CHAPTER 1

INTRODUCTION

The current busy world people are not ready to conserve electrical energy because they should not have interest about power dissipation. This is a worthy application for those who can unable to return back to home for switch off their electrical devices and it give more dissipation in electrical energy. To overcome the dissipation of electric energy, home automation is used. The main purpose of this project is control home devices remotely with the help of voice command.

Here, the system is easiest way to access device in the home which can connect the WI-FI or Internet. The main purpose of this system IoT (Internet of Things), AI (Artificial Intelligence) and Google firebase which are remotely control the device via Wi-Fi or Internet. Google assistant is based on virtual assistant available on mobile, smart home devices and it is developed by Google. IoT is the network of the devices which is a system to connect physical objects that are accessible through the Internet. Google's mobile platform named firebase helps to develop a high quality app quickly. NodeMCU is an open source IoT platform which is a single board microcontroller with Wi-Fi enabled and also acts as an access points. This project can be done by using the Google assistant which is easily available in the smart phone. Therefore, no need of an external device for controlling home appliances

CHAPTER 2

LITERATURE REVIEW

2.1 OVERVIEW OF LITERATURE REVIEW

Yash Mittal et.al.,[1]- The voice-controlled, multi-functional, smart home automation system (SHAS), proposed in this paper can receive voice commands by a specific person and perform corresponding functions. The multiple functions could include controlling electrical appliances such as lights or fan, and electronic gadgets such as refrigerator or washing machines. Though the proposed SHAS improves accessibility to appliances through a natural interface, i.e., human voice, it has few limitations. Voice recognition module needs to be placed at a common location in the room, or at a common place if the resident intends to use it from different locations inside the house. Another limitation is the need of training the system for every user or home resident, in the case of multiple residents. Testing also needs to be carried out further for effectiveness of the voice-commands' recognition with background noise or natural noise such as storm or rains etc. In the prototype SHAS, the experiments are conducted to measure the effectiveness and range of voice-commands for each of the 5 voice-command groups. The results couldn't be compared with any similar study, as we haven't come across other similar applications.

Cyril joe baby et.al.,[2]- Home Automation system is designed and implemented using web page for mobile phone and developed to control the appliances in a house. The user can interact with the system using either a web application or voice commands

Arriany A et.al., [3]- smart home in terms of concept, applications, system components, and networking methods. It also investigated the introduction of voice recognition technology to control smart home components. The investigation includes a review to voice recognition basics in terms of system description, how it works, available software, and its obstacles. A simple miniature smart home model using voice recognition feature is proposed and implemented. The proposed model is detailed in terms of design, operation, testing and evaluation. The proposed model consists of a number of LEDs and

fan to be remotely controlled by voice commands using a remote PC that is connected wirelessly to the model's controller. The evaluation tests confirm the out-performance of using external microphone in a quiet environment over other scenarios. Also, results show that the shorter distance between wireless nodes the lesser command execution time as expected.

Ying-TsungLee et.al.,[4]- Smart home-service architecture, which employed standard interface devices at the home end to separate the logic and user interfaces, and achieving multiple in-home displays. Moreover, this study applied a community broker role to integrate smart home services such as managing environment deployment operations, reducing the manual labour required of community management personnel, providing electronic information services, supporting diverse services, and extending the community's integration

Murad Khan et.al.,[5]- The coexistence of heterogeneous networks and energy efficiency in the smart home environment stand front among the multiple challenges in the realization of HEMS. energy and interference aware smart home design. The proposed smart home system employed a CoZNET mechanism to mitigate the effect of interference caused due to the co-existence of WSN and WIFI networks. A light control system is developed to use natural light with the light source to reduce the energy consumption of the light source. The proposed smart home architecture is tested for both energy consumption by the devices and the interference caused due to the co-existence of WIFI and WSN networks

Chayapathy. V et.al.,[6]- The Raspberry Pi personal assistant does have scope for further improvement. Some areas of improvement for the Rasberry Pi (RasPi) Personal Assistant are: Improving the quality of voice recognition .The Echo employs a special setup for Far-field Voice Recognition which enables the user to speak from any direction as well as from large distances from the Echo. The RasPi personal assistant employs a simple webcam for speech recording. Due to design limitations, the range and direction of users' speech is limited. Improving Image processing: The Image processing algorithm used in the project has its limitations. It can only pick up fingers when there is a definite background foreground demarcation. Presence of a large number of objects and lighting variations can cause the algorithm to malfunction. Improving range of support and features :Amazon Echo has extensive app support in the form of Amazon Skills (a set of apps to

extend functionality) .The Raspberry Pi personal assistant however, though including a wide array of features, does not have such widespread app support.

Pasd Putthapipat et.al.,[7]- The platform Pi Frame shown the strong capability as a based framework for developing Voice Recognition Gateway for Home Automation, with the advantage of its flexibility and scalability to add more features to the platform. Additional hardware can be added for more features, safety, convenient and comfortable to control any home appliances in the house by voice command. With more training, it would become smarter, cover more features and make it convenience for the user to use.

Norhafizah bt Aripin et.al.,[8]- In conclusion, generally speech or voice recognition interface can be implemented in many applications. Home automation based on voice recognition has been built and successfully developed in this project. The voice recognition system in this project are identified until 20m of range to transmit the signal from the smart phone to the home appliances via Bluetooth. This system was targeted for elderly and disabled people.

Khan M et.al.,[9]- The design and implementation of a unique consumer device called a Sense Pod that will help consumers communicate with their smart homes using simple physical gestures like tapping or rolling. The proposed device is based on the ZigBee protocol which is one of the industry standards for smart homes. The proposed system can be seamlessly integrated with any smart home system that supports a USB interface. Another advantage of the system is the low power consumption and low cost; each device can be manufactured for less than $10 per device which is significantly less than the cost of many ZigBee enabled electrical outlets available today. This also implies that a homeowner can easily afford tens of such devices for their home. Future extensions include adding the use of RSSI to automatically determine the location of each device in the home. In addition, since ZigBee supports mesh networking, multiple Sense Pods can be programmed to enact complex interactions as well. For example, a large number of smart home commands can be generated by using two Sense Pods as drum sticks, and by issuing a sequence of tapping instructions to form various drumming patterns

Tharaniya soundhari M, et.al.,[10]- The implementation of the home automation systems is to monitor and control the home appliances. The home automation system is fully functional for switching applications and the home appliances are switched on the user interface is updated to reflect the current status. The home automation is designed

with security features such as user authentication for accessing the home automation system. The low-cost home automation is designed to control the devices remotely by using keyword matching the set of activities are determined to classify automatically. The activities such as the control of lighting, air conditioner, and fan digital to analog conversion sensors, light sensors. The implementation is based on Keyword matching, Speech recognition using SVM classifier, Control signal program (CSP).

2.2 CONCLUSIONS FROM THE LITERATURE REVIEW

From this literature survey, the interfacing of Google Assistance with ESP 8266 module and make it cost effective as home appliances control device. Also, this proposed System is efficient for aged persons to operate home appliances and devices on mobile applications.

CHAPTER 3

TECHNICAL DETAILS

3.1 SMART HOME AUTOMATION

Home automation is part of "The Internet of Things", also known as IoT. The way all the devices and appliances can be networked together to provide us with seamless control over all aspects of your home and more. Home automation has been around for many decades in terms of lighting and simple appliance control. Recently technology caught up with the idea of the interconnected world at the touch of your fingertips or a simple voice command. One of the most targeted areas for the IoT is Home Automation. Home Automation, also called Domestic Computing, enables a user to control home devices using a remote control.

Controlling the devices at home using voice recognition. The light, fans that are controlled via a dedicated voice control. Most of these physical devices are now implemented as software apps running on smart phone, tablets and PCs. So far, the idea of controlling homes with the use of a button-driven control or voice control unit has been widely accepted by the world. Now a day, the technology for making homes smart, malleable, fast, and responsive with less errors and crashes is ready, and so are its consumers. The homes of today are getting ready to being controlled by a single universal smart remote. People are getting smart enough to automatically decide, how appliances are able to interact or how to adapt to their owner's requirements. There are already smart thermostats and sensors setup in modern homes, which can learn and study their users' needs and preferences.

Imagine where the smart home talks to the user when the user needs to be reminded. The home would understand the users' daily needs and takes care before the user gives any command. For example, if the smart home reminds the user to take his office folder before leaving to work. Another example, the smart home switch on the light or fan, when the user is leaving through the front door. Imagine if the smart home can do all of the above stuff without asking user for it. The above can be achieved via IoT

integration in home automation system. IoT becomes really important in understanding the environments and interacting with its dedicated network

3.2 BLOCK DIAGRAM

These systems and devices within a home can also be managed either from inside home, or can be linked to services and systems from outside the home are shown in figure 3.1

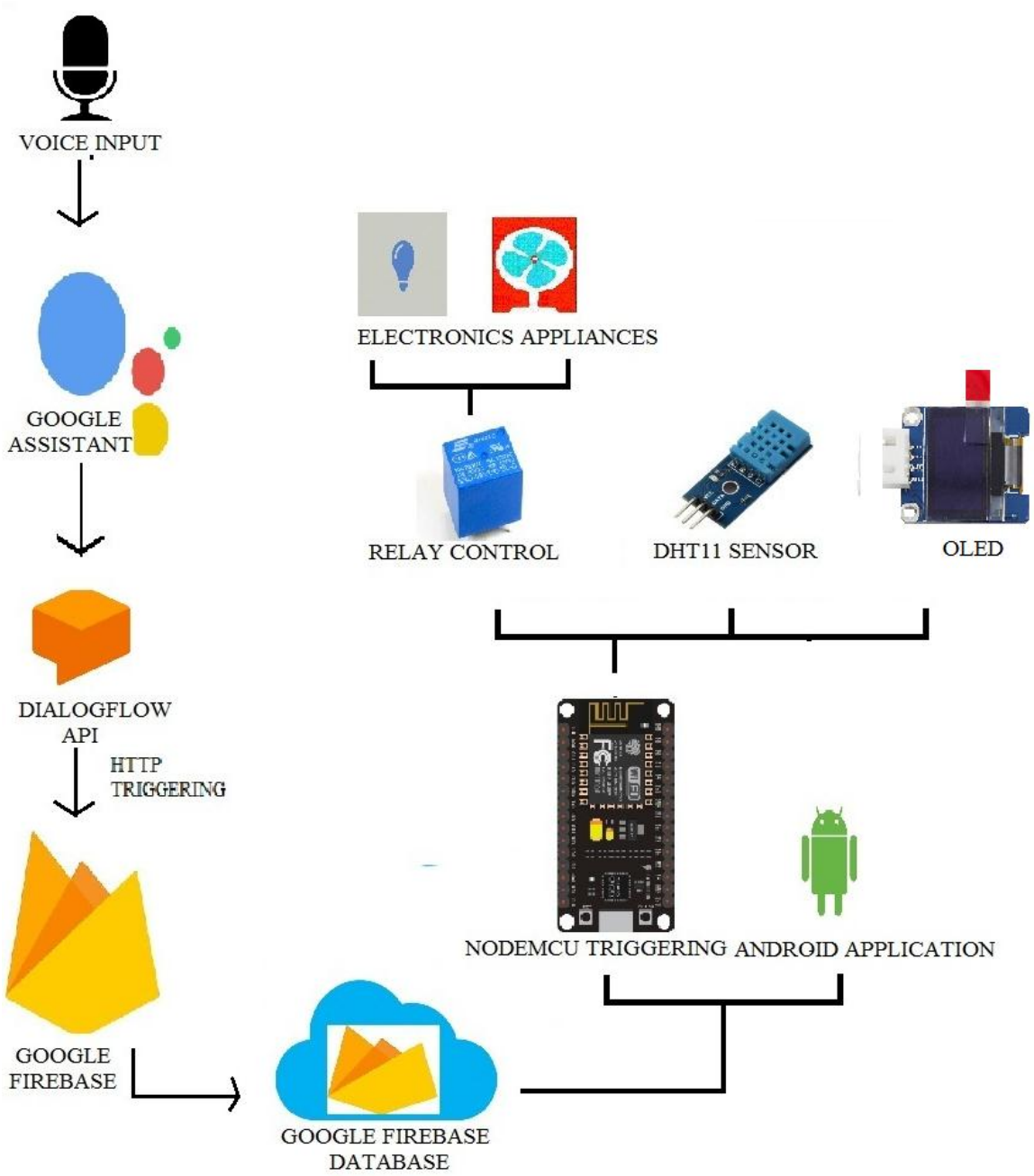

Figure 3.1: Google Assistant Relay Based Controller

3.3 BLOCK DIAGRAM DESCRIPTION

3.3.1 Voice Input

This system focuses on easy control in a smart home through voice, a natural user interaction. The paper implements a voice-controlled SHAS for reduction of human effort and minimising any human movement. In a multi-purpose SHAS, one can switch-OFF and ON any electrical household appliance using voice-commands. The voice-controlled system can also help the visually-impaired in controlling their home appliances. The SHAS is designed such that it is easy to install and use. Voice recognition differs distinctly speech recognition. In speech recognition the subject of analysis is the spoken text, while in voice recognition the subject of analysis is the voice of speaker and the spoken text remains. Thus, voice recognition is better for controlling and accessing the appliances then process the voice-commands for further actions.

3.3.2 Google Assistant

Google Assistant is an artificial intelligence-powered virtual assistant developed by Google that is primarily available on mobile and smart home devices. Unlike the company's previous virtual assistant, Google Now, Google Assistant can engage in two-way conversations. Users primarily interact with Google Assistant through natural voice. In the same nature and manner as Google Now, the Assistant is able to search the Internet, schedule events and adjust hardware settings on the user's device, and show information from the user's Google account.

3.3.3 Dialogflow API

Dialogflow API (Application Program Interface) is conservation of natural language technology and is owned by Google for human and computer interaction. To develop the assistant by using speaktoit and it performs the task and user interface in natural language. In the processing of natural language system that also can been mould the speaktoit and it is perform the task and user interface in natural language.

