Abstract:

Security and Safety are the most trending and challenging things now a days. We need safety for the things which we love in order to reduce damage and loss. With the increase in the fire accidents-gas leakages –it is very much necessary to design and implement an efficient safety system. Due to the less efficiency and high cost of the already present safety systems-we are providing a economical and low cost system than others. This system's core part is Raspberrypi. In this project we are using multiple sensors for multipurpose. We are also providing Security for the home through finger print module. We are receiving the information from the different sensors which we completely update in a website and in an android Application. Suspected activities and Illegal authorizations are conveyed to remote user through Alert on Android Application.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The 'Thing' in IoT can be any device with any kind of definite sensors with the potential to congregate and move data over a framework without manual interposition. The embedded development in the thing makes them take part with inside states and the outside condition, which along these lines helps in decisions making process. Pretty much, IoT is a thought that relates all the devices to the web and let them talk with each other over the web. IoT is a mammoth arrangement of related devices – all of which collect and sends the data about their uses and the circumstances in which they worked.

Fig:1.1 Internet Of Things

Internet innovation interfacing gadgets, machines and apparatuses to the web by methods for remote innovations. Over 9 billion 'Things' associated with the Internet, starting at now. 'Things' associated with the Internet are anticipated to cross 20 billion sooner rather than Unification of advances, for example, low-power installed frameworks, distributed computing, large information, AI, and organizing.

Communication: The central intension of IOT is to give a correspondence among the physical devices, systems and people. Each and every space needs the exchanging of information in a solitary way or the other. For example, the restorative space the information about the patients, from time to time the essential information must be sent, so a snappy move could be made. The fundamental information as either circulatory strain or the beat rate could be evaluated with the help of sensors. On the off chance that there ought to be an event of transport region a vehicle can be followed, which requires the enabling of the region of the device. In all of these cases the correspondence expects a fundamental part.

Control and Automation:

In the related world, the business and the customer have a choice to control the devices, either clearly or remotely. For example, a client can utilize IoT to open their auto or start the garments washer. Moreover, IOT can be used to check the improvement of people in a particular domain. It should be conceivable by masterminding a sensor which can recognize the advancement and this should be conceivable remotely i.e., subsequently, by sitting in some other spot.

Cost: IOT is refreshing for mechanizing the things and this would diminish the cost of the general project. With new sensor information, IOT can empower an association to save money by constraining equipment dissatisfaction and empowering the business to perform masterminded bolster. Sensors can similarly evaluate the driving behavior, lifestyle parameters

CHAPTER 2

LITERATURE SURVEY

2.1. Vikram.N, Harish K.S, Nihaal M.S,RakshaUmesh4, Shetty Aashik Ashok Kumar, 'A Low Cost Home Automation System Using Wi-Fi Based Wireless Sensor Network Incorporating Internetof Things(IoT)' 2017 IEEE 7th International Advance Computing Conference.

This paper mainly discusses about home automation using WSN.With the rapid increase in usage and reliance on the vivid features of smart devices, the need for interconnecting them is genuine. Many existing systems have ventured into the sphere of Home Automation but have apparently failed to provide cost-effective solutions for the same. This paper illustrates a methodology to provide a low cost Home Automation System (HAS) using Wireless Fidelity (Wi-Fi). This crystallizes the concept of internetworking of smart devices. A Wi-Fi based Wireless Sensor Network(WSN) is designed for the purpose of monitoring and controlling environmental, safety and electrical parameters of a smart interconnected home. The user can exercise seamless control over the devices in a smart home via the Android application based Graphical User Interface (GUI) on a smartphone. The overall cost of large scale implementation of this system is about INR 6000 or USD 100.

2.2. Praveen Kumar, Umesh Chandra Pati ' IoT Based Monitoring and Control of Appliances for Smart Home' IEEE International Conference On Recent Trends In Electronics Information Communication Technology, May 20-21, 2016,India.

This paper mainly discusses about monitoring and controlling of each and every appliances in and around the house. The recent technology in home automation provides security, safety and comfortable life at home. That is why in the competitive environment and fast world, home automation technology is required for every person. This purposed home automation technology provides smart monitoring and control of the home appliances as well as door permission system for interaction between the visitor and home/office owner. The control and monitoring the status (ON/OFF of the appliances) have been implemented using multiple ways such as The Internet, electrical switch, and Graphical User Interface (GUI) interface. The system has low-cost design, user-friendly interface, and easy installation in home or multi-purpose building. Using this technology, the consumer can reduce the wastage of electrical power by regular monitoring of home appliances or the proper ON/OFF scheduling of the devices.

2.3. Y. W. Prakash, Vishakha Biradar, Shenil Vincent, Minto Martin and Anita Jadhav"Smart Bluetooth Low Energy Security system" IEEE WiSPNET 2017 conference.

This paper mainly discusses about providing security to the home using smart Bluetooth low energy device.The need for security in today's world has become a mandatory issue to look after. With the increase in a number of thefts, it has become a necessity to implement a smart security system. Due to the high cost of the existing smart security systems which use conventional Bluetooth and other wireless technologies and their relatively high energy consumption, implementing a security system with low energy consumption at a low cost has become the need of the hour. The objective of the paper is to build a cost effective and low energy consumption security system using the Bluetooth Low Energy (BLE) technology. This system will help the user to monitor and manage the security of the house even when the user is outside the house with the help of webpage. This paper presents the design and implementation of a security system using PSoC 4 BLE which can automatically lock and unlock the door when the user in the vicinity and leaving the vicinity of the door respectively by establishing a wireless connection between the physical lock and the smartphone. The system also captures an image of a person arriving at the house and transmits it wirelessly to a webpage. The system also notifies the user of any intrusion by sending a message and the image of the intruder to the webpage. The user can also access the door remotely on the go from the website.

2.4. Eleni Isa, Nicolas Sklavos ‘Smart Home Automation: GSM Security System Design &Implementation’ ResearchGate Conference Paper · January2017.

This paper mainly discusses about providing security using GSM as the core part.it also provides Home automation.Smart home automation has attracted the interest of the research community during the last decade, at a great manner. Home security systems consist a constantly, year after year, developing research field. Some of these systems are limited to support basic operations, while some others satisfy a range of additional primitives. In this paper, a security system for smart home automation is proposed. The introduced system operation is supported by a GSM embedded mobile module, which enables the alert messages transmission to both mobile devices of end users, and central security offices. The proposed system is implemented on a microcontroller module, through an embedded platform. System’s operation is also based on cameras and sensors inputs. The proposed system operates on different levels of user’s access control, based on passwords policies. Each time, the involved end users and the security offices, can be informed for attacks, operation modes changes etc, through SMS communication, via the available GSM network.

2.5. Vaibhav Sharma, Chirag Fatnani , Pranjal katara,Vishnu shankar , ‘AdvancedLow-cost Security System Using Sensors, Arduino and GSM Communication Module’Proceedings of IEEE TechSym 2014 Satellite Conference, VIT University, 7th- 8th March.

This paper mainly discusses about providing low cost security system using arduino.Home security is essential for occupants’ convenience and protection. This paper aims to develop a low-cost means of home security system using temperature, passive infrared and proximity sensors. Data from all these sensors is continually received and processed by arduino Uno board which act as a microcontroller unit. In case of untoward situations, the arduino will trigger an alarm and alert messages will be sent to user’s mobile via GSM. The temperature sensor LM35 is used to prevent fire accidents by detecting the increase in temperature beyond a certain limit. The low-power Passive Infrared (PIR) detectors take advantage of pyro-electricity to detect a human body that is a constant source of infrared radiation while proximity sensor uses Hall Effect principle to detect any intrusion through doors and windows. Thus the system ensures home safety as well as security.

Raspberry pi is preferred over others due to these reasons:-

1. From the above literature survey we finally concluded that Raspberry pi has more advantages over all others
2. It has High processing Speed
3. It has many interfaces like HDMI, multiple USB, Ethernet, onboard Wi-Fi and Bluetooth, many GPIOs, USB powered, etc
4. It easily supports real time applications.

CHAPTER 3
SOFTWARE DESCRIPTION

3.1 Introducing Python

Python was developed in the late 1980s at the National Research Institute by Guido van Rossum as a successor to the ABC language. Python is a high-level language which is flexible as it is written in normal English, providing the Pi with commands in a manner that is quick to learn and easy to follow. Python makes use of clear syntax which is a valuable tool for anyone who wants to learn to program. It is also the language that suits best to the Raspberry Pi Foundation for those looking to progress from the simple scratch.

3.2 Example: Hello World

As we know the easiest way to learn a new programming language is to create a project that prints “Hello World!” on the screen. In Python instructions file can be created by using any text editor. For example, if you like working at the console or in a terminal window, you can use nano or if you prefer a graphical user interface (GUI), you can use Leaf pad. Another way is to use an integrated development environment (IDE) which provides specific functionality that’s missing from a standard text editor, including syntax checking, debugging facilities and the ability to run your program without leaving the editor. To begin the Hello World project, open IDLE from the Programming menu in the desktop environment. We can also create a blank document in our favorite text editor and skip the rest of this paragraph if we don’t like to use IDLE. By default, anything you type in the initial window will be immediately executed in IDLE that opens up in python shell mode. To open a new

Python project, click on the File menu and choose New Window to open a blank file.

It’s good practice to start all Python programs with #and! Characters at the beginning of the line. The operating system looks for python files with the help of this line. Although this is not entirely necessary for programs that will be run from IDLE or explicitly at the terminal, it is required for programs that run directly by calling the program’s file name. The first line of your program should read as follows:

```
#!/usr/bin/env python
```

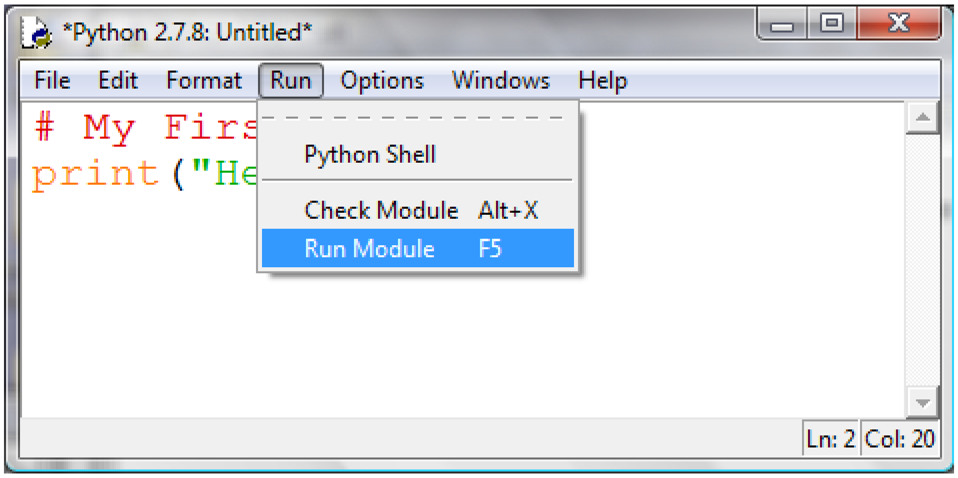
This line tells the operating system to look at the $PATH environment variable—where Linux stores the location of files .The $PATH variable contains a list of directories where executable files are stored, and is used to search programs when you type their name at the console or in a terminal window. To print out a message to the console or terminal window, you should use print command. Any text following the word print and placed between quotation marks will be printed to the standard output device.

```
print “Hello, World!”
```

The final program should look like this:

```
#!/usr/bin/env python
print “Hello, World!”
```

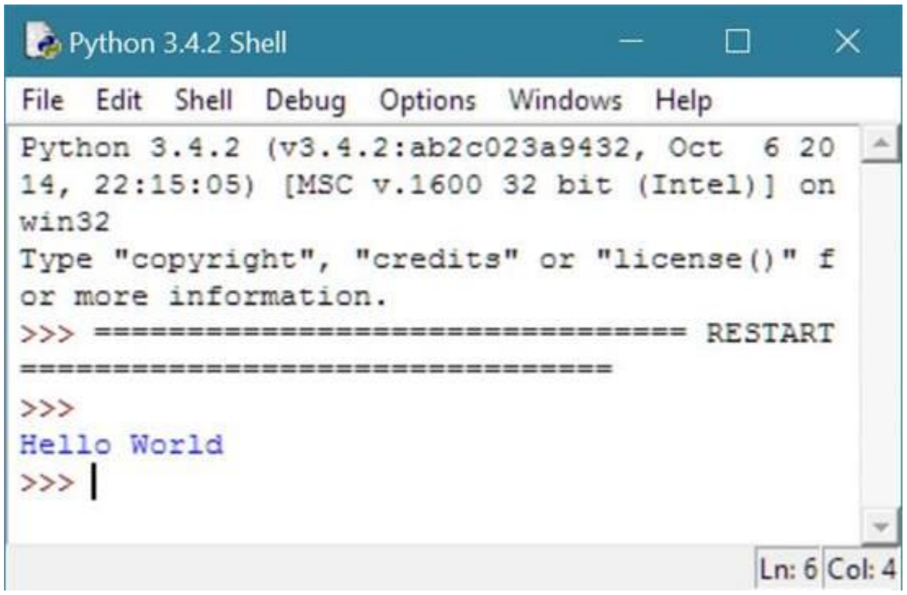
Before running program, save it as helloworld.py using the File menu. The file will be given the extension .py when you save it.



```
# My Firs
print("He
```

Fig: 3.2.1 EXAMPLE HELLO WORLD

Click on Run Module from the Run menu, or press the F5 key on the keyboard. Message Hello, World! appear on screen in blue . If not, check for syntax errors using syntax highlighting feature.



```
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 20
14, 22:15:05) [MSC v.1600 32 bit (Intel)] on
win32
Type "copyright", "credits" or "license()" f
or more information.
>>> ================================ RESTART
================================
>>>
Hello World
>>> |
```

Fig: 3.2.2 OUTPUT OF EXAMPLE

CHAPTER 4

HARDWARE

4.1 RASPBERRY PI

Raspberry Pi board is a small scale wonder, pressing significant registering power into an impression no bigger than a charge card. It's able to do some surprising things, however there are a couple of things you're going to need to know before you dive straight into the brier fix.

