Controlling Robots with Voice on Raspberry PI, an Offline Approach using CMU Sphinx

Rutvij Supekar, Susmita Patharkar, Chinmay Karandikar, Abhishek Chaudhari
Vivekanand Education Society's Institute of Technology

***Abstract*—**Voice recognition has found its applications in today's consumer electronics and yet, it remains underdeveloped. This paper shows a collection of embedded systems communicating and collaborating to perform a task. Specifically, our paper demonstrates a Raspberry Pi based voice controlled robot manually operated using CMUSphinx, an offline open source voice processing tool. Our primary aim is to show that numerous flexible voice-controlled applications can be implemented using a Raspberry Pi. The system has two parts; an in-hand device and a robot which performs the specified operations. A voice command is given as input to the microprocessor on Raspberry Pi via a microphone. This microphone is connected to a sound card which is in turn connected to Raspberry Pi's USB port. CMU sphinx will convert the spoken command into a string which is then sent via Bluetooth to an Arduino board placed on the robot. The string received by Arduino is compared with the instructions saved on it and the instruction matching the string received is executed to move the robot.

***Keywords*—** speech recognition, raspberry pi, CMUSphinx, offline voice processing, Arduino, robot control

I. INTRODUCTION

Speech is an analog quantity. When we speak, we produce a continuous wave of energy which is perceived. There is a common misconception that it is built with words made up of phones, which is not the case. It is fundamentally a mixture of two components, fundamentally stables states that lie with mercurial changing states. Using this, it is possible to define classes of sounds or phones which makes up words. The English language has more than a hundred thousand words using a different combination of phones. Another part of speech is something that is called as utterances, which is a combination of words and utterances. The common way of identifying speech is by taking an audio signal and splitting it into multiple parts. This duration of each frame is the duration where one or two words are spoken. All the possible combination of words available are matched with the audio frame and the words. The word or words that match most closely with the input is assigned as the words for that frame. This process is repeated for all the frames of the audio speech and we obtain a text as an output for the audio signal as input.
Since there are many features of a voice signal, it is important to optimize the different features. While processing the signal, we obtain a set of numbers for a particular frame of the entire voice signal, this is called a feature vector. Identification of phones and attribution of words is also dependent on the model that is used. A model follows a particular sequence of steps that it is based on to come at a final decision for obtaining the final information. Hidden Markov Model or HMM model is the model for speech. It defines or gives a relationship between the different states that lie in a sequence varying rapidly with each other. Hence this model is the model most suitable for speech to show the correlation between different states that vary with respect to each other with a certain probability and has proven to be of a practical use. The process of matching can take a lot of time because there can be infinite combinations of words and matching the words with all the available combinations one by one would take a lot of time even if it is to be done by machines. Hence the process of matching the feature vectors with the models available is generally optimized by optimizing the matching process. This would provide sufficient time to obtain the best matching variants and any additional time can be used for further matching and comparing before the time arrives for assignment of words of the next frame. Speech recognition is the technology where the instructions for performing a particular task are acquired from speech. It is one of the practical methods of operating or maneuvering a robot. The speech recognition technique presented in this paper is offline processing and provides a considerable advantage over the online speech processing techniques.

II. SYSTEM MODEL

The signal given to Raspberry Pi is the voice input given to it from the microphone, via a sound card as Raspberry Pi does not have a headphone jack. CMUSphinx, the offline open source voice processing software is used to acquire text from speech. It accesses it via the code written in Python.

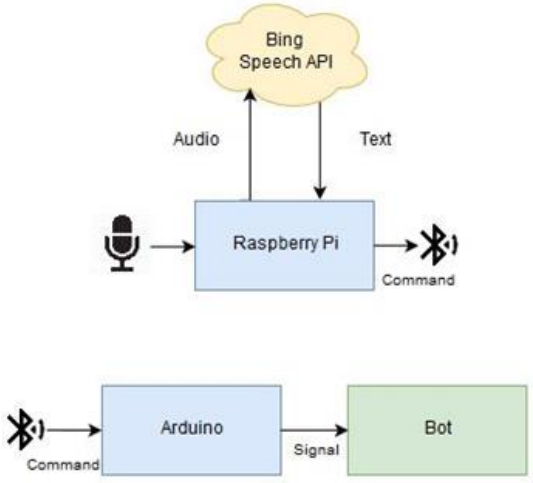

Fig 1: General block diagram of the system

The Raspberry Pi takes the command from the microphone and takes the help of CMUSphinx which further translates it into text in the Raspberry Pi. The converted word or words obtained is one that is explicitly mentioned in the dictionary created. The closest matching word is the given output for the string. After the text has been received on Raspberry, it is transmitted to the Arduino. Arduino and raspberry pi are paired with each other depending on the application, and in this project, Bluetooth is being used as it is a prototype. The command is transmitted on Arduino which energizes the different pins given to different motor, causing the robot to move in a particular way.