Speaktiot is a developer of third party that have been released the api.ai that allows the voice enable system by Google assistant. In the addition of speaktiot that allowing the apps is deploy voice recognition in the natural language. The SDK’s is a software development kit should have the voice interfaces to understanding the natural language

processing is used to convert the text to speech in the web interface of api.ai. It's mainly focused on process of natural languages in the application of assistant by speaktoit.Api.ai that allows only the natural languages that include the IoT. This speaktiot and Google assistant that allows and redirect the Dialogflow website.

Speaktoit released api.ai (the voice-enabling engine that powers Assistant) to third-party developers, allowing the addition of voice interfaces to apps based on commands from user. Figure 3.2 show the SDK's contain voice recognition, natural language understanding, and text-to-speech. api.ai offers a web interface to build and test conversation scenarios. The platform is based on the natural language processing engine built by Speaktoit for its Assistant application. Api.ai allows Internet of Things developers to include natural language voice interfaces in their products. Assistant and Speaktoit's websites now redirect to api.ai's website, which redirects to the Dialogflow website.

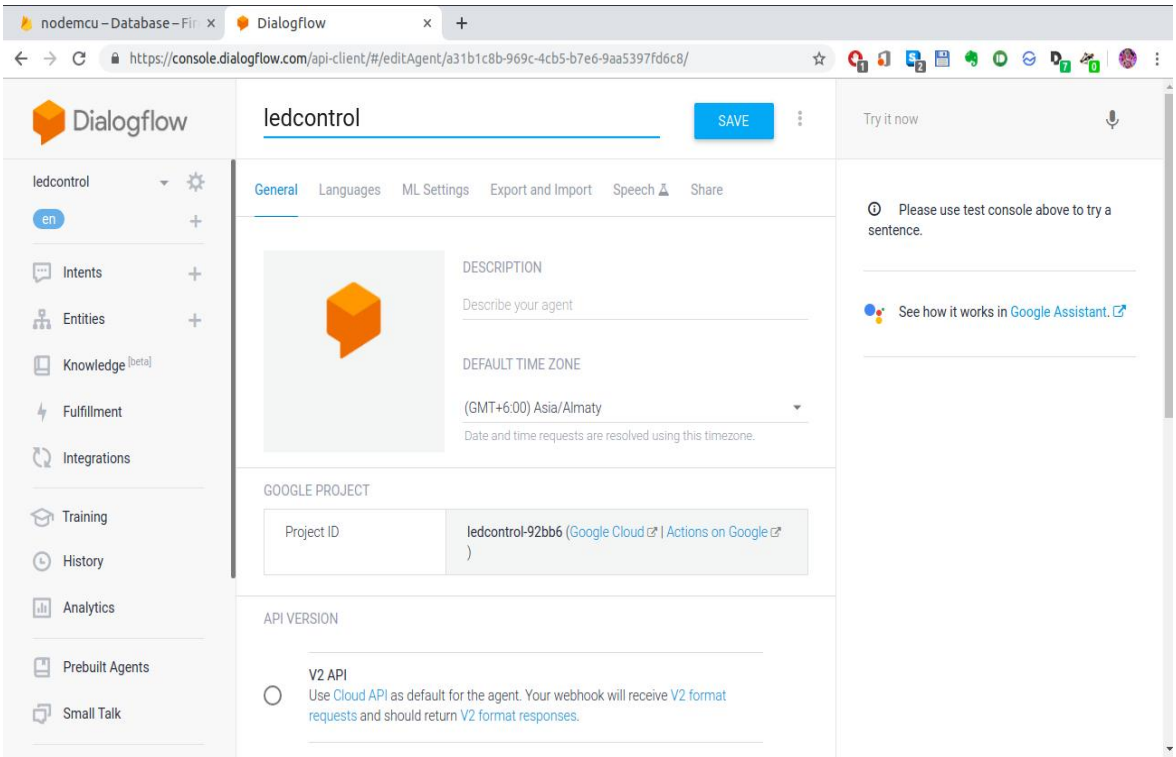

Figure 3.2: Dialogflow API console

Entity: The Name set in the entity will match the Firebase parameter which the device wants to be controlled. It contains the reference value and synonyms which needs to be given by the user so string is generated and connected with google firebase are shown in figure 3.3.

Dialogflow defines system entities, which are pre-built entities that correspond to commonly used categories like colour, time, and city names. You can also create developer-defined entities when you want Dialogflow to recognize a certain category of things that isn't represented by a system entity. For more information about entity types, see Entities.

Entities are very important in the intent matching process, as they give Dialogflow more information when trying to match utterances that don't exactly correspond to training phrases

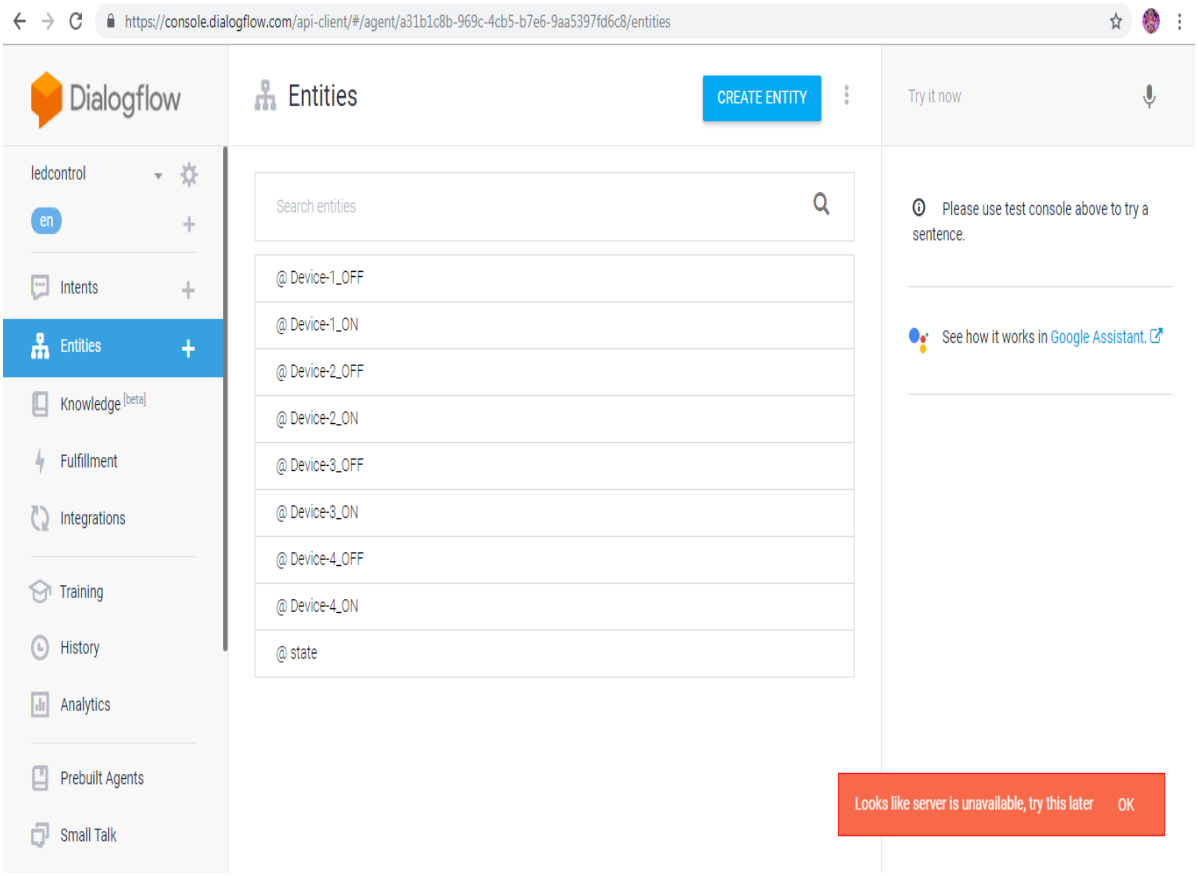

Figure 3.3: To Create Entity in Dialogflow API

Intents: It will allow the user to set the possible commands of infinite combination of sequence like ‘turn on’, ‘power on’, ‘switch off’ after the entity. It is just like a training the Google Assistant.

To define how conversations work, you create intents in your agent that map user input to responses. In each intent, you define examples of user utterances that can trigger the intent, what to extract from the utterance, and how to respond.

Intent components

Intents consist of four main components that allow you to map what your user says to what your agent responds with. These components include the following:

Intent name: The name of the intent. The intent name is passed to your fulfillment and identifies the matched intent are shown in figure 3.4.

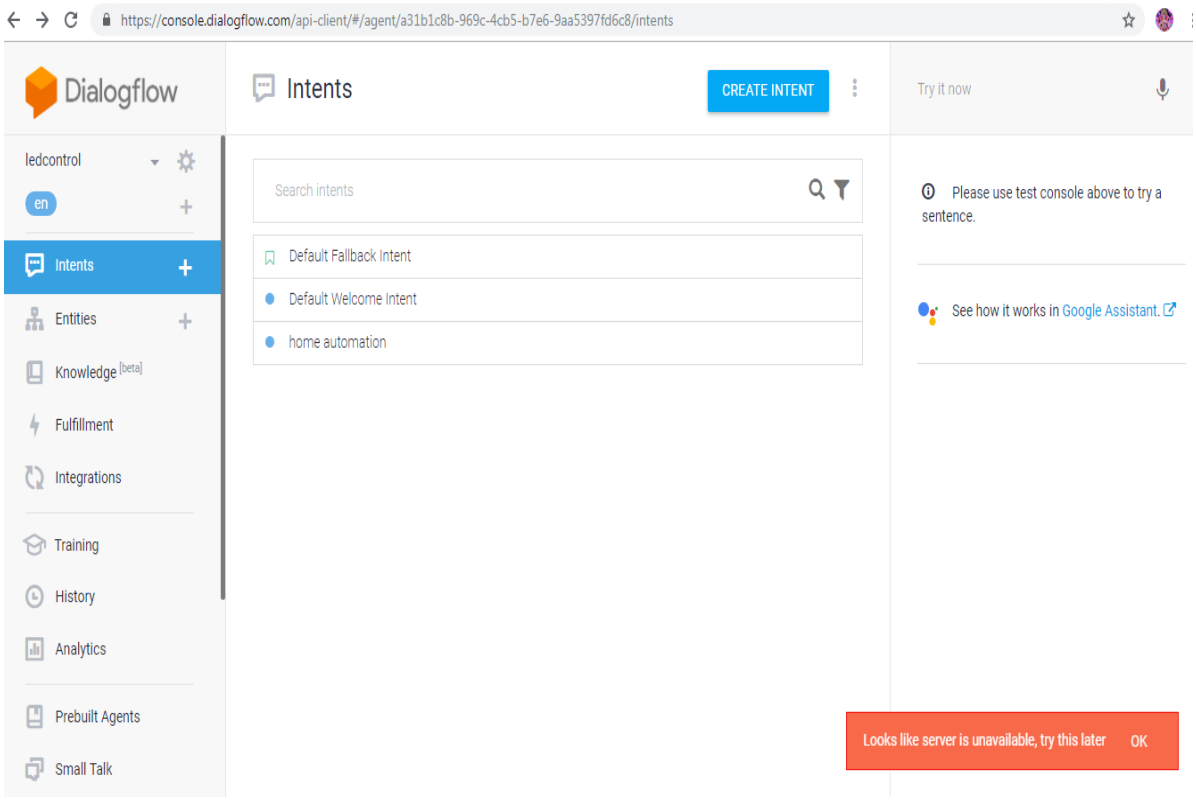

Figure 3.4: Creating Intent Name

Training phrases: Examples of what users can say to match a particular intent. Dialogflow automatically expands these phrases to match similar user utterances are shown in figure 3.5

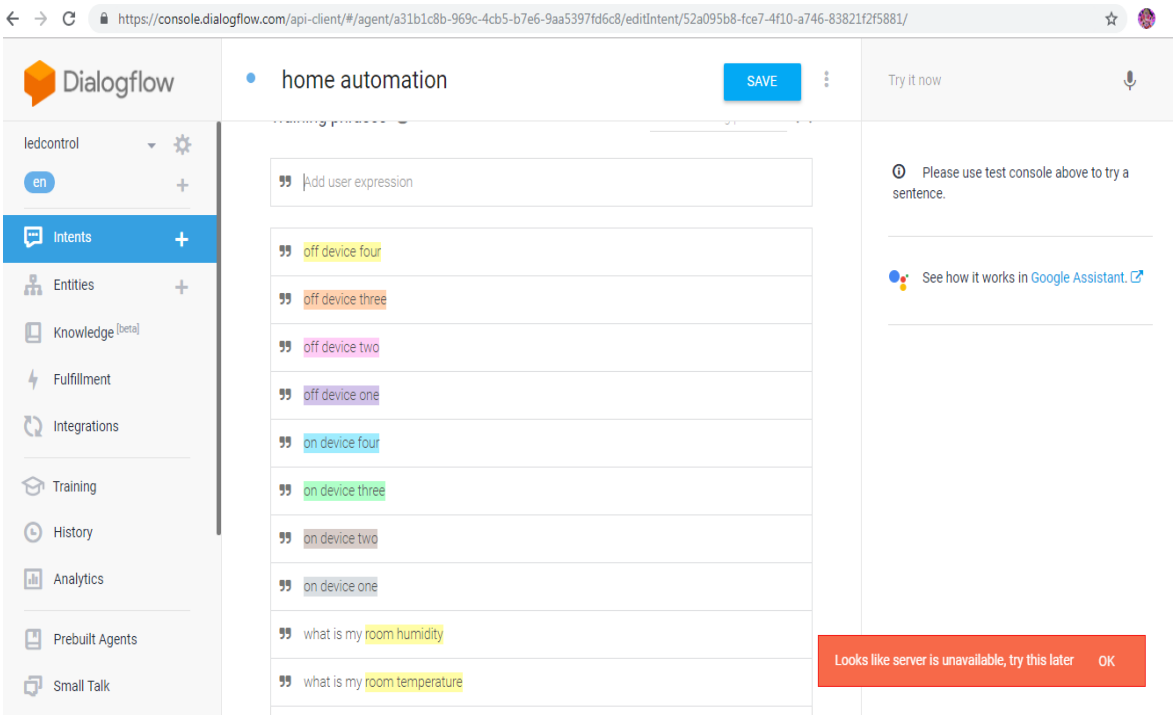

Figure 3.5: Training Phrases

Action and parameters: Defines how relevant information (parameters) are extracted from user utterances. Examples of this kind of information include dates, times, names, places, and more. You can use parameters as input into other logic, such as looking up information, carrying out a task, or returning a response are shown in figure 3.6

