

**Voice command and control system for Unmanned
Aerial and Ground Vehicles (UAV/UGV)**

*A
Project Report
Submitted in partial fulfilment of the
Requirements for the award of the Degree of*

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

**P. ANURAG VARMA (1602-18-737-006)
SONALI GODAVARTHY (1602-18-737-047)**

Under the guidance of

**Dr. K. RAM MOHAN RAO
PROFESSOR, HOD IT**



**Department of Information Technology
Vasavi College of Engineering (Autonomous)**

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2022

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology



DECLARATION BY THE CANDIDATE

We, **P. ANURAG VARMA, SONALI GODAVARTHY**, bearing hall ticket number, **1602-18-737-006, 1602-18-737-047**, hereby declare that the project report entitled **“VOICE COMMAND AND CONTROL SYSTEM FOR UNMANNED AERIAL AND GROUND VEHICLES”** under the guidance of **DR. K. RAM MOHAN RAO**, Head of Department (Information Technology), Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering in Information Technology**

This is a record of bonafide work carried out by me and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

P. ANURAG VARMA (1602-18-737-006)

SONALI GODAVARTHY (1602-18-737-047)

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology



BONAFIDE CERTIFICATE

This is to certify that the project entitled “**VOICE COMMAND AND CONTROL SYSTEM FOR UNMANNED AERIAL AND GROUND VEHICLES**” being submitted by **P. ANURAG VARMA, SONALI GODAVARTHY**, bearing **1602-18- 737-006, 1602-18-737-047** in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Information Technology is a record of bonafide work carried out by him/her under my guidance.

Dr K. RAM MOHAN RAO

HOD, IT

Internal Guide

Dr. K. Ram Mohan Rao

HOD, IT

External Examiner

भारत सरकार
अंतरिक्ष विभाग
उन्नत आंकड़ा संसाधन अनुसंधान संस्थान
(एड्रिन)
203, अकबर रोड, तारुण्ड
मानविकासनगर पोस्ट
सिकंदराबाद - 500 009, भारत
टेलिफोन : 040-27781271 / 76
फैक्स : 040-27781320



GOVERNMENT OF INDIA
DEPARTMENT OF SPACE
ADVANCED DATA PROCESSING RESEARCH INSTITUTE
(ADRI)
203, AKBAR ROAD, TARUNO
MANOVIKASNAGAR POST
SECUNDERABAD - 500 009, INDIA
Telephone : 040-27781271 / 76
Fax : 040-27781320

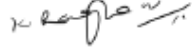
CERTIFICATE

The following students of IV-B.Tech Information Technology, Section IT-A, Vasavi College Of Engineering, Ibrahim Bagh, Hyderabad, have carried out final year internship project, duration from 17-01-2022 to 03-06-2022.

Project Title: Voice Command and Control System for Unmanned Aerial and Ground Vehicles.

S.no	Roll number	Name
1	1602-18-737-006	Penmetsa Anurag Varma
2	1602-18-737-047	Sonali Godavarthy

Dated: 03-06-2022


Dr.K.Raghavendra
Scientist 'SG'
SH SEG/HPCDS

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the project would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them.

We would like to take the opportunity to express our humble gratitude to **DR. K. RAM MOHAN RAO**, Head of Department (Information Technology), Department of Information Technology, our Internal Guide and **Dr K. Raghavendra**, Scientist SG, ADRIN(ISRO), our External Guide under whom we executed this project. We are grateful to their guidance, and constructive suggestions that helped us in the preparation of this project. Their constant guidance and willingness to share their vast knowledge made us understand this project and its manifestations in great depths and helped us to complete the assigned tasks. We would like to thank all faculty members and staff of the **Department of Information Technology** for their generous help in various ways for the completion of this project.

We would also like to take this opportunity to express our gratitude to our seniors **Sanjana Bagundam, Meghana Dundigalla, Sahithi Gaddam, Amrutha Padakanti, Kaparathi Pallavi, P. Bhargav, M. Diva Varshini, B. Jatin, K. Nava Prashant, A. Sai Ramana** having roll numbers 1602-16-737-100, 1602-16-737-084, 1602-16-737-094, 1602-16-737-063, 1602-16-737-088, 1602-17-737-071, 1602-17-737-073, 1602-17-737-077, 1602-17-737-085, 1602-17-737-099 who initiated this project under the guidance of **Dr K. Raghavendra**, Scientist SG, ADRIN(ISRO) and **Mr R. Dharma Reddy, Mrs. L. Divya**, Assistant Professor, Department of Information Technology. The implementation carried out by senior batch helped us to quickly understand the problem statement, get into the code, fix the bug issues and a quick start on to the IOT platforms.

We would also like to express our heartfelt thanks to our HOD **Dr K. Ram Mohan Rao** and our Principal **Dr. S. V. Ramana**, Vasavi college of Engineering, for their cooperation and encouragement for the successful completion of this project.

Finally, we also thank all the staff members, faculty of the Department of Information Technology, Vasavi college of Engineering, and my friends, who with their valuable suggestions and support, directly or indirectly helped us in completing this project work.

This work was implemented using Deep learning server with an experimental setup having IBM Power9 AC922 server with IBM Power AI Framework which was sanctioned by AICTE under the scheme of MODROBS in the year 2017 with a Financial Support of Rs. 19,31,000.00/ and an amount of Rs. 16,32,600.00/- was granted as Research seed money by college management for procurement of Deep Learning Server in Department of Information Technology, Vasavi College of Engineering, Hyderabad. We would like to extend our special thanks of gratitude to the AICTE and Department of Information Technology & college management for providing us with all the facility that was required.

ABSTRACT

Voice/speech is the import things for humans by which a lot communicating things happens with the outside world like ideas, emotions, personality etc. The way the humans communicate with the speech/voice, likewise, the emerging trends in the speech technology driving the use of speech/voice as the medium to communicate with human to machine and machine to machine interaction/communication. However, in such communications of human to machine and machine to machine, there are various challenges to address especially in terms of response of the system in real time and accuracy of the voice/speech being understood by the machine to reduce the risk of malfunctioning of the system.

Internet of Things (IoT) is an emerging trend of technology where in things/devices communicated to each other using network as a communication medium. IoT uses sensors and communication devices to interact with each other for data exchange and interaction to solve the last mile connectivity problems.

Unmanned Aerial / Ground Vehicles aka Drones are the vehicles are remotely pilot vehicles operated without human on board with the help of ground control station and/or joystick. IoT while made as part of UAV/UGV being emerging as IoT Drones (UAV/UGV) to communicate with other devices/things (UAV/UGV) for communication and coordination. The one of the difficulties in operating Drones with joystick always is limited with the fixed number of channels and the communication range. Hence, there is an urge of employing voice-based communication mechanisms to control the devices through other modes of communication like speech or voice.

In this project, we propose to build IoT based speech/voice-based system for controlling UAV/UGV. Here, we discuss one such Deep Learning based speech / voice commanding system and implement the system to control IOT Drones with speech/voice commands through Cellular communication.

Previously, Conventional Models like Hidden Markov Model is used for Speech recognition. There are many limitations of this model such as it is computationally expensive, takes longer time to compute, amount of data required to train the model

is large, it takes trial and error method into consideration, etc. There is a requirement to develop a system with high performance and high accuracy.

Communicating speech over network is also a major challenge. For this, we use Virtual Private Network (VPN - Zero Tier) which allows to communicate over a large range over internet.

To overcome the limitations of HMM, a Deep Learning Model is developed with better performance and high accuracy. The Deep Learning Model for Speech/Voice Recognition using Attention Recurrent Neural Network (AttRNN) is developed. The development of Deep Learning Model for Speech Recognition has achieved an accuracy of 94.5%. The training, testing and validation for the DL Model is carried out using Google Speech Command Dataset V2 which consists of 35-word recognition voices for 35 different commands with each voice audio file of 1 second long. Since, there is requirement of only certain commands, we use 24 of them for the training and testing the model. This DL model provides good performance which can convert voice commands into certain actions.

In this project, we discuss the integration of DL Model with UAV/UGV devices for controlling using speech-based command. The advantage of integration of DL Model and UAV/UGV device enables us to control the device using voice-based command through cellular module with unlimited range. A voice command is given through a microphone of Laptop/Mobile which recognizes the command and hence the action is performed by the device.

Table of Contents

LIST OF FIGURES	-----	xi
LIST OF TABLES	-----	xiii
1. INTRODUCTION	-----	1-3
1.1. Overview	-----	1
1.2. Motivation	-----	1
1.3. Problem Definition	-----	2
1.4. Objective	-----	3
2. LITERATURE SURVEY	-----	4-6
2.1. Review on IoT Devices	-----	4
2.2. Existing Model	-----	5
3. PROPOSED SYSTEM	-----	7-29
3.1. Components and Requirements	-----	7-15
3.1.1. Hardware Components	-----	7
3.1.2. Software Components	-----	10
3.1.3. Dataset	-----	14
3.2. Architecture	-----	16
3.3. Implementation	-----	19-28
3.3.1. Docker Container Configuration in AI Server		19
3.3.2. RNN model with Attention	-----	21-28
3.3.2.1. Architecture for Neural Attention Model		21
3.3.2.2. Pre-processing	-----	23
3.3.2.3. Training	-----	23
3.3.2.4. Adam Optimizer	-----	23
3.3.2.5. Attention Plots	-----	24
3.3.2.6. Mel-scale Spectrogram	-----	25
3.3.2.7. Early Stopping	-----	26
3.3.2.8. Activation Functions	-----	26
3.3.3. Raspberry Pi	-----	28
4. RESULTS / SCREENSHOTS	-----	29-33

5. TESTING	34-37
5.1. Test Cases for RNN Model with Attention	34
5.2. Labelled Drone parts	35
5.3. Real Time testing of IOT Drone	36
6. CONCLUSION AND FUTURE SCOPE	38
REFERENCES	39-40
APPENDIX	41-55

LIST OF FIGURES

Fig 2.1: Labelled Arduino Board	5
Fig 3.1: Raspberry Pi Logo	7
Fig 3.2: Hardware Specifications of Raspberry Pi	8
Fig 3.3: Raspberry Pi 4 Model-B	9
Fig 3.4: Pixhawk	9
Fig 3.5: LTE Module	10
Fig 3.6: Architecture of the Proposed System part-1	16
Fig 3.7: Architecture of the Proposed System part-2	17
Fig 3.8: Architecture of Neural Attention Model	18
Fig 3.9: Use Case diagram	19
Fig 3.10: Raw Waveform of Audio Signal	24
Fig 3.11: Attention Weights	24
Fig 3.12: Spectrogram Visualization	25
Fig 3.13: ReLU Activation Functions	26
Fig 3.14: Leaky ReLU Activation Functions	27
Fig 3.15: SoftMax Mapping	27
Fig 4.1: Model Summary	29
Fig 4.2: Model Training	30
Fig 4.3: Evaluation Scores	30
Fig 4.4: Categorical Accuracy	31
Fig 4.5: Model Loss	31
Fig 4.6: Confusion Matrix	33

Fig 5.1: Classes	-----	35
Fig 5.2: IOT Drone Front-View with labelled parts	-----	35
Fig 5.3: IOT Drone Left Side-View with labelled parts	-----	36
Fig 5.4: IOT Drone placed to start testing	-----	36
Fig 5.5: IOT Drone taking off	-----	37
Fig 5.6: IOT Drone moving forward	-----	37
Fig 5.7: IOT Drone flying	-----	38

LIST OF TABLES

Table 1: Confusion Matrix Metrics	34
---	----

1. INTRODUCTION

This chapter describes Overview, Motivation, Problem Definition and Objective of the project.