III. METHODOLOGY

CMUSphinx is an open source offline voice processing tool developed at Carnegie Mellon University. It can be configured and modified according to a user's own needs. It is also an offline speech API and hence it can be used anywhere without the need of internet. CMUSphinx can be implemented on Linux, windows or iOS and can be implemented on any platform. Since it is an offline API, it needs a library to compare the words it obtained from input to match and give an output from its library. There are several libraries of different languages already available. All of these are open source and can be edited. For users that have to use very limited words, they can create a file of only a selected few words. The text file needs to be uploaded on CMU sphinx site and we can acquire our customized dictionary and library module. For the project implemented, we wish to create a basic prototype that performs the basic movement according to the basic instructions. Hence simple one word commands 'BACK', 'FRONT', 'LEFT', 'RETURN', 'RIGHT', 'ROTATE', 'START', 'STOP'. All these commands need to be written down in a text file and after uploading on the link of official CMU Sphinx website, it provides us with its own pronunciation dictionary and language module.

```
BACK    B AE K
FRONT   F R AH N T
LEFT    L EH F T
RETURN  R IH T ER N
RETURN(2)       R IY T ER N
RIGHT   R AY T
ROTATE  R OW T EY T
START   S T AA R T
STOP    S T AA P
```

Fig 2: Library module for the project

Above is the library module for our project. We can see here that each word has some alphabets associated with it. These alphabets make up the pronunciation of the word. These alphabets basically represent the sound that these words make in ASCII format. Hence each and every word will have its own unique formation of characters to represents its sound. We can see that for the word 'RETURN' there are two lines. This happens because the word 'RETURN' has two pronunciations. In such a case, the same word is mentioned twice in the dictionary module followed by a parenthesis with a number inside it. This number represent which particular pronunciation of the total number of possible pronunciation the line represents.

```
\data\
ngram 1=10
ngram 2=16
ngram 3=8

\1-grams:
-0.7782 </s> -0.3010
-0.7782 <s> -0.2218
-1.6812 BACK -0.2218
-1.6812 FRONT -0.2218
-1.6812 LEFT -0.2218
-1.6812 RETURN -0.2218
-1.6812 RIGHT -0.2218
-1.6812 ROTATE -0.2218
-1.6812 START -0.2218
-1.6812 STOP -0.2218
```

Fig 3: First part of language model

The above picture is the first part of the customized language model created for the project. Language models are very complex to understand as it includes an important parameter, probability. The probability includes those of various words and different composition of words. Theoretically, each and every letter can combine with each and every other letter and this will give rise to a word. Similarly, different words can be combined together to form a sentence. However, the probability of formation of different words are more compared to others and similarly, the probability of different words that can come together or lie next to each other are more compared to others. Hence it is possible to eliminate different words and sentences based on their probabilities of occurrence and can greatly help in identifying the closest word in voice recognition. It tells us that there is a possibility of any combination of words. These probabilities are pre-obtained from a huge amount of pre-existing data. This information of these models is stored in the format known as ARPA format although various text formats and binary formats can save them and hence these models are also known as ARPA models. For usage, it is preferable to transform the ARPA formats into binary formats as even though they are good for storage, for obtaining the words, binary formats have proven to be far more efficient. We see the text '1-grams' 2- grams' and '3-grams' at the beginning of each set. We can summarize it as 'N-grams' where N is a number. If N=1, it is the probability of that word occurring by itself or 1 word only and without combination with any other words. Similarly, 2-grams defines the probability of that word occurring with any other word in a two-word phrase and we can explain the set of 3-grams as the probability of that word occurring with other words in the dictionary module in a 3-word phrase. As it is evidently seen. The overall probability of any word that occurs by itself is much more than occurring in combination with other words. The probability of the same words occurring in two-word phrase or three-word phrase is very similar for this