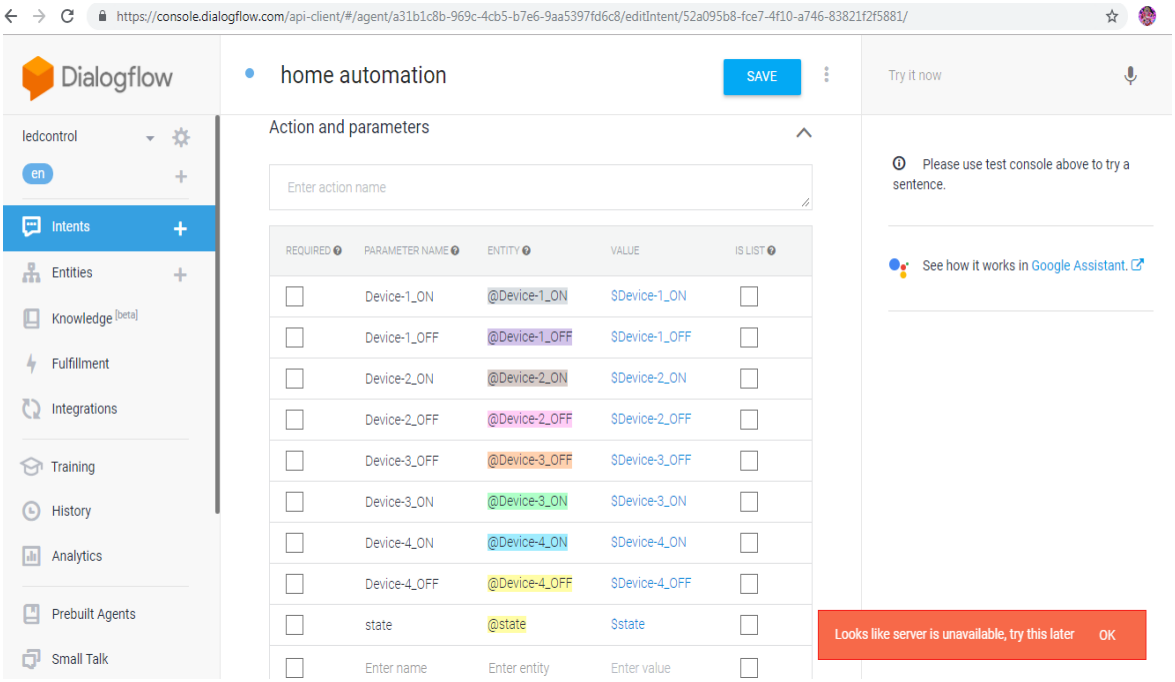

Figure 3.6 : Action and Parameter in Intent

Response: An utterance that's spoken or displayed back to the user are shown in figure 3.7.

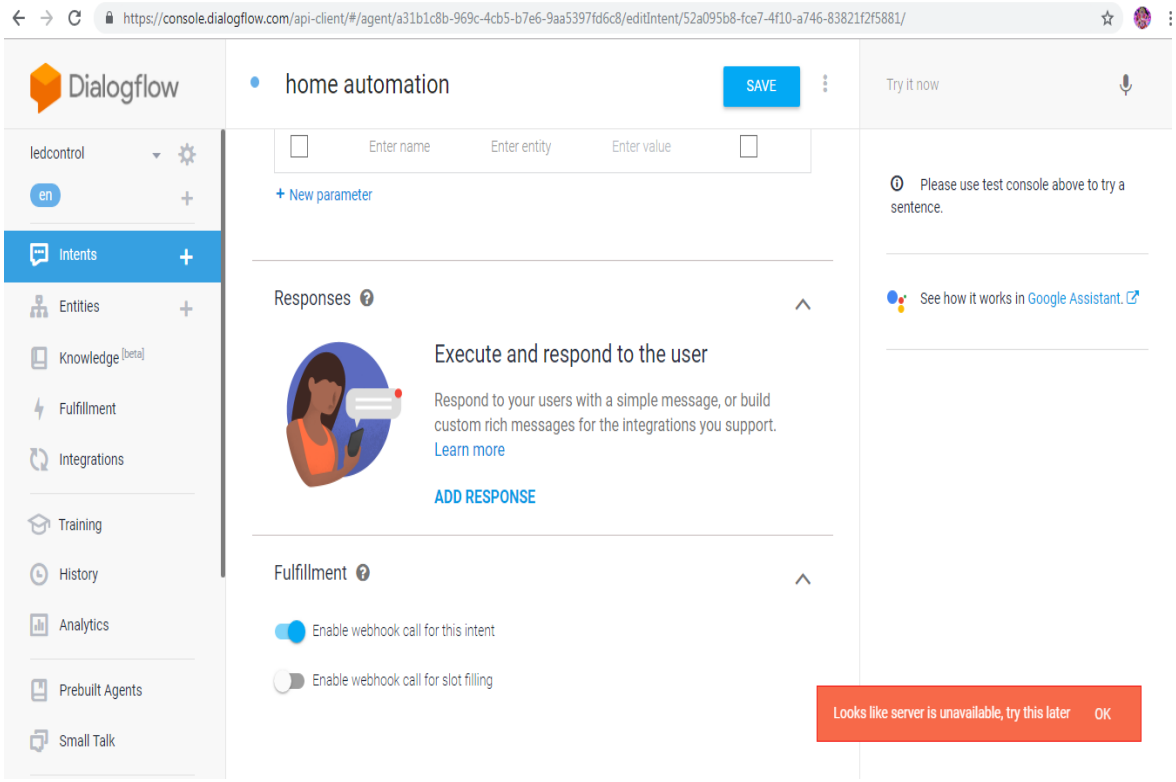

Figure 3.7: Intent Response

Fulfilment: If you want to return dynamic responses to the user, you must use fulfillment, which is code that is deployed as a webhook and responds to HTTP requests from Dialogflow. Your fulfillment code processes the information from the matched intent and constructs a response to return to the user. More specifically, once user input is parsed, Dialogflow sends the name of the intent that was matched, the values of extracted parameters, and other metadata as a JSON payload in an HTTP POST request to your webhook are shown in figure 3.8. Your webhook constructs a response and sends it as a JSON payload to your Dialogflow agent, which then delivers the response to the user.

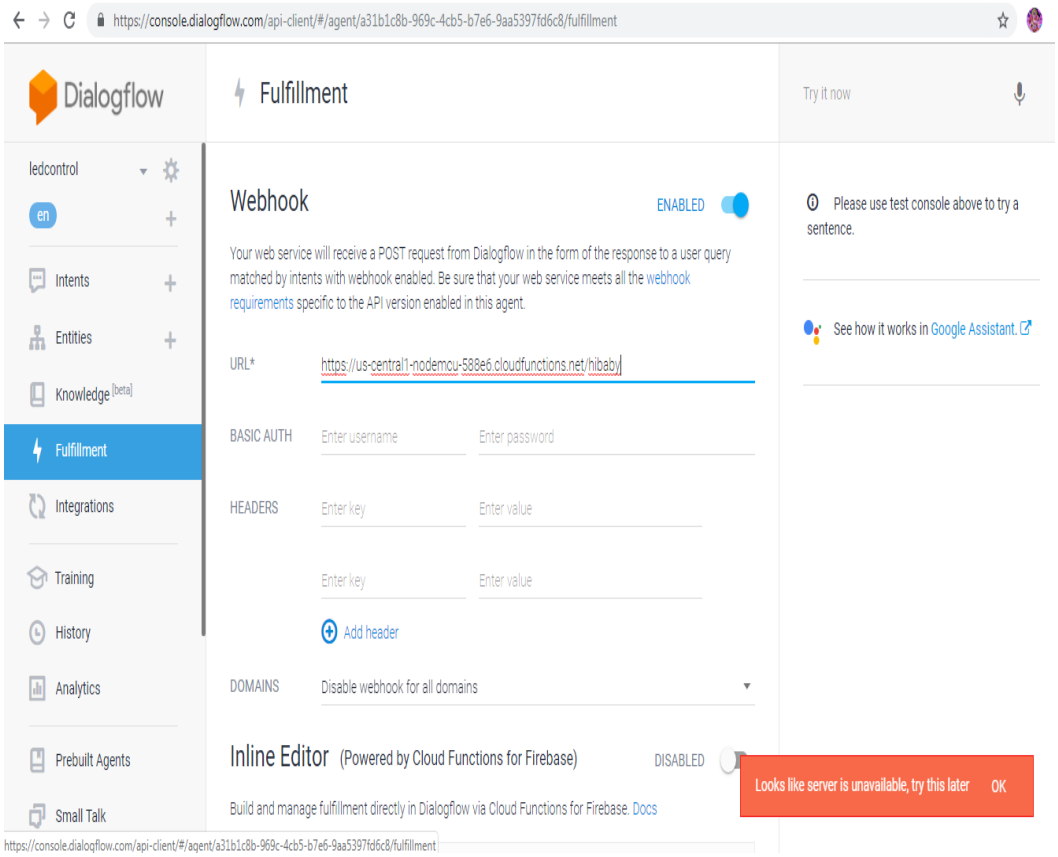

Figure 3.8: HTTP Post Request to Webhook

Integrations: It Integrate with the Google Assistant, which lets you deploy your Dialogflow agent as actions that users can invoke through the Assistant are shown in figure 3.9

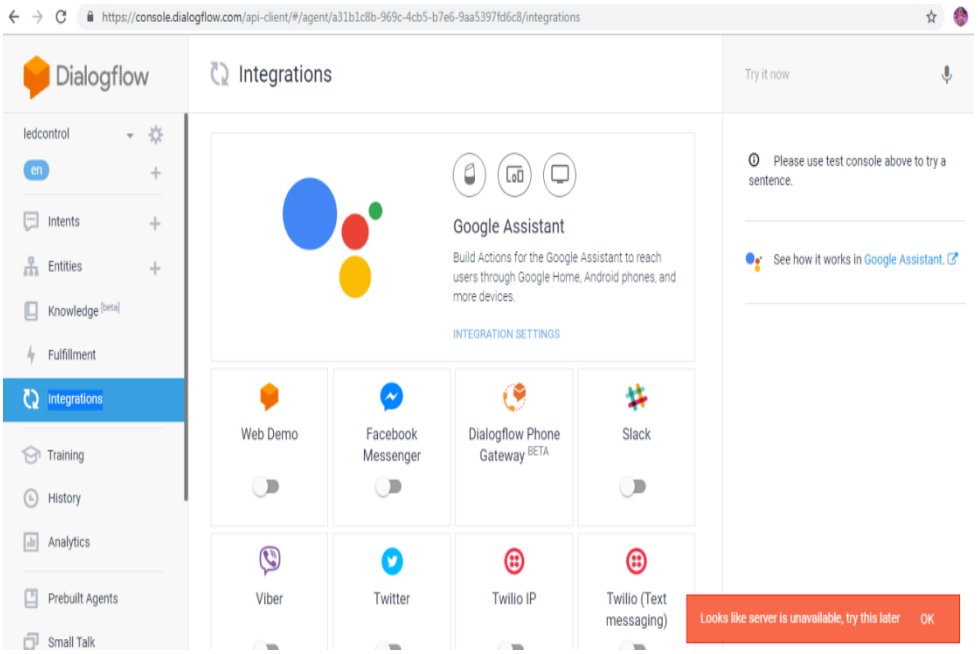

Figure 3.9: Integrating with Google Assistant

3.3.4 Google Firebase

Firebase is based on real time cloud managing system they are also known as Google Cloud Messaging system and its directly allow the database to access code from user side. Firebase cloud messaging have two components in their server side. First server is provided by the Google and another one is app server function that based on the firebase in the cloud function are shown in figure 3.10. They sync easily developer by API in firebase and they are stored data in real time and to manage the server side code to build their apps without having the server can use developers.

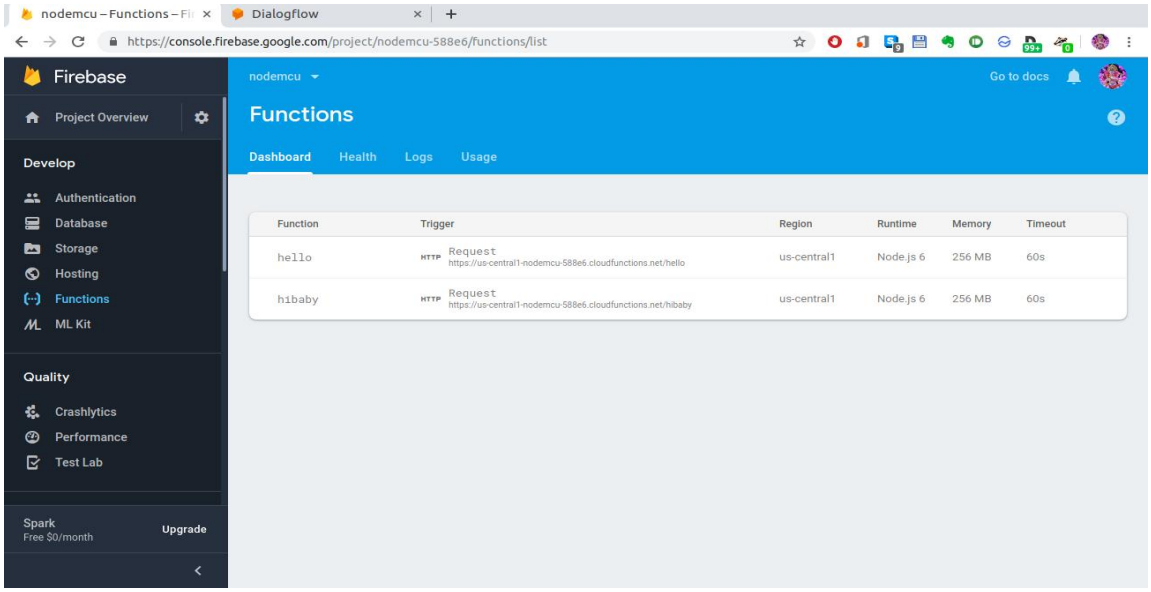

Figure 3.10: Google Firebase Function Implementation

The Firebase will show the newly updated data and triggering process in time to time. Then hardware library of Google Firebase will trigger in the real time and control the relay which in turn controls the appliances. It has the cloud function to initiate the triggering actions. Also the real time data connectivity will pass the string on to NodeMCU makes the function relay functioning efficiently. As the all cloud functioning takes place in Google makes the System faster and secure. The hardware connection is simple and it uses the supply efficiently. The available library of Google Firebase with NodeMCU makes Wi-Fi reception easy and faster than other cloud database. The data updating is faster and responsive.

