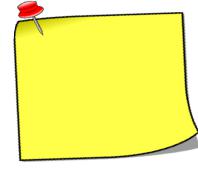
**CHAPTER 1**

**INTRODUCTION**

Notice Boards are an important medium for displaying information and keeping people aware. The traditional notice boards involve the pinning up of printed or handwritten information on a board. The digital electronics notice board is speedy alternative than the ordinary type of notice board. This idea can display multiple notices at a time to the number of users. This idea with an aim to increase the usability of electronic notice boards, deals with wireless reception and display of notices using raspberry-pi. Notice-board is a thing that can be used in multiple places like any institution or public utility places.



***Circular***

***It is to inform that tomorrow\****

***is observed as a holiday on***

***account of Sunday.***

Fig.1.1 A notice of daily routine

The way to digitize the notice board is necessary because traditional notice-board requires, separate person for pinning or sticking notices on the board and wastage of paper printer-ink etc. The internet is used for data transmission that sends notices. We are using raspberry-pi (model B) as transmitter, by using raspberry-pi we can make notices at any time and transmit over internet at receiving end after establishing connection between the transmitter (raspberry-pi) and receiver (LCD screen) by providing IP address, then we are able to send the notices remotely from anywhere in the world.

**CHAPTER-2**

**LITERATURE SURVEY**

**Paper- 1:** “An IoT Based Web Page Controlled Notice Board”, 2017- InternationalJournal of Innovative Technology and Research (IJITR) .

**Author:** P.Bhaskara Chary, M.Tech Student, Dept. Of ECE, KU College ofEngineering & Technology, Warangal, T.S, India.

**Description:**

“GSM is an advanced versatile communication framework, which is internationally gotten to by nearly 212 nations and domains. Worldwide framework for versatile work is totally upgraded for full duplex voice communication. At first produced for the substitution of real (1G) innovation, now GSM is accessible with heaps of hitting highlights with the consistent up degree of third era (3G) innovation Likewise, in trains and transports the data like stage number, ticket data is shown in computerized loads up. Individuals are presently adjusted to the possibility of the world readily available. The utilization cell phones have expanded definitely finished years. Control and correspondence has turned out to be imperative in every one of the parts of the world. This undertaking is a Web Controlled notice board with a GSM modem at the recipient’s end. So if the client needs to show any message, he can send the data by Web server (Thingspeak.com) and hence refresh the LCD show appropriately.

This Undertaking portrays the framework that message send from approved client to GSM module which is situated on the notice board. So this GSM module gets the message and showed on see load up, at same time this message will be send distinctive portable number store in memory of microcontroller. At the point when new message is landed at see board than the signal will beep. Max232 move the level of flag which changes over the flag between the microcontroller and GSM module. After the transformation of flag this message will be shown on see board

In this proposed system the idea of IoT based web controlled digital notice board using GSM technology has been presented. So the main aim is to reduce paper work and time. At present, when information has to be updated in a notice board, it has to be done manually. To change message on display, it needs to change microcontroller program code again. By adding web controlled IoT based communication interface to this system, we can make smart notice board to overcome these limitations. So we have interfaced web controlled IoT based SIM800L modem with microcontroller and implemented a text transmission and reception technique.”

**Algorithm:**

Step 1: Allocate storage space in Thingspeak.com server

Step 2: Create html page to enter text/message/data to be displayed on the notice board.

Step 3: Activate GSM SIM800L module by dumping AT commands using flash magic.

Step 4: Send the message/text/data using html page.

Step 5: SIM800L module will receives the data from the server and sends to microcontroller. Step 6: The microcontroller processes and stores the data in memory which will be displayed in LCD 20X4 display.

**Paper- 2**: “Digital Notice Board Using Raspberry Pi”, February 2016- InternationalJournal of Computing and Technology (IJCAT).

**Author:** Jadhav Vinod.

**Description**:

”Home automation includes all electronic components, building physics, middle ware, informatics technology and telecommunications used in buildings, more or less "interoperable" and to centralize the control of different systems and subsystems of the house and company (heating, shutters, garage door, entrance gate, electrical outlets, etc.. Home automation has been developed to provide technical solutions to meet the comfort needs such that energy management, optimization of lighting and heating. Home automation could be improved through an amelioration of communication network that uses a pair of twisted lines, radio signals, or fibre optics in a bus based network or an internet protocol as standards.

Design and Implementation of Digital notice board by using raspberry pi board. The application has been installed on a Smartphone, a web server and a raspberry pi card to display text on display device. The main objective of this system is to develop a wireless notice board that display message sent from the user and to design a simple, easy to install, user friendly system, which can receive and display notice in a particular manner with respect to date and time which will help the user to easily keep the track of notice board every day and each time he uses the system.

Notice Board is the primary thing in any institution or public utility places like bus stations, railway stations, colleges, malls, etc. But sticking various notice day to day is a difficult process. A separate person is required to take care of this notices display. This project is about innovative wireless notice board. A Wi-Fi is used for sharing Data. At any time we can add or re- move or alter the information (notices) according to our requirement. Legal PC is used for sending notices at the side of the transmitter. And Wi-Fi is used for sharing data at the receiver side. When an authorized user sends a notice from his system, it is received by the receiver. Wireless is a well-known technology that allows an electronic device to share data wirelessly over a computer network, including high speed wireless connections.”

The main goal of this project is to provide Exciting new innovative way to send information to your Staff visitors and Students - The main objective of this system is to develop a wireless notice board that display message sent from the user and to design a simple, easy to install, user friendly system, which can receive and display notice in a particular manner with respect to date and time which will help the user to easily keep the track of notice board every day and each time he uses the system. Wi-Fi is the wireless technology used.

**CHAPTER-3**

**SYSTEM ANALYSIS**

**3.1 REQUIREMENT ANALYSIS**

***3.1.1 FUNCTIONAL REQUIREMENTS***

A functional requirement document defines the functionality of a system or one of its subsystems. It also depends upon the type of software, expected users and the type of system where the software is used. Functional user requirements may be high-level statements of what the system should do but functional system requirements should also describe clearly about the system services in detail scope

A functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behaviour and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioural requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements specify particular results of a system. Functional requirements drive the application architecture of a system.

* **Functional Requirements 1.1**

**ID :** FR1

**TITLE :** Download mobile application

**DESC :** A user should be able to download the mobile applicationthrough either an application store or similar service on the Mobile phone. The application should be free to download.

**RAT :** In order for a user to download the mobile application

**DEP :** None

* **Functional Requirements 1.2**

**ID :** FR2

**TITLE :** Download and notify users of new releases

**DESC :** when a new/upload version of the software is Released, the user should check for these manually, the download of the new release should be done through mobile phone in the same way as downloading in mobile application

**RAT :** In order for a user to download a new/updated release.

**DEP :** FR1

* **Functional Requirement 1.3**

**ID :** FR3

**TITLE :** Mobile application – Send Notice

**DESC :**

* Notice will be stored in firebase.
* Through raspberry pi process the notice and display on lcd.

**RAT :** Notice viewed by user .

**DEP :** FR2

* **Functional Requirement 1.4**

**ID :** FR4

**TITLE :** Mobile application – Clear Notice

**DESC :**

* Empty Text will be stored in firebase.
* Through raspberry pi process the notice and display on lcd.

**RAT :** EmptyNotice viewed by user .

**DEP :** FR2

***3.1.2 NON FUNCTIONAL REQUIREMENTS***

**PERFORMANCE REQUIREMENTS**

* The performance exhibit with less amount of time.
* It handles the textual contents of the notices.
* The system would exhibit high performance and well maintain under the proper guidance.
* It requires the wifi connection to send or receive notices.
* The users must get the response within seconds i.e. the response time of a particular function should be minimum.

**SECURITY REQUIREMENTS**

Since the devices are connected within home network a basic firewall provided by the Raspbian OS is enough.

