VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belgaum-590014, Karnataka, INDIA

PROJECT REPORT

On

"VOICE BASED HOME AUTOMATION"

Submitted in partial fulfillment of the requirements for the VIII Semester

Bachelor of Engineering

In

ELECTRONICS AND COMMUNICATION ENGINEERING

For the Academic Year

2014-2015

BY

AZARUDDIN N	**1PE11EC025**
BHARGAVI DESHPANDE	**1PE11EC028**
SHREYAS P	**1PE11EC094**
SHRIDHAR VADEYAR	**1PE11EC095**

UNDER THE GUIDANCE OF

Mr. K. PATTABHI RAMAN

Associate Professor

Dept. of ECE, PESIT (BSC).

Department of Electronics and Communication Engineering

PESIT - Bangalore South Campus

Hosur Road, Bangalore-560100

PESIT - Bangalore South Campus
Hosur Road, Bangalore-560100
Department of Electronics and Communication Engineering

CERTIFICATE

This is to certify that the project work entitled "VOICE BASED HOME AUTOMATION" carried out by Azaruddin N, Bhargavi Deshpande , Shreyas P, Shridhar Vadeyar bearing USNs 1PE11EC025, 1PE11EC028, 1PE11EC094, 1PE11EC095, respectively in partial fulfillment for the award of Degree of Bachelors (Bachelors of Engineering) in Electronics and communication Engineering of Visvesvaraya Technological University, Belgaum during the year 2014-2015.

It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the Report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for said degree.

______________________	______________________	______________________
Signature of guide	**Signature of HOD**	**Signature of the Principal**
Mr. K.Pattabhi Raman	**Dr. Subhash Kulkarni**	**Dr. J Surya Prasad**
Associate Professor	**Head of Dept.**	**Principal / Director**
Dept. of ECE	**Dept. of ECE**	**PESIT- BSC**

External Viva

Name of the Examiners	**Signature with date**
1	____________________
2.	____________________

ABSTRACT

The evolution and development in Home Automation is moving forward toward the future in creating the ideal smart homes environment. Optionally, Home Automation system design has been developed for certain situation which for those who need a special attention such as elderly person, sick patients, and handicapped person. Thus, providing a suitable control scheme using wireless protocol mode can help them do their daily routine. The objective for this project is to design a smart home automation system using voice recognition.

This project presents the overall design of Home Automation System (HAS) with low cost and wireless system. Also, the smart home concept in the system improves the standard living at home. The switch mode and voice mode are used to control the home appliances. The main control system implements wireless technology to provide remote access from smart phone. The design involves the existing electrical switches and provides more safety control on the switches with low voltage activating method. The switches status is synchronized in all the control system whereby every user interface indicates the real time existing switches status. The system intended to control electrical appliances and devices in house with relatively low cost design, user-friendly interface and ease of installation.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all the lecturers and staff of the department of Electronics and Communication Engineering for extending their help and guidance towards our project.

We would like to thank the college management and express our sincere gratitude to **Dr. J. Surya Prasad,** Director/Principal of PESIT (BSC) for having given us the opportunity for the completion of this project.

We would like to thank **Dr. Subhash Kulkarni,** Head of Department Electronics and Communication, PESIT (BSC) for giving us the support and encouragement that was necessary for the completion of this report.

We convey our thanks to the project coordinators **Mr. K. Pattabhi Raman**, Associate Professor, and **Mr. R. Sastry,** Assistant Professor for providing us all the needful information and facilities which was of a great help to complete this project successfully.

We would like to thank our project guide **Mr. K. Pattabhi Raman** Associate Professor for providing us required assistance, encouragement and constant support which helped us a lot.

Last but not least, the project would not have been a success without the support of our parents and friends.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

The “Home Automation” concept has existed for many years. The terms “Smart Home”, “Intelligent Home” followed and has been used to introduce the concept of networking appliances and devices in the house. Home automation Systems (HASs) represents a great research opportunity in creating new fields in engineering, and Computing. HASs includes centralized control of lighting, appliances, security locks of gates and doors and other systems, to provide improved comfort, energy efficiency and security system. HASs becoming popular nowadays and enter quickly in this emerging market. However, end users, especially the disabled and elderly due to their complexity and cost, do not always accept these systems.

Due to the advancement of wireless technology there are several different types of connections which are introduced such as GSM, Wi-Fi and Bluetooth. Each of the connections has their own unique specifications and applications. Among the three popular wireless connections that often implemented in HAS project, Wi-Fi is chosen for our project as the capabilities of Wi-Fi are more than enough to be implemented in the design. Also, most of the current laptop/Tablet or Smartphone come with built-in Wi-Fi adapter. It will indirectly reduce the cost of this system.

1.2 OBJECTIVE OF THE STUDY

The objective of the study is as follows

RF Communication

To understand the concept of RF Communication

Controlling Home Appliances via Application (Manual and Voice mode)

To develop an application that includes the features of manual and voice mode Application. Manual mode or Voice mode can be used to control the switches of Home appliances.

Secure Connection Channels between Application and Arduino

Use of secure protocols over Wi-Fi so that other devices cannot control the Appliances. Options for secure connection is SSL over TCP.

Controlled by any device capable of Wi-Fi (Android)

To make the home appliances flexible in control, any device capable of Wi-Fi Connectivity will able to control the home appliances from remote location.

Extensible platform for future enhancement

The application is to be highly extensible, with possibility of adding features in the Future as needed.

1.3 SCOPE

The project aims at designing a prototype for controlling the home appliances that can be controlled wirelessly via an application that provides the features of speech recognition, and switch mode. An application is run on android device. The system can be used in wide range of areas.

The system integrated with different features can be applied in the following fields.

• **The system can be used in home, small offices to the big malls**

The system can be used from home to offices to control the electrical appliances.

• **For remote access of appliances in internet or intranet.**

The home/office appliances can be controlled in intra-network or can be accessed via internet.

• **For the development of technology friendly environment**

The system incorporates the use of technology and making smart home automation. By the use of day to day gadgets we can utilize them for different prospective.

1.4 TECHNOLOGY EXPOSURES

The various technology exposures that the project provides include,

- Google's Android Open Source Technology.
- Wi-Fi technology.
- Arduino Open Source Electronics Platform.
- Electromagnetic Relay switching principles.
- RF Transmitter and Receiver.

1.5 LITERATURE REVIEW

Existing Technologies for Smart Home:

Home based system automations can range from system as simple as for heating, ventilation, and air conditioning, lighting control or Audio and video distribution to multiple sources around the house, to more complicated systems such as for security and robotics for home care or home management.

Smart home applications or task automations in a general household can be grouped by their main functions such as,

- Alert and sensors- heat/ smoke sensors, temperature sensors
- Monitoring- Regular feed of sensor data i.e., heat, CCTV monitoring
- Control- switching on/off appliances i.e., sprinklers, lightings
- Intelligence and logic- Movement tracking i.e., security appliances
- Tele care/Tele health- distress sensor, blood pressure monitoring

Current smart home devices are usually a customized hybrid of one or more of these applications for broader applications. Access to these applications can be generally grouped into 4 access types that are the hardwired type using bus line or power line based technology, as well as the wireless type utilizing radio, infra-red or Bluetooth technology. Future smart home appliances are moving towards the wireless environment and hence the Bluetooth and radio spectrum will be widely used. It is to date, a rather new technology that needs to be further proven in terms of stability and security.

Problems with current systems:

Currently one of the existing issues that are associated to smart home applications are the fact that in a home with all sorts of automated application, there will be too many remote controls or monitoring terminal, if the user installed a range of proprietary applications from different providers. There is also the fact the access range to remotely control these devices are limited by either length of cables or wireless network coverage in a personal area network. It is a widely known fact that an important example of wireless technology application is the mobile phone technology. Mobility is now a lifestyle adopted by all walks of the society, where a United Nation survey has recently revealed that 60% of the world population has a mobile phone subscription. Taking into account a mobile phone's necessity in the majority of our society, this solution will attempt to transfer the functionalities of a smart home device's remote control to a mobile phone, to achieve a truly remote access convenience. Enabling a single corresponding server in a smart home household will also resolve the issue on too many control terminals as discussed previously.

Many of the solutions do not provide a user-friendly mobile interface to monitor the home and control the equipment. None of the existing solutions provide an illustrate view of the home and the status of each equipment as it appears in home in the mobile. This can be a problem since the user doesn't have a clear idea of what is being controlled. Some of the current systems provide a view of the home through a web application. But this can be troublesome since the user has to web every time he/she wants to view the status of the home.

Most of the existing systems are not affordable and cannot be integrated with an already built home without requiring re-wiring. This is one of the major issues with regards to usability and flexibility of many systems. But this problem is addressed in this system by providing a highly flexible solution which lets the user add and remove any item to the system with ease.

Different users have different security requirements and their budgets may vary according to the level of affordability. Most of the existing systems are not affordable for most of the users. And some of the systems provide solutions which are not much useful for domestic applications. This system can easily be customized to address different security requirements according to the level of criticality of security. At the same time this solution can be adjusted to different budget levels.

CHAPTER 2

SYSTEM ARCHITECTURE

2.1 BASIC BLOCK DIAGRAM

The two blocks shown below are that of the Base station and Satellite station. Various satellite stations are controlled by the Base station.

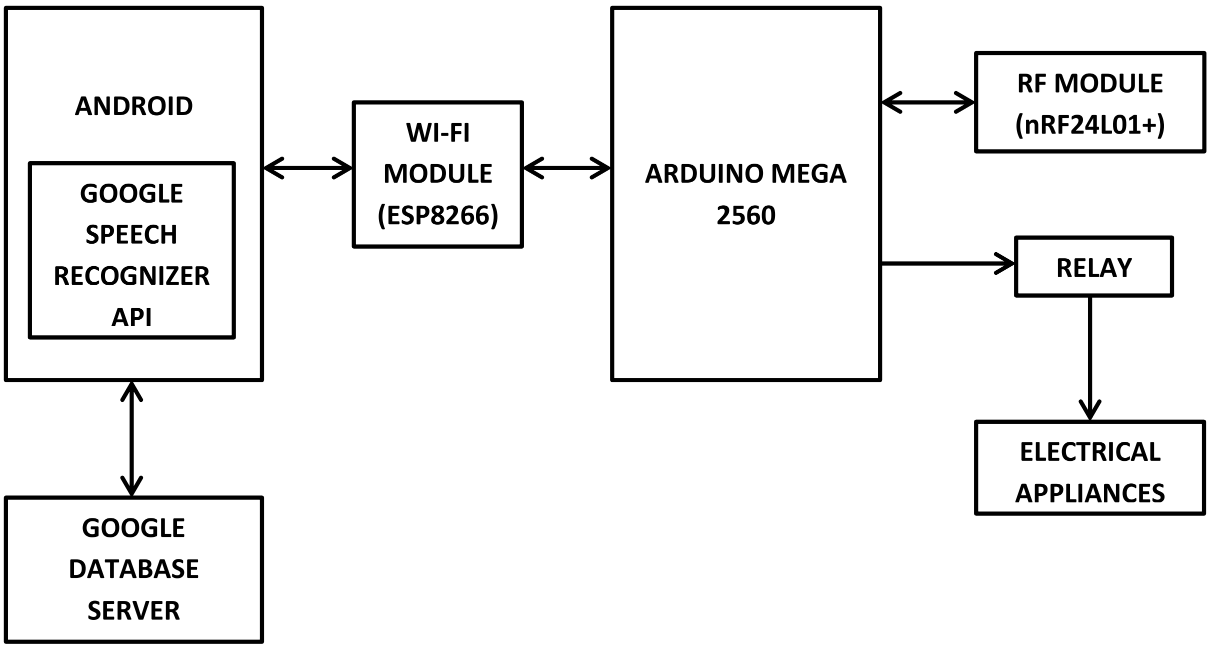


Figure 2.1.1 BASE STATION

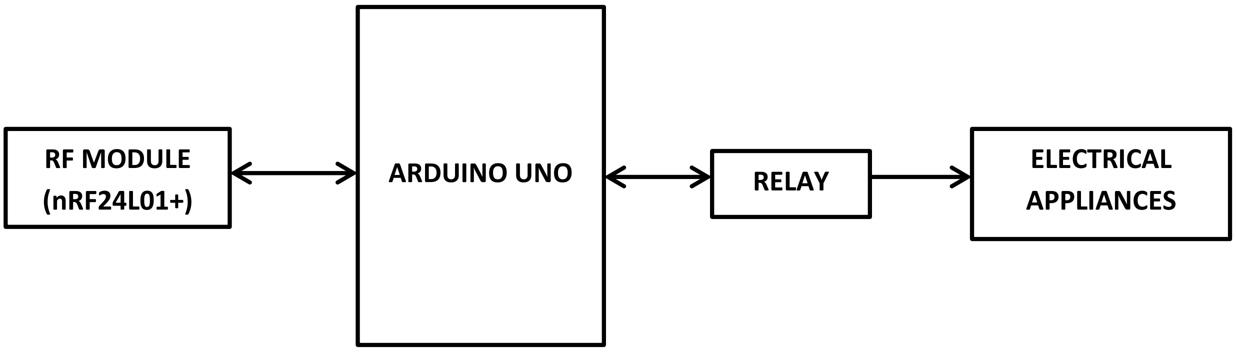


Figure 2.1.2 SATELLITE STATION

2.2 ARCHITECTURE

Main goal is to create secure, reliable and cheap platform so that each & every user who uses internet can control appliances from any location.

