Palestine Polytechnic University

College of Information Technology and Computer Engineering

Department of Computer System Engineering

# Speech commands recognition applied on a mobile robot (SCRAMR)

Team Members

**Ahmad Ali Horyzat**
**Mohammad Menwer Saleh**
**Mohammad Khalil Jbour**

Supervisor

**Eng. Wael Takrouri**

**May 2022**

## Acknowledgement

In the name of "Allah", the most beneficent and merciful who gave us strength, knowledge, and help to get through this project.

For those who deserve our thanks the most, our parents, we are indebted to you for the rest of our lives  for your unconditional love and support. We know that thank you is not enough and there are not enough words to describe how thankful we are.

To our families and friends, thank you for your endless encouragement all our lives, especially during this project's completion.

We would like to thank our supervisor for this project, Eng. Wael Takrouri for his valuable help and advice during this project.

We also thank our faculty and doctors at the college of information technology and computer engineering for their hard work and support.

## Abstract

Since the industrial revolution, people have been looking forward to technologies that will facilitate their lives, especially handicapped people who are willing to live normal lives.

In this project, we will focus on providing a real-time, offline speech recognition system that will help handicapped communicate with a mobile robot easily by voice. In essence, we propose an embedded system of a mobile robot controlled using speech recognition.

Our system utilizes deep learning algorithms for real-time speech commands detection in order to achieve a more natural approach of robotic systems control, which will help minimize the distance between the digital world and the physical world.

In this project, we designed, implemented, and trained an intelligent small-scale speech-controlled car. The system was able to achieve great dynamic speech commands detection as it was tested with different people's voices.

We've tested our system in different environments with variant noise rates and different obstacles arrangements and, as expected, it worked greatly. It returned very good results in minimal response time.

**Keywords:** Speech recognition, Deep Learning, Mobile robot, ESP-NOW.

## الخلاصة

منذ الثورة الصناعية، كان الناس يتطلعون إلى التقنيات التي من شأنها تسهيل حياتهم ، وخاصة الأشخاص المعاقين الذين يرغبون في عيش حياة طبيعية. في هذا المشروع، سنركز على توفير نظام التعرف على الصوت في الوقت الفعلي وغير المتصل بالإنترنت والذي يساعد ذوي الاحتياجات الخاصة على التواصل مع روبوت متنقل بسهولة. في الأساس، نقترح نظامًا مدمجًا لروبوت يمكن التحكم به باستخدام الأوامر الصوتية من المستخدم مباشرة.

يستخدم نظامنا خوارزميات Deep Learning لاكتشاف الأوامر الصوتية في الوقت الفعلي من أجل تحقيق نهج أكثر طبيعية للتحكم في الأنظمة الروبوتية، مما يساعد في تقليل المسافة بين العالم الرقمي والعالم المادي.

في هذا المشروع، قمنا بتصميم وتنفيذ وتدريب سيارة ذكية صغيرة الحجم يتم التحكم فيها بواسطة الأوامر الصوتية. تمكن النظام من تحقيق اكتشاف أوامر صوتية ديناميكية جيدة حيث تم اختباره بأصوات أشخاص مختلفين.

لقد اختبرنا نظامنا في بيئات مختلفة بمعدلات ضوضاء متغيرة وترتيبات عوائق مختلفة فأظهر نتائج جيدة بشكل عام, كما هو متوقع. أظهر النظام نتائج جيدة باستجابة في الوقت الفعلي.

**الكلمات المفتاحية:** التعرف على الكلام، التعلم العميق، نظام آلي.

## Table of contents

## List of Figures

## List of Tables

## List of Acronyms

| | |
|---|---|
| CNN: Convolutional Neural Networks | 1 |
| RNN: Recurrent Neural Networks | 9 |
| MFCCs: Mel Frequency Cepstral Coefficients | 9 |
| RTOS: Real Time Operating System | 10 |
| TCMC: Taiwan Semiconductor Manufacturing Company | 15 |
| AGC: Automatic Gain Control | 18 |
| VGA: Variable Gain Amplifier | 18 |
| ReLU: Rectified Linear Unit | 28 |

# Chapter 1: Introduction

## 1.1 Overview

Using voice is the most important way of communication between humans; we aim in this project to make this kind of communication between humans and machines more available.

In this project, we aim to make a robot that's able to understand some spoken words received from humans and respond to them in real-time without requiring an internet connection or using external speech-to-text services. This will be achieved by equipping the robot with the necessary resources that make it able to receive and recognize the voice signals.

We will train a Convolutional Neural Networks (CNN) model to recognize spoken commands in real-time. Then we will deploy the model on a microcontroller that reads voice signals from a microphone and when a command is detected, it will respond to it with the required actions according to the received command.

As a demonstration of the concept of the power of speech recognition in controlling certain types of systems, in this project, we will control the movement of our robot by speech with different tones of voices of different people.

## 1.2 Motivation

The usage of speech recognition adds speed and ease to human-machine interactions, the capability of receiving and processing the human speech is indeed a very big step ahead in the world of technology.

Integrating a system with speech recognition capability widens its scope of functionality and increases productivity such as in healthcare industries. It can also capture speech much faster than typing using a keyboard. The main motivation we had in our sight is to help the handicapped people; people who lost a hand or a leg or maybe both find it difficult to use machines and move around. Now by using only their voices they will be able to control the movement of the robot in real-time.

Although speech recognition is not an easy task, when a system is well-trained on different speech commands with different tones, it could be used to control multiple systems, not only hardware robotic systems, but also software systems, like opening an email or sending a text message.

The usage of speech recognition robotics can enhance the lives of all types of users, especially those with physical disabilities since they can’t practice their lives normally. A robot easily controlled by speech can be such a help in doing the daily stuff like cleaning or moving things around.

## 1.3 System Objectives

In this project we propose an embedded system that can provide the following;

- Design a system based on neural networks for real-time speech recognition and control of a robotic system using previously specified words.
- Propose an efficient approach to control systems, which will provide a reliable alternative to traditional approaches.
- Conduct experiments to provide accuracy reports of the system.

## 1.4 General System Elements

Our system will detect voice using a regular microphone that will be near the user, which will continuously be listening to the user and analyze the voice and try to detect one of the already specified speech commands. Once a valid speech command is detected, it will be sent to a microcontroller installed on the robot, and the robot would move accordingly, in real-time without the need for an internet connection.

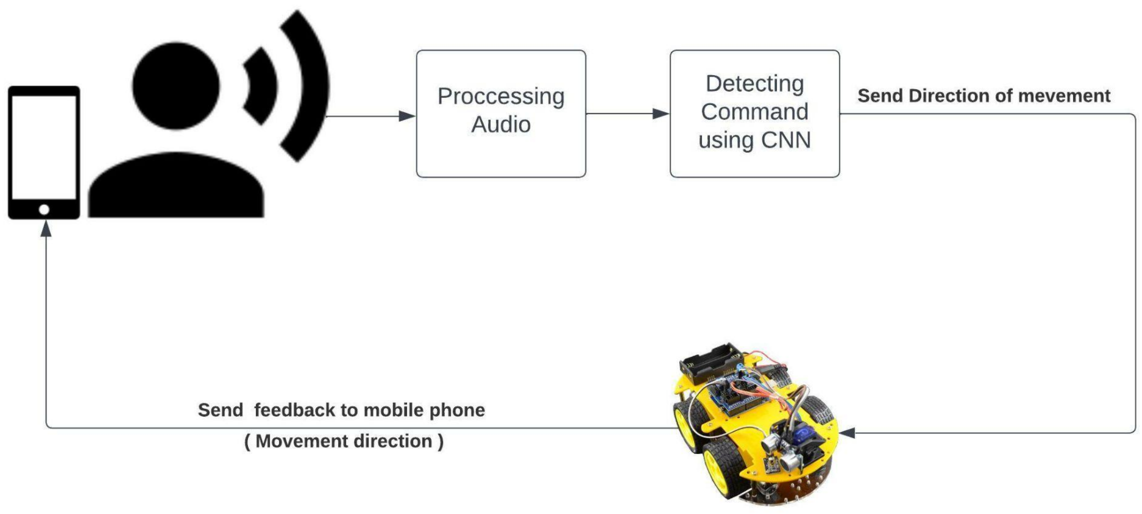

Proccessing Audio
Detecting Command using CNN
Send Direction of mevement
Send feedback to mobile phone
( Movement direction )

Figure 1.1: Block diagram of the system

## 1.5 Problem Statement

The challenging problem is to make the system recognize speech instructions given by users with different tone voices and respond to these commands with a suitable response in real-time and without relying on any external speech recognition platforms.

## 1.6 List of Requirements

### 1.6.1 Functional Requirements

1. Detect human speech and recognize commands in real-time.
2. Move the robot based on the detected speech command.
3. All the work should be done locally without requiring any kind of external connection.
4. Train the CNN model ( feature classification and extraction ).

### 1.6.2 Non-Functional Requirements

1. Accurate deep learning model.
2. Real-time command recognition and reliable responsivity.
3. Light system size.

## 1.7 Expected Results

1. A Fully functioning model that can detect human speech and recognize commands to control a robot in real-time.
2. Decrease the distance between the digital world and the physical world, by using human speech to control electronic devices, and using such systems to reach places that are unreachable to humans with disabilities without relying on other humans.

## 1.8 Overview of the rest of the report

Next chapter, “background”, contains the theoretical background and Literature review. Design options for hardware components, software, and design constraints will be introduced too. Chapter three, includes a detailed conceptual description of the system, detailed design, schematic diagrams, block diagrams, and structural diagrams. Chapter four, Implementation. Chapter five, testing and validation. Chapter six, conclusion and future work.

# Chapter 2: Background

## 2.1 Overview

In this chapter, we will discuss the background of the project's components. First, we will talk about the theoretical background, explaining the general terms like machine learning. Then, under literature review, we will mention and review the previous studies and papers related to the topic of our project and how our project differentiates from them. Next, we will look over the main hardware devices and tools we used and why we chose to use them along with other choices we could go with. Finally, we will talk about the software components, generally about the programming languages and the algorithms we used.

## 2.2 Theoretical background

Communication technology continues to evolve rapidly. New ways of communicating with systems and computers have emerged and one of the important communication techniques is speech.

Speech recognition is a term used in systems that can receive, analyze and understand voice data. Nowadays, every smartphone contains a speech assistant of some sort that can quickly take down notes for you or perform tasks such as opening applications and sending emails [4]. This has been possible in the recent past due to the high processing power of computers we now have.

Speech recognition is used in any possible environment and enters many industrial fields, it can be used to control big or small systems starting from controlling a smart mobile to controlling huge machines like rocket launchers.

### 2.2.1 Machine learning

Is a branch of Artificial Intelligence concerned with building applications that learn from the data and improve its accuracy over time without needing programmers to do that. In traditional programming, the algorithm is a sequence of statistical processing steps, but in machine learning, the algorithm is trained from the data features and patterns to make a decision [19].

There are three main types of machine learning, Supervised, Unsupervised, and Semi-supervised machine learning.

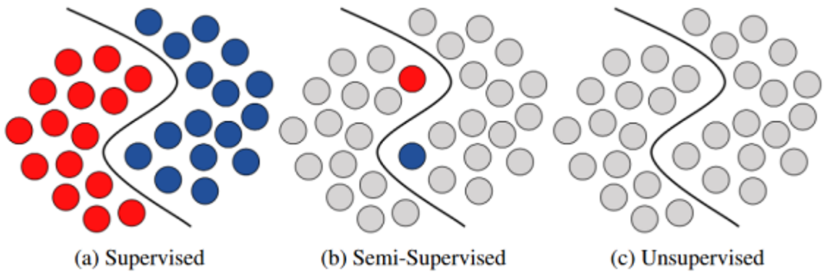


Figure 2.1: Main types of machine learning [19].