**MAINTAINABILITY REQUIREMENTS**

After the deployment of the project if any error occurs then it can be easily maintained by the software developer. The requirements modules that are explained in this document are enough to satisfy the customer’s needs and wants in case of change or addition demand after completing the system or in development process of the system.

**AVAILABILITY REQUIREMENT**

Since the devices are connected with in home network through Router they are available 24x7.

**SAFETY REQUIREMENTS**

* Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product.
* Defining any safeguards or actions that must be taken, as well as actions that must be prevented.

**3.2 SOFTWARE REQUIREMENTS**

**RASPBIAN OS**

Raspbian is a Debian-based computer operating system for Raspberry Pi. There are several versions of Raspbian including Raspbian Stretch and Raspbian Jessie. Since 2015 it has been officially provided by the Raspberry Pi Foundation as the primary operating system for the family of Raspberry Pi single-board computers.Raspbian was created by Mike Thompson and Peter Green as an independent project. The initial build was completed in June 2012. The operating system is still under active development. Raspbian is highly optimized for the Raspberry Pi line's low-performance ARM CPUs.

**Firebase Database**

The Firebase Real time Database lets you build rich, collaborative applications by allowing secure access to the database directly from client-side code. Data is persisted locally, and even while offline, real-time events continue to fire, giving the end user a responsive experience. When the device regains connection, the Real time Database synchronizes the local data changes with the remote updates that occurred while the client was offline, merging any conflicts automatically.

The Real time Database provides a flexible, expression-based rules language, called Firebase Real time Database Security Rules, to define how your data should be structured and when data can be read from or written to. When integrated with Firebase Authentication, developers can define who has access to what data, and how they can access it.

The Real time Database is a no SQL database and as such has different optimizations and functionality compared to a relational database. This enables you to build a great real-time experience that can serve millions of users without compromising on responsiveness. Because of this, it is important to think about how users need to access your data and then structure it accordingly.

**3.3 HARDWARE REQUIREMENTS**

**Raspberry PI 3**

Raspberry pi is heart of this project. It receives messages from an android application and send to LED to display Notice.



Fig: 3.1 Raspberry pi 3

**Bread Board**

This makes it easy to use for creating temporary prototypes and experimenting with circuit design

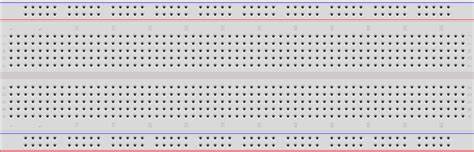


Fig: 3.2 Bread Board

**Jumper Wires**

For connecting devices we use these jump wires



Fig: 3.3 Jumper Wires

**LCD**

To display the Notice we use the LCD Display

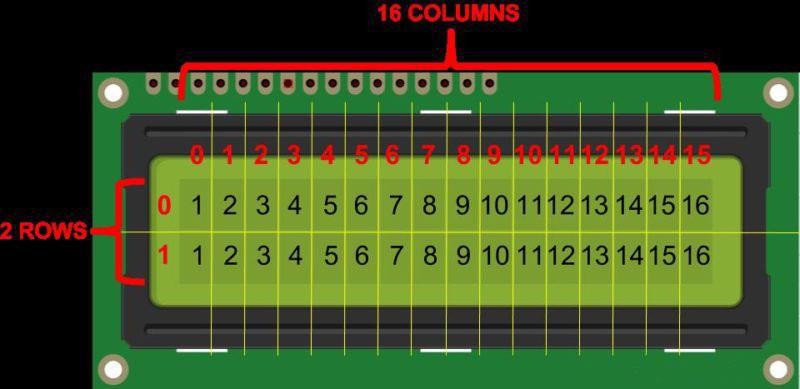


Fig: 3.4 LED

**USB Cable**

Usb cable is used power supply to raspberry pi



Fig: 3.5 USB Cable

**3.4 PROCESS MODEL**

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the spiral.

The Spiral Model is widely used in the software industry as it is in sync with the natural development process of any product, i.e. learning with maturity which involves minimum risk for the customer as well as the development firms.

The following pointers explain the typical uses of a Spiral Model −

* When there is a budget constraint and risk evaluation is important.
* For medium to high-risk projects.
* Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
* Customer is not sure of their requirements which is usually the case.
* Requirements are complex and need evaluation to get clarity.
* New product line which should be released in phases to get enough customer feedback.
* Significant changes are expected in the product during the development cycle.

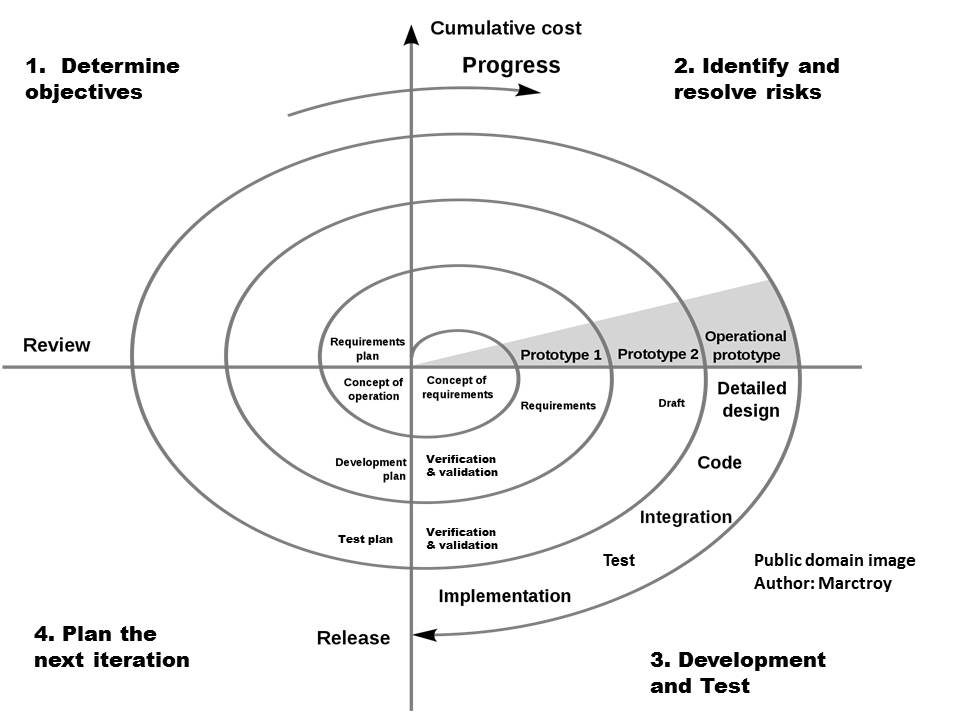


Fig: 3.6 Spiral Model

**CHAPTER-4**

**SYSTEM DESIGN**

**4.1 UML DIAGRAMS**

The Unified Modeling Language (UML) is used to specify, visualize, modify, construct and documents the artifacts of an object-oriented software intensive system under development. UML offers a standard way to visualize a system’s architectural blueprints inlcuding elements such as

* Actors
* Business processes
* Components
* Activities
* Programming language statements
* Database schemes
* Reusable software components

The UML represents the culmination of best pratical object oriented modeling. The UML is the product of several years of hardworking which we focused on bringing about a unification of the methods used around the world.

***4.1.1 USE CASE DIAGRAM***

The Use Case Diagram is a graphic depiction of the interactions among the elements of DigitalNoticeBoard System. It represents the methodology used in system analysis to identify, clarify and organize system requirements of the system.

