https://doi.org/10.24191/jeesr.v24i1.013

Wake Word and Speech Recognition Application on Edge Device: A Case of Improving the Electric Wheelchair

Low Jian He, Solahuddin Yusuf Fadhlullah*, Khadijah Kamarulazizi, Samihah Abdullah, and Shabinar Abdul Hamid

***Abstract*—Edge devices play an ever-increasing role as to reduce latency, improve efficiency and adapt simplicity. However, their lower processing capabilities compared to traditional setup means that their usage are also limited. This research explores the possibility of implementing both wake word and speech recognition on an edge device. The proposed wake word system, which is for voice activation, is developed based on the LSTM Neural Network model. The model is trained, modelled, and evaluated to respond to the wake word of "Hey SellTron". As for the speech recognition (voice command) system, Google Speech API was selected to recognize standard directional commands (left, right, forwards, backwards), as well as the new conceptual locational commands ("go to kitchen", "go to bedroom", etc.). To prove the feasibility of the developed system, these two features were integrated into an edge device on a mobile platform; representing a conceptual mini wheelchair 'prototype' due to project constraints. Evaluations showed that the prototype was able to activate and respond to voice commands correctly with over 80% accuracy in low ambient noise (<50dB).**

***Index Terms*—Voice activated wheelchair, wake word, speech recognition, LSTM, Google Speech API**

I. The Wheelchair as an Edge Device

Current commercial wheelchairs come in two modes: manual or electric. The manual wheelchairs operated by the hands and upper body can cause shoulder and palm injuries due to the repetitive pushing motions needed to propel it. According to [1], research estimate that upwards of 70 – 100% of wheelchair users will experience injuries and varying degrees of pain in their upper body region after usage; with the three primary regions being the wrist with 40 – 66% likelihood, the elbow with 5 – 16%, and the shoulder with 30 – 60%. In addition, injuries due to blisters, abrasions, and lacerations occurred 18% of the time, as well as carpal tunnel syndrome which has a 27 – 90% chance of occurrence [2].

The obvious solution to manual is the electrically powered wheelchair, which is normally operated using joysticks or buttons by hand. However, the drawbacks are mechanical in nature due to the factor of wear and tear of moving mechanical parts especially at the controller. Besides, maneuvering also requires a reasonable amount of awareness and motor skills. A hand's free solution could potentially address some of the issues by leveraging on the edge device capabilities.

Voice-controlled handsfree wheelchair is slowly but looks surely becoming a real possibility as speech recognition and detection technology have made major progress leaps in the past decade, even more so with the public release of ChatGPT and similar large language models (LLM). Looking slightly back, sparks of interest in wheelchair with voice recognition feature are already evident.

An instance of this can be seen in the implementation done by Vijay, which was achieved by pairing the Arduino platform with the HM2007 voice recognition module [3]. In a similar manner, Uchid also implemented a similar concept for the system by pairing an Arduino Uno R3 with a Greetech Voice Recognition Module V.3 [4]. Tan also went with the same route, by pairing a voice recognition module to the Atmega 328P Arduino microcontroller [5]. However, Tan's approach differs slightly, as the voice recognition module's connection to the Arduino microcontroller was done via Bluetooth, using the HC-05 Bluetooth Module. Hence, this evidently shows the usage possibility of pairing low-powered edge devices such as the Arduino with voice recognition modules for the voice-controlled wheelchair.

Further improvement can be seen from the introduction of the Internet of Things (IoT) to the wheelchairs. For example, Malik utilized the Google Web Service Application Programming Interface (API) for his Arduino Uno based voice-controlled wheelchair [6]. This API was integrated onto an

This manuscript is submitted on 10 January 2024, revised on 21 February 2024, accepted on 26 March 2024 and published on 30 April 2024. Low Jian He was with the School of Engineering, INTI International College Penang, 11900 Penang, Malaysia. He is now with the School of Computer Science and Electronic Engineering, University of Surrey, Guildford, UK (e-mail: jl02958@surrey.ac.uk).

Solahuddin Yusuf Fadhlullah and Khadijah Kamarulazizi are with the School of Engineering, INTI International College Penang, 11900 Penang, Malaysia. (e-mail: solah.fadhlullah@newinti.edu.my and khadijah.azizi@newinti.edu.my).

Samihah Abdullah and Shabinar Abdul Hamid are with the School of Electrical Engineering, College of Engineering, Universiti Teknologi MARA, Cawangan Pulau Pinang, 13500, Malaysia. (e-mail: samihah.abdullah@uitm.edu.my and shabinar@uitm.edu.my).

*Corresponding author
Email address: solah.fadhlullah@newinti.edu.my

Android app, and thus, the voice commands to the wheelchair will have to be given through the phone. The API on the app will first identify the speech given, before sending the classified data to the Arduino via Bluetooth. Similar implementations have also been done by both Anwer and Rakib, as they have utilized mobile applications for the speech recognition engine of their voice-controlled wheelchairs [7],[8].