3.3.5 Google Firebase Database

The Firebase Real time Database is a cloud-hosted database. Data is stored as JSON and synchronized in real time to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Real time Database instance and automatically receive updates with the newest data. The Firebase Real time Database can be accessed directly from a mobile device or web browser; there's no need for an application server. Security and data validation are available through the Firebase Real time Database Security Rules, expression-based rules that are executed when data is read or written

The Cloud Function type for HTTPS triggers. This should be exported from your JavaScript file to define a Cloud Function. This type is a special JavaScript function which takes Express Request and Response objects as its only arguments. HTTP requests the Firebase Real time Database uses data synchronization every time data changes, any connected device receives that update within milliseconds. Provide collaborative and immersive experiences without thinking about networking code.

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 not a SQL database and as such has different optimizations and functionality compared to a relational database. The Real time Database API is designed to only allow operations that can be executed quickly. 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.

3.3.5.1 Set up Node.Js and the Firebase Client

- Node.js environment to write functions and the Firebase CLI (which also requires Node.js and npm) to deploy functions to the Cloud Functions runtime.
- Cloud Functions can run on either Node version 6, or Node version 8. Have existing functions built on Node version 6 you can continue to use that version (version 6 is default). See Set runtime options to learn more.

- For installing Node.js and npm, Node Version Manager is recommended. Once have Node.js and npm installed, install the Firebase CLI via npm:

  npm install -g firebase-tools

- This installs the globally available firebase command. If the command fails, need to change npm permissions. To update to the latest version of firebase-tools, rerun the same command.

3.3.5.2 Initialize Firebase SDK for Cloud Functions

When initialize Firebase SDK for Cloud Functions, create an empty project containing dependencies and some minimal sample code and choose JavaScript for composing functions.

To Initialize Project

- Run firebase login to log in via the browser and authenticate the firebase tool.
- Go to your Firebase project directory.
- Run firebase init functions. The tool gives you an option to install dependencies with npm. It is safe to decline if you want to manage dependencies in another way.
- The tool gives two options for language support:
  - Java script
  - Typescript.

3.3.5.3 Import the Required Modules and Initialize an App

After completed the setup tasks, open the source directory and start adding code as described in the following sections. For this sample, project must import the Cloud Functions and Admin SDK modules using Node require statements. Add lines like the following to index.js file.

The lines load the firebase-functions and firebase-admin modules and initialize an admin app instance from which Realtime Database changes can be made. Wherever Admin SDK support is available, as it is for FCM, Authentication, and Firebase Realtime Database, it provides a powerful way to integrate Firebase using Cloud Functions.

The Firebase CLI automatically installs the Firebase and Firebase SDK for Cloud Functions Node modules when you initialize your project. To add 3rd party libraries to your project, you can modify package.json and run npm install. For more information, see Handle Dependencies.

3.3.5.4 Add the Message Function

The add Message () function is an HTTP endpoint. Any request to the endpoint results in Express JS-style Request and Response objects passed to the on Request () call back.

HTTP functions are synchronous (similar to callable functions), so should send a response as quickly as possible and defer work using the Realtime Database. The add Message () HTTP function passes a text value to the HTTP endpoint and inserts it into the Realtime Database under the path /messages/:push ID/original using the previously initialized admin app.

3.3.5.5 Deploy and Execute Add Message

To deploy and execute the add Message () function, follow these steps:

- Run this command to deploy your functions:

  $ firebase deploy --only functions

- After you run this command, the Firebase CLI outputs the URL for any HTTP function endpoints. In your terminal, you should see a line like the following:
- Function URL (add Message):

  https://us-central1-MY_PROJECT.cloudfunctions.net/addMessage

- The URL contains project ID as well as a region for the HTTP function. Don't need to worry about it now, some production HTTP functions should specify a location to minimize network latency.
- Add a text query parameter to the add Message () URL, and open it in a browser:

  https://us-central1 MY_ PROJECT, Cloud functions .net/ add Message? text= uppercase

- The function executes and redirects the browser to the Firebase console at the database location where the text string is stored. You should see your text value displayed in the console.
- After deploying and executing functions, can view logs in the Firebase console.

3.3.5.6 Add the Make Uppercase Function

- The make Uppercase () function executes when the Realtime Database is written to. The ref (path) function defines the part of the database to listen on. For performance reasons, you should be as specific as possible.
- Braces—for example, {push Id}—surround "parameters," wildcards that expose their matched data in the call back.
- The Realtime Database triggers the on Write() call back whenever data is written or updated on the given path

Event-driven functions such as Realtime Database events are asynchronous. The call back function should return either a null, an Object, or a Promise. Do not return anything, the function times out, signalling an error, and is retried. See Sync, Async and Promises.

3.3.5.7 Deploy and Execute Make Uppercase

To complete the tutorial, deploy your functions again, and then execute add Message () to trigger make Uppercase ().

- Run this command to deploy your functions:

  $ firebase deploy --only functions
- If you encounter access errors such as "Unable to authorize access to project," try checking your project aliasing.
- Using the add Message () URL output by the CLI, add a text query parameter, and open it in a browser:

  https://us-central1-MY_PROJECT.cloudfunctions.net/addMessage?

  text=upper case me too

- The function executes and redirects the browser to the Firebase console at the database location where the text string is stored. This write event triggers make Uppercase (), which writes an uppercase version of the string.
- After deploying and executing functions, you can view logs in the Firebase console for Cloud Functions. If you need to delete functions in development or production, use the Firebase CLI.

In the Google firebase database they show that which devices is connected and also show that room temperature and humidity are shown in Figure 3.11

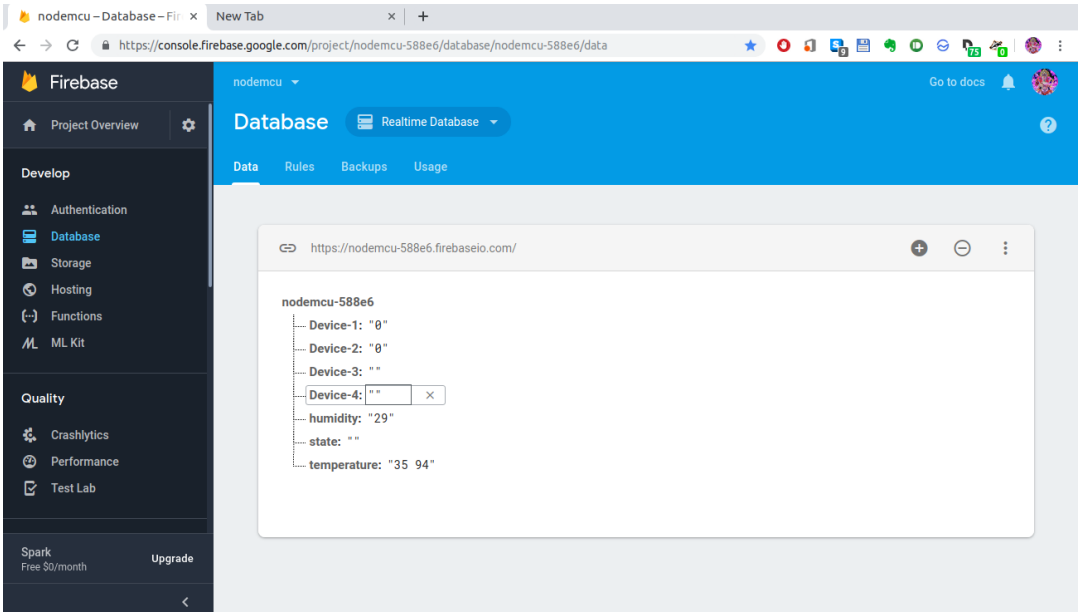

Figure 3.11: Google Firebase Database

3.3.6 Android Application

The android application is used to listen value from a particular child node in realtime database. It is also used to set the value from a particular child node in realtime database. Firebase in mobile platform helps quickly to develop high-quality apps, grow user's database. Firebase is made up of complementary features that mix-and-match to fit the application developer needs, with Google Analytics for Firebase at the core. Here the end user can explore and integrate Firebase services in the app directly from Android Studio using the Assistant window .The additional features in the android application

developed here is to monitor the room temperature and humidity which is asked by the consumer through google assistant. Today, majority of apps need backend,so here it is developed as the real time time monitoring and real time appliances controlling mobile application. The main benefit of this application is the users' application data will be available from any device that login with same authentication. The easy and simple way to use Firebase is creating android app demo with login/register (with Firebase authentication) by using Firebase Email & Password authentication.The key capabilities of interfacing Google firebase with android application are Realtime,offline,accessibility from client devices and security.

In order to know which Firebase database in Android is, It basically offers a set of authentication option right out-of-the-box. It means, Firebase automatically save your app users' data. It is actually a good benefit for Android developers as it separates user data from application data and allows you to focus on user interface & experience of your Android app. Android applications of controlling appliances in home is shown in figure 3.15.

3.3.6.1 Implementing Android Application with Google Firebase

- Select the Form Factor to develop the firebase application.
- Select Empty Activity and click on Next button.
- Write Activity name for the application project and click on Finish button.
- Once the project is created in Android Studio, it's time to create a new project on the Firebase console.
- First, to go to https://firebase.google.com/ and make an account to gain access to Firebase console. After getting access to the console, the developer start creating first project.
- After the new project is created in Firebase, Setup Sign-In method by using email and password is shown in figure 3.12.

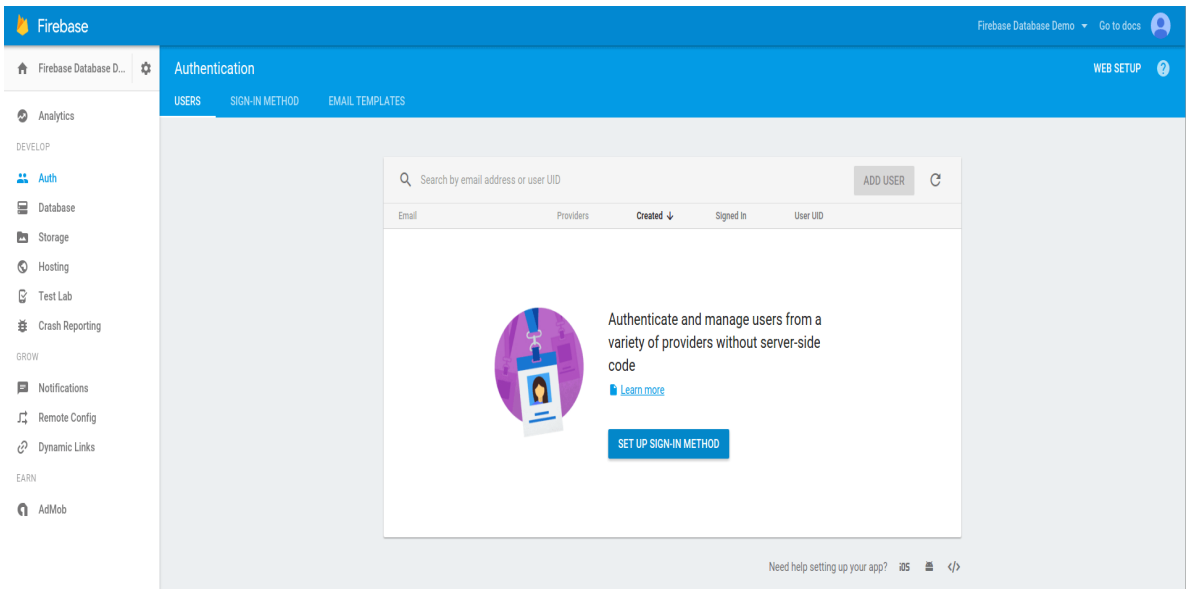

Figure 3.12: Firebase authentication setup with user mail ID and password

- Click the setup sign in method, the next panel will open where the user's password is updated and saved is shown in figure 3.13.

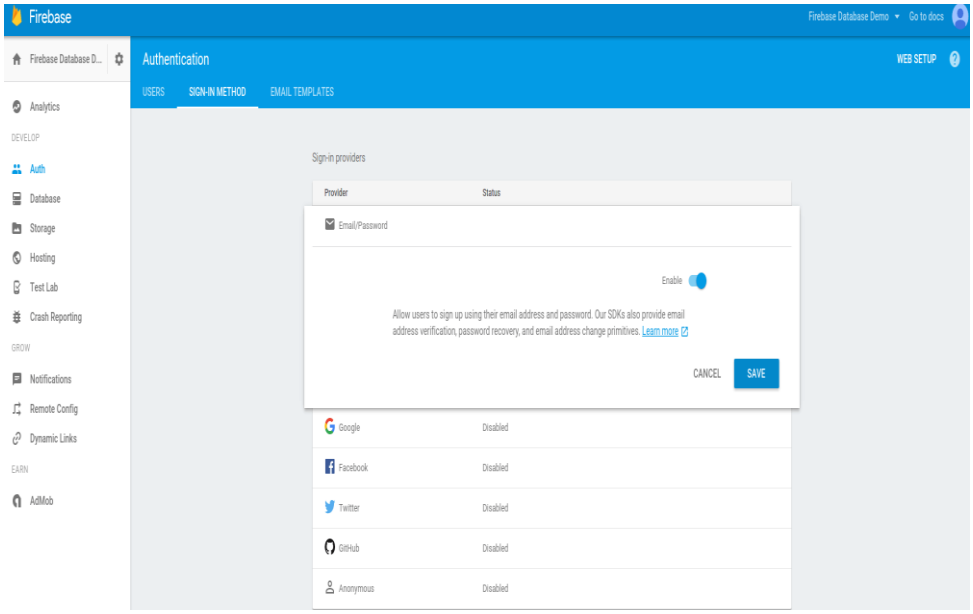

Figure 3.13: Firebase account with updated user mail ID and password.