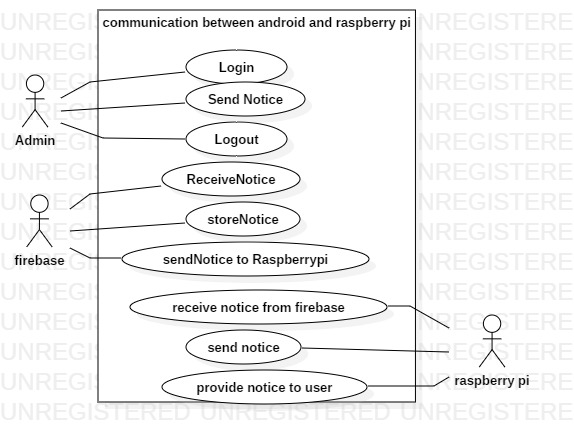
****

Fig: 4.1 Use case diagram for Digital Notice Board

**Table 4.1: Actores & usecases diagrams in usecase diagram for device and app**

|  |  |
| --- | --- |
| **ACTOR** | **USECASE** |
| Admin | Login  Send Notice  Clear Notice  Logout |
| Firebase | Receive Notice  store Notice  Send Notice to raspberry pi |
| Raspberry pi | Receive Notice From Firebase  Send notice  Provide Notice to user |

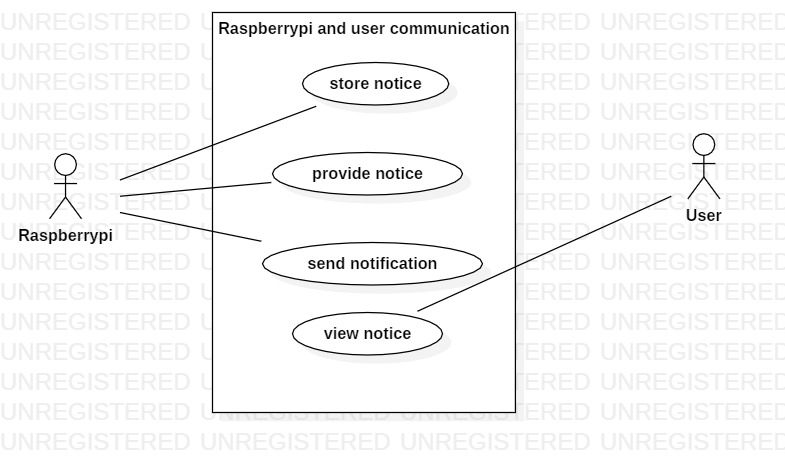
****

Fig: 4.2 Use case diagram for Digital Notice Board in the user scenario

**Table 4.2: Actores & usecases diagrams in usecase diagram for user and device**

|  |  |
| --- | --- |
| **ACTORS** | **USE CASES** |
| Raspberry pi | Store notice  Provide notice  Send notification |
| User | View notice |

***4.1.2 SEQUENCE DIAGRAM***

Sequence diagram also known as interaction diagram depicts how elements interact over time.

A horizontal axis shows elements involved in the interaction and the vertical axis represents time proceeding down the page. The sequence diagram has following types of elements.

* Classes and objects
* A lifeline
* A communication between objects

**Classes and objects**

Classes are so much the same way as on class diagram. Objects may also be shown much the same way as an object diagram.

**Lifeline**

Shown as an vertically dashed life from an element. This represents the existence of the element over time.

**Communication**

Shown as a horizontal solid arrow from the lifeline of the sender to the lifeline of the receiver and labelled with the name of the operation to be invoked. This represents that sender sends a message or stimulus to the receiver.

The UML Sequence Diagram of DigitalNoticeBoard System shows the interaction between the objects of Faculty, LCD. The diagram shows how the admin will be able to login to their accounts using their login credentials. After login user can manage all the operations. The diagram below demonstrates how the login page works.

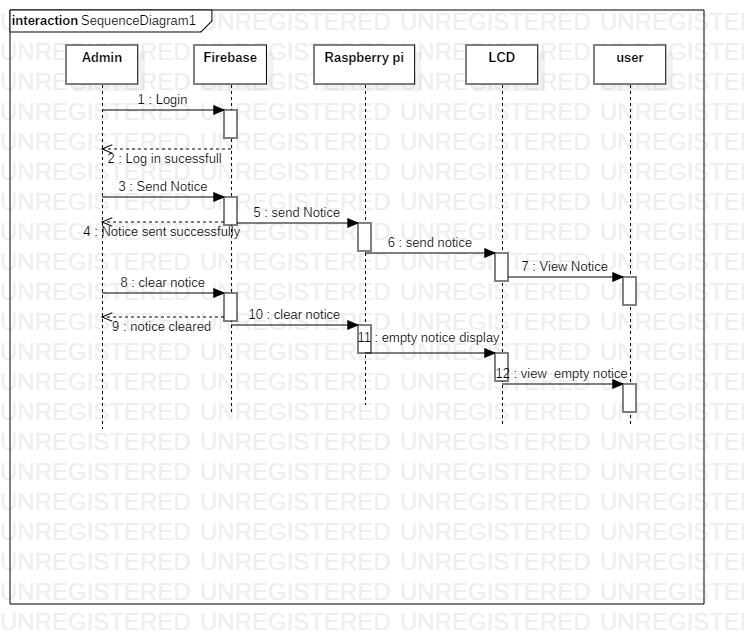
****

Fig: 4.3 Sequence Diagram for Digital Notice Board(SSID)

***4.1.3 Activity Diagram***

Activity diagrams are graphical representations of workflows of stepwise activities actions with support for choice, iteration and concurrency. Activity diagrams show the overall flow of control.

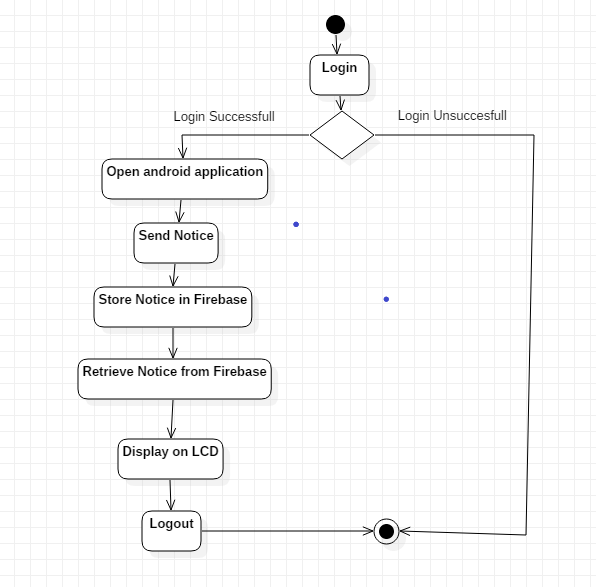
****

Fig: 4.4 Activity Diagram for displaying notice on the Digital Notice Board

**4.2 EQUIPMENT DESIGN**

***4.2.1 Raspberry Pi***

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV and used a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing and to learn how to program in languages like Scratch and Python. It is capable of doing everything that we expect a desktop computer do, from browsing internet and playing high-definition video, to making spreadsheets, word-processing and playing games.

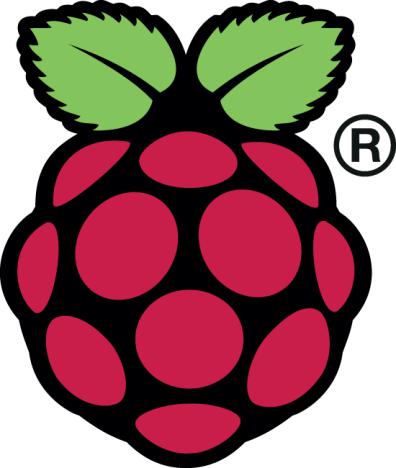


Fig. 4.5 Raspberry Pi registered logo

The Raspberry Pi is a series of small [single-board computers](https://en.wikipedia.org/wiki/Single-board_computer) developed in the United Kingdom by the [Raspberry Pi Foundation](https://en.wikipedia.org/wiki/Raspberry_Pi_Foundation) to promote the teaching of basic [computer science](https://en.wikipedia.org/wiki/Computer_science) in schools and in [developing countries.](https://en.wikipedia.org/wiki/Developing_countries) The original model became far more popular than anticipated, selling outside its [target market](https://en.wikipedia.org/wiki/Target_market) for uses such as [robotics.](https://en.wikipedia.org/wiki/Robotics) It does not include peripherals (such as [keyboards,](https://en.wikipedia.org/wiki/Keyboard_(computing)) [mice](https://en.wikipedia.org/wiki/Mouse_(computing)) and [cases)](https://en.wikipedia.org/wiki/Computer_case). However, some accessories have been included in several official and unofficial bundles. According to the Raspberry Pi Foundation, over 5 million Raspberry Pi’s were sold by February 2015, making it the best-selling [British computer.](https://en.wikipedia.org/wiki/British_computer)

**Models of Raspberry Pi**

The different types of raspberry pi models are following

* Raspberry pi 1 model B
* Raspberry pi 1model A
* Raspberry pi 1 model B+
* Raspberry pi 1 model A+
* Raspberry pi zero
* Raspberry pi 2
* Raspberry pi 3 model B
* Raspberry pi zero W

**Raspberry Pi 3 Model**

Comparing to Pi 2, there are two giant upgrades in the Pi 3. The first is a next generation Quad Core Broadcom BCM2837 64-bit ARMv8 processor, making the processor speed increase from 900 MHz on the Pi 2 to up to 1.2GHz on the Pi 3.

