Build Your Own Sci-fi Ai Personal Assistant.
- A module used to automate and communicate with Humanoid Robots.

[Just a Rather Very Intelligent System.]

This is a piece of Ai software that understands verbal or written commands and completes task assigned by the client. It is an example of weak AI that is it can only execute and perform quest designed by the user. With the python programming language, a script most commonly used by the developers can be used to build your personal AI assistant to perform task designed by the users.

Skills: **The implemented voice assistant can perform the following task it can open YouTube, Gmail, Google chrome and stack overflow. Predict current time, take a photo, search Wikipedia to abstract required data, predict weather in different cities, get top headline news from Times of India and can answer computational and geographical questions too.**

Packages required:

To build a personal voice assistant it's necessary to install the following packages in your system using the pip command.

1) **Speech recognition** — Speech recognition is an important feature used in house automation and in artificial intelligence devices. The main function of this library is it tries to understand whatever the humans speak and converts the speech to text.

```
conda install -c conda-forge speechrecognition
```

2) pyttsx3 — pyttxs3 is a text to speech conversion library in python. This package supports text to speech engines on Mac os x, Windows and on Linux.

```
pip install pyttsx3
```

3) wikipedia — Wikipedia is a multilingual online encyclopedia used by many people from academic community ranging from freshmen to students to professors who wants to gain information over a particular topic. This package in python extracts data's required from Wikipedia.

```
conda install -c conda-forge wikipedia
```

4**) datetime —** This is an inbuilt module in python and it works on date and time

5**) os —** This module is a standard library in python and it provides the function to interact with operating system

6**) time —** The time module helps us to display time

7**) Web browser —** This is an in-built package in python. It extracts data from the web

8**) Subprocess —** This is a standard library use to process various system commands like to log off or to restart your PC.

9**) Json-** The json module is used for storing and exchanging data.

10**) request -** The request module is used to send all types of HTTP request. Its accepts URL as parameters and gives access to the given URL'S.

11) wolfram alpha — Wolfram Alpha is an API which can compute expert-level answers using Wolfram's algorithms, knowledge base and AI technology. It is made possible by the Wolfram Language.

```
Pip install wolframalpha
```

12) instaloader - This Api deals with loading instagram data , downloading and uploading featured images and files to it.

```
pip install instaloader
conda install -c anaconda comtypes

conda install -c anaconda pyaudio


API KEYS :

Weather : 8ef61edcf1c576d65d836254e11ea420
https://api.openweathermap.org/data/2.5/weather?”



wolframalpha :

app_id=“R2K75H-7ELALHR35X"
```

Sample commands to deal with JARVIS :

1. Hey JARVIS , Who are you
2. Who created you ?
3. JARVIS can you show up some news
4. Whats the time now ?
5. What’s the weather now ?
6. Who is Elon Musk according to wikipedia
7. Who founded neural link according to wikipedia

8. Can I ask you something ? - what's tan 45 digrees , what's the distance between earth & moon
9. Do you have feelings for humans ?
10. Can you download instagram dp ?
11. Much more can be added …….

Following content to be used to prepare your college project reports -

Abstract— The main aim of the work is to develop an economically effective and performance wise efficient virtual assistant using Raspberry Pi for home automation based on the concepts of Internet of Things, Speech Recognition, Natural Language Processing and Artificial Intelligence. People who are using it can give voice inputs and the device itself responds through voice commands by itself. It can fetch the date, time, weather, play your favourite music and fetch search results from the internet along with controlling the home appliances. NodeMCU chips are used to control the appliances which receives the command from the Raspberry Pi. The Raspberry Pi processes the speech inputs online given by the user through the mic and converts it into text and executes the command. The whole project is put in action through a python script which includes online Speech to Text conversion and Text to Speech conversion codes written. The NodeMCU is coded separately using the Arduino IDE to make it control the appliances and allow it to be accessed through its IP address. The device will respond to the user in a casual manner so that the user has a friendly experience with the device and feels it like his or her own assistant. This device makes the day by day processes easier.