- Click save to see the updated password of the user for second step authentication which is shown in Figure 3.14.

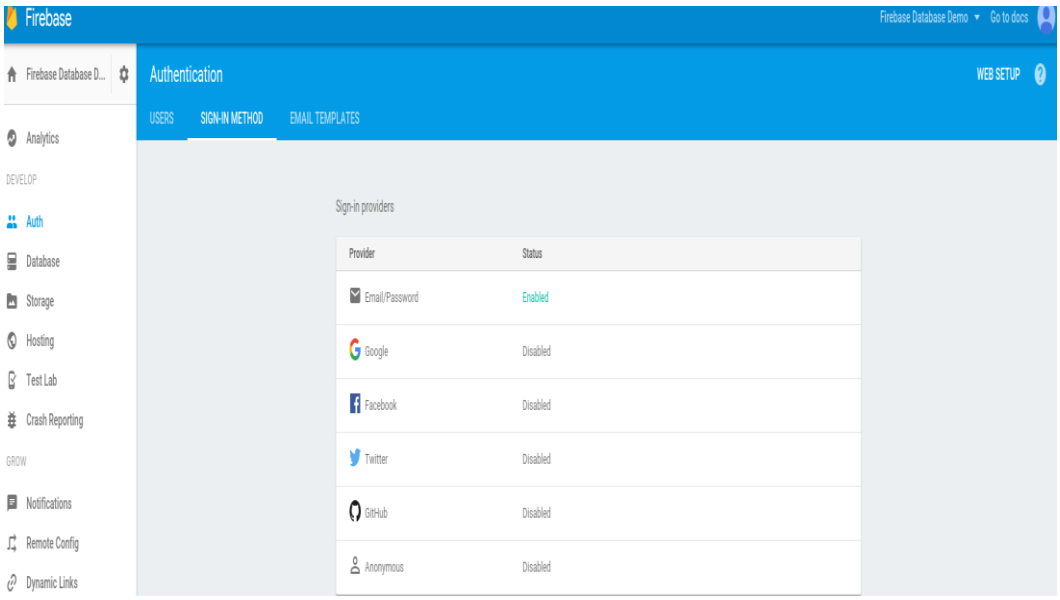

Figure 3.14: Confirmatory window for updated user password for next step authentication.

- After setting up the Sign-In method, open Firebase tools.
- Implement the authentication procedure.
- Next, after implementing authentication, then implement the Real-time Database.
- Once the Real-time database is added, set the database security rules.

Figure 3.15: Android application for controlling appliances in home.

3.3.7 NodeMCU

NodeMCU is an open source IoT platform device are shown in figure 3.16. It includes firmware which runs on the ESP8266 Wi-Fi So C from Espress if Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits

The ESP8266 based Feather HUZZAH & the HUZZAH ESP8266 breakout are both very popular options for connecting projects to Arduino IDE. In this guide we are going to walk through the setup needed to get your ESP8266 up and running with the Arduino IDE. This same basic setup can be used as you progress through our Basics series of guides Arduino IDE. Before you continue with this guide, you should consider running through the guides for the ESP8266 Feather or the ESP8266 breakout. We will cover all of the basic setup needed for connecting your ESP8266 to Arduino IDE.

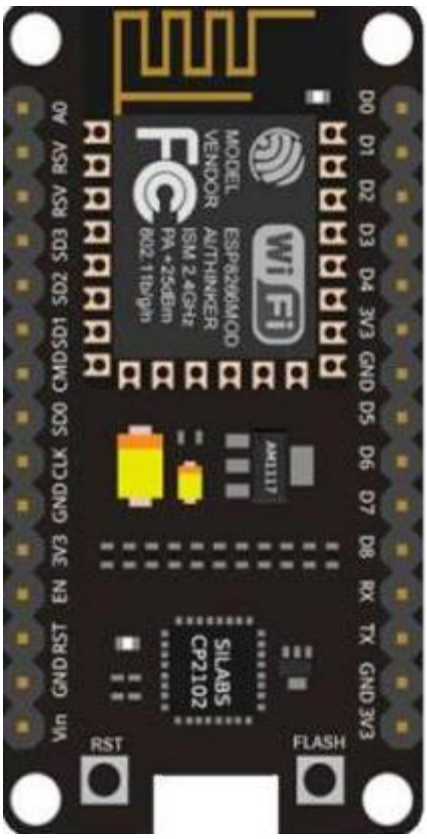

Figure 3.16: NODEMCU

3.3.8 DHT11 Sensor

The DHT11 is commonly used Temperature and humidity sensors are shown in figure 3.17**.** The sensor comes with a dedicated NTC to measure temperature and an 8-bit microcontroller to output the values of temperature and humidity as serial data. The sensor is also factory calibrated and hence easy to interface with other microcontrollers.

The sensor can measure temperature from 0°C to 50°C and humidity from 20% to 90% with an accuracy of ±1°C and ±1%. So if you are looking to measure in this range then this sensor might be the right choice for you.

Figure 3.17: DHT11 Sensor

3.3.9 Organic Light Emitting Diode (OLED)

SSD1306 is a single-chip CMOS OLED/PLED driver with controller for organic / polymer light emitting diode dot-matrix graphic display system which is shown in figure 3.18. It consists of 128 segments and 64commons. This IC is designed for Common Cathode type OLED panel. The SSD1306 embeds with contrast control, display RAM and oscillator, which reduces the number of external components and power consumption. It has 256-step brightness control. It is suitable for many compact portable applications, such as Smart watch, Real-time image display of camera on smart car, Battery management device, etc. The OLED display uses two types of protocols to display the data format with correct font,size and colour ,they are I2C and SPI protocols .But mostly the I2C type of OLED display is largely available in the market than SPI type of OLED's.As an I2C uses two wire for its communication than SPI which uses four wires for its communication which makes the Hardware complexity.

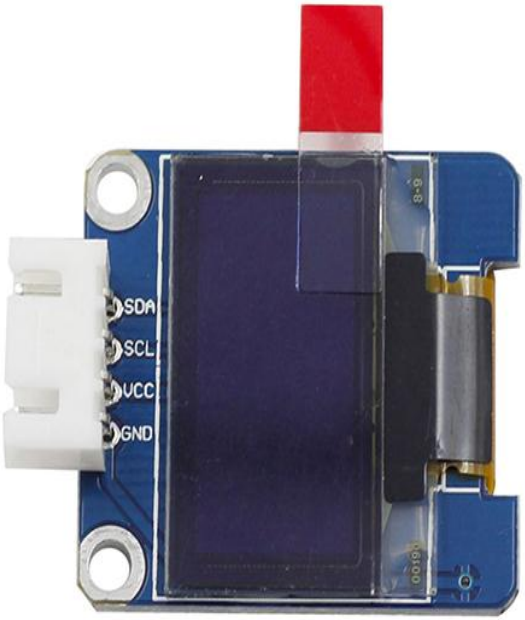

Figure 3.18: OLED

3.3.10 Relay

A relay is an electromagnetic switch. It is activated when a small current of some microampere is applied to it as a single transistor providing high current gain. In this pair the current amplified by the first transistor is further amplified by the next transistor providing high current to the output terminal. The text to speech is done by Google Translate. The speech synthesis is generated from text passed from the functions in .Node. The voice-controlled, multi-functional, smart home automation system (SHAS), can receive voice commands by a specific person and perform corresponding functions improves accessibility to appliances through a natural interface that turns on the light Speech recognition can be used to unlock the door of the house. The light and fans can also be made automated by using motion detectors and temperature sensors. Natural language processing has a lot of useful applications in machine translation.

CHAPTER 4

EXPERIMENTAL SETUP AND PROCEDURE

4.1 SOFTWARE TOOLS

- Arduino IDE
- Android studio

4.1.1 TOOLS DESCRIPTION

Arduino IDE

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

Open Source and Extensible Software

The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, the AVR-C code is also added directly into your Arduino program.

Open Source and Extensible Hardware

The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

4.2 OVERVIEW OF ARDUINO IDE

Multi-platform application

Arduino IDE works on the three most popular operating systems: Windows, Mac OS, and Linux. Aside from that, the application is also accessible from the cloud. These options provide programmers with the choice of creating and saving their sketches on the cloud or building their programs locally and upload it directly to the board.

Board Management

Arduino IDE comes with a board management module, where users can select the board they want to work with at the moment. If they wish to change it, they can do so easily from the dropdown menu. Modifying their selection also automatically updates the PORT info with the data they need in relation to the new board.

Straight forward Sketching

With Arduino IDE, users can create programs called sketches that are built with a text editor. The process is a straightforward one though it has several bells and whistles that make the experience more interactive.

Project Documentation

Arduino IDE offers programmers the option to document their projects. This function allows them to keep track of their advancements and any changes they make every time. Apart from that, documentations allow other people to easily employ the sketches to their own boards.

Simple Sketch Sharing

Aside from saving and archiving sketches and uploading them to the board, Arduino IDE is also capable of sharing sketches (available only on the cloud version).

Each sketch is given its own unique URL that users can share with their colleagues and fellow Arduino hobbyists. The recipient then has access to the code; they can save it in the cloud sketchbook or download it for their own use.

Vast Library

Arduino IDE has more than 700 libraries integrated. These were written and shared by members of the Arduino community that other users can utilize for their own projects without having to install anything. This enables programmers to add a different dimension to their sketches.

Third-Party Hardware Support

While Arduino IDE is designed specifically for Arduino boards, it also supports connections with third-party hardware. This makes the use of the application more extensive rather than limited to proprietary boards.

4.3 INTERFACING ESP BOARDS WITH POPULAR ARDUINO IDE

Firstly, you will need Arduino IDE to be installed in your system. If you don't have it download the latest version from here once IDE is installed open it. Arduino IDE are shown in figure 4.1

1. FILE > PREFERENCES

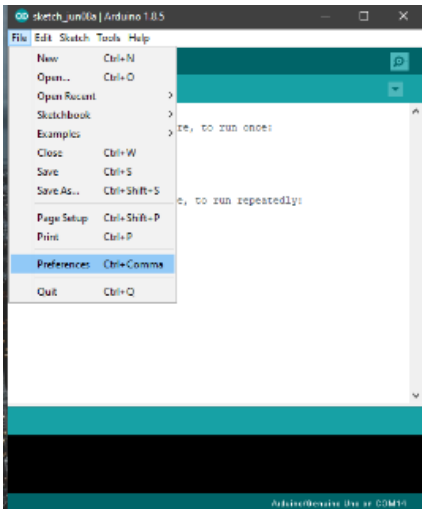

Figure 4.1 File References

2. Enter the below link in the Additional Boards Manager and press OK.

http://arduino.esp8266.com/stable/package_esp8266com_index.json

3. Now open Boards Manager are shown in figure 4.2

Tools > Board > Boards Manager .Search for ESP8266 and install it

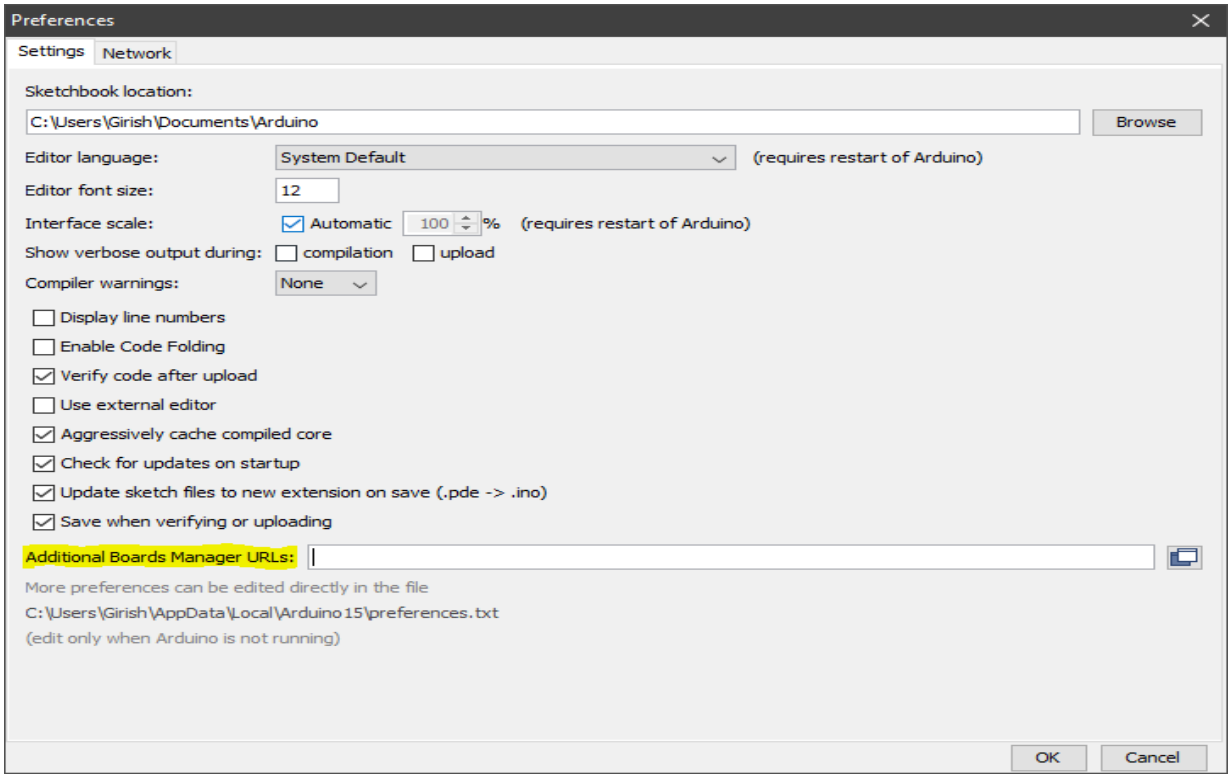

Figure 4.2 : Additional Boards Manager Search for ESP8266 and Install It

Once the installation is completed you are ready to program esp8266 with Arduino IDE. You can see a new list of boards added to boars section of your IDE in figure 4.3.