The second giant upgrade (and this is the one we’re personally most excited about) is the addition of a BCM43438 Wi-Fi chip BUILT-IN to your Raspberry Pi. The Pi is Wi-Fi ready. There’s also Bluetooth Low Energy (BLE) on board making the Pi an excellent IoT solution (BLE support is still in the works, software-wise) lastly, there’s an upgraded switched power source that goes up to 2.5 Amps instead of just 2 Amps - allowing your Pi to power even more powerful devices over USB ports.

[The power supply, 5V & 2 .4 A works great with the Pi 3 Model B and](https://www.adafruit.com/product/1995) [provides plenty of power to the chip, Ethernet, and any USB add-ons you plug](https://www.adafruit.com/product/1995) [in.](https://www.adafruit.com/product/1995) The best part about all this is that the Pi 3 keeps the same shape, connectors, and mounting holes as the Pi 2.



Fig: 4.6 Raspberry pi model B

**Technical Specifications**

**1**. Broadcom BCM2837 64 bit Quad Core Processor powered Single Board Computer running at 1.2GHz 1GB RAM

**2.** BCM43438 Wi-Fi on board

**3.** Bluetooth Low Energy (BLE) v4.1 on board

**4.** 40 pin extended GPIO

**5.** 4 x USB 2 ports

**6.** 4 pole Stereo output and Composite video port

**7.** Full size HDMI

**8.** CSI camera port for connecting the Raspberry Pi camera

**9.** DSI display port for connecting the Raspberry Pi touch screen display

**10.** Micro SD port for loading your operating system and storing data Upgraded switched

**11.** Micro USB power source (now supports up to 2.5 Amps)

**Antenna**

There’s no need to connect an external antenna to the Raspberry Pi 3. Its radios are connected to this chip antenna soldered directly to the board, in order to keep the size of the device to a minimum. Despite its diminutive stature, this antenna should be more than capable of picking up wireless LAN and Bluetooth signals – even through walls.

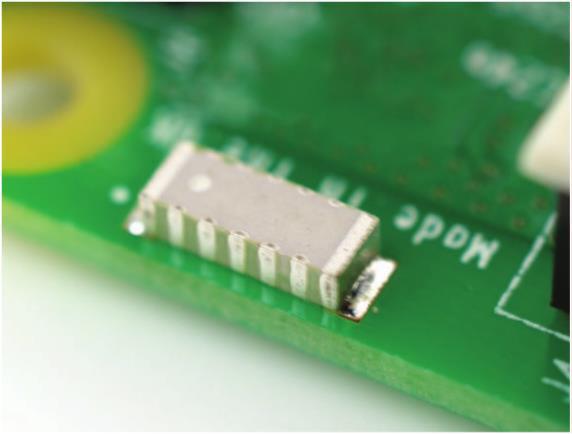


Fig.4.7 Raspberry Pi Antenna

**System on Chip (SoC)**



Fig.4.8 Raspberry Pi SoC

Built specifically for the new Pi 3, the Broadcom BCM2837 system-on-chip (SoC) includes four high-performance ARM Cortex-A53 processing cores running at 1.2GHz with 32kB Level 1 and 512kB Level 2 cache memory, a Video Core IV graphics processor, and is linked to a 1GB LPDDR2 memory module on the rear of the board.

**GPIO**

The Raspberry Pi 3 features the same 40-pin general-purpose input-output (GPIO) header as all the Pi’s going back to the Model B+ and Model A+. Any existing GPIO hardware will work without modification; the only change is a switch to which UART is exposed on the GPIO’s pins, but that’s handled internally by the operating system.

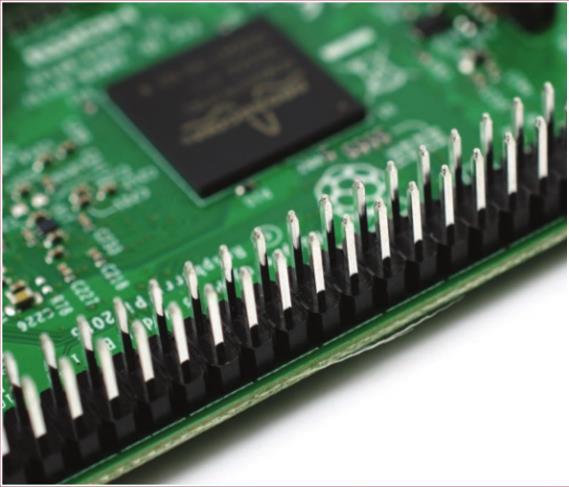


Fig. 4.9 Raspberry Pi GPIO Header

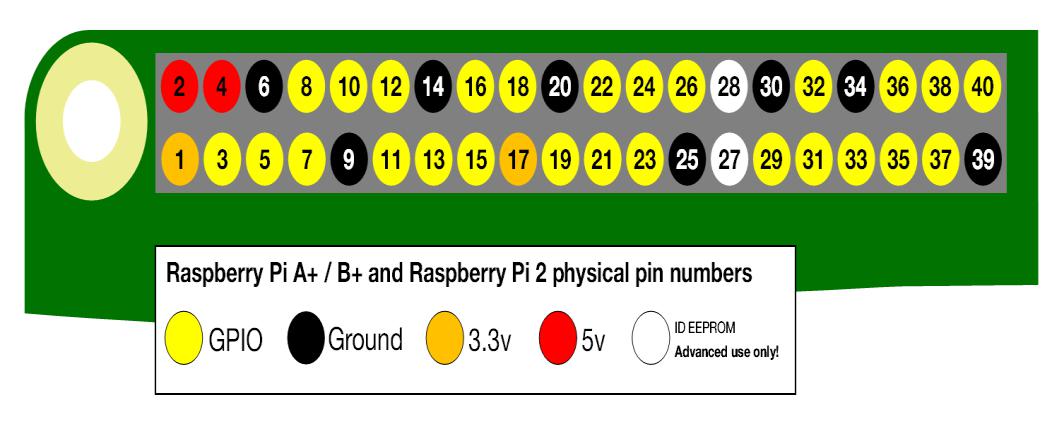


Fig.4.10 Raspberry Pi GPIO layout

**USB Chip**

The Raspberry Pi 3 shares the same SMSC LAN9514 chip as its predecessor, the Raspberry Pi 2, adding 10/100 Ethernet connectivity and four USB channels to the board. As before, the SMSC chip connects to the SoC via a single USB channel, acting as a USB-to- Ethernet adaptor and USB hub.