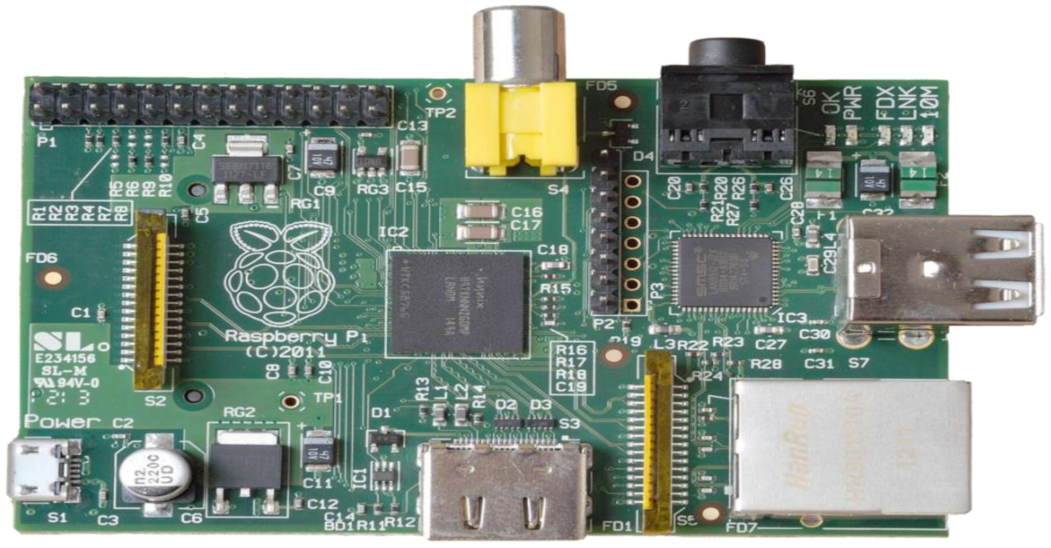


Fig: 4.1.1 RASPBERRYPI

4.1.1 Beginning with the Raspberry Pi

Since you have an essential comprehension of how the Pi varies from other processing gadgets, it's an ideal opportunity to begin. On the off chance that you've just got your Pi, remove it from its defensive enemy of static sack and spot it on a level, non-conductive surface before continuing with this section.

4.1.2 Interfacing a Showcase

Before you can begin utilizing your Raspberry Pi, you're going to need to associate a presentation. The Pi bolsters three distinctive video outputs: composite video, HDMI video and DSI video. Composite video and HDMI video are promptly available to the end user, as portrayed in this segment, while DSI video requires some particular equipment.

4.1.3 Composite Video

Composite video, accessible by means of the yellow-and-silver port at the highest point of the Pi known as a RCA phono connector (see Figure 1-2), is intended for associating the Raspberry Pi to more seasoned presentation gadgets. As the name proposes, the connector makes a composite of the hues found inside a picture—red, green and blue—and sends it down a solitary wire to the presentation device, typically an old cathode-beam tube (CRT) TV. Figure 1-2: The yellow RCA photo connector, for composite video output When no other showcase gadget is accessible, a composite video association will kick you off with the Pi. The quality, however, isn't incredible. Composite video associations are altogether progressively inclined to impedance, need clearness and keep running at a restricted resolution, meaning that you can fit less symbols and lines of content on the screen on the double.

4.1.4 HDMI Video

A superior quality picture can be gotten utilizing the HDMI (Top quality Interactive media Interface) connector, the main port found on the base of the Pi (see Figure 1-3). In contrast to the simple composite association, the HDMI port gives a high-speed digital association with pixel-ideal pictures on both PC screens and top notch Televisions. Utilizing the HDMI port, a Pi can show pictures at the Full HD 1920x1080 goals of most current HDT If you're planning to utilize the Pi with a current PC screen, you may find that your showcase doesn't have a HDMI input. That's not a debacle: the advanced signs present on the HDMI link guide to a typical PC screen standard called DVI(Digital Video Interconnect). By buying a HDMI-to-DVI link, you'll have the capacity to interface the Pi's HDMI port to a monitor with DVI-D connectivity. Figure 1-3: The silver HDMI connector, for top quality video output If your screen has a VGA input—a D-molded connector with 15 pins, commonly hued silver and blue—the Raspberry Pi can't associate with it. Connectors are accessible that will take in an advanced DVI flag and convert it to a simple VGA flag, but these are costly and massive. The best alternative here is just to purchase a progressively current screen with a DVI or HDMI input.

4.1.5 DSI Video

The final video output on the Pi can be establish above the SD card slot on the top of the written circuit panel—it's a small size ribbon connector sheltered by a layer of plastic. This is for a video standard known as Display Serial Interface (DSI), which is used in the flat-panel displays of tablets and smart phones. Displays with a DSI connector are rarely available for retail purchase, and are typically held in reserve for engineers looking to create a compact, independent system. A DSI display can be connected by inserting a ribbon cable into the matched connector on the Pi, but for beginners, the use of a composite or HDMI display is suggested.

4.1.6 Connecting Audio

If you're using the Raspberry Pi's HDMI port, audio is simple: when well configured, the HDMI port carries both the digital audio signal and a video signal. This earnings that you can connect a single cable to your display device to enjoy both sound and pictures. Pretentious you're connecting the Pi to a standard HDMI display, there is very little to do at this point. For now, it's enough to plainly connect the cable. If you're using the Pi with a DVI-D monitor via an adapter or cable, audio will not be incorporated. This highlights the main difference between HDMI and DVI, while HDMI can bring audio signals, DVI can't. For those with DVI-D monitors, or those using the composite video output, a black 3.5mm audio jack located on the top border of the Pi next to the yellow plug connector provides analogue audio (see Figure 1-2). This is the same connector used for headset and microphones on consumer audio equipment, and it's wired in exactly the same way. If you want, you can simply attach a pair of earphones to this port for quick access to audio.

While headphones can be connected directly to the Raspberry Pi, you may find the volume a little absent. If possible, connect a pair of powered speakers as an alternative. The amplifier inside will help improve the signal to a more audible level. If you're looking for something more unending, you can either use standard PC speakers that have a 3.5mm connector or you can buy some adapter cables. For composite video users, a 3.5mm to RCA plug cable is useful. This provides the two white and red RCA plug connections that sit alongside the video connection, each moving a channel of the stereo audio signal to the TV. For those connecting the Pi to an amplifier or CD player system, you'll either need a 3.5 mm to RCA plug cable or a 3.5 mm to3.5 mm cable, depending on what additional connections you have on your system. Both cable types are eagerly and economically available at consumer electronics shops, or can be purchase even cheaper at online retailers such as Amazon.

4.1.7 Connecting a Keyboard and Mouse

Now that you have got your Raspberry Pi's output devices sorted, it's time to think about put in. As a uncovered minimum, you're going to need a keyboard, and for the majority of users, a mouse or trackball is a need too. First, some bad news: if you have got a keyboard and mouse with a PS/2 connector—a round plug with a horseshoe-shaped collection of pins—then you are going to have to go out and buy a substitute. The old PS/2 connection has been outdated, and the Pi expects your peripherals to be connected over the Universal Serial Bus (USB) port. Depending on whether you purchased the Model A or Model B, you'll have moreover one or two USB ports available on the right side of the Pi (see Figure 1-4). If you're using Model B, you can connect the keyboard and mouse openly to these ports. If you're using Model A, you'll need to purchase a USB focal point in order to connect two USB devices simultaneously. Figure 1-4: Model B's two USB ports. A USB hub is a good venture for any Pi user: even if you've got a Model B, you'll use up both your accessible ports just connecting your keyboard and mouse, parting nothing free for extra devices such as an external optical drive, storage device or joystick. Make sure you buy a powered USB focal point: passive models are cheaper and slighter, but lack the ability to run current hungry devices like CD drives and external hard drives.

If you want to decrease the number of power sockets in use, connect the Raspberry Pi's USB power direct to your powered USB hub. This way, the Pi can draw its power directly from the hub, slightly than needing its own dedicated power socket and mains adapter. This will only work on hubs with a power supply accomplished of providing 700mA to the Pi's USB port, along with doesn't matter what power is required by other peripherals. Connecting the keyboard and mouse is as easy as plugging them in to the USB ports, either openly in the case of a Model Bore via a USB hub in the case of a Model A.

A Note on Storage As you've most likely noticed, the Raspberry Pi doesn't have a conventional hard drive. Instead it uses a Secure Digital (SD) memory card, a solid-state storage space system typically used in digital cameras. Almost any SD card will work with the Raspberry Pi, but because it holds the entire operating system, it is essential for the card to be at least 2 GB in capacity to store all the necessary files.

SD cards with the operating system preloaded are presented from the official Raspberry Pi Store along with several other sites on the Internet. If you've purchased one of these, or received it in a bunch with your Pi, you can simply plug it in to the SD card slot on the bottom side of the left-hand edge. If not, you'll need to install an operating system—known as blinking—onto the card before it's all set to go.

4.1.8 Connecting External Storage

Where as the Raspberry Pi uses an SD card for its main storage device—known as a boot device—you may come across that you run into space restrictions quite quickly. Although large SD cards holding 32GB, 64GB or more are available, they are often prohibitively exclusive. Thankfully, there are devices that supply an additional hard drive to any computer when connected via a USB cable. Known as USB Mass Storage (UMS) devices, these can be physical hard drives, solid-state drives (SSDs) or even convenient small flash drives.

Two USB Mass Storage devices: a pen drive and an exterior hard drive. The majority of USB Mass Storage devices can be read by the Pi, whether or not they have existing content. In order for the Pi to be able to access these devices, their drives must be mount—a process you will learn in Chapter 2, "Linux System management". For now, it's sufficient to connect the drives to the Pi in willingness.

4.1.9 Connecting the Network

While the bulk of these setup commands are equally valid to both the Raspberry Pi Model A and the Model B, networking is a special exemption. To keep the component count—and therefore the cost—as low as potential, the Model A doesn't feature any onboard networking. Fortunately, that doesn't mean you can't network the Model A; only that you will want some additional equipment to do so. Networking the Model

A To give the Model A the same networking capabilities as its added expensive Model B counterpart, you'll need a USB-connected Ethernet adapter. This connects to a free USB port on the Raspberry Pi or a connected hub and provides a wired Ethernet connection with an RJ45 connector, the similar as is accessible on the Model B.

Raspberry Pi vs Arduino Uno:

	Arduino Uno	Raspberry Pi Model B
Price	$30	$35
Size	7.6 x 1.9 x 6.4 cm	8.6cm x 5.4cm x 1.7cm
Memory	0.002MB	512MB
Clock Speed	16 MHz	700 MHz
On Board Network	None	10/100 wired Ethernet RJ45
Multitasking	No	Yes
Input voltage	7 to 12 V	5 V
Flash	32KB	SD Card (2 to 16G)
USB	One, input only	Two, peripherals OK
Operating System	None	Linux distributions
Integrated Development Environment	Arduino	Scratch, IDLE, anything with Linux support

Table.4.1.9.1 RASPBERRYPI VS ARDUINO UNO

4.2 ARM vs. x86

The processor at the core of the Raspberry Pi framework is a Broadcom BCM2835 framework on-chip (SoC) mixed media processor. This implies by far most of the framework's parts, including its focal and designs handling units alongside the sound and interchanges equipment, are fabricated onto that solitary segment covered up underneath the 256 MB memory chip at the focal point of the board (see Figure 1-1). It's not simply this SoC structure that makes the BCM2835 diverse to the processor found in your work area or PC, nonetheless. It additionally utilizes an alternate guidance set design (ISA), known as ARM.

