of all perimeter doors and windows. With home automation, the user can select and watch cameras live from an Internet source to their home or business. Security systems can include motion sensors that will detect any kind of unauthorized movement and notify the user through the security system or via cell phone. The automation system can simulate the appearance of an occupied home by automatically adjusting lighting or window coverings. Detection systems such as fire alarm, gas leak, carbon monoxide, or water leaks can be integrated. Personal medical alarm systems allow an injured home occupant to summon help.

1.1.3.6. Intercoms

An intercom system allows communication via a microphone and loud speaker between multiple rooms. Integration of the intercom to the telephone , or of the video door entry system to the television set, allowing the residents to view the door camera automatically.

1.1.3.7. Domotics

Journalist Bruno de Latour coined the term domotic in 1984. Domotic has been recently introduced in vocabulary as a composite word of Latin word domus and informatics and it refers to intelligent houses meaning the use of the automation technologies and computer science applied to the home. The term covers a range of applications of information technology to the problems of home automation. Domotics is the study of the realization of an intelligent home environment. Digital Home includes home automation, multimedia, telecommunications, e-commerce, etc. through home network .Domotics and home automation means that systems talk to each other for improved convenience, efficiency and safety.

1.1.3.8. Other systems

Using special hardware, almost any household appliance can be monitored and controlled automatically or remotely, including cooking appliances, swimming pool systems, and others.

1.2. Aim of the Project

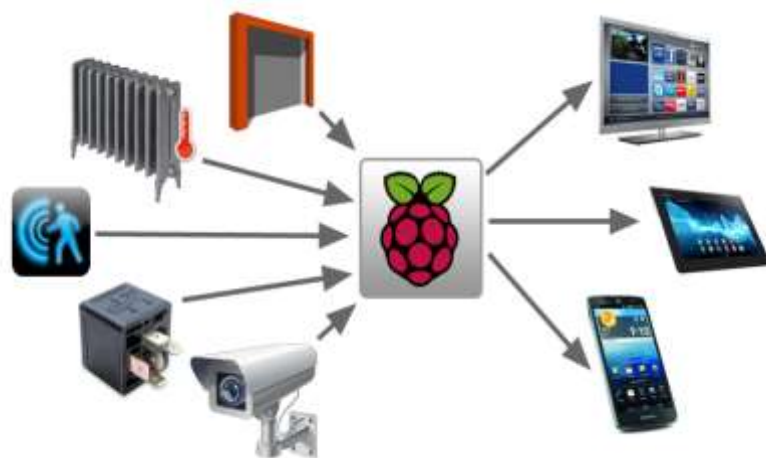


Fig.1.2: Raspberry Pi interface with devices

The goal of this project is to build a complete home automation system based on open-source hardware. By means of this project, we intend to put together a home automation system which has both voice recognition and email based functionality. In both home security systems and home automation systems, you've got a lot of the same things: sensors, servers, monitoring, alerts, etc. This project primarily deals with automating electronics at a domestic level in an easier and user-friendly way.



Fig.1.3: Raspberry Pi model B+ kit



Fig.1.4: Control of Home appliances with phone using RPi

CHAPTER-2

PROJECT DEFINITION AND REQUIREMENT

2.1. Raspberry Pi

Fortunately the technology related to home automation has progressed enormously in the last few years and new pieces of hardware have been developed which provide easy solutions to the above problems. The first important piece of the puzzle has emerged at the beginning of the year 2012, when the Raspberry Pi computer was officially released.

The Raspberry Pi is a small, noiseless computer, which has a very low power consumption (typically 1-2 Watts) and is very lightweight. It takes up little space in the room and most importantly it is cheap. In other words it is the perfect choice for a computer to control the lights in your home. The Raspberry Pi is a small, barebones computer developed by The Raspberry Pi Foundation, a UK charity, with the intention of providing low-cost computers and free software to students. Their ultimate goal is to foster computer science education and they hope that this small, affordable computer will be a tool that enables that.

. The Raspberry Pi is a series of credit card-sized single-board computers developed in the UK by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools. The original Raspberry Pi and Raspberry Pi 2 are manufactured in several board configurations through licensed manufacturing agreements with Newark element14 (Premier Farnell), RS Components and Egoman. These companies sell the Raspberry Pi online. Egoman produces a version for distribution solely in China and Taiwan, which can be distinguished from other Pis by their red coloring and lack of FCC/CE marks. The hardware is the same across all manufacturers.

The original Raspberry Pi is based on the Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S 700 MHz processor, Video Core IV GPU, and was originally shipped with 256 megabytes of RAM, later upgraded (models B and B+) to 512 MB. The system has Secure Digital (SD) (models A and B) or Micro SD (models A+ and B+) sockets for boot media and persistent storage. In 2014, the Raspberry Pi Foundation launched the Compute Module, which packages a BCM2835

with 512 MB RAM and an eMMC flash chip into a module for use as a part of embedded systems. The Foundation provides Debian and Arch Linux ARM distributions for download. Tools are available for Python as the main programming language, with support for BBC BASIC (via the RISC OS image or the Brandy Basic clone for Linux), C, C++, Java, Perl and Ruby.

The allure of the Raspberry Pi comes from a combination of the computer's small size and affordable price. Enthusiasts envision using the small form-factor PC as a cheap home theater PC (HTPC), or secondary low-power desktop. Institutions, like schools and businesses, could benefit from deploying a fleet of computers for a fraction of the cost of traditional desktop towers. The small size makes for an easy-to-hide computer that sips power and can be mounted behind the display with an appropriate case. It could also be used in niche applications, like digital signage. While it will not blow away any recent hardware in performance, it does make for a cheap secondary computer which could be useful for troubleshooting and researching solutions if your man rig fails to boot as well.

The Raspberry Pi is not the only small device of its kind — two prominent examples in the enthusiast community are the Arduino and BeagleBoard. Although the systems are of similar form-factors, the Raspberry Pi is a greatly different beast. On the hardware front, the Raspberry Pi is based around an ARM SoC that is very much closed source. Conversely, the Arduino and BeagleBoard systems are based on fully open source hardware. The BeagleBoards do use ARM processors (TI OMAP 3530 SoC), but different GPUs. The Arduino boards are even further dissimilar due to using 8-bit and 16-bit Atmel micro-controller chips.

The biggest difference between something like the Arduino and the Raspberry Pi is in the intended usage. The Arduino is meant to be used as a development board with micro-controllers that will be programmed and then integrated into larger machines or electronics and allowed to run on their own. On the other hand, the Raspberry Pi is meant to be used as a final product and operate as a traditional desktop computer (in fact, the distributors refused to ship the Raspberry Pi until it received CE/FCC EM interference certification). Admittedly, the BeagleBoards do dip into the Raspberry Pi's usage territory with projects like BeagleBoard Ubuntu and XBMC support, but the devices do have a higher barrier to entry due to its price.

When keeping expectations in check — especially where performance is concerned — the Raspberry Pi is a capable little computer for the price. Although the merits and future of the device as a low cost educational tool for UK children remains to be seen, the hacking, modding, and enthusiasts computing crowds are already hard at work designing and planning projects of their own around the tiny device.

The Raspberry Pi is a computer, very like the computers with which you're already familiar. It uses a different kind of processor, so you can't install Microsoft Windows on it. But you can install several versions of the Linux operating system that look and feel very much like Windows. If you want to, you can use the Raspberry Pi to surf the internet, send an email or write a letter using a word processor. But you can also do so much more. Easy to use but powerful, affordable and (as long as you're careful) difficult to break, the Raspberry Pi is the perfect tool for aspiring computer scientists.

The Raspberry Pi is one of computing's modern marvels—a credit-card-sized single board computer, capable of running Linux, its applications, and capable of handling playback of HD video. Launched on February 29, 2012, it has now been released in three different versions, with the current two versions being the Model A (256 Mbyte with USB) and the Model B (512 Mbyte, with 2 x USB and Ethernet).

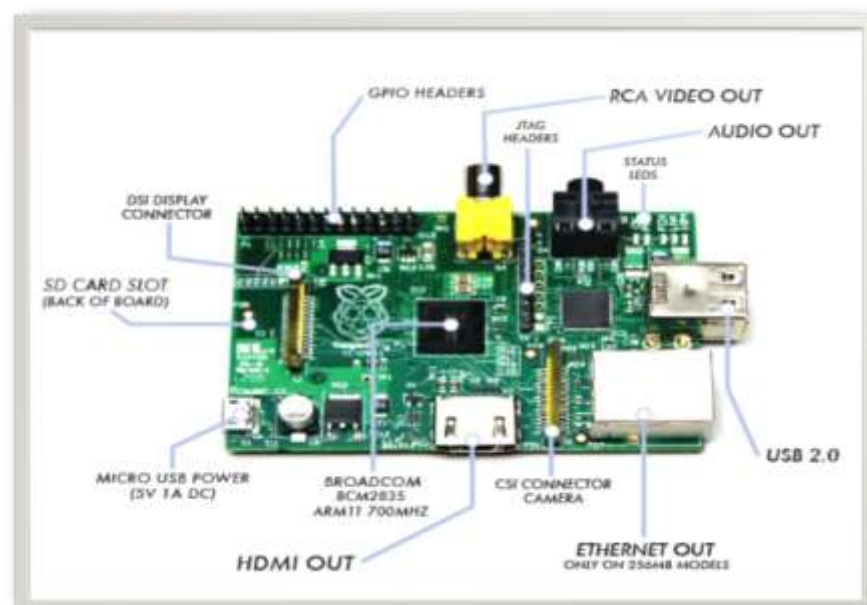


Fig.2.1: The Raspberry Pi Kit

One of the great things about the Raspberry Pi is that there's no single way to use it. Whether you just want to watch videos and surf the web, or you want to hack, learn, and make with the board, the Raspberry Pi is a flexible platform for fun, utility, and experimentation.

2.2. Different ways of using Raspberry Pi

The different ways of using a Raspberry Pi are given as follows:

2.2.1. General Purpose Computing

The Raspberry Pi is a computer and it can in fact be used as one. After you get it up and running you can choose to have it boot into a graphical desktop environment with a web browser, which is a lot of what we use computers for these days. Going beyond the web, you can install a wide variety of free software, such as the Libre Office productivity suite for working with documents and spreadsheets when you don't have an Internet connection.

2.2.2. Learning to program

Since the Raspberry Pi is meant as an educational tool to encourage kids to experiment with computers, it comes preloaded with interpreters and compilers for many different programming languages. For the beginner, there's Scratch, a graphical programming language from MIT. If you're eager to jump into writing code, the Python programming language is a great way to get started. You can write programs for your Raspberry Pi in many different programming languages like C, Ruby, Java, and Perl.

2.2.3. Project Platform

The Raspberry Pi differentiates itself from a regular computer not only in its price and size, but also because of its ability to integrate with electronics projects.

2.2.4. Media Center

Since the Raspberry Pi has both HDMI and composite video outputs, it's easy to connect to televisions. It also has enough processing power to play full screen video in high definition. To leverage these capabilities, contributors to the free and open source media player, XBMC, have ported their project to the Raspberry Pi. XBMC can play many different media formats and its interface is designed with large buttons and

text so that it can be easily controlled from the couch. XBMC makes the Raspberry Pi a fully customizable home entertainment center component.

2.2.5 “Bare Metal” Computer Hacking

Most people who write computer programs write code that runs within an operating system, such as Windows, Mac OS, or—in the case of Raspberry Pi—Linux. But what if you could write code that runs directly on the processor without the need for an operating system? You could even write your own operating system from scratch if you were so inclined.

2.3. Blocks of Raspberry Pi

All the parts of the Raspberry Pi can be described as below:

2.3.1. The processor

At the heart of the Raspberry Pi is the same processor you would have found in the iPhone 3G and the Kindle 2, so you can think of the capabilities of the Raspberry Pi as comparable to those powerful little devices. This chip is a 32 bit, 700 MHz System on a Chip, which is built on the ARM11 architecture. ARM chips come in a variety of architectures with different cores configured to provide different capabilities at different price points. The Model B has 512MB of RAM and the Model A has 256 MB.

2.3.2. The secure digital (SD) card slot

There’s no hard drive on the Pi; everything is stored on an SD Card. One reason you’ll want some sort of protective case sooner than later is that the solder joints on the SD socket may fail if the SD card is accidentally bent.

2.3.3. The USB Port

On the Model B there are two USB 2.0 ports, but only one on the Model A. Some of the early Raspberry Pi boards were limited in the amount of current that they could provide. Some USB devices can draw up 500mA. The original Pi board supported 100mA or so, but the newer revisions are up to the full USB 2.0 spec. One way to check your board is to see if you have two polyfuses limiting the current. In any case, it is probably not a good idea to charge your cell phone with the Pi. You can use a powered external hub if you have a peripheral that needs more power.

2.3.4. Ethernet Port

The model B has a standard RJ45 Ethernet port. The Model A does not, but can be connected to a wired network by a USB Ethernet adapter (the port on the Model B is actually an onboard USB to Ethernet adapter). WiFi connectivity via a USB dongle is another option.

2.3.5. HDMI Connector

The HDMI port provides digital video and audio output. 14 different video resolutions are supported, and the HDMI signal can be converted to DVI (used by many monitors), composite (analog video signal usually carried over a yellow RCA connector), or SCART (a European standard for connecting audio-visual equipment) with external adapters.

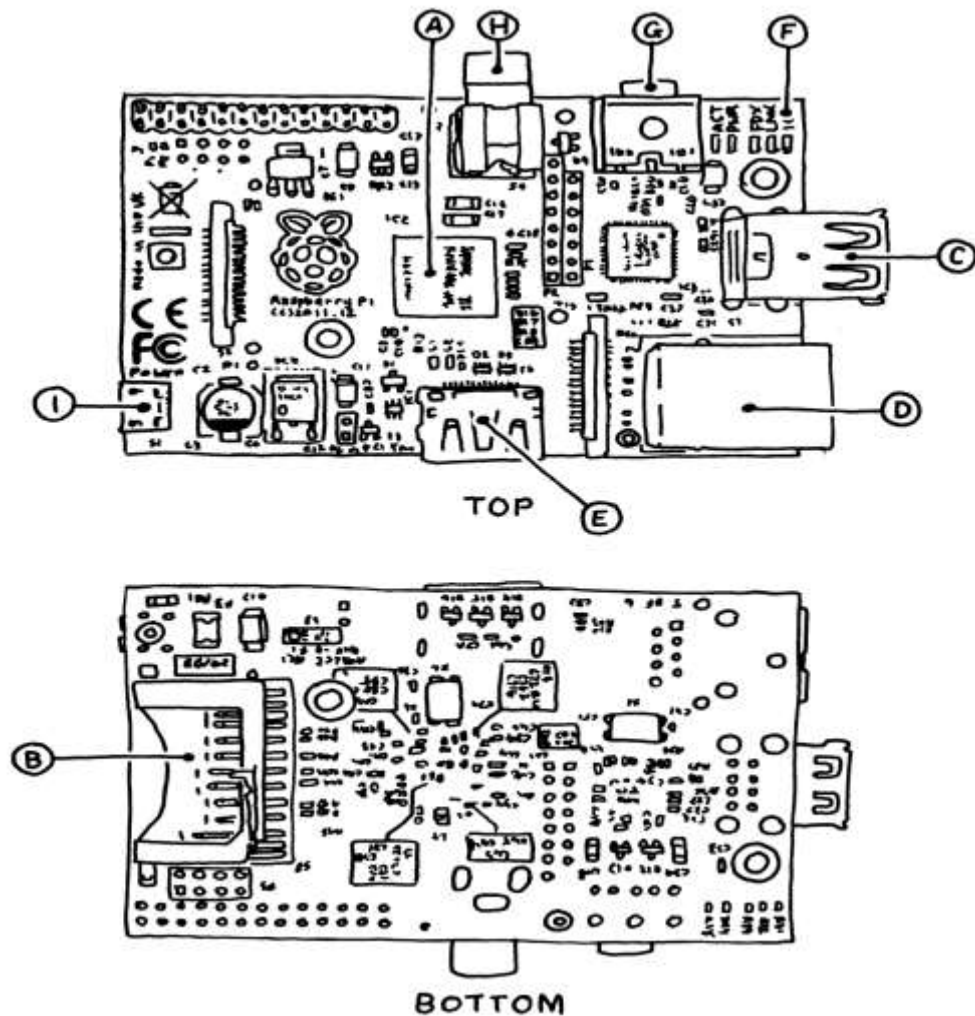


Fig.2.2: A Map of the Hardware Interface of the Raspberry Pi

2.3.6. Status LEDs

The Pi has five indicator LEDs that provide visual feedback.