Before we start building our system we have some basic requirements:

- secure - we don't want our neighbor (or stranger 1000 km away) switch on/off our lights
- Reliable – we want the system to be more reliable.
- Cheap - we want build our system from components easily available on market for low price.
- Access to our system from everywhere with different connection options.

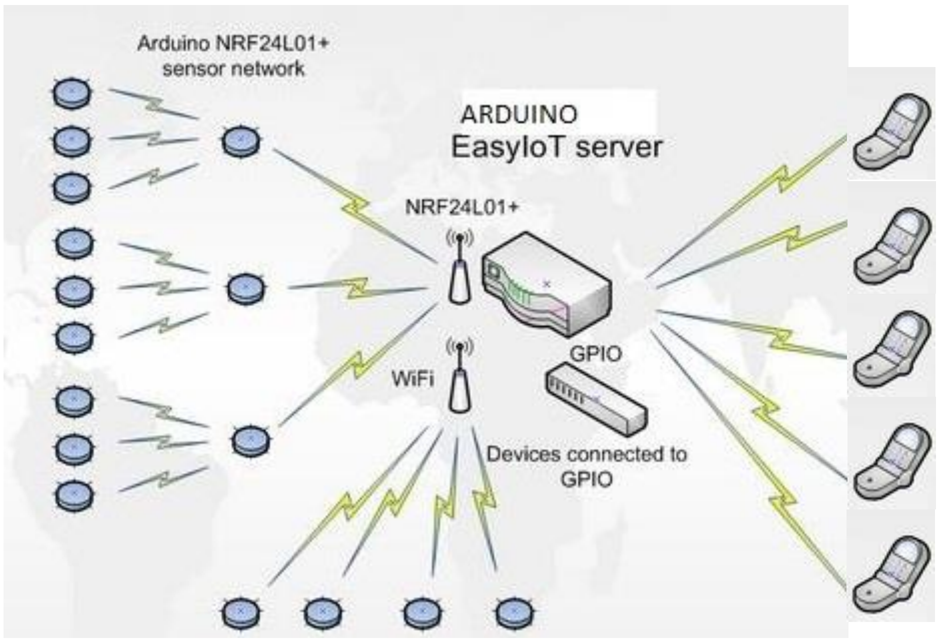


Figure 2.2.1 ARCHITECTURE DIAGRAM

2.3 PROPOSED SYSTEM

The android OS provides the flexibility of using the open source. The inbuilt sensors can be accessed easily. We have built an application with following features. Android Phone acts as a client and data are sent via socket programming. There are two modes through which we can control the electrical appliances they are:

1. Manual Mode
2. Voice Mode

Manual mode uses the radio buttons that are used to control the home appliances. The radio button sends the status of the switch i.e. whether appliance is in ON state or OFF state.

Voice Mode is used to control the home appliances using voice command. Using the inbuilt microphone of Smartphone, the application creates an intent that fetches the speech data to the Google server which responds with a string data. The string data are further analyzed and then processed.

All the devices are connected to a common network.

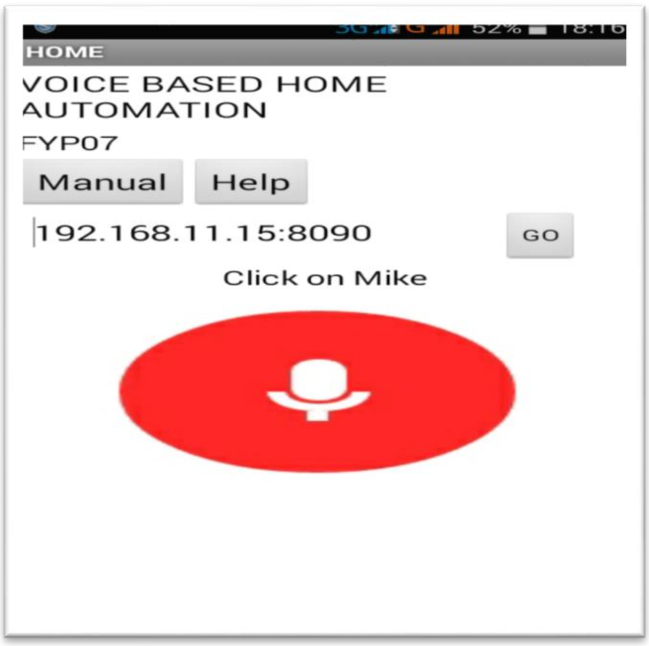


Figure 2.3.1 SNAPSHOT OF THE APP

Figure above shows the snapshot of the App where manual and voice modes are used to control the home appliances.

CHAPTER 3

HARDWARE REQUIREMENTS

3.1 INTRODUCTION

Hardware and controllers are an essential part of the projects that are expected to bring about some electrical or mechanical changes in its environment. To bring about any change in the physical world, the computer that runs a program can only send out signals in the form of electrical pulses. These pulses are of no use unless they are properly coupled to a developed hardware system that can perform the physical changes to realize what the software intended to do.

The hardware implementation is not just about connecting the wires and the components together; it forms the basis of many user case and system requirements. It requires proper and precision designing that is done while keeping in mind not just the users requirements but also the aspects of the system performance ranging from the systems power-performance ratio, cost of implementation , system usability, system reliability and system flexibility. There is a lot of programming involved in developing the hardware to run the controllers efficiently and to maintain the connectivity between it and all the mechanical devices it is in contact with.

This section includes the description of all the hardware requirements for the project.

⇒ Arduino mega series and UNO
⇒ Android Smartphone
⇒ RF transmitter and Receiver
⇒ Wi-Fi Module
⇒ Relays
⇒ USB cables

3.2 ARDUINO BOARD

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software running on a computer (e.g. Flash, Processing, and Max MSP).

Arduino UNO

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter. "Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0

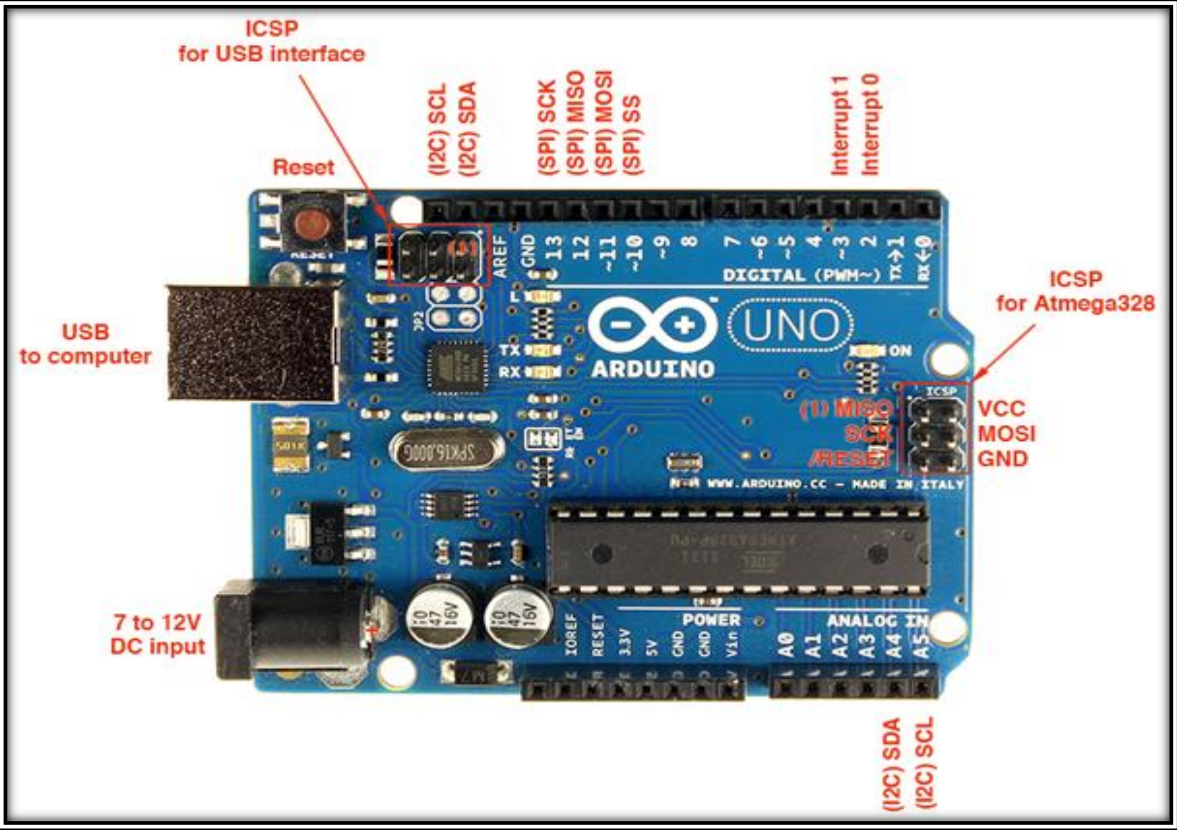


Figure 3.2.1 ARDUINO UNO BOARD

UNO Specifications

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 Ma
DC Current for 3.3V Pin	50 Ma
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by boot loader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

ARDUINO MEGA 2560

Arduino Mega 2560 The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 .It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller The important feature that is being utilized in this project is that it has four Serial RX and TX Ports. Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip.

Communication The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers.

The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega16U2 on the board channels one of these over USB and provides a virtual com port to software on the computer. The Arduino software includes a serial monitor that allows simple textual data to be sent to and from the board.

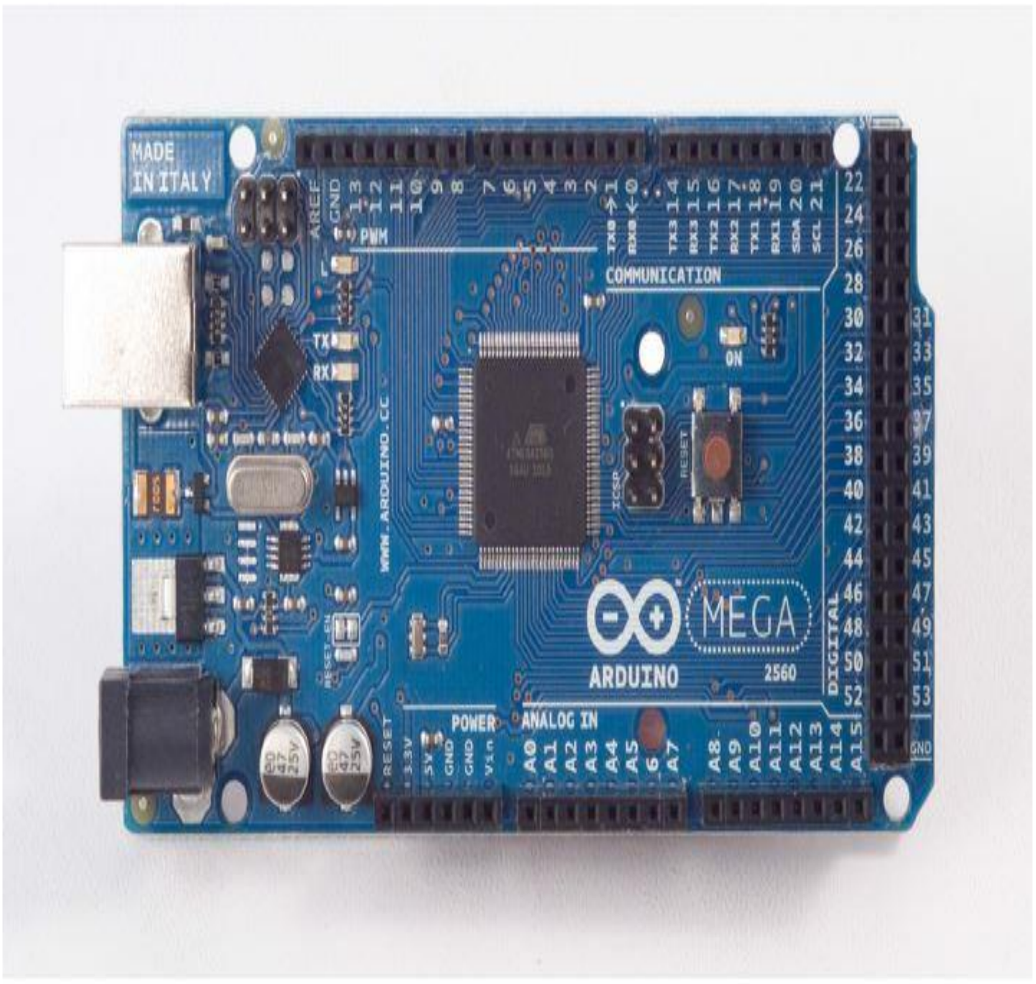


Figure 3.2.2 ARDUINO MEGA BOARD

Programming The Arduino Mega can be programmed with the Arduino software. The ATmega2560 on the Arduino Mega comes pre burned with a boot loader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol. You can also bypass the boot loader and program the microcontroller through the ICSP (In-Circuit Serial Programming).

Arduino Mega 2560's Specifications

MicroController	ATmega2560
Operating Voltage	5V
Input Voltage(recommended)	7-12V
Input Voltage(limits)	6-20V
Digital I/o Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40mA
DC Current for 3.3V Pin	50mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8KB
EEPROM	4KB
Clock Speed	16MHz

3.3 ANDROID SMART PHONE

Android is an open-source platform founded in October 2003 by Andy Rubin and backed by Google, along with major hardware and software developers (such as Intel, HTC, ARM, Motorola and Samsung) that form the Open Handset Alliance. In October 2008, HTC released the HTC Dream, the first phone to use Android. The software suite included on the phone consists of integration with Google's proprietary applications, such as Maps, Calendar, and Gmail, and a full HTML web browser. Android supports the execution of native applications and third-party apps which are available via Google Play, which launched in October 2008 as Android Market. By Q4 2010, Android became the best-selling smartphone platform.