Go to Example > ESP8266 > Blink. Select the corresponding board from the list

Here I have used ESP8266 NODEMCU and if you are using another boards/ Wi-Fi module. Then select your corresponding boards from the list.

NODEMCU has 2 inbuilt LED on it one connected pin 2 and other to pin 16

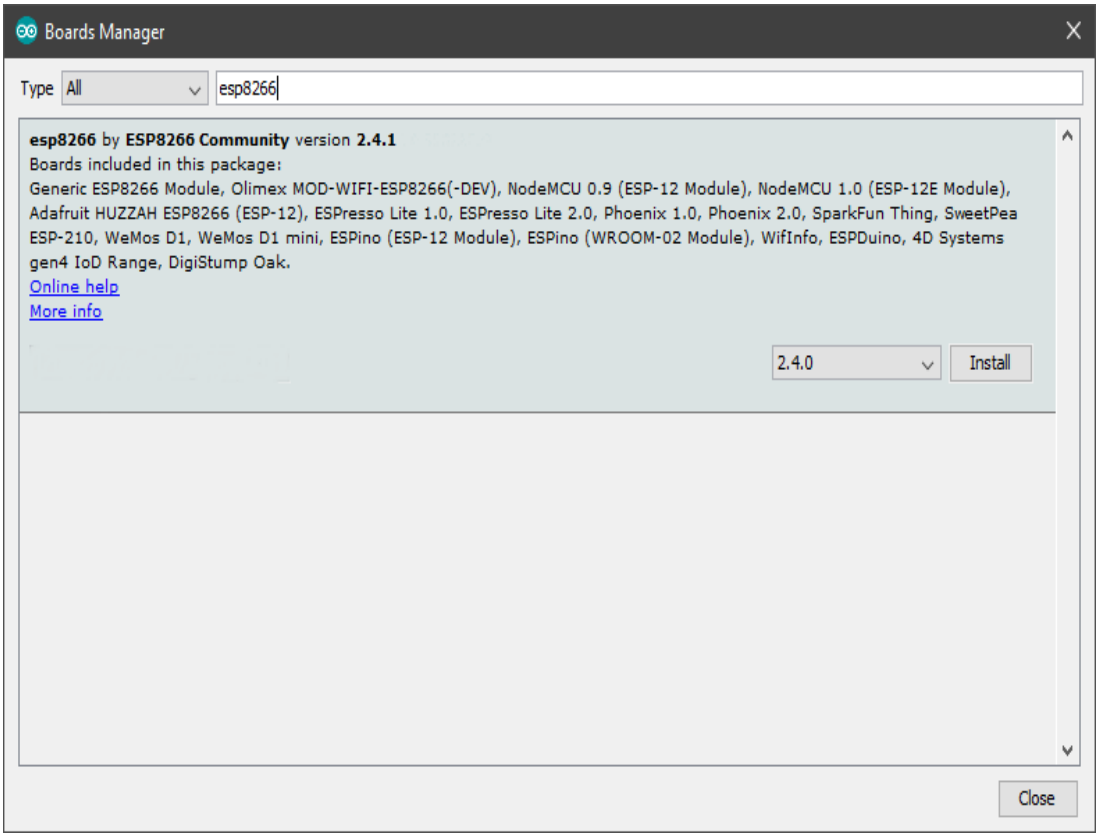

Figure 4.3 : Board Manager Search For Esp8266 And Install

Now test the IDE by uploading an example sketch from examples are shown in figure 4.4

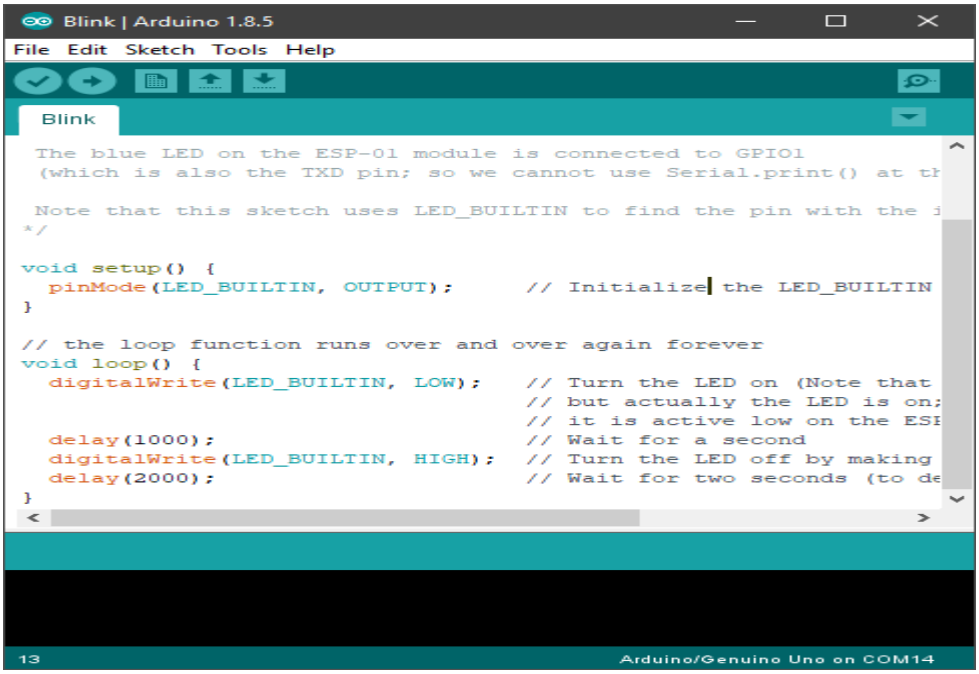

Figure 4.4 : NODEMCU Sketch Examples

4.4 INTERFACING FIREBASE WITH ARDUINO IDE

1. Download and install firebase-Arduino-master library in Arduino IDE.
2. Need Gmail account for create Firebase project.
3. Go to firebase console and create new project
4. Click on Database now you will see the host name
5. Copy that host name and past in Arduino code at appropriate line
6. Go to Setting>Project Setting>SERVICE ACCOUNTS>DATABASE Secretes.
7. Copy "Database Secrets"
8. Copy and paste Database Secrets at the appropriate line in the Arduino code
9. Change your Wi-Fi router name and password in Wi-Fi section
10. After resetting ESP8266check serial terminal whether the ESP is get connected with your router and got IP address.

4.5 ANDROID STUDIO

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development Android studio contains all the Android SDK tools to design, test, debug and profile your app. By looking at the development tools and environment, we can its similar to eclipse with the ADT plug-in but as I have mentioned above its android focused IDE, there are many cool features available in Android Studio which can foster and increase your development productivity.

4.6 INSTALLATION

1. Install JDK 6 or later
2. Set the JAVA_HOME environment variable to the location of your JDK installation.
3. Download and install Android Studio

4.6.1 Connecting android studio with firebase

First make sure you have installed Google Repository version 26 or higher, using the following steps:

1. Click Tools > SDK Manager.

2. Click the SDK Tools tab.
3. Check the Google Repository checkbox, and click OK.
4. Click OK to install.
5. Click Background to complete the installation in the background, or wait for the installation to complete and click Finish.

You can now open and use the Assistant window in Android Studio by following these steps:

1. Click Tools > Firebase to open the Assistant window.
2. Click to expand one of the listed features (for example, Analytics), then click the Get Started tutorial to connect to Firebase and add the necessary code to your app.

4.7 LIST OF HARDWARE COMPONENTS

- NODEMCU (a microcontroller with Wi-Fi module)
- Relay Channels
- Relay Driver

4.7.1 NodeMCU (a microcontroller with Wi-Fi module)

NODEMCU is an open source LUA based firmware developed for ESP8266 Wi-Fi chip. By exploring functionality with ESP8266 chip, NODEMCU firmware comes with ESP8266 Development board/kit i.e. NODEMCU Development board. NODEMCU Device Kit has Arduino like Analog (i.e. A0) and Digital (D0-D8) pins on its board.

It supports serial communication protocols i.e. UART, SPI, I2C etc. Using such serial protocols we can connect it with serial devices like I2C enabled LCD display, Magnetometer HMC5883, MPU-6050 Gyro meter + Accelerometer, RTC chips, GPS modules, touch screen displays, SD cards etc. Since NODEMCU is open source platform, their hardware design is open for edit/modify/build.

NODEMCU Device Kit/board consist of ESP8266 Wi-Fi enabled chip. The ESP8266 is a low-cost Wi-Fi chip developed by Espress if Systems with TCP/IP protocol. For more information about ESP8266, you can refer ESP8266 Wi-Fi Module. There is Version2 (V2) available for NODEMCU Device Kit i.e. NODEMCU Development Board v1.0 (Version2), which usually comes in black coloured PCB.

4.7.2 Relay Channels

The main usage of the Relay is for transmitting and receiving the information, that was called as Morse code where the input signals used to be either 1 or 0, these change in signals were mechanically noted in terms of ON and OFF of a light bulb or a beep sound it means those pulses of 1s and 0s are converted as mechanical ON and OFF using electromagnets. Later this was improvised and used in various applications. Let's see how this electromagnet acts as a switch and why it is named as RELAY.A relay is classified into many types, a standard and generally used relay is made up of electromagnets which in general used as a switch. The same meaning can be applied to this device because the signal received from one side of the device controls the switching operation on the other side. So relay is a switch which controls (open and close) circuits electromechanically. The main operation of this device is to make or break contact with the help of a signal without any human involvement in order to switch it ON or OFF. It is mainly used to control a high powered circuit using a low power signal. Relay are shown in figure 5.1.

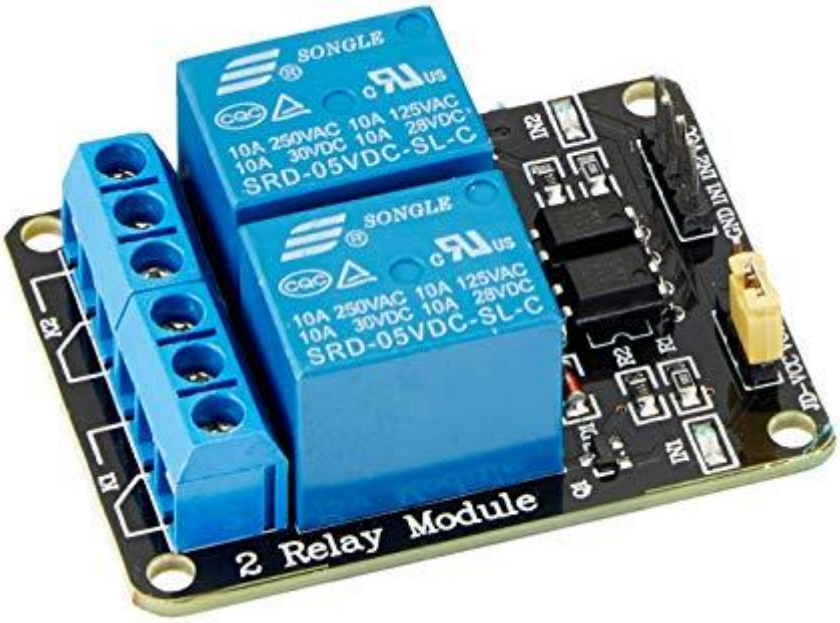

Figure 4.5: Two Channel Relay

4.7.3 Relay Driver

The ULN2003A is an array of seven NPN Darlington transistors capable of 500 mA, 50 V output. It features common-cathode fly back diodes for switching inductive loads. It can come in PDIP, SOIC, SOP or TSSOP packaging. [In the same family are ULN2002A, ULN2004A, as well as ULQ2003A and ULQ2004A, designed for different

logic input levelness. The UNL2003A is also similar to the ULN2001A (4 inputs) and the ULN2801A, ULN2802A, ULN2803A, ULN2804A and ULN2805A, only differing in logic input levels (TTL, CMOS, PMOS) and number of in/outputs (4/7/8)

The ULN2003 is known for its high-current, high-voltage capacity. The drivers can be paralleled for even higher current output. Even further, stacking one chip on top of another, both electrically and physically, has been done. Generally it can also be used for interfacing with a stepper motor, where the motor requires high ratings which cannot be provided by other interfacing devices.