Table 2.1: The Five status LEDs

ACT	Green	Lights when the SD card is accessed (marked OK on earlier boards)
PWR	Red	Hooked up to 3.3V power
FDX	Green	On if network adapter is full duplex
LNK	Green	Network activity light
100	Yellow	On if the network connection is 100Mbps (some early boards have a 10M misprint)

2.3.7. Analog Audio Output

This is a standard 3.5mm mini analog audio jack, intended to drive high impedance loads (like amplified speakers). Headphones or unpowered speakers won't sound very good; in fact, as of this writing the quality of the analog output is much less than the HDMI audio output you'd get by connecting to a TV over HDMI. Some of this has to do with the audio driver software, which is still evolving.

2.3.8. Composite Video Out

This is a standard RCA-type jack that provides composite NTSC or PAL video signals. This video format is extremely low-resolution compared to HDMI. If you have a HDMI television or monitor, use it rather than a composite television.

2.3.9. Power Input

One of the first things you'll realize is that there is no power switch on the Pi. This microUSB connector is used to supply power (this isn't an additional USB port;

it's only for power). MicroUSB was selected because the connector is cheap USB power supplies are easy to find.

2.4. Peripherals of Raspberry Pi

There are a bunch of pre-packaged starter kits that have well-vetted parts lists; there are a few caveats when fitting out your Raspberry Pi. An Ethernet cable, a powered USB 2.0 Hub, heat sinks, real time clock, camera module, LCD display, Wi-Fi Dongle, and a laptop dock are required.

2.4.1. Power supply

This is the most important peripheral to get right and a microUSB adapter that can provide 5V and at least 700mA of current (500mA for the Model A) should be used. A cell phone charger won't cut it, even if it has the correct connector. A typical cell phone charger only provides 400mA of current or less, but check the rating marked on the back. An underpowered Pi may still seem to work but will be flaky and may fail unpredictably.

With the current version of the Pi board, it is possible to power the Pi from a USB hub that feeds power. However, there isn't much protection circuitry so it may not be the best idea to power it over the USB ports. This is especially true if you're going to be doing electronics prototyping where you may accidentally create shorts that may draw a lot of current.

2.4.2. SD card

At least 4GB SD Card should be used, and it should be a Class 4 card. Class 4 cards are capable of transferring at least 4MB/sec. Some of the earlier Raspberry Pi boards had problems with Class 6 or higher cards, which are capable of faster speeds but are less stable. A microSD card in an adapter is perfectly usable as well.

2.4.3. HDMI Cable

If you're connecting to a monitor you'll need this, or an appropriate adapter for a DVI monitor. You can also run the Pi headless. HDMI cables can vary wildly in price.

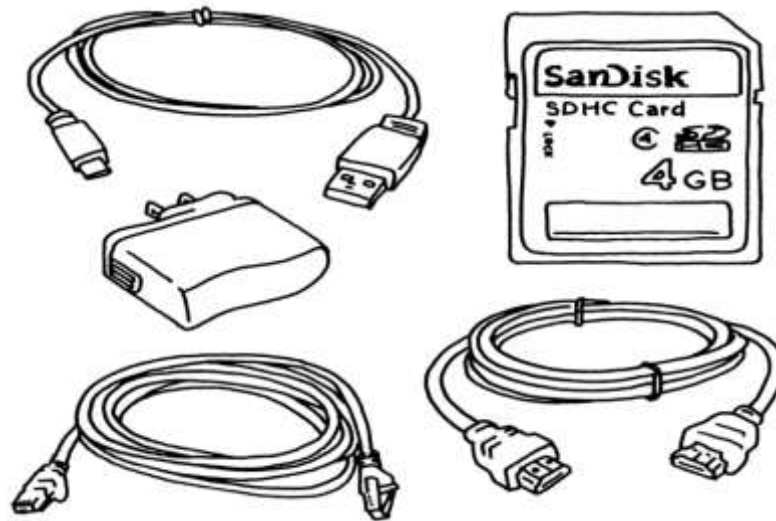


Fig.2.3: The Basic peripherals

2.5. Components of Raspberry Pi

As for the specifications, the Raspberry Pi is a credit card-sized computer powered by the Broadcom BCM2835 system-on-a-chip (SoC). This SoC includes a 32-bit ARM1176JZFS processor, clocked at 700MHz, and a Videocore IV GPU. It also has 256MB of RAM in a POP package above the SoC.

The Raspberry Pi is powered by a 5V micro USB AC charger or at least 4 AA batteries (with a bit of hacking). While the ARM CPU delivers real-world performance similar to that of a 300MHz Pentium 2, the Broadcom GPU is a very capable graphics core capable of hardware decoding several high definition video formats. However, in order to keep costs of the Raspberry Pi low, the UK charity has only licensed the H.264 codec for hardware decoding (and it is unclear if users will be able to purchase/activate additional codecs). In that regard, the Videocore IV GPU is rather potent as it is capable of hardware decoding 1080p30 H.264 with bit-rates up to 40Mb/s.

The Raspberry Pi model available for purchase at the time of writing — the Model B — features HDMI and composite video outputs, two USB 2.0 ports, a 10/100 Ethernet port, SD card slot, GPIO (General Purpose I/O Expansion Board) connector, and analog audio output (3.5mm headphone jack). The less expensive Model A strips out the Ethernet port and one of the USB ports but otherwise has the same hardware. It is this model that is the “\$25 PC” that originally made so many headlines.

2.5.1. Hardware

In the above block diagram for model A, B, A+, B+; model A and A+ are the lowest two blocks and the rightmost block are missing (note that these three blocks are in a chip that actually contains a three-port USB hub, with a USB Ethernet adapter connected to one of its ports). In model A and A+ the USB port is connected directly to the SoC. On model B+ the chip contains a five point hub, with four USB ports fed out, instead of the two on model B.

2.5.1.1. Processor

The SoC used in the first generation Raspberry Pi is somewhat equivalent to the chip used in older smartphones (such as iPhone / 3G / 3GS). The Raspberry Pi is based on the Broadcom BCM2835 system on a chip (SoC), which includes an 700 MHz ARM1176JZF-S processor, Video Core IV GPU, and RAM. It has a Level 2 cache of 128 KB, used primarily by the GPU, not the CPU. The SoC is stacked underneath the RAM chip, so only its edge is visible.

While operating at 700 MHz by default, the first generation Raspberry Pi provided a real world performance roughly equivalent to 0.041 GFLOPS. On the CPU level the performance is similar to a 300 MHz Pentium II of 1997-1999. The GPU provides 1 Gpixel/s or 1.5 Gtexel/s of graphics processing or 24 GFLOPS of general purpose computing performance. The graphics capabilities of the Raspberry Pi are roughly equivalent to the level of performance of the Xbox of 2001.

The LINPACK single node compute benchmark results in a mean single precision performance of 0.065 GFLOPS and a mean double precision performance of 0.041 GFLOPS for one Raspberry Pi Model-B board. The first generation Raspberry Pi chip operated at 700 MHz by default and did not become hot enough to need a heat sink or special cooling, unless the chip was overclocked. The second generation runs on 900 MHz by default, and also does not become hot enough to need a heat sink or special cooling, again overclocking may heat up the SoC more than usual.

Most Raspberry Pi chips could be overclocked to 800 MHz and some even higher to 1000 MHz. There are reports the second generation can be similarly overclocked, in extreme cases, even to 1500 MHz (discarding all safety features and over voltage limitations). In the Raspbian Linux distro the overclocking options on boot

can be done by a software command running "sudo raspi-config" without voiding the warranty, see note 9 below. In those cases the Pi automatically shuts the overclocking down in case the chip reaches 85 °C (185 °F), but it is possible to overrule automatic over voltage and overclocking settings (voiding the warranty). In that case, one can try putting an appropriately sized heat sink on it to keep the chip from heating up far above 85 °C.

Newer versions of the firmware contain the option to choose between five overclock ("turbo") presets that when turned on try to get the most performance out of the SoC without impairing the lifetime of the Pi. This is done by monitoring the core temperature of the chip, and the CPU load, and dynamically adjusting clock speeds and the core voltage. When the demand is low on the CPU, or it is running too hot, the performance is throttled, but if the CPU has much to do, and the chip's temperature is acceptable, performance is temporarily increased, with clock speeds of up to 1 GHz, depending on the individual board, and on which of the turbo settings is used. The five settings are:

- None- 700 MHz ARM, 250 MHz core, 400 MHz SDRAM, 0 overvolt,
- Modest-800 MHz ARM, 250 MHz core, 400 MHz SDRAM, 0 overvolt,
- Medium- 900 MHz ARM, 250 MHz core, 450 MHz SDRAM, 2 overvolt,
- High- 950 MHz ARM, 250 MHz core, 450 MHz SDRAM, 6 overvolt,
- Turbo- 1000 MHz ARM, 500 MHz core, 600 MHz SDRAM, 6 overvolt.

In the highest (turbo) preset the SDRAM clock was originally 500 MHz, but this was later changed to 600 MHz because 500 MHz sometimes causes SD card corruption. Simultaneously in high mode the core clock speed was lowered from 450 to 250 MHz, and in medium mode from 333 to 250 MHz.

2.5.1.2. RAM

On the older beta model B boards, 128 MB was allocated by default to the GPU, leaving 128 MB for the CPU. On the first 256 MB release model B (and model A), three different splits were possible. The default split was 192 MB (RAM for CPU), which should be sufficient for standalone 1080p video decoding, or for simple 3D, but probably not for both together. 224 MB was for Linux only, with just a 1080p framebuffer, and was likely to fail for any video or 3D. 128 MB was for heavy 3D,

possibly also with video decoding (e.g. XBMC). Comparatively the Nokia 701 uses 128 MB for the Broadcom Video Core IV. For the new model B with 512 MB RAM initially there were new standard memory split files released(arm256_start.elf, arm384_start.elf, arm496_start.elf) for 256 MB, 384 MB and 496 MB CPU RAM (and 256 MB, 128 MB and 16 MB video RAM). The second generation has 1 GB of RAM.

2.5.1.3. Networking

Though the model A and A+ do not have an 8P8C ("RJ45") Ethernet port, they can be connected to a network using an external user-supplied USB Ethernet or Wi-Fi adapter. On the model B and B+ the Ethernet port is provided by a built-in USB Ethernet adapter.

2.5.1.4. Peripherals

Generic USB keyboards and mice are compatible with the Raspberry Pi.

2.5.1.5. Video

The video controller is capable of standard modern TV resolutions, such as HD and Full HD, and higher or lower monitor resolutions and older standard CRT TV resolutions; capable of the following: 640×350 EGA; 640×480 VGA; 800×600 SVGA; 1024×768 XGA; 1280×720 720p HDTV; 1280×768 WXGA variant; 1280×800 WXGA variant; 1280×1024 SXGA; 1366×768 WXGA variant; 1400×1050 SXGA+; 1600×1200 UXGA; 1680×1050 WXGA+; 1920×1080 1080p HDTV; 1920×1200 WUXGA. It can generate 576i and 480i composite video signals for PAL-BGHID, PAL-M, PAL-N, NTSC and NTSC-J.

2.5.1.6. Real-time clock

The Raspberry Pi does not come with a real-time clock, which means it cannot keep track of the time of day while it is not powered on. As alternatives, a program running on the Pi can get the time from a network time server or user input at boot time. A real-time clock (such as the DS1307) with battery backup can be added (often via the I²C interface).

2.5.2. Software

The Raspberry Pi primarily uses Linux-kernel-based operating systems. The ARM11 chip at the heart of the Pi (pre-Pi 2) is based on version 6 of the ARM. The

current releases of several popular versions of Linux, including Ubuntu, will not run on the ARM11. It is not possible to run Windows on the original Raspberry Pi, though the new Raspberry Pi 2 will be able to run Windows 10. The Raspberry Pi 2 currently only supports Ubuntu Snappy Core and Raspbian. The install manager for the Raspberry Pi is NOOBS. The operating systems included with NOOBS are:

- Archlinux ARM
- OpenELECPidora (Fedora Remix)
- Puppy Linux
- Raspbmc and the XBMC open source digital media center
- RISC OS– The operating system of the first ARM-based computer
- Raspbian (recommended for Raspberry Pi 1) – Maintained independently of the Foundation; based on the ARM hard-float(armhf) Debian 7 'Wheezy' architecture port originally designed for ARMv7 and later processors (with Jazelle RCT/ThumbEE, VFPv3, and NEON SIMD extensions), compiled for the more limited ARMv6 instruction set of the Raspberry Pi. A minimum size of 4 GB SD card is required. There is a Pi Store for exchanging programs. The Raspbian Server Edition is a stripped version with fewer software packages bundled as compared to the usual desktop computer oriented Raspbian.
- PiBang Linux is derived from Raspbian.
- Raspbian for Robots - A fork of Raspbian for robotics projects with LEGO, Grove, and Arduino.
- Xbian using the Kodi(formerly XBMC) open source digital media center
- Raspberry Pi Fedora Remix
- Slackware ARM – Version 13.37 and later runs on the Raspberry Pi without modification. The 128–496 MB of available memory on the Raspberry Pi is at least twice the minimum requirement of 64 MB needed to run Slackware Linux on an ARM or i386 system. (Whereas the majority of Linux systems boot into a graphical user interface, Slackware's default user environment is the textual shell .The Fluxbox window manager running under the X Window System requires an additional 48 MB of RAM.
- FreeBSD and NetBSD
- Plan 9 from Bell Labs and Inferno (in beta)

- Moebius – A light ARM HF distribution based on Debian. It uses Raspbian repository, but it fits in a 128 MB SD card. It has just minimal services and its memory usage is optimized to keep a small footprint.
- OpenWrt – Primarily used on embedded devices to route network traffic.
- Kali Linux – A Debian-derived distro designed for digital forensics and penetration testing.
- Instant WebKiosk – An operating system for digital signage purposes (web and media views)
- Ark-OS– Website and email self-hosting
- Minepion – Dedicated operating system for mining cryptocurrency
- Kano OS
- Nard SDK For industrial embedded systems

2.6. Installing Linux on the Raspberry Pi

The majority of modern Linux distributions are user-friendly, with a graphical user interface (GUI) that provides an easy way to perform common tasks. It is, however, quite different to both Windows and OS X, so if you're going to get the most out of your Raspberry Pi, you'll need a quick primer in using the operating system.

“The Raspberry Pi”, Linux is an open-source project which was originally founded to produce a kernel that would be free for anyone to use. The kernel is the heart of an operating system, and handles the communication between the user and the hardware. Although only the kernel itself is rightly called Linux, the term is often used to refer to a collection of different open-source projects from a variety of companies. These collections come together to form different flavours of Linux, known as distributions.

The original version of Linux was combined with a collection of tools created by a group called GNU. The resulting system, known as GNU/Linux, was basic but powerful. Unlike other operating systems of the era, it offered facilities like multiple user accounts where several users can share a single computer. That's something rival closed-source operating systems have taken on board, with both Windows and OS X now supporting multiple user accounts on the same system. It's also still present in Linux, and provides security and protection for the operating system.

In Linux, you'll spend most of your time running a restricted user account. This doesn't mean you're being limited in what you can do. Instead, it prevents you from accidentally doing something that will break the software on your Raspberry Pi. It also prevents viruses and other malware from infecting the system by locking down access to critical system files and directories.

2.6.1. Using the Command Line

Before you start, open up the LXTerminal program. There are two tricks that make life much easier in the shell: *auto complete* and *command history*. Often you will only need to type the first few characters of a command or filename, then hit tab. The shell will attempt to auto complete the string based on the files in the current directory or programs in commonly used directories (the shell will search for executable programs in places like /bin or /usr/bin/). If you hit the up arrow on the command line you'll be able to step back through your command history, which is useful if you mistyped a character in a long string of commands.