Figure 3.3.1 SMART PHONE FOR ANDROID APP

3.4 RF TRANSMITTER AND RECEIVER

We prefer RF Transmission over Infrared as RF signal can travel longer distance than infrared. Only line of sight communication is possible through infrared, while radio frequency signals can be transmitted without line of sight. The range of RF communication can be varied as per our requirement by varying the operating voltage (Approximately 20-200mtrs).

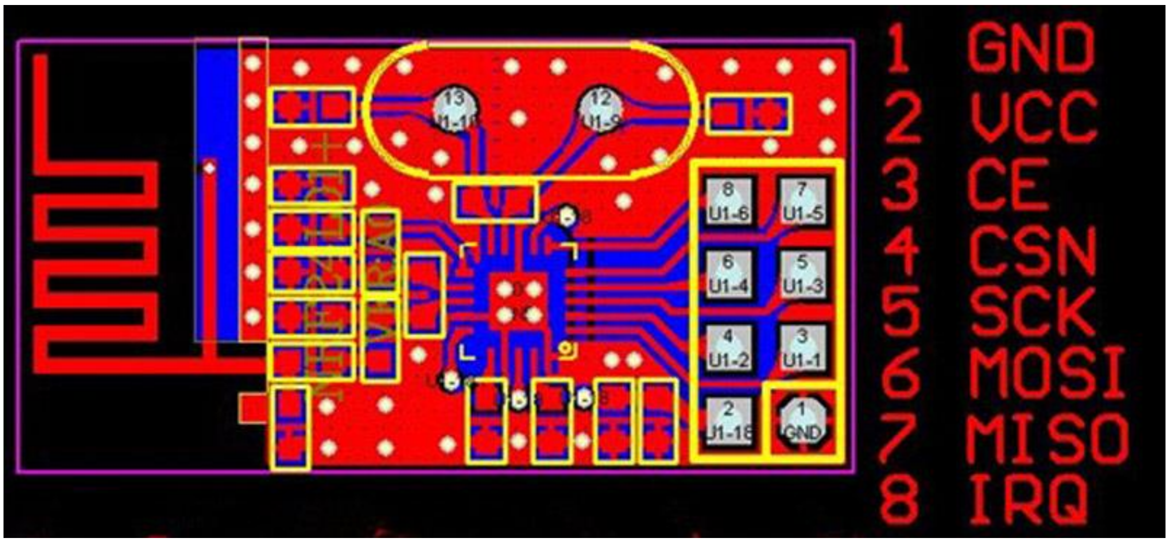


Figure 3.4.1 nRF24L01+ PIN NUMBERS

FEATURES

- True single chip GPSK transceiver
- Complete OSI Link Layer in hardware
- Enhanced shock Burst
- Auto ACK &retransmit
- Address and CRC computation
- On the air data rate 1 or 2Mbps
- Digital interface(SPI) speed 0-8 Mbps
- 125 RF channel operation
- 50 V tolerant signal input pads
- 20- pin package(QFN20 4*4)
- Power supply range 1.9 to 3.6V
- Uses ultra-low cost +/-60 ppm crystal

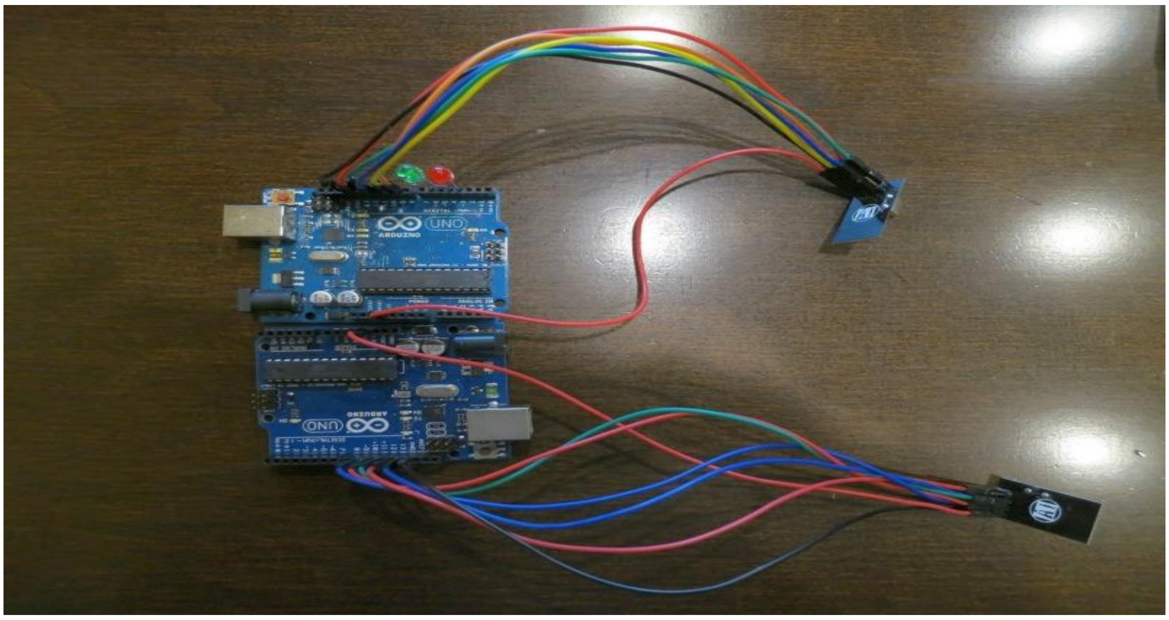


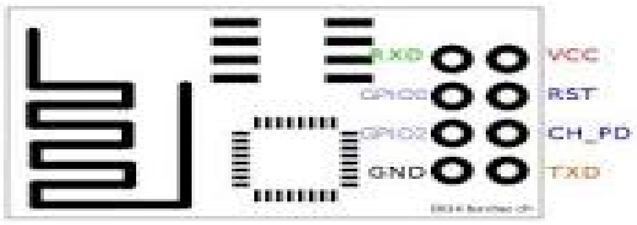
Figure 3.4.2 RF MODULE CONNECTED TO ARDUINO

NRF24L01+ is a single chip radio transceiver for the worldwide 2.4 - 2.5 GHz ISM band. The transceiver consists of a fully integrated frequency synthesizer, a power amplifier, a crystal oscillator, a demodulator, modulator and Enhanced Shock Burst™ protocol engine. Output power, frequency channels, and protocol setup are easily programmable through a SPI interface. Current consumption is very low, only 9.0mA at an output power of -6dBm and 12.3mA in RX mode. Built-in Power Down and Standby modes makes power saving easily realizable.

APPLICATIONS

- Wireless mouse, keyboard, joystick
- Industrial Sensors
- Wireless data communication
- Alarm security system
- Home Automation
- Surveillance
- Telemetry

3.5 Wi-Fi MODULE



Pin diagram

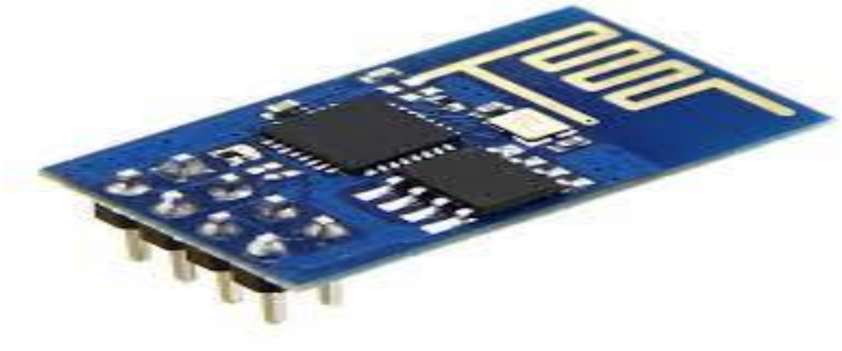


Figure 3.5.1 ESP8266 WIFI MODULE

ESP8266 is an impressive, low cost Wi-Fi module suitable for adding Wi-Fi functionality to an existing microcontroller project via a UART serial connection. The module can even be reprogrammed to act as a standalone Wi-Fi connected device–just add power! ESP8266 is a highly integrated chip designed for the needs of a new connected world. It offers a complete and self-contained Wi-Fi networking solution, allowing it to either host the application or to offload all Wi-Fi networking functions from another application processor.

ESP8266 has powerful on-board processing and storage capabilities that allow it to be integrated with the sensors and other application specific devices through its GPIOs with minimal development up-front and minimal loading during runtime. Its high degree of on-chip integration allows for minimal external circuitry, and the entire solution, including front-end module, is designed to occupy minimal PCB area.

Specifications:

- 802.11 b/g/n
- Wi-Fi Direct (P2P), soft-AP
- Integrated TCP/IP protocol stack
- Integrated TR switch, LNA, power amplifier and matching network
- Integrated PLLs, regulators, DCXO and power management units
- +19.5dBm output power in 802.11b mode
- Power down leakage current of <10uA
- Integrated low power 32-bit CPU could be used as application processor
- SDIO 1.1/2.0, SPI, UART
- STBC, 1×1 MIMO, 2×1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4ms guard interval
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)

Advantages of Wi-Fi over other wireless technologies like Bluetooth and ZigBee:

Bluetooth is generally used for point to point networks and Bluetooth operates at a much slower rate of around 720 Kbps which is very small for data transfer or moving large amount of data like the image captured from a camera, whereas the bandwidth of Wi-Fi can be up to 150Mbps and very ideal for data transmission. Wi-Fi is very much secure means of communication than Bluetooth. Wi-Fi connection to send video, audio, and telemetry operation, while accepting remote control commands from an operator who can be located virtually anywhere in the world.

3.6 RELAYS

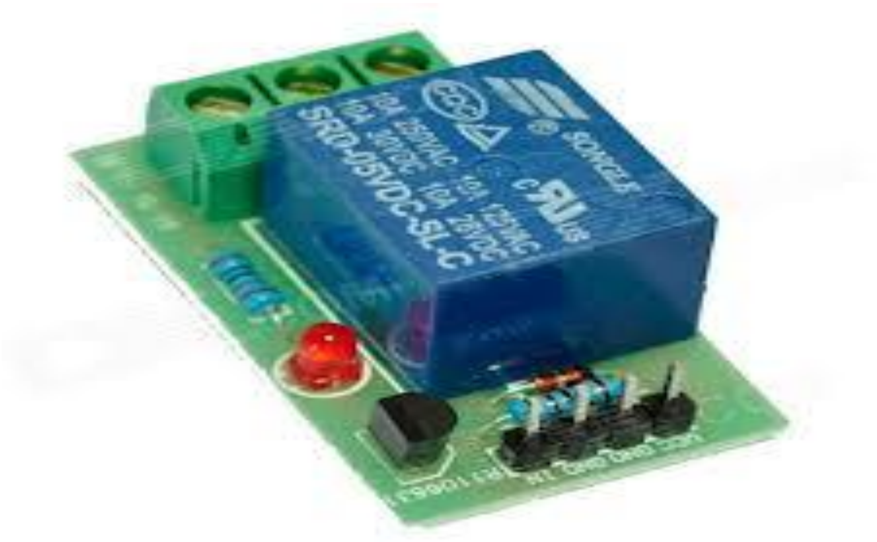


Figure 3.6.1 RELAYS

The relay board that we're using comes with 3 relays that can each handle a load of up to 6A. The relays are driven using a ULN2003 high voltage high current Darlington array IC.A **relay** is an electrically operated switch. Many relays use an electro magneto mechanically operate a switch, but other operating principles are also used, such as solid-state relays. Relays are used where it is necessary to control a circuit by a low-power signal (with complete electrical isolation between control and controlled circuits), or where several circuits must be controlled by one signal.

3.7 USB CABLE

Figure 3.7.1 USB CABLE

Universal Serial Bus (USB) is an industry standard developed in the mid-1990s that defines the cables, connectors and communications protocols used in a bus for connection, communication, and power supply between computers and electronic devices. USB was designed to standardize the connection of computer peripherals (including keyboards, pointing devices, digital cameras, printers, portable media players, disk drives and network adapters) to personal computers, both to communicate and to supply electric power. It has become commonplace on other devices, such as smartphones, PDAs and video game consoles. USB has effectively replaced a variety of earlier interfaces, such as serial and parallel ports, as well as separate power chargers for portable devices.

CHAPTER 4

SOFTWARE REQUIREMENTS

4.1ARDUINO ENVIRONMENT AND DRIVERS

Programming: The Arduino mega can be programmed with the Arduino software. The Atmega2560 on the Arduino mega comes pre-burned with a boot loader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol. You can also bypass the boot loader and program the microcontroller through the ICSP (In circuit serial programming). The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino or other microcontrollers. The ATmega2560 provides 4 hardware UARTs for TTL (5V) serial communication. An ATmega16U2 on the board channels one of these over USB and provides a virtual com port to software on the computer. The Arduino software includes a serial monitor that allows simple textual data to be sent to and from the board.

All the necessary drivers can be easily installed by going to arduino.cc website, which is a free version released by Arduino.

4.2 MIT APP INVENTOR

App Inventor for Android is an open-source web application originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT).

It allows newcomers to computer programming to create software applications for the Android operating system (OS). It uses a graphical interface, very similar to Scratch and the Star Logo TNG user interface, which allows users to drag-and-drop visual objects to create an application that can run on Android devices. In creating App Inventor, Google drew upon significant prior research in educational computing, as well as work done within Google on online development environments.