4.7.3.1 Main Specifications

- 500 mA rated collector current (single output)
- 50 V output (there is a version that supports 100 V output)
- Includes output fly back diodes
- Inputs compatible with TTL and 5-V CMOS logic

CHAPTER 5

RESULT ANALYSIS AND DISCUSSION

The performance analysis the user give the voice command input to the Google assistant and listen the voice command of user they provide the speech into text are shown in figure 5.1. After that transformation speech to text format goes to the Dialogflow in the natural language process and they triggering the voice command of the user interface

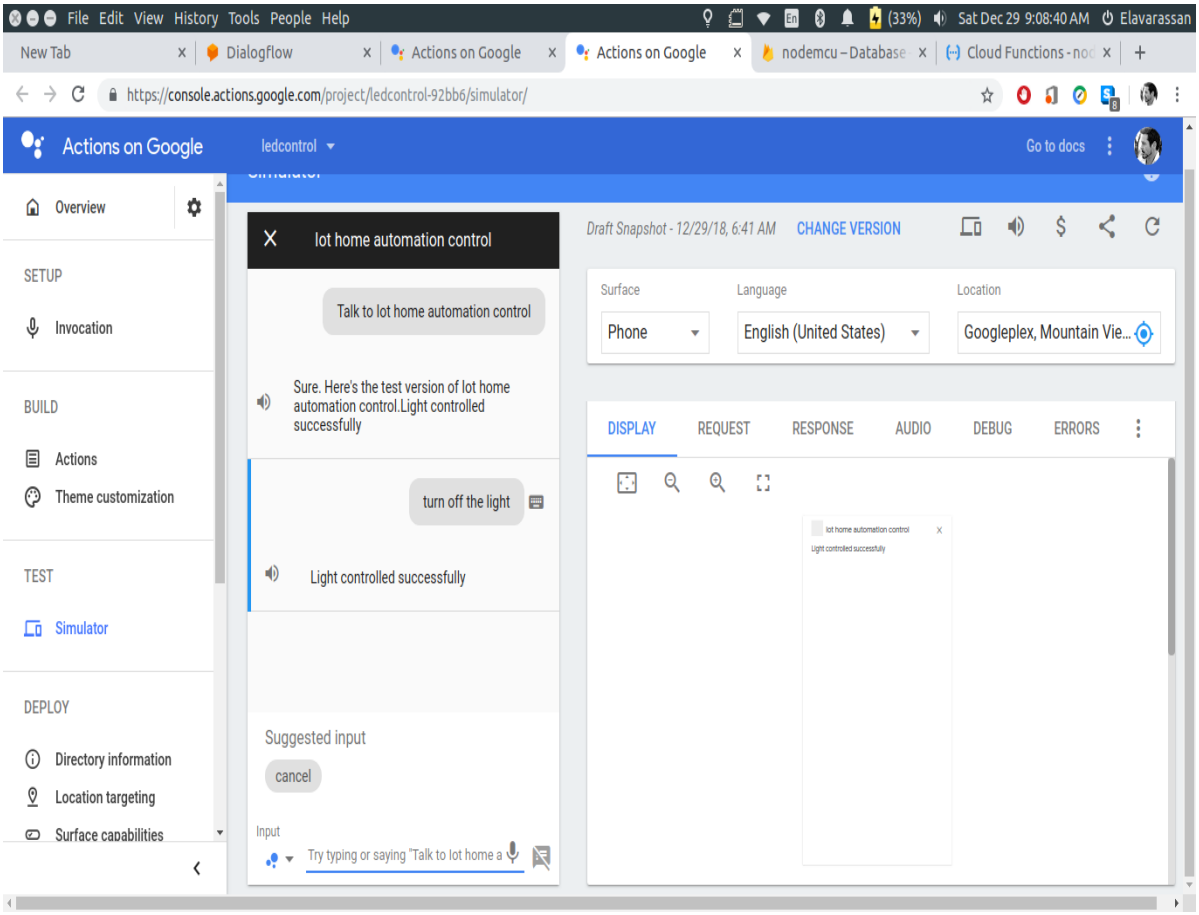

Figure 5.1 : Performance Analysis of Google Action

Analytics for indicating the number of time users using the cloud service which is shown in figure 5.2.

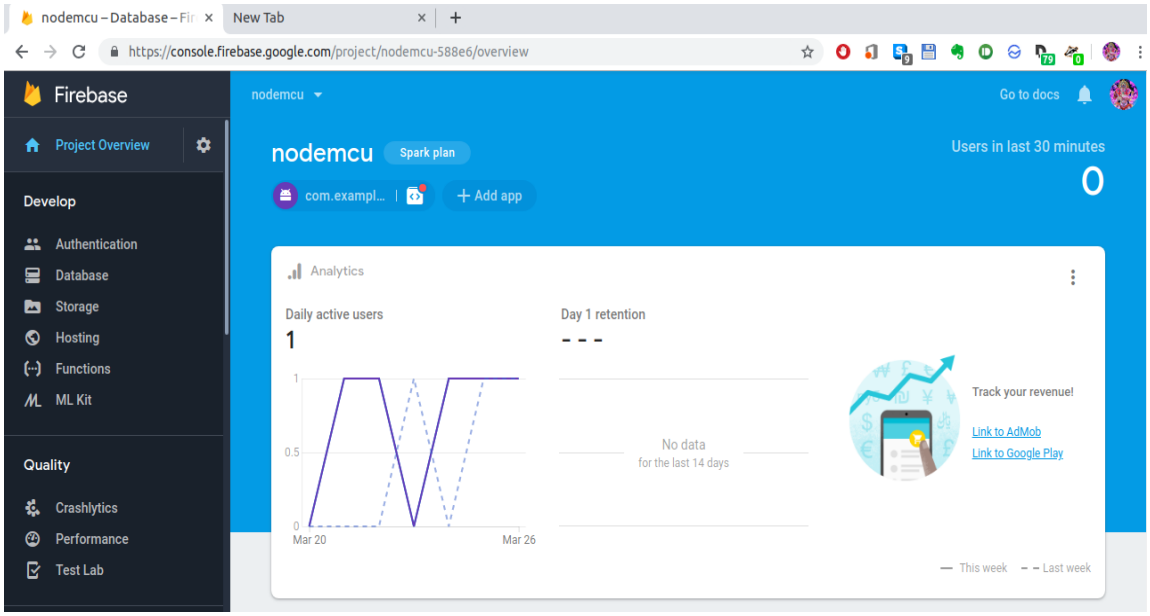

Figure 5.2: Firebase Analytics for Daily Usage Timeline of Users

Analytics for cloud function and real time database used by the user with respect to date is shown in figure 5.3.

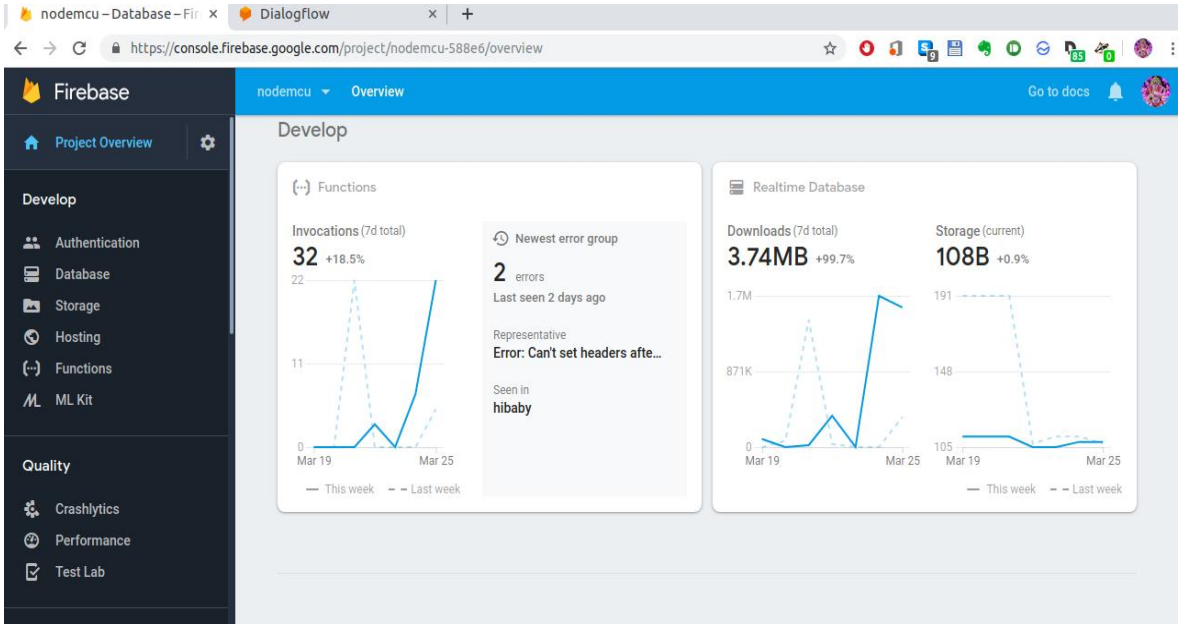

Figure 5.3: Firebase Analytics for Cloud Function and Real Time Database

In that performance don't use third party website, use Google serve because they server very quickly response voice command to activate the home appliances devices they show that many devices are connected in the assistant like light activate or deactivate the device using voice command

Deactivate the devices using our voice command are shown in figure 5.4

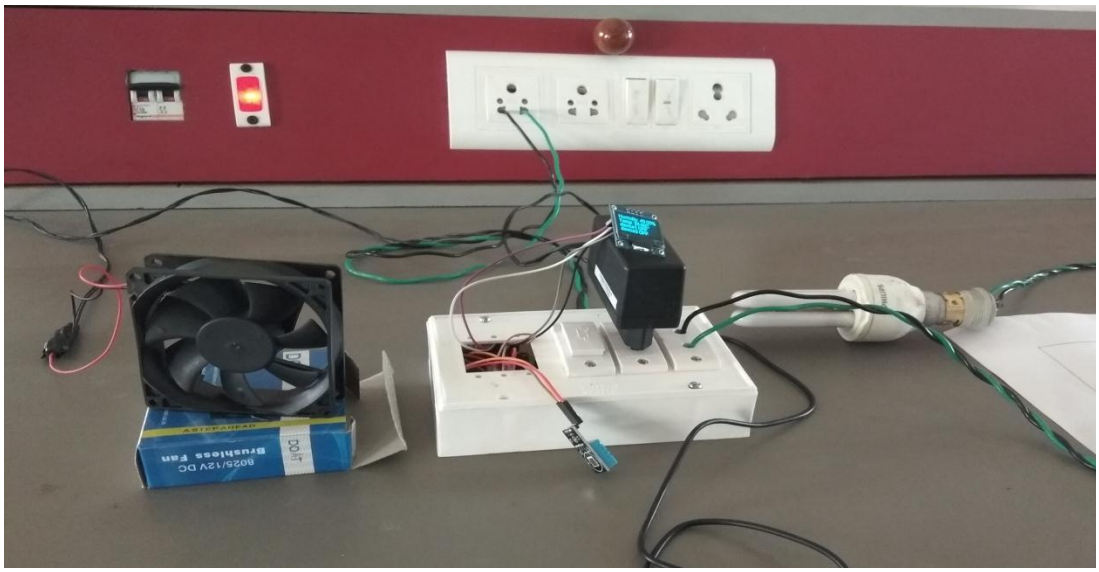

Figure 5.4: Light in off Condition

Activate our device using voice command are shown in figure 5.5

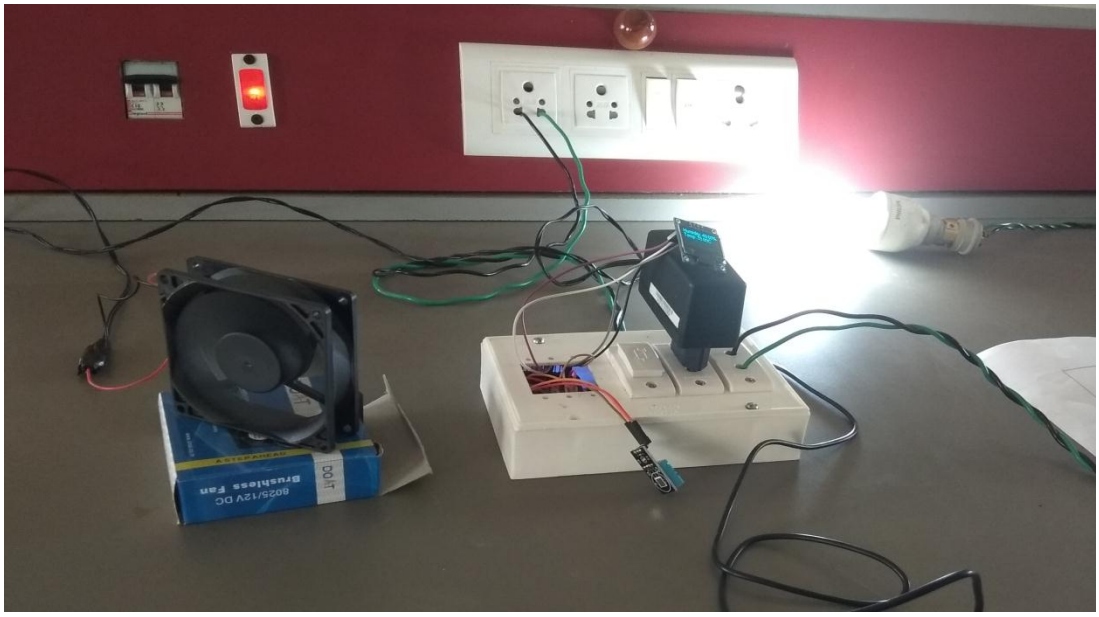

Figure 5.5: Light in on Condition

CHAPTER 6

CONCLUSION

In the multifunctional system that should using by voice input of the user interface and it perform various function of the device is controlled by voice as well as by web access and the whole process can be achieved by triggering the firebase and NodeMCU. In the voice recognition function can be used in the common places or different places in the home whether it inside or outside the home. Initially the voice command is given as input to the Google AI and it converts effectively into text commands, then the command is analyses by the Dialogflow API then it trigger the firebase function. Firebase function changes the value according to the value passes by the Dialogflow API. Now this change in the real-time database triggers the NodeMCU which controls the home appliances and other devices connected to it through relay. Also it is observed that the accessing is fast when the router or Wi-Fi signal is strong.

This whole process can be accessed from anywhere but firebase is secured with a mail. The relay controlled devices are tested using Google AI through Dialogflow API, http triggering and firebase. These settings are created under Dialogflow API login mail used for firebase. The device is turn ON when the voice command is given through Google Assistance as “Turn ON the device” and the device is turned OFF when the voice command is given as “Turn OFF the Device”. These commands are either given as voice input to trigger the NodeMCU ON or by clicking the ON /OFF button on the Android app will trigger and drive the NodeMCU.

.

LIST OF PUBLICATION

1.Dhivya S, Dhivyabharathi N, Elavarassan K, Guhan M and Mugilan D, 2019 ”ARTIFICIAL INTELLIGENCE BASED RELAY CONTROLLER” 2nd International Conference on Recent Advances in Engineering and Technology.