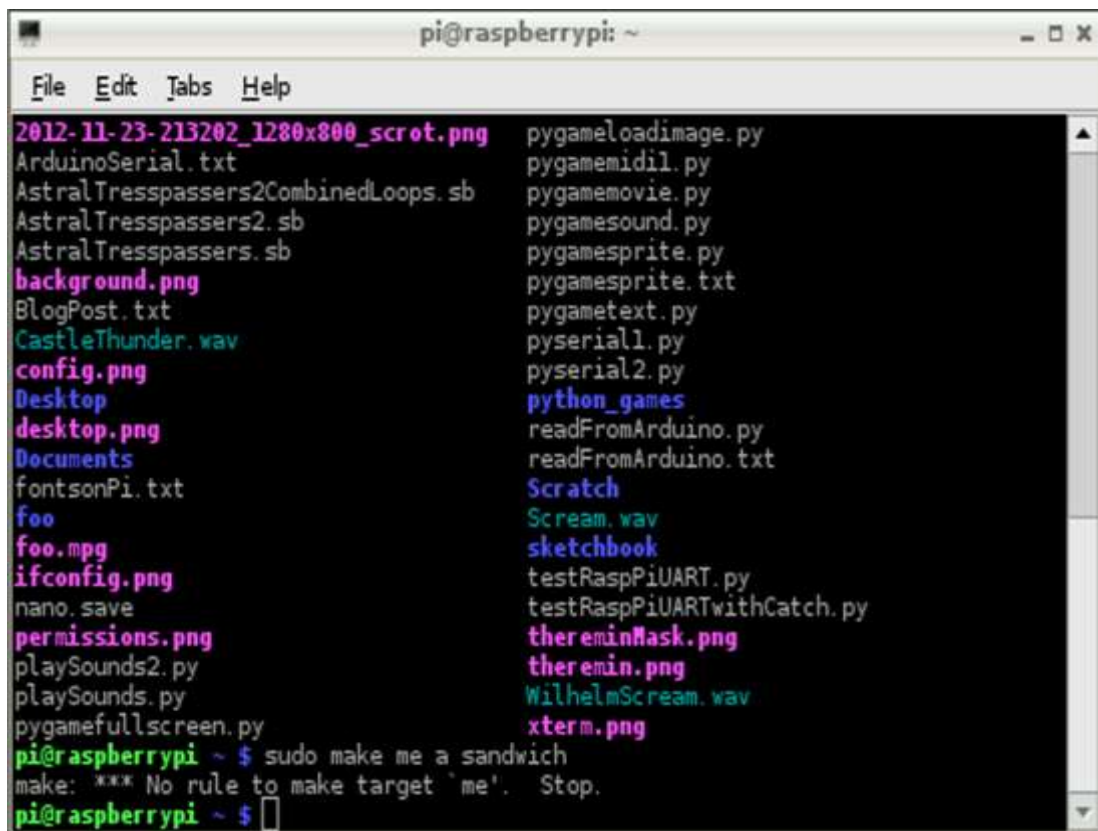


Fig.2.4: LXTerminal Program

Some of the most important directories in the Raspbian filesystem are shown below:

Table 2.2: Directories of Raspbian filesystem

Directory	Description
/bin	Programs and commands that all users can run
/boot	All the files needed at boot time
/dev	Special files that represent the devices on your system
/etc	Configuration files
/etc/init.d	Scripts to start up services
/etc/X11	X11 configuration files
/home	User home directories
/home/pi	Home directory for pi user
/lib	Kernel modules/drivers
/media	Mount points for removable media
/proc	A virtual directory with information about running processes and the OS
/sbin	Programs for system maintenance

2.6.2. Installation

Installing VNC server on Pi allows you to see your Raspberry Pi's desktop remotely in a graphical way, using the mouse as if you were sitting in front of your Pi. It also means you can put your Pi somewhere else on the network, but still control it.

Also, internet can be shared from laptop's WiFi over Ethernet. This helps in accessing internet on Raspberry Pi.

Step 1: Setting up your Raspberry Pi.

After getting Raspberry Pi you need SD Card which has desired OS. You will find lots of blogs and links about making SD card for Raspberry Pi. You can follow any of them to make your own SD Card, but if you want, you can check out this link. This has very good start up guide for Raspberry Pi. As you are ready with your SD Card insert it into Raspberry Pi. Attach Micro USB Power Cable with it. Also attaché your Raspberry Pi with your Laptop via Ethernet Cable. Connect Keyboard & Mouse with it. Also connect HDMI Display. Now power on your Pi.

Step 2: Sharing Internet over Ethernet.

For sharing internet for multiple users over Ethernet, go to Network and Sharing Center. Click on the WiFi network. Click on Properties, then go to Sharing and click on Allow other network users to connect. Make sure that networking connection is changed to Local Area Connection.

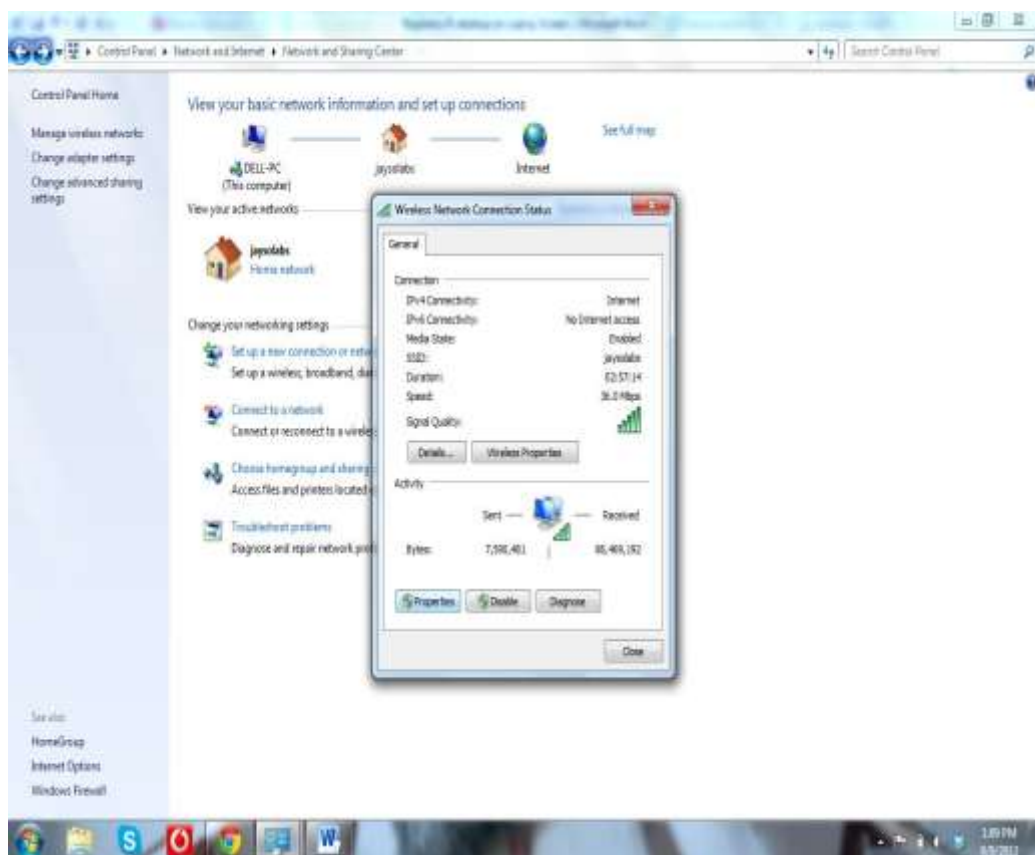


Fig.2.5: Windows showing local area connection

The IP assigned to your laptop is 192.168.137.1

Step 3: For Checking the IP assigned to the connected Ethernet Device.

Consider that IP assigned to your Laptop is 192.168.137.1 and subnet mask is 255.255.255.0

- Open command prompt.
- Ping on broadcast address of your IP i.e. ping 192.168.137.255
- Stop the ping after 5 seconds.
- Watch device replies : arp -a

HDMI Display: Now you should install VNC server in Raspberry Pi. For that if you have HDMI display and Raspbian OS running on it, open LX-Terminal and type following commands to install VNC.

```
$ sudo apt-get update
```

```
$ sudo apt-get install tightvncserver
```

Don't have Display : If you do not arrange display even for one time setup than also no need to worry you can install Putty as per your windows configuration and via SSH you can connect with your Raspberry PI. As you will get access of your Pi terminal run following command to install VNC.

```
$ sudo apt-get update
```

```
$ sudo apt-get install tightvncserver
```

Start VNC Server on Pi: For starting VNC, enter the following command in SSH terminal,

```
$ vncserver :1
```

You will be prompted to enter and confirm a password. This will be asked only once, during one time setup. Enter an 8 digit password. Note that this is the password that you will need to use to connect to the Raspberry Pi remotely.

You will also be asked if you want to create a separate “read-only” password – say no (n).

The VNC server is now running on your Pi and so we can attempt to connect to it, but first we must switch to the computer from which we want to control the Pi and setup a VNC client to connect to the Pi.

Step 4: Client Side (Laptop)

Download VNC client from <http://www.realvnc.com/download/vnc/> and install it. When you first run VNC Viewer, you will see following:



Fig.2.6: VNC Viewer

Enter IP address of your Raspberry Pi given dynamically by your laptop and append with: 1 (denoting port number) and press connect. You will get a warning message, press 'Continue'.

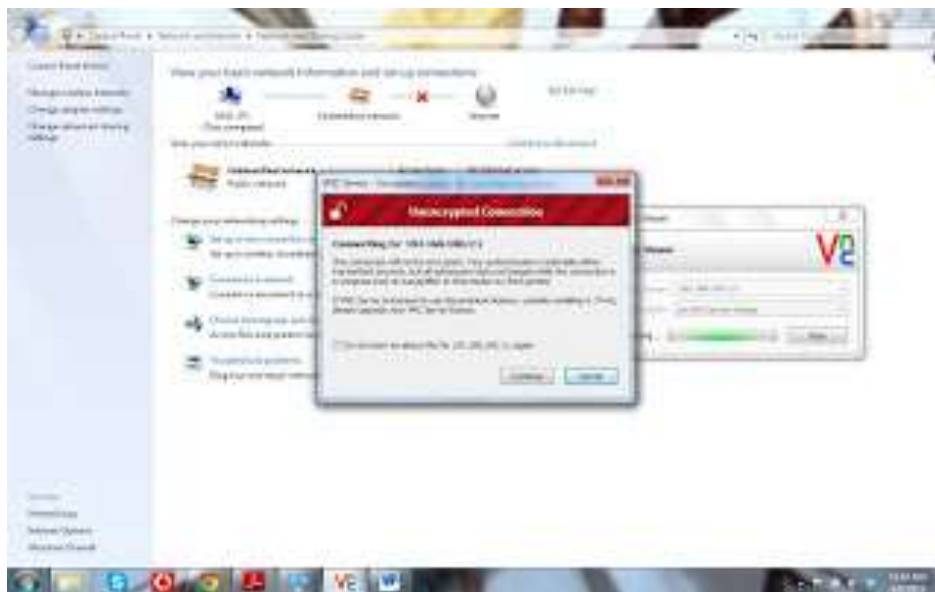


Fig.2.7: Warning message shown

Enter the 8 digit password which was entered in VNC server installation on Raspberry Pi.



Fig.2.8: Enter password

Finally, the VNC window itself should appear. You will be able to use the mouse and do everything as if you were using the Pi's keyboard mouse and monitor, except through your other computer. As with SSH, since this is working over your network, your Pi could be situated anywhere, as long as it is connected to your network.

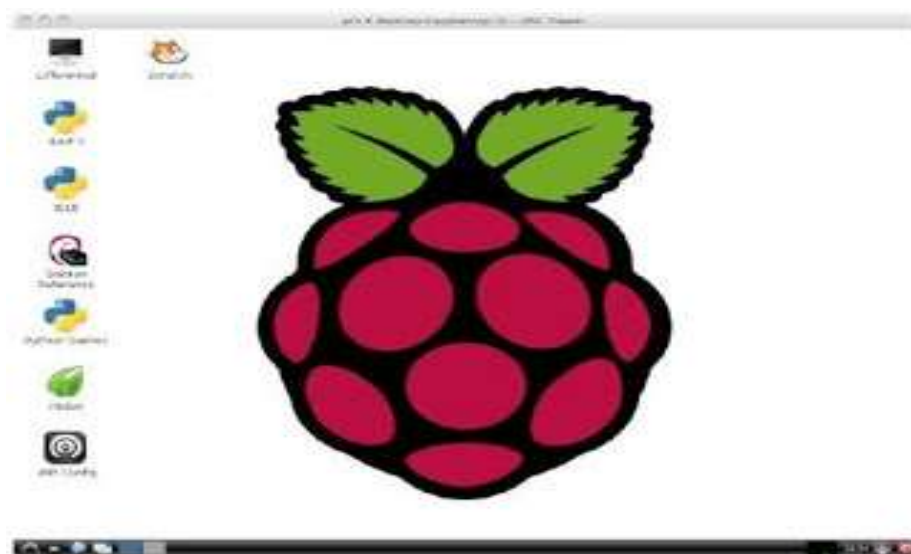


Fig.2.9: VNC window

Step 5: Running VNC server at start-up in Raspberry Pi

Connecting to your Raspberry Pi remotely with VNC is fine as long as your Pi does not reboot. If it does, then you either have to connect with SSH and restart the VNC Server or arrange for the VNC Server to run automatically after the Raspberry Pi reboots. To ensure that VNC starts automatically, run following commands on terminal. Open “.config” folder from Pi user folder. (It is hidden folder)

```
$ cd /home/pi
```

```
$ cd .config
```

Create folder called “autostart” in it. Also create file called “tightvnc.desktop” in that folder. To create file you can use any known text editor. Here I use gnome-text-editor for this.

```
$ mkdir autostart
```

```
$ cd autostart
```

```
$ gnome tightvnc.desktop
```

Edit the contents of file with following text and save the file.

```
[Desktop Entry]
```

```
Type=Application
```

```
Name=TightVNC
```

```
Exec=vncserver :1
```

```
StartupNotify=false
```

CHAPTER-3

VOICE RECOGNITION

3.1. Speech

3.1.1. Basic concepts of speech

Speech is a complex phenomenon. People rarely understand how is it produced and perceived. The naive perception is often that speech is built with words, and each word consists of phones. The reality is unfortunately very different. Speech is a dynamic process without clearly distinguished parts. It's always useful to get a sound editor and look into the recording of the speech and listen to it. The following is an example of a speech recording in an audio editor.

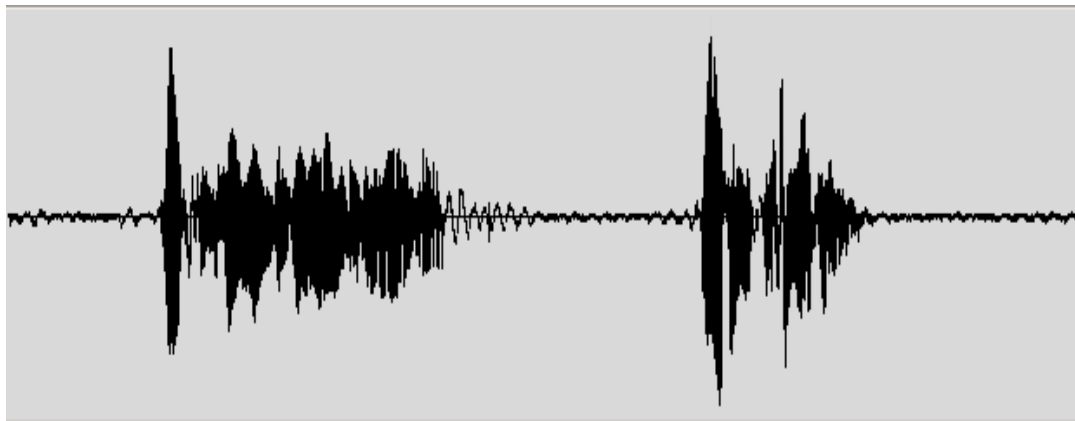


Fig.3.1: A Speech recording in Analog Form

All modern descriptions of speech are to some degree probabilistic. That means that there are no certain boundaries between units, or between words. Speech to text translation and other applications of speech are never 100% correct. That idea is rather unusual for software developers, who usually work with deterministic systems. And it creates a lot of issues specific only to speech technology.

3.1.2. Structure of speech

In current practice, speech structure is understood as follows:

Speech is a continuous audio stream where rather stable states mix with dynamically changed states. In this sequence of states, one can define more or less similar classes of sounds, or phones. Words are understood to be built of phones, but

this is certainly not true. The acoustic properties of a waveform corresponding to a phone can vary greatly depending on many factors - phone context, speaker, style of speech and so on. The so called co-articulation makes phones sound very different from their “canonical” representation. Next, since transitions between words are more informative than stable regions, developers often talk about diphones - parts of phones between two consecutive phones. Sometimes developers talk about sub-phonetic units - different substates of a phone. Often three or more regions of a different nature can easily be found. The number three is easily explained. The first part of the phone depends on its preceding phone, the middle part is stable, and the next part depends on the subsequent phone. That's why there are often three states in a phone selected for speech recognition.