App Inventor and the projects on which it is based are informed by constructionist learning theories, which emphasizes that programming can be a vehicle for engaging powerful ideas through active learning.

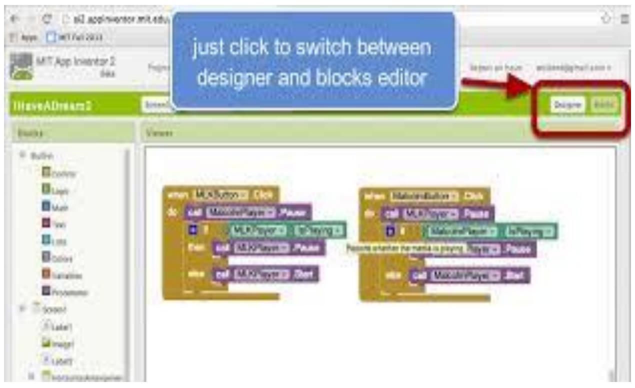


Figure 4.2.1 SNAPSHOT OF APP INVENTOR PLATFORM

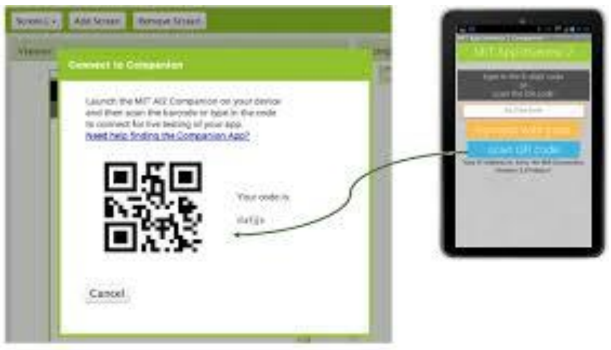


Figure 4.2.2 QR CODE SCANNING

MIT App Inventor is a blocks-based programming tool that allows everyone, even novices, to start programming and build fully functional apps for Android devices. Newcomers to App Inventor can have their first app up and running in an hour or less, and can program more complex apps in significantly less time than with more traditional, text-based languages. Initially developed by Professor Hal Abelson and a team from Google Education while Hal was on sabbatical at Google, App Inventor runs as a Web service administered by staff at MIT's Center for Mobile Learning - a collaboration of MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) and the MIT Media Lab. MIT App Inventor supports a worldwide community of nearly 3 million users representing 195 countries worldwide. The tool's more than 100 thousand active weekly users have built more than 7 million android apps.

IMPLEMENTATION

App Inventor includes:

- A designer, in which a program's components are specified. This includes visible components, such as buttons and images, which are placed on a simulated screen, and non-visible components, such as sensors and web connections.
- A *blocks editor*, in which the program's logic is created.
- A compiler based on the Kawa language framework and Kawa's dialect of the Scheme programming language, developed by Per Bothner and distributed as part of the GNU operating system by the Free Software Foundation.
- An app for real-time debugging on a connected Android device.

CHAPTER 5

DESIGN IMPLEMENTATION

5.1 DESIGN AND INTERFACING ARDUINO AND Wi-Fi

The interfacing of the Arduino AT Mega to the Wi-Fi module (ESP8266) is achieved through the use of serial interface.

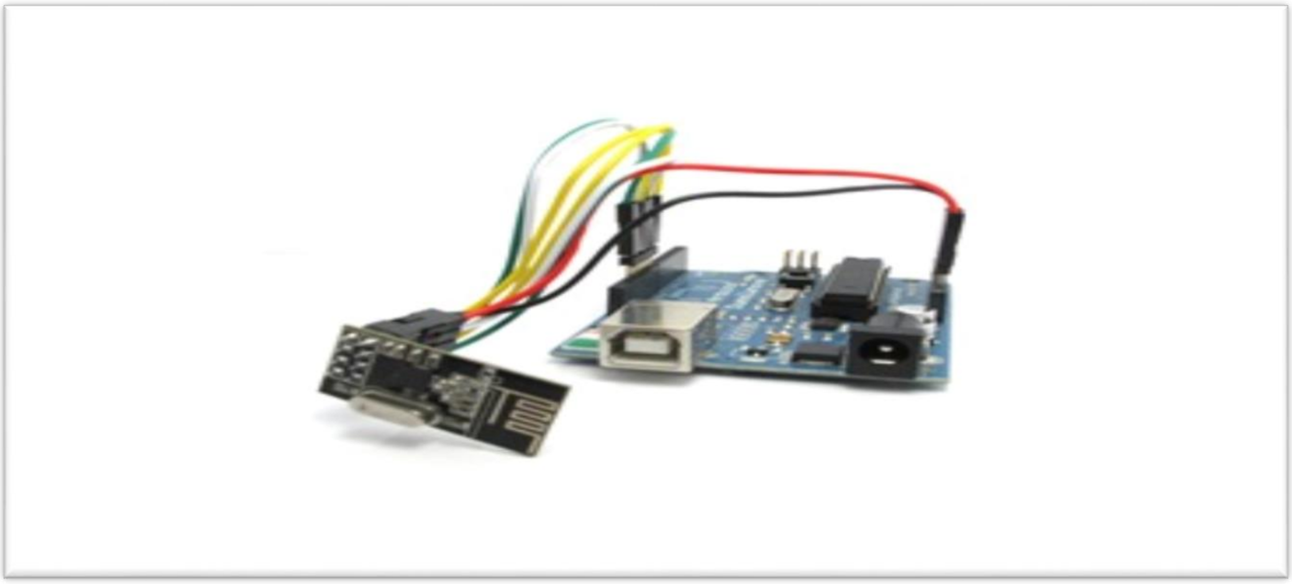


Figure 5.1.1 ARDUINO INTERFACING WITH WiFi MODULE

The Tx and Rx pin of both the Arduino and Wi-Fi module is connected along with the 3.3V power supply and ground pins.

Depending on the firmware version, baud rate varies it can be 9600, 57600 or 115200 baud. But this is considered as a drawback as it can cause communication issues. But to interface this module we have used a baud rate of 9600.Once the baud rate is decided, one can send the **AT** commands to the module. With the help of **AT** commands we can check the connection establishment between the Arduino and the Wi-Fi module.

5.2 DESIGN AND INTERFACING ARDUINO AND RF's

The power pins are connected as 3.3V for VCC and GND pin to GROUND. CE and CSN pins can be connected to any digital pins. We have used Radio Head library.

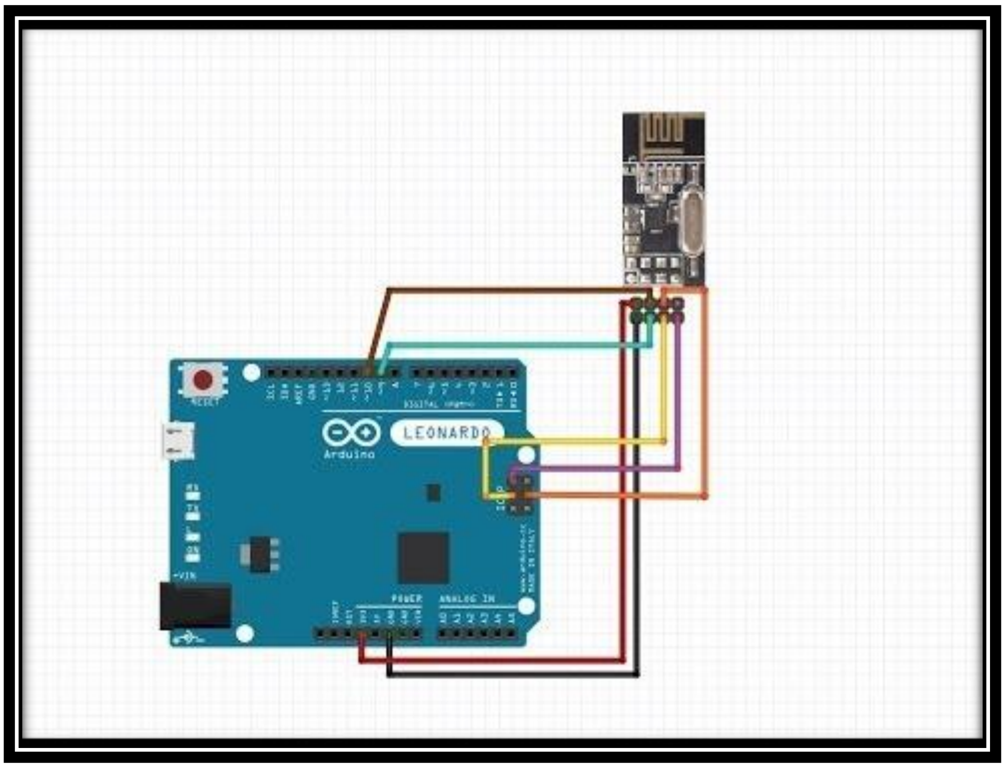


Figure 5.2.1 RF PIN CONNECTIONS

A series of 2.4 GHz Radio modules that are all based on the Nordic Semiconductor nRF24L01+ chip. The Nordic nRF24L01+ integrates a complete 2.4GHz RF transceiver, RF synthesizer, and baseband logic including the Enhanced ShockBurst™ hardware protocol accelerator supporting a high-speed SPI interface for the application controller. The low-power short-range (200 feet or so) Transceiver is available on a board with Arduino interface and built-in Antenna

5.3 IMPLEMENTATION OF RELAYS

Relays are devices that act as the actuators of this system. Depending on the signals that are given to the relay it triggers either one or the other connection made o it. The relay ha one of the live lines of the mains connected to it, with the other line going to the appliance. The relay in itself acts an open switch and stays that way until the command is sent on its input pins, in which case the relay gets trigged and switch is closed, supplying the appliance with its required power.

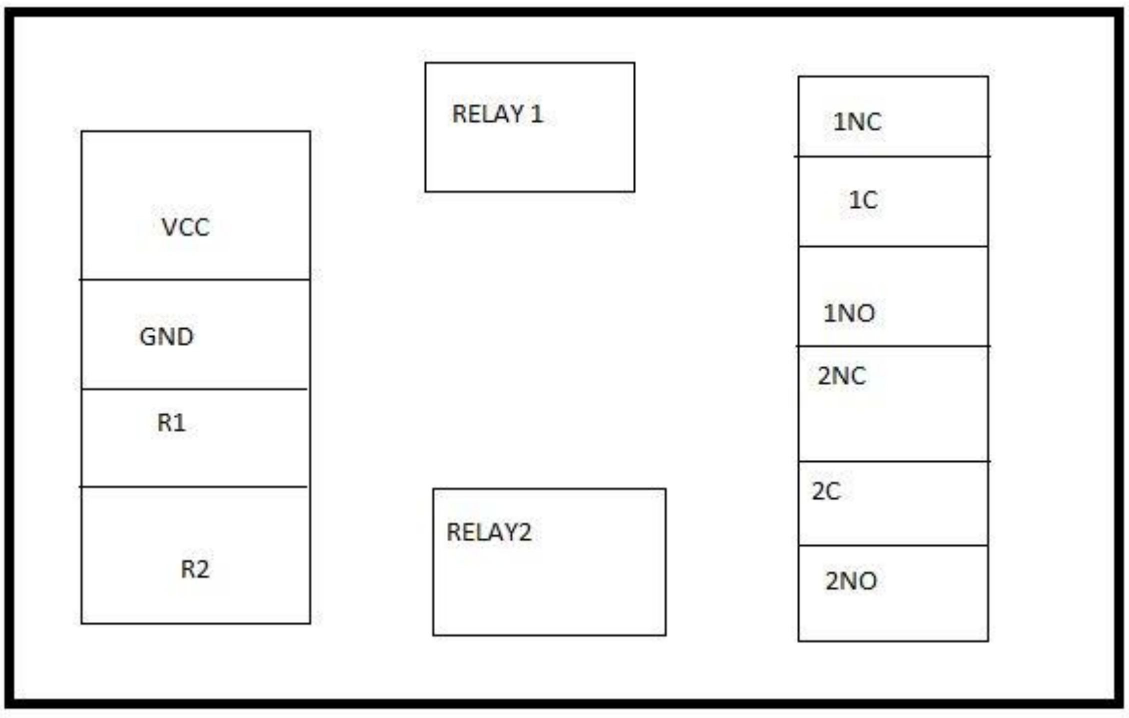


Figure 5.3.1 RELAY PIN CONFIGURATION

The diagram above shows the block diagram of the relay, the port that faces the input side consists of 4 pins that are interfaced to the satellites unit; This include 2 pins for VCC and GND ,and 2 for input of relays.

The other end consists of the connections that are made between the appliances and the mains. The 3 pins are NC, C and NO (normally closed, closed and normally opened). The connection between C and NC is closed, when the input to the relay is low. When it becomes high then closed and normally open are connected and isolated.