Index Terms—Artificial Intelligence, Home Automation System,

Natural Language Processing, NodeMCU, Raspberry Pi, Speech Recognition, Speech to Text, Text to Speech.

INTRODUCTION

HIS work is constructed based on the basic concepts on Internet of things (IoT) and Natural Language Processing (NLP). We have put forth a method to integrate all the home appliances through a central command center. This will eventually reduce the human interaction with the appliances and make the work easier. This system is designed to provide a user friendly experience as well as an easier interface so that anyone can use this effortlessly. This project has been implemented with the help of Raspberry Pi 3 Model B and NodeMCU which is an open source platform in Internet of Things that runs on the ESP8266Wi-Fi SoC and the ESP-12 module based hardware.

Raspberry Pi is known as the credit-card sized computer which was initially designed for education, inspired by the 1981 BBC Micro. It is a low-cost device through which various applications of IoT can be developed. Knowledge of Linux Operating System will be required to work with a Raspberry Pi board. It runs on a Quad core ARM Cortex A-53 at a clock speed of 1.2GHz. It has a 1 Gigabyte DDR2 RAM with a clock speed of 900MHz. It has inbuilt Wi-Fi and Bluetooth connectivity.

Fig.1 - Raspberry Pi 3 Model B

NodeMCU is an Arduino-like hardware IO which comes along with an inbuilt ESP8266Wi-Fi module. It is a development kit which helps us to create prototypes of our IoT product using Lua Script. We can also use the Arduino IDE to code the NodeMCU. NodeMCU has to be included in the IDE which will help us use the kit with it.

Fig.2 - NodeMCU

The Internet of Things is a major breakthrough in technology. It is nothing but communication of things (devices) with each other using the internet as a medium. The major protocol used in IoT is the MQTT protocol. The MQTT protocol works based on the TCP/IP protocol. It is where the concept of message brokers come into play. The brokers act as a vector which directs the information that is needed to be communicated throughout the devices. Knowing this MQTT protocol will help in better understanding of this system. Also basic knowledge of TCP, UDP will also be needed for better interpretation. In this project the devices communicate with each other with the help of MQTT protocol.

Natural Language Processing is simply a bridge-way that reduces the distance between human communication and machine communication. The main objective of NLP is to make the machines understand the natural human language so that the usage becomes very much comfortable. In technical terms, NLP is the algorithm which analyzes and synthesizes human speech. This algorithm is based on artificial intelligence and computational linguistics. [1]

II. RELATED WORKS IN THE FIELD

There are a lot of commercial companies which are into Home Automation like Schneider, Home Automation Inc., Logitech, Honeywell, etc., They have made successful attempts in integrating the home appliances to a central server and controlling them using the same. Certain companies include Security also.

Natural Language Processing is a real difficult part in this domain. Many legends including Google, Amazon, Apple, Microsoft have NLP in action in the form of Google Now, Alexa, Siri and Cortana respectively. The NLP is based on machine learning which breaks down the words to the smallest unit called phonemes and compare it with the pre-recorded speech which had been trained previously. Speech to Text also comes under the NLP. The primary aim of NLP is to make the human to computer interaction easier by making it more like human to human interaction. By the term human to human interaction, we mean the communication through native languages like English, French, Spanish, etc., The computers usually communicate through artificial languages like C, C++, Java, Python, etc., Our project aims to combine all of this and create a Virtual Assistant which would be handy in Home Automation also.[9]

Devices like Amazon Alexa, Google Home are existing systems which are dedicated personal assistants which comes with a lot of features. But they are equally expensive which cannot be affordable by everyone. This system is developed to be less expensive than the existing system providing features that they provide along with add-ons.

Facebook CEO Mark Zuckerberg tried developing an Artificially Intelligent personal assistant named *'Jarvis'* inspired from the movie *'Iron Man'*. He quite succeeded in the project. It took so much effort and so much coding to make it possible.