1.1. Overview

Deep Learning (DL) is an emerging technology that teaches the computers to do what humans do. It is a next generation technology for classification and detection under supervised class of learning [3]. Deep Learning is being used in various fields like fraud detection, Lung cancer detection, etc. [4]

UAVs and UGVs are being extensively used in various fields like aerial videography, photography, mapping of wildfire, crop monitoring, asset inspection, etc. The technology of UAV/UGV is picking up, controlling of these devices is done through a joystick which has certain actions and can be controlled only within a range.

An Unmanned Aerial Vehicle (UAV), prominently known as Drone, is an airborne device or an airplane worked from a distance by a human administrator or independently by an installed PC [1]. An Unmanned Ground Vehicle (UGV), commonly known as Rover, is a robotic system that operate on land without an onboard human operator [2].

Speech command recognition using DL is also an emerging technique in the field of Internet of Things (IoT). Speech command recognition is the ability of a machine or program to identify or recognize the words spoken aloud. Generally, it is used for information searching and extraction using Speech based commanding.

In this project, we propose a Deep Learning Model for speech command recognition and integrate the Models into IoT for real time application. UAVs and UGVs require efficient and real-time control system for effective use.

1.2. Motivation

Deep learning is an emerging technology which is a part of Machine Learning (ML) methods, commonly work on Artificial Neural Networks for representation. Deep learning techniques have evolved over diverse areas like self-

driving cars, computer vision, medical diagnosis, speech recognition, language processing, etc over the past few years. [5]

With the new headway in advanced advances, the size of informational collections has become too huge in which customary information handling and AI strategies can't adapt to successfully. To change over the information into reasonable structure, profound learning (DL) models have shown extraordinary exhibitions in the new ten years [6]. Profound learning (DL) has upset the fate of man-made reasoning (AI). It has tackled numerous perplexing issues that existed in the AI people group for a long time. Ongoing advances in profound learning have extraordinarily had an impact on the way that figuring gadgets process human-driven content like pictures, video, discourse, and sound.

So deep learning models can be deployed in UAVs and UGVs by using IOT devices, then this device can get input speech and identify the commands using the client-server program and control the drone using the Flight Controller (Pixhawk).

1.3. Problem Definition

Generally, both UAVs and UAGs use Joystick or Ground Control Station (Laptop or Controller) to operate. There are four main drone controls:

- **Roll:** Done by pushing the right stick to the left or right. Rolls the drone, which moves the drone left or right.
- **Pitch:** Done by pushing the right stick forwards or backward. It tilts the drone, which moves the drone forwards or backward.
- **Yaw:** Done by pushing the left stick to the left or to the right. It rotates the drone to left or right.
- **Throttle:** This adjusts the altitude, or height, of the drone.

Operating the UAVs and UGVs by Joysticks has their own pros and cons. UAVs and UGVs lack the recognition of speech command for operation. This speech command recognition can be modelled using Deep Learning.

In this project, we consider the Deep Learning mechanisms with attention RNN model to build a system for recognizing the 23 Speech Commands. It uses a IOT device like raspberry which is present on the drone and has the deep learning model. Based on the voice/speech given on client, it is sent to this IOT which acts like a server and all devices are connected with VPN to form local network. Based on the output of the model, it is used to control the UAVs or UGVs.

1.4. Objective

Both UAVs and UGVs lack in the recognition of speech-based commanding, hence, our objective is to design and develop a Voice/Speech Recognition System for speech-based commanding to control Unmanned Aerial Vehicles (UAV) and Unmanned Ground Vehicles (UGV) which can be controlled over the internet with the help of IOT device.

The system development is based on emerging technologies like Deep Learning. The Deep Learning Model is developed and deployed on IOT device which runs a server program, this can identify the command after giving audio file as input received from client over the internet and this command is transferred to Pixhawk which controls the drone according to the respective command.

2. LITERATURE SURVEY

This chapter gives a review on few IoT Devices and an overview of the existing model is described.

2.1. Review on IoT Devices

The Internet of Things, or IoT, is an arrangement of interrelated figuring gadgets, mechanical and advanced machines, items, creatures or individuals that are given special identifiers (UIDs) and the capacity to move information over an organization without expecting human-to-human or human-to-PC cooperation.

IoT has become quite popular for connecting devices over internet. IoT is making the world around us smarter and more responsive. There is increase in demand of IoT as globe moves towards urbanization with the increase in number of smart cities. Due to IoT there is a positive impact on citizens, businesses, government. [7]

Few of the most popular IoT devices used are Arduino, NodeMCU.

Arduino:

Arduino is an open-source equipment (electronic) and programming and is a solitary board microcontroller. It is utilized for building electronic ventures. It contains both actual programmable circuit board (frequently alluded to as a microcontroller) and a piece of programming, or IDE (Integrated Development Environment) that sudden spikes in demand for your PC, used to compose and transfer PC code to the actual board [8]. Arduino has become very famous because it doesn't require separate piece of equipment to stack a piece of new code onto the board (just we can utilize a USB). Arduino IDE utilizes improved on variant of C++ and gives a standard structure factor that breaks out elements of the miniature regulator into open bundle.

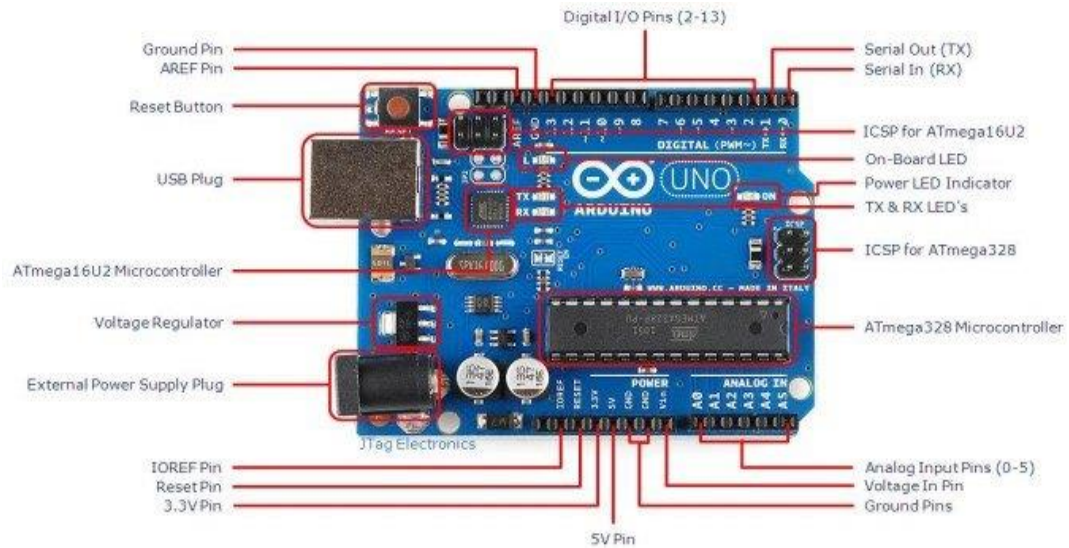


Fig 2.1: Labelled Arduino Board

DJI (Da-Jiang Innovations):

DJI (Da-Jiang Innovations) is a technology company that manufactures drones for commercial purposes like photography, videography. It has camera, robotics, flight platforms, flight control systems, propulsion systems that they even design and manufacture according to the requirement. It gives smooth flight experience with a maximum flight time of 45 minutes and a maximum range of 15kms.

These drones are for commercial purpose and it doesn't support the use of speech commanding feature.

2.2. Existing Model

Hidden Markov Model (HMM) is a widely used technology for speech recognition, before the Deep Learning (DL) technology.

Hidden Markov Model (HMM) is a factual Markov model in which the framework being displayed is thought to be a Markov process. A Markov process is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event.

In HMM, the discourse order is parted into the littlest discernible substances. This large number of elements are addressed as states in the Markov Model. As a word enters the Hidden Markov Model it is contrasted with the most

ideal substance. As per progress probabilities there exist a change starting with one state then onto the next [9].

DRAWBACKS

- HMMs are computationally expensive.
- HMMs take longer time to compute than the Neural Network.
- The HMM needs to be trained on a set of speech sequences and generally requires a larger speech.
- Amount of data required to train the model is large.
- Trial and error method is chosen.
- For a given set of speech sequences, there are many possible HMMs, and choosing one can be difficult.
- HMMs make various assumptions about the speech and as a result fail to generalize.

3. PROPOSED SYSTEM

This chapter describes about the components and requirements of our project that needs to be implemented. Later, Architecture of the whole proposed system is explained. Finally, the detailed steps of the project implementation in various phases are mentioned.

3.1. Components and Requirements

There are various components from both software and hardware (Requirements) that can be used to design a system for Speech Recognition for controlling Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs).

3.1.1. Hardware Components

Raspberry Pi

- What is a Raspberry Pi?

The Raspberry Pi is a part of single-board PCs. It is a modest, charge card (little) measured PC that plugs into a PC screen or Laptop or TV, and utilizations a standard console and mouse. It can do all that you'd anticipate that a work station should do, from perusing the web and playing superior quality video, to making calculation sheets, word-handling, and messing around. The utilization of Raspberry Pi is to master programming abilities, construct equipment projects, do home computerization, execute Kubernetes bunches and Edge registering, and even use them in modern applications.

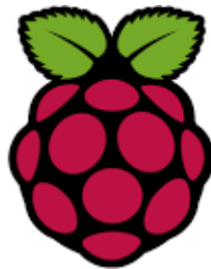


Fig 3.1: Raspberry Pi Logo

- Components of Raspberry Pi:

The raspberry pi board includes a program memory (RAM), processor and illustrations chip, CPU, GPU, Ethernet port, GPIO pins, Xbee attachment, UART, power source connector. Also, different points of interaction for other outer gadgets. It likewise requires mass capacity, for that we utilize a SD streak memory card.

In this undertaking, we are utilizing Raspberry Pi 4 model B with 4GB RAM. It utilizes a Broadcom BCM2711 SoC with a 1.5 GHz 64-digit quad-center ARM Cortex-A72 processor, with 1 MB shared L2 reserve.