### 2.2.2 Deep learning

Is a subset of machine learning that enables the machine to learn without human supervision. Neural networks are the basic building block of deep learning. It contains nodes that work together to mimic the human brain [14].

In terms of architecture, a neural network consists of an input layer, an output layer, and hidden layers. Each layer consists of one or more connected nodes, each node called a neuron or a perceptron. Each neuron has a weight and an activation function. If the output of the neuron is above a specific threshold the neuron fires the output data to the input of the next neuron. Otherwise, no data passes to the next neuron. The deep learning models need a huge amount of data passed through their layers to apply tuning for the weights and biases for each layer to get the best outcome.

The Convolutional Neural Networks (CNNs) is a great implementation of that, and it is used primarily for computer vision applications, which can detect patterns and features and recognize objects from images or videos [15].

## 2.3 Audio Signal processing

### 2.3.1 Audio Signals

When an object vibrates, the air molecules oscillate to and from their resting position and transmit their energy to neighboring molecules. This results in the transmission of energy from one molecule to another which in turn produces a sound wave [3].

Parameters of an audio signal:

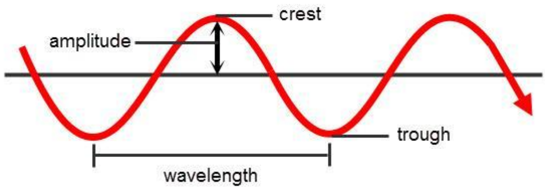


Figure 2.2: Analog audio signal [3].

### 2.3.2 Audio Sampling

Sampling the signal is a process of converting an analog signal to a digital signal by selecting certain number of samples per second from the analog signal so that it can be stored and processed efficiently in memory [3]. Discretizing audio signal is illustrated in Figure 2.3:

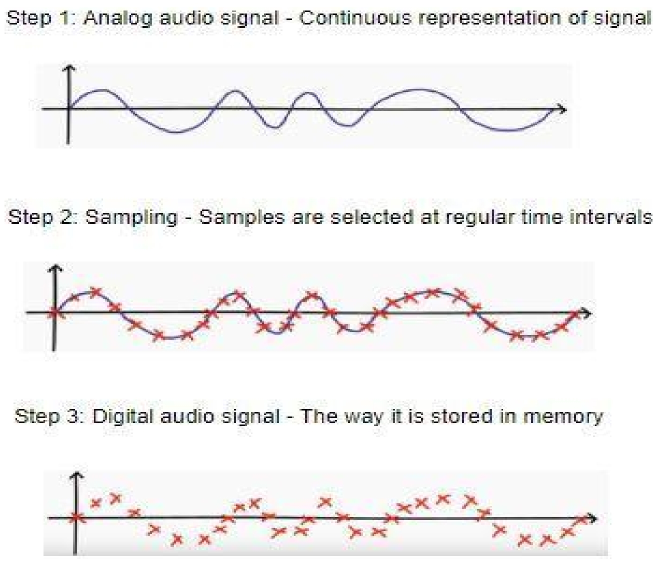

Step 1: Analog audio signal - Continuous representation of signal
Step 2: Sampling - Samples are selected at regular time intervals
Step 3: Digital audio signal - The way it is stored in memory

Figure 2.3: Audio sampling steps [3].

The sampling rate or sampling frequency is defined as the number of samples selected per second.

## 2.4 Voice Recognition

Voice-enabled devices use the principle of speech recognition. It is the process of electronically converting a speech waveform (as the realization of a linguistic expression) into words (as a best-decoded sequence of linguistic units) [31].

### 2.4.1 Neural network-based speech recognition

The most common approach in speech recognition modeling is the use of neural networks. They are capable of solving complicated recognition tasks. Rather than being used in general-purpose speech recognition applications, they can handle low quality, noisy data and speaker independence. Such systems can achieve great accuracy, as long as there is training data and the vocabulary is limited. A more general approach using neural networks is phoneme recognition.

Types of neural network speech recognition:

**Recurrent neural network (RNN)**

A recurrent neural network (RNN) is an artificial neural network that uses sequential data or time-series data. recurrent neural networks utilize training data to learn. It’s commonly used for ordinal and temporal problems, such as image captioning, language translation, natural language processing, and speech recognition [16].

**Convolutional Neural Network (CNN)**

A typical CNN architecture is formed of several convolutional and pooling layers with fully connected layers for classification. A convolutional layer is composed of kernels that are convolved with the input. A convolutional kernel divides the input signal into smaller parts namely the receptive field of the kernel. Furthermore, the convolution operation is performed by multiplying the kernel with the corresponding parts of the input that are in the receptive field.

Our speech recognition model will be neural networks based recognition using CNN since our commands are a single word long that can easily fit into a 1-second-long audio clip that can be converted into MFCC Spectrogram, which is an image. In the end, the model will classify images (spectrograms) for their specified commands.

## 2.5 System Software

### 2.5.1 Programming Languages

System functionality will be divided into two phases, the first phase is voice handling, processing, and storing. The second phase is command detection, predicting the Deep Learning model.

There's no doubt that Python is the best language to use for AI and ML and therefore deep learning and speech recognition, so we will be using Python for the first phase which is the speech recognition phase. After we handle the coding of the first phase, we will no longer have to use Python for the rest of the coding, so we will use C++ which is a much faster programming language that will guarantee a real-time responsive system and it works very well with RTOS.

### 2.5.2 TensorFlow

TensorFlow is an open-source platform that was developed by Google's Brain team, that contains a comprehensive, flexible ecosystem of tools, libraries, and community resources [12]. It is widely used in training machine learning models.

### 2.5.3 Real-time Operating System (FreeRTOS)

An RTOS is a lightweight operating system (OS) designed to run on microcontrollers. Much like general-purpose operating systems, they offer a scheduler to run multiple threads or tasks, resource management (such as file I/O), and device drivers.

FreeRTOS is designed to be small and simple. It is mostly written in the C programming language to make it easy to port and maintain. It also comprises a few assembly language functions where needed, mostly in architecture-specific scheduler routines. FreeRTOS provides methods for multiple threads or tasks, mutexes, semaphores, and software timers [13].

The task scheduling of RTOS is shown in figure 2.4.

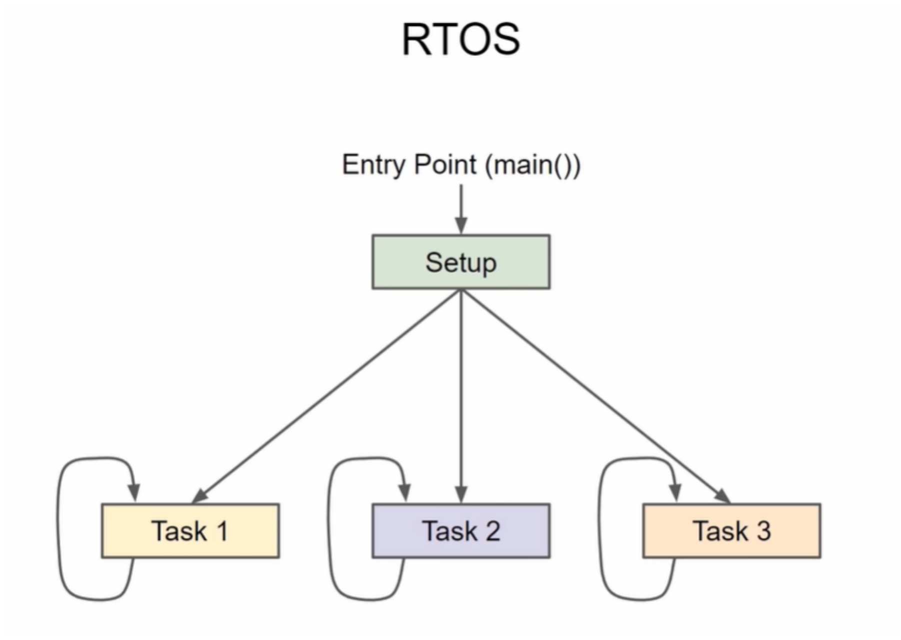


Figure 2.4: Task scheduling of RTOS [13].

FreeRTOS comes in quite handy when using a multi-core processor such as in our ESP32 which includes a dual-core processor. Using FreeRTOS will enable us to exploit the ESP32 and get the most out of the processor by concurrently using both cores and getting more tasks done in a specific period of time.

## 2.6 Literature review

Many studies have examined speech recognition. Looking at those studies with a critical eye helped us choose the techniques that suit the needs of our project. There are many techniques to be used in recognizing speech and analyzing it, two similar projects may vary in a lot of details like if one is using Arduino and the other is using Raspberry Pi for example. So, it’s the same concept for all the projects but the small details make a difference.

These are some of the studies and projects that are similar to our project. We will discuss the basics of each one of them and differentiate them from our work.

1. **"Voice Controlled Robot" by Nouman Salameen and Anas Al-Shweiki, 2019. (Salameen and Al-Shweiki 2019)**

Using BitVoicer, which is a ready-to-use software for recognizing voice in many languages including English, the designers managed to come up with a reliable system for controlling robotics with voice. The voice is recorded using a USB microphone, the voice data is received by BitVoicer which analyzes and translates it to written sentences and words, then the words are sent to the microcontroller which moves the robot depending on the command received.

Although we will use similar hardware, the software is quite different since we're building it from scratch without using any ready-to-use software like BitVoicer which needs a TCP/IP connection to do the processing, but in the current work, the system doesn't need any TCP/IP connection and can function offline.

2. **"Voice Controlled Robot" by Pratik Chopra and Harshad Dange, 2007. (Chopra and Dange 2007)**

This project proposes a design for a Machine Learning model that is trained on some specific words. The system is built using HM2007 which is a voice recognition chip with an on-chip analog front end, voice analysis, recognition process, and system control functions. The input voice command is analyzed, processed, recognized, and then obtained at one of its output ports which are then decoded, amplified, and given to the motors of the robot car.

Chopra's model is using a specially dedicated chip to process the speech and analyze the commands connected to a controller. This differentiates from our scheme since we're using a microprocessor that offers high processing capability, therefore, it's faster and more reliable.

3. **"Arduino Based Voice Controlled Robot Vehicle" by M Saravanan, 2020. (Saravanan 2020)**

This project was developed in a way that the robot is controlled by voice commands. An android application with a microcontroller is used for required tasks. The connection between the android app and the vehicle is facilitated with Bluetooth technology. The robot is controlled by buttons on the application or by spoken commands of the user [29].

The designer of this model used a mobile phone to identify the user's commands which uses Google's voice recognition service that allows the application deployed on a smartphone to understand everything the user says. The only back down in this technique is that the mobile app needs to be connected to the internet to analyze the data received and if the availability of internet connection is not an issue, still, a fast and reliable connection must be used or the system will suffer an annoying delay that would stop it from being a real-time system.

## 2.7 Hardware components

### 2.7.1 Microcomputer

a. **Raspberry Pi 4 Model B**: it is the latest product in the popular Raspberry Pi range of computers. It offers ground-breaking processor speed, multimedia performance, memory, and connectivity [26].

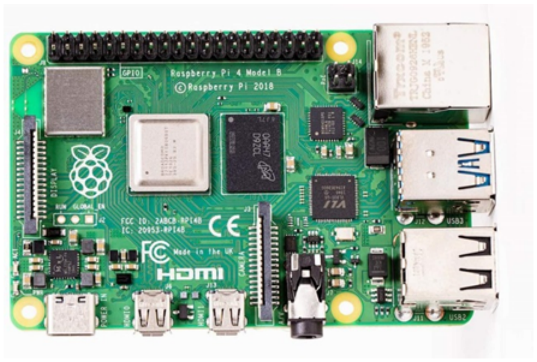


Figure 2.5: Raspberry Pi 4 [26].

b. **Jetson Nano Developer Kit**: This is a small, powerful computer that lets you run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing [23].