Sometimes phones are considered in context. Such phones in context are called triphones or even quinphones. For example “u with left phone b and right phone d” in the word “bad”. And it sounds a bit different from the same phone “u” with left phone b and right phone n “in word “ban”. Please note that unlike diphones, they are matched with the same range in waveform as just phones. They just differ by name because they describe slightly different sounds. For computational purpose it is helpful to detect parts of triphones instead of triphones as a whole, for example, to create a detector for a beginning of triphone and share it across many triphones.

The whole variety of sound detectors can be represented by a small amount of distinct short sound detectors. Usually we use 4000 distinct short sound detectors to compose detectors for triphones. We call those detectors senones. A senone's dependence on context could be more complex than just left and right context. It can be a rather complex function defined by a decision tree, or in some other way.

Next, phones build subword units, like syllables. Sometimes, syllables are defined as “reduction-stable entities”. To illustrate, when speech becomes fast, phones often change, but syllables remain the same. Also, syllables are related to intonational contour. There are other ways to build subwords - morphologically-based in morphology-rich languages or phonetically-based. Subwords are often used in open vocabulary speech recognition. Subwords form words. Words are important in speech recognition because they restrict combinations of phones significantly. If there are 40 phones and an average word has 7 phones, there must be 40^7 words. Luckily, even a

very educated person rarely uses more than 20k words in his practice, which makes recognition way more feasible. Words and other non-linguistic sounds, which we call fillers (breath, um, uh, cough), form utterances. They are separate chunks of audio between pauses. They don't necessary match sentences, which are more semantic concepts. On the top of this, there are dialog acts like turns, but they go beyond the purpose of the document.

3.1.3. Recognition process

The common way to recognize speech is the following: we take waveform, split it on utterances by silences then try to recognize what's being said in each utterance. To do that we want to take all possible combinations of words and try to match them with the audio. We choose the best matching combination. There are few important things in this match.

First of all it's a concept of features. Since number of parameters is large, we are trying to optimize it. Numbers that are calculated from speech usually by dividing speech on frames. Then for each frame of length typically 10 milliseconds we extract 39 numbers that represent the speech. That's called feature vector. The way to generate numbers is a subject of active investigation, but in simple case it's a derivative from spectrum.

Second it's a concept of the model. Model describes some mathematical object that gathers common attributes of the spoken word. In practice, for audio model of senone is gaussian mixture of it's three states - to put it simple, it's a most probable feature vector. From concept of the model the following issues raised - how good does model fits practice, can model be made better of it's internal model problems, how adaptive model is to the changed conditions.

The model of speech is called Hidden Markov Model or HMM, it's a generic model that describes black-box communication channel. In this model process is described as a sequence of states which change each other with certain probability. This model is intended to describe any sequential process like speech. It has been proven to be really practical for speech decoding. Third, it's a matching process itself. Since it would take a huge time more than universe existed to compare all feature vectors with all models, the search is often optimized by many tricks. At any points we maintain best

matching variants and extend them as time goes producing best matching variants for the next frame.

3.1.4. Models

According to the speech structure, three models are used in speech recognition to do the match:

- An acoustic model contains acoustic properties for each senone. There are context-independent models that contain properties (most probable feature vectors for each phone) and context-dependent ones (built from senones with context).
- A phonetic dictionary contains a mapping from words to phones. This mapping is not very effective. For example, only two to three pronunciation variants are noted in it, but it's practical enough most of the time. The dictionary is not the only variant of mapper from words to phones. It could be done with some complex function learned with a machine learning algorithm.
- A language model is used to restrict word search. It defines which word could follow previously recognized words (remember that matching is a sequential process) and helps to significantly restrict the matching process by stripping words that are not probable. Most common language models used are n-gram language models-these contain statistics of word sequences-and finite state language models-these define speech sequences by finite state automation, sometimes with weights. To reach a good accuracy rate, your language model must be very successful in search space restriction. This means it should be very good at predicting the next word. A language model usually restricts the vocabulary considered to the words it contains. That's an issue for name recognition. To deal with this, a language model can contain smaller chunks like subwords or even phones. Please note that search space restriction in this case is usually worse and corresponding recognition accuracies are lower than with a word-based language model.

Those three entities are combined together in an engine to recognize speech. If you are going to apply your engine for some other language, you need to get such structures in place. For many languages there are acoustic models, phonetic dictionaries and even large vocabulary language models available for download.

3.1.5. Other Concepts

- A **Lattice** is a directed graph that represents variants of the recognition. Often, getting the best match is not practical; in that case, lattices are good intermediate formats to represent the recognition result.
- **N-best** lists of variants are like lattices, though their representations are not as dense as the lattice ones.
- **Word confusion networks** (sausages) are lattices where the strict order of nodes is taken from lattice edges.
- **Speech database** - a set of typical recordings from the task database. If we develop dialog system it might be dialogs recorded from users. For dictation system it might be reading recordings. Speech databases are used to train, tune and test the decoding systems.
- **Text databases** - sample texts collected for language model training and so on. Usually, databases of texts are collected in sample text form. The issue with collection is to put present documents (PDFs, web pages, scans) into spoken text form. That is, you need to remove tags and headings, to expand numbers to their spoken form, and to expand abbreviations.

3.1.6. Optimization

When speech recognition is being developed, the most complex issue is to make search precise (consider as many variants to match as possible) and to make it fast enough to not run for ages. There are also issues with making the model match the speech since models aren't perfect. Usually the system is tested on a test database that is meant to represent the target task correctly. The following characteristics are used:

- **Word error rate:** Let we have original text and recognition text of length of N words. From them the I words were inserted D words were deleted and S words were substituted Word error rate is

$$\text{WER} = (I + D + S) / N$$

WER is usually measured in percent.

- **Accuracy:** It is almost the same thing as word error rate, but it doesn't count insertions.

$$\text{Accuracy} = (N - D - S) / N$$

Accuracy is actually a worse measure for most tasks, since insertions are also important in final results. But for some tasks, accuracy is a reasonable measure of the decoder performance.

- **Speed:** Suppose the audio file was 2 hours and the decoding took 6 hours. Then speed is counted as 3 x RT.
- **ROC curves:** When we talk about detection tasks, there are false alarms and hits/misses; ROC curves are used. A curve is a graphic that describes the number of false alarms vs number of hits, and tries to find optimal point where the number of false alarms is small and number of hits matches 100%.

3.2. Voice Recognition Softwares

Voice or speech recognition is the ability of a machine or program to receive and interpret dictation, or to understand and carry out spoken commands. The field of computer science that deals with designing computer systems that can recognize spoken words. Note that voice recognition implies only that the computer can take dictation, not that it understands what is being said. Comprehending human languages falls under a different field of computer science called processing.

A number of voice recognition systems are available on the market. The most powerful can recognize thousands of words. However, they generally require an extended training session during which the computer system becomes accustomed to a particular voice and accent. Such systems are said to be speaker dependent.

Many systems also require that the speaker speak slowly and distinctly and separate each word with a short pause. These systems are called discrete speech systems. Recently, great strides have been made in continuous speech systems -- voice recognition systems that allow you to speak naturally. There are now several continuous-speech systems available for personal computers.



Fig.3.2: Block Diagram of a Voice Recognition Software

Because of their limitations and high cost, voice recognition systems have traditionally been used only in a few specialized situations. For example, such systems are useful in instances when the user is unable to use a keyboard to enter data because his or her hands are occupied or disabled. Instead of typing commands, the user can simply speak into a headset. Increasingly, however, as the cost decreases and performance improves, speech recognition systems are entering the mainstream and are being used as an alternative to keyboards.

There are many voice recognition softwares. Some of them are given below:

- **Julius:** It is an open source speech recognition engine. Julius is a high-performance, two-pass large vocabulary continuous speech recognition (LVCSR) decoder software for speech-related researchers and developers. It can perform almost real-time decoding on most current PCs in 60k word dictation task using word 3-gram and context-dependent HMM. Major search techniques are fully incorporated. It is also modularized carefully to be independent from model structures, and various HMM types are supported such as shared-state triphones and tied-mixture models, with any number of mixtures, states, or phones. Standard formats are adopted to cope with other free modeling toolkit. The main platform is Linux and other Unix workstations, and also works on Windows. Julius is open source and distributed with a revised BSD style license.

- **Kaldi:** It is a speech recognition toolkit, freely available under the Apache License. Kaldi aims to provide software that is flexible and extensible. It supports linear transforms, MMI, boosted MMI and MCE discriminative training, feature-space discriminative training, and deep neural networks.
- **RWTH ASR:** It is a proprietary speech recognition toolkit. The toolkit includes newly developed speech recognition technology for the development of automatic speech recognition systems. It has been developed by the Human Language Technology and Pattern Recognition Group at RWTH Aachen University. RWTH ASR includes tools for the development of acoustic models and decoders as well as components for speaker adaptation, speaker adaptive training, unsupervised training, discriminative training, and word lattice processing. The software runs on Linux and Mac OS X. The project homepage offers ready-to-use models for research purposes, tutorials, and **comprehensive documentation**.
- **CMU Sphinx:** It is also called Sphinx in short, is the general term to describe a group of speech recognition systems developed at Carnegie Mellon University. These include a series of speech recognizers (Sphinx 2 - 4) and an acoustic model trainer (SphinxTrain).
- **HTK (Hidden Markov Model Toolkit):** It is software toolkit for handling HMMs. It is mainly intended for speech recognition, but has been used in many other pattern recognition applications that employ HMMs, including speech synthesis, character recognition and DNA sequencing. Originally developed at the Machine Intelligence Laboratory (formerly known as the Speech Vision and Robotics Group) of the Cambridge University Engineering Department (CUED), HTK is now being widely used among researchers who are working on HMMs.
- **Google STT:** It is the speech-to-text system by Google. If you have an Android smartphone, you might already be familiar with it, because it's basically the same engine that performs recognition if you say *OK, Google*. It can only transcribe a limited amount of speech a day and needs an active internet connection.
- **AT&T STT:** It is a speech decoder by the telecommunications company AT&T. Like Google Speech, it also performs decoding online and thus needs an active internet connection.

- **Wit.ai STT:** It relies on the wit.ai cloud services and uses crowdsourcing to train speech recognition algorithms. Like you'd expect from a cloud service, you also need an active internet connection.

3.3. Google API Voice Recognition Software

Voice Recognition can be achieved in various ways. One of the easiest ways to achieve this is by using Google API. The software being described here uses Google Voice and speech APIs. The voice command from the user is captured by the microphone. This is then converted to text by using Google voice API. The text is then compared with the other previously defined commands inside the commands configuration file. If it matches with any of them, then the bash command associated with it will be executed. You can also use this system as an interactive voice response system by making the raspberry pi respond to your commands via speech. This is achieved by using the Google speech API, which converts the text into speech. It basically converts our spoken question into to text, process the query and return the answer, and finally turn the answer from text to speech. This is divided into four parts:

1. Speech to Text
2. Query Processing
3. Text to Speech
4. Combining them Together

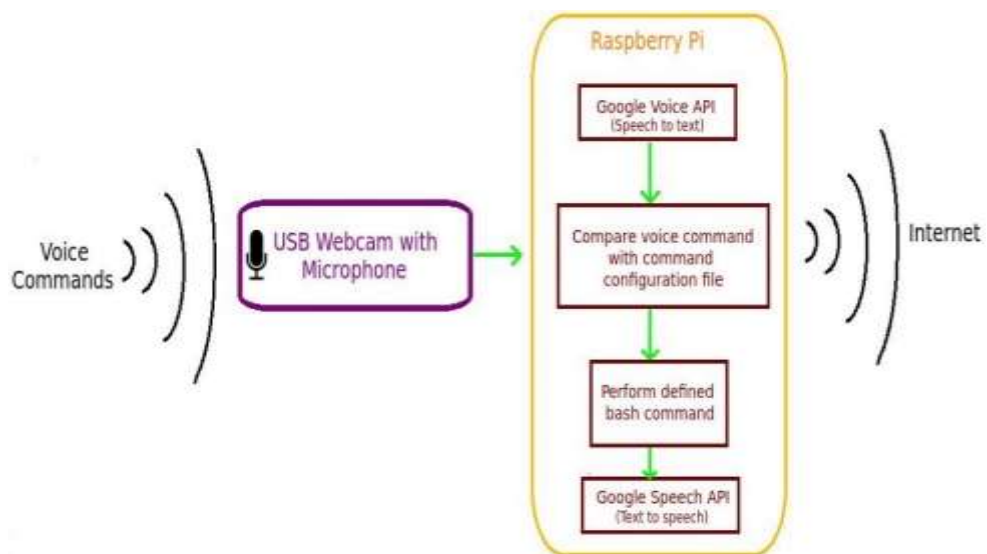


Fig.3.3: Voice Recognition Software for Raspberry Pi

The Hardware components required are:

1. Raspberry Pi Board
2. An SD Card with Pre-Installed Operating System
3. A USB Microphone
4. A Wi-Fi Dongle or an Ethernet Cable.

3.3.1. Speech to Text

Speech recognition can be achieved in many ways on Linux (so on the Raspberry Pi), The easiest way is to use Google voice recognition API. as the accuracy is very good. To ensure recording is setup, ffmpeg must be installed:

```
sudo apt-get install ffmpeg
```

To use the Google's voice recognition API, I use the following bash script. Copy the following text and save it as '*speech2text.sh*'

```
#!/bin/bash
echo "Recording... Press Ctrl+C to Stop."
arecord -D "plughw:1,0" -q -f cd -t wav | ffmpeg -loglevel panic -y -i - -ar 16000 -
acodec flac file.flac > /dev/null 2>&1
echo "Processing..."
wget -q -U "Mozilla/5.0" --post-file file.flac --header "Content-Type: audio/x-flac;
rate=16000" -O - "http://www.google.com/speech-api/v1/recognize?lang=en-
us&client=chromium" | cut -d\" -f12 >stt.txt
echo -n "You Said: "
cat stt.txt
rm file.flac > /dev/null 2>&1
```

As soon as it is executed, it starts recording and save the audio in a flac file. The Recording can be stopped by pressing CTRL+C. The audio file is then sent to Google for conversion and text will be returned and saved in a file called "stt.txt". And the audio file will be deleted.

And to make it executable.

```
chmod +x speech2text.sh
```

To run it

```
./speech2text.sh
```

3.3.2. Query Processing

Processing the query works similar to searching a question in Google, but in this scenario only one answer must be returned. Wolfram Alpha seems to be a good choice here. It is a Python interface library. The API allows clients to submit free-form queries similar to the queries one might enter at the Wolfram Alpha website, and for the computed results to be returned in a variety of formats. The API is implemented in a standard REST protocol using HTTP GET requests. Each result is returned as a descriptive XML structure wrapping the requested content format.

3.3.2.1. Installing Wolframalpha Python Library

Download package from <https://pypi.python.org/pypi/wolframalpha>, unzip it somewhere. And then you need to install setuptools and build the setup.

```
apt-get install python-setuptools easy_install pip
sudo python setup.py build
```

And finally run the setup.

```
sudo python setup.py
```

3.3.2.2. Getting the App_ID

You should now be signed in to the Wolfram Alpha Developer Portal and, on the My Apps tab, click the “Get an AppID” button and fill out the “Get a New AppID” form. Use any Application name and description you like. Click the “Get AppID” button.