I. EXISTING SYSTEM

There are Home Automation Systems which are voice controlled. The primary objective of these systems is switching i.e., Switching ON and OFF the appliances and nothing more than that. The Existing system restricts the possibility of technological advancements which can be put forth as outdated. [5]

There are some systems which use mobile phones to process the Natural Languages and put them into work. But that remains a void of a dedicated device which can be relied upon. So the existing system is not very much reliable. There are certain virtual assistants available as dedicated devices which are very much costlier to afford. So it becomes economically ineffective.

Though devices like Amazon Echo, Google Home are available, additional configurations and setup maybe required to make it viable for automation of the appliances which requires technical assistance. Moreover, there has to be some physical interaction with the devices to wake it up in order to perform the tasks in the existing system.[10]

The Open Source platforms like *Jasper* which can be used to develop voice controlled applications can be used. But it requires a lot of practical knowledge in Raspberry Pi and Linux Operating Systems. It takes too much time in set up as it works default in the CMUSphinx Speech engines which are basically offline.[2][4]

As to overcome all the issues of the existing system, our project works on a dedicated single python script which requires the most minimum configuration and setup and makes it easy for the installation.

I. PROPOSED SYSTEM

This work focuses on providing with the easiest and the most effective method to communicate with the system by giving voice commands through natural languages. This project eliminates the hectic process of tiring configurations and setups and overheating of the system which ultimately affects the performance. The conversation by natural languages to the system and its response makes the user feel like he is talking to another human and makes him ignore the fact that a system is performing all the tasks.

A. Architecture

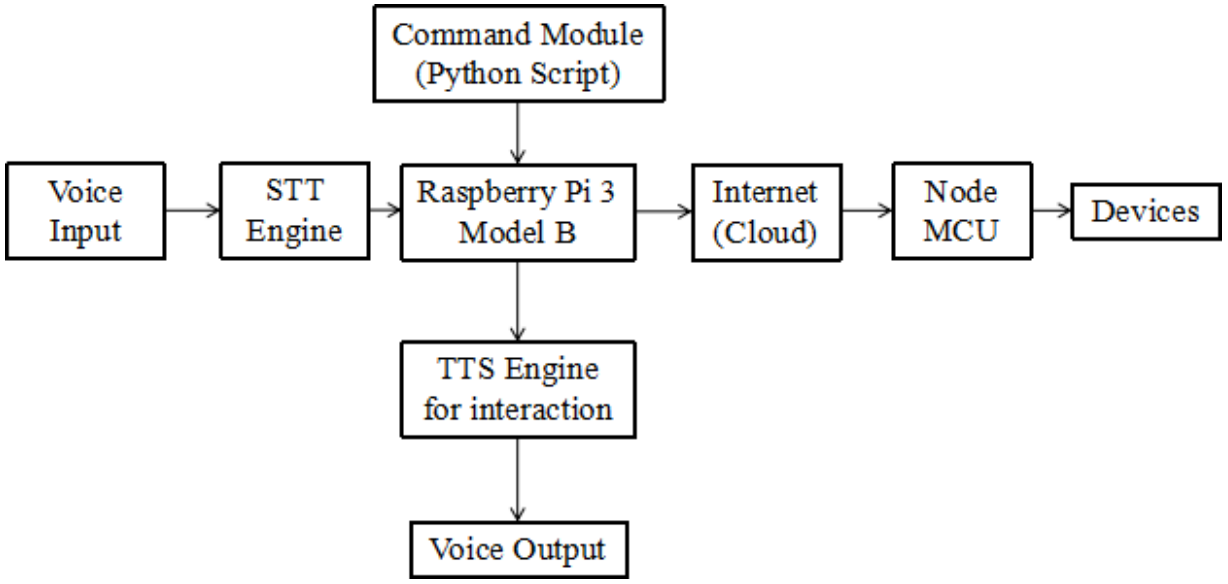


Fig.3 - Architecture

Fig.3 shows the architecture of the proposed system. The work initially seemed difficult as a whole. So to make it simpler it was divided into modules to make the work easier and also for debugging.