Figure 2.6: Jetson nano [23].

c. Keyes ESP32 Core board: a mini development board designed based on the ESP-WROOM-32 module. This development board leads out 2.54mm pitch headers from I/O to both sides. The users can connect peripheral devices to their needs. When using and debugging, the standard headers on both sides can make your operation more concise and convenient [10].

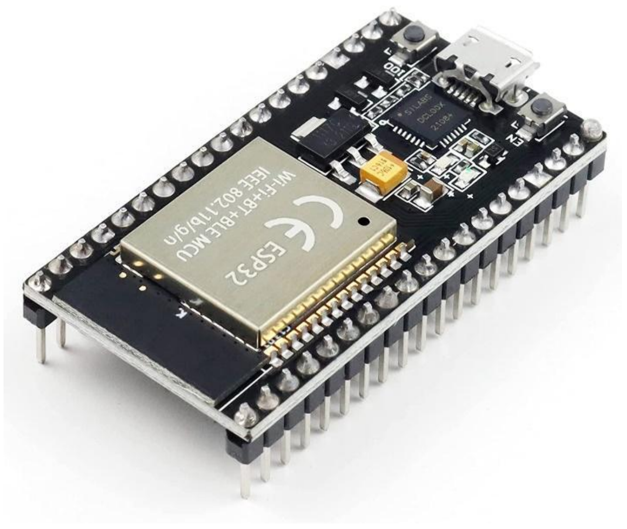


Figure 2.7: ESP32 [10].

The ESP-WROOM-32 module is the leading Wi-Fi + Bluetooth solution with less than 10 external components. It integrates antenna switches, RF balun, power amplifiers, low noise amplifiers, filters, and power management modules. At the same time, it also adopts TSMC low-power 40nm technology. The cost-effectiveness and radio frequency performance make it safe and reliable to expand various applications [10].

**ESP-NOW**

One of the main features of ESP32 is ESP-NOW, a protocol developed by Espressif which enables multiple devices to communicate with one another without using Wi-Fi. The protocol is similar to the low-power 2.4GHz wireless connectivity that is often deployed in wireless mice. So, the pairing between devices is needed before their communication. After the pairing is done, the connection is secure and peer-to-peer, with no handshake being required [11].

The following table compares the hardware specifications of Raspberry Pi 4, NVIDIA Jetson Nano Developer Kit, and Keyes ESP32:

| **Category** | **Raspberry Pi 4** | **NVIDIA Jetson Nano** | **Keyes ESP32** |
|---|---|---|---|
| **CPU** | Quad-core ARM Cortex-A72 65-bit @ 1.5 Ghz | Quad-core ARM Cortex-A57 65-bit @ 1.42 Ghz | Dual-core Xtensa LX6 32-bit @ 600Mhz |
| **GPU** | Broadcom VideoCore VI (32-bit) | NVIDIA Maxwell w/ 128 CUDA cores @ 921 Mhz | - |
| **Memory** | 4 GB LPDDR4 | 4 GB LPDDR4 | 984 KB |
| **Networking** | Gigabit Ethernet / Wi-Fi 802.11.ac | Gigabit Ethernet / Wi-Fi 802.11.ac | Bluetooth v4.2, Wi-Fi 802.11bgn |
| **Display** | 2x micro-HDMI (up to 4Kp60) | HDMI 2.0 and eDP 1.4 | - |
| **USB** | 2x USB 3.0, 2x USB 2.0 | 4x USB 3.0, 2x Micro-B | - |

| | | | |
|---|---|---|---|
| **Other** | 40-pin GPIO | 40-pin GPIO | 38-pin GPIO |
| **Storage** | Micro SD | Micro SD | Micro SD |
| **Price** | 55$ | 99$ | 15$ |

Table 2.1: Differences between proposed microcomputers [11][7].

The specs of Raspberry Pi 4 and NVIDIA Jetson Nano are quite close, they both offer high memory and processing capabilities along with other high-end specs. However, we will use the Keyes ESP32, although its hardware specs are lower than the other options it includes a more reliable connection through the ESP-NOW protocol which enables us to use two ESPs with strong connectivity rather than a weak connectivity wireless mic, it also offers the needed processing power and connectivity for our project at a cheaper price.

### 2.7.2 Motor Driver

It is an IC that is used to control the DC Motors, it has the capability of controlling more than one DC Motor at the same time and it enables the motor to move in any direction. There are many types of Motor Drivers and these are the best choices we have found:

- **L293D**

It's the most popular option to be used with microcontrollers when high current output is not required [22].

Figure 2.8: L293D [22].

- **L298N**

It's used when DC Motors require high peak output and high current [22].

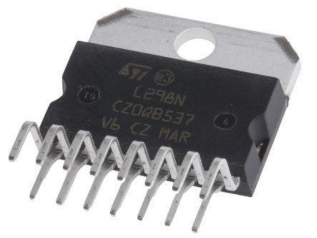


Figure 2.9: L298N [22].

The following table displays the differences between the two drivers:

| **Category** | **L293D** | **L298n** |
|---|---|---|
| **Max supply voltage** | 36V | 40V |
| **Peak output current** | 1.2A | 2.5A |
| **Continues output current** | 0.6A | 2A |
| **Power dissipation** | 5W | 25W |
| **Price** | ~1$ | ~2$ |

Table 2.2: Comparison between L293D and L298N [22].

We will use the L293D because it's sufficient and no need for the higher specification driver, also it's more available and can be easily found in any nearby tech store.

### 2.7.3 DC Motors

It is a device that converts direct current electrical energy into mechanical energy and uses the electricity generated to produce motion for wheels.

### 2.7.4 MAX9814 Microphone

The MAX9814 is a low-cost, high-quality microphone amplifier with automatic gain control (AGC) and low-noise microphone bias. The device features a low-noise preamplifier, variable gain amplifier (VGA), output amplifier, microphone-bias-voltage generator, and AGC control circuitry.

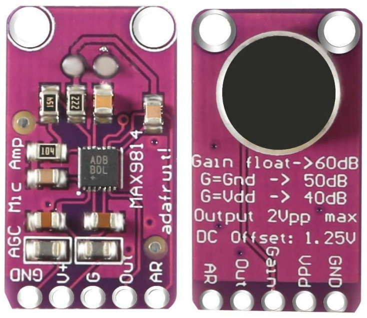


Figure 2.10: MAX9814 [6].

**Features**

- Automatic Gain Control (AGC)
- Three Gain Settings (40dB, 50dB, 60dB)
- 2.7V to 5.5V Supply Voltage Range
- Low-Power
- Internal Low-Noise Microphone Bias, 2V
- -40°C to +85°C Extended Temperature Range

### 2.7.5 Obstacle Avoidance Sensor

As the robot is moving around, it needs to have some vision of the environment to avoid collisions with other objects.

We will equip our mobile robot with a pair of obstacle detection sensors to detect obstacles and stop moving instead of just running into them.

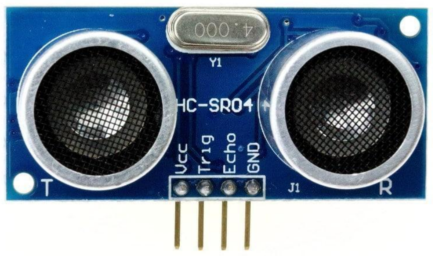


Figure 2.11: Ultrasonic sensor [27].

The table below compares two of the most popular obstacle detectors:

| **Features** | **Ultrasonic sensor** | **Infrared sensor** |
|---|---|---|
| **Detection range** | • Suitable to detect objects which are more than 1 meter away.<br>• capable of placing objects within 5 mm more accurately.<br>• Some ultrasonic sensors detect objects with max. range of 20 meters. | • More appropriate for targets that are closer than 10 mm |
| **Interference from light sources** | • Unaffected | • affected |
| **Frequency range** | • Ultrasonic/ultrasound devices operate from 20 kHz up to several GHz. | • 430 THz down to 300 GHz |
| **Wavelength range** | • ultrasonic waves use wavelengths of about 1.9 cm or less. | • 700 nm to 1 mm |

Table 2.3: Comparison between two obstacle detectors [27].

We will use the Ultrasonic sensor because it’s not affected by light and it’s cheaper.

# Chapter 3: Design

## 3.1 Overview

In this chapter, we will discuss the overall design of the speech-controlled robotic embedded system and the way its components are integrated, showing the block diagram and schematic diagram for the design, in addition to some details about the speech recognition software.

## 3.2 System Design

This design scheme includes two parts connected via ESP-NOW. The first part contains ESP32 we will call it ”user ESP32” and a microphone, this part will be near the user. It will be responsible for reading voice signals, extracting features from voice, detecting commands, and sending these commands to the second part. The second part consists of another ESP32 which will be called “Robot’s ESP32”, an L293D motor driver, and two motors as shown in the figure below, this part will receive the detected command and move the robot if there are no obstacles in the way. This part will be installed on the Robot to control its movement. Figure 13 illustrates the system block diagram.

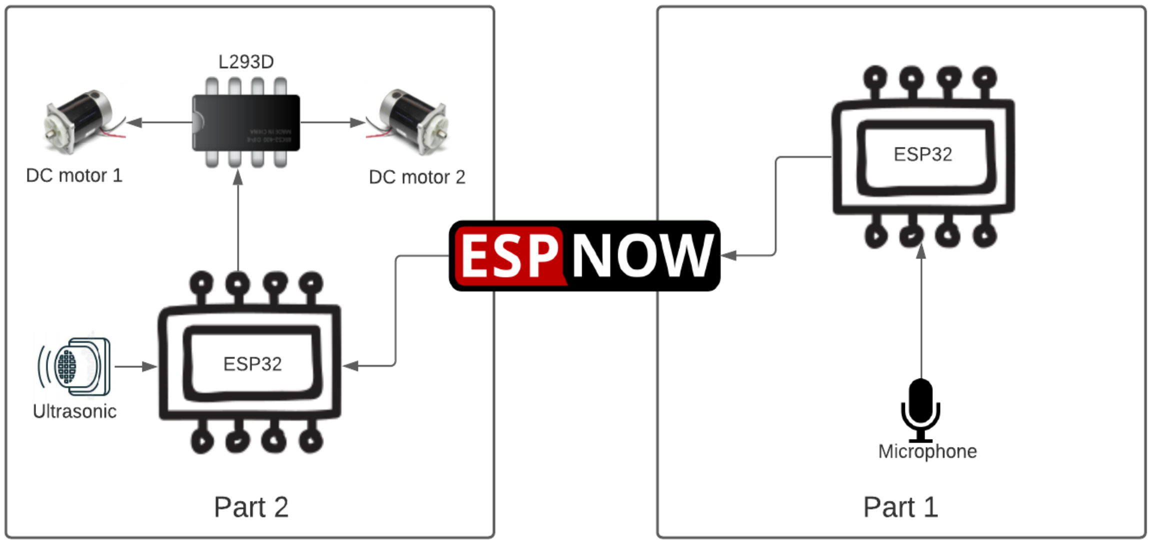


Figure 3.1: Diagram of the system design

## 3.3 Block Diagram

The block diagram in figure 3.2 shows the steps to recognize speech commands, and shows how the components of the system communicate with each other. The two ESP32 communicate via ESP-NOW which allows high-speed data exchange from long distances. The Robot ESP32 communicates with a mobile app via Bluetooth.