3.3.2.3. Wolframalpha Python Interface

Save this Python script as “*queryprocess.py*”.

```
#!/usr/bin/python
import wolframalpha
import sys

# Get a free API key here http://products.wolframalpha.com/api/
# This is a fake ID, go and get your own, instructions on my blog.
app_id='HYO4TL-A9QOUALOPX'
client = wolframalpha.Client(app_id)
query = ' '.join(sys.argv[1:])
res = client.query(query)
```

```

if len(res.pods) > 0:
    texts = ""
    pod = res.pods[1]
    if pod.text:
        texts = pod.text
    else:
        texts = "I have no answer for that"
    # to skip ascii character in case of error
    texts = texts.encode('ascii', 'ignore')
    print texts
else:

```

3.3.3. Text To Speech

From the processed query, we are returned with an answer in text format. What we need to do now is turning the text to audio speech. There are a few options available like Cepstral or Festival, but Google's speech service is chosen due to its excellent quality. First of all, to play audio we need to install mplayer:

```
sudo apt-get install mplayer
```

We have this simple bash script. It downloads the MP3 file via the URL and plays it. Copy and name it as "*text2speech.sh*":

```

#!/bin/bash

say() { local IFS=+;/usr/bin/mplayer -ao alsa -really-quiet -noconsolecontrols
"http://translate.google.com/translate_tts?tl=en&q=$*"; }

say $*

```

And to make it executable.

```
chmod +x text2speech.sh
```

To test it, you can try

```
./text2speech.sh "My name is Oscar and I am testing the audio."
```

3.3.3.1. Google Text To Speech Text Length Limitation

Although, Google Text to Speech is a great service, there is a limit on the length of the message of about 100 characters. To work around this, here is an upgraded bash

script that breaks up the text into multiple parts so each part is no longer than 100 characters, and each parts can be played successfully. The modified script can be seen as below.

```
#!/bin/bash

INPUT=$*

STRINGNUM=0

ary=($INPUT)

for key in "${!ary[@]}"
do
    SHORTTMP[$STRINGNUM]="${SHORTTMP[$STRINGNUM]} ${ary[$key]}"
    LENGTH=$(echo ${#SHORTTMP[$STRINGNUM]})
    if [[ "$LENGTH" -lt "100" ]]; then
        SHORT[$STRINGNUM]="${SHORTTMP[$STRINGNUM]}"
    else
        STRINGNUM=$((STRINGNUM+1))
        SHORTTMP[$STRINGNUM]="${ary[$key]}"
        SHORT[$STRINGNUM]="${ary[$key]}"
    fi
done

for key in "${!SHORT[@]}"
do
    say() { local IFS=+;/usr/bin/mplayer -ao alsa -really-quiet -noconsolecontrols
"http://translate.google.com/translate_tts?tl=en&q=${SHORT[$key]}"; }
    say $*
done
```

3.3.4. Putting It Together

For all of these scripts to work together, we have to call them in an another script and “*main.sh*”.

```
#!/bin/bash

echo "Recording... Press Ctrl+C to Stop."

./speech2text.sh

QUESTION=$(cat stt.txt)
```

```

echo "Me: ", $QUESTION
ANSWER=$(python queryprocess.py $QUESTION)
echo "Robot: ", $ANSWER
./text2speech.sh $ANSWER

```

To make *main.sh* executable, run it

```

chmod +x text2speech.sh
./main.sh

```

This idea of combining the Speech recognition ability on the Raspberry Pi with the powerful digital/analog I/O hardware, to build a useful voice control system, can be adopted in Robotics and Home Automation.

3.4. Pocketsphinx

Pocket Sphinx is a lightweight speech recognition engine. A version of Sphinx that can be used in embedded systems like those based on an ARM processor. Incorporates features such as fixed-point arithmetic and efficient algorithms for GMM computation. PocketSphinx first originated as CMU Sphinx is a group of speech recognition systems developed at Carnegie Mellon University. These include a series of speech recognizers (Sphinx 2-4) and and acoustic model trainer (Sphinx Train).

In 2000, the Sphinx group at Carnegie Mellon committed to open source several speech recognizer components, including Sphinx 2 and later Sphinx 3 (in 2001). The available resources include in addition software for acoustic model training, Language model compilation and a public-domain pronunciation dictionary, cmudict.

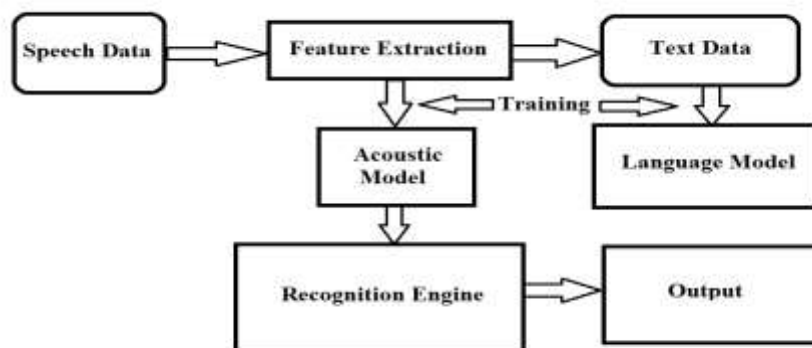


Fig.3.4: Block Diagram of PocketSphinx

Sphinx encompasses a number of software systems, described below.

▪ **Sphinx:**

Sphinx is a continuous-speech, speaker-independent recognition system making use of hidden Markov acoustic models (HMMs) and an n-gram statistical language model. It was developed by Kai-Fu Lee. Sphinx featured feasibility of continuous-speech, speaker-independent large-vocabulary recognition, the possibility of which was in dispute at the time (1986). Sphinx is of historical interest only; it has been superseded in performance by subsequent versions.

▪ **Sphinx 2:**

A fast performance-oriented recognizer, originally developed by Xuedong Huang at Carnegie Mellon and released as Open source with a BSD-style license on SourceForge by Kevin Lenzo at LinuxWorld in 2000. Sphinx 2 focuses on real-time recognition suitable for spoken language applications. As such it incorporates functionality such as end-pointing, partial hypothesis generation, dynamic language model switching and so on. It is used in dialog systems and language learning systems. It can be used in computer based PBX systems such as Asterisk. Sphinx 2 code has also been incorporated into a number of commercial products. It is no longer under active development (other than for routine maintenance). Current real-time decoder development is taking place in the Pocket Sphinx project.

▪ **Sphinx 3:**

Sphinx 2 used a semi-continuous representation for acoustic modeling (i.e., a single set of Gaussians is used for all models, with individual models represented as a weight vector over these Gaussians). Sphinx 3 adopted the prevalent continuous HMM representation and has been used primarily for high-accuracy, non-real-time recognition. Recent developments (in algorithms and in hardware) have made Sphinx 3 "near" real-time, although not yet suitable for critical interactive applications. Sphinx 3 is under active development and in conjunction with SphinxTrain provides access to a number of modern modeling techniques, such as LDA/MLLT, MLLR and VTLN, that improve recognition accuracy.

▪ **Sphinx 4:**

Sphinx 4 is a complete re-write of the Sphinx engine with the goal of providing a more flexible framework for research in speech recognition, written entirely in the Java

programming language. Sun Microsystems supported the development of Sphinx 4 and contributed software engineering expertise to the project. Participants included individuals at MERL, MIT and CMU.

Current development goals include:

- developing a new (acoustic model) trainer
- implementing speaker adaptation (e.g. MLLR)
- improving configuration management
- creating a graph-based UI for graphical system design

▪ **PocketSphinx:**

A version of Sphinx that can be used in embedded systems (e.g., based on an ARM processor). PocketSphinx is under active development and incorporates features such as fixed-point arithmetic and efficient algorithms for GMM computation.

3.4.1. Features of PocketSphinx

The main features of Pocketsphinx are given below:

- All the processing takes place on the Raspberry Pi, so it is capable of being used offline.
- It supports real time speech recognition.
- Pocketsphinx is resource-efficient.
- Pocketsphinx supports many languages out-of box. It supports US English, Chinese, French, Russian, German, Dutch and more without need to train anything.
- Pocketsphinx is completely free software.
- Available bindings for several programming languages are present.

3.4.2. Installation

Pocketsphinx is a library that depends on another library called SphinxBase which provides common functionality across all CMUSphinx projects. To install Pocketsphinx, you need to install both Pocketsphinx and Sphinxbase. It's possible to use Pocketsphinx both in Linux, Windows, on MacOS, iPhone and Android. To build pocketsphinx in a unix-like environment (such as Linux, Solaris, FreeBSD etc) you need to make sure you have the following dependencies installed: gcc, automake, autoconf, libtool, bison, swig at least version 2.0, python development package, pulseaudio development package. If you want to build without dependencies you can

use proper configure options like `--without-swig-python` but for beginner it is recommended to install all dependencies. Install is simple, we need to setup and properly configure alsa, then you can just build and run `pocketsphinx`

```
sudo apt-get update
sudo apt-get upgrade
cat /proc/asound/cards
```

Check your microphone is visible or not and if on which usb extension.

```
sudo nano /etc/modprobe.d/alsa-base.conf
```

Now change this

```
# Keep snd-usb-audio from being loaded as first soundcard
options snd-usb-audio index=-2
```

To

```
options snd-usb-audio index=0
```

If there is some other options `snd-usb-audio index=1`, comment it out

```
sudo reboot
cat /proc/asound/cards
check your device is at 0
sudo apt-get install bison
sudo apt-get install libasound2-dev
```

Download sphinxbase latest , extract

```
./configure --enable-fixed
make
sudo make install
```

Download pocketsphinx, extract

```
./configure
make
sudo make install
export LD_LIBRARY_PATH=/usr/local/lib
export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig
```

To Test PocketSphinx, Run the following in the terminal

```
pocketsphinx_continuous -samprate 16000/8000/48000
```

3.5. Modes of Operation

Using voice recognition software we have operated LED's in four modes:

1. On Mode
2. Off Mode
3. Blink Mode
4. Multiple LED's Mode

3.5.1. Program for On Mode

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7,GPIO.OUT)
GPIO.output(7,True)
```

3.5.2. Program for Off Mode

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7,GPIO.OUT)
GPIO.output(7,False)
```

3.5.3. Program for Blink Mode

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7,GPIO.OUT)
for x in range(0,5):
    GPIO.output(7,True)
    time.sleep(0.5)
    GPIO.output(7,False)
    time.sleep(0.5)
GPIO.cleanup()
```

3.5.4. Program for Multiple LEDs Mode

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7,GPIO.OUT)
GPIO.setup(11,GPIO.OUT)
GPIO.setup(12,GPIO.OUT)
for x in range(0,5):
    GPIO.output(7,True)
    time.sleep(2)
    GPIO.output(11,True)
    time.sleep(2)
    GPIO.output(12,True)
    time.sleep(2)
for x in range(0,5):
    GPIO.output(7,False)
    time.sleep(2)
    GPIO.output(11,False)
    time.sleep(2)
    GPIO.output(12,False)
    time.sleep(2)
GPIO.cleanup()
```

3.5.5. Voice Recognition Program controlling modes of operation

```
import time
import sys, select, subprocess
import os
proc = subprocess.Popen(['sh', '-c', 'pocketsphinx_continuous -adcdev
hw:1,0 -nfft 2048 -samprate 48000 2>/dev/null'], stdout=subprocess.
PIPE)
while True:
    line = proc.stdout.readline()
```

```
if line != "":
    output = line.rstrip()
    print output
    if (len(output.split("john"))>1):
        os.system('sudo python /home/pi/on.py')
    if (len(output.split("off"))>1):
        os.system('sudo python /home/pi/off.py')
    if (len(output.split("light"))>1):
        os.system('sudo python /home/pi/leds.py')
    if (len(output.split("hi"))>1):
        os.system('sudo python /home/pi/multiled.py')
else:
    break
```

CHAPTER-4

E-MAIL BASED CONTROL

4.1. Introduction

There are four methods of communication within the technology arena:

- Computer-to-computer
- Human-to-human
- Computer-to-human
- Human-to-computer

These are all important to us for different reasons. The first one allows devices to be controlled automatically according to some time- or logic-based programming. Human-to-human communications are those that take place every day but can now be facilitated by technology. Before the advent of the telephone, our only form of real-time communication was face-to-face. Now we have e-mail, Internet relay chat (IRC), instant messaging (IM), and SMS to perform the same task. All remove the “face” element. We have also streamlined our existing communication mediums.

Telephones, which were once low quality and hardwired to a physical location, are now mobile. Through Voice over IP (VoIP) technology, you can make use of the (near) free cost of the Internet to provide financial savings and, when combined with mobile technology, facilitate the amusing situation in which using a mobile phone is used to order pizza online through a web page!

When we talk of computer-to-human communication, we are looking at reports and information about the house that the computer sends to us, as appropriate. In the simplest of cases, this might be the daily wakeup call or an e-mail containing the day’s TV schedule. In more complex scenarios, it could be a full report of the computers in the house and how they are performing. And, finally, human-to-computer communication takes place most often and involves us telling the machine what we want to do via e-mail, SMS, or a web page.

To be a truly smart and automated house, this list would include haptic interfaces. We don’t need to issue an explicit command to tell the computer what to do; it knows what to do by studying the environment. For example, the computer would

know to switch on the lights when the front door has been opened; shortly afterward, the inside doormat sensor will close because it had realized that someone is entering the house.

4.2. E-Mail

E-mail is now the lifeblood of personal and professional life the world over. It is very easy to send and receive messages from anyone at any time—too easy, in fact, as the state of most spam folders will testify! But it is here to stay, so we can add e-mail to the list of protocols our house will support, allowing us to send messages to our video, light switches, or TV, and for our house to send messages back.

4.2.1. Preparing E-mail in Linux

The travel path of an e-mail is the same everywhere and consists of three parts:

- **Mail transfer agent (MTA):** The MTA is also known as the e-mail server and is the software that communicates with other MTAs over the Simple Mail Transfer Protocol (SMTP) to route the e-mail messages it receives to the correct recipient, noting the destination address and passing it to the server on that machine.
- **Mail retrieval:** This is the method by which e-mail is transferred from the mail server and onto the client. The transfer of this data occurs through either Post Office Protocol (POP) or Internet Message Access Protocol (IMAP). In our case, these will be on the same machine because we'll run our own MTA, but they needn't be as we could also download our Google Mail to our local machine for processing.
- **Mail user agent (MUA):** This is the client software used to actually read the e-mail as well as send it. This includes large GUI applications such as Thunderbird, web mail solutions such as AtMail, and smaller console-based ones such as Mutt.

Although corporate drones will bleat incessantly about the benefits of Exchange as an MTA you have four primary e-mail servers to choose from and many more MUAs than simply Outlook. Furthermore, because of the design of Linux (and Unix-like systems in general), you can automatically process incoming mail with great flexibility and issue noninteractive commands to send replies. Each MTA has benefits and features the others don't. The big four—Exim, qmail, Postfix, and Sendmail—each has its own advocates and detractors. We personally used Exim because it has a guided install and

“just worked” afterward. For alternate opinions, there is a wiki page covering the latest versions of these packages, along with some commercial offerings.

4.2.2. Sending E-mail

After installing the server and testing it by sending yourself (and a second user) an e-mail or two, you can begin the short task of writing an automatic send script. This is the easiest thing to do with Linux and involves the mail command, which sends e-mail with any number of additional headers and settings. Here, you need only an abstraction script such as the following:

```
#!/bin/bash

SUBJECT=$1; shift
TOADDR=$1; shift
MSG=$*

echo "$MSG" | mail -s "$SUBJECT" "$TOADDR"
```

which can be called with this:

```
xmitemail "Hello" "steev@workaddress.com" "I bet you didn't think this would work!"
```

This command will send the simplistic e-mail shown and can be either invoked by typing it on the command line triggering it from a daily crontab or run as a consequence of some other household event. For example, someone coming through the front could issue such as e-mail, or it could be sent as a warning when one of the hard disks get too full. However, there will be times when you want to revert to the original usage of mail by allowing the script to accept any input from STDIN. This requires the three-line replacement shown here to usurp MSG:

```
if [ $# -eq 0 ]; then
    while read LINE ; do
        MSG="$MSG""^M""$LINE"
    done
else
    MSG=$*
fi
```


Note the ^M character, which is entered into editors like vi with Ctrl+V followed by Ctrl+M. Now the message can now be fed in from a file, like this:

```
cat filename | xmitemail "Here's the file" steev@myworkaddress.com
```

4.2.3. Autoprocessing E-mails

Accepting e-mails on behalf of a program, instead of a human user, can be summed up in one word: Procmail. Procmail was a project begun in 1990 by Stephen R. van den Berg to control the delivery of e-mail messages. Procmail is triggered by the e-mail server (an MTA, such as Exim) by passing each message for further processing to each of a series of recipes. If none of these recipes lays claim to the message, it is delivered as normal. We'll begin by creating a simple example whereby you can e-mail your bedroom light switch. So, create a user with the following, and fill in all the necessary user details:

```
adduser bedroom
```

Then, create a .procmailrc file (note the dot!) in their home directory, and add the following recipe code:

```
:0
* ^From steev
* ^Subject: light on
|heyu turn bedroom_light on
```

This requires that the sender is steev and that the subject is “light on” before it runs the heyu command to control the light. Both conditions must be met. You can, and should, extend these arguments to include the full e-mail address (to prevent any steev from having control over the light) and perhaps a regular expression to make the subject line case insensitive.

