Voice Controlled Home Automation System

Birla Institute of Technology and Science, Pilani

VidyaVihar campus, Pilani, Rajasthan - 333031.

Prepared by :

Meet Parikh

2015A8PS0375P

Guided by :

Dr. Navneet Gupta

Associate Professor and Head

Department of Electrical and Electronics Engineering

BITS Pilani

ACKNOWLEDGEMENT

I, Meet Parikh, a student of Electronics and Instrumentation Engineering at BITS Pilani, am grateful to **Dr. Navneet Gupta,** HOD, EEE/E&I Department, BITS Pilani for his guidance and for providing all the material and direction to progress throughout the duration of my project. It wouldn't have been possible without your able guidance, sheer determination and support. It would be no overstatement to state that this project has been an enriching and a great learning experience for me.

INDEX

1. Voice Controlled Home Automation System

1.1 Problem Statement

To design and develop a digital assistant system capable of answering simple questions and automating day-to-day tasks of a typical household via voice commands.

1.2 Abstract

With time, machines are getting smarter and smarter and the dependence of humans on them to get small tasks done is ever increasing. In such a world scenario, systems which can take voice commands and get the task at hand done are in huge demand. Programs which can answer simple questions like weather report of a specific location, a simple math question, definitions, playing music of your choice etc can be developed. Programs which can complete simple household tasks like switching lights and fans on/off, closing and opening curtains, an all kill to switch off all electrical appliances can also be developed. An amalgamation of all these systems would prove as a perfect home digital assistant. The main aim of this project is to develop a digital assistant just like popular digital assistants such as Siri, Alexa, Google Assistant etc. which would be able to control the major electrical appliances of a household on the basis of voice commands. Some other features like reading out weather reports, solving simple math questions involving addition, subtraction, multiplication and division, etc would also be included. The assistant would also provide voice outputs providing a personalized experience of having a real human assistant.

2. Proposed course of action

2.1 Turning voice to text

The first step in the process would obviously be to create a medium which takes voice commands as input. Since a computer cannot directly understand voice, it becomes necessary to convert that voice into a more computer readable format such as string. An Android application can be developed to get this task done. Almost all smartphones have Google Speech to Text (STT) service nowadays. An Android app can be made which would call this service and turn voice commands to text. Next, this text can be transferred to a microcontroller (in our case, Raspberry Pi) for further processing via socket programming.

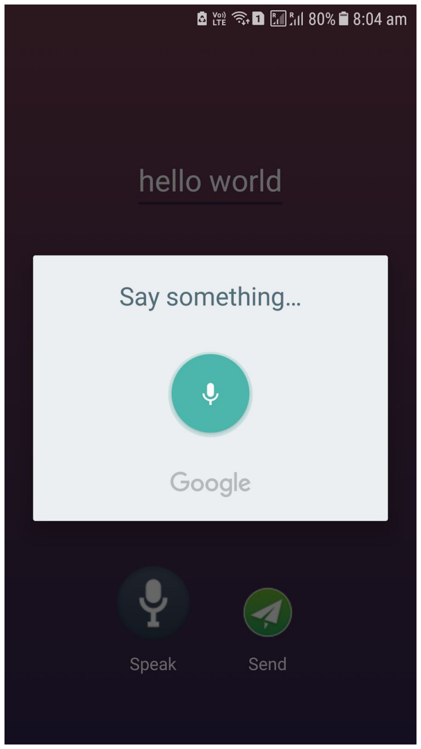

Fig 2.1 : A typical STT Android app screen

2.2 Communicating command text to microcontroller

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket (node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server. Here, the Android app acts as the client while the Raspberry Pi is the server. Hence, server side scripting is to be done in the Raspberry Pi. A listener is to be developed which would keep on checking for data sent via the socket by the Android app.