APPENDIX 1

ARDUINO SOURCE CODE:

```
#include "DHT.h"
#define DHTPIN 2
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>
#define DHTTYPE DHT11
// Set these to run example.
//DHT dht;
#define FIREBASE_HOST "nodemcu-588e6.firebaseio.com"
#define FIREBASE_AUTH "3lBvXOv5aFaU9Xiz1C8mCr8TKOystsWQx08uxcMO"
#define WIFI_SSID "My"
#define WIFI_PASSWORD "Prince1707"
const int dev4=4;
const int dev3=16;
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  pinMode(dev4,OUTPUT);
  pinMode(dev3,OUTPUT);
  dht.begin();
  digitalWrite(dev4,1);
  digitalWrite(dev3,1);
  Serial.begin(9600);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");
  while (WiFi.status() != WL_CONNECTED) {
  Serial.print(".");
  delay(500);

    }
  Serial.println();
```

```
erial.print("connected: ");
Serial.println(WiFi.localIP());
 Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
}
String n="";
void loop() {
   // Wait a few seconds between measurements.
  delay(2000);
  // Reading temperature or humidity takes about 250 milliseconds!
  // Sensor readings may also be up to 2 seconds 'old' (its a very slow sensor)
  float h = dht.readHumidity();
  String humidity=String(h,0);
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();
  String temperature=String(t,0);
  // Read temperature as Fahrenheit (isFahrenheit = true)
  float f = dht.readTemperature(true);
   String temperature_f=String(f,0);
  //* Get temperature value */
     if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
 Firebase.setString("temperature",temperature+" "+temperature_f);
  Firebase.setString("humidity",humidity);
  n=Firebase.getString("Device-4_ON");
   Serial.println(n);
   if(Firebase.getString("Device-4").equals("1"))
    {
      Serial.println(" sucsessfully device 4 on");
      digitalWrite(dev4,LOW);
    }
```

```
 if(Firebase.getString("Device-4").equals("0"))
   {
      Serial.println(" sucsessfully device 4 off");
     digitalWrite(dev4,HIGH);
   }
   if(Firebase.getString("Device-3").equals("1"))
   {
     Serial.println(" sucsessfully device 3 on");
     digitalWrite(dev3,LOW);
   }

   if(Firebase.getString("Device-3").equals("0"))
   {
      Serial.println(" sucsessfully device 3 off");
      digitalWrite(dev3,HIGH);
    }
  if (Firebase.failed()) // Check for errors
   {
   Serial.print("setting /number failed:");
   Serial.println(Firebase.error());
   return; }
  delay(3000);
}
```

NODEJS SOURCE CODE:

```
const functions = require('firebase-functions');
var admin=require('firebase-admin');
admin.initializeApp(functions.config().firebase);
var database=admin.database();
// // Create and Deploy Your First Cloud Functions
// // https://firebase.google.com/docs/functions/write-firebase-functions
//
exports.hibaby = functions.https.onRequest((request, response) => {
```

```
        let params=request.body.result.parameters;
        console.log(params);
    database.ref().update(params);
    global.state="";
    var ref_state = database.ref("state");
                ref_state.on("value", (data) => {
      state=data.val().toString();
      console.log(state);
    });
    console.log(state);
    if(state==="1")
    {
            var ref_temperature = database.ref("temperature");
                ref_temperature.on("value", (data) => {
            var temperature=data.val().toString().split(" ");
            response.send({
            speech:"your room temperature is "+temperature[0]+" degree
celcius and "+temperature[1]+" degree Farenhit"
      })


    });
    }
    else if(state==="0")
    {
    var ref_humidity= database.ref("humidity");
                        ref_humidity.on("value", (data) => {
                var humidity=data.val().toString();
                response.send({
                        speech:"your room humidity is "+humidity
                })
```

```
                });
        }
        else
        {
                response.send({
                        speech:"Ok device is controlled sucessfully"
                })
        }
});
```

SOURCE CODE ANDROID BACKEND:

```
package com.example.elavarassan.sweethome;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
public class MainActivity extends AppCompatActivity {
   Button on1;
   Button off1;
   Button on2;
   Button off2;
   Button on3;
   Button off3;
   Button on4;
   Button off4;
   TextView status1;
```

```
TextView status2;
TextView status3;
TextView status4;
TextView status5;
TextView  status6;
DatabaseReference dref;
String status;
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    on1 = (Button) findViewById(R.id.on_1);
    off1 =(Button) findViewById(R.id.off_1);
    on2 = (Button) findViewById(R.id.on_2);
    off2 = (Button) findViewById(R.id.off_2);
    on3 = (Button) findViewById(R.id.on_3);
    off3 = (Button) findViewById(R.id.off_3);
    on4 = (Button) findViewById(R.id.on_4);
    off4 = (Button) findViewById(R.id.off_4);
    status1 = (TextView) findViewById(R.id.textView1);
    status2 = (TextView) findViewById(R.id.textView2);
    status3 = (TextView) findViewById(R.id.textView3);
    status4 = (TextView) findViewById(R.id.textView4);
    status5 = (TextView) findViewById(R.id.textView5);
    status6 = (TextView) findViewById(R.id.textView6);


    dref= FirebaseDatabase.getInstance().getReference();
    dref.addValueEventListener(new ValueEventListener() {
        public void onDataChange(DataSnapshot dataSnapshot) {
            status=dataSnapshot.child("Device-1").getValue().toString();
            if(status.equals("1"))
                status = "ON";
```

```
    else
        status="OFF";
    status1.setText(status);

    status=dataSnapshot.child("Device-2").getValue().toString();
    if(status.equals("1"))
        status = "ON";
    else
        status="OFF";
    status2.setText(status);

    status=dataSnapshot.child("Device-3").getValue().toString();
    if(status.equals("1"))
        status = "ON";
    else
        status="OFF";
    status3.setText(status);

    status=dataSnapshot.child("Device-4").getValue().toString();
    if(status.equals("1"))
        status = "ON";
    else
        status="OFF";
    status4.setText(status);

    status=dataSnapshot.child("temperature").getValue().toString();
    status5.setText(status);

    status=dataSnapshot.child("humidity").getValue().toString();
    status6.setText(status);

}
```

```
    public void onCancelled(DatabaseError databaseError) {
    }
});


on1.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view) {
        FirebaseDatabase database = FirebaseDatabase.getInstance();
        DatabaseReference myRef = database.getReference("Device-1");
        myRef.setValue("1");

    }
});
off1.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view) {
        FirebaseDatabase database = FirebaseDatabase.getInstance();
        DatabaseReference myRef = database.getReference("Device-1");
        myRef.setValue("0");

    }
});
on2.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view) {
        FirebaseDatabase database = FirebaseDatabase.getInstance();
        DatabaseReference myRef = database.getReference("Device-2");
        myRef.setValue("1");

    }
```

```
});
off2.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view) {
        FirebaseDatabase database = FirebaseDatabase.getInstance();
        DatabaseReference myRef = database.getReference("Device-2");
        myRef.setValue("0");

    }
});
on3.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view) {
        FirebaseDatabase database = FirebaseDatabase.getInstance();
        DatabaseReference myRef = database.getReference("Device-3");
        myRef.setValue("1");

    }
});
off3.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view) {
        FirebaseDatabase database = FirebaseDatabase.getInstance();
        DatabaseReference myRef = database.getReference("Device-3");
        myRef.setValue("0");

    }
});
on4.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view) {
        FirebaseDatabase database = FirebaseDatabase.getInstance();
```

```
            DatabaseReference myRef = database.getReference("Device-4");
            myRef.setValue("1");

        }
    });
    off4.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View view) {
            FirebaseDatabase database = FirebaseDatabase.getInstance();
            DatabaseReference myRef = database.getReference("Device-4");
            myRef.setValue("0");

        }
    });

  }
}
```

SOURCE CODE ANDROID FRONT END:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:layout_height="match_parent"
  android:visibility="visible"
  tools:context=".MainActivity">
  <Button
    android:id="@+id/on_4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
```

```
    android:layout_alignParentTop="true"
    android:layout_marginStart="52dp"
    android:layout_marginLeft="52dp"
    android:layout_marginTop="389dp"
    android:background="@color/colorPrimary"
    android:text="ON"
    android:textAppearance="@style/TextAppearance.AppCompat.Button" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginStart="142dp"
    android:layout_marginLeft="142dp"
    android:layout_marginTop="338dp"
    android:text="Devices-4"
    android:textAppearance="@style/TextAppearance.AppCompat.Display1"
    android:textColor="#0040ff"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/off_4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
```

```
    android:layout_marginTop="391dp"
    android:layout_marginEnd="43dp"
    android:layout_marginRight="43dp"
    android:background="?attr/colorControlActivated"
    android:text="OFF" />

<Button
    android:id="@+id/off_3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
    android:layout_marginTop="287dp"
    android:layout_marginEnd="45dp"
    android:layout_marginRight="45dp"
    android:background="?attr/colorControlActivated"
    android:text="OFF" />

<Button
    android:id="@+id/on_2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginStart="51dp"
    android:layout_marginLeft="51dp"
    android:layout_marginTop="177dp"
    android:background="@color/colorPrimary"
    android:text="ON"
    android:textAppearance="@style/TextAppearance.AppCompat.Button" />
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginStart="135dp"
    android:layout_marginLeft="135dp"
    android:layout_marginTop="120dp"
    android:text="Devices-2"
    android:textAppearance="@style/TextAppearance.AppCompat.Display1"
    android:textColor="#0040ff"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginStart="139dp"
    android:layout_marginLeft="139dp"
    android:layout_marginTop="232dp"
    android:text="Devices-3"
    android:textAppearance="@style/TextAppearance.AppCompat.Display1"
    android:textColor="#0040ff"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
```

```
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/on_1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginStart="48dp"
    android:layout_marginLeft="48dp"
    android:layout_marginTop="68dp"
    android:background="@color/colorPrimary"
    android:text="ON"
    android:textAppearance="@style/TextAppearance.AppCompat.Button" />

<Button
    android:id="@+id/off_1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
    android:layout_marginTop="65dp"
    android:layout_marginEnd="50dp"
    android:layout_marginRight="50dp"
    android:background="?attr/colorControlActivated"
    android:text="OFF" />

<Button
    android:id="@+id/off_2"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
    android:layout_marginTop="175dp"
    android:layout_marginEnd="51dp"
    android:layout_marginRight="51dp"
    android:background="?attr/colorControlActivated"
    android:text="OFF" />

<Button
    android:id="@+id/on_3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginStart="52dp"
    android:layout_marginLeft="52dp"
    android:layout_marginTop="283dp"
    android:background="@color/colorPrimary"
    android:text="ON"
    android:textAppearance="@style/TextAppearance.AppCompat.Button" />

<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
```

```
    android:layout_marginTop="78dp"
    android:layout_marginEnd="173dp"
    android:layout_marginRight="173dp"
    android:text="Loading.."
    android:textColor="#e62e00" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
    android:layout_marginTop="194dp"
    android:layout_marginEnd="172dp"
    android:layout_marginRight="172dp"
    android:text="Loading..."
    android:textColor="#e62e00" />

<TextView
    android:id="@+id/textView4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
    android:layout_marginTop="402dp"
    android:layout_marginEnd="167dp"
    android:layout_marginRight="167dp"
    android:text="Loading.."
    android:textColor="#e62e00" />
```

```
<TextView
    android:id="@+id/textView3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
    android:layout_marginTop="298dp"
    android:layout_marginEnd="168dp"
    android:layout_marginRight="168dp"
    android:text="Loading.."
    android:textColor="#e62e00" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="99dp"
    android:layout_marginLeft="99dp"
    android:layout_marginBottom="74dp"
    android:text="Temperature:"
    android:textAppearance="@style/TextAppearance.AppCompat"
    android:textColor="#0040ff"
    android:textSize="18sp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
```

```
    android:layout_alignParentBottom="true"
    android:layout_marginStart="70dp"
    android:layout_marginLeft="70dp"
    android:layout_marginBottom="87dp" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="115dp"
    android:layout_marginLeft="115dp"
    android:layout_marginBottom="34dp"
    android:text="Humidity:"
    android:textAppearance="@style/TextAppearance.AppCompat"
    android:textColor="#0040ff"
    android:textSize="18sp" />

<TextView
    android:id="@+id/textView5"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="94dp"
    android:layout_marginRight="94dp"
    android:layout_marginBottom="75dp"
    android:text="loading..."
    android:textColor="#e62e00"
    android:textSize="18sp" />
```

```
<TextView
    android:id="@+id/textView6"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="94dp"
    android:layout_marginRight="94dp"
    android:layout_marginBottom="32dp"
    android:text="loading..."
    android:textAppearance="@style/TextAppearance.AppCompat.Display1"
    android:textColor="#e62e00"
    android:textSize="18sp" />
   <ImageView
    android:id="@+id/imageView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentStart="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginStart="76dp"
    android:layout_marginLeft="76dp"
    android:layout_marginTop="121dp"
    android:visibility="gone"
    tools:srcCompat="@tools:sample/backgrounds/scenic[13]" />

<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
android:layout_alignParentStart="true"
android:layout_alignParentLeft="true"
android:layout_alignParentTop="true"
android:layout_marginStart="141dp"
android:layout_marginLeft="141dp"
android:layout_marginTop="2dp"
android:text="Devices-1"
android:textAppearance="@style/TextAppearance.AppCompat.Display1"
android:textColor="#0040ff" />
</RelativeLayout>
```

REFERENCES

[1] **Yash Mittal**, 'A Voice Controlled Multi-Functional Smart Home Automation', International Journals of Engineering and Sciences, vol. 13, no. 3, 2015.

[2] **Cyril Joe Baby**, 'Home Automation Using Web Application and Speech Recognition', International Journal of Modern Engineering Research (IJMER), Vol, vol. 2, pp. 3406–3414, 2017.

[3]**Arriany A**, 'Applying Voice Recognition Technology for Smart Home Networks', IEEE Transaction. Consumer Electron, vol. 60, no. 3, pp. 534-539, 2016.

[4] **Ying-Tsung Lee**, 'An Integrated Cloud Based Smart Home Management System with Community Hierarchy ', Consumer Electron, vol. 59, no. 3, pp. 492-498, 2016.

[5] **Murad khan**, 'IOT Based Energy Aware Smart Home Control System', IEEE Access, vol. 3, pp. 2176-2190, 2017.

[6] **Chaypathy V**, 'IOT Based Home Automation By Using Personal Assistant', vol. 3, Issue 2, 213, 2017.

[7] **Pasd putthapipat**, 'Speech recognition gateway for home automation', Vol. 2 Issue 4, 2017.

[8]**Norhafizah**, 'Voice Control of Home Appliances Using Android', in 2016 International Conference on Electronics, Information, and Communications (ICEIC), pp. 1–4, 2014.

[9] **M Khan**, 'A Zigbee Based Tangible Smart Home Interface', Vol. 2, Issue 4, 2018.

[10] **M.Tharaniya**, 'Intelligent Interface Based Speech Recognition for Home Automation using Android Application', The International Journal of Engineering and Science (IJES), Volume 2, Issue 01, Pages 149-153, 2013.