5.4 FINAL PROTOTYPE PICTURE

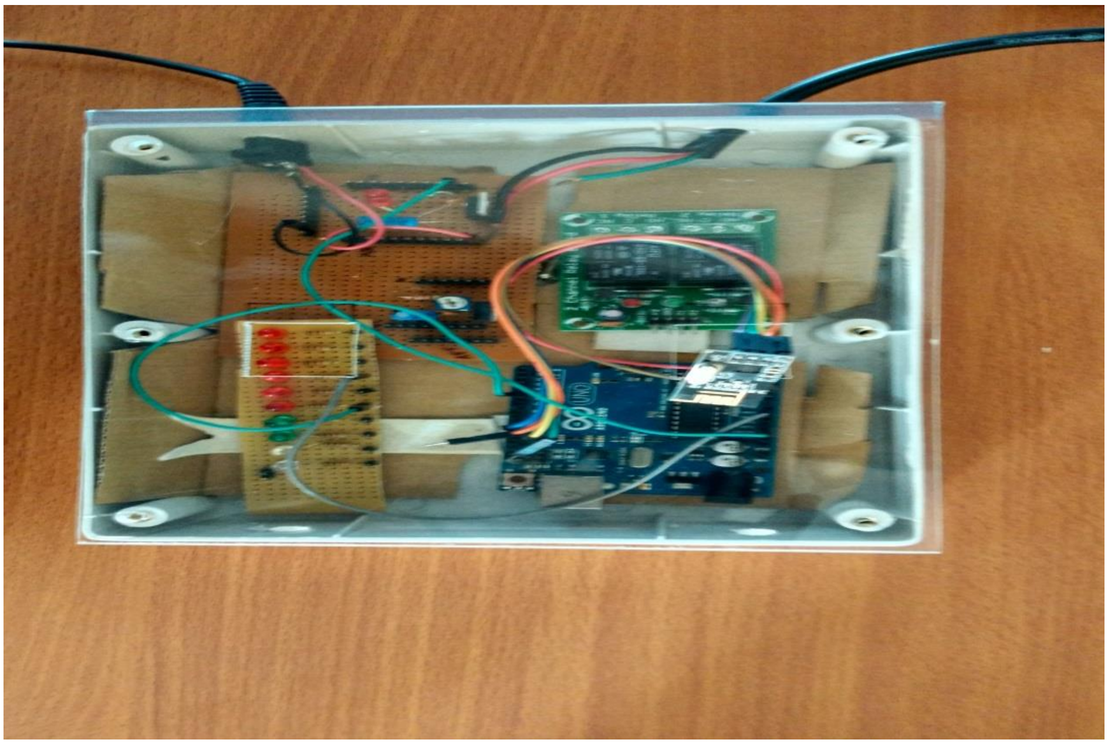


Figure 5.4.1 FINAL PROTOTYPE PICTURE

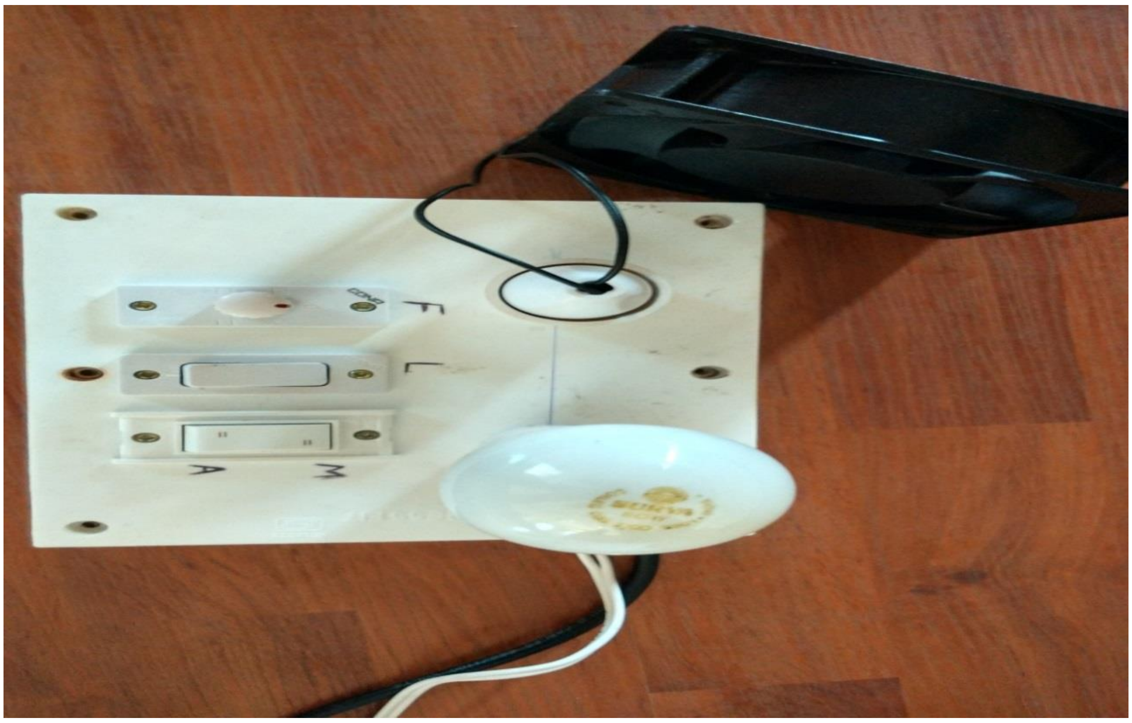


Figure 5.4.2 FINAL PROTOTYPE PICTURE

5.5 FLOW DIAGRAM

The figure below shows the flow chart of our project.

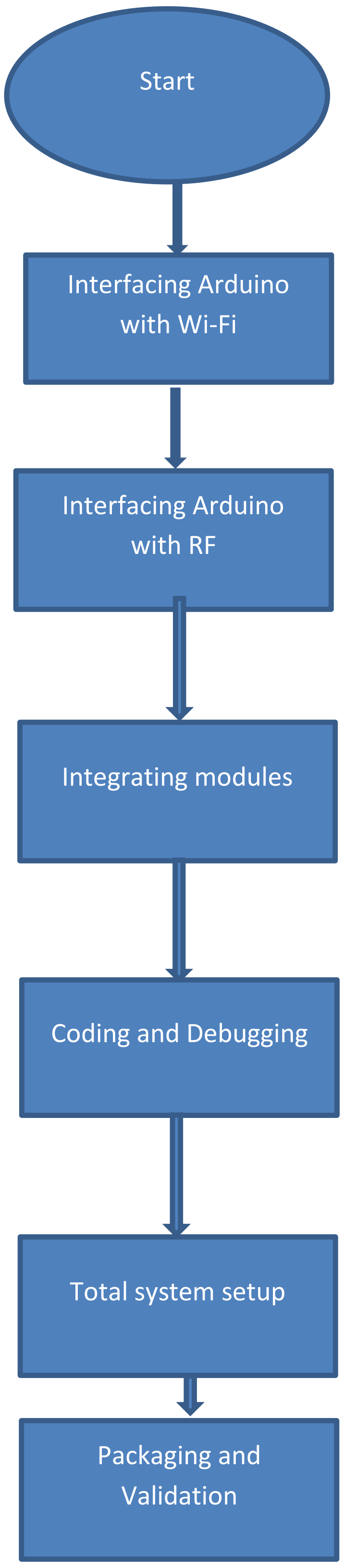


Figure 5.5.1 FLOW DIAGRAM OF PROJECT

CHAPTER 6

TESTING

6.1 TESTING ARDUINO

(For both UNO and Mega)

Figure 6.1.1 Arduino connected to PC

Connect the Arduino to a PC using a USB Type A Male to USB Type B Male connecting Cable.

1. Install the necessary drivers required for the Arduino board.
2. Upload the ‘Blink’ sketch form the Examples from the Arduino IDE.
3. We see that the On-Board LED present on the Arduino Board (connected to pin 13) starts blinking with a time delay of 1 second between each state.

6.2 TESTING OF RF TRANSCEIVERS

RF transceivers are used to establish communication between main station and satellite stations. We are using NRF-24L01+ RF transceiver in our system. According to the datasheets it supports 50 feet to 2000 feet distance ideally, but practically around 40 feet horizontally and 2 floors vertically. The nRF24L01+ is a Low Power Transceiver Module which is used for communication between devices wirelessly over a short distance.

For testing for the proper functioning of the nRF24L01+ modules, we connect two nRF4L01+ modules to 2 different Arduino UNO modules as shown.

Arduino	***nRF24l01+***
3V3 ----------------------------------	VCC (3.3V)
Pin D8-------------------------------	CE (chip enable in)
SS pin D10-------------------------	CSN (chip select in)
SCK pin D13-----------------------	SCK (SPI clock in)
MOSI pin D11---------------------	SDI (SPI Data in)
MISO pin D12---------------------	SDO (SPI data out)
	IRQ (Interrupt output, not connected)
GND--------------------------------	GND (ground in)

1. Upload the nrf24_reliable_datagram_client program from the RadioHead library on to one of the Arduino UNO with its address as 0x01. This acts as Client.
2. Upload the nrf24_reliable_datagram_server program from the RadioHead library on to the other Arduino UNO with its address as 0x02. This acts as Server.
3. Connect both modules to 2 different PCs and open the Serial monitors in the Arduino IDEs on their respective PCs.
4. The Client monitor says “Sending to nrf24_reliable_datagram_server”, and it sends “Hello” message to the server (0x02).
5. The Server receives the message and displays it on its serial monitor along with the address of the client.
6. The Server now responds with the message "And hello back to you" to the client (0x01).
7. The Client receives the message and displays it on its serial monitor.
8. If step 4 or step 7 fails, the client generates an error message ad displays it on the serial monitor.
9. If step 6 fails, the server generates an error message and displays it on its serial; monitor.
10. This process happens in a continuous loop.

Range Testing-

1. The client module is connected to a PC and the serial monitor in the Arduino IDE is opened.
2. The server module is powered by an external power supply, say a battery or external power bank for portability purposes, so that the distance between the client and the server can be varied.
3. Initially both the server and the client are placed at close proximity to each other and the serial monitor indicates successful communication by displaying the received message "And hello back to you" along with the address of the server.
4. Now the server module is moved farther and farther apart from the client module and the client's serial monitor is continuously monitored.
5. From the client's serial monitor it can be seen that after some distance, the communication starts losing packets (a few). This is indicated by a few error messages among many successful messages.
6. The number of error messages starts increasing as the distance between the client and server starts increasing. And at some distance the successful message completely stops.

Note that the above procedure for range testing provides different results based on the surrounding where it is done. The practical range is open space is found be 35-40 meters (with 50%-70% successful communications).

6.3 TESTING OF WIFI MODULE

Wi-Fi module is used to send or receive data wirelessly. We have used ESP8266 Wi-Fi module in our system. According to the datasheets ESP8266 requires 3.3V power. We have tested the Wi-Fi module with AT commands as given in the datasheets. Once the Wi-Fi module is tested using AT commands it is now ready to use for communication.

In Wi-Fi module, to make CH_PD pin high, both VCC and CH_PD pins are shorted. This Wi-Fi module is placed at main station with Arduino Mega. While uploading the code to this board we came across "sync" error which is due to interchanged connections of Tx and Rx pins of Wi-Fi module. Also you can upload the code first and then connect the Tx and Rx pins to the Arduino board.

While connecting the Wi-Fi module to Arduino board, by mistake we gave 5V to VCC pin of Wi-Fi module and immediately the module got burnt. Since ESP8266 does not have 5V tolerant inputs, you could destroy your Wi-Fi module by connecting it to 5V. To avoid this problem in giving 3.3V supply, we have designed a separate power supply board in which we have dedicated several pins to get 3.3V supply.

The module to recognize the device (mobile phone), the internet strength should be good so that faster communication is established.

6.4 TESTING OF ANDROID APP

- Since our project is based on Voice recognition, we are using GOOGLE for voice recognition in the app.
- Our app is basically designed to provide both Auto and Manual mode controlling of the home appliances.
- Under the HELP button we have provided a list of available commands in our app for the user to understand what commands to say.
- In Auto mode, before giving voice commands, your smart phone should get connected to the IP address generated by the Wi-Fi router.
- After entering the IP address with a defined port number 8090, click on GO
- As soon as you click on GO, it will establish a connection to your base station and a packet containing 6 bits is sent as to confirm the connection.
- Then tap on Mike button and say the command, as soon as the command is received by OK Google it will send it to Google server
- Here we are splitting the command string into four words, and each word is compared with all possible similar words obtained by testing different accents of the user.
- Once the spoken command gets matched with the defined one you will receive a voice reply back as “OK Sir”, if not you will receive a voice reply back as “Invalid command please try again”.
- As soon as you receive voice reply as “OK sir”, a 6 bits of data is sent to base station where it is programmed to perform a particular task.
- In Manual mode apart from using ‘OK Google’, user is provided with a list of available rooms with the control on respective appliances in corresponding rooms.

After making the control settings of appliances, user should click on Submit button and actions happen in the same way as explained in Auto mode description.

6.5.1 TESTING RELAYS

The Relays are used for the controlling (switching) of high power devices or appliances using a low power control circuitry such as microcontrollers.

Connect VCC and Ground of the Relay Board to 12V and GND of the Power Supply Board. Connect the other end to a load along with power supply as shown.