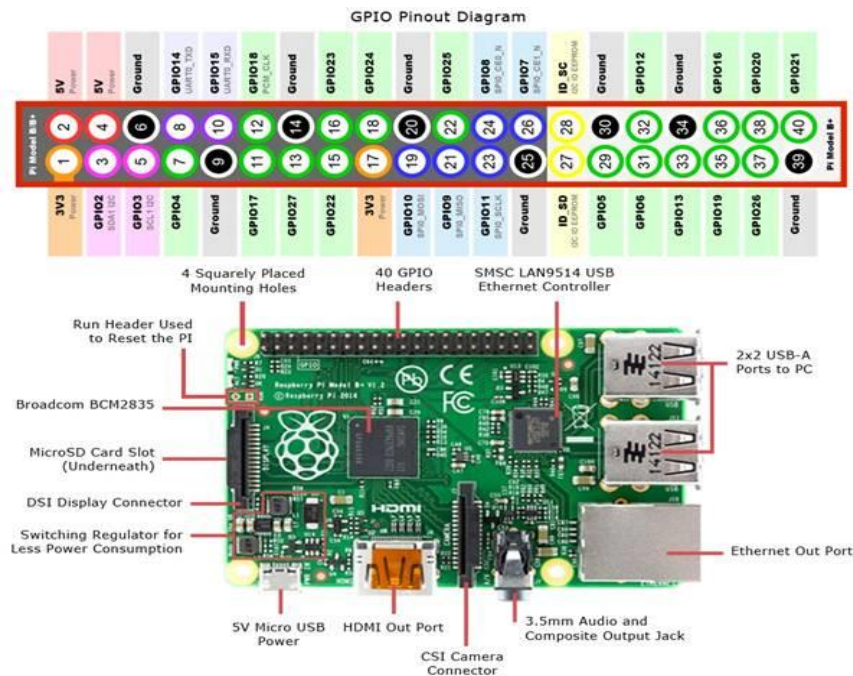


Fig 3.2: Hardware Specifications of Raspberry Pi

GPRS stands for General Packet Radio Service. It is a wireless modem that supports the technology for data transmission. It enables communication between any IOT device and the GSM network.

The LTE Module is a portable travel modem that allows you to connect to the Internet wherever you are. It allows you to access 3G, 4G, 5G data depending on the dongle plan. It is a small modem that plugs into USB port of laptop or desktop computer to add 3G/4G/5G cellular service.



Fig 3.5: LTE Module

3.1.2. Software Components

- **Python:**

Python is an undeniable level, broadly useful programming language. Its accentuation code meaningfulness with critical space. It assists with creating different applications in information science, AI, implanted frameworks, PC vision, web advancement, organizing programming and some more. It can work on different platforms like Windows, Mac, Linux, etc. It can either be run as a functional program or object-oriented programming language. There are several versions of python and the version that is being used in this project is '3.7.x'. [10]

- **Raspberry Pi OS**

The Raspberry Pi Foundation gives Raspberry Pi OS likewise called Raspbian, a Debian-based (Bullseye) 64-bit Linux dissemination , as well as outsider Ubuntu, Windows 10 IoT Core, RISC OS, and Libre ELEC i.e., particular media place circulation. Raspberry Pi OS accompanies more than 35,000 bundles: precompiled programming packaged in a decent organization for simple establishment on your Raspberry Pi.

- **ZeroTier:**

ZeroTier gives network control and P2P usefulness. It utilizes ZeroTier to make items which run on their own decentralized organizations. It makes a 5G-proficient secure P2P network for any IoT gadget that can run on 64MB of RAM [11]

- **NumPy:**

NumPy is the library for Python programming language, adding support for huge, complex exhibits and frameworks, alongside an enormous assortment of undeniable level numerical capacities to work on these arrays. It gives a superior presentation multi-faceted cluster item, and devices for working with these clusters. [12]

- **Pandas:**

Pandas is the famous library composed for the Python programming language for information control and examination. It offers information designs and activities for controlling mathematical tables and time series. It gives exceptionally enhanced execution back-end source code that is absolutely written in Python. We can break down information in pandas with series and information outlines. [13]

- **Matplotlib:**

Matplotlib is a perception plotting library for 2D plots for the Python programming language and its mathematical science expansion NumPy. It gives an article situated API to implanting plots into applications utilizing universally useful GUI tool compartments.[14]

- **Sklearn:**

Scikit-learn is a free open-source programming AI library for the Python programming language. It highlights different characterization, relapse and grouping calculations. It executes a scope of AI, pre-handling, cross-approval and representation calculations utilizing a brought together point of interaction. Straightforward and productive devices for information mining and information investigation. Utilizing sklearn, we import arbitrary woodland classifier to identify extortion exchanges and exactness score to realize the precision pace of recognizing misrepresentation exchanges in the wake of applying the expected calculation. [15]

- **Librosa:**

Librosa is a python bundle for sound examination. It is utilized to recover analysis of music or sound documents. The librosa.core submodule incorporates a scope of regularly utilized capacities. Extensively, center usefulness falls into four classifications: sound and time series tasks, spectrogram computation, time and recurrence transformation, and pitch activities. [16]

- **Keras:**

Keras is an API utilized for Artificial Neural Networks (ANN). Keras is an open-source brain network library written in Python. It is fit for running on top of TensorFlow. It is easy to use, secluded, and extensible. Keras contains numerous executions of brain network building blocks, for example, layers, goals, enactment capacities, enhancers, and a large group of instruments. [17]

- **TensorFlow:**

TensorFlow free and open-source programming library stage for AI. It tends to be utilized for profound brain networks for preparing, testing the model. It has a number related library in view of dataflow and differentiable programming. It is utilized for both innovative work. TensorFlow was created by the Google. [18]

- **Kapre:**

Kapre is utilized for sound preprocessing. It executes time-recurrence changes, standardization, and information increase as Keras layers. It is predictable with 1D/2D tensorflow group shapes.

- **DroneKit:**

DroneKit assists you with making strong applications that discuss straightforwardly with MAVLink vehicles. DroneKit-Python permits engineers to make applications that sudden spike in demand for an installed buddy PC and speak with the ArduPilot flight regulator utilizing a low-dormancy connect.

- **Mission Planner:**

Mission Planner is a ground control station for Plane, Copter and Rover. It is viable with Windows as it were. Mission Planner can be utilized as a setup utility or as a powerful control supplement for your independent vehicle.

- **MAVProxy:**

MAVProxy is a completely working GCS for Uav's, planned as a moderate, versatile and extendable GCS for any independent framework supporting the MAVLink convention (like one utilizing ArduPilot). MAVProxy is a strong order line based "engineer" ground station programming.

- **Mobaxterm**

MobaXterm is your definitive tool kit for remote registering. In a solitary Windows application, it gives heaps of capacities that are custom fitted for software engineers, website admins, IT overseers and essentially all clients who need to deal with their remote positions in a less complex design.

- **SIP Protocol**

The Session Initiation Protocol is a flagging convention utilized for starting, keeping up with, and ending ongoing meetings that incorporate voice, video and informing applications.

- **PyAudio**

PyAudio gives Python ties to PortAudio, the cross-stage sound I/O library. With PyAudio, you can undoubtedly utilize Python to play and record sound on an assortment of stages.

- **Playsound**

A module contains single capacity named playsound(). It requires one contention that is the way to the document which must be played. It works with both WAV and MP3 records.

3.1.3. Dataset

The dataset we have utilized is from Google Speech Commands Dataset. It was made by the TensorFlow and AIY groups to grandstand the discourse acknowledgment model utilizing the TensorFlow API. sd_GSCmdV2 is the dataset we are utilizing to prepare and test the model.

- Training dataset contains 20 commands: Background noise, bed, bird, dog, down, eight, go, house, left, marvin, nine, no, one, seven, Sheila, tree, two, wow, yes. All the commands are in wav format and has 39896 recordings.
- Testing dataset contains 12 commands: silence, unknown, down, go, left, no, off, on, right, stop, up, yes. All the commands are in wav format and has 3081 recordings.

- The all 35 commands: unknown/silence, nine, yes, no, up, down, left, right, on, off, stop, go, zero, one, two, three, four, five, six, seven, eight, backward, bed, bird, cat, dog, follow, forward, happy, house, learn, marvin, sheila, tree, visual, wow
- A custom dataset is used to improve the accuracy, which includes the following 23 commands: nine, yes, no, up, down, left, right, on, off, stop, go, zero, one, two, three, four, five, six, seven, eight, backward, follow, forward
- Silence- Sample with no word
- Unknown- All other words apart from the above-mentioned words are labeled as “unknown”
- Background noise- There are also files that contain background noise

3.2. Architecture

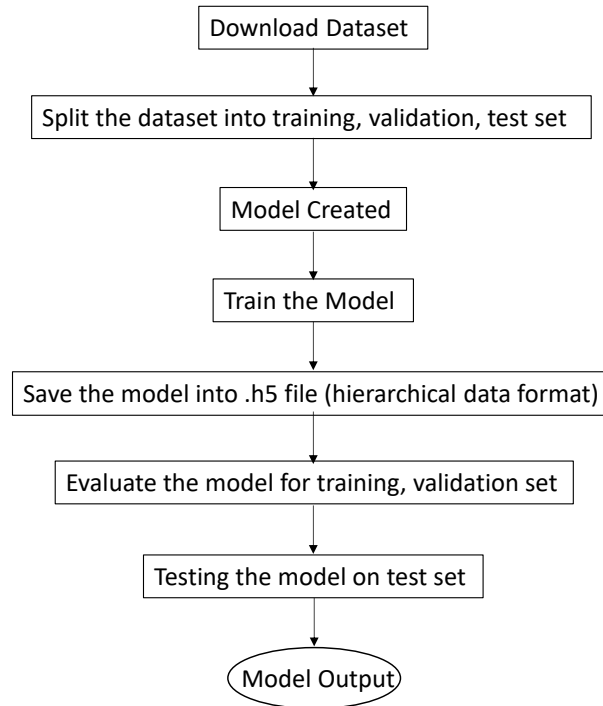


Fig 3.6: Architecture of the Proposed System part - 1

The above figure represents the 1st part of architecture which tells about the model creation:

1. The google speech command dataset is downloaded.
2. The data is split into training and testing sets.
3. The attention RNN model is created.
4. Training of the model is done on the dataset.
5. The model is saved after each epoch if there is increment in accuracy based on early stopping.
6. The model is evaluated for training, validation sets.
7. The model is tested on the test dataset, along with different accuracy measures, etc.
8. The final model is given as output.