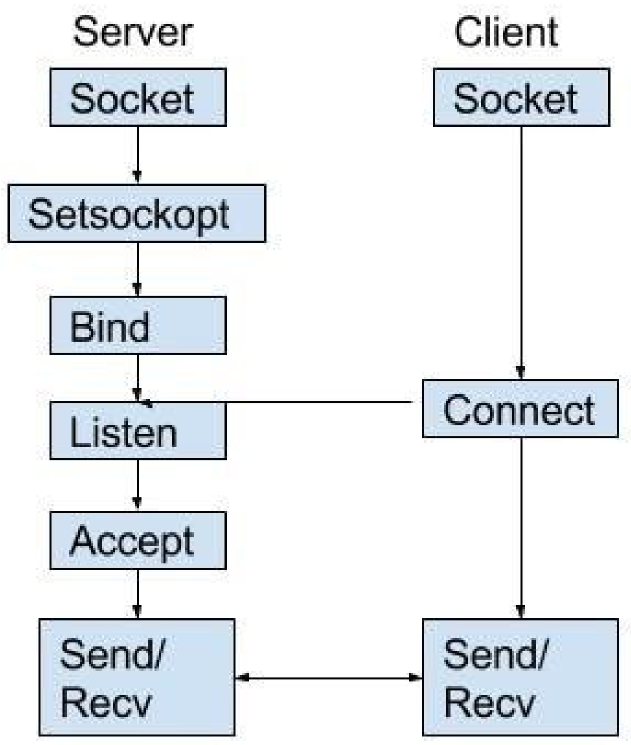

Fig 2.2 : A flowchart on how sockets work

Next we need to perform some processing on the command to understand the intent of the command. Intent classification is required to make the system understand what the user is talking about and what action is being suggested. Using a microcontroller would be the best way to get these tasks done. Since python has a lot of libraries for processing natural language text and since Raspberry Pi supports python, we decided to use a Raspberry Pi.

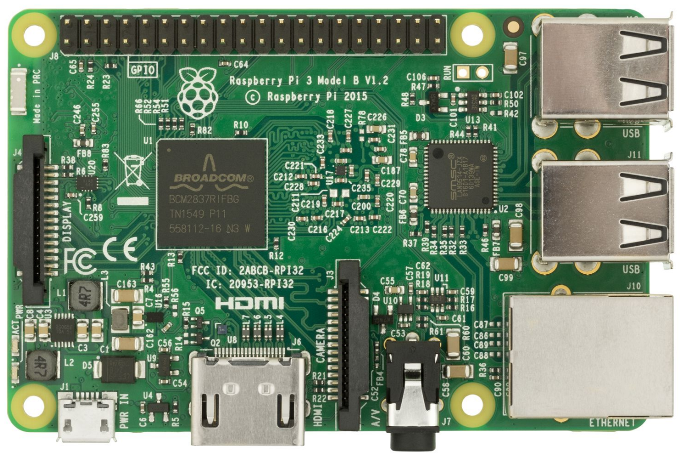

Fig 2.3 : A Raspberry Pi

Fig 2.4 : USB WiFi Adapter

2.3 Pre-processing and Intent classification

Intent classification is the most important aspect of the entire program. Here,the program tries to understand what the user has asked it to do and then, it calls a suitable API to complete the task it thinks has been allotted to it by the user. But before doing this, we can optimize the process by removing those words from the sentence which are not important from intent classification point of view. This would reduce the number of words we need to take care of and hence saves processing time leading to a faster output.

To get this done, we would need a list of such words, commonly known as stop-words. This is where the **Natural Language Toolkit (NLTK)** comes into picture. NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. NLTK has this list of stop-words already. So, we can extract them into a list and then remove them from the command statement, if present. Further, we can also tag the remaining words on the basis with their respective part of speech. This will help us know what words are of most importance for intent classification. This Part of Speech tagging can also be done using the NLTK module.

After this pre-processing is done, it is time for intent classification. Here we can implement if-else statements with conditions like if certain words or list of words are present in the statement, this specific API is to be called. For eg., if (‘weather’, NN) exists in the list of tokens after part of speech tagging, it is most probable that the user is asking for a weather report. The task can be completed by calling the weather API. This way, more and more features can be added and more tasks can be automated.

2.4 Turning text to voice

After the task is done, how would we know? This is the point where it becomes necessary to give voice to our digital assistant. Hence, a Text to Speech engine is required. There are a lot of text to speech engines available for python like pyttsx, gTTS etc. Here, we would be using gTTS (Google Text to Speech) as it has a more human voice as compared to others. gTTS (Google Text to Speech) is a Python interface of the Google Text to Speech API. gTTS makes an audio file using the words that are to be spoken and then plays it. Doing this, we would get an idea of whether the task has been completed or not and it also leads to a personalized experience of having a real human assistant.

3. Current Progress

Currently, the Android app can listen to voice commands and transform them to text. Sockets are in place to communicate the command to the Raspberry Pi. A test code was developed and was run on a local machine. The STT and TTS engines work perfectly. The pre-processing of command via NLTK has been implemented and part of speech tagging has also been included. API calling has been tested by implementing the Wikipedia API. Any question with 'what' in it calls the Wikipedia API. The term in question is searched and the first two lines of its wikipedia page are scraped. These sentences are then spoken out via TTS. Transfer of this entire code onto the Raspberry Pi has been successfully made. Code to control LED lights depicting lights, fans etc and an all kill switch have also been added. Intent classification has been made smoother using NLTK. A lot of other features are yet to be added.