The three modules are as follows:

- Speech to Text Module
- Command Module
- Text to Speech Module

B. Speech to Text Module

The first module is the Speech to Text Module whose main function is to convert the speech signals into text so that it can be easier to put it in the form of a program. Initially the Raspberry Pi is set up to obtain the audio through a USB microphone in the default mode. Also ALSA configuration is done for better audio experience. After the basic setup is done, an STT engine must be installed in the Raspberry Pi.

There are various STT engines namely Pocketsphinx, Wit.ai STT, Google STT, Julius, AT&T STT. Pocketsphinx engine is known for its offline capabilities and works well without internet. It is specially designed to work well with systems like Raspberry Pi. The disadvantages are that it takes so much time to configure it. It also requires a lot of dependencies. In that case, Wit.ai or Google STT could be a better choice.[4]

Wit.ai is designed to work based on the wit.ai cloud services. It uses an unique technique called the crowdsourcing to train the voice recognition algorithms. A fast reliable internet connection is required to use this engine.

Google STT is directly from the Google. It requires an active internet connection to work. It is the one that is used in android phones for speech inputs. So it is reliable as in recognition.

In this project, We have used Google STT which is open- source and easily accessible and known to provide promising results. A speech recognition library is installed which will be imported into the python script. The whole script will be written based on the library.

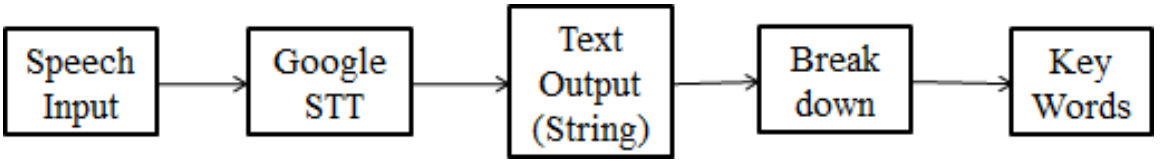


Fig.4 - STT Module

Initially, the microphone is assigned as source and then the input is acquired from the user. Before that, it is made to adjust itself according to the ambient noise or else the microphone would mistake noise as inputs. Usually the system takes about a second to adapt itself to the surrounding noise and get used to it. Once it gets adapted, the system will filter out the noise and take in the speech audio separately and sends it to the Google cloud.

The Google cloud converts the speech into text and we can fetch the data and store it in a string. This string will consist of unnecessary phrases which may not be required in the programming. So the string is broken into words and only the required words are looked for and corresponding actions are assigned to them under separate functions.[3][5]

A. Command Module

This is the module which consists most of the programming including the code of NodeMCU. First we will look at the part of NodeMCU setup. The IO of NodeMCU is similar to that of Arduino. So people with prior knowledge to Arduino will find it easier to code.

Initially the NodeMCU is considered the server in which the SSID and the password of the router will be provided before-handed in the code. The hardware connections are very minimal. The board is

connected to a 5V relay circuit and the appliances are connected through the relay. The relay acts as the switching. The power supply to the relay will be given from the NodeMCU.

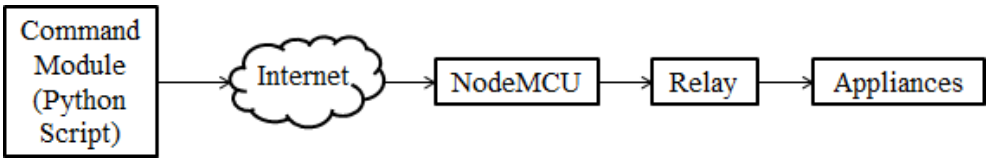


Fig.5 - Command Module

Motor drivers must be used for interfacing High Voltage motors and also other suitable drivers must be used for Air Conditioners (HVAC) and water heaters which uses up high voltage for their functioning.[5]

The software part includes assignment of a static IP to the NodeMCU. Once the IP is assigned to the board, a code is written to create a webpage which consists of ON and OFF buttons respective to the number of devices to be controlled. The buttons can be accessed through the IP which can be included in the python script of the Raspberry Pi.