project. The probability of any word occurring is actually 1+3 where the first of the other three is the word beginning that sentence followed by that particular word which is denoted by <s>, the second being the word at the end of sentence containing the particular word which is denoted by </s> and the final one is an unknown word which is denoted by <UNK>. The last word is not used in very simple models like ours whose dictionary has a very limited vocabulary.

```
\2-grams:
-1.2041 <s> BACK 0.0000
-1.2041 <s> FRONT 0.0000
-1.2041 <s> LEFT 0.0000
-1.2041 <s> RETURN 0.0000
-1.2041 <s> RIGHT 0.0000
-1.2041 <s> ROTATE 0.0000
-1.2041 <s> START 0.0000
-1.2041 <s> STOP 0.0000
-0.3010 BACK </s> -0.3010
-0.3010 FRONT </s> -0.3010
-0.3010 LEFT </s> -0.3010
-0.3010 RETURN </s> -0.3010
-0.3010 RIGHT </s> -0.3010
-0.3010 ROTATE </s> -0.3010
-0.3010 START </s> -0.3010
-0.3010 STOP </s> -0.3010

\3-grams:
-0.3010 <s> BACK </s>
-0.3010 <s> FRONT </s>
-0.3010 <s> LEFT </s>
-0.3010 <s> RETURN </s>
-0.3010 <s> RIGHT </s>
-0.3010 <s> ROTATE </s>
-0.3010 <s> START </s>
-0.3010 <s> STOP </s>
```

Fig 4: Second part of language model

Accuracy is highly dependent on the accent of the language being spoken. It is also possible to adjust different values of certain parameters like threshold and chunk size to edit the amount of data being processed to obtain an output. The threshold is the minimum level of sound that can be perceived and can be recorded by the software. It should be sufficiently high to eliminate noise and low enough to allow the words to be perceived. The entire speech recorded is processed frame by frame and chunk size denotes the amount of data that is processed per frame. It must be sufficiently large to process the speech to obtain one or two words. The code of CMU sphinx can be written in python and availability of various tools that assist in writing a program makes it extremely user-friendly.

IV. WORKING

The various block diagrams with the sequential of flow is given as follows:

Microphone: As seen in the block diagram, the first step is the input given to the microphone. This acts as the interface between data to be obtained and the data to be processed. The raspberry pi does not have audio port of mike and hence we cannot connect the mike directly to raspberry pi. We can make use of the usb ports provided. For this reason, we connect the microphone to a sound card which is then given to raspberry pi. Microphone converts the analog input into corresponding output in electrical form which can be further processed.

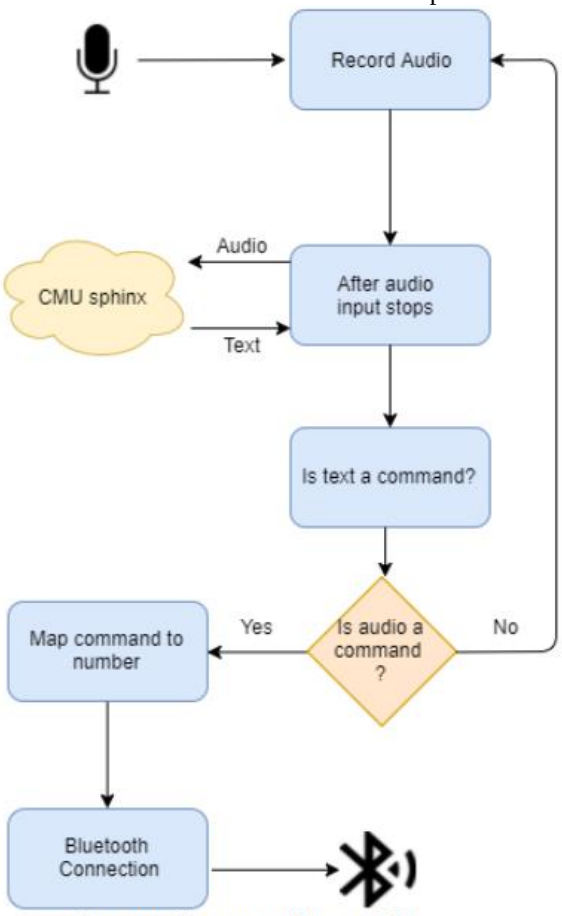

Fig 5: Working on Raspberry Pi side

Recording of audio: The audio needs to be recorded before it is analyzed. The audio is recorded in a wav file and after it is done recording, it sends the WAV file to the speech API for further processing to for get the required command.