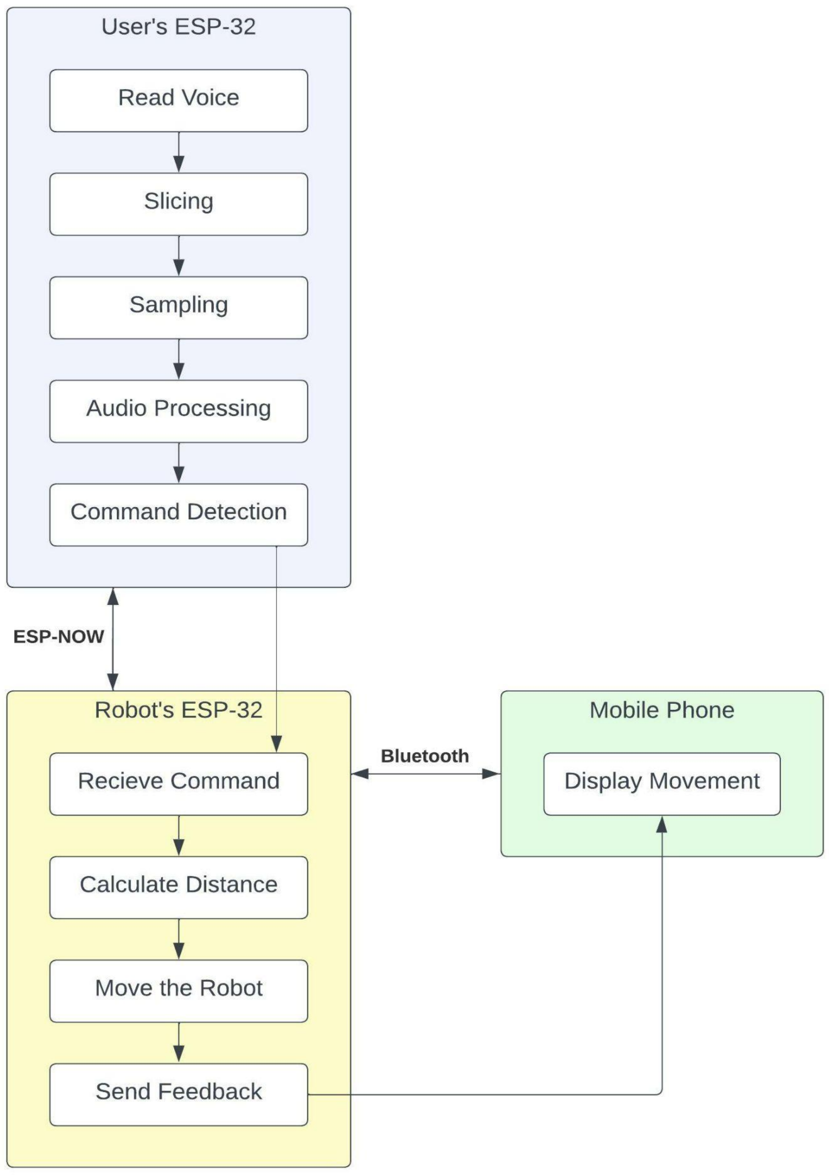


Figure 3.2: Block Diagram

## 3.4 Pseudo-Code

The pseudocode for ESP32 at user side is shown in Figure 3.3, it shows how it detects commands and the way it communicates with the other part of the system.

```
**User ESP32**

Initialize network connection
Connect to robot ESP32 via ESP-NOW
While the program is running do
        Process audio for prediction
        Detect command
        Send command to robot ESP32
End while
```

Figure 3.3: Pseudocode of the ESP32 at user side.

The second pseudo-code in figure 3.4 shows how the ESP32 at the user side connects to the robot ESP32, and with a mobile app via Bluetooth to send the live movement of the robot. Also, it describes how to move the robot based on the received command from user ESP32.

```
** Robot ESP32 **

Initialize network connection
Connect to user ESP32 via ESP-NOW
Connect to mobile app via Bluetooth
While the program is running do
        Receive command from user ESP32
        Move the robot depending on the received command
        Send feedback to mobile app
End while
```

Figure 3.4: Pseudo Code for Robot’s ESP32

## 3.5 Schematic Diagram

The schematic diagram of the system is shown in Figure 3.5, it contains a microphone connected to ESP32 and a 5V power supply.

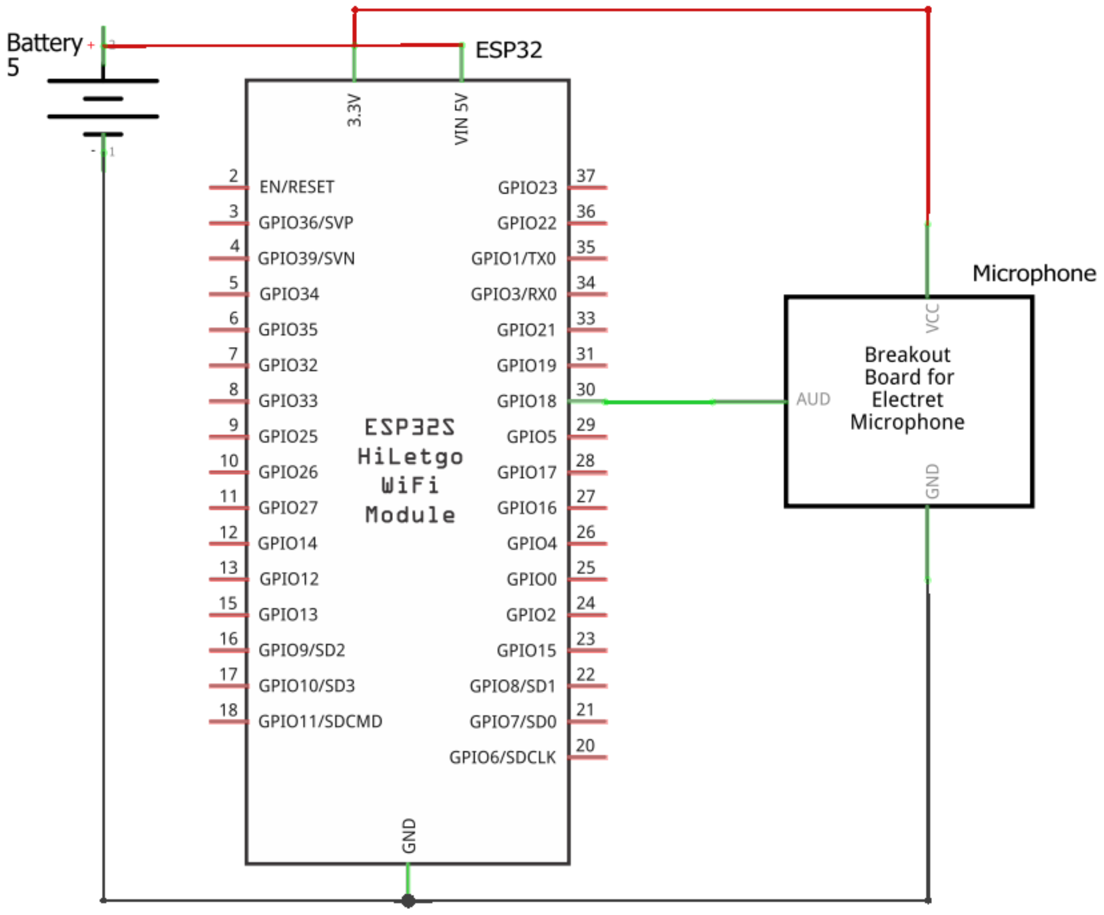


Figure 3.5: Schematic diagram of user's side

The ESP32 and the microphone are connected to the Ground of the battery, AUD from the microphone connects to the GPIO18 pin of the ESP32, VCC from the mic is connected to 3.3V of ESP32 and 5V of ESP connects to the positive terminal of the battery.

The schematic diagram of the mobile robot illustrated in Figure 3.6 displays the hardware components and their interconnections. A DC motor is connected to the L293D motor driver to supply power. Therefore the motor driver is directly powered by a 6V battery.

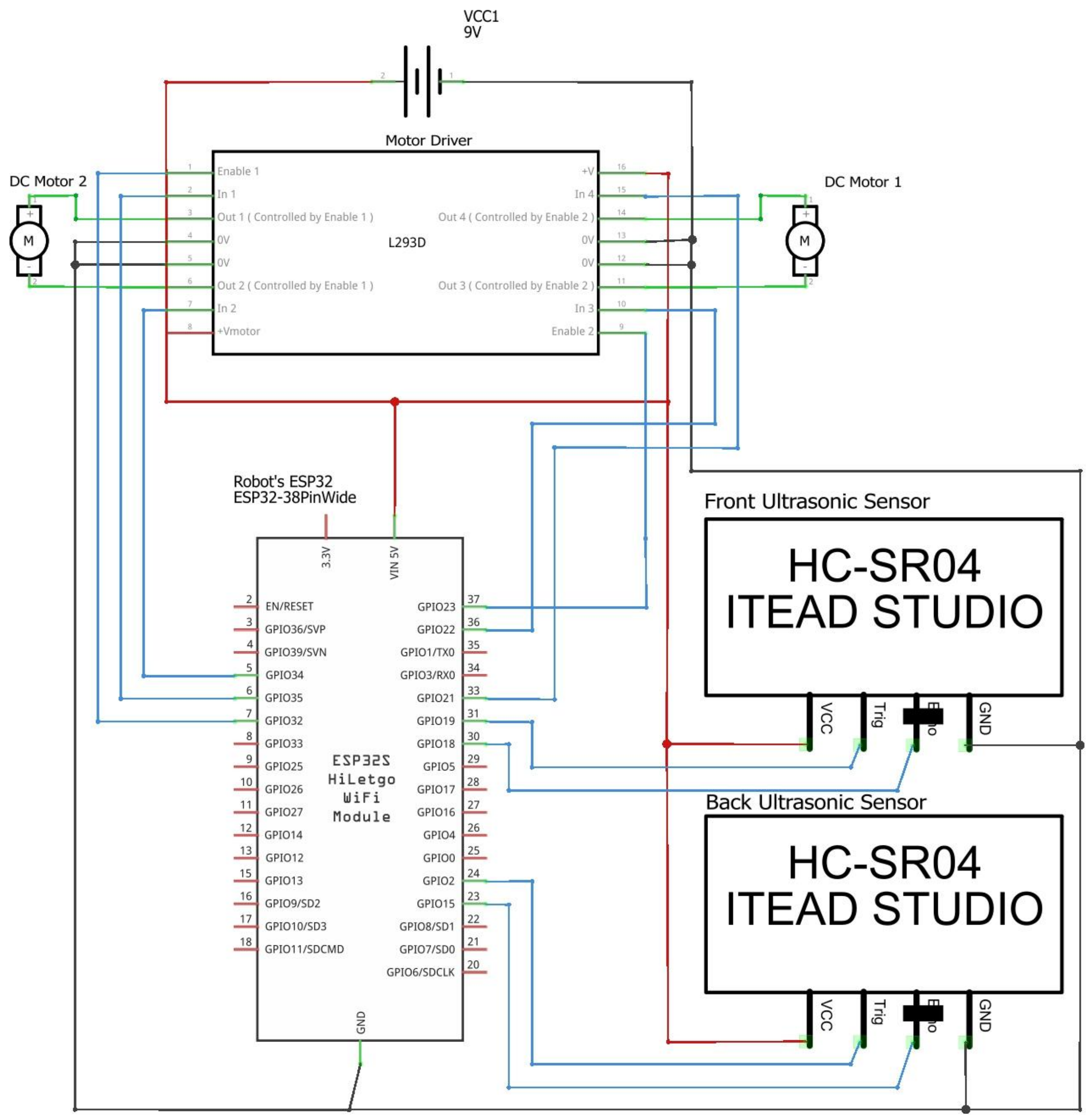


Figure 3.6: Schematic diagram of mobile robot

The Ultrasonic sensors are connected to the ESP32 directly, while the motors are connected to the Motor driver which is then connected to the ESP32. Ultrasonics, Motor drive, and the ESP32 are connected to the positive terminal of the 6V battery.

## 3.6 System Software Design

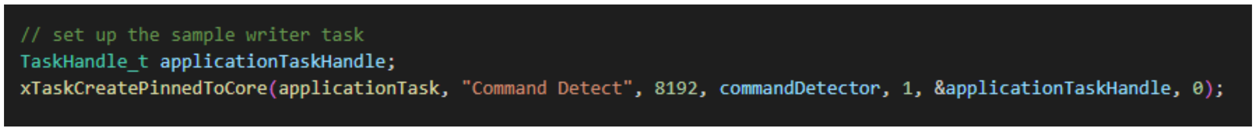
The software of the system will be responsible for recognizing a set of commands in the voices around the mic and ignoring unrelated voices. The commands are Forward, Backward, Right, Left, and Stop. The recognition is done by training a neural network on a large dataset of recorded commands. The below section shows more details about the training algorithm.