Each recipe consists of three parts:

- **Mode:** This is generally :0 but can also include instructions for locking (so that the recipe cannot be run multiple times simultaneously) by appending another colon, with the name of a lock file (for example, :0:mylock).
- **Conditions:** Zero or more lines (beginning with an asterisk) indicating how the e-mail must appear for processing to occur. This also supports regular expressions. Because

every condition must be satisfied in an AND logical fashion, you can accept all mail by not including any condition lines.

- **Action:** The final line indicates whether the message should be forwarded to another e-mail account (with ! forwarded@othermail.com), passed to a script or program (| command arguments), or merely copied to a file (the name of the file, without prefix characters).

Each recipe is evaluated in order until it finds one that fulfills all conditions, at which point it stops. You can verify the input to Procmail by using the formail tool as part of the action in a catchall recipe:

```
:0
|formail >> ~steev/procmail-log
```

You can review this in real time by opening a separate terminal window, typing the following, and watching the mail messages appear:

```
tail -f ~steev/procmail-log
```

You can also use this technique when debugging Procmail-invoked scripts by taking a copy of a sent e-mail and redirecting it to the script's input. You can also debug Procmail scripts by using the LOGFILE directive. Here's an example:

```
LOGFILE=$HOME/procmail.logfile
```

The .procmailrc script itself also has some of the functionality of a standard bash script, so you can also prepare the PATH variables for the commands and preprocess the mail to extract the subject line, like this:

```
PATH=/usr/bin:/usr/local/bin:/usr/local/minerva/bin
SUBJECT=`formail -zxSubject:`
```

You could now create a separate recipe for switching the light off again, and it would be as simple as you'd expect. However, for improved flexibility, We'll show how to run a separate script that looks also at the body of the e-mail and processes the message as a whole so that you can include commands to dim or raise the light level. Begin by passing the subject as an argument⁶ and e-mail content (header and body) into STDIN, which is launched from a new recipe:

```
:0
* ^From - steev.*
* ^Subject: light
|~steev/lightcontrol $SUBJECT
```

You then use the `lightcontrol` script to concatenate the body into one long string, separated by spaces, instead of newlines:

```
#!/usr/bin/perl
# Skip the header, i.e. any non-empty line
while(<STDIN>) {
    last if /\s*$/;
}
my $body = "";
my $separator = "";
# Begin the message with the subject line, if it exists
if (defined $ARGV[0]) {
    $body = $ARGV[0];
    $separator = " ";
}
# Then concatenate all other lines
while(<STDIN>) {
    chomp;
    if ($_ !~ /\s*$/) {
        $body .= $separator;
        $body .= $_;
        $separator = " ";
    }
}
```

You can then process the `$body` to control the lights themselves, with either straight comparisons (meaning the text must include the command and only the command) or simple regular expressions to allow it to appear anywhere, as with the “dim” example.

```

if ($body eq "light on") {
    system("heyu turn e3 on");
} elseif ($body eq "light off") {
    system("heyu turn e3 off");
} elseif ($body =~ /light dim (\d+)/) {
    system("heyu dimb e3 $1");
}

```

With these simple rules, you can now create user accounts (and consequently e-mail addresses) for each of the rooms in your house and add scripts to control the lights, appliances, and teakettles, as you see fit. You can also use a house@ e-mail address to process more complex tasks, such as waiting for a message that reads “coming home” and then waiting one hour (or however long your commute is) before switching on the teakettle just ahead of time, as well as the porch and living room lights.

To stop Procmail from processing this mail and discarding it, you must “clone” the message before passing it to the recipe by adding a c to the first line. The following example demonstrates this by making a vocal announcement on receipt of such a mail and sending the original to the inbox:

```

:0c
* ^From- steev.*
|/usr/bin/play /media/voices/messages/youve-got-mail.wav

```

4.2.4. Security Issues

As a plain-text method of data transfer, e-mail is often likened to the sending of a postcard rather than a letter, because its contents (in theory) can be read by any delivery server en route. It is also a public protocol, allowing anyone in the world to send a message to your server. These two elements combined make it difficult to ensure that no one else is going to try to e-mail your light switches.

Some basic precautions here include the following:

- Nondisclosure of the e-mail address or format
- A strict command format (an e-mail signature will cause the parse to fail in most cases)
- No acknowledgment of correct, or incorrect, messages
- Restricting the sender (albeit primitively)

Again, we've adopted security through obscurity. But, even so, there is still the possibility for hackers to create mischief. If you are intending to use e-mail as a primary conduit, then it is worth the time and effort to secure it properly by installing GnuPG, generating certificates for all of your e-mail accounts, and validating the sender using their public keys. This does mean that new users cannot control the house without first having their key manually acknowledged by the system administrator.

The only time that this method breaks down is when you're unable to get to a registered e-mail account (when you're on vacation, for example) and you need to send a command from a temporary address. This is a rare case, however, and it is hoped that anything that serious would be dealt with through an SSH connection, or you'd have a suitable spare e-mail account configured for such an emergency.

For a quicker installation and one that works anywhere, you can have a cyclic list of passwords held on the server, and the e-mail must declare the first one on that list to be given access. Once you've been validated, the command is carried out, and the list cycles around, with the first element being pushed to the bottom:

```
tail -n +2 list >tempfile
head -n 1 list >>tempfile
mv tempfile list
```

In this way, anyone watching you type the e-mail or monitoring your traffic only gets access to an old password. Naturally, both methods can be combined.

4.3. Sending E-mails From Raspberry Pi

There are many cases when it can be very useful to be able to send emails from the Raspberry Pi to arbitrary recipients. This is not the same as having a real MTA running on the Pi (like Sendmail, Postfix, Exim, QMail, etc.) which can also receive and store emails.

In the following we are only going to cover the possibility of sending emails, not receiving. In most cases this is enough, as people tend to use Gmail, Yahoo! Mail and other major email service providers and they store their emails on the servers of these providers. Still, sending out emails from the Raspberry Pi can come in handy in many situations. For example, you could have some sensors connected to the GPIO pins of the Pi and you could program the Pi to send you an email when the temperature

in the room rises above or drops below certain threshold values, when a gas sensor registers unwanted gas leaks or when the measured voltage of a monitored battery becomes too low. You could also have your Pi send you daily or weekly emails with summarized system data. Or maybe you could connect a webcam to the Raspberry Pi and set up some motion detection software, which would send you an email as soon as it detects motion in a given area of your house.

In order to achieve this we are going to install a piece of software called SSMTP, which is a simple tool for sending emails. We are also going to configure PHP in a way which is going to make it possible to send emails from inside PHP scripts. This way it's going to be easy for web applications (like WordPress plugins) to send mails to chosen recipients. Many email servers today have very strict rules for accepting emails. For example if the email is not coming from a machine with a static IP address, they might classify the email as SPAM. We don't want that to happen with the emails sent from the Raspberry Pi, so we are going to send the emails to a Google server, which will send them forward to the real recipients. In order to be able to accomplish this, you must have a Gmail account.



Fig.4.1: Sending E-mails from Raspberry Pi

4.3.1. Installing and configuring SSMTP

In order to properly install and configure SSMTP, the steps mentioned below have to be followed.

1. Make sure your repositories are up-to-date:

```
apt-get update
```

2. Install SSMTP and mail utilities:

```
apt-get install ssmtp  
apt-get install mailutils
```

3. Edit the SSMTP configuration file:

```
nano /etc/ssmtp/ssmtp.conf
```

a) Mandatory lines:

```
root=postmaster  
mailhub=smtp.gmail.com:587  
hostname=raspberrypi  
AuthUser=YourGMailUserName@gmail.com  
AuthPass=YourGMailPassword  
UseSTARTTLS=YES
```

Be sure to specify the correct Gmail user name and password here, otherwise you will get authentication errors. If the host name of your Raspberry Pi is different from “raspberrypi”, specify your actual host name here instead.

b) Optional lines:

```
rewriteDomain=your.domain
```

Specify this if you would like the outgoing emails to appear to be sent from your.domain (instead of from gmail.com).

```
FromLineOverride=YES
```

Specify this if you would like SSMTP to leave the From field of the emails untouched. Otherwise it will overwrite the from field with the name of the Linux user which sends the email. The overwriting name is taken from the 5th value in the line that corresponds to the sending user, from the /etc/passwd file. If you plan to send emails from a website (for example from a WordPress plugin) and wish to have nice sender names like “John Doe”, I recommend commenting this line (which is equal to setting the value to NO), otherwise your website will only be able to send emails with less nice sender names, like johndoe@your.domain. In other words, you probably want SSMTP to overwrite the sender field with a nice name taken from the/etc/passwd file.

4. Edit the SSMTP aliases file:

```
nano /etc/ssmtp/revaliases
```

This file contains data about the email accounts for existing Linux users in the format:

```
local_account:outgoing_address:mailhub [:port]
```

You should create one line for all the users in your system from which you plan to be able to send emails. For example:

```
root:root@your.domain:smtp.gmail.com:587
```

```
www-data:yourwebpagesname@your.domain:smtp.gmail.com:587
```

In case you wish to send out emails from a WordPress plugin, you must make sure that you have a line for the user `www-data`, which is the user under which WordPress runs.

5. Set the permissions of the SSMTP configuration file:

```
chmod 774 /etc/ssmtp/ssmtp.conf
```

The permissions of the file `/etc/ssmtp/ssmtp.conf` determine who will be able to send emails from the Raspberry Pi. By default this file is owned by the user `root` and the group of the file is also `root`. So if you want other users, like `www-data` to be able to send emails (which you definitely want if you're using a WordPress plugin for example to send out emails), then you need to give read rights to the users who are not the owner of the file and not in the group of the file. The above permissions (774) mean that the owner (`root`) will be able to read/write/execute the file (7), the other users in the `root` group will be able to do the same (7) and the users which are not in the group will only have the right to read the file (4). For more details type `chmod --help`. If you prefer not to allow every user on your system to send emails, then add the `www-data` user (or the user who you would like to grant permission for sending emails) to the `root` group and only give the rights to the users in this group:

```
sudo usermod -a -G root www-data
```

```
chmod 770 /etc/ssmtp/ssmtp.conf
```

Be careful though. Adding the `www-data` user to the `root` group might sometimes not be very safe, as it will allow your website to do many things on your system.

6. Nice sender names:

If you would like your website (WordPress for example) to be able to send emails which appear to be sent from a person with a nice name like "John Doe" or

“Your super web site” instead of from a simple sender name like you@your.domain, then you need to make sure that in the /etc/ssmtp/ssmtp.conf file the FromLineOverride line is commented (#) or set to NO and you need to give a nice name to the www-data user. You can do this by editing the passwords file:

```
nano /etc/passwd
```

Find the line corresponding to www-data and set the fifth value in it to a nice name like “Your super website”.

4.3.2. Sending emails from command line

Once you’re done with the above setup and configuration process, you can send emails very easily from command line, which is great because you can also put the sending into Bash scripts, which can be called by other applications running on your Pi. A simple example looks like this:

```
echo “Test text” | mail -s “Test Mail” targetperson@example.com
```

The text “Test text” is sent to the email address targetperson@example.com (you may also specify multiple addresses separated by spaces) with the subject “Test Mail”.

4.3.3. Sending emails from PHP scripts

If you would also like to be able to send out emails from PHP scripts (which is the case if you plan to send emails from your website, perhaps from a WordPress plugin like Subscribe2), then you need to configure PHP to find the mail sending application.

1. Edit your PHP configuration file:

```
nano etc/php5/apache2/php.ini
```

Find (F6) the line which contains sendmail_path and set it to the appropriate value:

```
sendmail_path = /usr/sbin/sendmail -t -i
```

2. To test if PHP is indeed able to send out emails, create a file named mailtest.php and put the following code into it:

```
<?php
$to = “targetperson@example.com”;
$subject = “PHP Test mail”;
$message = “This is a test email”;
$from = “you@your.domain”;
$headers = “From:” . $from;
```

```
mail($to,$subject,$message,$headers);  
echo "Mail Sent.";  
?>
```

Test by calling your script from command line:

```
PHP mailtest.php
```

4.3.4. Troubleshooting

If you fail to configure things properly, you might run into some errors when trying to send emails. Some common error messages are the following:

1. Authentication failure:

```
Authorization failed (535 5.7.1 http://support.google.com/mail/bin/answ ...  
swer=14257 a1sm11494547eep.2 – gsmtip
```

The most probable cause of this is that you have specified a wrong username or password for your Gmail account in the `/etc/ssmtp/ssmtp.conf` file. Double check the values of the `AuthUser` and `AuthPass` fields.

2. Connection lost in the middle of processing:

```
send-mail: Connection lost in middle of processing
```

Chances are you specified the wrong port for the Google smtp somewhere in `/etc/ssmtp/ssmtp.conf` or in `/etc/ssmtp/revaliases`.

3. Your website (for example your WordPress plugin) gives an error message when you try to send an email from it or it appears to have sent the email but it never arrives to the recipient(s):

Check the last lines of your logs. There should be some relevant error messages in there:

```
tail /var/log/mail.log
```

```
tail /var/log/syslog
```

One of the most common problems is that you have not given the `www-data` user rights to read the `/etc/ssmtp/ssmtp.conf` file. Check the permissions of the file:

```
ls -l /etc/ssmtp/ssmtp.conf
```

and if something is not right, set the permissions, for example:

```
chmod 774 /etc/ssmtp/ssmtp.conf
```

4.4. Preparing Raspberry Pi to Send Mail through Gmail

Suppose you have set up your Raspberry Pi to do some stand-alone work. It would be nice if it could email you occasionally, for instance if there's something wrong. Or it may send you status updates on the work its doing. For this you could use the SMTP server of your ISP. However, if your Pi isn't stationary and roams around on multiple networks, this is not an ideal situation. If you connect your Pi to a different network, operated by a different ISP, chances are that you can't send any messages.

A better solution would be to use a Gmail account for that. We presume you already have a Gmail account. If not, you can get one for free at www.gmail.com. You may also decide to create a new one for your Raspberry Pi. The same account can even be shared among multiple Raspberry Pies. Rumour has it though that Gmail will only allow 100 mails sent per account per day.

If you share your account among too many machines you may start hitting the ceiling soon. Because your account's password will be readable by the root user on your Raspberry Pi, it is better to use the setup of two step authentication in your Gmail account. That way you can create an application specific password for your Raspberry Pi. With such a password it's not possible to access your account settings, and it can easily be discarded should it get compromised, without jeopardizing the rest of your email account.

4.4.1. Getting a Message Transfer Agent

Before your Pi can send emails it needs a message transfer agent (MTA). I have to flavours for you. The first one is `exim4`, the other one is `SSMTP`. The choice is yours. Needless to say, you will only need one of the two MTAs. Thanks to the Debian packaging system this is easily done by typing one of the following commands.

4.4.1.1. Exim4

```
sudo apt-get install exim4
```

After installing `exim4` we need to configure it. This is done by the following command:

```
sudo dpkg-reconfigure exim4-config
```

Some questions are asked for which answers are provided. The following steps talk about the various requirements needed for the configuration of `exim4`.

- The first screen asks you what type of mail server you need. Select the second option: "mail sent by smarthost; received via SMTP or fetchmail"
- The next question asks for the system mail name: Set to same as hostname (raspberrypi)
- Now it asks you what IP addresses should be allowed to use the server. Leave as is (127.0.0.1 ; ::1)
- Other destinations for which mail is accepted: raspberrypi
- Machines to relay mail for: Leave blank.
- IP address or host name of outgoing smarthost: Enter: smtp.gmail.com::587
- Hide local mail name in outgoing mail: Select: No
- Keep number of DNS-queries minimal: Select: No
- Delivery method for local mail: Select: "Maildir format in home directory"
- Split configuration into small files: Select: No

After answering all these questions exim4 will restart and we're halfway home. Now you'll have to enter your account details. As root, edit the file /etc/exim4/passwd.client and add the next three lines at the end of the file.

```
gmail-smtp.l.google.com:YOU@gmail.com:PASSWORD
*.google.com:YOU@gmail.com:PASSWORD
smtp.gmail.com:YOU@gmail.com:PASSWORD
```

Needless to say that you'll have to change YOU to your Gmail login name, and PASSWORD to your password on all three lines. After that you only have to update and restart exim and you're done! The next two lines will do that for you:

```
sudo update-exim4.conf
sudo /etc/init.d/exim4 restart
```

4.4.1.2. SSMTP

```
sudo apt-get install ssmtp mailutils mpack
```

Now edit the file /etc/ssmtp/ssmtp.conf as root and add the next lines. Please note that some of the lines already exist and may need to be changed. Others don't exist yet and need to be added to the end of the file.

```
mailhub=smtp.gmail.com:587
hostname=ENTER YOUR RPI'S HOST NAME HERE
AuthUser=YOU@gmail.com
AuthPass=PASSWORD
useSTARTTLS=YES
```

Again you'll have to replace YOU with your Gmail login name and PASSWORD with your (application specific) Gmail password. After this you're done. You don't even have to restart the SSMTP server (in fact, there is none).