CMU Sphinx: CMU sphinx takes the audio recorded by the code to analyze it to convert the input into text. It will try to match the word spoken to the nearest word that it can match with. It compares the audio with the words in its library module explicitly given to it. There is a possibility of the wrong word getting identified if the library created contains many words because of the fact that there are a lot of words having the similar pronunciation. After converting the speech to text, it is then sent back to the program for the further procedure.

Matching of text to command: The words contained in customized library is the basis for matching the text to command. When a customized library is used, any word spoken will be matched to one of the words in the library and

some or the other command is executed. But if one of the pre-existing library and dictionary model is used, the word identified may or may not be associated with a command. If no command is matched, a message saying 'no command found' is displayed on the screen and the code goes back to recording the audio again for taking input. However, if there is a match it goes to the next step.

Map command to number: Every command to be executed is given a code number associated with the command. This is done because it is a simplified way of identifying the command and in this case, less data is transmitted via Bluetooth which would increase the accuracy of transmitted data. When the software gives a perfect or close match with one of the commands given in the code, the code number is transmitted to raspberry pi via wireless serial communication and not the command itself. Hence this step is necessary.

Bluetooth connection: Bluetooth connection from raspberry pi has to be setup up before in hand. When the appropriate command is matched, it can be directly transmitted over the Bluetooth of raspberry pi using Arduino. This completes one cycle of voice processing and transmitting the data from raspberry pi. A new cycle of recording the audio starts again and voice processing continues in loop till termination command is given from command prompt.

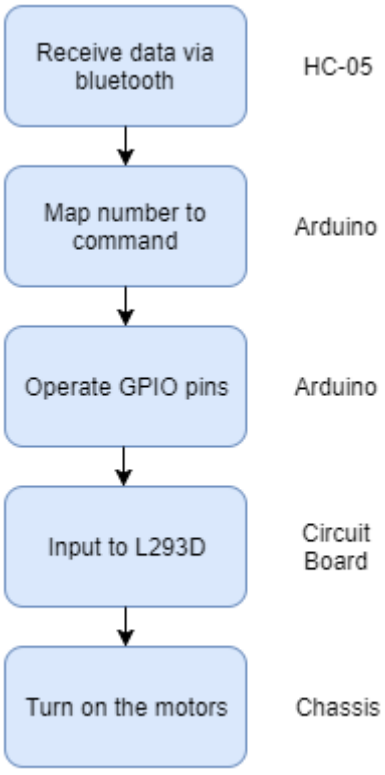

Fig 6: Working on Arduino Side

The Arduino part involves the entire part that is mounted on chassis. It involves the part from receiving the data from raspberry pi on Bluetooth module HC-05 to turning the motors on. This is predominantly the Hardware part of the project which reacts in accordance to the instructions given to the raspberry pi which is obtained after processing.

HC-05: The data is sent from raspberry pi is received at HC-05 Bluetooth module. HC-05 is the Bluetooth module that is used at Arduino side to pair the two devices using Bluetooth. The data received is sent to Arduino on its receiver pin (RX GPIO pin). The Bluetooth module is given power supply from the pins of Arduino. HC-05 is used in slave mode in this project. When HC-05 operates in slave mode, it is only capable of receiving data from another device it is paired. When HC-05 operates in master mode, it is capable of sending the data to another device. The default configuration of HC-05 is in slave mode. There are a set of AT commands which can be given to change the mode of HC-05. If the device is not in slave mode, it can be set in slave mode by giving commands from the Raspberry Pi command prompt after ensuring that the key pin is open.

HC-05 has 4 pins
VCC: Connected to the 5V supply of Arduino
GND: This pin is grounded to the ground GPIO pin of Arduino
RX: This is the receiver pin of HC-05 and is connected to the transmitter GPIO pin of the Arduino. The data to be transmitted from Arduino is received on this pin of HC-05 and is then sent by the module
TX: This is the transmitter pin of HC-05 which is connected to the receiver pin of Arduino. Any data that the Bluetooth module receives is to be given to Arduino and it sends it via the TX pin of HC-05 and is received by Arduino on its RX pin

Serial transmission of corresponding letter of the command to be given takes place from raspberry pi which is received by the Bluetooth module HC-05. The serially transmitted bits are sent by the TX pin of HC-05 and received on Arduino on its RX pin of Arduino from where it further controls the output of GPIO pins of Arduino.