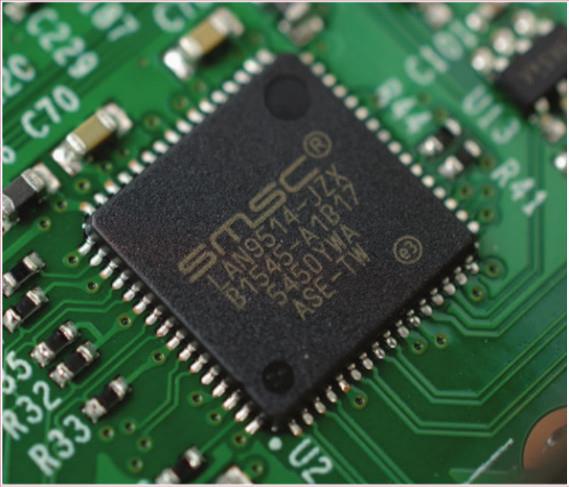


Fig.4.11 Raspberry Pi USB Chip

**Monitors**

For regular use, you'll want to plug the Raspberry Pi in to a visual display: a monitor or a TV.

**HDMI Port**

The Raspberry Pi has an HDMI port which you can connect directly to a monitor or TV with an HDMI cable. This is the easiest solution; some modern monitors and TVs have HDMI ports, some do not, but there are other options.



Fig: 4.12 HDMI port

**DVI**

For monitors with a DVI port, we can use an HDMI-to-DVI cable or an HDMI cable with a DVI adapter. But the DVI standard does not support audio.



Fig. 4.13 HDMI to DVI connector

**VGA**

For monitors with VGA only, we can use an HDMI-to-VGA adapter. Note that VGA does not support audio.



Fig. 4.14 HDMI to VGA connector

**Composite Port**

The original Raspberry Pi used an RCA connector and a standard RCA composite video lead will work. Others models (Raspberry Pi B+ and later) combine the audio out and composite out on to the same 3.5mm jack plug. This requires a particular type of lead, with audio left on the tip, audio right on ring 1, ground on ring 2, and video on the sleeve. This is the same as leads used on the Zune and on apple devices



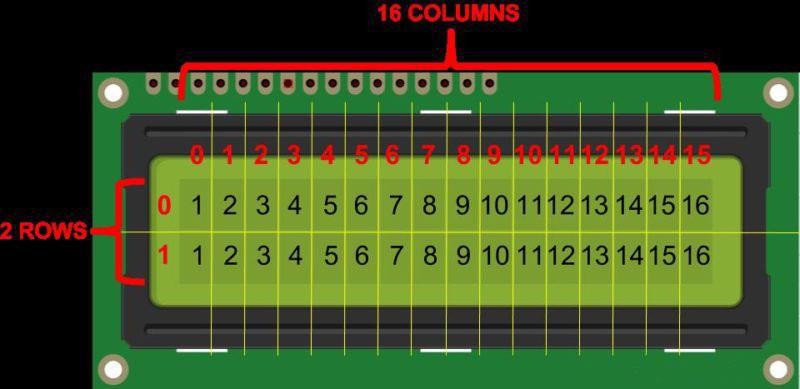
Fig. 4.15 RCA connector

**Raspberry Pi Operating System**

Raspbian is a [Debian](https://en.wikipedia.org/wiki/Debian)-based [computer operating system](https://en.wikipedia.org/wiki/Operating_system) for [Raspberry Pi.](https://en.wikipedia.org/wiki/Raspberry_Pi) There are several versions of Raspbian including Raspbian Stretch and Raspbian Jessie. Since 2015 it has been officially provided by the [Raspberry Pi](https://en.wikipedia.org/wiki/Raspberry_Pi_Foundation) [Foundation](https://en.wikipedia.org/wiki/Raspberry_Pi_Foundation) as the primary operating system for the family of Raspberry Pi [single-board computers.](https://en.wikipedia.org/wiki/Single-board_computers) Raspbian was created by Mike Thompson and Peter Green as an independent project. The initial build was completed in June 2012. The operating system is still under active development. Raspbian is highly optimized for the Raspberry Pi line's low-performance [ARM](https://en.wikipedia.org/wiki/ARM_architecture) CPU’s.

Raspbian comes pre-installed with plenty of software for education, programming and general use. It has Python, Scratch, Sonic Pi, Java, Mathematical and more. The Raspbian with Desktop image contained in the ZIP archive is over 4GB in size, which means that these archives use features which are not supported by older unzip tools on some platforms. If we find that the download appears to be corrupt or the file is not unzipping correctly, please try using [7-Zip](http://www.7-zip.org/download.html) (Windows) or [The Un archive](http://wakaba.c3.cx/s/apps/unarchiver.html) (Macintosh). Both are free of charge and have been tested to unzip the image correctly

**4.2.2 Liquid Crystal Display (LCD)**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Fig.4.16 16X2 LCD Display | | | |
|  |  |  |  |  |

Liquid crystals are do not emit directly, instead of using a back light or reflector to produce images in colour or monochromatic. LCDs are available to display arbitrary images or fixed images with low information content, which can be displayed or hidden, such as preset words, digits, and [7-segment](https://en.wikipedia.org/wiki/7-segment) displays, as in a [digital clock.](https://en.wikipedia.org/wiki/Digital_clock) They use the same basic technology, except that arbitrary images are made up of a large number of small [pixels,](https://en.wikipedia.org/wiki/Pixel) while other displays have larger elements.

**Table 4.3: LCD 16X2 pin layout**

|  |  |  |
| --- | --- | --- |
| **Pin Number** | **Symbol** | **Function** |
|  |  |  |
| 1 | Vss | GND |
|  |  |  |
| 2 | Vdd | +3V or +5V |
|  |  |  |
| 3 | Vo | Contrast Adjustment |
|  |  |  |
| 4 | RS | H/L Register Select Signal |
|  |  |  |
| 5 | R/W | H/L Read/Write Signal |
|  |  |  |
| 6 | E | H → L Enable Signal |
|  |  |  |
| 7 | DB0 | H/L Data Bus Line |
|  |  |  |
| 8 | DB1 | H/L Data Bus Line |
|  |  |  |
| 9 | DB2 | H/L Data Bus Line |
|  |  |  |
| 10 | DB3 | H/L Data Bus Line |
|  |  |  |
| 11 | DB4 | H/L Data Bus Line |
|  |  |  |
| 12 | DB5 | H/L Data Bus Line |
|  |  |  |
| 13 | DB6 | H/L Data Bus Line |
|  |  |  |
| 14 | DB7 | H/L Data Bus Line |
|  |  |  |
| 15 | A/Vee | + 4.2V for LED/Negative Voltage Output |
|  |  |  |
| 16 | K | Power Supply for B/L (OV) |
|  |  |  |

**LCD Types**

The different types of LCD displays are given below:

**i) 20x4” Character LCD**

It is a 20 character by 4 line display. LCD Utilizes the extremely common HD44780 parallel interface chipset. We will need ~11 general I/O pins to interface to this LCD screen. It also includes LED backlight.



Fig.4.17 20x4” Character LCD

**ii) 7-Segment Serial LCD Display**

In addition to the ten digits, seven-segment displays can be used to show letters of the [Latin,](https://en.wikipedia.org/wiki/Latin_alphabet) [Cyrillic](https://en.wikipedia.org/wiki/Cyrillic_script) and [Greek alphabets](https://en.wikipedia.org/wiki/Greek_alphabet) including [punctuation,](https://en.wikipedia.org/wiki/Punctuation) but only very few representations are unambiguous and intuitive at the same time, and a number of letters in all three scripts have either too many vertical strokes or have diagonal strokes that cannot fit into the seven segments given. The Serial 7-Segment Display can be controlled in one of three ways: Serial TTL communication, SPI serial communication or I2C serial.