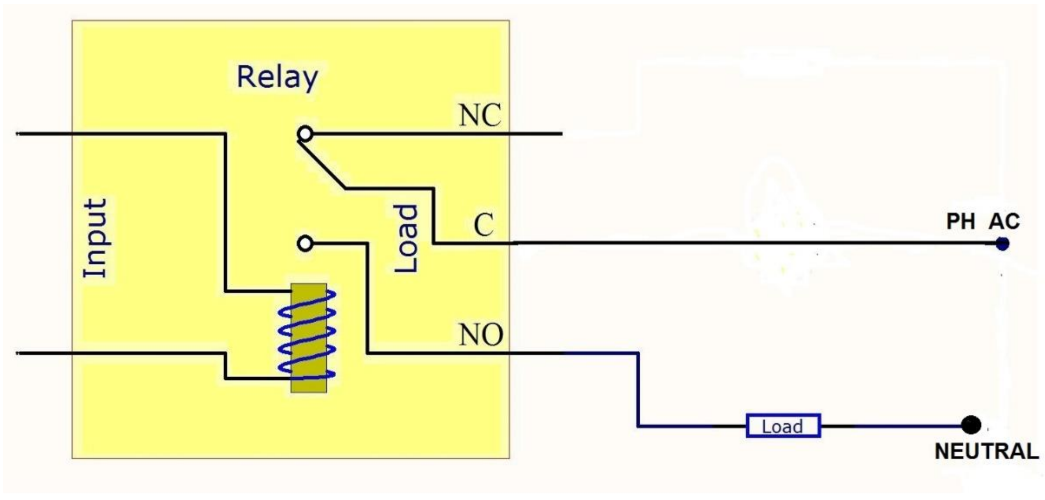


Figure 6.5.1 RELAY CONNECTION

The load can be an LED or a Buzzer for testing purposes. When we give a logic high Value at the input of the relay, the load turns ON. And for logic Zero at the input of relay, the load is OFF.

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

Voice recognition Wireless Home Automation Based is a very useful project for the adults and physically disabled persons, who are not able to do various activities efficiently when they are at home and need one’s assistant to perform those tasks. With the Voice Recognition along with Wi-Fi network we can eliminate the complication of wiring in case of wired automation and also it prevent to get up and down again and again to on/off appliances. Wi-Fi based Home Automation provides operating range much higher as compared to Bluetooth and other wireless sensor module .With the use of Wi-Fi module in Home Automation circuit ,considerable amount of power saving is possible and it is flexible and compatible with future technologies so it can be easily customized for individual requirements. On the other hand with voice recognition system, it provides secure access to home. So when we are living in a fast world where everything is changing with in no time such security is essential.

7.2 FUTURE SCOPE

- We can send this data to remote location using mobile or internet.
- We can implement other related modules like fire sensor, wind sensor.
- We can add the module of voice alarm system to indicate theft entry or gas leakage.

REFERENCES

1. Muhammad Izhar Ramli, Mohd Helmy Abd Wahab, Nabihah, "TOWARDS SMART HOME: CONTROL ELECTRICAL DEVICES ONLINE", Nornabihah Ahmad International Conference on Science and Technology: Application in Industry and Education (2006)
2. http://developer.android.com
3. www.arduino.cc/
4. http://stackoverflow.com/
5. http://electronics.howstuffworks.com/
6. https://arduino-info.wikispaces.com
7. http://www.airspayce.com/mikem/arduino/NRF24/classNRF24.html
8. https://github.com/itead/ITEADLIB_Arduino_WeeESP8266
9. http://www.airspayce.com/mikem/arduino/RadioHead/classRHReliableDatagram.html#ad282ac147986a63692582f323b56c47f
10. http://docs.iteadstudio.com/ITEADLIB_Arduino_WeeESP8266/index.html
11. https://nurdspace.nl/ESP8266
12. http://www.electrodragon.com/w/Wi07c
13. https://docs.google.com/file/d/0B_ctPy0pJuW6LVdrSFctX1dmTzA/edit
14. https://docs.google.com/file/d/0B_ctPy0pJuW6aHBqOHdyTmRnYTA/edit
15. http://bbs.espressif.com/index.php?sid=7e5b2cb2e2635af408d41e83fc717768
16. http://www.esp8266.com/
17. http://wiki.iteadstudio.com/ESP8266_Serial_WIFI_Module

APPENDIX

A1.CODES

A1.1 ARDUINO UNO CODE (SATELLITE STATION-1)

```
/*

            Arduino Code for Living Room Station.

This satellite station receives its command from the Base Station wirelessly using
nRF24L01+ module and preforms the required actions.


                         Connections

             Arduino                      nRF24L01+

             GND--------------------------------GND (1)
             3V3 --------------------------------VCC (2)
             Pin D8-------------------------------CE (3)
             Pin D10-----------------------------CSN (4)
             Pin D13-----------------------------SCK (5)
             Pin D11-----------------------------SDI (6)
             Pin D12-----------------------------SDO (7)
                                                 IRQ (8)

             Arduino                      Relay

             Pin 3--------------------------------R1 of Relay (Light)
             Pin 4--------------------------------R2 of Relay (Fan)
*/


#include <RHReliableDatagram.h>

#include <RH_NRF24.h>

#include <SPI.h>


#define CLIENT_1 2                  //Dining Room

#define SERVER_ADDRESS 1            //Bed Room

#define CLIENT_2 3                  //Living Room


#define light_pin 3
```

```
#define fan_pin 4


RH_NRF24 driver;


RHReliableDatagram manager(driver, CLIENT_2);


void setup()
{
  //Initialization
  Serial.begin(9600);
  rf_init();
  pinMode(light_pin, OUTPUT);
  pinMode(fan_pin, OUTPUT);
}


uint8_t buf[RH_NRF24_MAX_MESSAGE_LEN];  //Global Variables
uint8_t light = 0, fan = 0;
int i;


void loop()
{
  //Check to see if data is available at the Receiver Buffer
  if (manager.available())
  {
    uint8_t len = sizeof(buf);
    uint8_t from;
    //Checks if the receiver message is for the destined address
```

```
if (manager.recvfromAck(buf, &len, &from))
{
  //Print Received message on the Serial Monitor for Debugging purpose
  Serial.print("got request from : 0x");
  Serial.print(from, HEX);
  Serial.print(": ");
  Serial.println((char*)buf);

  uint8_t x = buf[0];
  uint8_t y = buf[1];
  uint8_t datt[10] = {9};

  if(x == '0' )      //light
  {
    light = 0;
  }
  else if ( x == '1')
  {
   light = 1;
  }
  else if ( x == '9')
  {
   if(light == 0)
   {
     x = '0';
   }
   else if(light == 1)
   {
     x = '1';
   }
  }
  else
  {
   x = '8';
  }



  if(y == '0' )       //fan
  {
    fan = 0;
  }
  else if ( y == '1' || y == '2' || y == '3' || y == '4' || y == '5' )
  {
   fan = 1;
  }
  else if ( y == '9')
  {
   if(fan == 0)
   {
     y = '0';
   }
   else if(fan == '1')
   {
     y = '1';
   }
  }
  else
  {
   y = '8';
  }



  digitalWrite(light_pin, light);
  Serial.print("light is ");
  if(light == 0)
  {
   Serial.println("OFF");
  }
  else if(light == 1)
  {
       Serial.println("ON");
  }

  digitalWrite(fan_pin, fan);
   Serial.print("fan is ");
  if(fan == 0)
  {
   Serial.println("OFF");
  }
  else if(fan == 1)
  {
```

```
          Serial.println("ON");
      }



      //Send a reply back to the originator Server
      datt[0] = x;
      datt[1] = y;
      if (!manager.sendtoWait(datt, sizeof(datt), from))
       Serial.println("Reply to SERVER failed");
      else
        Serial.println("Reply sent to SERVER");
    }
  }
}

void rf_init(void)    //nRF24L01+ initialization
{
  Serial.print("RF setup begin\r\n");

  if (!manager.init())
    Serial.println("RF communication initialization failed");
  else
    Serial.println("RF communication initialized");

  Serial.print("RF setup complete\r\n");
}
```

A1.2 ARDUINO UNO CODE (SATELLITE STATION-2)

```
/*
                    Arduino Code for Dining Room Station.

This satellite station receives its command from the Base Station wirelessly using
nRF24L01+ module and preforms the required actions.


                                Connections

              Arduino                          nRF24L01+

              GND--------------------------------GND (1)
              3V3 --------------------------------VCC (2)
              Pin D8-------------------------------CE (3)
              Pin D10-----------------------------CSN (4)
              Pin D13-----------------------------SCK (5)
              Pin D11-----------------------------SDI (6)
              Pin D12-----------------------------SDO (7)
                                                   IRQ (8)

              Arduino                          Relay

              Pin 3--------------------------------R1 of Relay (Light)
              Pin 4--------------------------------R2 of Relay (Fan)
*/


#include <RHReliableDatagram.h>

#include <RH_NRF24.h>

#include <SPI.h>


#define CLIENT_1 2                    //Dining Room

#define SERVER_ADDRESS 1        //Bed Room

#define CLIENT_2 3                    //Living Room
```

```
#define light_pin 3

#define fan_pin 4


RH_NRF24 driver;


RHReliableDatagram manager(driver, CLIENT_2);


void setup()

{

  //Initialization

  Serial.begin(9600);

  rf_init();                              //Initialization of nRF24L01+

  pinMode(light_pin, OUTPUT);

  pinMode(fan_pin, OUTPUT);

}


uint8_t buf[RH_NRF24_MAX_MESSAGE_LEN];

uint8_t light = 0, fan = 0;

int i;


void loop()

{

  //Check to see if data is available at the Receiver Buffer

  if (manager.available())

  {


    uint8_t len = sizeof(buf);
```

```
uint8_t from;


//Checks if the receiver message is for the destined address
if (manager.recvfromAck(buf, &len, &from))
{
  //Print Received message on the Serial Monitor for Debugging purpose
  Serial.print("got request from : 0x");
  Serial.print(from, HEX);
  Serial.print(": ");
  Serial.println((char*)buf);


  uint8_t x = buf[0];
    uint8_t y = buf[1];
    uint8_t datt[10] = {9};


  if(x == '0' )         //light
  {
     light = 0;
  }
  else if ( x == '1')
  {
    light = 1;
  }
  else if ( x == '9')
  {
    if(light == 0)
     {
```

```
    x = '0';
   }
  else if(light == 1)
   {
    x = '1';
   }
 }
 else
 {
  x = '8';
 }

 if(y == '0' )           //fan
 {
   fan = 0;
 }
 else if ( y == '1' || y == '2' || y == '3' || y == '4' || y == '5' )
 {
  fan = 1;
 }
 else if ( y == '9')
 {
  if(fan == 0)
   {
    y = '0';
   }
  else if(fan == '1')
   {
    y = '1';
   }
 }
 else
 {
   y = '8';
 }

 digitalWrite(light_pin, light);
 Serial.print("light is ");
 if(light == 0)
 {
   Serial.println("OFF");
 }
 else if(light ==1)
 {
      Serial.println("ON");
 }

 digitalWrite(fan_pin, fan);
  Serial.print("fan is ");
 if(fan == 0)
 {
   Serial.println("OFF");
 }
 else if(fan == 1)
```

```
      {
           Serial.println("ON");
      }

      //Send a reply back to the originator Server
      datt[0] = x;
      datt[1] = y;

      if (!manager.sendtoWait(datt, sizeof(datt), from))
       Serial.println("Reply to SERVER failed");
      else
        Serial.println("Reply sent to SERVER");
    }
  }
}
void rf_init(void)              //nRF24L01+ initialization
{
  Serial.print("RF setup begin\r\n");

  if (!manager.init())
    Serial.println("RF communication initialization failed");
  else
    Serial.println("RF communication initialized");

  Serial.print("RF setup complete\r\n");
}
```