Arduino: The Arduino receives data on the receiver pin from HC-05. This is the code number that was sent from raspberry pi which is finally received on Arduino. The Arduino on receiving the code number searches for the associated commands related to the code number. It accesses these commands and runs them to operate the GPIO pins of Arduino.
There are 14 GPIO pins on Arduino that are available for different connections and we can use any of them except pin 0 and 1. This is because these pins are automatically allocated for usage of HC-05 when it is configured in its program to use it. Setting any other pins as transmitter and receiver for Bluetooth module leads to grounding issues at the pins and it is not possible to make a whole connection using pin 0 and 1. The power supply for motor IC to amplify the input is also taken from any other available GPIO pins of Arduino. The

output from GPIO pins is given to the input of L293D whose output is sufficiently high enough to drive the motors. The Arduino has Bluetooth module HC-05 connected to it, the circuit board containing the 2 motor driver ICs L293D with all its connections and the power supply are all mounted on the bot and stuck in a place by tape so that the wiring doesn't interfere with the operation of the robot. The Bluetooth module receives the code number by serial communication from raspberry pi which it uses to drive the GPIO pins of the Arduino.

Circuit Board

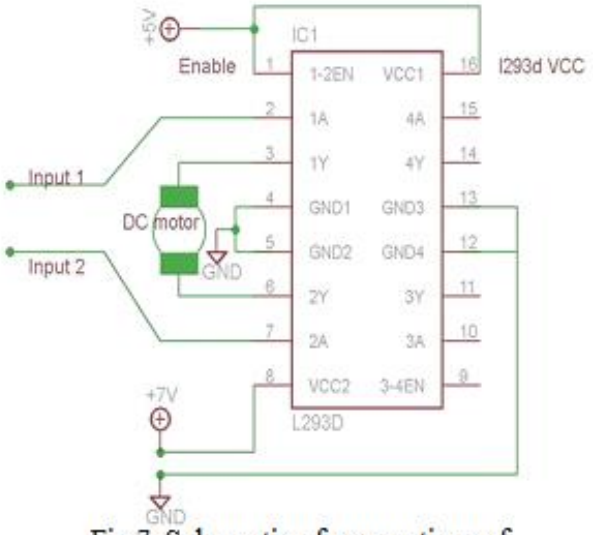

Fig 7: Schematic of connections of IC and motor

The input from GPIO pins is not directly connected to the motors as it is insufficient to drive the motor directly. Hence, we make use of two motor ICs L293D to drive the 4 motors through Arduino. 8 outputs are needed to drive 4 motors and hence 2 ICs are sufficient on the circuit board as each L293D IC has 4 inputs and 4 outputs. It is possible to use a single IC if we can double the power source given to a single IC to satisfy the power requirement. The outputs are of 9V and have sufficient current capability to drive the motors. For the power supply of the circuit board, two 9V batteries are used one for each IC which is sufficient to drive the IC and operate the two motors. It is possible to use a single IC to provided that we get a power source of 36V which can be given to the single IC so that sufficient current drive can be provided to have high acceleration and velocities of each motor. Two inputs in pairs, the ones that lie on the opposite sides in IC L293D are shorted to give one common input. The corresponding outputs are separate but are the exact same. Out of the two outputs given to each motor, one of them is at a higher voltage and the other one is at 0V so that a path is completed for the current to flow in the circuit. Depending on which output is high, the direction of the motor will change.

Motors: The motors used in this project are 4 motors of 270 RPM each, which have been attached to the chassis. The connections from the motor have been taken out and above the bot such that they can be easily accessed. The entire hardware setup has been mounted on the bot so that it is able to move freely. The motors that are attached to the chassis are driven indirectly by the GPIO pins of Arduino. Each motor receives two inputs from L293d motor drive IC on circuit board. Out of the two inputs, one is at a higher voltage and one is at 0V and it acts as the ground. Depending on which input is high, the direction of the motor can change and successful combination on all 8 inputs will make the robot move as expected and programmed.

V. RESULTS

The voice-controlled robot is able to take instructions on raspberry pi and the robot is able to move in any direction according to the command given.