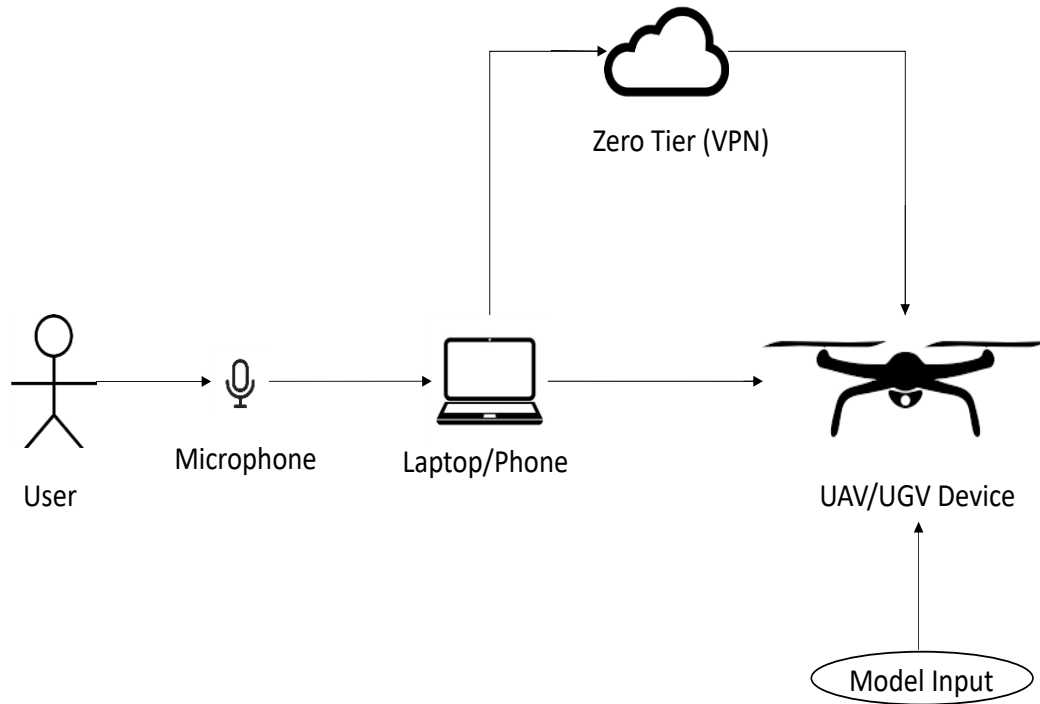


Fig 3.7: Architecture of the Proposed System part - 2

The above diagram represents the 2nd part of architecture which tells about the working of whole system:

9. The input voice command is given from any client which has python and is connected to internet.
10. The voice is recorded in the client and all the clients are connected to each other by using a vpn as a local network.
11. The model obtained from part 1 of the architecture, is saved in the raspberry pi.
12. The raspberry pi is connected to internet via cellular module with sim card which is attached to the drone and acts as server.
13. The client sends the recorded audio to the raspberry pi which has the attention RNN model running, and the audio is given as input, which gives the predicted command as output.
14. All the steps take in real time and based on the command, it is sent to Pixhawk which is connected to raspberry pi, and it implements the command on the drone.

15. Based on what instruction the Pixhawk gives to the drone, the drone works accordingly.

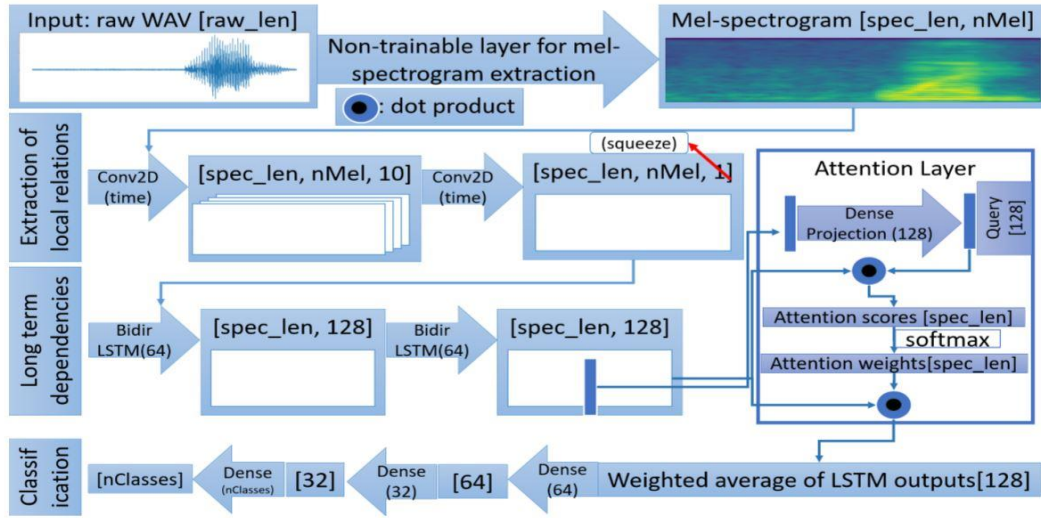


Fig 3.8: Architecture of Neural Attention Model

The model starts from downloading the dataset i.e., Google Speech Command Dataset V2. This dataset is split into training set, validation set, test set.

Then the AttRNN Model is created. In this Model, the audio (raw WAV) file is converted into Mel-spectrogram using non-trainable layers. Later, this Mel-spectrogram is used to extract local features in audio files by applying this to a set of two convolution layers. Later, it is passed through two bidirectional Long Short-Term Memory (LSTM) units, which is used to detect long term dependencies in the audio file. Then, one of the output vectors of LSTM is passed to the Attention layer. Attention layer takes the input and passes it through the Dense Layer with SoftMax activation function, which is used to identify the most important or relevant part of the audio file. Generally, the command is expected at the middle of the audio file, but it can be random and since we use two bidirectional LSTM, it carries enough memory. Lastly, the weighted average of LSTM is computed can fed through two fully connected dense layers for classification.

Then, this model is trained. This model is saved into hierarchical data format (.h5) file. Then the model is evaluated on training set and validation set. Later, the model is tested on test set which gives different accuracy measures, confusion metrics, etc. Finally, this model is given as the final output.

Use Case Diagram:

There are two modules: the Speech Recognition model and the Command Controlling model. The Speech Recognition process starts with collecting data from the dataset. Then the dataset is split into 3 sets as training set, validation set, test set. Then the Deep Learning model is trained on the data, and its working is validated and tested. Once the model is ready, it is integrated into a client-server program which is loaded into the Raspberry Pi. In the Command Controlling system, the Rover or Drone user gives a voice command, then the command interface will convert that speech command into text by using the deep learning model. Then a flight controller such as Pixhawk in the case of drones feeds the output text to the Rover or Drone thereby controlling the drone.

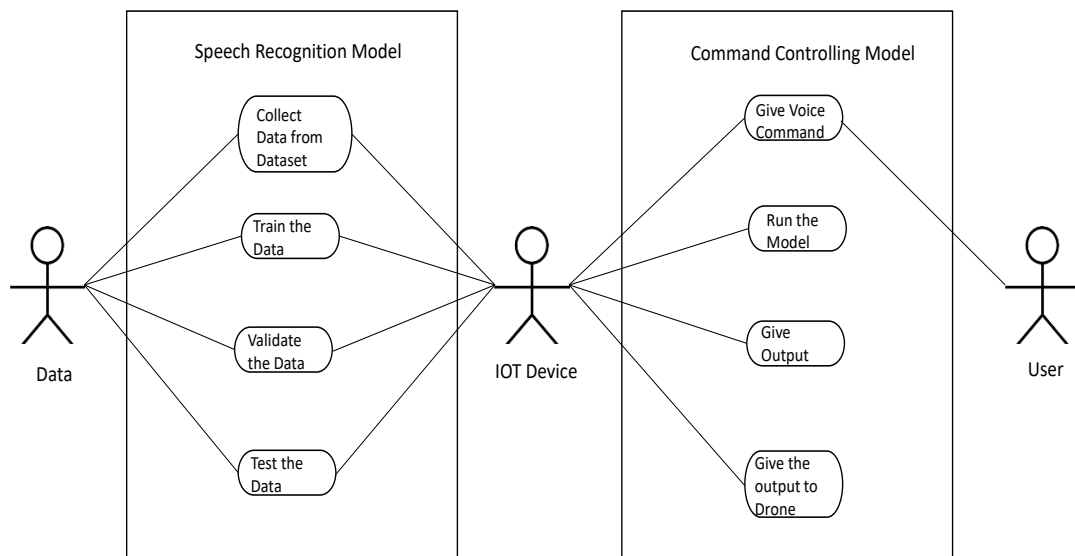


Fig 3.9: Use Case Diagram

3.3. Implementation

3.3.1. Docker Container Configuration on AI Server

A Docker holder is a lightweight, independent, executable bundle of programming that incorporates everything expected to run an application: code, runtime, framework devices, framework libraries and settings. It empowers

engineers to bundle applications into holders - normalized executable parts consolidating application source code with the working framework (OS) libraries and conditions expected to run that code in any climate.

We install required libraries and save that to the docker image. Whenever a new library is installed on AI server, it will be available for that session and whenever code needs to be executed on the server, it can be executed directly without installing all the packages require libraries again.

Steps to create the Docker image:

First, connect to the AI server using MobaXterm. Then install all the required libraries as following-

1. Keras:

Command: **pip install keras**

2. Update apt get:

Command: **apt get update**

3. Librosa:

Command: **pip install librosa**

4. Update pip:

Command: **python3 -m pip install --upgrade pip**

5. Kapre:

Command: **pip install kapre**

6. Pandas:

Command: **pip install pandas**

7. Pocketphinx

Command: **pip install pocketphinx**

8. Pyaudio

Command: **pip install pyaudio**

9. Tensorflow

Command: **pip install tensorflow**

8. Dronekit

Command: **pip install dronekit**

Training time on AI server:

- Our model took approx. 60 mins to train on the AI server when trained for 100 epochs, but due to early stopping it stopped after training 25 epochs.

3.3.2. RNN model with Attention

A Neural Network comprises of various layers associated with one another, dealing with the construction and capacity of a human mind. It gains from tremendous volumes of information and utilizations complex calculations to prepare a brain net. There are many kinds of Neural organizations like Perceptron, Convolution Neural organization (CNN), Recurrent Neural Network (RNN), Recurrent Neural Network with consideration (attRNN), Long Short-Term Memory (LSTM), Feedforward Neural Network, and so on. [19]

Intermittent Neural Network (RNN) is a piece of Artificial Neural Networks (ANN) where the associations between certain hubs structure a coordinated or undirected chart. This displays fleeting unique way of behaving. Repetitive Neural Network (RNN) are a sort of Neural Network where the result from past advance is taken care of as contribution to the ongoing advance. In Artificial Neural Networks (ANN), all the inputs and outputs are independent of each other. But, in some cases there is a need to remember the previous words e.g., prediction of upcoming word in a sentence. Hence, RNN comes into existence, which solves this issue with the help of a Hidden Layer. The important feature is hidden layer which remembers the information in sequence.

Consideration system assists with seeing all stowed away states from encoder succession for making expectations dissimilar to vanilla Encoder-Decoder approach. It is a component joined in the RNN permitting it to zero in on specific pieces of the info succession while foreseeing a specific piece of the result grouping, empowering simpler learning and of better caliber.

An AttRNN has various advantages like it uses important wights to identify which part of audio input is important, it controls vanishing/exploding gradients.

3.3.2.1. Architecture for Neural Attention Model

The model saves our command after recognition of word “Milind” in audio recorded formatted of WAV file. The sound document contains a solitary word order that can be anyplace during the 1s length of the WAV record, so the model can group a depends on proper locale of interest. Hence, we utilize the consideration component. The contribution to the model is the crude WAV information with unique testing pace of ~ 16 kHz.