A1.3 MEGA SERVER CODE (BASE STATION)

```
/*

                    Arduino Code for Bed Room Station.

This Base Station Receives command from the app through WiFi using the ESP8266
WiFi module and communicates it with the other satellite station.


                              Connections

                 Arduino                      nRF24L01+

                 GND--------------------------------GND (1)
                 3V3 --------------------------------VCC (2)
                 Pin D8-------------------------------CE (3)
                 Pin D7-------------------------------CSN (4)
                 Pin D52-----------------------------SCK (5)
                 Pin D51-----------------------------SDI (6)
                 Pin D50-----------------------------SDO (7)
                                                    IRQ (8)

                 Arduino                      Relay

                 Pin D30-----------------------------R1 of Relay (Light)
                 Pin D32-----------------------------R2 of Relay (Fan)

                 Arduino                      ESP8266

                 GND--------------------------------GND
                 3V3 --------------------------------VCC
                 Pin D18 (Tx1) ---------------------Rx
                 Pin D19 (Rx1) ---------------------Tx
*/


#include "ESP8266.h"


#include <RHReliableDatagram.h>

#include <RH_NRF24.h>

#include <SPI.h>
```

```
#define CLIENT_1 2                       //Dining Room

#define SERVER_ADDRESS 1          //Bed Room

#define CLIENT_2 3                       //Living Room


#define light_pin 30

#define fan_pin 32


RH_NRF24 driver(8, 7);                   //  Arduino Mega


RHReliableDatagram manager(driver, SERVER_ADDRESS);


ESP8266 wifi(Serial1);


void setup(void)

{

    //Initialization

    Serial.begin(9600);

    rf_init();                            //RF24 initialization

    wifi_init();                         //WIFI initialization

  }


void loop(void)

{

   uint8_t buffer[1024] = {0};

   uint8_t msg[1024]={"HTTP/1.1 200 OK\n\rContent-Type: text/html; charset=utf-

   8\n\rContent-Length: length\n\r\n\r"};        //Response to HTTP POST request
```

```
uint32_t len_msg = sizeof(msg);

uint8_t mux_id;


uint8_t MyArray[10] = {0};

uint8_t data_1[10];

uint8_t data_2[10];

uint8_t data_3[10];


//Receive any message from the WiFi if any message is available

uint32_t len = wifi.recv(&mux_id, buffer, sizeof(buffer), 100);

uint32_t len_buffer = sizeof(buffer);

 if (len > 0)

 {

    //Display the received message on the Serial Monitor for Verification and Debugging

    Serial.print("Status:[");

    Serial.print(wifi.getIPStatus().c_str());

    Serial.println("]");


    Serial.print("Received from :");

    Serial.print(mux_id);

    Serial.print("[");

    for(uint32_t i = 0; i < len; i++)

    {

       Serial.print((char)buffer[i]);

    }

    Serial.print("]\r\n");
```

```
//Extract the required data for the received message by removing the headers
for (int j = 0; j < len_buffer; j++)
{
  if(buffer[j] == '$')
  {
    j++;
    int k=0;
    while( buffer[j] != '$')
    {
      MyArray[k]=buffer[j];
      k++;
      j++;
    }
  }
}

 //Display the received message on the Serial Monitor
 Serial.println("MyArray is");
 Serial.println((char*)MyArray);

 //Split the received data for the different stations
 data_1[0]=MyArray[0];
 data_1[1]=MyArray[1];
 data_2[0]=MyArray[2];
 data_2[1]=MyArray[3];
 data_3[0]=MyArray[4];
 data_3[1]=MyArray[5];
```

```
Serial.println("data_1 is");
Serial.println((char*)data_1);
Serial.println("data_2 is");
Serial.println((char*)data_2);
Serial.println("data_2 is");
Serial.println((char*)data_3);

//Perform switching actions related to this station
swi(data_1);

//Communicate with the client modules with their respective data
Serial.println("communicating with client 1");
 comm(data_2, CLIENT_1);
Serial.println("done cwith client 1");
Serial.println("communicating with client 2");
  comm(data_3, CLIENT_2);
Serial.println("done client 2");

//Send acknoledgement back using Wifi
if(wifi.send(mux_id, msg, len_msg))
{
   Serial.print("send back ok\r\n");
}
else
{
   Serial.print("send back err\r\n");
```

```
        }


        //Release socket at TCP Port
        if (wifi.releaseTCP(mux_id))
        {
          Serial.print("release tcp ");
          Serial.print(mux_id);
          Serial.println(" ok");
         }
         else
         {
          Serial.print("release tcp");
          Serial.print(mux_id);
          Serial.println("err");
         }


       Serial.print("Status:[");
       Serial.print(wifi.getIPStatus().c_str());
       Serial.println("]");
    }
}


uint8_t light = 0, fan = 0;


void swi(uint8_t dat[])
{
       uint8_t x = dat[0];
```

```
  uint8_t y = dat[1];

  uint8_t datt[10] = {9};


if(x == '0' )          //light

{

   light = 0;

}

else if ( x == '1')

{

  light = 1;

}

else if ( x == '9')

{

  if(light == 0)

  {

    x = '0';

  }

  else if(light == 1)

  {

    x = '1';

  }

}

else

{

  x = '8';

}
```

```
if(y == '0' )          //fan
{
   fan = 0;
}
else if ( y == '1' || y == '2' || y == '3' || y == '4' || y == '5' )
{
  fan = 1;
}
else if ( y == '9')
{
  if(fan == 0)
  {
    y = '0';
  }
  else if(fan == 1)
  {
    y = '1';
  }
}
else
{
  y = '8';
}



digitalWrite(light_pin, light);
Serial.print("light is ");
```

```
    if(light == 0)
    {
      Serial.println("OFF");
    }
    else if(light ==1)
    {
        Serial.println("ON");
    }

    digitalWrite(fan_pin, fan);
     Serial.print("fan is ");
    if(fan == 0)
    {
      Serial.println("OFF");
    }
    else if(fan == 1)
    {
        Serial.println("ON");
    }

    datt[0] = x;
    datt[1] = y;
}


//WiFi Initilization Procedure
void wifi_init(void)
```

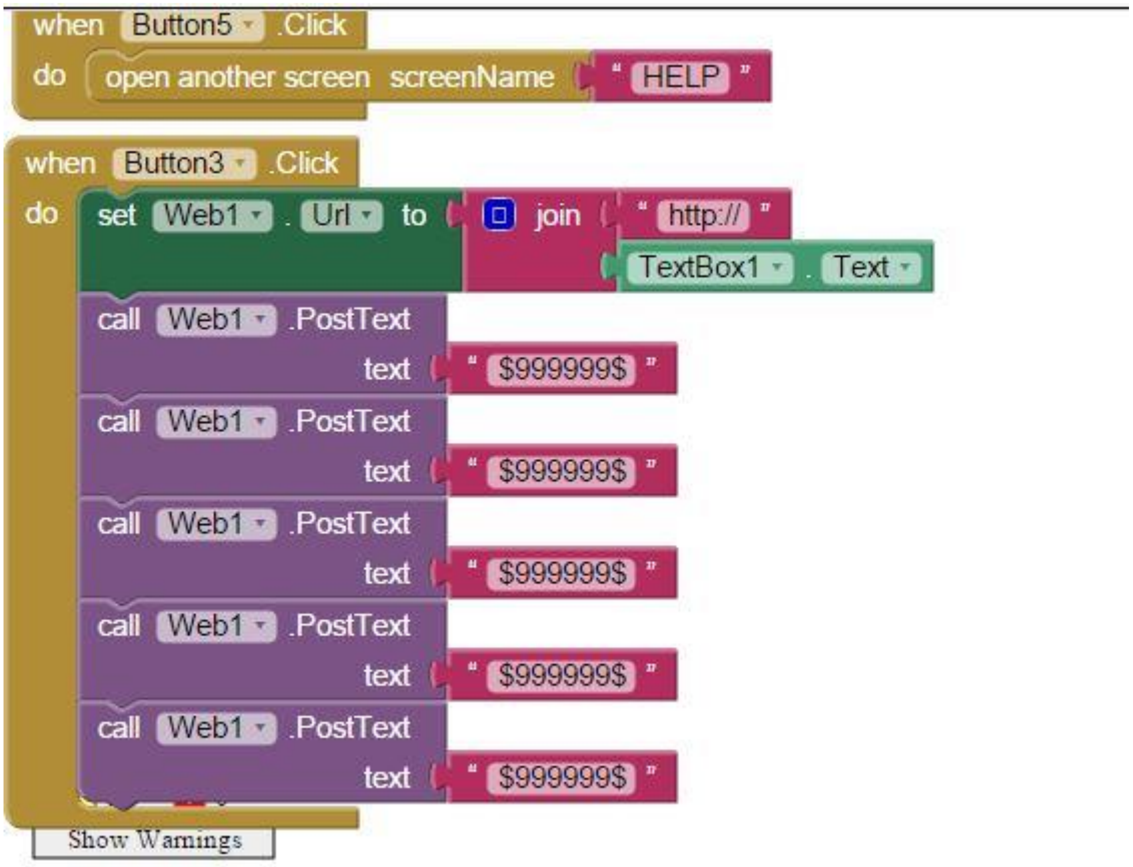
```
{
 Serial.print("WIFI setup begin\r\n");

   Serial.print("FW Version:");
   Serial.println(wifi.getVersion().c_str());

   if (wifi.setOprToStationSoftAP())
   {
       Serial.print("to station + softap ok\r\n");
   }
   else
   {
       Serial.print("to station + softap err\r\n");
   }

   if (wifi.joinAP("shreyas","qwertyuio"))
   {
       Serial.print("Join AP success\r\n");
       Serial.print("IP: ");
       Serial.println(wifi.getLocalIP().c_str());
   }
   else
   {
       Serial.print("Join AP failure\r\n");
   }

   if (wifi.enableMUX())
```

```
{
    Serial.print("multiple ok\r\n");
}
else
{
    Serial.print("multiple err\r\n");
}

if (wifi.startTCPServer(8090))
{
    Serial.print("start tcp server ok\r\n");
}
else
{
    Serial.print("start tcp server err\r\n");
}

if (wifi.setTCPServerTimeout(10))
{
    Serial.print("set tcp server timout 10 seconds\r\n");
}
else
{
    Serial.print("set tcp server timout err\r\n");
}

Serial.print("WIFI setup end\r\n");
```

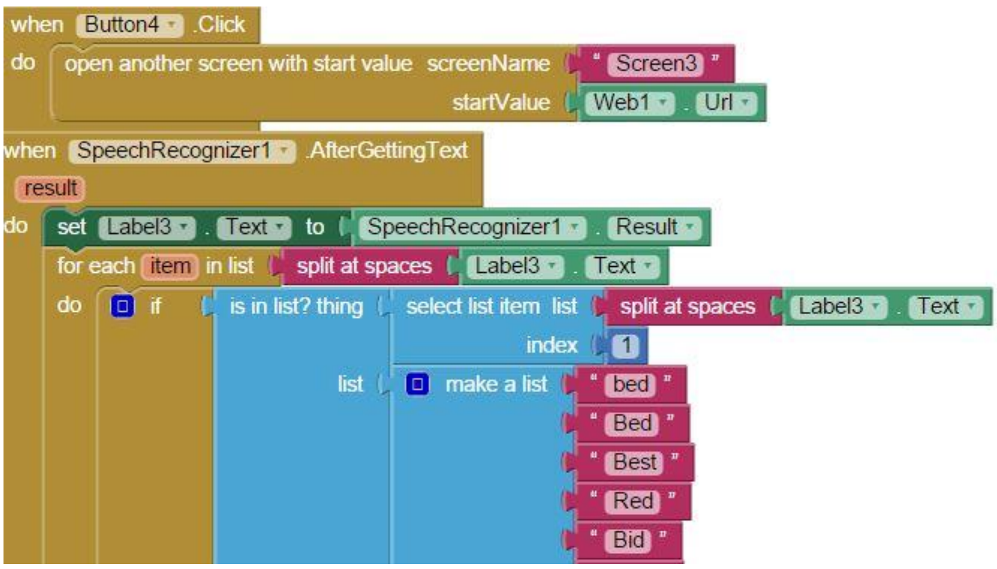
```
}


uint8_t buf[RH_NRF24_MAX_MESSAGE_LEN];
#define x 6


//Procedure for Communication with the Clients
void comm(uint8_t data[], uint8_t ADDRESS)
{
  Serial.print("Sending to ");
  Serial.println(ADDRESS, HEX);

  // Send a message to manager_server
  if (manager.sendtoWait(data, sizeof(data), ADDRESS))
  {
    // Now wait for a reply from the server
    uint8_t len = sizeof(buf);
    uint8_t from;

    if (manager.recvfromAckTimeout(buf, &len, 5000, &from))
    {
      Serial.print("got reply from : 0x");
      Serial.print(from, HEX);
      Serial.print(": ");
      Serial.println((char*)buf);
      //Staatus of the devices connected to the Client modules
      uint8_t a = buf[0];
      uint8_t b = buf[1];
      Serial.print("light is ");
      if(a == '0')
      {
        Serial.println("OFF");
      }
      else if(a == '1')
      {
           Serial.println("ON");
      }



       Serial.print("fan is ");
      if(b == '0')
      {
        Serial.println("OFF");
      }
      else if(b == '1')
      {
           Serial.println("ON");
      }
    }
    else
    {
      Serial.println("No reply, is nrf24_reliable_datagram_server running?");
    }
  }
```

```
  else
    Serial.println("sending failed");
}



//nRF24L01+ Initialization Procedure
void rf_init(void)
{
  Serial.print("RF setup begin\r\n");

  if (!manager.init())
    Serial.println("RF communication initialization failed");
  else
    Serial.println("RF communication initialized");

  Serial.print("RF setup complete\r\n");
}
```