The BCM2835 SoC, situated underneath a Hynix memory chip Developed by Oak seed PCs back in the late 1980s, the ARM engineering is a generally unprecedented sight in the desktop world. Where it exceeds expectations, be that as it may, is in cell phones: the telephone in your pocket in all likelihood has no less than one ARM-based processing center covered up away inside. Its mix of a basic diminished guidance set (RISC) engineering and low power draw settle on it the ideal decision over work area chips with high power requests and complex guidance set (CISC) architectures. The ARM-based BCM2835 is the mystery of how the Raspberry Pi can work on simply the 5V 1A power supply provided by the installed smaller scale USB port. It's additionally the motivation behind why you won't discover any warmth sinks on the gadget: the chip's low power draw straightforwardly converts into next to no waste warmth, not with standing amid muddled handling tasks. It does, in any case, imply that the Raspberry Pi isn't good with customary PC programming. Most of programming for desktops and workstations is worked in view of the x86 guidance set engineering, as found in processors from any semblance of AMD, Intel and By means of. Subsequently, it won't keep running on the ARM-based Raspberry Pi. The BCM2835uses an age of ARM's processor configuration known as ARM11, which thus is structured around a form of the guidance set design known as ARMv6. This merits recalling: ARMv6 is a lightweight and amazing architecture, but has an opponent in the further developed ARMv7 engineering utilized by the ARM Cortex group of processors. Programming developed for ARMv7, similar to programming produced for x86, is tragically not good with the Raspberry Pi's BCM2835—in spite of the fact that developer scan generally convert the product to make it suitable. That's not to say you will be limited in your decisions. As you'll find later in the book, there is a lot of software available for the ARMv6 guidance set, and as the Raspberry Pi's fame keeps on developing, that will just increment. In this book, you'll additionally figure out how to make your very own product for the Pi regardless of whether you have no involvement with programming.

4.2.1 Windows vs. Linux

Another vital contrast between the Raspberry Pi and your work area or PC, other than the size and cost, is the operating framework—the product that enables you to control the computer. The dominant part of work area and PCs today run one of two working frameworks: Microsoft Windows or Apple OSX. The two stages are shut source, made in a hidden domain utilizing exclusive techniques. These working frameworks are known as shut hotspot for the idea of their source code, the coding languages formula that tells the framework what to do. In shut source programming, this formula is kept a firmly monitored mystery. Clients can acquire the finished programming, however never to perceive how it's made.

The Raspberry Pi, paradoxically, is intended to run a working framework called GNU/Linux—in the future eluded to just as Linux. Unlike Windows or OS X, Linux is open source: it's conceivable to download the source code for the whole working framework and make whatever transforms you want. Nothing is covered up, and all progressions are made in full perspective on people in general. This open source development ethos has enabled Linux to be immediately adjusted to keep running on the Raspberry Pi, a procedure known as porting. At the time of this composition, a few adaptations of Linux—known as appropriations—have been ported to the Raspberry Pi's BCM2835 chip, including Debian, Fedora Remix and Curve Linux. The distinctive conveyances take into account diverse requirements, however they all share something for all intents and purpose: they're all open source. They're also all, all things considered, good with one another: product composed on a Debi an framework will work consummately well on Curve Linux and vice versa. Linux isn't selective to the Raspberry Pi. Many diverse conveyances are accessible for work areas, workstations and even mobile devices; and Google's well known Android stage is created over a Linux center. On the off chance that you find that you appreciate the experience of utilizing Linux on the Raspberry Pi, you could consider adding it to other registering gadgets you use too. It will happily coexist with your current working framework, enabling you to appreciate the advantages of both while giving you a natural environment when your Pi is inaccessible.

Likewise with the distinction among ARM and x86, there's a key point to have about the useful effect between Windows, OS X and Linux: programming composed for Windows or OS X won't keep running on Linux. Fortunately, there are a lot of compatible alternatives for the larger part of normal programming items—even better, the lion's share are allowed to utilize and as open source as the working framework itself.

4.3 Wired Networking

To obtain your Raspberry Pi on the network, you will need to attach an RJ45 Ethernet patch cable among the Pi and a switch, router or hub. If you don’t have a router or hub, you can obtain your desktop or laptop talking to the Pi by connecting the two straight together with a patch cable. Typically, connecting two network clients mutually in this way requires a special cable, known as a crossover cable. In a crossover cable, the receive and transmit pairs are swapped so that the two devices are prohibited from talking over each other—a task frequently handled by a network switch or hub. The Raspberry Pi is cleverer than that, still. The RJ45 port on the side of the Pi includes a feature known as auto-

MDI, which allows it to reconfigure itself by design. As a result, you can use any RJ45 cable—crossover or not—to connect the Pi to the network, and it will change its configuration as a result.

The Raspberry Pi Model B's Ethernet port If you do connect the Pi openly to a PC or laptop, you won't be able to connect out onto the Internet by default. To do so, you'll require to configure your PC to bridge the wired Ethernet port and another (typically wireless) connection. Doing so is outside the scope of this book, but if you are completely not capable to connect the Pi to the Internet in any other way, you can try searching your operating system's help file for "bridge network" to find more assistance. With a cable associated, the Pi will automatically receive the details it needs to access the Internet when it loads its operating system through the Dynamic Host Configuration Protocol (DHCP). This assigns the Pi an Internet Protocol (IP) address on your network, and tells it the gateway it needs to use to access the Internet (typically the IP address of your router or modem).For some networks, there is no DHCP server like that to provide the Pi with an IP address. When related to such a network, the Pi will need physical configuration. You'll study more about this in section 4, "Network design".

A 10/100 USB Ethernet adapters—with the figures referring to its two-speed mode, 10 Mb/s and 100 Mb/s—can be purchased from online retailers for very little funds. When selling an Ethernet adapter, be sure to verify that Linux is listed as a supported operating system. A few models only work with Microsoft Windows, and are unsuited with the Raspberry Pi. Don't be tempt to go for a gigabit-class adapter, which will be referred to as a 10/100/1000 USB Ethernet adapter. Standard USB ports, as used on the Raspberry Pi, cannot cope with the speed of a gigabit Ethernet connection, and you'll see no advantage to the more exclusive adapter.

4.4 Wireless Networking

Existing Raspberry Pi models don't feature any form of wireless network facility onboard, but—as with adding wired Ethernet to the Model A—it's achievable to add Wi-Fi to maintain any Pi using a USB wireless adapter. Two USB wireless adapters, suitable for use with the Raspberry Pi Using such a device, the Pi can connect to a broad range of wireless networks, including those running on the latest 802.11n high speed standard. Before purchasing a USB wireless adapter, verify the following:

• The Linux is listed as a supported operating system. Some wireless adapters are provided with drivers for Windows and OSX only, making them mismatched with the Raspberry Pi.

• Make sure that your Wi-Fi network type is supported by the USB wireless adapter. The network type will be listed in the disclaimer as a number followed by a letter. If your network type is 802.11a, for example, an 802.11g wireless adapter won't work.

• Check the frequencies supported by the card. Some wireless network standards, like 802.11a, support more than one frequency. If a USB wireless adapter is designed to work on a 2.4GHz network, it won't connect to a 5GHz network.

• Verify the encryption type used by your wireless network. Most modern USB wireless adapters bear all forms of encryption, but if you are buying a second-hand or older model, you may find it won't connect to your network. familiar encryption types include the outdated WEP and more modern WPA and WPA2.Configuration of the wireless connection is done within Linux, so intended for now it's sufficient to just attach the adapter to the Pi(ideally through a powered USB hub.) You'll learn how to organize the connection in Chapter 4, "Network arrangement".

4.5 Connecting Power

The Raspberry Pi is powered by the minute micro-USB connector obtained on the lower left side of the circuit board. This connector is similar to as found on the majority of smart phones and some tablet devices. Many chargers designed for smart phones will work with the Raspberry Pi, but not all. The Pi is extra power-hungry than most micro-USB devices, and requires up to 700mA in order to operate. Some chargers can only supply up to 500mA, cause irregular problems in the Pi's operation Connecting the Pi to the USB port on a desktop or laptop computer is possible, but not optional. While with smaller chargers, the USB ports on a computer can't provide the power required for the Pi to work properly.

Only connect the micro-USB power supply when you are complete to start using the Pi. With no power button on the device, it will start working the immediate power is connected and can only be turned off again by physically removing the power cable.

4.6 RASPBERRY PI COMPUTE MODULE

4.6.1 GPIO

BCM283x has in total 54 GPIO lines in 3 types of voltage banks. All GPIO pins have at least two different functions within the SoC. When we does not used for the alternate peripheral function, each GPIO pin may be set as an input pin (optionally as an interrupt) or an output pin. The alternating functions are usually peripheral I/Os, and most peripherals appear twice to allow flexibility on the choice of I/O voltage. On CM1, CM 3 and CM 3L bank 2 is used on the module to connect to the MMC device and, on CM 3and CM 3L, for an on-board I2C. On CM 3L mainly of bank 2 is exposed to allow a user to connect their choice of SD card or MMC device (if required).

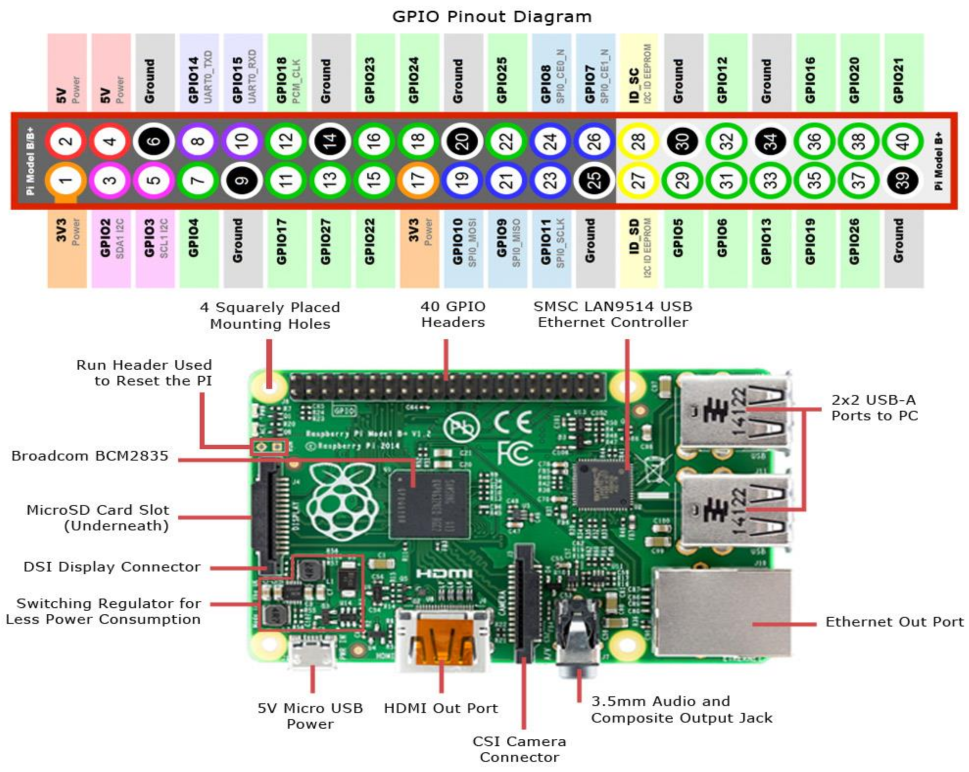


Fig: 4.7.1.1PIN CONFIGURATION

4.6.2 Secondary Memory Interface (SMI)

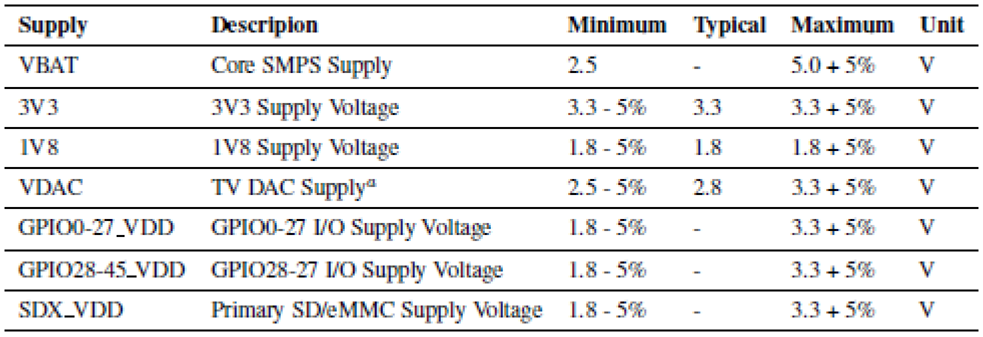
The SMI peripheral is an asynchronous NAND type bus supporting Intel mode80 type transfers at 8 or16 bit widths and available in the ALT1 positions on GPIO banks 0 and 1. It is not publicly documented in the Broadcom Peripherals Specification but a Linux driver is available in the Raspberry Pi Linux repository (bcm2835 smi.c in linux/drivers/misc).

4.6.3 Power Supplies

The Compute Module has six supplies that must be powered at all times even if a specific interface or GPIO bank is unused. They are:

1. VBAT to power the BCM283x processor core and to feed the SMPS that generates the chip core voltage.
2. 3V3 powers various BCM283x PHYs, IO.
3. 1V8 powers SDRAM.
4. VDAC to power the composite (TV-out) DAC.
5. GPIO(0-27) VREF powers the GPIO (0-27)IO bank.