In this Model, the audio (raw WAV) file is converted into Mel-spectrogram using non-trainable layers. Later, this Mel-spectrogram is used to extract local features in audio files by applying this to a set of two convolution layers. [20] Later, it is passed through two bidirectional Long Short-Term Memory (LSTM) units, which is used to detect long term dependencies in the audio file. Then, one of the output vectors of LSTM is passed to the Attention layer. Attention layer takes the input and passes it through the Dense Layer with SoftMax activation function, which is used to identify the most important or relevant part of the audio file. Generally, the command is expected at the middle of the audio file, but it can be random and since we use two bidirectional LSTM, it carries enough memory. Lastly, the weighted average of LSTM is computed can fed through two fully connected dense layers for classification.

3.3.2.2. Pre-processing

The WAV file are converted into numpy arrays which is efficient to load into keras. We use kapre library which provides versatility to change spectrogram and mel-frequency parameters without having to pre-process the audio in any way.

3.3.2.3. Training

The Neural Attention model was trained for a maximum of 100 epochs. The model with the best accuracy performance on the validation set was saved and training was stopped at 25 epochs if no improvement was made. The training was done using the “adam” algorithm.

3.3.2.4. Adam Optimizer

Versatile Moment Estimation is a calculation for improvement strategy for inclination drop. The strategy is truly productive while working with enormous issue including a ton of information or boundaries. It requires less memory and is effective. Naturally, it is a blend of the 'inclination drop with energy' calculation and the 'RMSP' calculation. Adam enhancer includes a blend of two inclination plummet strategies: Momentum and Root Mean Square Propagation. Energy: This calculation is utilized to speed up the inclination plummet calculation by thinking about the 'dramatically weighted normal' of the angles. Utilizing midpoints causes the calculation to meet towards the minima in a quicker pace. Root Mean Square Propagation (RMSP): Root mean square prop or RMSprop is a versatile learning calculation that attempts to further develop AdaGrad. Rather than taking the combined amount of squared angles like in AdaGrad, it takes the 'remarkable moving normal'.

Adam Optimizer acquires the qualities or the positive credits of the over two strategies and expands upon them to give a more enhanced slope plummet.

Adam takes 4 hyperparameters:

- Alpha: Known learning rate or step size. Larger the value, faster the initial learning. Smaller the value, slower the initial learning
- Beta1: The exponential decay rate for the first moment estimates
- Beta2: The exponential decay rate for the second-moment estimates (used in sparse gradient)
- Epsilon: To prevent any division by zero in the implementation (usually small) [21]

3.3.2.5. Attention Plots

Figures below show attention weights along with raw waveform, attention weights and mel-scale spectrogram. Attention weights are plotted in a log-scale.

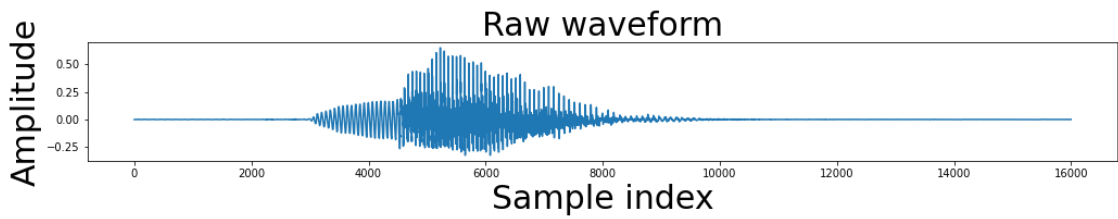


Fig 3.10: Raw Waveform of Audio Signal

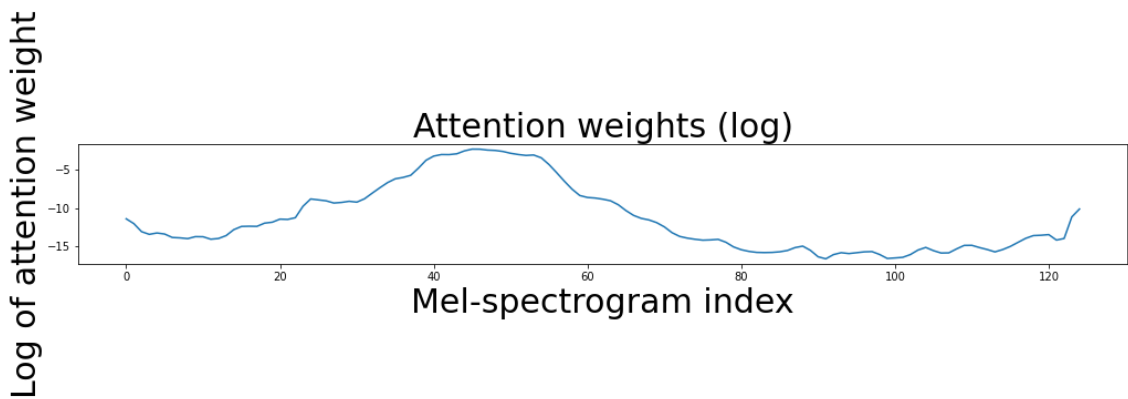


Fig 3.11: Attention Weights

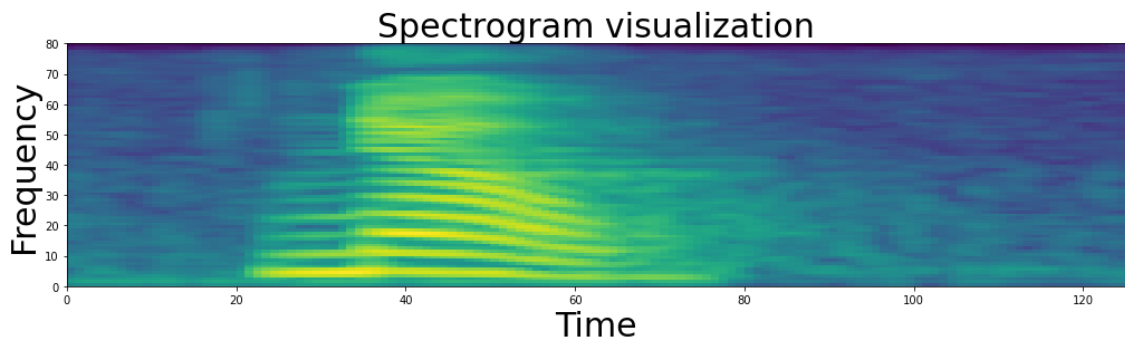


Fig 3.12: Spectrogram Visualization

3.3.2.6. Mel-scale Spectrogram

Mel spectrogram is a spectrogram that is changed over completely to a Mel scale. A spectrogram is a representation of the recurrence range of a sign, where the recurrence range of a sign is the recurrence range that is contained by the sign. The Mel-Scale is the consequence of some non-direct change of the recurrence scale. This Mel Scale is developed with the end goal that hints of equivalent separation from one another on the Mel Scale, moreover "sound" to people as they are equivalent in separation from each other. The Mel Spectrogram is a Spectrogram with the Mel Scale as its y-pivot.

Steps to compute Mel-Scale spectrogram:

1. Take samples over time to digitally represent an audio signal
2. Map the audio signal from the time domain to the frequency domain using the fast Fourier transform (FFT) and perform this on overlapping windowed segments of the audio signal.
3. Convert the y-axis (frequency) to a log scale and the colour dimension (amplitude) to decibels to form the spectrogram.
4. Map the y-axis (frequency) onto the Mel scale to form the Mel spectrogram.

3.3.2.7. Early Stopping

Early halting is a strategy that permits you to determine huge number of preparing ages and quit preparing once the model execution quits enhancing the approval dataset. An age demonstrates the quantity of passes of the whole preparation dataset the profound learning calculation has finished. Datasets are typically assembled into clusters when how much information is exceptionally huge. Early pausing and model designated spots are the call-backs to quit preparing the brain network with flawless timing and to save the best model after each age. A designated spot is a halfway dump of a model's whole inner state which incorporates its loads, current learning rate, and so on.

3.3.2.8. Activation Functions

Activation Functions that are commonly used based on few desirable properties like Non linearity, range, continuously differentiable, etc.

ReLU (Rectified Linear Unit) Activation Function: The rectified linear (ReLU) activation function is a linear function that will output the input directly if it is positive, otherwise, it will output zero. It is a default activation function used in many models because it is easy to use it.

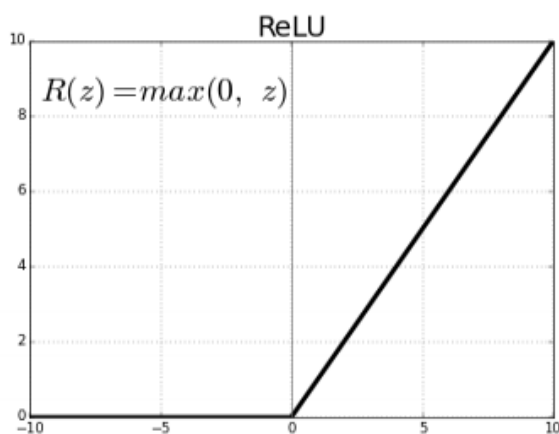


Fig 3.13: ReLU Activation Functions

Range: $[0, \text{infinity})$.

Leaky ReLU: Leaky rectified linear (ReLU) has a small deviation from zero i.e., has negative values. Instead of the function being zero when $x < 0$, a leaky ReLU will instead have a small negative slope.

$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases}$, where α is a small constant.

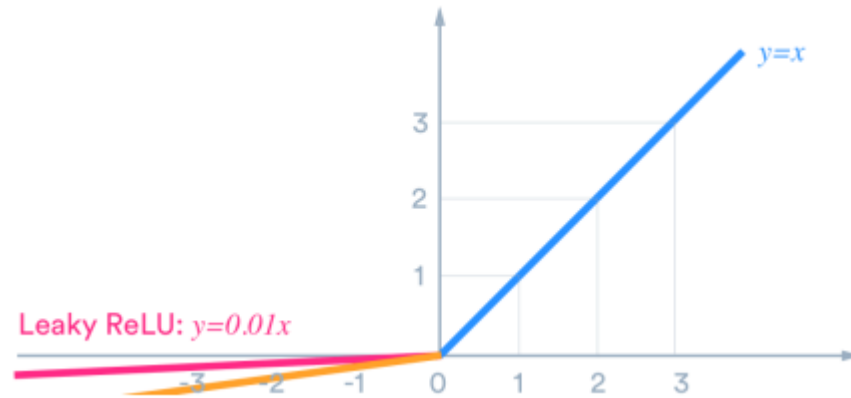


Fig 3.14: Leaky ReLU Activation Function

Softmax Activation Function: For ReLUs to not vanish, the sigmoid function can be applied. The softmax work maps the results of every unit to be somewhere in the range of 0 and 1, very much like a sigmoid capacity, however it separates each result with the end goal that the all out amount of the results is equivalent to 1.