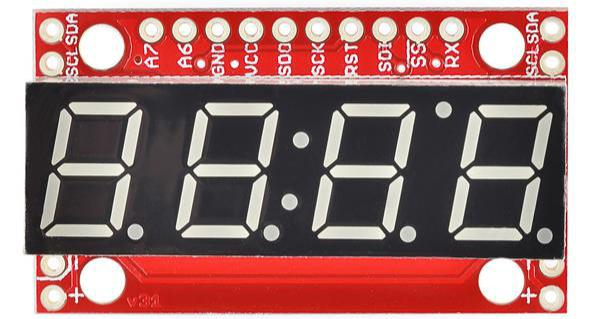


Fig.4.18 7- segment LCD Display

**iii) Dot Matrix Display**

A dot-matrix display is a [display device](https://en.wikipedia.org/wiki/Display_device) used to display information on machines, clocks, railway departure indicators and many other devices requiring a simple display device of limited resolution. The display consists of a [dot matrix](https://en.wikipedia.org/wiki/Dot_matrix) of lights or mechanical indicators arranged in a rectangular configuration (other shapes are also possible, although not common) such that by switching on or off selected lights, text or graphics can be displayed. A dot matrix controller converts instructions from a processor into signals which turns on or off lights in the matrix so that the required display is produced.



Fig.4.19 Dot Matrix Display

**Circuit Diagram**

Connections for the Wireless Message Board can be made by [connecting](http://circuitdigest.com/microcontroller-projects/raspberry-pi-lcd-display-tutorial) [LCD with the Raspberry Pi](http://circuitdigest.com/microcontroller-projects/raspberry-pi-lcd-display-tutorial) board by using some connectors over bread board. RS, RW and EN pins of LCD are directly connected to pin 18, GND and 23. And data pins of LCD D4, D5, D6, D7 are directly connected to Raspberry Pi’s GPIO 24, 16, 20, 21. A 10K pot is used to control the brightness of LCD.

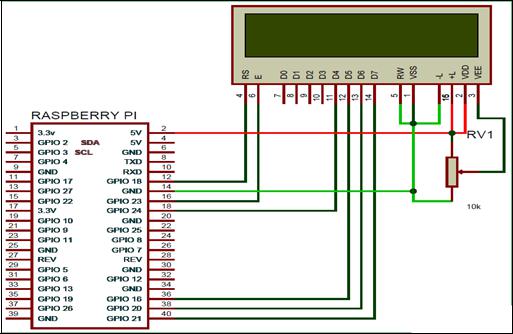


Fig.4.20 LCD Wiring Diagram with Raspberry Pi along with Potentiometer

**CHAPTER 5**

**IMPLEMENTATION**

**5.1 TECHNOLOGIES USED**

The various technologies used in DIGITAL NOTICE BOARD are as follows

* Java
* XML
* Android SDK
* Embedded Python

**JAVA**

Java was conceived by James Gosling, Patrick Naughton, Chris Warth, Ed Frank and Mike Sheridan at Sun Microsystems Inc. in 1991. This language was initially called “Oak” but was renamed as “Java” in 1995. This is mostly used programming language because of its various features. The various features of the java, which made java so popular, are as follows

* Simple
* Secure
* Portable
* Object-oriented
* Robust
* Multithreaded
* Architectural-neural
* Interpreted
* Distributed
* Dynamic

**Java Environment**

Java environment includes a large number of development tools and hundreds of classes and methods. The development tools are part of the system known as Java Development Kit (JDK), the classes and methods are part of the Application Programming Interface (API).

**Java Development Kit**

The Java Development Kit comes with a collection of tools that are used for developing and running Java programs. They include

* Applet viewer (for viewing Java applets)
* Javac (Java Compiler)
* Java (Java Interpreter)
* Javap (Java Dis assembler)
* Javah (for C header files)
* Javadoc (for creating HTML documents)
* Jdb (Java debugger)

**XML**

Extensible Markup Language (XML) is the predominant markup language for web pages. XML is designed to transport and store the data. XML tags are not predefined. We can define our own tags. With XML, data can be stored in separate XML files. This makes it much easier to create data that can be shared by different applications.

XML data is stored in text format. This makes it easier to expand or upgrade to new operating system, new applications, or new browsers without losing the data. The syntax of XML is very simple and logical. An element can contain text, attributes, other elements, or mixture of all the above.

**Introduction to Android**

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

The Android SDK includes a comprehensive set of development tools. Currently supported development platforms include x86-architecture computers running Linux (any modern desktop Linux distribution), Mac OS X 10.4.8 or later, Windows XP or Vista. The officially supported integrated development environment (IDE) is Eclipse (3.2 or later) using the Android Development Tools (ADT) Plug in, though developers may use any text editor to edit Java and XML files then use command line tools to create, build and debug Android applications.

**Android SDK**

The Android Software Development Kit (SDK) includes a comprehensive set of development tools. The Android SDK can be broken down into several components. These include

* Platform-tools
* Build-tools
* SDK-tools
* Android Debug Bridge (ADB)
* Android Emulator

## Android SQLite

Android SQLite is a very lightweight database which comes with Android OS. Android SQLite combines a clean SQL interface with a very small memory footprint and decent speed. For Android, SQLite is “baked into” the Android runtime, so every Android application can create its own SQLite databases.

Android SQLite native API is not JDBC, as JDBC might be too much overhead for a memory-limited smart phone. Once a database is created successfully its located in **data/data//databases/** accessible from Android Device Monitor.

SQLite is a typical **relational database**, containing tables (which consists of rows and columns), indexes etc. We can create our own tables to hold the data accordingly. This structure is referred to as a **schema**.

**EMBEDDED PYTHON**

Python might be at its strongest when used as a communication middleman between the user and the embedded system they're working with. Sending messages through Python to or from an embedded system allows the user to automate testing. Python scripts can put the system into different states, set configurations, and test all sorts of real-world use cases. Python can also be used to receive embedded system data that can be stored for analysis. Programmers can then use Python to develop parameters and other methods of analyzing that data.

Currently the main debate about the merits of Python and C/C++ comes down to what's more important to your team: development speed or runtime speed. In the future, though, it might not be up to Python programmers to make their case for its use in embedded systems, but rather for embedded systems designers to figure out how to accommodate the relentlessly increasing popularity of Python

**-**

**5.2 SAMPLE CODE**

**AndroidMainifest.xml**

<?xml version="1.0" encoding="utf-8"?>

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

xmlns:app="http://schemas.android.com/apk/res-auto"

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

android:layout\_width="match\_parent"

android:orientation="vertical"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<EditText

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingTop="30dp"

android:layout\_marginTop="150dp"

android:layout\_gravity="center"

android:id="@+id/notice"

android:textSize="25dp"

android:hint="Enter notice to sent"/>

<Button

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="60dp"

android:layout\_marginTop="20dp"

android:padding="20dp"

android:layout\_marginRight="60dp"

android:text="send Notice"

android:id="@+id/submit"

android:onClick="onSubmit"/>

<Button

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="60dp"

android:layout\_marginTop="20dp"

android:padding="20dp"

android:layout\_marginRight="60dp"

android:text="Clear Notice"

android:id="@+id/submit2"

android:onClick="onSubmit2"/>

</LinearLayout>

**MainActity.java**

package com.noticeboard.example.pavan.digitalnotice;

import android.support.annotation.NonNull;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;

import com.google.android.gms.tasks.Task;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

public class MainActivity extends AppCompatActivity {

EditText mNotice;

Button mSubmit;

DatabaseReference mDatabase;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mNotice = findViewById(R.id.notice);

mSubmit = findViewById(R.id.submit);

mDatabase = FirebaseDatabase.getInstance().getReference();

}

public void onSubmit(View view) {

String str = mNotice.getText().toString();

mDatabase.child("Notice").setValue(str).addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if(task.isComplete())

Toast.makeText(MainActivity.this,"Notice sent is "+mNotice.getText().toString(), Toast.LENGTH\_SHORT).show();

Else

Toast.makeText(MainActivity.this, "failed to sent", Toast.LENGTH\_SHORT).show();

}

});

}

public void onSubmit2(View view) {

mDatabase.child("Notice").setValue("").addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if(task.isComplete())

Toast.makeText(MainActivity.this,"Notice Cleared Sucessfullly "+mNotice.getText().toString(), Toast.LENGTH\_SHORT).show();

Else

Toast.makeText(MainActivity.this, "Notice not Cleared", Toast.LENGTH\_SHORT).show();

}

});