4. Further Recommendations and Improvements

We are presently using sockets to transfer text from the Android app to the Raspberry Pi. Though this is enough for a digital assistant meant for home automation since a home is a classic local network system. However, instead of sockets, a dedicated web server that can be accessed over any network can be set up. This would provide boundless access to the digital assistant.

An engine taking care of multiple input streams can be set up so that the app is not the only way of communicating with the assistant. This way, multiple users can use the assistant via multiple modes, without bothering about downloading the app. Also, for security purposes, speech verification can be included so that the assistant only carries out tasks assigned by predetermined users.

A major scope of improvement lies in the intent classification part of the system. At present, it works on an if-else system, however when the number of features would be fairly high, this might take a lot of time. So, instead of the if-else architecture, we can use datasets to train our model using weights. Through this method, we won't have to go through all if-else statements until a condition is true. We can directly predict the intent on the basis of weighted probability. This would save a lot of time and computing power in the long run.

Next, we can provide a wake-up command just like Ok Google and Alexa. This way, the assistant would listen to us only when we call it’s name.

A chatbot mechanism can be added to give the assistant a more personalized look. A simple Q/A mechanism can be implemented. Personalized AI algorithms can be inculcated so that the assistant understands the user’s preferences

4.1 Proposed Features to be added

A lot of features can be added. Some of my proposed features are :

1. **Weather Report** : The user can ask about the weather of any specific location. In response, a short weather report involving temperature, sky demographic, windy/non-windy etc would be provided as output.
2. **Wolfram Alpha** : Simple Math questions involving addition, subtraction, multiplication, division etc can be solved using the Wolfram Alpha API.
3. **Time information :** The time at a specific location on the globe can be known.
4. **Alarm and Reminders** : Alarms and reminders can be set up. The digital assistant would inform you by playing the note in case of a reminder and playing an alarm tone in case of an alarm.
5. **Music Player** : The digital assistant will play music from the user’s collection.
6. **To-do List** : Another form of a reminder to keep the user on schedule.

6. References

- gTTS docs : https://pypi.python.org/pypi/gTTS
- Pocketsphinx : https://github.com/cmusphinx/pocketsphinx
- NLTK : http://www.nltk.org/
- Speech Recognition : https://pypi.python.org/pypi/SpeechRecognition/
- http://electronicsforu.com/electronics-projects/voice-controlled-home-automation-system
- http://www.instructables.com/id/Voice-Activated-Arduino-Bluetooth-Android/
- https://www.androidhive.info/2014/07/android-speech-to-text-tutorial/
- https://www.allaboutcircuits.com/projects/control-an-arduino-using-your-phone/
- https://create.arduino.cc/projecthub/AnuragVasanwala/home-automation-0dcefc
- https://github.com/kevlened/pyrson
- https://github.com/ab-anand/Digital-Assistants
- https://github.com/KhanradCoder/PyDa
- https://thepihut.com/blogs/raspberry-pi-tutorials/27968772-turning-on-an-led-with-your-raspberry-pis-gpio-pins
- Android Client Python Server : https://www.youtube.com/watch?v=qU1woglr_yU
- https://stackoverflow.com/questions/45851162/create-a-python-server-to-send-data-to-android-app-using-socket
- http://android-er.blogspot.in/2013/12/implement-socket-on-android-to.html
- Android - Raspberry Pi communication : http://www.instructables.com/id/Raspberry-Pi-Android-App-communication/

7. Glossary

- **Automation** : Automation can be defined as the technology by which a process or procedure is performed without human assistance.
- **Speech Recognition** : Speech recognition is the inter-disciplinary sub-field of computational linguistics that develops methodologies and technologies that enables the recognition and translation of spoken language into text by computers.
- **Socket Programming** : Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection.
- **IP address** : An Internet Protocol address (IP address) is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication. An IP address serves two principal functions: host or network interface identification and location addressing.
- **Pre-processing** : Pre-processing is a technique used in machine learning and data mining to make input data easier to work with by performing some modifications on it.
- **Intent Classification** : In computer science, intent classification means applying algorithms to classify input data into what might seem to be the most probable intention behind the said data. It is used mainly in Natural language processing.