6. GPIO (28-45) VREF is used to power the GPIO (28-45) IO bank



Supply	Descripion	Minimum	Typical	Maximum	Unit
VBAT	Core SMPS Supply	2.5	-	5.0 + 5%	V
3V3	3V3 Supply Voltage	3.3 - 5%	3.3	3.3 + 5%	V
1V8	1V8 Supply Voltage	1.8 - 5%	1.8	1.8 + 5%	V
VDAC	TV DAC Supply[a]	2.5 - 5%	2.8	3.3 + 5%	V
GPIO0-27_VDD	GPIO0-27 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
GPIO28-45_VDD	GPIO28-27 I/O Supply Voltage	1.8 - 5%	-	3.3 + 5%	V
SDX_VDD	Primary SD/eMMC Supply Voltage	1.8 - 5%	-	3.3 + 5%	V

Table.4.6.3.1 POWER SUPPLES

4.6.4 Display Parallel Interface (DPI)

A pattern parallel RGB (DPI) interface is available on bank 0 GPIO's. This is upto-24-bit parallel interface can support a secondary display. Again this interface is not documented in the Broadcom Peripherals Specification but documentation can be found here and depth expansion, laneways have become blind zones

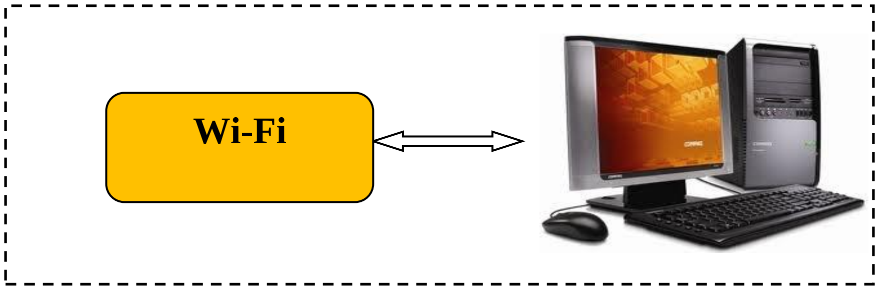


Fig.4.7.4.1 DPI

4.7 Gas Sensor

This unit can be easily included into an alarm unit, to sound an alarm or give a visual indication of the LPG concentration. The sensor has excellent sensitivity joint with a quick response time. The sensor can also sense icon-butane, propane, LNG and cigarette smoke.

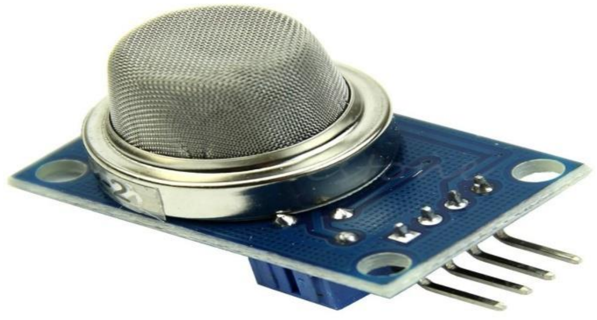


Fig.4.8.1 GAS SENSOR

4.7.1 Applications

- Gas leak detection system
- Fire/Safety detection system
- Gas leak alarm
- Gas detector

4.7.2 Features

- It has high sensitivity compared to wired networking
- The Detection Range is from 100 - 10,000
- The Response Time is fast that is less than 10s
- The Heater Voltage range is 5V
- It has the Dimensions are: 18mm, 17mm High excluding pins, Pins - 6mm High

4.8 Rain sensor:

A **rain sensor** or *rain switch* is a switching device activated by rainfall. There are two main applications for rain sensors. The first is a water conservation device connected to an automatic irrigation system that causes the system to shut down in the event of rainfall. The second is a device used to protect the interior of an automobile from rain and to support the automatic mode of windscreen wipers. An additional application in professional satellite communications antennas is to trigger a rain blower on the aperture of the antenna feed, to remove water droplets from the Mylar cover that keeps pressurized and dry air inside the wave-guides.

The rain sensor works on the principle of total internal reflection. ... An infrared light beams at a 45-degree angle on a clear area of the windshield from the sensor inside the car. When it rains, the wet glass causes the light to scatter and lesser amount of light gets reflected back to the sensor.

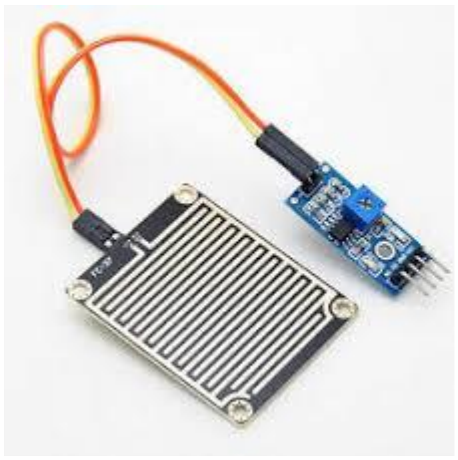


Fig.4.9.1 RAIN SENSOR

4.9 Fire sensor:

The majority fire detection technology focus on the aspects of detecting heat, smoke (particle matter) or flame (light) – the three major

characteristics of fire. All of these characteristics also have begin sources other than fire, such as heat from steam pipes, particle matter from aerosols, and light from the sun. Other factors further confuse the process of fire detection by masking the characteristic of interest, such as air temperature, and air movement.
In addition to that, smoke and heat are from fires can disperse too rapidly or accumulate too slowly for effective detection. In contrast, because flame detectors are optical devices, they can react to flames in less than a second. This optical feature also limits the flame detector as not all fires have a burn. As through any type of detection method its use must equivalent the environment and the risk within the environment.

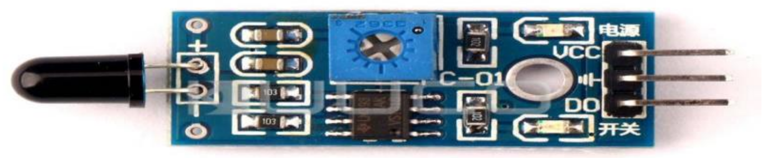


Fig.4.10.1 FIRE SENSOR

4.10 Humidity sensor:

Humidity is the presence of water vapor in air. The amount of water vapor in air can influence human comfort as well as many manufacturing processes in industries. The presence of water vapor also influences various types of processes like physical, chemical, and biological processes.

Humidity measurement in industries is dangerous because it may cause the business cost of the product and the health and safety of the personnel. Thus, **humidity sensing** is very important, in particular in the control systems for industrial processes and human comfort.

Calculating or monitoring humidity is of dominant importance in many industrial & home applications. In semiconductor trade, humidity or moisture levels want to be properly controlled & monitored during wafer processing. In medical applications, humidity control is necessary for respiratory equipments, sterilizers, incubators, pharmaceutical processing, and biological products. Humidity control is also necessary in chemical gas purification, dryers, ovens, film desiccation, paper and textile construction, and food processing. In cultivation, measurement of humidity is significant for plantation protection (dew prevention), soil moisture monitoring, etc. For household applications, humidity control is required for living environment in buildings, cooking control for microwave ovens, etc. During all such applications and many others, **humidity sensors** are employed to provide an indication of the moisture levels in the environment.

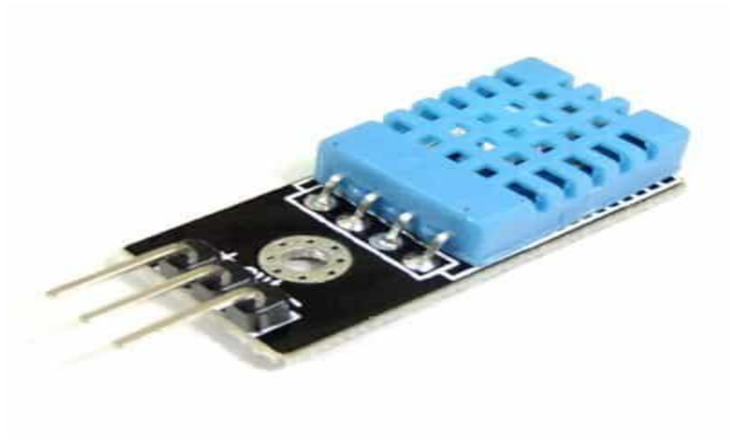


Fig.4.11.1 HUMIDITY SENSOR

The characteristics of humidity sensor are

- Accuracy,
- Hysterisis,
- Linearity
- Interchangeability
- Relaibility

4.11 Finger print sensor:

Fingerprint scanners are security systems using biometrics. They are utilized to open entryways and in other security applications. Individuals have an example of edges on their fingers. This unique finger impression can't be expelled or changed. Each unique finger impression is not quite the same as some other on the planet. Since there are endless mixes, fingerprints are quite utilized for distinguishing proof.

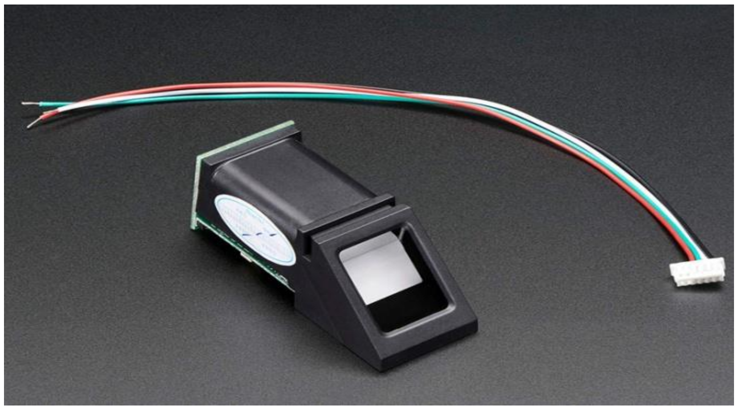


Fig.4.12.1 FINGER PRINT SENSOR

4.12 BUZZER:

A buzzer is an audio signaling device, which is either electromechanical or piezoelectric. Typical buzzers include alarm devices, timers and conformation of user input such as a mouse click or keystroke.

Fig.4.13.1 BUZZER

4.13 PIR sensor:

A PIR sensor can detect changes in the amount of infrared radiation impinging upon it, which varies depending on the temperature and surface characteristics of the objects in front of the sensor.[2] When an object, such as a person, passes in front of the background, such as a wall, the temperature at that point in the sensor's field of view will rise from room temperature to body temperature, and then back again. The sensor converts the resulting change in the incoming infrared radiation into a change in the output voltage, and this triggers the detection. Objects of similar temperature but different surface characteristics may also have a different infrared emission pattern, and thus moving them with respect to the background may trigger the detector as well.

PIRs come in many configurations for a wide variety of applications. The most common models have numerous Fresnel lenses or mirror segments, an effective range of about 10 meters (30 feet), and a field of view less than 180. Models with wider fields of view, including 360°, are available, typically designed to mount on a ceiling. Some larger PIRs are made with single segment mirrors and can sense changes in infrared energy over 30 meters (100 feet) from the PIR. There are also PIRs designed with reversible orientation mirrors which allow either broad coverage (110° wide) or very narrow "curtain" coverage, or with individually selectable segments to "shape" the coverage.

Fig.4.14.1 PIR SENSOR

4.14 <u>LCD (Liquid Cristal Display)</u>

A liquid crystal display (LCD) is a thin, flat display device made up of any number of color or monochrome pixels arrayed in front of a light source or reflector. Each pixel consists of a column of liquid crystal molecules suspended between two transparent electrodes, and two polarizing filters, the axes of polarity of which are perpendicular to each other. Without the liquid crystals between them, light passing through one would be blocked by the other. The liquid crystal twists the polarization of light entering one filter to allow it to pass through the other.

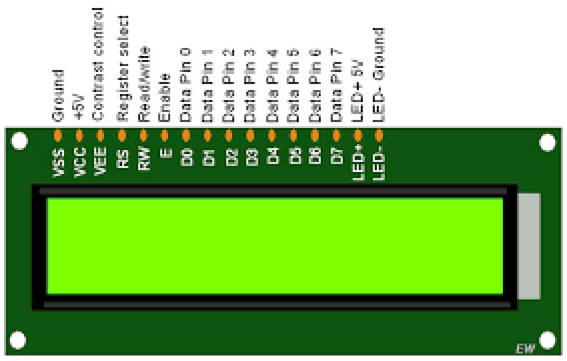
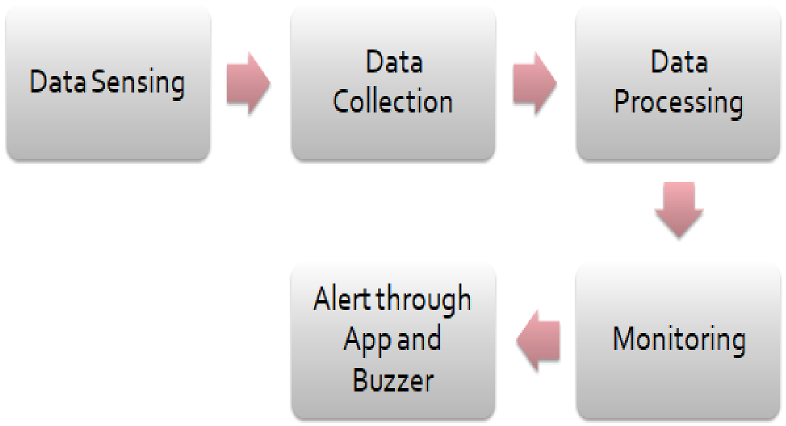


Fig.4.15.1 LCD

CHAPTER 5

DESIGN METHODOLOGY

Methodology:



First we sense the data from the surroundings through the sensors that have been established ,sensed data from them are collected by the controller Raspberry Pi and collected data is processed and find out if any parameter exceeded the threshold value.if exceeded we provides alert through android application and buzzer. In this we are using multiple sensors for the monitoring purpose .