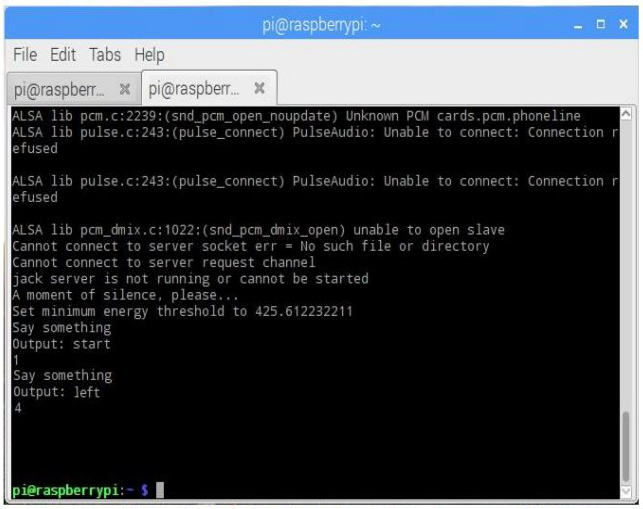

Fig 8: Screenshot of the terminal Window of Raspberry pi showing successful Voice recognition

Raspberry Pi takes input from the microphone and was able to process it to get a string. It matched the obtained word with the words specified in the code to see if there is a match and sent the corresponding keyword of the code word to Arduino using Bluetooth. The pairing of raspberry pi and Arduino could be successfully made using Bluetooth and was able to send the required information on Arduino for further steps. The Arduino board was placed on the bot in which the outputs are given indirectly to the motor for control. The hardware part was also constructed without much trouble. There are a lot of wires and connections so precautions must be taken that all these wires stay in their allocated position and do not come out. All the wires coming out must be taped in proper places to not come in the vicinity of rotating motors or else they will get tangled. It is also advisable to give a single power supply and split it among two opposite motors or among all 4 motors. In this way, the supply given to each motor is symmetrically distributed and we would know if any motor is not operating properly or is faulty.

The project uses Bluetooth as the technology for interfacing raspberry pi and Arduino and this limits the range of

application. The implemented project can have its applications in a range of 20 or 30 meters like a voice-controlled robot assisting in shops, restaurant, cafes or a storage unit. To expand its applications into applications like military, the range of operability needs to be increased and this is possible if Bluetooth is replaced by a similar technology but with a higher range of connectivity like ZigBee.

The accuracy of voice processing using CMU sphinx was successfully increased by varying the customized dictionary for the project and also by adjusting the chunk size and threshold. It is important to vary the parameters with changing environments as a certain set of parameters suitable for one type of surroundings may not be compatible to yield optimum results with other environments.

The output was successfully obtained on the hardware side corresponding to the command given at the output. However, there are certain limitations to the project that does not give us a smooth output. Overcoming these limitations would greatly improve the quality of the output and also reduce the time required to obtain the output.