For edge devices, the Raspberry Pi may currently be the most suitable platform for local deep learning speech recognition due to its low-cost, efficiency, size, and relatively high compute power [19]. However, even with those benefits, applying full scale deep learning models such as transformers on Raspberry Pi is not exactly feasible due to the level of compute needed. Thus, light-weight and effective neural networks will have to be used instead. To achieve this, several papers had explored utilizing the long short-term memory (LSTM) Neural Network, as it achieved impressive results while being light-weight [22]. For instance, Nimisha et al. demonstrated that LSTMs were lightweight enough to be implemented on a Raspberry Pi 3 as they successfully implemented a real-time speech emotion recogntion system that achieved great accuracies of 86% [17]. In terms of LSTM utilization for voice control, Sutikno proposed a voice-controlled wheelchair that implemented both the convolutional neural network (CNN) and LSTM for the voice recognition engine, and the end results showed an extremely positive 95% accuracy when the wheelchair was given voice commands [9]. Additionally, Bakouri had also explored a similar voice controlled wheelchair idea which integrated an LSTM model. However, in this work, the LSTM model was instead implemented onto an android phone; but nonetheless, it was still able to achieve a 98.2% accuracy for the 5 commands that was used [18].

Other than that, the maneuverability of the wheelchair is another major design decision. For instance, Rakib [8] has included directional push buttons into his wheelchair, while Answer [7] has opted to include an extremely innovative and experimental maneuvering system which is controlled by the human pupil. Traditional sensors such as ultrasonic sensors also play a vital role for obstacle avoidance such as the work by Malik [6] and Umchid [4].

To heighten the safety of the wheelchair, the speech recognition system should only be activated once the assigned wake word has been invoked. This would avoid unintentional voice commands from the background or accidental conversation from triggering the mobility of the wheelchair. The wake word functions by listening to the user's environment for the trigger word. This could violate the user's privacy as private conversations could potentially be recorded by the servicing company. This argument is strengthened by a Microsoft report [10], which states that 41% of voice assistant users are worried about trust, privacy, and passive listening regarding these assistants. Still, the physical safety of the user should offset the privacy concern, as with other services.

This research will demonstrate the implementation of a standalone wake word system onto the voice-controlled edge device (wheelchair) mini prototype. To the best of our knowledge, this is a first for such integration. The functionality of the voice control will be expanded beyond directional commands such as "right", "left", "forwards" and "backwards", as to also accept locational commands that enable the wheelchair to move to a specific location in just one sentence, such as "move to the kitchen" or "move to the living room". Locational commands are much more efficient and practical for traversing for wheelchair users.

II. Methodology and Implementation

A. Hardware

Fig. 1 shows all the required modules in the proposed system and describes how they are connected to one another. Raspberry Pi 4 was the platform of choice and responsible for the central processing of the system. The Pi 4 was powered by a 20,000 mAh power bank, and a Deity V4 Mini microphone was connected via USB to it as the audio input for voice. A Maker Pi HAT was also plugged onto the Pi 4 via the GPIO pins, which helped in simplifying the connections to the maker drive. The maker drive is a small circuit board which was used as the DC motor's driver; thus, it was connected to both the left and right DC motors and was powered by 3 AA batteries. These batteries could also be replaced with the power bank as the power source.

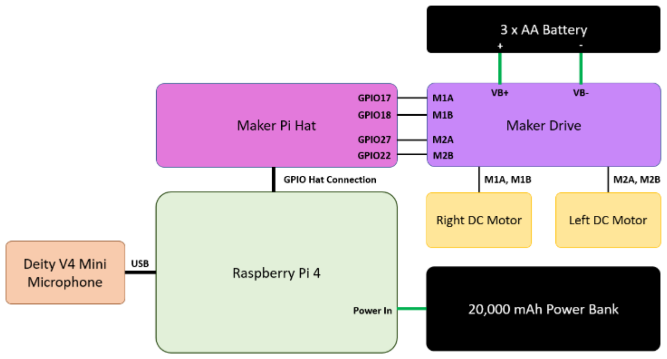

Fig. 1. Hardware block diagram.

Fig. 2 and 3 show the implementation of the prototype. Due to the scope and limitation of the project, the mechanical implementation of the wheelchair was omitted and replaced with a mini prototype as proof of concept.