5.1 Block diagram:

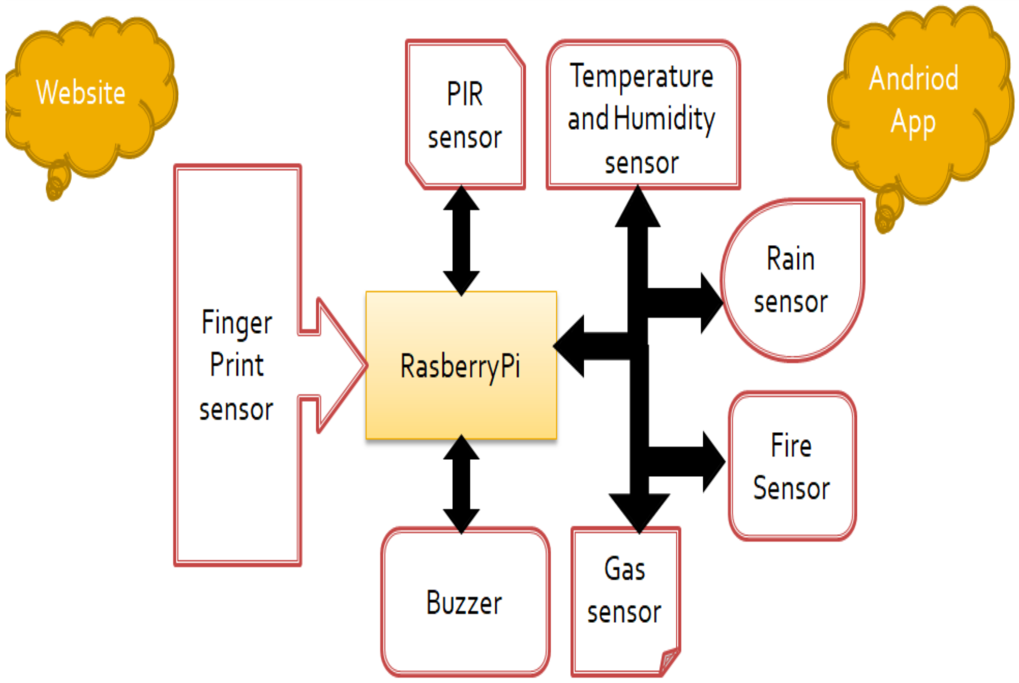


Fig.5.1.1 BLOCK DIAGRAM

Gas Sensor, Fire Sensor, Humidity Sensor, LDR are given as inputs to the raspberry pi. Monitor, buzzer, LED, camera are connected as outputs. The outputs from various sensors interfaced are processed by raspberry pi and given as inputs to the devices connected to output pins of raspberry pi.

5.2 System Setup:

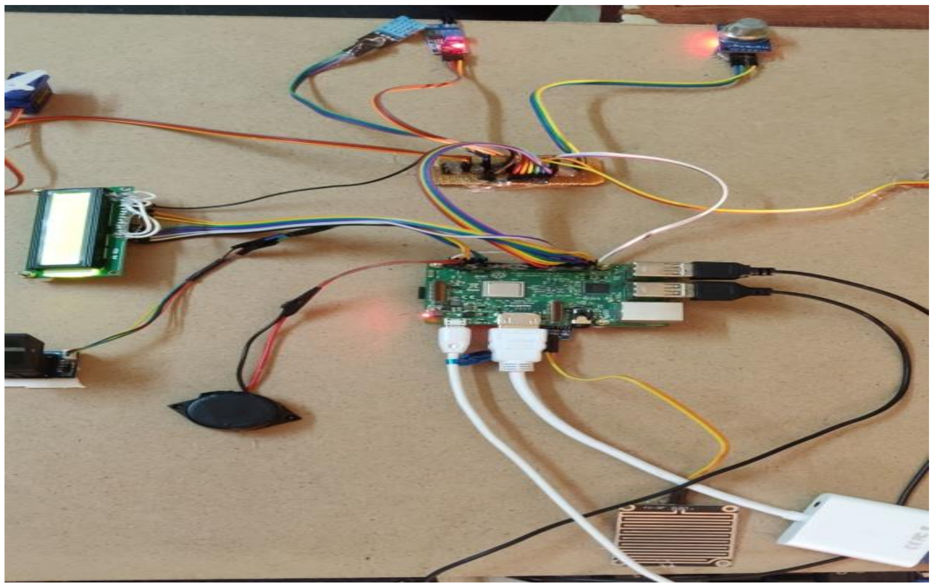


Fig.5.2.1 SYSTEM SETUP

The raspberry pi takes the input from fire sensor and continuously displays whether the fire is detected or not. It also rings the buzzer when fire is detected.

The gas sensor senses the gases present in the atmosphere and when any of the gas present in atmosphere exceeds the threshold value then the system alerts by ringing the buzzer.

Humidity sensor senses the humidity present in environment and gives the values on monitor and when any value crosses the threshold value buzzer is activated.

IP webcam app acts as a camera. It takes the picture whenever a buzzer is activated and uploads it to the server.

5.3 system code:

Python code for Servo motor:

```
from tkinter import *
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
pwm = GPIO.PWM(18, 100)
pwm.start(5)

GPIO.setmode(GPIO.BCM)
GPIO.setup(15, GPIO.OUT)
pwm2 = GPIO.PWM(15, 100)
pwm2.start(5)

class App:

    def __init__(self, master):
        frame = Frame(master)
        frame.pack()
        scale = Scale(frame, from_=0, to=180,
              orient=HORIZONTAL, command=self.update)
        scale.grid(row=0)

        scale = Scale(frame, from_=0, to=180,
              orient=HORIZONTAL, command=self.update2)
        scale.grid(row=0,column=10)
   def update(self, angle):
        duty = float(angle) / 10.0 + 2.5
        pwm.ChangeDutyCycle(duty)

    def update2(self, angle):
        duty = float(angle) / 10.0 + 2.5
        pwm2.ChangeDutyCycle(duty)

root = Tk()
root.wm_title('Servo Controler')
app = App(root)
root.geometry("200x50+0+0")
root.mainloop()
```