### CPU and Memory Optimization

Since ESP32 has limited processing power and memory, we used freeRTOS to operate the system, since it's real-time and makes efficient use of the CPU and memory and it also takes advantage of ESP32 being a dual-core microprocessor.

Instead of working in a single loop, RTOS divides work into multiple tasks, each task has a priority. Tasks with the highest priority execute first. The tasks will run interchangeably, while others will run concurrently on different cores.

“xTaskCreatePinnedToCore" is a freeRTOS function that tells the scheduler to run a task in only one of the cores. Its parameters are “the task to be executed”, “name of the task”, “stack-allocated size in bytes”, “pointer to memory”, “task priority (0-24) where 0 is the lowest”, “pointer to another task that manages this task”, and “CPU core (0 or 1 in our case)” as shown in figure 3.7.



```
// set up the sample writer task
TaskHandle_t applicationTaskHandle;
xTaskCreatePinnedToCore(applicationTask, "Command Detect", 8192, commandDetector, 1, &applicationTaskHandle, 0);
```

Figure 3.7: Run task on a specific core

To slice the continuous voice signal into multiple 1-second windows, we take advantage of the RTOS function “vTaskDelay”. It has an advantage over the ordinary delay() function because it tells the scheduler to run other tasks until the specified delay time is up, and then come back to continue running this task.

We make a delay for 1-second so the buffer gets filled with a voice signal to be ready for processing and then do the command detection.

## Convolution Neural Network (CNN) Specifications

The input for the training is a list of spectrograms, each spectrogram is represented with a 3D list [time sample, frequency, amplitude], its fed to the first layer which is a Convolution layer, the max-pooling layer is then used to reduce the spatial dimensions of the output volume. The output is then fed to another convolution layer, another max-pooling layer is used. The next layer is fully connected and contains 80 units. The next layer is also a fully connected layer with 6 units corresponding to the 6 classification classes. The rectified linear unit (ReLU) is used as an activation function over all the layers except for the last dense layer softmax is used. The output is a list of six numbers, each value is between 0 and 1, each number represents the probability of each command, and the highest probability will be selected as the detected command. The output is six classes representing the 5 commands and "Invalid" which represents that no command is detected.

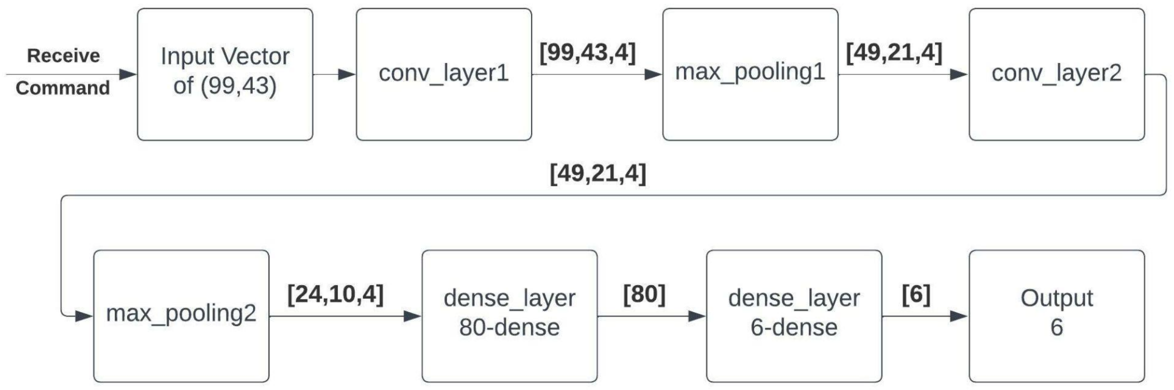


Figure 3.8: CNN Architecture

# Chapter 4: Implementation

## 4.1 Overview

This chapter explains the implementation part for the hardware components and the software algorithms. It dives into more details about the project's overall different hardware components and software modules.

## 4.2 Hardware Implementation

To send the detected movement from the first ESP32 to the second ESP32 after detecting the speech command, we connect them using the ESP-NOW protocol using the ESP-NOW configuration in C++.

Figure 4.1: ESP-NOW two-way communication [11].

The steps below show the hardware implementation for the robot:

1. Connect the DC motors with the L293D driver, the following steps describe the process:
    a. The first DC motor connects with the pins (3,6) on the IC board to set it as output.

   b. The second DC motor connects with the pins (11,14) on the IC board to set it as output.
2. Connect the L293D driver with the ESP32, follow the steps below:
   a. Pins (2,7) from the IC connect with GPIO13 and GPIO25 pins respectively from ESP32 to control the direction of the first motor.
   b. Pins (10,15) from the IC connect with GPIO12 and GPIO14 pins from ESP32 to control the direction of the second motor.
3. Connect the Ultrasonic sensors with the ESP32:
   a. Every sensor has 4 pins (VCC, Trigger, Echo, Ground).
   b. The **trigger** pin from the first sensor connects with the GPIO27 pin from the ESP32, and the echo pin connects with the GPIO32 pin.
   c. The **trigger** pin from the second sensor connects with the GPIO26 pin from the ESP32, and the echo pin connects with the GPIO33 pin.
   d. The VCC and Ground pins from the sensors connect with the VCC and Ground pins from the ESP32.
4. Connect the microphone with the ESP:
   a. The mic has 3 pins (VCC, Ground, Output).
   b. Connect the VCC and Ground pins from the mic with the VCC and Ground pins from the ESP32.
   c. Connect the output pin from the mic with pin GPIO35 in the ESP32.

Figure 4.2 shows the final look of the robot after connecting its parts.

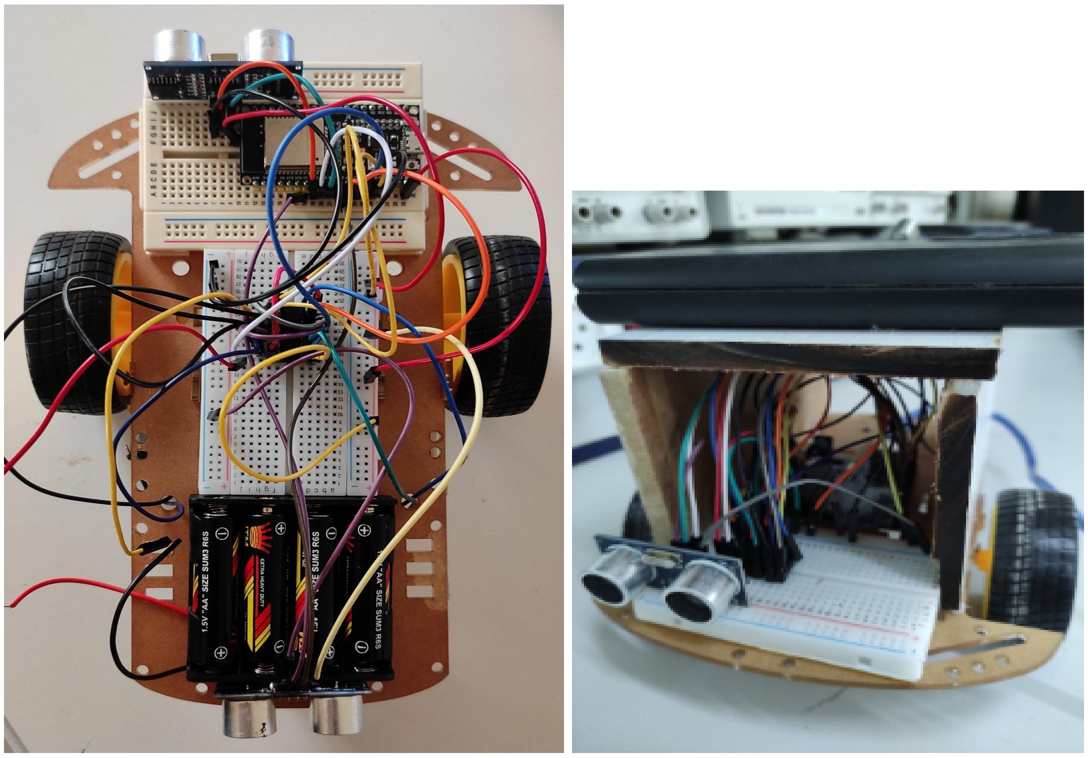


Figure 4.2: Final look of the robot

## 4.3 Software Implementation

The major work in this project has been in software implementation, since it's responsible for reading the analog voice signal from the mic, converting it into MFCC, feeding the resulting MFCC into the trained model, the trained model will determine the meaning of the signal, and finally make the robot respond accordingly.

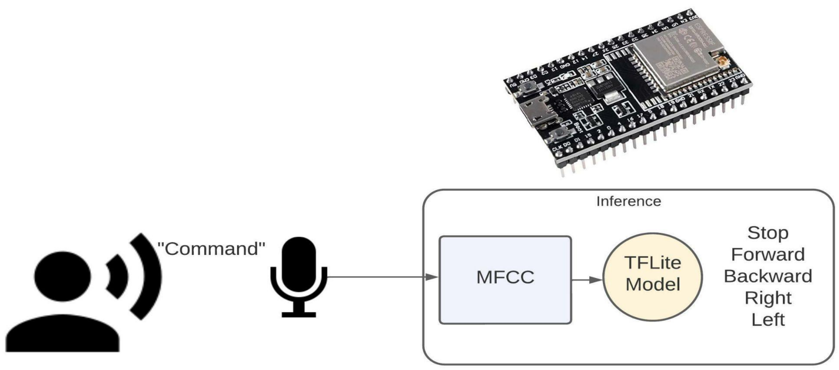


Figure 4.3: Software modules

### 4.3.1 Processing Module

In this module one-second-long audio will be sliced from the input voice signal, then it will be sampled by 16KHz, so 1 second is 16000 samples. Finally, MFCCs will be calculated for this portion.

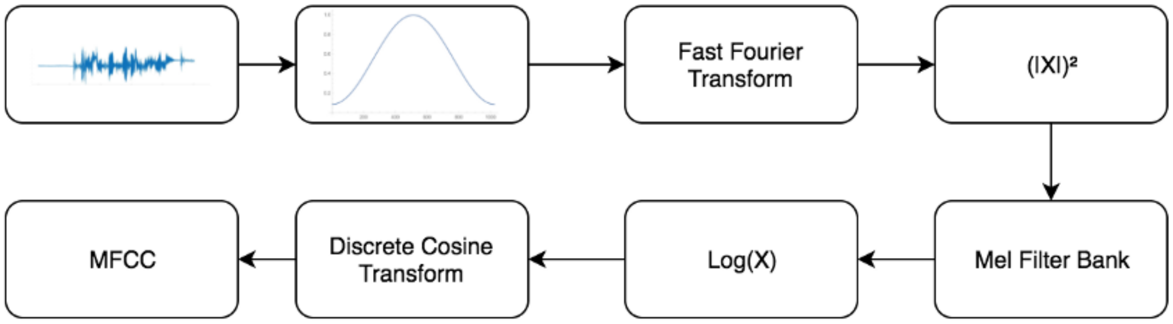


Figure 4.4: MFCC's extraction algorithm [5].

#### 4.3.1.1 Sliding window (Hamming)

The robot will be continuously analyzing voices around it while receiving, but as we know the voice signal is continuous and the processing power the robot has is limited. so we split the input signal into equally long parts (windows), we choose the windows to be one-second-long because our commands are single word commands and can fit easily in one second.

All the voice processing then will be applied to every window, one at a time, and the robot will determine if that window contains a command or not.