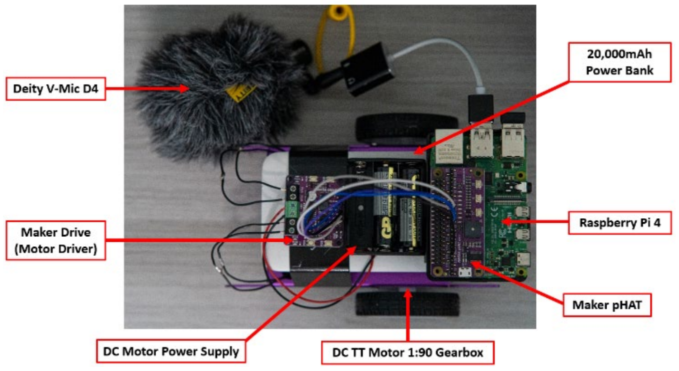

Fig. 2. Labelled modules of the prototype.

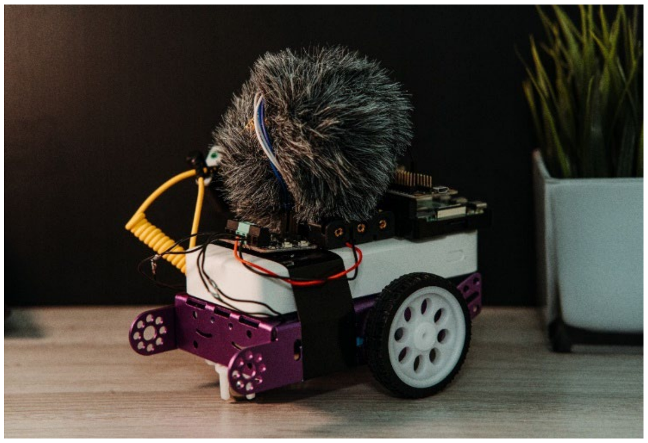

Fig. 3. The prototype.

B. *Working Principle*

The UML diagram in Fig. 4 summarizes the working principle of the proposed voice activated wheelchair system. The Raspberry Pi will act as the main platform for the project, loaded with a wake word listener, a deep learning wake word model and a Speech Recognition API.

"Hey Selltron" was assigned as the wake word as the word was not used anywhere else, and the "s" sound made the wake word more unique as compared to normal words. Due to this, manual recordings of the wake word had to be done and its variations were generated using software.

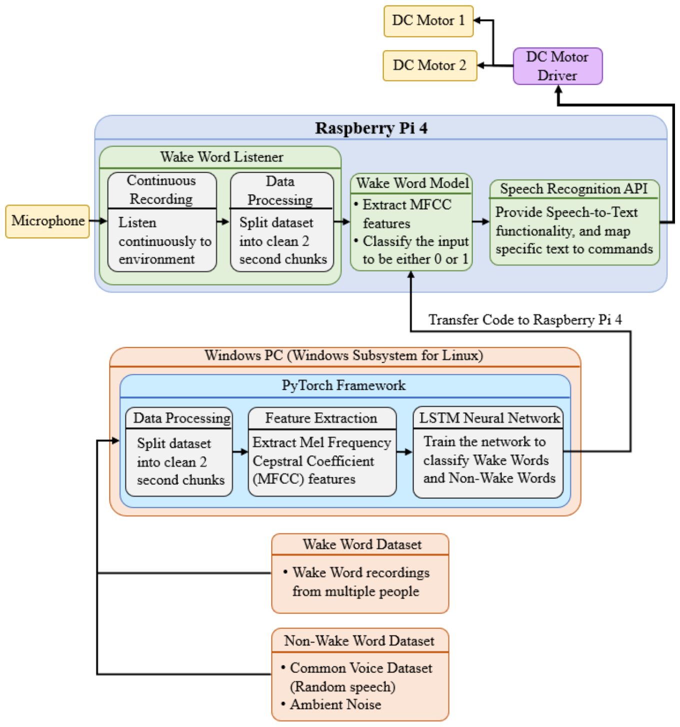
DC Motor 1
DC Motor 2
DC Motor Driver
Raspberry Pi 4
Wake Word Listener
Continuous Recording
Listen continuously to environment
Data Processing
Split dataset into clean 2 second chunks
Wake Word Model
• Extract MFCC features
• Classify the input to be either 0 or 1
Speech Recognition API
Provide Speech-to-Text functionality, and map specific text to commands
Microphone
Transfer Code to Raspberry Pi 4
Windows PC (Windows Subsystem for Linux)
PyTorch Framework
Data Processing
Split dataset into clean 2 second chunks
Feature Extraction
Extract Mel Frequency Cepstral Coefficient (MFCC) features
LSTM Neural Network
Train the network to classify Wake Words and Non-Wake Words
Wake Word Dataset
• Wake Word recordings from multiple people
Non-Wake Word Dataset
• Common Voice Dataset (Random speech)
• Ambient Noise

Fig. 4. UML Diagram of the proposed system

To create the wake word artificial intelligence (AI) model, the LSTM neural network was chosen for the project with further justification given in Section II-C. The wake word model was then built and trained in the WSL2 application on a Windows-based personal computer. To ease the modelling, the deep learning framework of PyTorch was utilized, as it had ready-made templates for the required neural network.