}

}

**Firebase.py**

#import Libraries

import RPi.GPIO as GPIO

import time

from pyrebase import pyrebase

import LiquidCrystalPi

#Firebase Configuration

config = {

"apiKey": "AIzaSyACRUoCQQ9TQSCblg8NahYOrQH-CjK1Ycs",

"authDomain":"digitalnoticeboardproject.firebaseapp.com",

"databaseURL": "https://digitalnoticeboardproject.firebaseio.com",

"projectId": "digitalnoticeboardproject",

"storageBucket": "digitalnoticeboardproject.appspot.com",

"messagingSenderId": "200473268318"

}

firebase = pyrebase.initialize\_app(config)

GPIO.setmode(GPIO.BOARD)

GPIO.setwarnings(False)

LCD = LiquidCrystalPi.LCD(29, 31, 33, 35, 37, 38)

LCD.begin(16,2)

time.sleep(0.5)

LCD.write(" VVIT ")

LCD.nextline()

time.sleep(0.5)

LCD.write(" D-Notice Board")

#Firebase Database Intialization

db = firebase.database()

#While loop to run until user kills program

while(True):

notice = db.child("Notice").get()

str = notice.val()

if(len(str)==0):

LCD.clear()

LCD.write(" VVIT ")

LCD.nextline()

LCD.write(" D-Notice Board")

elif(len(str)<=16):

LCD.clear()

LCD.write(str)

else:

str1 ="".join(str[:16])

str2 ="".join(str[16:])

LCD.clear()

LCD.write(str1)

LCD.nextline()

LCD.write(str2)

**5.3 SCREEN SHOTS**

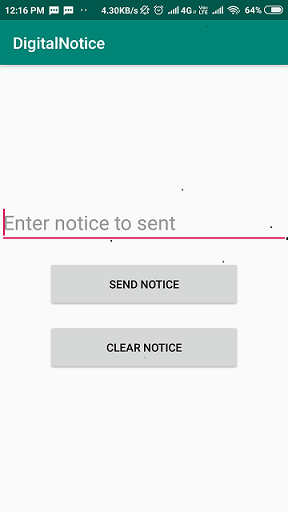


Fig: 5.1 Home Page



Fig:5.2 Output

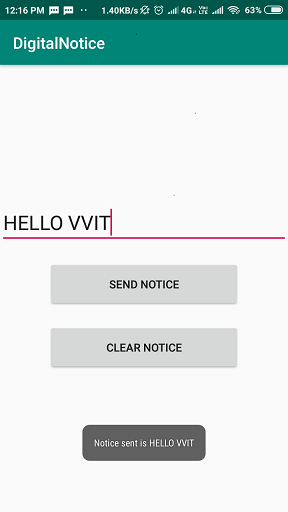


Fig: 5.3 Send Notice



Fig: 5.4 Send notice output

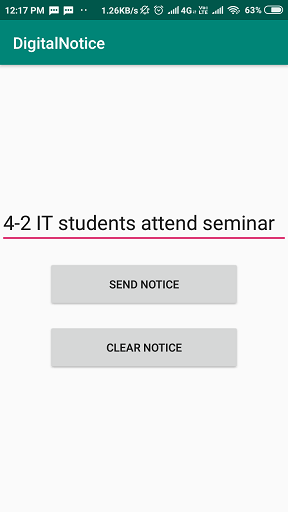


Fig:5.5 Send Notice Eaxmple2



Fig: 5.6 Send Notice eaxmple2 Output

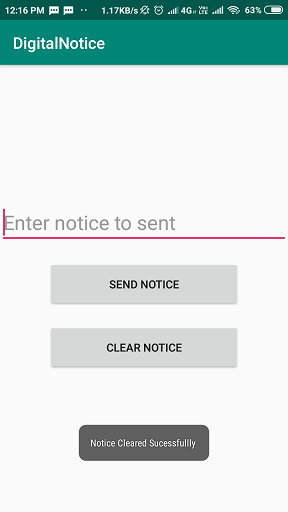


Fig:5.7 Clear Notice



Fig: 5.8 Clear Notice output

**CHAPTER 6**

**TESTING**

**6.1 PURPOSE OF TESTING**

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, testing is a process of executing a program with the intent of finding an error.

* A successful test is one that uncovers an as yet undiscovered error.
* A good test case is one that has a high probability of finding error, if it exists.
* The tests are inadequate to detect possibly present errors.
* The software more or less confirms to the quality and reliable stationards.

**6.2 TESTING STRATEGIES**

In order to uncover the errors, present in different phases we have the concept of levels of testing.

Types of Testing

* Unit Testing
* System Testing
* User input Validation Testing

***6.2.1 UNIT TESTING***

Unit testing focuses verification effort on the smallest unit of software i.e., the module. Using the detailed design and the process specifications testing is done to uncover errors within the boundary of the module. All modules must be successful in the unit test before the start of the integration testing begins.

In Digital Notice Board it application, each activity is tested with unit testing. It has been seen that each activity class runs fine after its development using unit testing.

***6.2.2 SYSTEM TESTING***

Here the entire software system of Find It application is tested. The reference document for this process is the requirements document and the goal is to see if software meets its requirement.

***6.2.3 USER INPUT VALIDATION TESTING***

The user input must be validated to confirm to expected values. The fields should also not be empty.

**6.3 TEST CASES**

**Test case 1**

**Table 6.1 Test case1**

|  |  |  |
| --- | --- | --- |
| **1.** | **Test case ID** | **String length = 0** |
| **2.** | **Precondition** | Notice length equals to zero |
| **3.** | **Description** | LCD will be cleared |
| **4.** | **Test Steps** | Click on SEND NOTICE icon |
| **5.** | **Expected Output** | VVIT  D-Notice Board |
| **6.** | **Actual Output** | VVIT  D-Notice Board |
| **7.** | **Status** | Pass |
| **8.** | **Remarks** |  |

**TEST CASE 2**

**Table 6.2 Test case2**

|  |  |  |
| --- | --- | --- |
| **1.** | **Test case ID** | **String length less than < = 16** |
| **2.** | **Precondition** | Enter notice |
| **3.** | **Description** | Display the notice on LCD |
| **4.** | **Test Steps** | Click on SEND NOTICE icon |
| **5.** | **Expected Output** | Enter notice will be display on LCD |
| **6.** | **Actual Output** | Enter Notice displayed successfully |
| **7.** | **Status** | Pass |
| **8.** | **Remarks** |  |

**TEST CASE 3**

**Table 6.3 Test case3**

|  |  |  |
| --- | --- | --- |
| **1.** | **Test case ID** | **String length less than > 16** |
| **2.** | **Precondition** | Enter notice |
| **3.** | **Description** | Display the notice on LCD |
| **4.** | **Test Steps** | Click on SEND NOTICE icon |
| **5.** | **Expected Output** | Enter notice will be display on LCD |
| **6.** | **Actual Output** | Enter Notice displayed successfully |
| **7.** | **Status** | Pass |
| **8.** | **Remarks** |  |

**Test case 4**

**Table 6.4 Test case4**

|  |  |  |
| --- | --- | --- |
| **1.** | **Test case ID** | **Clear Notice** |
| **2.** | **Precondition** | Notice length equals to zero |
| **3.** | **Description** | LCD will be cleared |
| **4.** | **Test Steps** | Click on SEND NOTICE icon |
| **5.** | **Expected Output** | VVIT  D-Notice Board |
| **6.** | **Actual Output** | VVIT  D-Notice Board |
| **7.** | **Status** | Pass |
| **8.** | **Remarks** |  |

**CHAPTER - 7**

**CONCLUSION AND FUTURE SCOPE**

**7.1 CONCLUSION**

Thus we deploy a remote accessible system which can employ the real time application of the Internet of Things (IoT) using Python. This system is now accessible throughout the world from any computer that supports the android application feature. Previously the system was limited to the communication side (SMS limited to 160 characters), now it is a changed form to internet which we can send number of characters of messages or notices unlimitedly with a better communication than GSM. Thus Raspberry Pi, a small handy computer yet a powerful device which is user friendly providing world wide range of communication.