Python code for Lcd display:

```
#!/usr/bin/python
#-------------------------------------
#    ___  ___  _ ____
#   / _ \/ _ \(_) __/__  __ __
#  / , _/ ___/ /\ \/ _ \/ // /
# /_/|_/_/  /_/___/ .__/\_, /
#                /_/   /___/
#
#  lcd_16x2.py
#  16x2 LCD Test Script
#
# Author : Matt Hawkins
# Date   : 06/04/2015
#
# http://www.raspberrypi-spy.co.uk/
#
# Copyright 2015 Matt Hawkins
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See
the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program.  If not, see <http://www.gnu.org/licenses/>.
#
#-------------------------------------

# The wiring for the LCD is as follows:
# 1 : GND
# 2 : 5V
# 3 : Contrast (0-5V)*
# 4 : RS (Register Select)
# 5 : R/W (Read Write)       - GROUND THIS PIN
# 6 : Enable or Strobe
# 7 : Data Bit 0             - NOT USED
# 8 : Data Bit 1             - NOT USED
# 9 : Data Bit 2              - NOT USED
# 10: Data Bit 3              - NOT USED
# 11: Data Bit 4
# 12: Data Bit 5
# 13: Data Bit 6
# 14: Data Bit 7
# 15: LCD Backlight +5V**
# 16: LCD Backlight GND


#import
import RPi.GPIO as GPIO
import time
import serial
import httplib, urllib
import datetime

# Define GPIO to LCD mapping
LCD_RS = 7
LCD_E  = 8
LCD_D4 = 25
LCD_D5 = 24
LCD_D6 = 23
LCD_D7 = 18

led = 21
sw = 17
buz = 2
mot = 3

clk = 6
din = 19
dout = 13
cs = 26

# Define some device constants
LCD_WIDTH = 16    # Maximum characters per line
LCD_CHR = True
LCD_CMD = False

LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line
LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line
```

```
# Timing constants
E_PULSE = 0.0005
E_DELAY = 0.0005



ser = serial.Serial('/dev/ttyUSB0', 9600, timeout=1)

def main():
  # Main program block

  GPIO.setwarnings(False)
  GPIO.setmode(GPIO.BCM)       # Use BCM GPIO numbers
  GPIO.setup(LCD_E, GPIO.OUT)  # E
  GPIO.setup(LCD_RS, GPIO.OUT) # RS
  GPIO.setup(LCD_D4, GPIO.OUT) # DB4
  GPIO.setup(LCD_D5, GPIO.OUT) # DB5
  GPIO.setup(LCD_D6, GPIO.OUT) # DB6
  GPIO.setup(LCD_D7, GPIO.OUT) # DB7

  GPIO.setup(led, GPIO.OUT)    # led
  GPIO.setup(mot, GPIO.OUT)      # switch

  GPIO.setup(sw, GPIO.IN)      # switch
  GPIO.setup(buz, GPIO.OUT)      # switch


  GPIO.setup(clk, GPIO.OUT)  # E
  GPIO.setup(din, GPIO.OUT) # RS
  GPIO.setup(dout, GPIO.IN) # DB4
  GPIO.setup(cs, GPIO.OUT) # DB4

  # Initialise display
  lcd_init()
  lcd_byte(0x01,LCD_CMD)
  lcd_string("   WELCOME",LCD_LINE_1)
  lcd_string("Intializing..",LCD_LINE_2)
  print 'welcome'
  time.sleep(2)

  GPIO.output(buz, True)
  GPIO.output(mot, True)
  GPIO.output(led, True)
  time.sleep(2)
  lcd_byte(0x01,LCD_CMD)
```

```
lcd_string("T:      H:     ",LCD_LINE_1)
lcd_string("S:      L:     ",LCD_LINE_2)

ec=0;

while True:

  hum=adc(0)                    # MEMS  X-access

  lcd_byte(0x82,LCD_CMD)
  lcd_byte((hum/1000)+48,LCD_CHR)
  lcd_byte((hum/100)%10+48,LCD_CHR)
  lcd_byte((hum/10)%10+48,LCD_CHR)
  lcd_byte(hum%10+48,LCD_CHR)

  ldr=adc(1)                    # MEMS  Y-access
  lcd_byte(0x89,LCD_CMD)
  lcd_byte((ldr/1000)+48,LCD_CHR)
  lcd_byte((ldr/100)%10+48,LCD_CHR)
  lcd_byte((ldr/10)%10+48,LCD_CHR)
  lcd_byte(ldr%10+48,LCD_CHR)

  smk=adc(3)                    # SMOKE Sensor
  lcd_byte(0xc2,LCD_CMD)
  lcd_byte((smk/1000)+48,LCD_CHR)
  lcd_byte((smk/100)%10+48,LCD_CHR)
  lcd_byte((smk/10)%10+48,LCD_CHR)
  lcd_byte(smk%10+48,LCD_CHR)

  tmp=adc(2)
  tmp=tmp/10# EYE Sensor
  lcd_byte(0xc9,LCD_CMD)
  lcd_byte((tmp/1000)+48,LCD_CHR)
  lcd_byte((tmp/100)%10+48,LCD_CHR)
  lcd_byte((tmp/10)%10+48,LCD_CHR)
  lcd_byte(tmp%10+48,LCD_CHR)
  print "Temp: %d" %tmp
  print "Hum:  %d " %hum
  print "Ldr:  %d " %ldr
  print "smk:  %d " %smk
  pfile=""
  if((tmp > 40) or (hum > 4000) or (ldr > 3000) or (smk > 3000)):

    GPIO.output(buz, False)
    lcd_byte(0x01,LCD_CMD)
```

```
      time.sleep(1)
      lcd_byte(0x01,LCD_CMD)
      lcd_string("Abnormal condtn...",LCD_LINE_1)
      lcd_string("Sending SMS...",LCD_LINE_2)
      if(tmp > 40):
        send_sms(1)

      if(hum > 4000):
        print 'sending sms:hum'
        send_sms(2)

      if(ldr > 3000):
        print 'sending sms:ldr'
        send_sms(3)

      if(snk > 3000):
        print 'sending sms:Smk'
        send_sms(4)

    line = ""
    data = ""
    data = ser.read()
    if(data != ""):
        while(data != "\r"):
          data = ser.read()
          line = line + data

    rcv=""
    rcv = ser.read(100)
    print rcv


    f=open('logdata.txt','a')
    now = datetime.datetime.now()
    timestamp = now.strftime("%Y/%m/%d %H:%M")

    outstring = "\n"+str(timestamp)+"INDIA\n"

    f.write(outstring)
    f.close()
   params = urllib.urlencode({'field1': tmp, 'field2': hum, 'field3':
ldr,  'field4':           smk,'key':'2PXWFAHQAYNIGGJ1'})
    headers = {"Content-type": "application/x-www-form-
urlencoded","Accept": "text/plain"}
    conn = httplib.HTTPConnection("api.thingspeak.com:80")
    conn.request("POST", "/update", params, headers)
    response = conn.getresponse()
    print response.status, response.reason
    data = response.read()
    print data
    conn.close()

    link='https://api.thingspeak.com/talkbacks/10518/commands/2648
307?api_key=SU6BZO1E6K4ISKGB'
    f = urllib.urlopen(link)
    myfile = f.readline()

    if(myfile == 'D1 ON'):
      GPIO.output(led, True)

    elif(myfile == 'D1 OFF'):
      GPIO.output(led, False)

    elif(myfile == 'D2 ON'):
      GPIO.output(mot, True)

    elif(myfile == 'D2 OFF'):
      GPIO.output(mot, False)

    else:

      if(myfile != pfile):
       print myfile
       pfile=myfile

    line = ""
    data = ""
    data = ser.read()
    if(data != ""):
        while(data != "\r"):
          data = ser.read()
          line = line + data

  rcv=""
  rcv = ser.read(10)
  print rcv
  time.sleep(10)
  def gsm_init():
 ser.write('AT'+'\r\n')
 rcv = ser.read(10)
 print rcv
 time.sleep(1)
 ser.write('ATE0'+'\r\n')      # Disable the Echo
 rcv = ser.read(10)
 print rcv
 time.sleep(1)
 ser.write('AT+CMGF=1'+'\r\n')  # Select Message format as Text mode
 rcv = ser.read(10)
 print rcv
 time.sleep(1)

 ser.write('AT+CNMI=2,2,0,0,0'+'\r\n')   # New SMS Message Indications
 rcv = ser.read(10)
 print rcv
 time.sleep(1)
 def adc(k):
i = 0
bt0 = 0
bt1 = 0
bt2 = 0
val = 0
 x = 0
 A=0
 C=0
 GPIO.output(clk, False)
 GPIO.output(cs, True)
 GPIO.output(cs, False)
x = (k*4)+0x60
      A = x>>4
      C = x<<4
mask=0x80
while(mask>0):
         GPIO.output(clk, True)
          if(A & mask):
              GPIO.output(din, True)
          else:
              GPIO.output(din, False)

          GPIO.output(clk, False)

          if(GPIO.input(dout)==True):

              bt0 |= mask
          mask>>=1
```

```
mask=0x80

while(mask>0):

    GPIO.output(clk, True)
    if(C & mask):
        GPIO.output(din, True)
    else:
        GPIO.output(din, False)

    GPIO.output(clk, False)

    if(GPIO.input(dout)==True):

        bt1 |= mask
    mask>>=1

mask=0x80

while(mask>0):

    GPIO.output(clk, True)
    if(0x00 & mask):
        GPIO.output(din, True)
    else:
        GPIO.output(din, False)

    GPIO.output(clk, False)

    if(GPIO.input(dout)==True):

        bt2 |= mask
    mask>>=1


GPIO.output(cs, True)
val = (bt1 & 0x0f)
val = ((val<<8) | bt2)


return val
```

```
def send_sms(p):


  ser.write('AT+CMGS="9676767270"'+'\r\n')
  rcv = ser.read(10)
  print rcv
  time.sleep(1)
  if(p==1):
      ser.write('OVER TEMPERATURE'+'\r\n')  # Message

  if(p==2):
      ser.write('OVER HUMIDITY^'+'\r\n')  # Message

  if(p==3):
      ser.write('LOW LIGHT'+'\r\n')  # Message

  if(p==4):
      ser.write('SMOKE DETECTED'+'\r\n')  # Message
  rcv = ser.read(10)

  time.sleep(1)
  ser.write("\x1A") # Enable to send SMS
  for i in range(10):
   rcv = ser.read(10)
   print rcv
  time.sleep(1)
def lcd_init():
  # Initialise display
  lcd_byte(0x33,LCD_CMD) # 110011 Initialise
  lcd_byte(0x32,LCD_CMD) # 110010 Initialise
  lcd_byte(0x06,LCD_CMD) # 000110 Cursor move direction
  lcd_byte(0x0C,LCD_CMD) # 001100 Display On,Cursor Off, Blink Off
  lcd_byte(0x28,LCD_CMD) # 101000 Data length, number of lines, font
size
  lcd_byte(0x01,LCD_CMD) # 000001 Clear display
  time.sleep(E_DELAY)

def lcd_byte(bits, mode):
  # Send byte to data pins
  # bits = data
  # mode = True  for character
  #        False for command

  GPIO.output(LCD_RS, mode) # RS
  # High bits
  GPIO.output(LCD_D4, False)
  GPIO.output(LCD_D5, False)
  GPIO.output(LCD_D6, False)
  GPIO.output(LCD_D7, False)
  if bits&0x10==0x10:
    GPIO.output(LCD_D4, True)
  if bits&0x20==0x20:
    GPIO.output(LCD_D5, True)
  if bits&0x40==0x40:
    GPIO.output(LCD_D6, True)
  if bits&0x80==0x80:
    GPIO.output(LCD_D7, True)

  # Toggle 'Enable' pin
  lcd_toggle_enable()

  # Low bits
  GPIO.output(LCD_D4, False)
  GPIO.output(LCD_D5, False)
  GPIO.output(LCD_D6, False)
  GPIO.output(LCD_D7, False)
  if bits&0x01==0x01:
    GPIO.output(LCD_D4, True)
  if bits&0x02==0x02:
    GPIO.output(LCD_D5, True)
  if bits&0x04==0x04:
    GPIO.output(LCD_D6, True)
  if bits&0x08==0x08:
    GPIO.output(LCD_D7, True)

  # Toggle 'Enable' pin
  lcd_toggle_enable()

def lcd_toggle_enable():
  # Toggle enable
  time.sleep(E_DELAY)
  GPIO.output(LCD_E, True)
  time.sleep(E_PULSE)
  GPIO.output(LCD_E, False)
  time.sleep(E_DELAY)

def lcd_string(message,line):
  # Send string to display
message = message.ljust(LCD_WIDTH," ")
```

```
lcd_byte(line, LCD_CMD)
for i in range(LCD_WIDTH):
    lcd_byte(ord(message[i]),LCD_CHR)

if __name__ == '__main__':
try:
    main()
  except KeyboardInterrupt:
    pass
  finally:
    lcd_byte(0x01, LCD_CMD)
    lcd_string("CLOSED !",LCD_LINE_1)
    GPIO.cleanup()
```

Python code for Finger searching:

```
import serial, time, datetime, struct
import sys

ser = serial.Serial('/dev/ttyS0',57200)
pack = [0xef01, 0xffffffff, 0x1]

def printx(l):
for i in l:
print hex(i),
print ''

def readPacket():
time.sleep(1)
w = ser.inWaiting()
ret = []
if w >= 9:
s = ser.read(9) #partial read to get length
ret.extend(struct.unpack('!HIBH', s))
ln = ret[-1]

time.sleep(1)
w = ser.inWaiting()
if w >= ln:
s = ser.read(ln)
form = '!' + 'B' * (ln - 2) + 'H'
ret.extend(struct.unpack(form, s))
return ret
```

```
def writePacket(data):
pack2 = pack + [(len(data) + 2)]
a = sum(pack2[-2:] + data)
pack_str = '!HIBH' + 'B' * len(data) + 'H'
l = pack2 + data + [a]
s = struct.pack(pack_str, *l)
ser.write(s)


def verifyFinger():
data = [0x13, 0x0, 0, 0, 0]
writePacket(data)
s = readPacket()
return s[4]

def genImg():
data = [0x1]
writePacket(data)
s = readPacket()
return s[4]

def img2Tz(buf):
data = [0x2, buf]
writePacket(data)
s = readPacket()
return s[4]

def regModel():
data = [0x5]
writePacket(data)
s = readPacket()
return s[4]

def search():
data = [0x4, 0x1, 0x0, 0x0, 0x0, 0xff]
writePacket(data)
s = readPacket()
return s[4:-1]

while(1):

        if verifyFinger():
                print 'Verification Error'
                sys.exit(-1)
```

```
	print 'Put finger',
	sys.stdout.flush()

	time.sleep(1)
	for _ in range(15):
		g = genImg()
		if g == 0:
			break
		#time.sleep(1)

		print '.',
		sys.stdout.flush()

	print ''
	sys.stdout.flush()
	if g != 0:
		sys.exit(-1)

	if img2Tz(1):
		print 'Conversion Error'
		sys.exit(-1)

	r = search()
	print r
	print 'Search result', r


	########################################################
############

	#NORMAL PERSON
	########################################################
############


	if ((r[0] == 0) and (r[2] ==1 or r[3]==2 or r[4]==3)):
		print 'AUTHORIZED....'


	else:
		print 'UN AUTHORIZED....'
```

Python code for finger adding:

```
import serial, time, datetime, struct
import sys

ser = serial.Serial('/dev/ttyS0',57200)
pack = [0xef01, 0xffffffff, 0x1]

def printx(l):
for i in l:
print hex(i),
print ''

def readPacket():
time.sleep(1)
w = ser.inWaiting()
ret = []
if w >= 9:
s = ser.read(9) #partial read to get length
ret.extend(struct.unpack('!HIBH', s))
ln = ret[-1]

time.sleep(1)
w = ser.inWaiting()
if w >= ln:
s = ser.read(ln)
form = '!' + 'B' * (ln - 2) + 'H'
ret.extend(struct.unpack(form, s))
return ret
def writePacket(data):
pack2 = pack + [(len(data) + 2)]
a = sum(pack2[-2:] + data)
pack_str = '!HIBH' + 'B' * len(data) + 'H'
l = pack2 + data + [a]
s = struct.pack(pack_str, *l)
ser.write(s)

def verifyFinger():
data = [0x13, 0x0, 0, 0, 0]
writePacket(data)
s = readPacket()
return s[4]

def genImg():
data = [0x1]
writePacket(data)
```

```
s = readPacket()
return s[4]

def img2Tz(buf):
data = [0x2, buf]
writePacket(data)
s = readPacket()
return s[4]

def regModel():
data = [0x5]
writePacket(data)
s = readPacket()
return s[4]

def store(id):
data = [0x6, 0x1, 0x0, id]
writePacket(data)
s = readPacket()
return s[4]

def search():
data = [0x4, 0x1, 0x0, 0x0, 0x0, 0x5]
writePacket(data)
s = readPacket()
return s[4:-1]

if verifyFinger():
print 'Verification Error'
sys.exit(0)

print 'Put finger',
sys.stdout.flush()

time.sleep(1)
while genImg():
time.sleep(0.1)

print '.',
sys.stdout.flush()

print ''
sys.stdout.flush()

if img2Tz(1):
```

```
print 'Conversion Error'
sys.exit(0)

print 'Put finger again',
sys.stdout.flush()

time.sleep(1)
while genImg():
time.sleep(0.1)
print '.',
sys.stdout.flush()

print ''
sys.stdout.flush()

if img2Tz(2):
print 'Conversion Error'
sys.exit(0)

if regModel():
print 'Template Error'
sys.exit(0)
id = 2
if store(id):
print 'Store Error'
sys.exit(0)

print "Enrolled successfully at id %d"%id
```