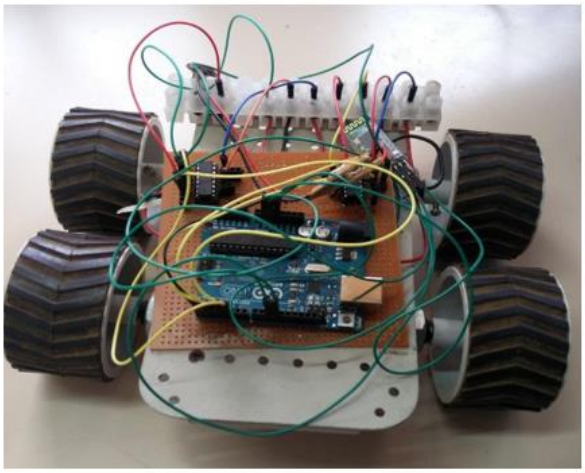

Fig 9: Completed robot with arduino module mounted on it

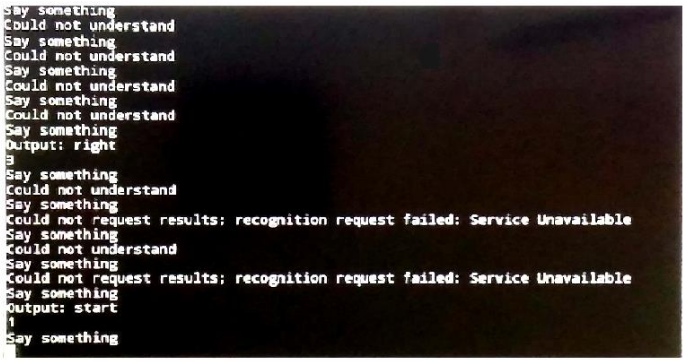

Fig 10: Screenshot of failed voice recognition due to faulty parameters

Above is a picture showing how the accuracy is lowered if the required parameters are not met. Here, it can be seen that we get a successful output on the 10th attempt as the parameters in the code were not set properly. All the noise in the surrounding environment is also being picked up by the microphone and considered as in input. If the amount of noise is too high, it will be very difficult to differentiate the actual input given by the user from the redundant data and we will never obtain the string we wanted from the word spoken. There is a threshold that is set in all speech APIs which serves as the minimum reference level of the input to be processed by the Speech API. This threshold can be adjusted to a level which is higher than the surrounding noise as most of the surrounding noise will not be recorded for processing by this technique. This technique can eliminate a majority of noise but there are certain higher-level disturbances that still get recorded and give difficulty in acquiring text from speech. Such impulse like noises needs to be eliminated from time to time to improve accuracy.

The different steps and adjustments to be made like adjustment in chunk size or threshold will also greatly affect the accuracy of voice processing. Pronunciation of a word also plays a great role in identifying a word irrespective of the type of API used. Two accents of the same language have different voice processing accuracy. Speaking the word with the right accent makes it easier for the speech API tool to recognize it.

Offline speech recognition tool like CMU sphinx or pocket sphinx using python allows for creation of library using only the set of words that are to be used by the user. The user can keep a few specific words so that matching the word to the word spoken will be easier. Although there are some standard language libraries that can be used, creating your one's dictionary and library module works better for applications that use a limited set of words. For better speech recognition, a few words having different pronunciation can be put in a library for better speech recognition.

VI. CONCLUSION

An offline voice controlled robot using CMU sphinx on a new Linux based platform, raspberry pi was successfully implemented. The accuracy of voice processing by CMU is sufficiently high but there is a scope for further improvement.

The project presents a unique vision of the concepts which are used in the field of Embedded Systems. It aims to promote technological innovation to achieve a reliable and efficient outcome in voice processing. With a common digitized platform, these latest voice processing tools will enable increased flexibility in control, operation, and expansion; allow for embedded intelligence and interoperability with other devices. Voice processing is a developed, yet developing field as its ever-increasing challenges with various accents and pronunciation further complicate its algorithms and already existing processing techniques. Voice controlled robot did achieve its function of producing a movement corresponding to the command spoken. However, to increase its accuracy and to add further functionalities to its operation, voice processing by itself needs enhancement to be able to optimize it with hardware and other parallel software.

VII. FUTURE SCOPES

The project implemented serves as a prototype in its field. By changing the robot on the Arduino side, we can increase the no. of possible applications. There are many possible applications like a waiter robot to be used in restaurant, eliminating the need of waiters. A robot can be used for transportation of goods in a factory when and where required. It can be effectively used where there are a wide variety of tasks that are required to be performed with no specific pattern. For this, a set of possible paths that can be taken by the robot needs to be hardcoded. It can be used to add different items to a shopping cart like an assistant or even in a factory for a performing a task on command. These robots can also have its application in assisting at huge storage units where it is practically impossible for humans to physically remove the package. Some other applications in the other fields such as home automation or self-driving car can also be built.

VIII. REFERENCES

[1] Raspberry pi[online]. Available: http/raspberrypi.org.

[2] http://www.speech.cs.cmu.edu/tools/lmtool.html.

[3] https://github.com/Uberi/speech_recognition/blob/master/reference/library-reference.rst

[4] R. Aswinbalaji, A.Arunraja , *"Wireless Voice Controlled Robotics Arm"*, IJETCSE(Vol. 4, Issue 12) http://arfinfotech.com/PROJECTS/voice-controlled-robot.pdf.

[5] K. Kannan, Dr. J. Selvakumar, *"Arduino Based Voice Controlled Robot"*, IRJET(Vol. 2, Issue 1) https://www.irjet.net/archives/V2/i1/Irjet-v2i109.pdf.

[6] http://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/.

[7] http://www.rakeshmondal.info/L293D-Motor-Drive.

[8] https://cmusphinx.github.io/wiki/tutoriallm/.

[9] https://cmusphinx.github.io/wiki/arpaformat.