Fig 3.15: Softmax Mapping

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$$

3.3.3. Raspberry Pi

After developing the above stated RNN model, it is stored in the Raspberry Pi. This stored model is used to predict the speech commands. Below are the steps of implementation-

1. Raspberry Pi Hardware Setup

Raspberry Pi 4 Model-B is being used in this project. In this step, all the required hardware components are connected properly. The required hardware components include- Raspberry Pi board, Heat sinks, Power cable, Ethernet cable, SD card

2. Boot the Raspbian OS

First the SD card is loaded with NOOBS which contains Raspbian OS and then it is connected to Pi through which when connected to monitor, the Pi loads the Raspbian OS and then boots.

3. Install the required packages

After booting the OS, all applications are upgraded. Both Python and pip versions are checked, and a Virtual environment is created.

4. To import and use the model, the required source code has been written.
5. Finally, we run the program.

4. RESULTS / SCREENSHOTS

This chapter describes the results of the various phases of the project implementation.

- Model Summary:

Model: "model"			
Layer (type)	Output Shape	Param #	Connected to
input (InputLayer)	[(None, None)]	0	[]
reshape (Reshape)	(None, 1, None)	0	['input[0][0]']
mel_stft (Melspectrogram)	(None, 80, None, 1)	1091664	['reshape[0][0]']
mel_stft_norm (Normalization2D)	(None, 80, None, 1)	0	['mel_stft[0][0]']
permute (Permute)	(None, None, 80, 1)	0	['mel_stft_norm[0][0]']
conv2d (Conv2D)	(None, None, 80, 10)	60	['permute[0][0]']
batch_normalization (BatchNormalization)	(None, None, 80, 10)	40	['conv2d[0][0]']
conv2d_1 (Conv2D)	(None, None, 80, 1)	51	['batch_normalization[0][0]']
batch_normalization_1 (BatchNormalization)	(None, None, 80, 1)	4	['conv2d_1[0][0]']
squeeze_last_dim (Lambda)	(None, None, 80)	0	['batch_normalization_1[0][0]']
bidirectional (Bidirectional)	(None, None, 128)	74240	['squeeze_last_dim[0][0]']
bidirectional_1 (Bidirectional)	(None, None, 128)	98816	['bidirectional[0][0]']
lambda (Lambda)	(None, 128)	0	['bidirectional_1[0][0]']
dense (Dense)	(None, 128)	16512	['lambda[0][0]']
dot (Dot)	(None, None)	0	['dense[0][0]', 'bidirectional_1[0][0]']
attSoftmax (Softmax)	(None, None)	0	['dot[0][0]']
dot_1 (Dot)	(None, 128)	0	['attSoftmax[0][0]', 'bidirectional_1[0][0]']
dense_1 (Dense)	(None, 64)	8256	['dot_1[0][0]']
dense_2 (Dense)	(None, 32)	2080	['dense_1[0][0]']
output (Dense)	(None, 36)	1188	['dense_2[0][0]']
Total params: 1,292,911			
Trainable params: 201,225			
Non-trainable params: 1,091,686			

Fig 4.1: Model Summary

This summarizes the proposed AttRNN model. It shows us the layers used with their order. It also shows the shape of each layer along with the number of parameters.

- Model Training:

```
Epoch 00023: val_sparse_categorical_accuracy did not improve from 0.95525
2056/2056 - 92s - loss: 0.0158 - sparse_categorical_accuracy: 0.9954 - val_loss: 0.2833 - val_sparse_categorical_accuracy: 0.9520 - lr: 4.0000e-04 - 92s/epoch - 45ms/step
Changing learning rate to 0.0004
Epoch 24/100

Epoch 00024: val_sparse_categorical_accuracy did not improve from 0.95525
2056/2056 - 93s - loss: 0.0171 - sparse_categorical_accuracy: 0.9949 - val_loss: 0.2852 - val_sparse_categorical_accuracy: 0.9542 - lr: 4.0000e-04 - 93s/epoch - 45ms/step
Changing learning rate to 0.0004
Epoch 25/100
Restoring model weights from the end of the best epoch: 15.

Epoch 00025: val_sparse_categorical_accuracy did not improve from 0.95525
2056/2056 - 92s - loss: 0.0148 - sparse_categorical_accuracy: 0.9955 - val_loss: 0.3388 - val_sparse_categorical_accuracy: 0.9457 - lr: 4.0000e-04 - 92s/epoch - 45ms/step
Epoch 00025: early stopping
```

Fig 4.2: Model Training

We have initialized the number of epochs to 100 but due to early stopping it stopped at 25 epochs. Early stopping and model checkpoints are the call-backs to stop training the neural network at the right time and to save the best model after every epoch.

- Evaluation Scores:

```
Evaluation scores:
Metrics: ['loss', 'sparse_categorical_accuracy']
Train: [0.026131467893719673, 0.9925978779792786]
Validation: [0.19125361740589142, 0.9552469253540039]
Test: [0.1913890391588211, 0.9587224721908569]
```

Fig 4.3: Evaluation Scores

This shows the accuracy and loss of the AttRNN model on train, validation and test sets. The accuracy of the model on train set is 0.99, in validation is 0.95 and on test set is 0.95. The loss of the model on train set is 0.02, in validation is 0.19 and on test set is 0.19.

- Categorical Accuracy:

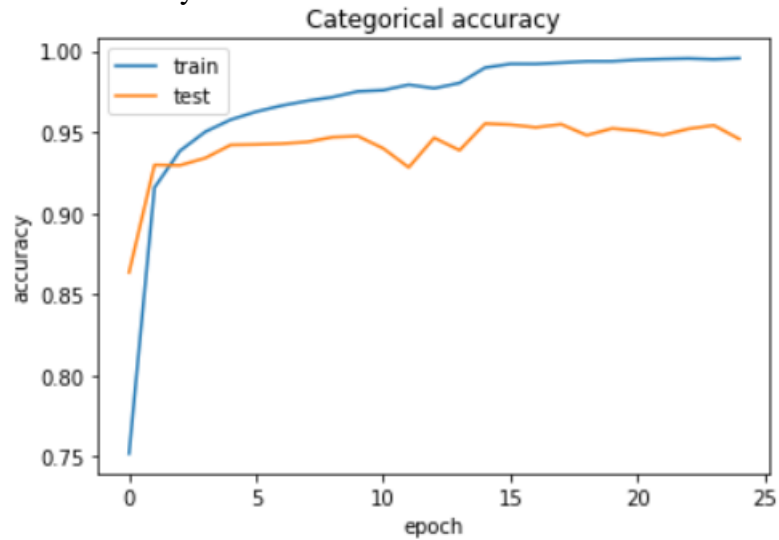


Fig 4.4: Categorical Accuracy

This shows the graphical representation of the accuracy of the AttRNN model on train and test sets. Categorical Accuracy calculates the percentage of predicted values that match with actual values.

- Model Loss:

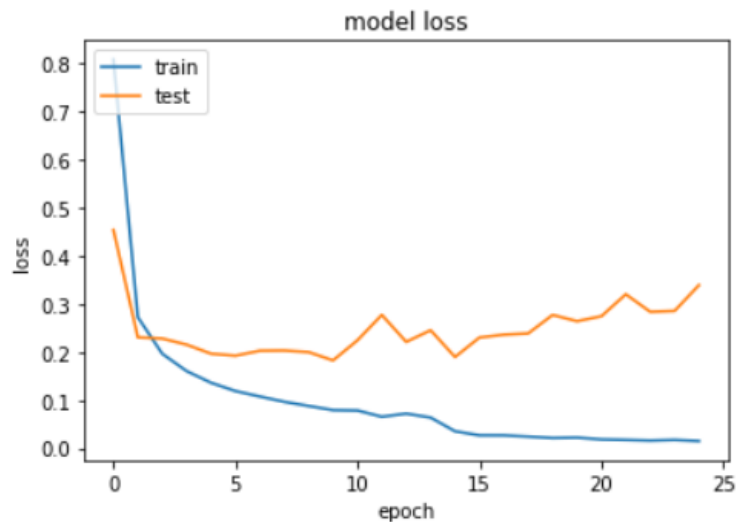


Fig 4.5: Model Loss

This shows the graphical representation of the loss of the RNN model on train and test sets. Loss is a number that indicates how bad the model's prediction was on a single example.

- Confusion Matrix:

"Confusion Matrices" are a nice way to visualize how our model is doing over all classes. Confusion matrices are presented for the 24 custom word recognition task on Google Speech Command dataset V2.

The diagonal entries in the confusion matrix correspond to correctly classified examples i.e., True Positives (TP)

A custom dataset is used to improve the accuracy, which includes the following 23 commands: nine, yes, no, up, down, left, right, on, off, stop, go, zero, one, two, three, four, five, six, seven, eight, backward, follow, forward