Separate functions are created to fetch the date, time, weather (using any weather API), Google Search and to access the NodeMCU through the browser. These functions are programmed to be executed based on the keywords found by using ‘if’ statements.

It should be noted that all the required libraries should be properly installed in the corresponding directories to avoid unnecessary bugs in the program.

This project was initially started with the aim of connecting it to Jasper. The configuration of Jasper took a lot of time and also the setup did not complete successfully which caused many errors which eventually crashed the system repeatedly. So after deep learning about Jasper, it was found to be nothing but a combination of many python scripts. So we decided to create a dedicated python script which will

consist of only the required functions and rule off all the unnecessary functions and modules which makes it lighter and lesser complex when compared to Jasper.

Thus this command module serves as the brain to our system. Most of the processes takes place here. This script is made executable and also made to run default on startup using Linux commands.

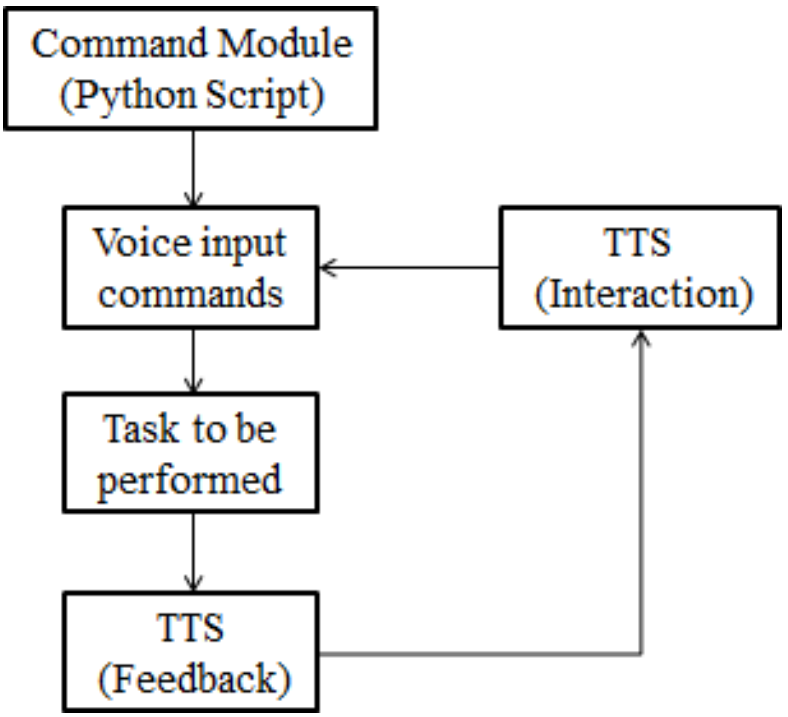
A. Text to Speech Module

This Text to Speech Module is used mainly as a feedback response which is executed at the successful or unsuccessful completion of any task. It uses the Text to Speech Engine whose primary function is to convert the text inputs into Voice outputs.

There are various Text to Speech Engines available both online and offline like eSpeak, Festival, Flite, SVOX Pico TTS, Ivona TTS, Google TTS, Mary TTS, Mac OS X TTS.[4]

eSpeak is an offline Speech synthesizer. It is very handy and light weighted and it is compatible with many platforms. It is open source and has easier configuration process. There are choices of voice from 5 male voices and 4 female voices and also the speed at which the words are uttered can be controlled based on the requirement.

Google TTS uses API which will require an active and fast internet connection to synthesize the text to speech. It sounds good yet the requirement of connectivity doubts the privacy of the users.