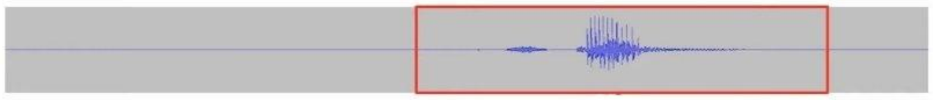


Figure 4.5: Extracting One second from voice signal [5].

A very serious problem we faced using this approach which highly affects the performance is that there is no guarantee that splitting comes in the middle of the speech command. take for example command "forward", it may happen that "for" comes in one window while "ward" comes in the next window. In this case, the command “forward” will not be recognized in any one of these windows.

To overcome this problem we make the step size between two windows 0.3 seconds, not a full second. to put it in other words the second window does not start from the end of the previous window, but after 30% of the beginning of the previous window. For example, a 2-seconds voice signal will result in 6 windows. If the command hadn't been detected in the first or the second window, it surely will in the third.

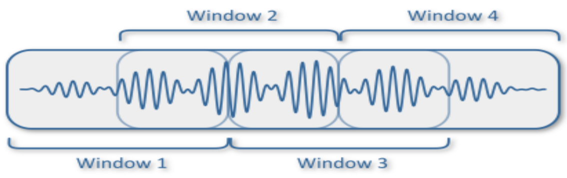


Figure 4.6: Signal sliding windows [5].

### 4.3.1.2 Feature extraction

To make the robot able to differentiate between given speech commands, it must be able to extract features from the voice signal read from the mic, these features are used to distinguish speech commands. The analog signal alone is not sufficient which is represented by frequencies and amplitudes, so we calculate Mel-frequency cepstrum (MFCCs) from the voice signals. The steps are clarified in figure 4.7.

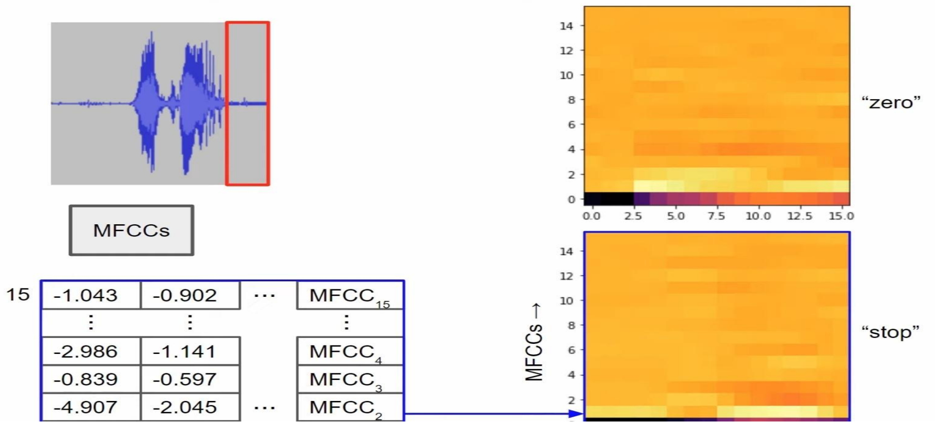


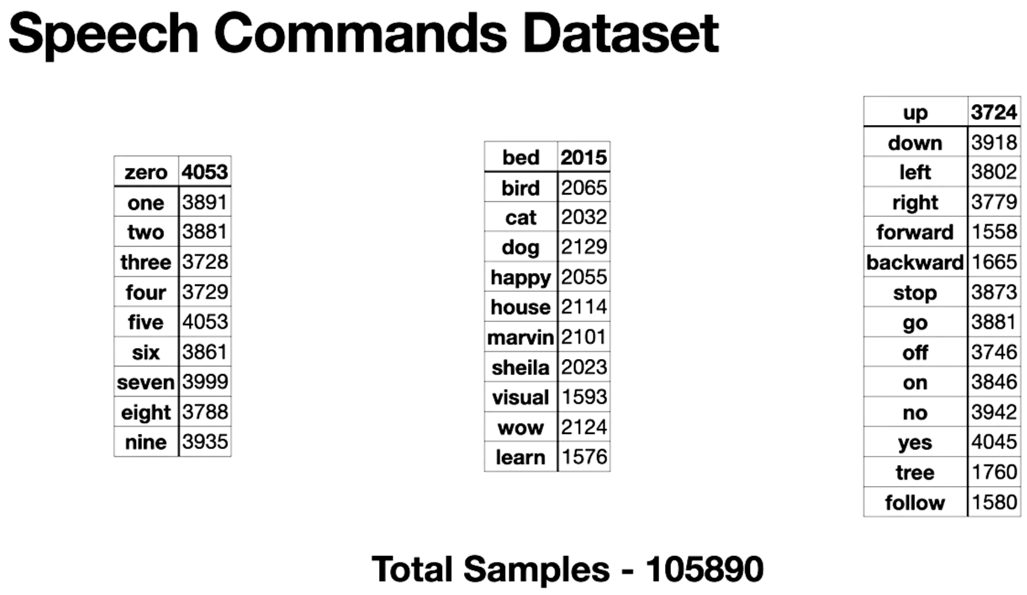
Figure 4.7: Getting MFCCs from voice signal [8].

### 4.3.2 Detection Module

The detection module is responsible for detecting the commands in the MFCCs received from the processing module.

#### 4.3.2.1 Training the CNN model

We used Google's commands speech dataset to train our model for commands ('forward", "backward", "right", "left", "stop") . We used all available samples in the dataset and to make the robot ignore other commands, we added another classification class besides previous commands, we called it "invalid". This class contains samples from all words in the dataset except words used as commands; 600 samples from each word were used. Figure 4.8 shows more details about Google's speech commands dataset.



**Speech Commands Dataset**

| zero | 4053 |
|---|---|
| one | 3891 |
| two | 3881 |
| three | 3728 |
| four | 3729 |
| five | 4053 |
| six | 3861 |
| seven | 3999 |
| eight | 3788 |
| nine | 3935 |

| bed | 2015 |
|---|---|
| bird | 2065 |
| cat | 2032 |
| dog | 2129 |
| happy | 2055 |
| house | 2114 |
| marvin | 2101 |
| sheila | 2023 |
| visual | 1593 |
| wow | 2124 |
| learn | 1576 |

| up | 3724 |
|---|---|
| down | 3918 |
| left | 3802 |
| right | 3779 |
| forward | 1558 |
| backward | 1665 |
| stop | 3873 |
| go | 3881 |
| off | 3746 |
| on | 3846 |
| no | 3942 |
| yes | 4045 |
| tree | 1760 |
| follow | 1580 |

**Total Samples - 105890**

Figure 4.8: Commands Dataset

To make data classes balanced we multiplied each command data as required, we noticed that some commands like "stop" generalized well with less data, if we add more it will overfit, so the command "stop" required smaller data than other commands. On the other hand, "forward", and "backward" commands required more data to generalize well. Figure 4.9 shows the training data distribution.

```
(array([11130., 12250.,  9380., 10555.,  8016., 16738.]),
 array([0, 1, 2, 3, 4, 5, 6]),
 <a list of 6 Patch objects>)
```

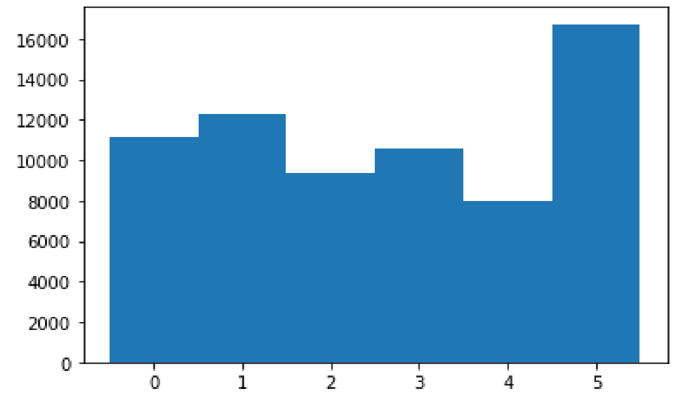
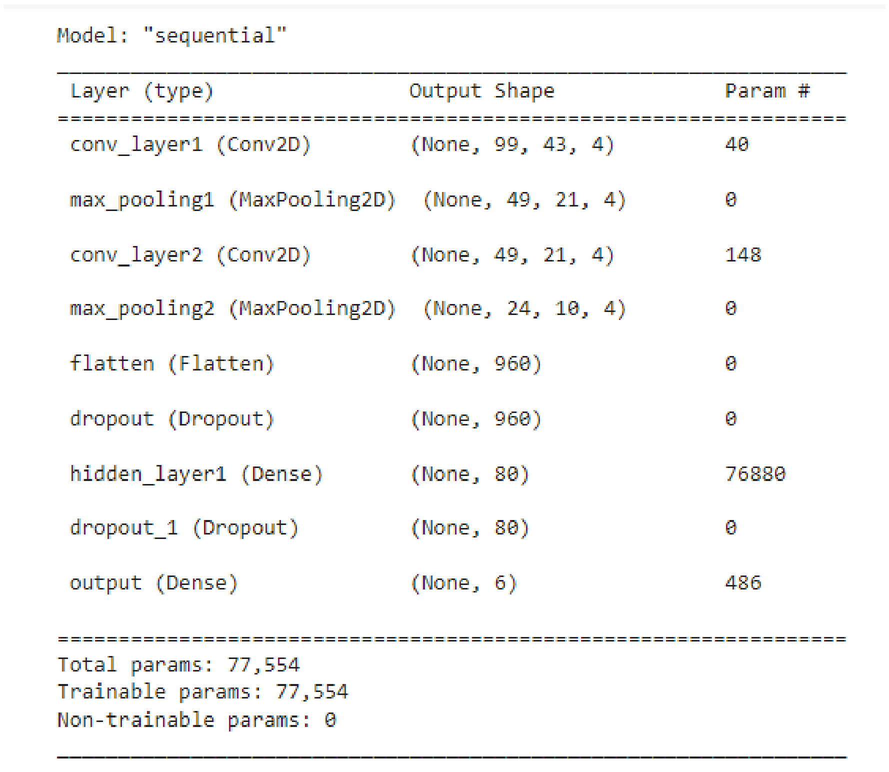


Figure 4.9: data distribution

The six numbers in the x-axis [0-5] represent the 6 classification classes of the model, which are respectively: forward, backward, left, right, stop, and invalid. A summary of the training is seen in figure 4.10



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv_layer1 (Conv2D)        (None, 99, 43, 4)         40

 max_pooling1 (MaxPooling2D)  (None, 49, 21, 4)        0

 conv_layer2 (Conv2D)        (None, 49, 21, 4)         148

 max_pooling2 (MaxPooling2D)  (None, 24, 10, 4)        0

 flatten (Flatten)           (None, 960)               0

 dropout (Dropout)           (None, 960)               0

 hidden_layer1 (Dense)       (None, 80)                76880

 dropout_1 (Dropout)         (None, 80)                0

 output (Dense)              (None, 6)                 486

=================================================================
Total params: 77,554
Trainable params: 77,554
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.10: CNN Neural Network Model Summary

Our training process was limited to 15 epochs which is the number of complete passes through the training dataset, given the small number of parameters we have in the model, and we noticed that beyond 15 epochs the model will over train providing a higher loss. Figure 4.11 presents the accuracy metric data throughout the training process, also Figure 4.12 shows the loss metric data throughout the training process.