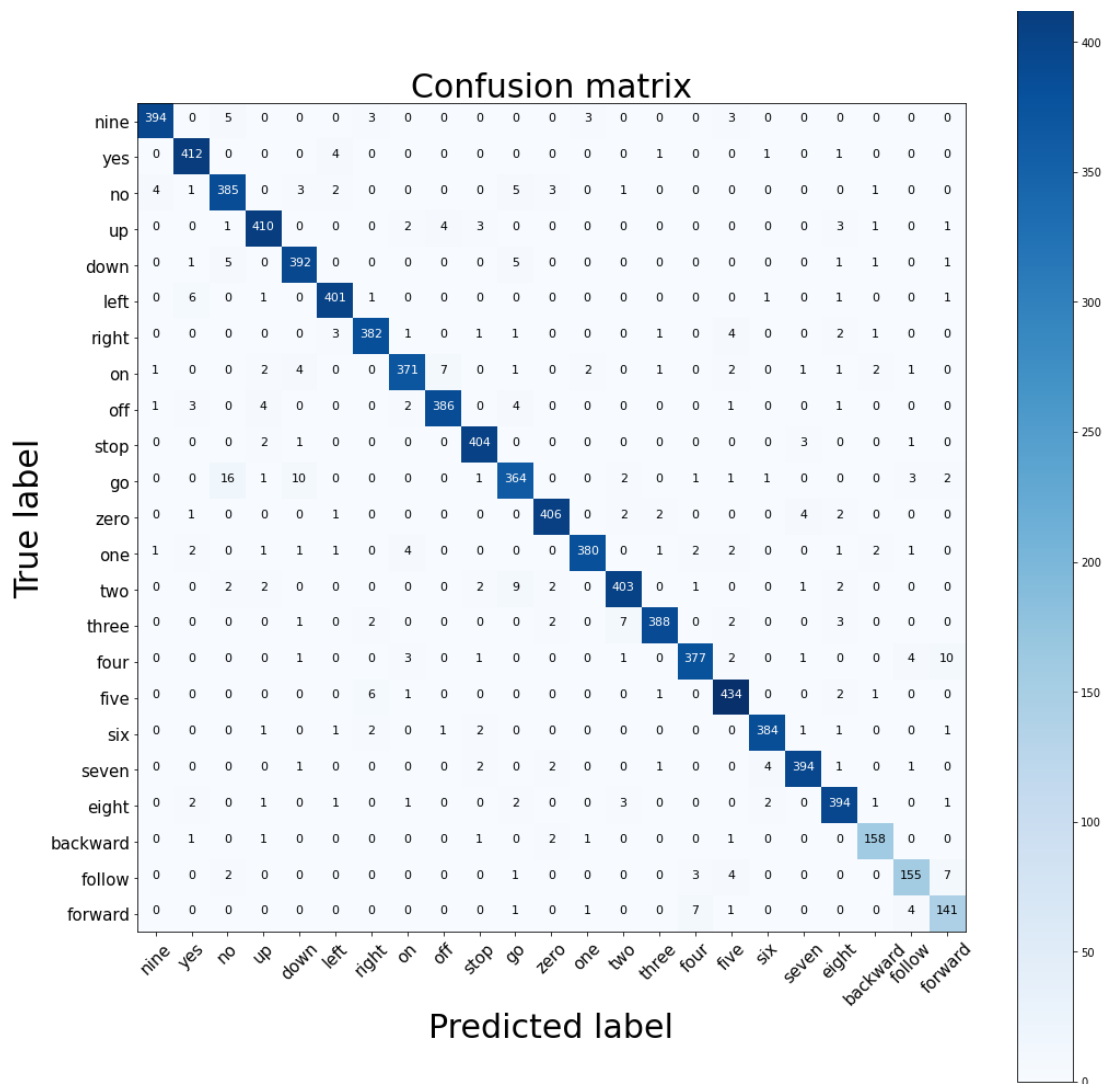


Fig 4.6: Confusion Matrix

Table 1: Confusion Matrix Metrics

Command	TPR	FNR	TNR	FPR
Nine	0.9656	0.0344	0.9991	0.0009
Yes	0.9832	0.0168	0.9936	0.0064
No	0.9506	0.0494	0.9962	0.0038
Up	0.9647	0.0353	0.9980	0.0020
Down	0.9655	0.0345	0.9973	0.0027
Left	0.9733	0.0267	0.9984	0.0016
Right	0.9646	0.0354	0.9983	0.0017
On	0.9368	0.0632	0.9983	0.0017
Off	0.9601	0.0399	0.9985	0.0015
Stop	0.9829	0.0174	0.9984	0.0016
Go	0.9054	0.0946	0.9964	0.0036
Zero	0.9712	0.0288	0.9986	0.0014
One	0.9523	0.0477	0.9993	0.0007
Two	0.9504	0.0496	0.9980	0.0020
Three	0.9580	0.0420	0.9990	0.0009
Four	0.9425	0.0575	0.9983	0.0017
Five	0.9752	0.0248	0.9972	0.0028
Six	0.9746	0.0254	0.9989	0.0011
Seven	0.9704	0.0296	0.9986	0.0014
Eight	0.9656	0.0344	0.9973	0.0027
Backward	0.9575	0.0425	0.9988	0.0012
Follow	0.9011	0.0989	0.9987	0.0013
Forward	0.9096	0.0904	0.9971	0.0029

5. TESTING

This chapter describes the various phases of our project which are tested.

5.1. Test Cases for RNN model with Attention

```
classes = ['nine', 'yes', 'no', 'up', 'down', 'left', 'right', 'on', 'off', 'stop', 'go',  
          'zero', 'one', 'two', 'three', 'four', 'five', 'six',  
          'seven', 'eight', 'backward', 'follow', 'forward']
```

Fig 5.1: Classes

Confusion Matrix is used to visualize how our model is doing over all classes. Confusion matrices are presented for the 23-word recognition task on Google Speech Command dataset V2. The diagonal entries in the confusion matrix correspond to correctly classified i.e., True Positives (TP).

5.2. Labelled Drone parts

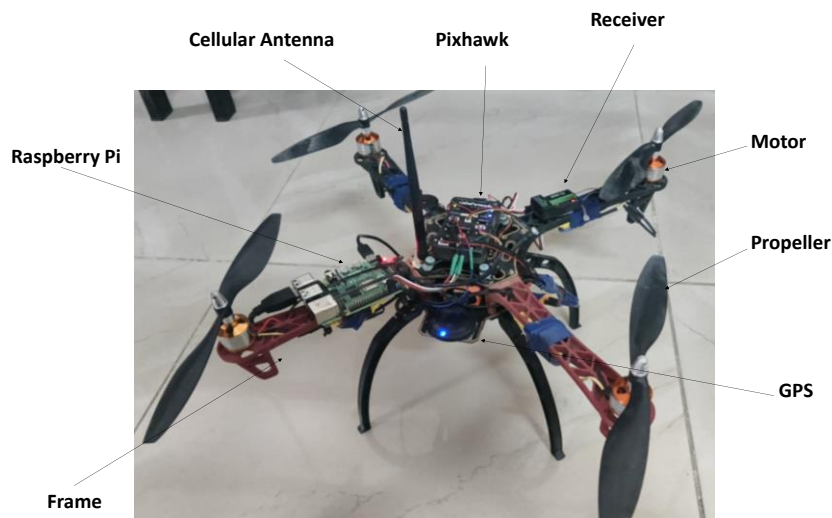


Fig 5.2: IOT Drone Front-View with labelled parts

The above picture shows the IOT drone with labels from Front-view. The picture shows the Frame, Motor, Propeller, Pixhawk, Cellular Antenna, Raspberry Pi, GPS, Receiver.

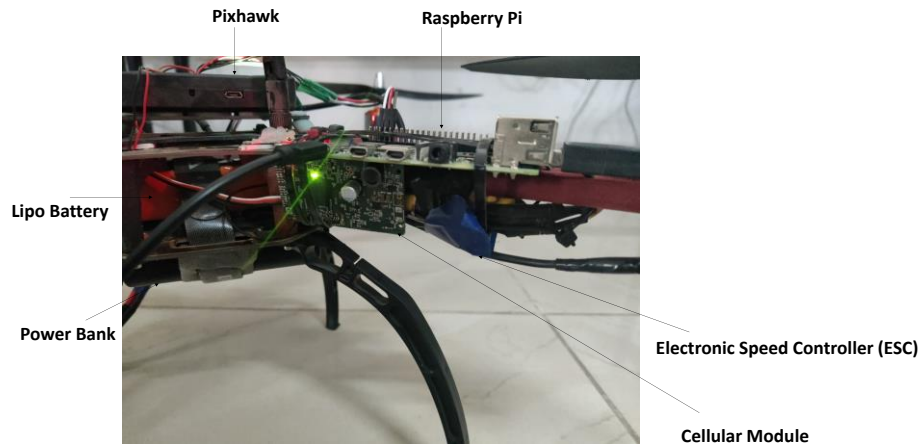


Fig 5.3: IOT Drone Left Side-View with labelled parts

The above picture shows the IOT drone with labels from Left Side-view. The picture shows the Pixhawk, Raspberry Pi, Lipo Battery, Power Bank, Electronic Speed Controller (ESC), Cellular Module.

5.3. Real Time Testing of IOT Drone



Fig 5.4: IOT Drone placed to start testing

The above picture shows the IOT Drone when placed on the surface/land to start the testing process.



Fig 5.5: IOT Drone taking off

The above picture shows the IOT Drone taking off when the command “Milind On” is said from the microphone of the laptop.



Fig 5.6: IOT Drone moving forward

The above picture shows the IOT Drone moving forward when the command “Milind Forward” is said from the microphone of the laptop.



Fig 5.7: IOT Drone flying

The above picture shows the IOT Drone flying when testing the drone.

6. CONCLUSION AND FUTURE SCOPE

Conclusion:

The Speech Recognition model for commanding is trained, validated, and tested. The accuracy of the model achieved is 95.5%.

Since, the model has achieved a good accuracy, it has been applied to real time application for controlling UAV/UGV device using speech/voice commands.

The trained model is then deployed into client-server program to recognize the command and then loaded into Raspberry Pi. The speech command is taken as an input from the microphone of laptop then the model predicts the corresponding voice command which is transferred to flight controller which controls the drone accordingly.

Future Scope:

Some of the future scopes can include:

- FPV (first-person view) camera
- Gesture controlled flying
- Live location tracking
- Return to user/controller can be used which makes the drone to return way back to the user/controller.
- Obstacle detection and avoidance
- Model can be deployed into Android Application through which user can control the UAV/UGV device
- Can be tested with UGV devices

REFERENCES

1. Das, Lyla B., et al. "Human Target Search and Detection using Autonomous UAV and Deep learning." *ieeexplore 2020*
<https://ieeexplore.ieee.org/abstract/document/9172031>
2. Narayan, S., et al. "A priority based exploration algorithm for path planning of an unmanned ground vehicle." *Ieeexplore 2014*
<https://ieeexplore.ieee.org/abstract/document/6953174>
3. Khan, Abdullah Ayub, Asif Ali Laghari, and Shafique Ahmed Awan. "Machine learning in computer vision: A review." *EAI Transactions on Scalable Information Systems eprints 2021*
<http://eprints.eudl.eu/id/eprint/5177/>
4. Bhatt, Chandradeep, et al. "The state of the art of deep learning models in medical science and their challenges." *springer 2021*
<https://link.springer.com/article/10.1007/s00530-020-00694-1>
5. <https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network>
6. Zhang, Liangpei, Lefei Zhang, and Bo Du. "Deep learning for remote sensing data: A technical tutorial on the state of the art." *IEEE 2016*
<https://ieeexplore.ieee.org/abstract/document/7486259>
7. <https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT>
8. <https://learn.sparkfun.com/tutorials/what-is-an-arduino/all>
9. sciencedirect.com/topics/medicine-and-dentistry/hidden-markov-model
10. <https://www.discoverdatascience.org/articles/what-is-python-used-for-why-is-it-important-to-learn/>
11. <https://www.zerotier.com/>
12. <https://data-flair.training/blogs/numpy-features/>
13. <https://pandas.pydata.org/docs/pandas.pdf>
14. <https://scikit-learn.org/stable/visualizations.html>
15. <https://scikit-learn.org/stable/>
16. <https://dawenl.github.io/publications/McFee15-librosa.pdf>
17. <https://britewire.com/keras/>
18. <https://en.wikipedia.org/wiki/TensorFlow>

19. Kukreja, Harsh, et al. "An introduction to artificial neural network." ReSearchGate 2016
20. Nekvinda, Tomáš. "Multilingual speech synthesis." 2020
<https://dspace.cuni.cz/handle/20.500.11956/119461>
21. <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>