Mary TTS is an offline speech synthesizer written based on Java. To use Mary TTS, a server must be set up and it can also be hosted on the same machine. As it takes in a lot of steps, it is quite a hectic process. We have used the eSpeak TTS to get the output voice feedback from the system.

Initially the eSpeak engine is installed in the Raspberry Pi using *'sudo apt-get install espeak'* command. The engine can be called easily through a python script. The speed, voice modulation can be adjusted as per the necessity and comfort.[4]

Fig.6 - TTS Module

The TTS is called for interacting with the user to receive the voice commands to perform tasks as well as to acknowledge the task if completed or not. The eSpeak is an offline synthesizer so it does not require an active internet connection. The Voice outputs from the system are pre- programmed which are put in random orders. The commands are executed based on the users input in a random manner.

The script is coded in such a way that the user feels like he is talking to a person himself. It is programmed in such a way that it gives personalized replies based on the user queries. It also is connected to Google and Wikipedia to find out the general questions that the user asks and interprets the required information and reads it out to the user. The system is designed to give a personalized feel at home.

V.RESULTS AND DISCUSSION

The above mentioned modules were systematically implemented using the necessary hardware and software and tested. The results were promising and this section consists of snap shots of some of the results.

Fig.7 - Virtual Assistant Setup

Fig.7 shows the setup of the prototype of the above stated system. The Raspberry Pi is connected with a USB microphone for audio input and a USB powered speaker for audio output.

The Raspberry Pi needs a 5V/2A power supply to function properly.

```
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "copyright", "credits" or "license()" for more informati
>>>
========== RESTART: /home/pi/Desktop/Final code.py ==
Hello How are you?
How can I help you?
Please respond using the mic...
```

Any changes in the voltage rating affects the board such as rebooting often, not performing efficiently, etc., So it is necessary to give it a proper power supply.

The peripherals used in this prototype does not need a separate power supply as they get their power from the USB ports of the Raspberry Pi.

Fig.8 - Output screenshot of the system

The proposed system was implemented and the outputs were also printed on screen for the follow up of the working process. Fig.8 shows the initial output during the startup of the program. It talks to the user and acquires the output through the microphone in the form of natural language.

```
>>>
================== RESTART: /home/pi/Desktop/Final code.py =
Hello How are you?
How can I help you?
Please respond using the mic...
Processing...
can you please tell me the weather
29.0
mist
How can I help you?
```