Orange: training accuracy; Blue: testing accuracy

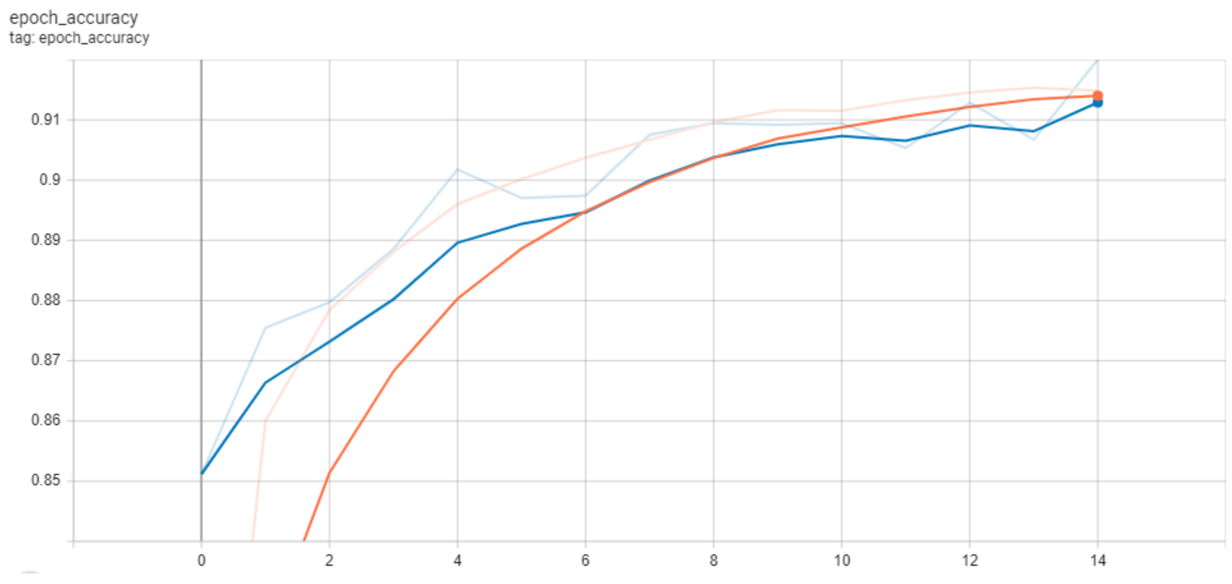


Figure 4.11: Epoch Accuracy

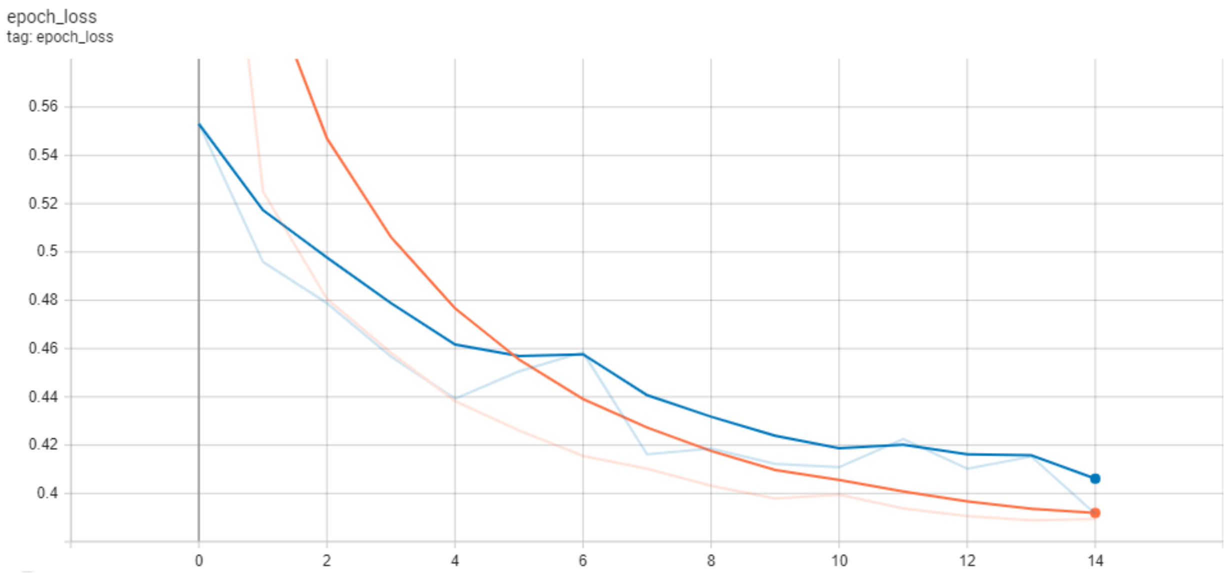


Figure 4.12: Epoch Loss

Figure 4.13 shows the confusion matrix which shows the accuracy of recognizing each command alone.

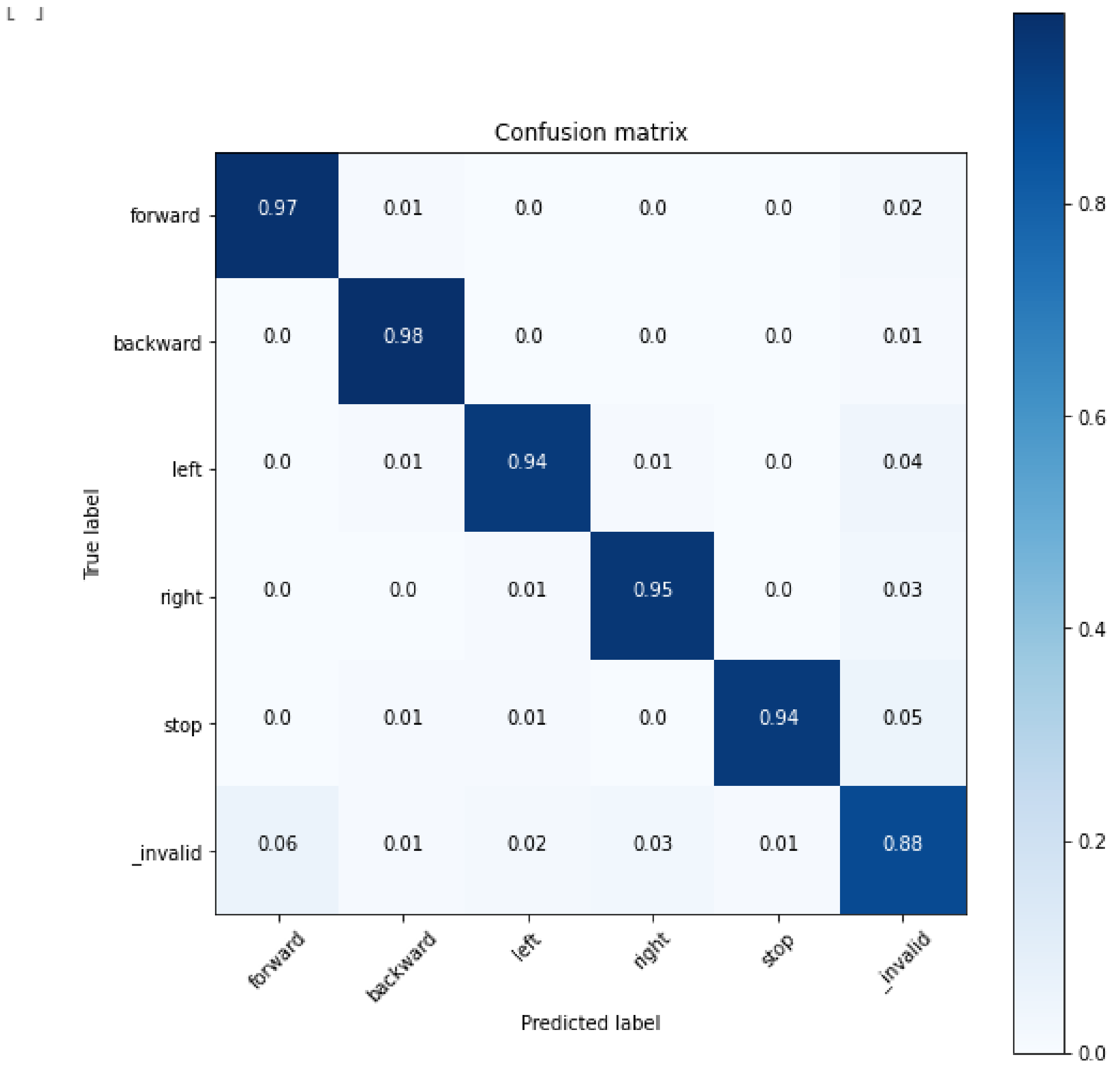


Figure 4.13: Training accuracy confusion matrix

## 4.4 ESP32 Implementation

Just by simply connecting the ESP32 to a pc via a USB port, it will be ready to work perfectly with any IDE like the Arduino IDE or Visual Studio Code. We used Visual Studio Code with platformIO extension in our project because we organized code into modules for easier implementation and debugging.

The ESP32 on the Robot is responsible for controlling the hardware components, which are responsible for moving or stopping the robot depending on the speech commands and distance measurements using ultrasonic sensors.

The following steps explain how we have implemented the robot components:

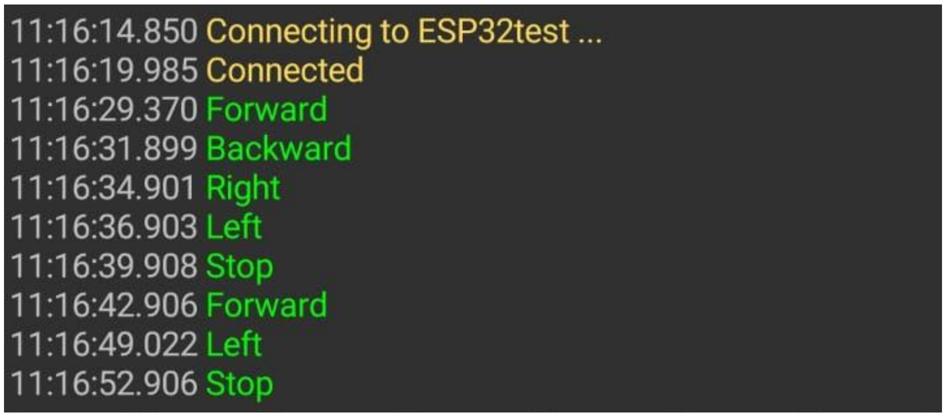
1. The robot will be standing when the program is running.
2. The ESP32 will read the distance between the robot and the objects in front and back of it. There are two scenarios for that as follows:
    a. If the movement is one of forwarding or right or left and the distance between the robot and any object in front of it is less than 30cm, the robot will stop immediately.
    b. If the movement is backward and the distance between the robot and any object behind it is less than 30cm, the robot will stop immediately.
3. As we mentioned previously, the first ESP32 pairs with the second ESP32 with a short packet transmission protocol called ESP-NOW to allow transferring of the data. The data is a struct containing 5 boolean variables to represent the 5 commands, only one will be true at a time. The robot will move depending on this data as follows:
    a. If "forward" is HIGH and there is no object in front of the robot, the robot moves forward.
    b. If "right" is HIGH and there is no object in front of the robot, the robot moves to the right.
    c. If "left" is HIGH and there is no object in front of the robot, the robot moves to the left.
    d. If "backward" is HIGH and there is no object behind the robot, the robot moves backward.
    e. If "stop" is HIGH, the robot stops

### 4.4.1 Movement feedback

Since the robot might not be in sight during its movement, we had to implement a method to give feedback on the robot's movement whether it's a successful movement or a failed one. So, we used a mobile app to receive a feedback signal from the robot to inform the user of the robot's position.

After receiving and detecting the command from the user the command shall be applied in real-time, the signal will be displayed on the user's cellphone which will be connected to the Robot ESP32 via Bluetooth.