4.4.2. The Final Touches

Some processes, for instance crontabs, can send mails to root or other system users. If you don't want to miss any of them you can setup some aliases. You can do that by editing the file /etc/aliases. Here's what it looks like:

```
# /etc/aliases
mailer-daemon: postmaster
postmaster: root
nobody: root
hostmaster: root
usenet: root
news: root
webmaster: root
www: root
ftp: root
abuse: root
noc: root
security: root
root: pi
pi: youremail@example.com
```

This tells the system to redirect all mail to root, while mail to root is redirected to the user pi, while mail to user pi is finally redirected to my own mail account. This way all mail will eventually be sent to my own mail account, no matter to what local user the mail was originally sent. You will probably have more than one Raspberry Pi

mailing you some status information. And unless you use different mail accounts for all your RPi's it becomes harder and harder to find out which one is mailing you. The next command will setup a new full name (pi @ domotics) for the user name pi, with which you can identify the source of the e-mail.

```
sudo chfn -f "pi @ domotics" pi
```

4.4.3. Testing

To test your outgoing mail simply execute the next command:

```
mail -s "This is the subject line" root@localhost
```

Then type the body of the message, this is only a test so anything will do. When you're done typing the body type a dot (.) at the beginning of a new line and hit Enter. The mail should now be sent to root, which is redirected to pi, which is redirected to your normal email address. You can also send a mail directly to your own email address if you want to, however that would have skipped the redirections from the test. Below you see two other ways of sending mail from your RPi. Both methods will send the file body.txt as message body.

```
mail -s "This is the subject line" someone@example.com < body.txt  
cat body.txt | mail -s "This is the subject line" someone@example.com
```

4.5. Extracting Emails From Gmail With Python via IMAP

A filter setup in a Gmail account to automatically collect what are essentially automatically generated emails from a particular source is present, and they can be filed neatly into a label, leaving the inbox relatively uncluttered with their consistently formatted regularness. The nice thing about Python is that there's a module for just about everything. The not-so-nice thing about Python is that there's usually more than one module for everything. And often a bewildering number of packages offering the same functionality.

There may be more than one way to programmatically get emails out of Gmail with python, one option is IMAP. Gmail can be accessed via IMAP, and conveniently enough the Python Standard Library has an IMAP interface, so it appears not unreasonable to use IMAP. The imaplib implements a client for communicating with Internet Message Access Protocol (IMAP) version 4 servers. The IMAP protocol

defines a set of commands sent to the server and the responses delivered back to the client. Most of the commands are available as methods of the IMAP4 object used to communicate with the server.

First step is to create an IMAP4 instance, preferably the SSL variant for security, connected to the Gmail server at `imap.gmail.com`:

```
#!/usr/bin/env python
import sys
import imaplib
import getpass
import email
import datetime
M = imaplib.IMAP4_SSL('imap.gmail.com')
```

Next we can attempt to login. If the login fails, an exception of type `imaplib.IMAP4.error`: will be raised:

```
try:
    M.login('notatallawhistleblowerIswear@gmail.com', getpass.getpass())
except imaplib.IMAP4.error:
    print "LOGIN FAILED!!! "
    # ... exit or deal with failure...
```

If the login is successful, we can now do IMAPy things with our IMAP4 object. Most methods of IMAP4 return a tuple where the first element is the return status of the operation (usually 'OK' for success), and the second element will be either a string or tuple with data from the operation. For example, to get a list of mailboxes on the server, we can call `list()`:

```
rv, mailboxes = M.list()
if rv == 'OK':
    print "Mailboxes:"
    print mailboxes
```

With Gmail, this will return a list of labels. To open one of the mailboxes/labels, call `select()`:

```
rv, data = M.select("Top Secret/PRISM Documents")
```

```

if rv == 'OK':
    print "Processing mailbox...\n"
    process_mailbox(M) # ... do something with emails, see below ...
    M.close()
M.logout()

```

So with the mailbox selected, we can now get the emails within it. For example, we can get all the emails in the selected mailbox and for each one output the message number, subject, and date:

```

# Note: This function definition needs to be placed
#       before the previous block of code that calls it.
def process_mailbox(M):
    rv, data = M.search(None, "ALL")
    if rv != 'OK':
        print "No messages found!"
        return
    for num in data[0].split():
        rv, data = M.fetch(num, '(RFC822)')
        if rv != 'OK':
            print "ERROR getting message", num
            return
        msg = email.message_from_string(data[0][1])
        print 'Message %s: %s' % (num, msg['Subject'])
        print 'Raw Date:', msg['Date']
        date_tuple = email.utils.parsedate_tz(msg['Date'])
        if date_tuple:
            local_date = datetime.datetime.fromtimestamp(
                email.utils.mktime_tz(date_tuple))
            print "Local Date:", \
                local_date.strftime("%a, %d %b %Y %H:%M:%S")

```

We use the `search()` method to get a list of message sequence numbers, then loop over these, calling `fetch()` to get the actual messages. `fetch()` returns the raw

message contents. To avoid having to parse the actual message data from `fetch()` ourselves, we can use the email package from the standard library. Once again, there are a few different packages floating around for doing this kind of thing, but I think email is currently the one least likely to get you down-voted on Stack Overflow. `message_from_string()` returns a message object, and we can then access header items as a dictionary on that object.

If you don't care about the date/time the emails were sent, then things are much simpler. But if you do care about such matters, note that the contents of the "Date" header may vary depending on the email client sending the email, and the timezone of the sender. The code snippet shows one possible way of converting to local time, using the capabilities of `email.util`. The message body can be obtained by calling `msg.get_payload()`, which will return the payload data as a string (if the message is not multi-part). For text messages, you could then parse the data using regular expressions. For parsing contents of HTML emails however, you must not use regular expressions.

4.6. System Configuration

Raspberry Pi has been chosen as the processing unit for the system because of its user friendly features and economical benefits. Further, python coded algorithm has been fed into the Raspberry Pi and is connected to the internet through Modulator Demodulator (MODEM) interface to access and send e-mails to the consumer. The Devices to be controlled have been interfaced with Raspberry Pi using relay driver circuit due to different power ratings of devices and Raspberry Pi. A display (optional) may also be connected to view the instantaneous status and processing of Raspberry Pi.

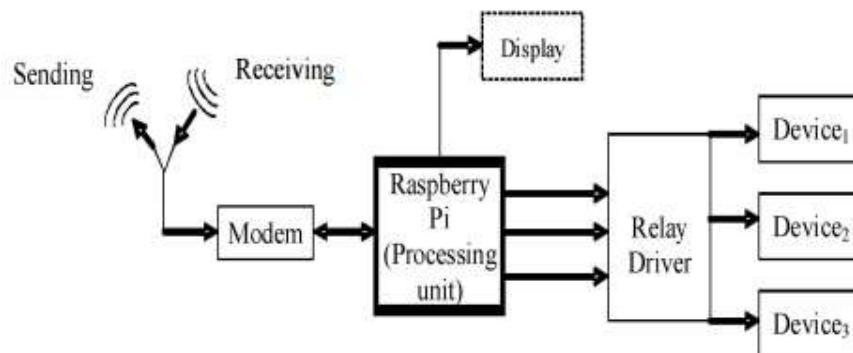


Fig. 4.2: Layout of the proposed system configuration

The GPIO pins for input and output have been defined to control different devices. The Raspberry Pi board has GPIO pin layout. Out of the 26 pins, 3 pins have been used to control three devices in this project which have been represented by 3 LEDs for testing the switching signal. For practical purposes a relay driver circuit and relays can be interfaced with Raspberry Pi and appliances, respectively, for their controlling.

The pins used in this project were: pin7 (GPIO4), pin11 (GPIO17) and pin12 (GPIO18). The code for implementing the control strategy for home automation was written in python environment on Raspberry Pi. Firstly, the code was set to initialize and log in into home g-mail account (gmail- imap) using the e-mail library of pythonIDE.

After successful initialization, Raspberry Pi starts reading the subjects of e-mails from the account specified in the code. The subject of these e-mails is then compared from the initializing commands of the interfaced devices and the control signal is generated according to it on the corresponding GPIO pin. This process is repeated continuously at an interval of 0.5 seconds.

Here, the subject read from the e-mail is stored in an array x [], and the if structure was defined as per the elements of that array, i.e. if subject is 'ONI', Raspberry Pi replies 'Turning On switch I' to the sender and simultaneously the switch at pin.7 is turned ON and the structure is looped for checking new mail after every 0.5 seconds.

4.6.1. Control structure for the proposed algorithm coded in pythonIDE

The control structure for the proposed algorithm coded in PythonIDE is given below:

```
if(len(x)<>0):
    GPIO.setmode (GPIO.BOARD)
    #signal to devices
    if(x [0] == 'ONI'):
        Reply ('Turning ON switch 1', y [0])
        GPIO.setup (7, GPIO.OUT)
        GPIO.output (7, GPIO.HIGH)           #Turn ON LED1
```

```

if(x[O] == 'ON2'):
    Reply ('Turning ON switch 2', y [O])
    GPIO.setup (II, GPIO.OUT)
    GPIO.output (II, GPIO.HIGH)          #Turn ON LED2
if(x[O] == 'ON3'):
    Reply ('Turning ON switch 3', y [O])
    GPIO.setup (12, GPIO.OUT)
    GPIO.output (12, GPIO.HIGH)          #Turn ON LED3
if(x[O] == 'OFF1 '):
    Reply ('Turning OFF switch 1', y [O])
    GPIO.setup (7, GPIO.OUT)
    GPIO.output (7, GPIO.LOW)            #Turn OFF LED1
if(x[O] == 'OFF2'):
    Reply ('Turning OFF switch 2', y [O])
    GPIO.setup (II, GPIO.OUT)
    GPIO.output (II, GPIO.LOW)           #Turn OFF LED2
if(x[O] == 'OFF3'):
    Reply ('Turning OFF switch 3', y [O])
    GPIO.setup (12, GPIO.OUT)
    GPIO.output (12, GPIO.LOW)           #Turn OFF LED3
time.sleep(0.5)                          #call delay