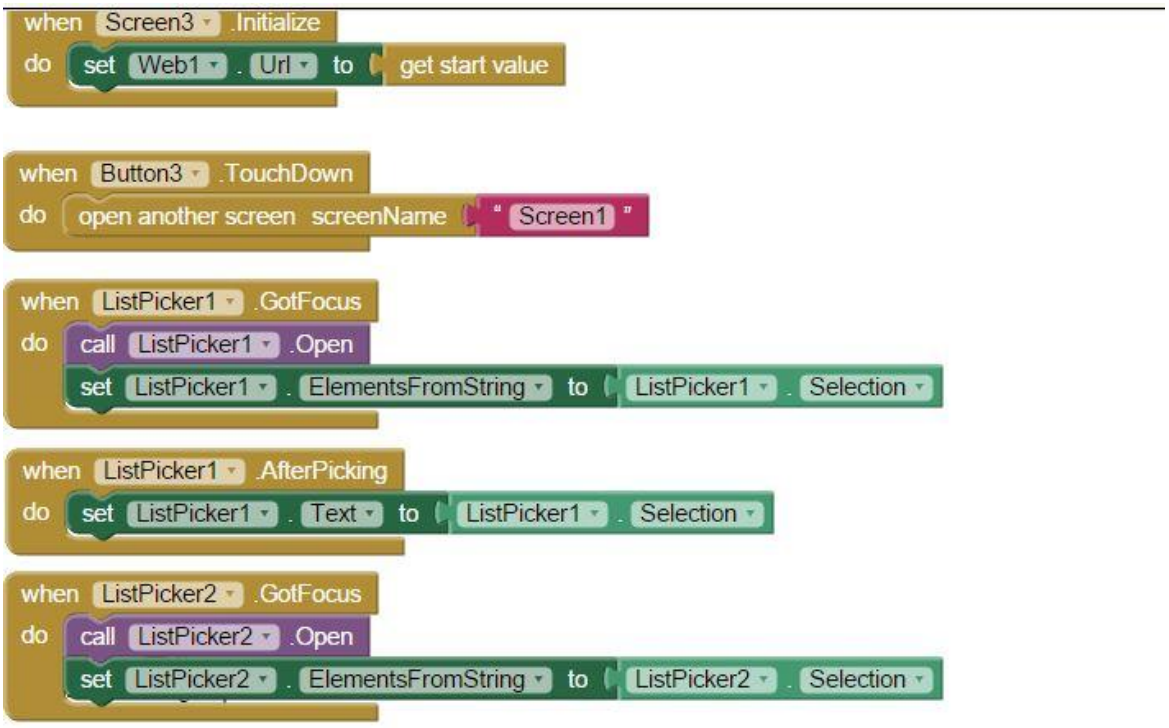
A1.3 ANDROID APP SCREENSHOTS



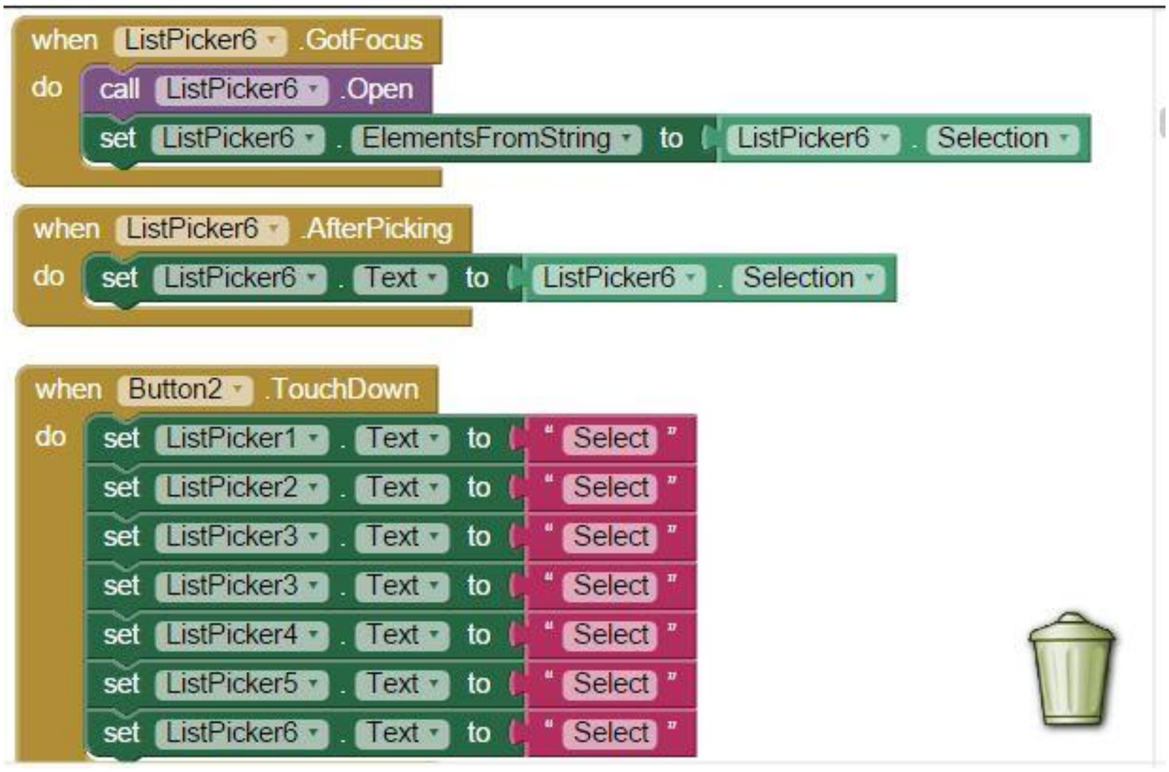
```
when Button5 .Click
do  open another screen  screenName  " HELP "

when Button3 .Click
do  set Web1 . Url to   join  " http:// "
                              TextBox1 . Text
    call Web1 .PostText
                   text  " $999999$ "
    call Web1 .PostText
                   text  " $999999$ "
    call Web1 .PostText
                   text  " $999999$ "
    call Web1 .PostText
                   text  " $999999$ "
    call Web1 .PostText
                   text  " $999999$ "
```



```
when Button4 .Click
do  open another screen with start value  screenName  " Screen3 "
                                          startValue  Web1 . Url

when SpeechRecognizer1 .AfterGettingText
 result
do  set Label3 . Text to  SpeechRecognizer1 . Result
    for each item in list  split at spaces  Label3 . Text
    do  if  is in list? thing  select list item  list  split at spaces  Label3 . Text
                                                 index  1
                        list  make a list  " bed "
                                           " Bed "
                                           " Best "
                                           " Red "
                                           " Bid "
```



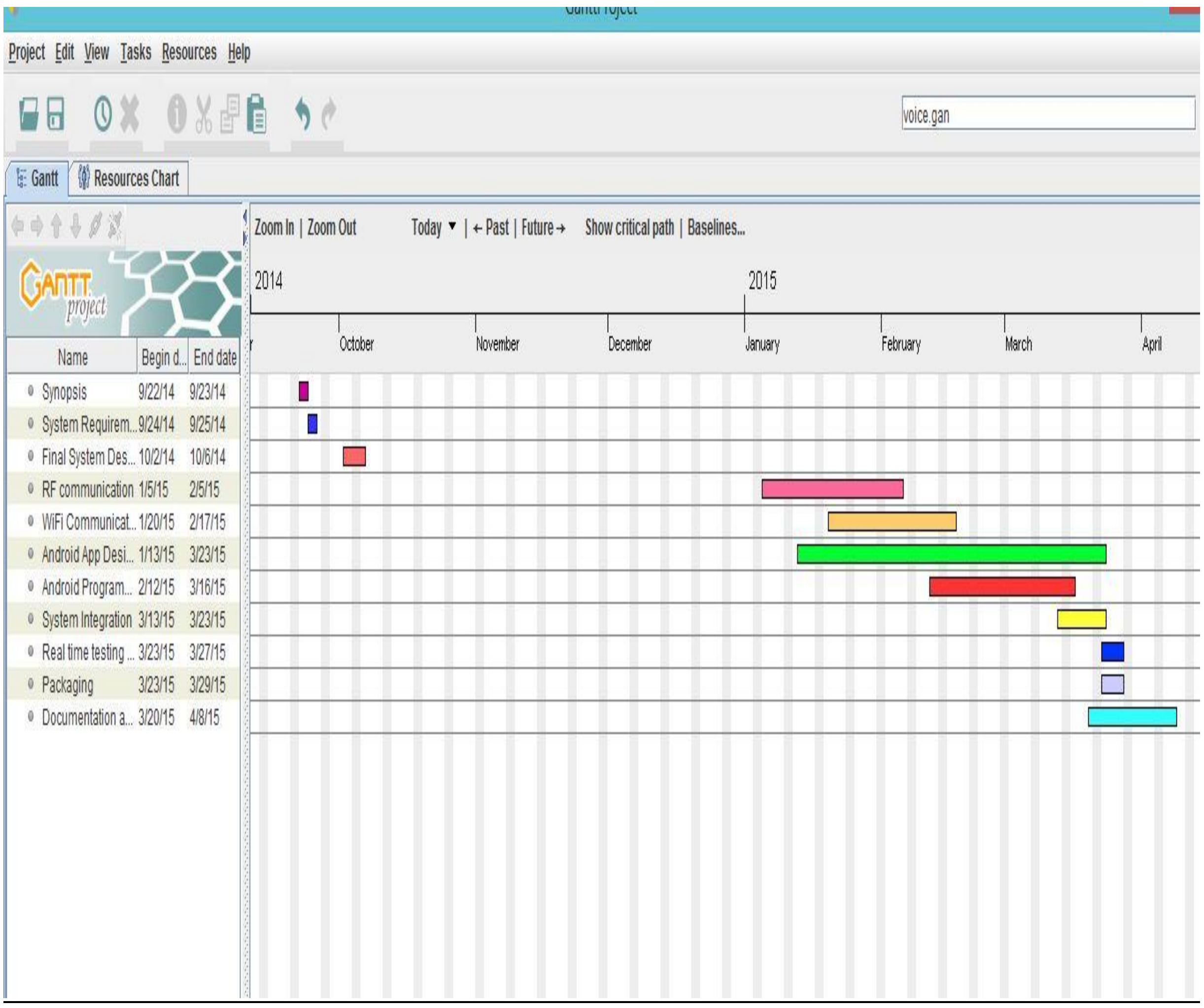
```
when Screen3 .Initialize
do  set Web1 . Url to  get start value

when Button3 .TouchDown
do  open another screen  screenName  " Screen1 "

when ListPicker1 .GotFocus
do  call ListPicker1 .Open
    set ListPicker1 . ElementsFromString to  ListPicker1 . Selection

when ListPicker1 .AfterPicking
do  set ListPicker1 . Text to  ListPicker1 . Selection

when ListPicker2 .GotFocus
do  call ListPicker2 .Open
    set ListPicker2 . ElementsFromString to  ListPicker2 . Selection
```



```
when ListPicker6 .GotFocus
do  call ListPicker6 .Open
    set ListPicker6 . ElementsFromString to  ListPicker6 . Selection

when ListPicker6 .AfterPicking
do  set ListPicker6 . Text to  ListPicker6 . Selection

when Button2 .TouchDown
do  set ListPicker1 . Text to  " Select "
    set ListPicker2 . Text to  " Select "
    set ListPicker3 . Text to  " Select "
    set ListPicker3 . Text to  " Select "
    set ListPicker4 . Text to  " Select "
    set ListPicker5 . Text to  " Select "
    set ListPicker6 . Text to  " Select "
```

A2. PROJECT MANAGEMENT

A2.1 ACTIVITIES AND GANTT CHART



Name	Begin d...	End date
Synopsis	9/22/14	9/23/14
System Requirem...	9/24/14	9/25/14
Final System Des...	10/2/14	10/6/14
RF communication	1/5/15	2/5/15
WiFi Communicat...	1/20/15	2/17/15
Android App Desi...	1/13/15	3/23/15
Android Program...	2/12/15	3/16/15
System Integration	3/13/15	3/23/15
Real time testing ...	3/23/15	3/27/15
Packaging	3/23/15	3/29/15
Documentation a...	3/20/15	4/8/15

Figure A2.1.1 GANTT CHART

The screenshot above is of the Gantt chart that was created and followed throughout the project.

A2.2 MILESTONES AND TARGETS

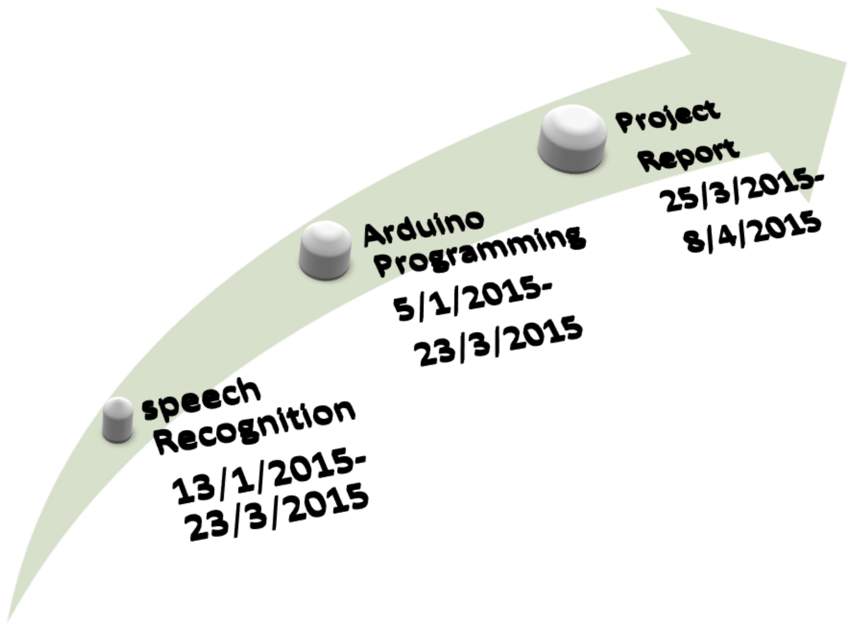


Figure A2.2.1 MILESTONES AND TARGETS

The figure above illustrates the various milestones that were setup at the start of the project and the tentative date by which we expected to achieve them. The detailed explanation of the milestones is provided in the upcoming chapters.

A2.3 WORK DISTRIBUTION TABLE

The various hardware and software based works which includes module testing, integration of various modules, App building and coding were divided among the team members present in the group and table shows the same.

SL NUMBER	TASK	TEAM MEMBERS
1.	Android application	Shridhar Vadeyar, Azaruddin M
2.	Testing and Integration of Wi-Fi module (ESP8266)	Bhargavi Deshpande, Shreyas P
3.	Testing and Integration of Wi-Fi module (ESP8266)	Bhargavi Deshpande, Shreyas P
4.	Code designing of Base Station	Bhargavi Deshpande, Shreyas P
5.	Code designing of Satellite station	Bhargavi Deshpande, Shreyas P
6.	Electrical Wiring and Switching	Shreyas P, Shridhar Vadeyar
7.	Packaging	Bhargavi Deshpande, Shreyas P, Shridhar Vadeyar
8.	Documentation	Bhargavi Deshpande, Shridhar Vadeyar

Figure A2.3.1 WORK DISTRIBUTION TABLE

USER MANUAL

A3. USER MANUAL

- Install the “FYP07” android application package on any of the android smart phone.
- Change the phone settings and allow the application to access your phone.
- Once installed open the Android app “FYP07” on your phone.
- Enter the IP address of the Wi-Fi module (ESP8266) that is connected to the same Wi-Fi network.
- Press ‘GO’ to establish a connection.
- To work in voice mode, press the Mike button and say any default commands. To know the default commands click on “HELP” button.
- To work in manual mode, press the “Manual” button and choose desired options and finally press the “Submit” button.
- Anytime to switch from manual mode to voice mode, press the “Back” button.