Python code for DTH11:

```
import Adafruit_DHT
import time
import httplib, urllib
import RPi.GPIO as  GPIO

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

LCD_RS = 21
LCD_E  = 20
LCD_D4 = 16
LCD_D5 = 7
```

```
LCD_D6 = 12
LCD_D7 = 1


LCD_WIDTH = 16    # Maximum characters per line
LCD_CHR = True
LCD_CMD = False

LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line
LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line

# Timing constants
E_PULSE = 0.0005
E_DELAY = 0.0005

GPIO.setup(LCD_E, GPIO.OUT)  # E
GPIO.setup(LCD_RS, GPIO.OUT) # RS
GPIO.setup(LCD_D4, GPIO.OUT) # DB4
GPIO.setup(LCD_D5, GPIO.OUT) # DB5
GPIO.setup(LCD_D6, GPIO.OUT) # DB6
GPIO.setup(LCD_D7, GPIO.OUT) # DB7

import serial, time, datetime, struct
import sys

ser = serial.Serial('/dev/ttyS0',57200)
pack = [0xef01, 0xffffffff, 0x1]

def lcd_init():
  # Initialise display
  lcd_byte(0x33,LCD_CMD) # 110011 Initialise
  lcd_byte(0x32,LCD_CMD) # 110010 Initialise
  lcd_byte(0x06,LCD_CMD) # 000110 Cursor move direction
  lcd_byte(0x0C,LCD_CMD) # 001100 Display On,Cursor Off, Blink Off
  lcd_byte(0x28,LCD_CMD) # 101000 Data length, number of lines, font
size
  lcd_byte(0x01,LCD_CMD) # 000001 Clear display
  time.sleep(E_DELAY)

def lcd_byte(bits, mode):
  # Send byte to data pins
  # bits = data
  # mode = True  for character
  #        False for command
  GPIO.output(LCD_RS, mode) # RS

  # High bits
  GPIO.output(LCD_D4, False)
  GPIO.output(LCD_D5, False)
  GPIO.output(LCD_D6, False)
  GPIO.output(LCD_D7, False)
  if bits&0x10==0x10:
    GPIO.output(LCD_D4, True)
  if bits&0x20==0x20:
    GPIO.output(LCD_D5, True)
  if bits&0x40==0x40:
    GPIO.output(LCD_D6, True)
  if bits&0x80==0x80:
    GPIO.output(LCD_D7, True)

  # Toggle 'Enable' pin
  lcd_toggle_enable()

  # Low bits
  GPIO.output(LCD_D4, False)
  GPIO.output(LCD_D5, False)
  GPIO.output(LCD_D6, False)
  GPIO.output(LCD_D7, False)
  if bits&0x01==0x01:
    GPIO.output(LCD_D4, True)
  if bits&0x02==0x02:
    GPIO.output(LCD_D5, True)
  if bits&0x04==0x04:
    GPIO.output(LCD_D6, True)
  if bits&0x08==0x08:
    GPIO.output(LCD_D7, True)

  # Toggle 'Enable' pin
  lcd_toggle_enable()

def lcd_toggle_enable():
  # Toggle enable
  time.sleep(E_DELAY)
  GPIO.output(LCD_E, True)
  time.sleep(E_PULSE)
  GPIO.output(LCD_E, False)
  time.sleep(E_DELAY)
```

```
def lcd_string(message,line):
  # Send string to display

  message = message.ljust(LCD_WIDTH," ")

  lcd_byte(line, LCD_CMD)

  for i in range(LCD_WIDTH):
    lcd_byte(ord(message[i]),LCD_CHR)


def printx(l):
for i in l:
print hex(i),
print ''

def readPacket():
time.sleep(1)
w = ser.inWaiting()
ret = []
if w >= 9:
s = ser.read(9) #partial read to get length
ret.extend(struct.unpack('!HIBH', s))
ln = ret[-1]

time.sleep(1)
w = ser.inWaiting()
if w >= ln:
s = ser.read(ln)
form = '!' + 'B' * (ln - 2) + 'H'
ret.extend(struct.unpack(form, s))
return ret


def writePacket(data):
pack2 = pack + [(len(data) + 2)]
a = sum(pack2[-2:] + data)
pack_str = '!HIBH' + 'B' * len(data) + 'H'
l = pack2 + data + [a]
s = struct.pack(pack_str, *l)
ser.write(s)


def verifyFinger():
data = [0x13, 0x0, 0, 0, 0]
```

```
writePacket(data)
s = readPacket()
return s[4]

def genImg():
data = [0x1]
writePacket(data)
s = readPacket()
return s[4]

def img2Tz(buf):
data = [0x2, buf]
writePacket(data)
s = readPacket()
return s[4]

def regModel():
data = [0x5]
writePacket(data)
s = readPacket()
return s[4]

def search():
data = [0x4, 0x1, 0x0, 0x0, 0x0, 0xff]
writePacket(data)
s = readPacket()
return s[4:-1]


sensor = Adafruit_DHT.DHT11

sensor_pin = 5

smk=26
rain=19
pir=13
fire=6

servo=11

buz=2

GPIO.setup(buz,GPIO.OUT)
GPIO.setup(smk,GPIO.IN)
GPIO.setup(rain,GPIO.IN)
GPIO.setup(pir,GPIO.IN)
GPIO.setup(fire,GPIO.IN)
GPIO.setup(servo,GPIO.OUT)

pwm = GPIO.PWM(servo, 100)
pwm.start(10)
time.sleep(1)
pwm.start(0)

GPIO.output(buz,1)

lcd_init()
lcd_byte(0x01,LCD_CMD)
lcd_string("    WELCOME",LCD_LINE_1)
lcd_string(" ",LCD_LINE_2)

while True:
    humidity, temperature = Adafruit_DHT.read_retry(sensor, sensor_pin)

    s_val = GPIO.input(smk)
    r_val = GPIO.input (rain)
    p_val = GPIO.input(pir)
    f_val = GPIO.input(fire)


    print "TEMP:" + str(temperature)
    print "HUMD:" + str(humidity)
    print "SMOKE:" + str(s_val)
    print "RAIN:" + str(r_val)
    print "PIR :" + str(p_val)
    print "FIRE:" + str(f_val)

    lcd_string("T:" + str(temperature) + " H:" + str(humidity) ,LCD_LINE_1)

    lcd_string("R:" + str(r_val) + " P:" + str(p_val) + " F:" + str(f_val)+ " S:" +
str(s_val),LCD_LINE_2)

 if(temperature>40 or humidity>90 or s_val==0 or f_val==0 or p_val==1 or
r_val==0):
        GPIO.output(buz,0)
        time.sleep(1)
        GPIO.output(buz,1)

w_link="https://api.thingspeak.com/update?api_key=M79UVU95D1W1T
ES0&field1="
```

```
w_link += str(temperature)
w_link += "&field2="
w_link += str(humidity)
w_link += "&field3="
w_link += str(s_val)
w_link += "&field4="
w_link += str(r_val)
w_link += "&field5="
w_link += str(p_val)
w_link += "&field6="
w_link += str(f_val)


urllib.urlopen(w_link)
time.sleep(2)

if verifyFinger():
        print 'Verification Error'
        sys.exit(-1)

print 'Put finger',
sys.stdout.flush()

time.sleep(1)
for _ in range(5):
        g = genImg()
        if g == 0:
                break
        #time.sleep(1)

        print '.',
        sys.stdout.flush()

print ''
sys.stdout.flush()
if g != 0:
        time.sleep(1)

if img2Tz(1):
        print 'Conversion Error'
        time.sleep(1)

r = search()
print r
print 'Search result', r
```

```
        ############################################################
############

        #NORMAL PERSON
        ############################################################
############


    if ((r[0] == 0) and (r[2] ==1 or r[3]==2 or r[4]==3)):
            print 'AUTHORIZED....'

            pwm.start(90)
            time.sleep(1)
            pwm.start(0)

            time.sleep(4)

            pwm.start(10)
            time.sleep(1)
            pwm.start(0)



    else:
            print 'UN AUTHORIZED....'
            GPIO.output(buz,False)
            time.sleep(2)
            GPIO.output(buz,True)

#    r_link='https://api.thingspeak.com/channels/812713/fields/1/last?
api_key=HL53WMDT7QEF88M7'
#    f = urllib.urlopen(r_link)
#    rcvcmd = f.readline()
#    if(rcvcmd == "1"):
#        GPIO.output(buz,True)

#    if(rcvcmd == "2"):
#        GPIO.output(buz,False)
```

CHAPTER 6
EXPERIMENTAL RESULTS

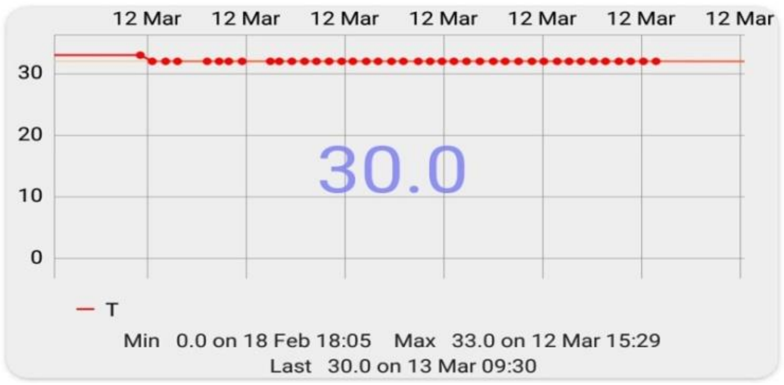


Fig : 6.1 temperature v/s time graph

- it shows the temperature variation for the 24 hours If the temperature exceeds threshold value it provides alert through the android application. We can monitor the each and every thing like temperature ,humidity, fire accidents etc.

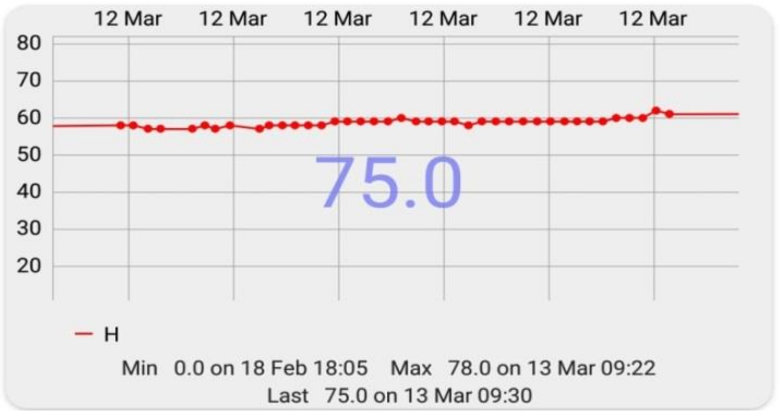
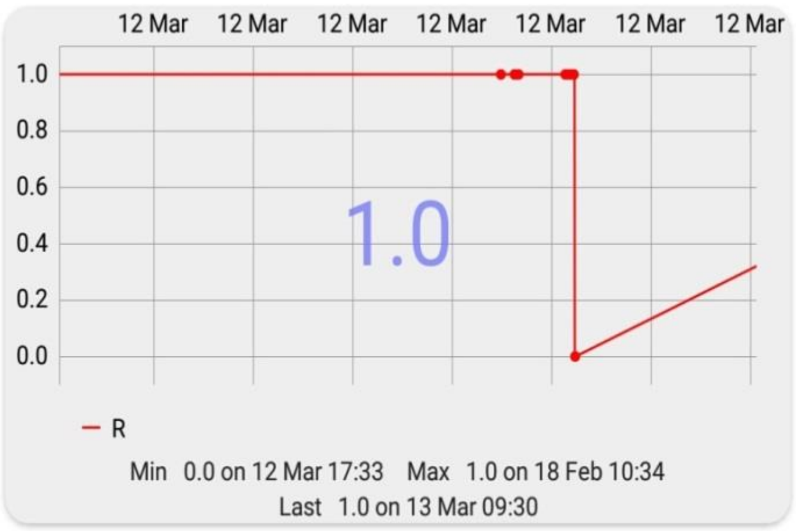


Fig: 6.2 Humidity v/s time graph

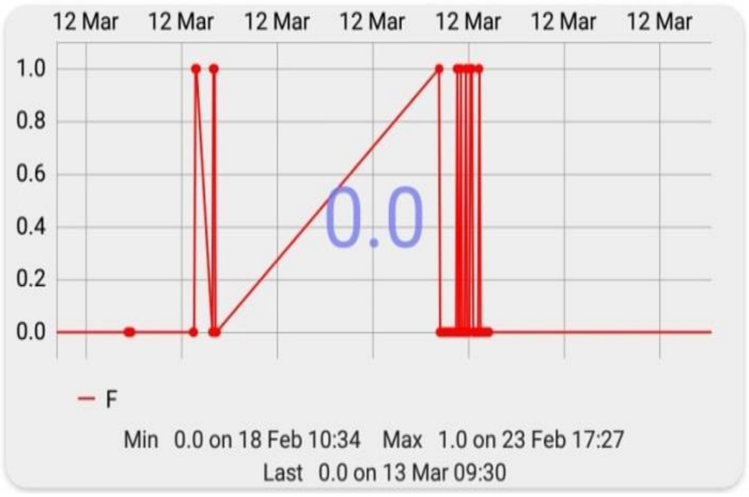
- it shows about the humidity level in day which updates for every 15 seconds time . If the Humidity exceeds threshold value it provides alert through the android application.



Min 0.0 on 12 Mar 17:33 Max 1.0 on 18 Feb 10:34
Last 1.0 on 13 Mar 09:30

Fig: 6.3 rain detection v/s time graph

- it indicates whether rain is present or not. if the rain is not falling it shows1 or else it shows 0 which will be displayed in the android application.



Min 0.0 on 18 Feb 10:34 Max 1.0 on 23 Feb 17:27
Last 0.0 on 13 Mar 09:30

Fig: 6.4 Fire detection v/s time graph

- it indicates whether Fire accident occurs or not. If the intensity of fire is high then it indicates 1 or it indicates 0.If there is fire accident ,it provides alert through android application.