The following snippet from the mobile app page shows the feedback signal displayed on the user's cellphone.



```
11:16:14.850 Connecting to ESP32test ...
11:16:19.985 Connected
11:16:29.370 Forward
11:16:31.899 Backward
11:16:34.901 Right
11:16:36.903 Left
11:16:39.908 Stop
11:16:42.906 Forward
11:16:49.022 Left
11:16:52.906 Stop
```

Figure 4.14: Feedback signals from the robot

## 4.5 Implementation Issues

During the planning and implementation phases we faced some issues that we managed to resolve, some issues occurred when we were choosing parts for the hardware model and other issues happened when we tried to implement these parts we chose, here we mention some of the main problems we managed to resolve:

### 4.5.1 Hardware selection

The main issue we faced was when we started to select the appropriate hardware considering the tasks we wanted to execute. First of all, we went with other microcontrollers instead of the current ESP32 but we couldn't get the speed and connectivity needed, especially since we had to make sure the system works in real-time.

Regarding the input method, at first, we went with wireless microphones and we started to compare the best microphones we could find, but we found that wireless microphones do not provide the connectivity and speed we need so we went with two ESP32 microcontrollers that connect to each other via ESP-NOW protocol.

### 4.5.2 Real-time execution

After implementing many demos and trying several approaches we had a hard time reaching real-time system execution but we could successfully reach suitable software codes and approaches that fulfill our needs and function in real-time.

# Chapter 5: Validation and Results

## 5.1 Overview

This chapter illustrates the testing process of the overall system, its components, and software. We test all the parts to ensure that all the functions work as expected and without errors.

## 5.2 Hardware Testing

This section discusses the testing process for the hardware components.

- **Ultrasonic Sensors**

The mobile robot is turned on and we check that the robot is able to stop when there is an obstacle in the way of the robot and it successfully stops as intended.

- **DC Motor and L293D driver**

After we've connected the car's components, we tested it by connecting it to the laptop. We've used Visual Studio Code to send data to the robot. The process was successful, and the robot moved into all sides without problems.

- **Connectivity**

We tested the connectivity between the two ESPs by sending random data from variant distances and it showed reliable and consistent connectivity. We also tested the connectivity between the robot ESP32 and the phone for feedback responses which also returned reliable and consistent results.

## 5.3 Software Testing

This section discusses the testing of our system's features.

### 5.3.1 Speech Commands Recognition Model

The entire speech detection model is implemented on the user's ESP32. We tested the speech commands detection model with the help of several volunteers with different voice tones to give the system a little challenge to detect the commands and it showed very good results. To push the challenge a little further, we manipulated the noise rates by repeatedly changing the test environments with every punch of samples. A preview of these tests and the calculations of accuracy rates are shown in the next section.

## ● Error Rate

After we've tested every hardware part of the system, we started testing the speech recognition by trying several different voice tones, so we asked  four colleagues students, one of them is a female, she's identified by "Person 4" to give 20 trials of each command so we can calculate the actual accuracy of the speech recognition system. and the results were as follows:

| **Word/Person** | **Person 1** | **Person 2** | **Person 3** | **Person 4** | **Average Accuracy** |
|---|---|---|---|---|---|
| **Right** | 18* | 19 | 17 | 18 | 0.9 |
| **Left** | 17 | 18 | 18 | 17 | 0.875 |
| **Backward** | 17 | 17 | 18 | 16 | 0.85 |
| **Forward** | 18 | 17 | 16 | 17 | 0.85 |
| **Stop** | 15 | 18 | 16 | 16 | 0.8125 |

*18 successful attempts from 20 attempts.

Table 4.1: Speech commands accuracy with different volunteers.

$$\text{Error rate for all the system} = \frac{\sum_{i=0}^{n-1} error\ rate\ for\ each\ word}{n}$$

$$= \frac{(0.4+0.5+0.6+0.75)}{20}$$

$$= 0.1125$$

Hence, the success rate of the system is 88.75%.

As the above table shows, The accuracy rates are quite lower than the model training accuracy rates and that's due to the environmental variants such as noise and voice tone consistency, these factors and more cause a slight difference in final results from the abstract results.

# Chapter 6: Conclusions and Future Work

## 6.1 Conclusion

In this project, we used deep learning algorithms to create an embedded system of a mobile robot that can be moved in real-time by speech commands which can be detected from the user.

The system consists of two parts, the user side which includes a mic that receives voice data to be processed to extract the command which then will be sent to the second part includes a mobile robot that will be moved according to the command received.

## 6.2 Future Work

After accomplishing this project successfully, The project can be expanded in the following features and the following is a description:

- Extend the set of words to further help the user communicate with the system.
- Add more features to the project to make it more useful to the user like adding a camera to get a live stream of the robot's movement.
- Add Arabic words to the system so it can be used in domestic or local applications.

## Summary

We know that everybody deserves to live a decent life where they can perform their daily life normally, but some people have been living with special body disabilities where they find difficulties walking and moving things around, so we believe that this project will help those who struggle to be able to move and interact just like normal people.

Machine learning and deep learning have already decreased the distance between the physical and digital world by equipping systems with smart thinking capabilities. In this project, we have been able to exploit these technologies along with many other techniques to make a speech recognition system that's able to detect and recognize speech commands to move a mobile robot.

To the best of our knowledge we used the best hardware option available, we have implemented this embedded system using two ESP32 to detect and recognize speech commands that control a mobile robot. The software system is mainly concerned with training neural networks to recognize speech commands. All captured signals are fed to the microcontroller to be processed and then fed to neural networks to be classified. It will eventually be a machine learning classification problem and the commands are the classes for classification.

## References

[1] Allan, A. 2019. "Meet the New Raspberry Pi 4, Model B." Hackster.io. https://www.hackster.io/news/meet-the-new-raspberry-pi-4-model-b-9b4698c284.

[2] Anandakanda. 2021. "Is Raspberry Pi a Microcontroller or a Microprocessor? –." anandakanda. https://www.anandakanda.in/is-raspberry-pi-a-microcontroller-or-a-microprocessor/.

[3] Aravindpai, Pai. 2019. "Signal Processing | Building Speech to Text Model in Python." Analytics Vidhya. https://www.analyticsvidhya.com/blog/2019/07/learn-build-first-speech-to-text-model-python/.

[4] Butler, Sydney. 2020. "How To Control Your Windows 10 PC With Your Voice." Online Tech Tips. https://www.online-tech-tips.com/windows-10/how-to-control-your-windows-10-pc-with-your-voice/.

[5] CHONÉ, Antoine. 2018. "Computing MFCCs voice recognition features on ARM systems." Medium. https://medium.com/linagoralabs/computing-mfccs-voice-recognition-features-on-arm-systems-dae45f016eb6.

[6] Chopra, P., and H. Dange. 2007. "VOICE CONTROLLED ROBOT." *University of Mumbai, Department of Electronics Engineering*.

[7] CSDN. 2019. "树莓派4 与英伟达Jetson Nano 性能大比拼，谁是最佳的嵌入式"电脑"？." 每日頭條. https://kknews.cc/zh-cn/tech/bmo56qj.html.

[8] DigiKey. n.d. “TensorFlow Lite Tutorial Part 1: Wake Word Feature Extraction.” Digikey. Accessed May 23, 2022. https://www.digikey.com/en/maker/projects/tensorflow-lite-tutorial-part-1-wake-word-feature-extraction/54e1ce8520154081a58feb301ef9d87a.

[9] Dyakonov, Alex. 2020. “Self-Supervised Machine Learning.” Dasha.AI. https://dasha.ai/en-us/blog/self-supervised-machine-learning.

[10] Espressif. 2020. “ESP32WROOM32E ESP32WROOM32UE.” Espressif Systems. https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf.

[11] Espressif Systems. n.d. “ESP-Now Overview.” Espressif Systems. Accessed February 15, 2022. https://www.espressif.com/en/products/software/esp-now/overview.

[12] Google. n.d. “TensorFlow.” TensorFlow.org. Accessed February 15, 2022. https://www.tensorflow.org/.

[13] HackerBoxes. n.d. “HackerBox 0065: Realtime : 11 Steps.” Instructables. Accessed May 22, 2022. https://www.instructables.com/HackerBox-0065-Realtime/.

[14] Hewlett Packard Enterprise. 2021. “DEEP LEARNING & MACHINE LEARNING SOLUTIONS.” Hewlett Packard Enterprise. https://www.hpe.com/il/en/compute/hpc/deep-learning.html?jumpid=ps_iv1byfw7se_aid-520042864&ef_id=Cj0KCQiA4b2MBhD2ARIsAIrcB-TVr1U5ZAt1uNhMOtXpV4sImBSLzcffGlKXxBSCIOLugj68JokA9s0aAqBfEALw_wcB:G:s&s_kwcid=AL!13472!3!543281946661!p!!g!!deep%20learning!1450377.

[15] IBM. 2020. “What is Deep Learning?” IBM. https://www.ibm.com/cloud/learn/deep-learning.

[16] IBM. 2020. “What are Recurrent Neural Networks?” IBM. https://www.ibm.com/cloud/learn/recurrent-neural-networks.

[17] JavaTPoint. n.d. “TensorFlow | Types of RNN.” Javatpoint. Accessed February 15, 2022. https://www.javatpoint.com/tensorflow-types-of-rnn.

[18] Logitech. n.d. “Logitech C270 HD Webcam, 720p Video with Noise Reducing Mic.” Logitech. Accessed February 15, 2022. https://www.logitech.com/en-us/products/webcams/c270-hd-webcam.960-000694.html.

[19] Makhi, M. 2021. “Machine learning With Python.” GitHub. https://github.com/mandarmakhi/Machine-learning-With-Python-.

[20] Maxim Integrated. 2016. “MAX9814 - Microphone Amplifier with AGC and Low-Noise Microphone Bias.” Maxim Integrated. https://datasheets.maximintegrated.com/en/ds/MAX9814.pdf.

[21] Meta. n.d. “PyTorch.” PyTorch.org. Accessed February 15, 2022. https://pytorch.org/.

[22] Miller, Liz. 2020. “How to use L298N Motor Driver.” Learn Robotics. https://www.learnrobotics.org/blog/how-to-use-l298n-motor-driver/.

[23] Nvidia developer. 2019. “NVIDIA® Jetson Nano™ Developer Kit.” NVIDIA® Jetson Nano™ Developer Kit. https://docs.rs-online.com/5d0f/A700000006773861.pdf.

[24] Osiński, Błażej, and Konrad Budek. 2018. “What is reinforcement learning? The complete guide - deepsense.ai.” Deepsense.ai. https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/.

[25] RandomNerdTutorials. n.d. “Installing ESP32 in Arduino IDE (Windows, Mac OS X, Linux).” Random Nerd Tutorials. Accessed March 31, 2022. https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/.

[26] Raspberry Pi. n.d. “Raspberry Pi 4 Model B specifications – Raspberry Pi.” RaspberryPi.com. Accessed February 15, 2022. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/.

[27] RF Wireless world. n.d. “Ultrasonic vs Infrared (IR) | Difference between Ultrasonic sensor and Infrared sensor.” RF Wireless World. Accessed February 15, 2022. https://www.rfwireless-world.com/Terminology/Ultrasonic-vs-Infrared.html.

[28] Salameen, Nouman, and Anas Al-Shweiki. 2019. “Voice Controlled Robot.” Dspace. http://scholar.ppu.edu/handle/123456789/1994.

[29] Saravanan, M. 2020. “Arduino Based Voice Controlled Robot Vehicle.” *IOP Conference Series: Materials Science and Engineering* 993.

[30] Smales, Mike. 2019. “Sound Classification using Deep Learning | by Mike Smales | Medium.” Mike Smales. https://mikesmales.medium.com/sound-classification-using-deep-learning-8bc2aa1990b7 .

[31] Snehi, Jyoti. 2006. *Computer Peripherals and Interfacing*. N.p.: Laxmi Publications.

[32] Venkateswarlu, R.L.K, Vasantha Kumari, and Vani JayaSri. 2011. “Speech Recognition By Using Recurrent Neural Networks.” *International Journal of Scientific & Engineering Research* 2 (6): 7.

[33] Yildirim, Yetkin, and Akif Celepcikay. 2021. *artificial intelligence and machine learning applications in education*. N.p.: London International Conference. https://londonic.uk/js/index.php/ljis/article/view/49.