**7.2 FUTURE SCOPE**

The Raspberry Pi system that was developed can further be improvised for displaying not only the messages or notices of characters or letters but also can display the images or the videos that can be played of different types of formats depending upon the screen that is connected to the Raspberry Pi. It also lets users post gif’s and typoW3graphy on the fly - this has to be fun after all.



Fig.7.1 Large LED Display Screen

We could then interpret all this data and output it to the LED’s themselves as shown in the below figure.

One of the most exciting things about the Raspberry Pi is that each version comes equipped with a HDMI connector, meaning that anyone with a HDMI-compatible TV (which is the majority of people in North America and Europe) can easily connect the device to their living room television.



Fig. 7.2 LED Screen displaying image

Along with the SD card and power supply, the HDMI (High Definition Media Interface) cable is one of the [most important pieces of equipment](https://www.makeuseof.com/tag/build-a-raspberry-pi-media-center/) that you can use with your Raspberry Pi, which means that in theory you can connect it to a wide selection of televisions and even modern desktop computer monitors.

HDMI is ubiquitous, available on the original Raspberry Pi, the Model B revisions, the Raspberry Pi 2, and the [Raspberry Pi 3.](https://www.makeuseof.com/tag/raspberry-pi-3-faster-better-wi-fi-bluetooth/) It is also present on the [Pi](https://www.makeuseof.com/tag/getting-started-raspberry-pi-zero/) [Zero,](https://www.makeuseof.com/tag/getting-started-raspberry-pi-zero/) although we’ll need a mini HDMI adaptor to use HDMI here.

**CHAPTER-8**

**REFERENCES**

1. Cuno Pfister “Getting Started With the Internet of Things – Connecting Sensors and Wireless Networks to the Cloud”, Make Projects Publishers, 2011.
2. P. Bhaskara Chary “An IoT Based Web Page Controlled Notice Board”,

INTERNATIONAL JOURNAL OF INNOVATIVE TECHNOLOGY AND RESEARCH (IJITR), Volume No.5, Issue No.5 , August, 2017, ISSN (print) : 7224 -7225 .

1. Li, J., Da-You, L., and Bo, Y., “Smart home research, ”Proceedings of the

Third International Conference on MachineLearning and Cybernetics, vol. 2, pp. 659– 663, Shanghai,.

1. Choi, J., Shin, D., and Shin, D., Research on Design and Implementation of the Artificial Intelligence Agent for Smart Home Based on Support Vector Machine, Berlin/Heidelberg: Springer, p. 417, 2005.
2. Li, B., Hathaipontaluk, P., and Luo, S., “Intelligent oven in smart home environment,” International Conference on Research Challenges in Computer Science (ICRCCS ’09), pp. 247–250, Shanghai.
3. https://www.dexterindustries.com/howto/access -your - raspberry -pi -from - outside -your -home -or -local -network/
4. https://learn.adafruit.com/character -lcds/circuitpython - code
5. http://raspberrywebserver.com/html -and -css/making - a - complete -html - page.html
6. [https://fallthrough.io/2017/02/external -access -pi - ngrok/](https://fallthrough.io/2017/02/external%20-access%20-pi%20-%20ngrok/)
7. <https://www.raspberrypi.org/help/>
8. <https://circuitdigest.com/microcontroller-projects/webcontrolled-iot-notice-board-using-raspberry-pi>
9. [https://www.raspberrypi.org/products/raspberry-pi-3- model-b/](https://www.raspberrypi.org/products/raspberry-pi-3-%20model-b/)
10. [https://diyhacking.com/connect-raspberry-pi-tolaptop- display/](https://diyhacking.com/connect-raspberry-pi-tolaptop-%20display/)
11. <https://pimylifeup.com/raspberry-pi-lcd-16x2/>
12. <https://www.adafruit.com/product/3055>
13. <https://code-maven.com/using-templates-in-flask>
14. [https://www.tutorialspoint.com/flask/flask\_templates .html](https://www.tutorialspoint.com/flask/flask_templates%20.html)
15. [https://prezi.com/ezvpbuu446zm/bluetooth-baseddigital- notice-board-with-display-on-scrolli/](https://prezi.com/ezvpbuu446zm/bluetooth-baseddigital-%20notice-board-with-display-on-scrolli/)
16. <http://pythonhow.com/html-templates-in-flask/>
17. <https://iotbytes.wordpress.com/python-flask-webapplication-on-raspberry-pi-with-nginx-and-uwsgi/>
18. https://www.ntu.edu.sg/home/ehchua/programming/ webp programming/Python3\_Flask.html
19. <https://ngrok.com/>
20. https://www.dexterindustries.com/howto/accessyour- raspberry-pi-from-outside-your-home-or-localnetwork/
21. [https://https.cio.gov/hsts/.](https://https.cio.gov/hsts/)
22. [http://jamesdotcom.com/?p=400.](http://jamesdotcom.com/?p=400)
23. <https://www.raspberrypi.org/about>
24. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b>
25. https://www.raspberrypi.org/documentation/configuration/raspi-config.md.
26. <https://www.raspberrypi.org/forums/viewtopic.php?&t=44617>
27. http://productimage.bdstall.com/giant\_11203.jpg
28. https://learn.adafruit.com/adafruit-16x2-character-lcd-plus-keypad-forraspberry-pi?view=all.
29. https://www.raspberrypi.org/forums/viewtopic.php?&t=44617
30. https://www.raspberrypi.org/about/
31. http://www.thoughtcrime.org/software/sslstrip/
32. http://www.clker.com/clipart-stick-note-with-pin-1.html
33. http://www.driverlesstransportation.com/experts-20-billion-things-will-connected-internet-2020-including-cars-12959
34. https://www.haileybury.com/news/haileybury-reach-raspberry-pi-finals-again
35. https://www.woltersadvertising.com/how-to-set-default-raspberry-pi-sound-output-to-usb-device-on-raspbian-jessie-and-retropie/
36. https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/
37. http://www.robo-dyne.com/en/shop/basic-20x4-character-lcd-black-on-green-5v/
38. https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/
39. https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/
40. https://projects.raspberrypi.org/en/projects/burping-jelly-baby/4
41. https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/
42. https://www.pinterest.com/pin/42995371419824788/
43. http://ijetech.com/R-1151/
44. https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/
45. https://en.wikipedia.org/wiki/Composite\_video
46. https://arachnoid.com/raspberry\_pi/index.html
47. https://www.youtube.com/watch?v=srVoBC3yfcs
48. https://electronicsforu.com/resources/learn-electronics/16x2-lcd-pinout-diagram
49. http://www.robo-dyne.com/en/shop/basic-20x4-character-lcd-black-on-green-5v/
50. https://www.distrelec.ch/en/segment-lcd-mm-display-elektronik-de-117-ts-20/p/17506647
51. https://dir.indiamart.com/impcat/dot-matrix-display.html
52. http://www.zem.fr/raspberry-pi-et-afficheur-lcd-hitachi-hd44780-1602-part-2/
53. https://wiki.mchobby.be/index.php?title=Fichier:Rasp -Hack-Afficheur-LCD-Montage-a02.png
54. https://www.aliexpress.com/item/1Pcs-WH148-Type-B500K-Ohm-Linear-Taper-Rotary-Potentiometer-Panel-Pot-3-Pin/32807783837.html
55. http://electro-notes.com/articles/electronic\_components/resistors/variable-resistors-potentiometers.php
56. http://sound.whsites.net/pots.htm
57. http://raspberrypituts.com/python-web-server-with-flask-raspberry-pi-tutorial/
58. http://raspberrypituts.com/python-web-server-with-flask-raspberry-pi-tutorial/
59. http://www.recantha.co.uk/blog/?p=15663
60. https://projects.raspberrypi.org/en/projects/python-web-server-with-flask/print
61. https://www.digitalmunition.me/2016/04/ngrok-secure-tunnels-localhost/
62. <http://www.timingcctv.com/led.html>