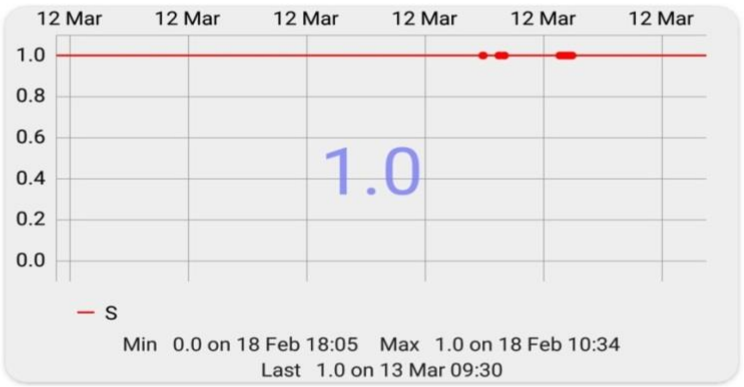


Fig : 6.5 Smoke detection v/s time graph

- it indicates whether smoke is detected or not if the smoke is present then will be indicated as 1 in the graph or else 0 will be indicated and it provides alert through the android application if it exceeds threshold value.

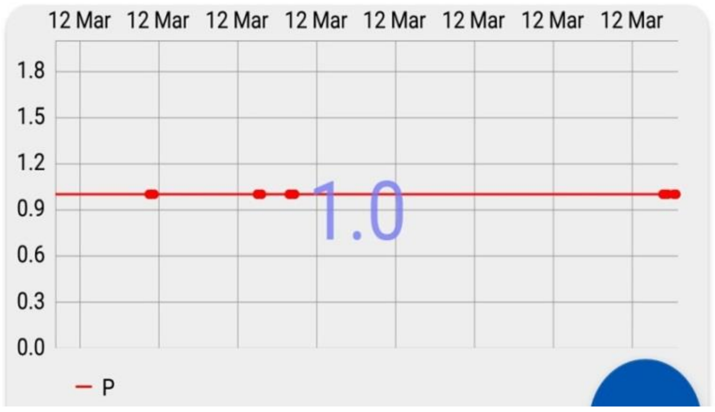
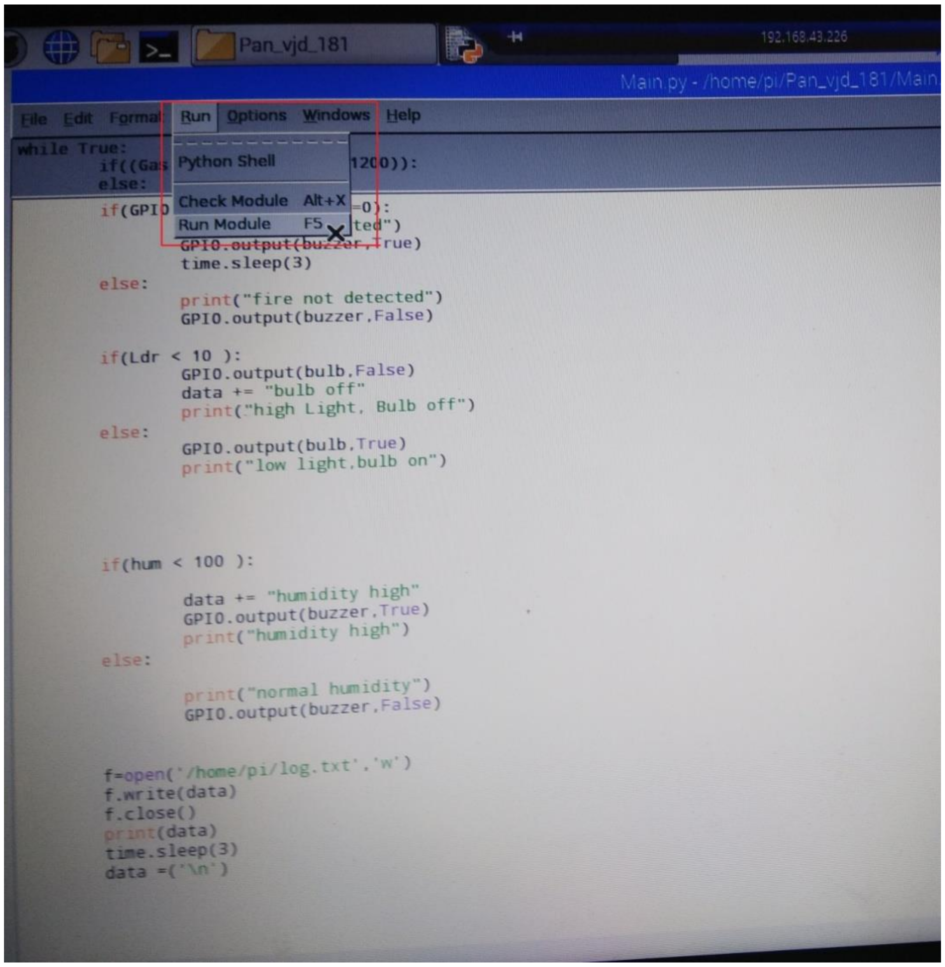


Fig: 6.6 human detection v/s time graph

- it indicates whether motion is detected or not. if PIR sensor detects the motion then1 will be indicated or else 0 will be indicated in the graph . it provides alert through the android application if it exceeds threshold value.

Running module:

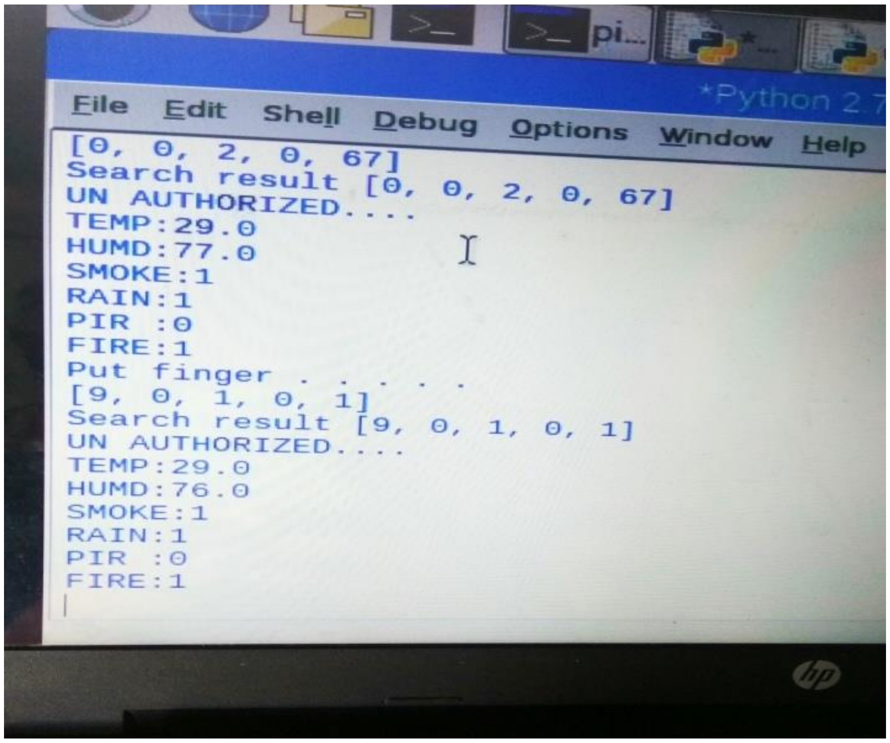
Code is written in python language for interfacing the inputs and outputs to raspberry pi. The outputs are activated based on the threshold values given in the program. Run the module after saving the program.


Pan_vjd_181
192.168.43.226
Main.py - /home/pi/Pan_vjd_181/Main
File Edit Format Run Options Windows Help
Python Shell
Check Module Alt+X
Run Module F5

```
while True:
        if((Ga                    1200)):
        else:
        if(GPI                    =0):
                                  ted")
                GPIO.output(buzzer,True)
                time.sleep(3)
        else:
                print("fire not detected")
                GPIO.output(buzzer,False)

        if(Ldr < 10 ):
                GPIO.output(bulb,False)
                data += "bulb off"
                print("high Light, Bulb off")
        else:
                GPIO.output(bulb,True)
                print("low light,bulb on")



        if(hum < 100 ):

                data += "humidity high"
                GPIO.output(buzzer,True)
                print("humidity high")
        else:

                print("normal humidity")
                GPIO.output(buzzer,False)


        f=open('/home/pi/log.txt','w')
        f.write(data)
        f.close()
        print(data)
        time.sleep(3)
        data =('\n')
```

Fig.6.7 running module

The following window appears showing the outputs.It can also be viewed from log.txt file



```
[0, 0, 2, 0, 67]
Search result [0, 0, 2, 0, 67]
UN AUTHORIZED....
TEMP:29.0
HUMD:77.0
SMOKE:1
RAIN:1
PIR :0
FIRE:1
Put finger . . . . .
[9, 0, 1, 0, 1]
Search result [9, 0, 1, 0, 1]
UN AUTHORIZED....
TEMP:29.0
HUMD:76.0
SMOKE:1
RAIN:1
PIR :0
FIRE:1
```

Fig 6.8 Code Output

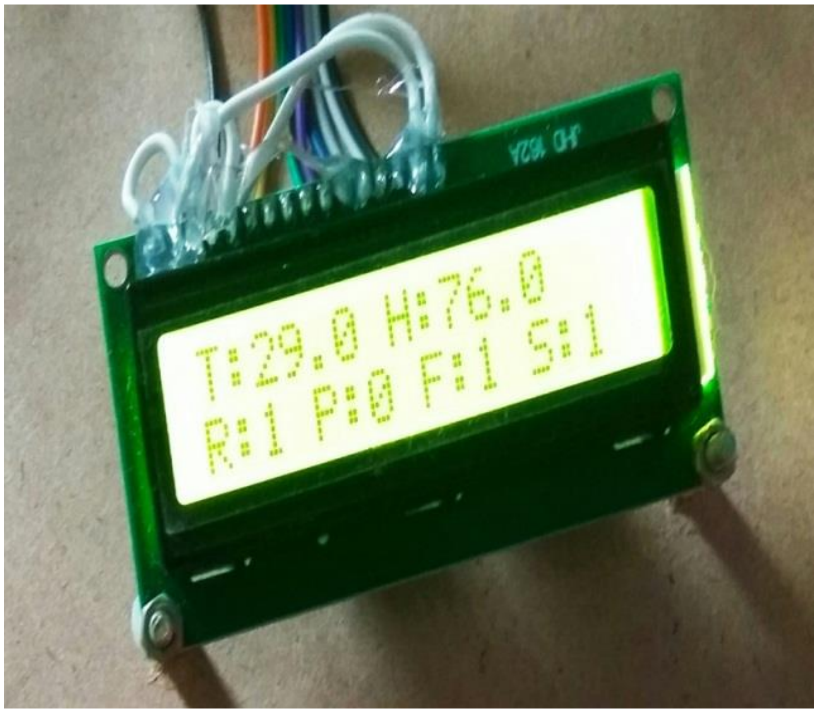


Fig 6.9 LCD Display

CHAPTER 7

CONCLUSION

This project describes an IOT based interactive ,home wireless system and embedded data acquisition system to display on web page and Android Application.The research on instant monitoring of toxic gases and other various factors present in home has analyzed . A real time monitoring system is developed to provide clearer and more point to point perspective . This system is displaying the parameters on the LCD at the section where sensor unit is installed as well as on the monitoring unit; it will be helpful to all miners present inside the mine to save their life before any casualty occurs. Alarm triggers when the value of sensor crosses the threshold level. This system additionally stores all the information in the computer for future inspection .It has the advantage of low price and low power consumption make it easy to implement with high speed nevertheless ,many interesting application are remaining for future research.

REFERENCES

1. Vikram.N, Harish K.S, Nihaal M.S,RakshaUmesh4, Shetty Aashik Ashok Kumar, 'A Low Cost Home Automation System Using Wi-Fi Based Wireless Sensor Network Incorporating Internetof Things(IoT)'2017 IEEE 7th International Advance Computing Conference.
2. Praveen Kumar, Umesh Chandra Pati ' IoT Based Monitoring and Control of Appliances for Smart Home' IEEE International Conference On Recent Trends In Electronics Information Communication Technology, May 20-21, 2016,India.
3. Y. W. Prakash, Vishakha Biradar, Shenil Vincent, Minto Martin and Anita Jadhav"Smart Bluetooth Low Energy Security system" IEEE WiSPNET 2017 conference.
4. Eleni Isa, Nicolas Sklavos 'Smart Home Automation: GSM Security System Design &Implementation' ResearchGate Conference Paper · January2017.
5. Vaibhav Sharma, Chirag Fatnani , Pranjal katara,Vishnu shankar , 'AdvancedLow-cost Security System Using Sensors, Arduino and GSM CommunicationModule'Proceedings of IEEE TechSym 2014 Satellite Conference, VIT University, 7th- 8th March.
6. K. Page, "Blood on the coal: The effect of organizational size and differentiationon coal mine accidents," *J. Safety Res.*, vol. 40, no. 2, pp.85–95, 2009.
7. L. Mallet, C. Vaught, and M. J. Brnich Jr., "Sociotechnical communication in anunderground mine fire: A study of warning messages duringan emergency evacuation," *Safety Sci.*, vol. 16, no. 5, pp. 709–728,1993.

8. M. Ndoh and G. Y. Delisle, “Underground mines wireless propagationmodeling,” in *Proc. 60th IEEE Veh. Technol. Conf.*, 2004, vol. 5, pp.3584–3588.