```

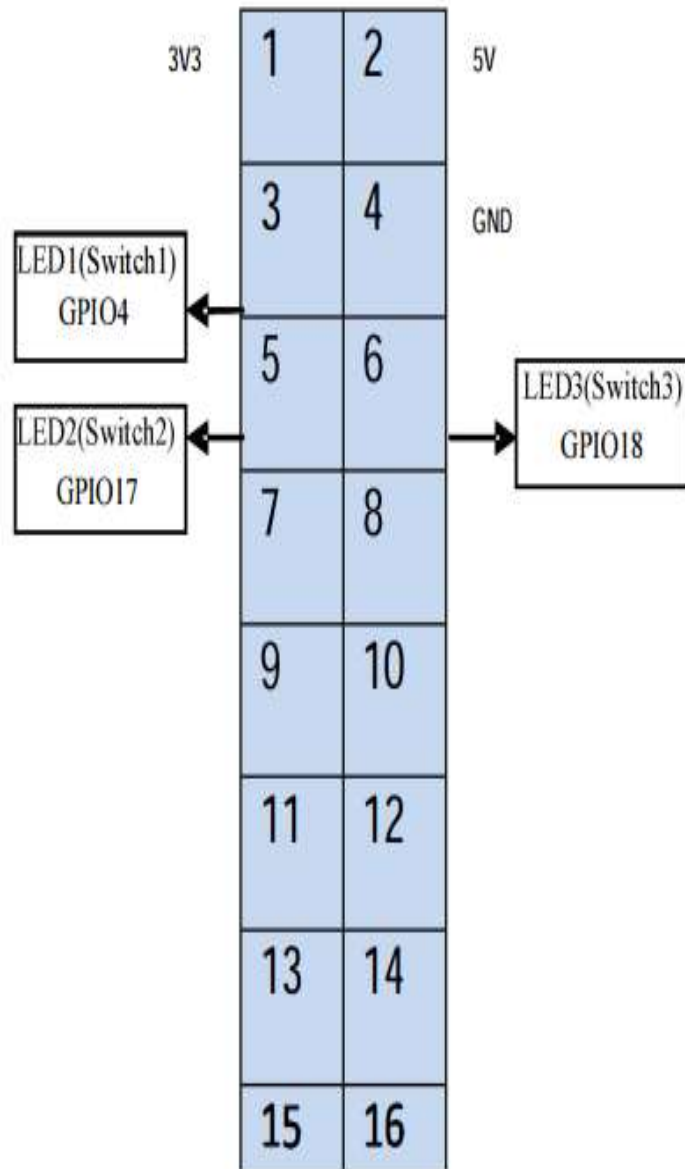


Fig.4.3: Pin Layout of Raspberry Pi GPIO used in the system

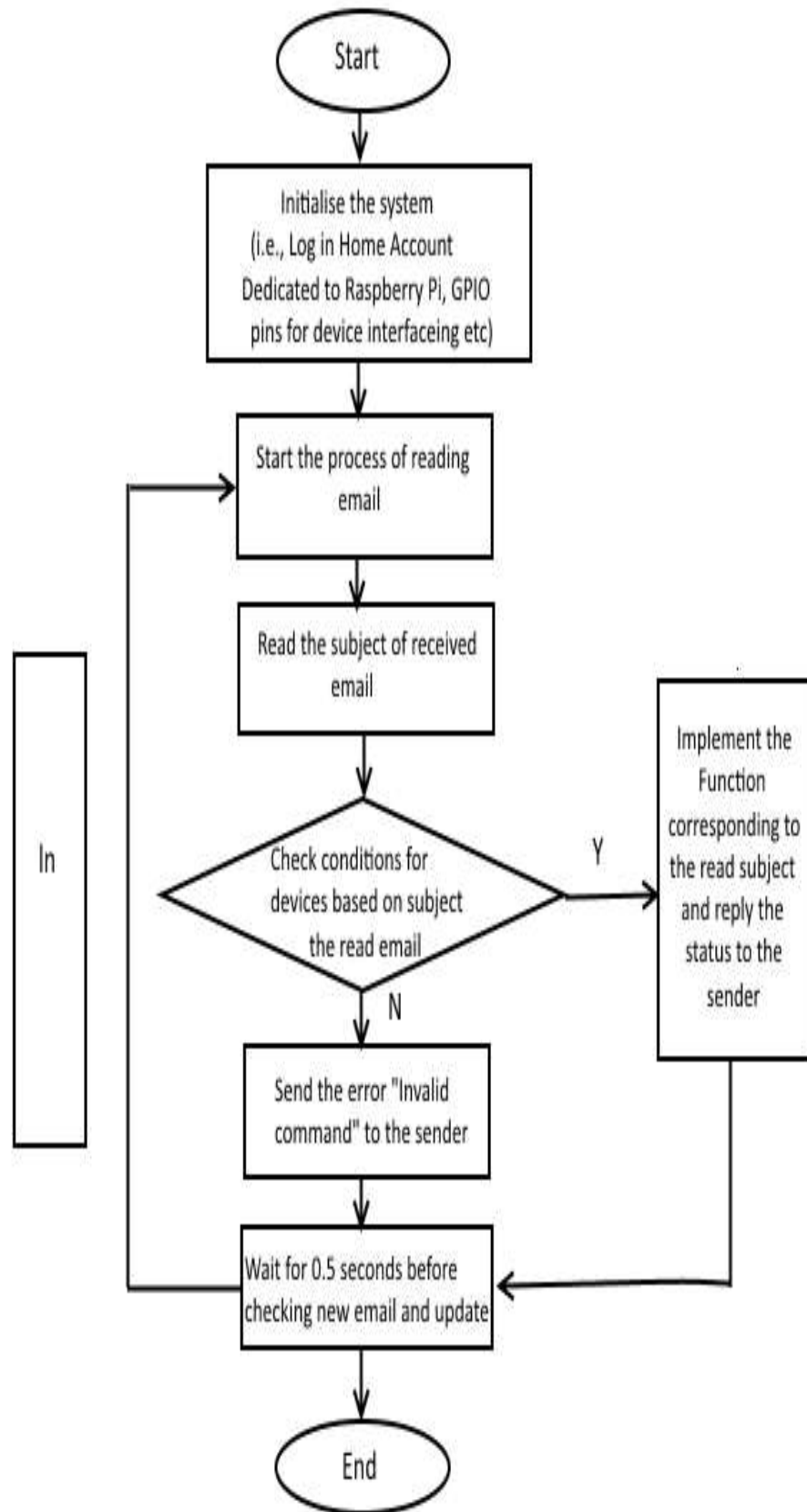
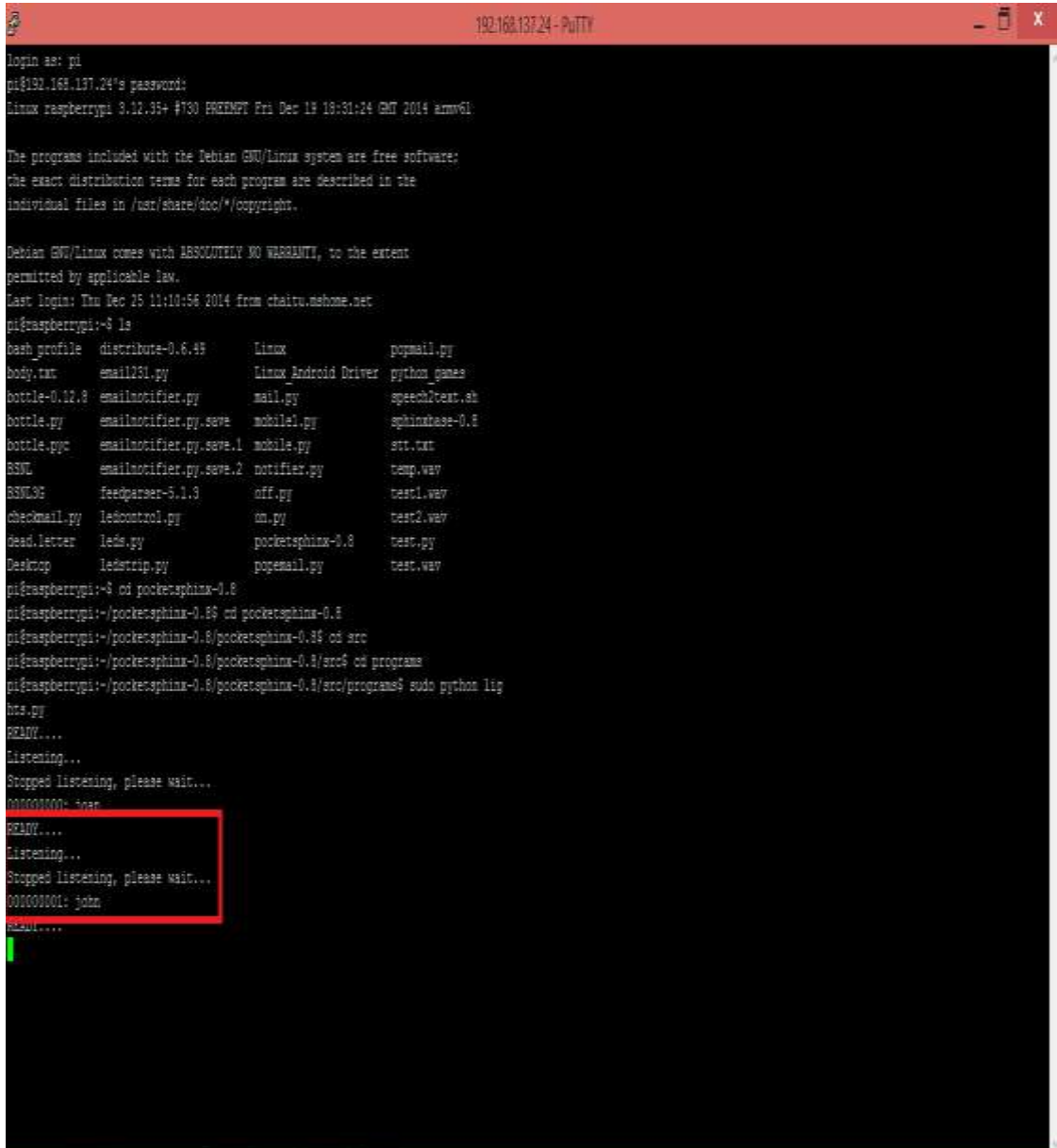


Fig.4.4: Flowchart of the control algorithm used

CHAPTER-5

PROJECT RESULTS AFTER EXECUTION OF CODE

5.1. Snapshots of Project Results



```
login as: pi
pi@192.168.137.24's password:
Linux raspberrypi 3.12.35+ #730 PREEMPT Fri Dec 19 19:31:24 GMT 2014 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: Thu Dec 18 11:10:58 2014 from chairu.mshome.net
pi@raspberrypi:~$ ls
hash_profile  distribute-0.6.49  Linux          popmail.py
body.txt      email251.py        Linux_Audio_Driver  python_games
bottle-0.12.8 emailnotifier.py    mail.py          speech/text.sh
bottle.py     emailnotifier.py.save  mobile1.py       sphinxbase-0.8
bottle.pyo    emailnotifier.py.save.1  mobile.py        sut.txt
BSNL         emailnotifier.py.save.2  notifier.py       temp.wav
BSNL3G       feedparser-5.1.3      off.py           test1.wav
checkmail.py ledcontrol.py        tm.py            test2.wav
dead.letter  leds.py              pocketsphinx-0.8  test.py
Desktop      ledstrip.py          popmail.py        test.wav
pi@raspberrypi:~$ cd pocketsphinx-0.8
pi@raspberrypi:~/pocketsphinx-0.8$ cd pocketsphinx-0.8
pi@raspberrypi:~/pocketsphinx-0.8/pocketsphinx-0.8$ cd src
pi@raspberrypi:~/pocketsphinx-0.8/pocketsphinx-0.8/src$ cd programs
pi@raspberrypi:~/pocketsphinx-0.8/pocketsphinx-0.8/src/programs$ sudo python lig
hrs.py
REMY....
Listening...
Stopped listening, please wait...
00000000: 3var
REMY....
Listening...
Stopped listening, please wait...
00000000: john
REMY....
```

Fig.5.1: Execution of On Mode



Fig.5.2: LED working in On Mode

```

/home/pi/on.py:4: RuntimeWarning: This channel is already in use, continuing anyway. Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(7,GPIO.OUT)
READY....
Listening...
Stopped listening, please wait...
00000001: i see it
READY....
Listening...
Stopped listening, please wait...
00000002: to talk the it
READY....
Listening...
Stopped listening, please wait...
00000003: up
READY....
Listening...
Stopped listening, please wait...
00000004: he
READY....
Listening...
Stopped listening, please wait...
00000005: yeah
READY....
Listening...
Stopped listening, please wait...
00000006: both
READY....
Listening...
Stopped listening, please wait...
00000007: wolf
READY....
Listening...
Stopped listening, please wait...
00000008: golf
READY....
Listening...
Stopped listening, please wait...
00000009: a!a
READY....
Listening...
Stopped listening, please wait...
00000010: off
/home/pi/off.py:4: RuntimeWarning: This channel is already in use, continuing anyway. Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(7,GPIO.OUT)
READY....

```

Fig.5.3: Execution of Off Mode



Fig.5.4: LED working in Off Mode

```

192.168.132.24 - PuTTY
Linux raspberrypi 3.12.35+ #790 PREEMPT Fri Dec 19 18:31:24 GMT 2014 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: Thu Dec 25 08:26:44 2014 from chairu.mahima.net
pi@raspberrypi:~$ ls
bash_profile  emailnotifier.py.save  nail.py  sphinbase-0.8
body.txt     emailnotifier.py.save.1  nocifier.py  str.txt
build        emailnotifier.py.save.2  off.py      temp.wav
checkmail.py email.py               on.py       test1.wav
dead.letter  email.py               pocketphinx-0.8  test2.wav
Desktop      feedparser-5.1.3       popmail.py   test.wav
distribute-0.6.49  leds.py               popmail.py
email1231.py  ledstrip.py           python_games
emailnotifier.py  Linux Android Server  speech2text.sh
pi@raspberrypi:~$ cd pocketphinx-0.8
pi@raspberrypi:~/pocketphinx-0.8$ cd pocketphinx-0.8
pi@raspberrypi:~/pocketphinx-0.8/pocketphinx-0.8$ cd src
pi@raspberrypi:~/pocketphinx-0.8/pocketphinx-0.8/src$ cd programs
pi@raspberrypi:~/pocketphinx-0.8/pocketphinx-0.8/src/programs$ ls
batch.c  lights.py  mdef_convert.c  pocketphinx_mdef_convert
batch.o  Makefile  mdef_convert.o
continuous.c  Makefile.am  pocketphinx_batch
continuous.o  Makefile.in  pocketphinx_continuous
pi@raspberrypi:~/pocketphinx-0.8/pocketphinx-0.8/src/programs$ sudo python lig
m3.py
READY....
Listening...
Stopped listening, please wait...
READY....
Listening...
Stopped listening, please wait...
000000001: hi
/home/pi/leds.py:4: RuntimeWarning: This channel is already in use, continuing anyway. Use GPIO.setwarnings(False) to disable warnings.
  GPIO.setup(7,GPIO.OUT)
READY....
Listening...
Stopped listening, please wait...
000000002: but
READY....

```

Fig.5.5: Execution of Blink Mode



Fig.5.6: LED working in Blink Mode

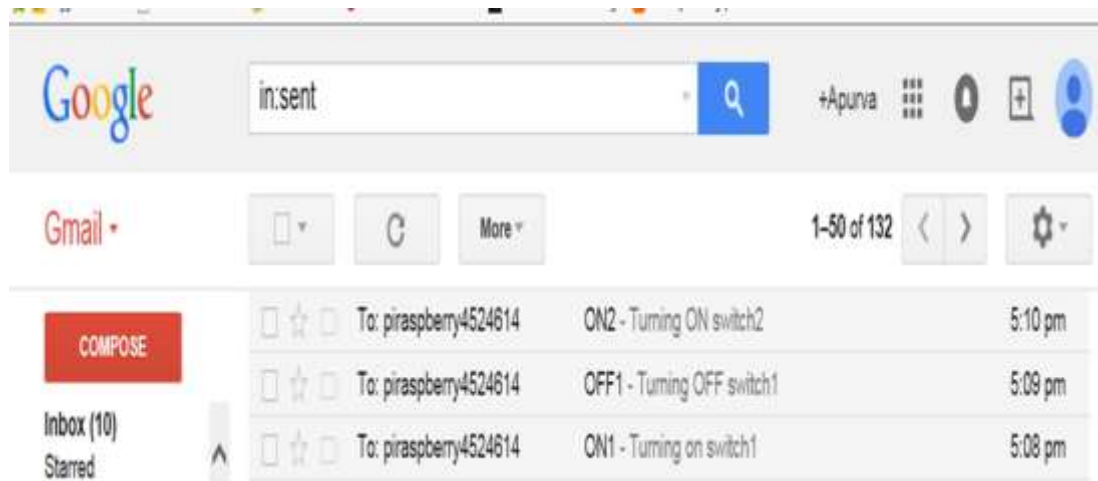


Fig.5.7: Email sent to Raspberry Pi to control switches

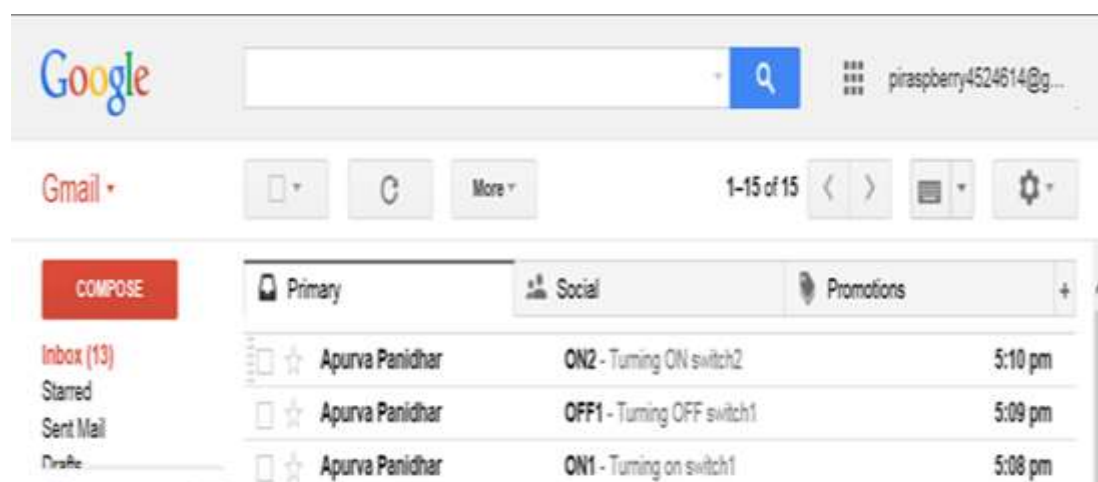


Fig.5.8: Email received by Raspberry Pi to control switches

5.2. Conclusion and Future Scope

The Raspberry Pi is a capable machine, able to support many maker-oriented projects in the home automation arena. The provision of an easy-to-program GPIO port makes this into a super powered Arduino that is capable of running several services at once, which means that it can be easily expanded with new features and update much more easily than other, similar, devices. Being so young in its development cycle, and such a charming device, has ensured that a number of specific modules are already available for its expansion that will hopefully ensure a long shelf life and a lot of new projects.

With so many ways of communicating into and out of a system, you must begin with a solid framework. One method is to separate the input systems from the processing, allowing any input mechanism (mobile phone, e-mails, or web interface) to generate a command in a known common format that can be processed by a single script. In a similar way, all messages are sent to a single script that then formats the message in a particular format, suitable for the given communication channel. You can also add an automatic process on receipt of any, or all, of these messages. So, once you have code to control a video, light switch, or alarm clock, you can process them in any order to either e-mail your video, SMS your light switch, speak to your alarm clock, or any combination of these.

In this highly developing era, where directly or indirectly, everything is dependent on computation and information technology, Raspberry Pi proves to be a smart, economic and efficient platform for implementing the home automation. This paper provides a basic application of home automation using Raspberry Pi which can be easily implemented and used efficiently. The code provided is generic and flexible in a user friendly manner and can be extended for any future applications like power control, surveillance etc., easily. Moreover, this technique is better than other home automation methods in several ways. For example, in home automation through DTMF, the call tariff is a huge disadvantage, which is not the case in proposed method. Also, in Web server based home automation, the design of web server and the space required is eliminated by this method, because it simply uses the already existing webserver provided by G-mail.

REFERENCES

- [1]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, pp. 277-280, 2014.
- [2]<http://cmusphinx.sourceforge.net/wiki/>
- [3]<http://www.raspberrypi.org/use-your-desktop-or-laptop-screen-and-keyboard-with-your-pi/>
- [4]<https://pihw.wordpress.com/guides/direct-network-connection/>
- [5]<http://raspberrypi.stackexchange.com/questions/26517/how-to-make-pocketsphinx-use-alsa-instead-of-pulseaudio-or-jack>
- [6]<http://raspberrypi.stackexchange.com/questions/26535/how-should-i-install-drivers-for-my-wireless-adaptor-on-raspi-b-moderaspbian>
- [7]<http://www.rmnd.net/speech-recognition-on-raspberry-pi-with-sphinx-racket-and-arduino/>
- [8]<http://diyhacking.com/best-voice-recognition-software-for-raspberry-pi/>
- [9]<https://sites.google.com/site/observing/Home/speech-recognition-with-the-raspberry-pi>
- [10]<http://pymotw.com/2/imaplib/>
- [11]<http://www.voidnullness.net/blog/2013/07/25/gmail-email-with-python-via-imap/>
- [12]<http://stackoverflow.com/questions/2230037/how-to-fetch-an-email-body-using-imaplib-in-python>
- [13]<http://stackoverflow.com/questions/1225586/checking-email-with-python>
- [14]<http://stackoverflow.com/questions/8669202/get-emails-with-python-and-pop-lib>
- [15]<http://www.codemiles.com/python-tutorials/reading-email-in-python-t10271.html>
- [16]<http://programming.mesexamples.com/python/network-python/python-how-to-read-email-messages-from-pop3-accounts/>
- [17]http://en.wikipedia.org/wiki/CMU_Sphinx