APPENDIX

Code for DL Model

```
sr = samplingrate

iLen = inputLength

inputs = L.Input((inputLength,), name='input')

x = L.Reshape((1, -1))(inputs)

m = Melspectrogram(n_dft=1024, n_hop=128, input_shape=(1, iLen),
padding='same', sr=sr, n_mels=80,
fmin=40.0, fmax=sr / 2, power_melgram=1.0,
return_decibel_melgram=True, trainable_fb=False,
trainable_kernel=False,
name='mel_stft')

m.trainable = False

x = m(x)

x = Normalization2D(int_axis=0, name='mel_stft_norm')(x)

x = L.Permute((2, 1, 3))(x)

x = L.Conv2D(10, (5, 1), activation='relu', padding='same')(x)

x = L.BatchNormalization()(x)

x = L.Conv2D(1, (5, 1), activation='relu', padding='same')(x)

x = L.BatchNormalization()(x)

x = L.Lambda(lambda q: K.squeeze(q, -1), name='squeeze_last_dim')(x)

x = L.Bidirectional(rnn_func(64, return_sequences=True))(x)

x = L.Bidirectional(rnn_func(64, return_sequences=True))(x)

attVector = L.Dot(axes=[1, 1])([attScores, x]) # [b_s, vec_dim]
```

```

x = L.Dense(64, activation='relu')(attVector)

x = L.Dense(32)(x)

output = L.Dense(nCategories, activation='softmax', name='output')(x)

model = Model(inputs=[inputs], outputs=[output])

return model

```

Code in Raspberry Pi (Client-Server)

Server:

```

import socket

import struct

from dronekit import connect, VehicleMode, LocationGlobalRelative

from pymavlink import mavutil

import time

import argparse

parser = argparse.ArgumentParser()

parser.add_argument('--connect', default='127.0.0.1:14550')

args = parser.parse_args()

print('Connecting to vehicle on: %s' % args.connect)

vehicle = connect(args.connect, baud=57600, wait_ready=True)

import sys

import os

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

```

```

import SpeechModels

import tensorflow as tf

import librosa

import numpy as np


from tensorflow.keras.models import Model, load_model

from kapre.time_frequency import Melspectrogram, Spectrogram

from kapre.utils import Normalization2D


commands=["unknown/silence","nine","yes","no","up","down","left","right","on",
"off","stop","go","zero","one","two","three","four","five","six","seven","eight","b
ackward","bed","bird","cat","dog","follow","forward","happy","house","learn","m
arvin","sheila","tree","visual","wow"]


model = load_model('./model-attRNN-35words.h5',
custom_objects={'Melspectrogram': Melspectrogram, 'Normalization2D':
Normalization2D })

print("Model loaded")


def addr_to_bytes(addr):

    return socket.inet_aton(addr[0]) + struct.pack('H', addr[1])


def bytes_to_addr(addr):

    return (socket.inet_ntoa(addr[:4]), struct.unpack('H', addr[4:])[0])


def arm_and_takeoff(aTargetAltitude):

    print ("Basic pre-arm checks")

    while not vehicle.is_armable:

        print (" Waiting for vehicle to initialise...")

```



```

time.sleep(1)

print( "Arming motors")

vehicle.mode = VehicleMode("GUIDED")

vehicle.armed = True


while not vehicle.armed:

    print ( " Waiting for arming...")

    time.sleep(1)


print ("Taking off!")

vehicle.simple_takeoff(aTargetAltitude)


while True:

    print ( " Altitude: ", vehicle.location.global_relative_frame.alt )

    if vehicle.location.global_relative_frame.alt>=aTargetAltitude*0.95:

        print ("Reached target altitude")

        break

    time.sleep(1)


def send_body_ned_velocity(velocity_x, velocity_y, velocity_z, duration=0):

    msg = vehicle.message_factory.set_position_target_local_ned_encode(

        0,      # time_boot_ms (not used)

        0, 0,   # target system, target component

        mavutil.mavlink.MAV_FRAME_BODY_NED,

```

frame Needs to be MAV_FRAME_BODY_NED for forward/back left/right control.

0b0000111111000111, # type_mask

0, 0, 0, # x, y, z positions (not used)

velocity_x, velocity_y, velocity_z, # m/s

0, 0, 0, # x, y, z acceleration

0, 0)

for x in range(0,duration):

vehicle.send_mavlink(msg)

time.sleep(1)

val=1024*4

CHUNK = val

CHANNELS = 1

RATE = 16000

HOST = '0.0.0.0'

PORT = 40002

addr = ("", PORT)

soc = socket.socket()

try:

soc.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

soc.bind(addr)

soc.listen(5)

print("listening")

```

conn1, client_a = soc.accept()

print(conn1,client_a)

print("connected")

conn1.send(b"success")


if 1:

    print("Started drone:")

    while 1:

        fileName = "test audio.wav"

        print("Receiving file")

        bytes_read = conn1.recv(val)

        with open(fileName, "wb") as f:

            print("Inside while")

            i=0

            while True:

                print(i)

                if bytes_read[-4:]==bytes("stop",'utf-8'):

                    f.write(bytes_read[:-4])

                    break

                f.write(bytes_read)

            bytes_read = conn1.recv(val)

            i=i+1

            if i>100:

                raise Exception("Connection closed")

```

```

f.close()

print("audio received")

y, sr = librosa.load(fileName, sr=None)
X = np.empty((1, 16000))
curX=y
if curX.shape[0] == 16000:
    X[0] = curX
elif curX.shape[0] > 16000:
    randPos = np.random.randint(curX.shape[0]-16000)
    X[0] = curX[randPos:randPos+16000]
else:
    randPos = np.random.randint(16000-curX.shape[0])
    X[0, randPos:randPos + curX.shape[0]] = curX

print("Predicting model")
y_predict = model.predict(X[:1])

output=str(commands[np.argmax(y_predict,axis=1)[0]])
print("Command is : "+output)

if output=="on":
    arm_and_takeoff(10)
    print("Take off complete")
elif output=="off":
    print("Now let's land")

```

```

    vehicle.mode = VehicleMode("RTL")

elif output=="right":
    velocity_x = 0
    velocity_y = 5
    velocity_z = 0
    duration = 5

    send_body_ned_velocity(velocity_x, velocity_y, velocity_z, duration)

elif output=="left":
    velocity_x = 0
    velocity_y = -5
    velocity_z = 0
    duration = 5

    send_body_ned_velocity(velocity_x, velocity_y, velocity_z, duration)

elif output=="up":
    velocity_x = 0
    velocity_y = 0
    velocity_z = -1.5
    duration = 1

    send_body_ned_velocity(velocity_x, velocity_y, velocity_z, duration)

elif output=="down":
    velocity_x = 0
    velocity_y = 0
    velocity_z = 1.5
    duration = 1

    send_body_ned_velocity(velocity_x, velocity_y, velocity_z, duration)

elif output=="backward":
    velocity_x = -5

```

```

        velocity_y = 0
        velocity_z = 0
        duration = 5
        send_body_ned_velocity(velocity_x, velocity_y, velocity_z, duration)
    elif output=="forward":
        velocity_x = 5
        velocity_y = 0
        velocity_z = 0
        duration = 5
        send_body_ned_velocity(velocity_x, velocity_y, velocity_z, duration)
    elif output=="stop":
        print("Now let's land")
        vehicle.mode = VehicleMode("RTL")
        vehicle.close()
        conn1.send(b"stop")
        break

    conn1.send(b"received")

conn1.shutdown(socket.SHUT_RDWR)
conn1.close()
soc.close()
except Exception as e:
    print("Error:",e)
    conn1.shutdown(socket.SHUT_RDWR)
    conn1.close()
    print("Vehicle going home location due to error")

```

```
vehicle.mode = VehicleMode("RTL")

vehicle.close()

soc.close()
```

Client:

```
import socket

#from playsound import playsound

import pyaudio

from pocketsphinx import Decoder, get_model_path

import struct

import wave

import os

import signal, sys


def signal_handler(signal, frame):

    print("Program exiting ")

    soc.shutdown(socket.SHUT_RDWR)

    soc.close()

    print("stopped")

    sys.exit(0)


def addr_to_bytes(addr):

    return socket.inet_aton(addr[0]) + struct.pack('H', addr[1])


def bytes_to_addr(addr):

    return (socket.inet_ntoa(addr[:4]), struct.unpack('H', addr[4:])[0])
```

```
model_dir = get_model_path()
ps_config = Decoder.default_config()
ps_config.set_string('-logfn', 'nul')
ps_config.set_string('-hmm', os.path.join(model_dir, 'en-us'))
ps_config.set_string('-dict', 'hotword.dict')
ps_config.set_string('-keyphrase', 'milind')
ps_config.set_float('-kws_threshold', 1e-30)
decoder = Decoder(ps_config)
```

```
val=1024*4
FORMAT=pyaudio.paInt16
CHUNK=val
CHANNELS=1
RATE=16000
```

```
audio=pyaudio.PyAudio()
```

```
HOST="10.242.222.142"
PORT = 40002
```

```
soc = socket.socket()
try:
    soc.connect((HOST, PORT))
```



```

_=soc.recv(val)

print(HOST,PORT)

print("connected")


stream=audio.open(format=FORMAT,
                    channels=CHANNELS,
                    rate=RATE,
                    input= True,
                    frames_per_buffer=CHUNK)

decoder.start_utt()


stream1=audio.open(format=FORMAT,
                    channels=CHANNELS,
                    rate=RATE,
                    output= True,
                    frames_per_buffer=CHUNK)


print("Starting:")

while 1:

    data=stream.read(val)

    stream1.write(data)

    print("Waiting for activation command:")

    if data:

        decoder.process_raw(data, False, False)


    if decoder.hyp() is not None:

```

```
print([(seg.word, seg.prob, seg.start_frame, seg.end_frame) for seg in
decoder.seg()])
```

```
print("Detected %s" % (decoder.hyp().hypstr))
```

```
decoder.end_utt()
```

```
print("milind activated")
```

```
#playsound("sound.wav")
```

```
frames = []
```

```
data=stream.read(val)
```

```
stream1.write(data)
```

```
data=stream.read(val)
```

```
stream1.write(data)
```

```
print("Sending audio")
```

```
for i in range(0, int(RATE / CHUNK * 1.5)):
```

```
    print("In for loop",i)
```

```
    data=stream.read(val)
```

```
    stream1.write(data)
```

```
    frames.append(data)
```

```
wf = wave.open("test audio.wav", 'wb')
```

```
wf.setnchannels(CHANNELS)
```

```
wf.setsampwidth(2)
```

```
wf.setframerate(RATE)
```

```
wf.writeframes(b''.join(frames))
```

```
wf.close()
```

```
filename="./test audio.wav"
```

```
with open(filename, "rb") as f:
```

```
    while True:
```

```
        bytes_read = f.read(val)
```

```
        if not bytes_read:
```

```
            break
```

```
        soc.sendall(bytes_read)
```

```
soc.sendall(b"stop")
```

```
print("audio sent")
```

```
decoder.start_utt()
```

```
mess=soc.recv(val)
```

```
print("Command is:"+mess.decode('utf-8'))
```

```
if mess==b"stop":
```

```
    break
```

```
signal.signal(signal.SIGINT, signal_handler)
```

```
soc.shutdown(socket.SHUT_RDWR)
```

```
soc.close()
```

```
except KeyboardInterrupt:
```

```
    print("keyboard error")
```

```
except Exception as e:
```

```
    print("Error: ",e)
```

```
    soc.shutdown(socket.SHUT_RDWR)
```

```
    soc.close()
```

```
    print("stopped")
```