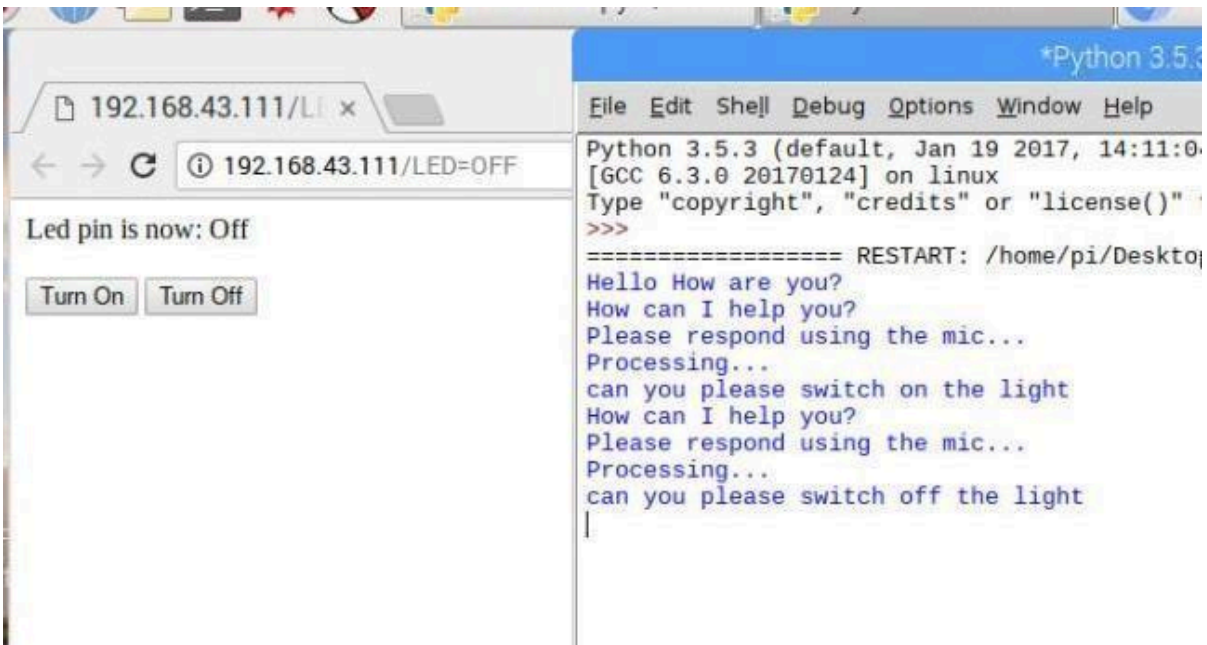


Fig.9 - Fetching Weather

In Fig.9, the user asks the system for the weather. The system fetches the weather from the internet and displays and as well as reads out the weather. Similarly, the system can fetch the date, time, play any music you wish from YouTube, perform searches and much more.

Fig.10 - Controlling NodeMCU (Switch ON)

Fig.10 shows the output where the user asks the system to switch on the lights and as a result a browser window opens up with an IP address along with the ON command. This enables the assigned pin to HIGH which switches the relay ON which in turn switches on the appliance.

Fig.11 - Controlling NodeMCU (Switch OFF)

Fig.11 shows the output where the user asks the system to switch off the lights and similar to the previous output, the

browser opens up with the IP address corresponding to the NodeMCU and makes the assigned pin LOW which switches OFF the relay which in turn cuts off the supply to the appliance which ultimately switches it off. Similarly the motors and high voltage devices can be controlled using the NodeMCU.

The outputs were taken from the Raspberry Pi which runs on the latest version of the Raspbian OS at the time of publication of this paper. A lot of troubleshooting was done before the prototype gave successful results. It takes much patience and repeated trials to make this a successfully working model. The voice recognition is initially slower for new voices. So testing must be done repeatedly to make it get used to the voice. This is also a part of the training process. Also noise reduction codes must be included to avoid the misinterpretation of the words.

VI. CONCLUSION

The existing voice recognition based Home Automation systems basically run by only a fixed set of commands which makes it a stereotypic functioning. The user loses interest as time moves on. To break the stereotype and to overcome all the issues and problems in the existing system, this proposed system takes in a dedicated python script which completely is interactive not with a fixed set of commands but with dynamically changing responses. This kind of approach makes the user feel personal while using the system.[7][8]

The software and hardware implemented in this system are mostly open-source and inexpensive when produced in huge quantities will reduce the cost of production very much which ultimately becomes affordable to almost everyone.

I. FUTURE WORKS THAT CAN BE IMPLEMENTED

The only major drawback in any speech recognition system is its dependency on an active internet connection. It is easier to recognize speech through internet yet at times or at remote places where a reliable connection may not be available all the time, the device becomes difficult to be used.

To overcome this, offline speech recognition system should be

implemented. It will be a great breakthrough when it comes to independent stand-alone home automation devices or virtual assistants because most of the Assistants are connected to the internet.[6]

Another important improvement which can be done is the addition of native languages. It is a hectic process yet it will be useful to all the people who do not know English. It makes this device usable by almost all of the people in the world. By the addition of native languages, the device becomes much more user friendly and easily accessible.

Machine Learning should be implemented completely into this system. By that way, the system will be able to learn new processes by itself and adapt to the user based on its past experiences. This makes it easier for the user to interact with the assistant as well.

Memory can be improved and let the system store the new information gathered from the user and use it in the future if required. Much more modules can be created based on the necessities.