The model was trained with both a wake word and non-wake word dataset with its class labels included, so that the model could differentiate between a wake word and non-wake word. Further details on the datasets used will be discussed in Section II-D. Additionally, it should be noted that in the training process, data processing and feature extraction was needed. This is because the model is unable to understand raw data, thus data processing is applied in PyTorch to help extract the features that the model will understand.

After the training, the model will then be stored onto the Raspberry Pi 4, and it will enable it to predict the wake word when given audio data. For the Raspberry Pi to retrieve any audio data, a wake word listener system was programmed to carry out continuous audio recording via the microphone. In this listening system, data processing will also be performed, so that it matches with the understanding of the AI model. With that, the processed audio recordings will be fed to the trained wake word model, which will then allow the Raspberry Pi to do wake word detection.

The wake word listener will run continuously until the wake word is detected by the prototype. Once detected, the speech recognition API will then be activated. At this time, voice commands can then be given, and the API will try to recognize the command based on the user's speech. If a valid command like "front", "back", "left", or "right" is given, the DC Motor Driver will then be sent signals to move the DC motors accordingly.

C. *Deep Learning Model*

LSTM Networks, in short, is a deep learning, sequential neural network that is ideal for performing sequence prediction tasks. It is highly effective in understanding and predicting patterns in sequential data such as text and speech, making it very useful in speech recognition applications [21].

Structurally, LSTMs have a similar control flow when compared to RNNs; however, LSTMs have stark differences in their internal structures which allow them to retain long term information much better. One of these structural differences is the inclusion of an additional transport highway, that carries information throughout the entire sequence chain.

This metaphorical transport highway is called a cell state. Referring to Fig. 5, the cell state is the straight line at the most upper part of the LSTM's structure. In theory, this cell state will be the "memory" of the network as it will carry information all the way from earlier sequences along the process and eventually bringing it to later sequences; thereby reducing the effects of short-term memory [11].

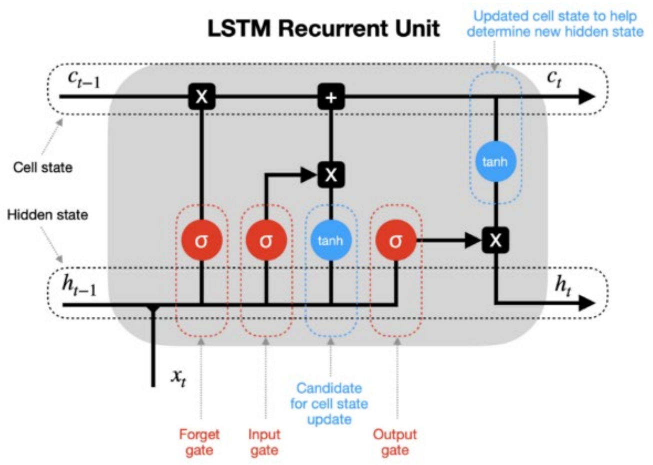

Fig. 5. LSTM recurrent unit block diagram [12].

It is also seen in Fig. 5 that there are various pointwise operations and gates which are used. These gates are vital in the LTSM structure as there are different neural networks that are trained to decide which data in the sequence will have to be learnt, which data must be retained and which previous data to forget due to irrelevancy. They work as a unit to add, retain, or remove information in the cell state, and they perform it in such a way that only a few linear interactions are required so that the information in the cell state can flow without much manipulation.

Since LSTMs are much better at retaining data as compared to RNNs, they are mostly the neural network algorithm of choice when it comes to cutting edge applications of speech recognition, machine translation, image captioning, handwriting generation, and question answering chatbots. LSTMs have also been proven to have the lowest error rates for wake word in clean condition [13]. Additionally, LSTM networks are also better than recurrent neural network (RNN) and Gated Recurrent Unit (GRU) as it excels in complex tasks and is also better at retaining long-term dependencies [20]. Thus, the LSTM was selected for the present work after considering its advantages over alternatives and its widely available resources.

D. Training Dataset

For the training, 2 classes of dataset will be needed, one being the non-wake word class, and one being the wake word class.

Fig. 6 shows the non-wake word class subdivided into 3 categories. For the category of random human voices, the open-sourced Common Voice Corpus from Mozilla was used. A small portion of the corpus, consisting of about 15000 clips was used for the dataset. As for the silent ambient data, the dataset was collected by leaving the microphone to record in a quiet room for 30 minutes. It was then sliced into 2 second clips and added into the dataset. For the noisy ambient data, this was recorded by leaving the microphone to record for 30 minutes; however, this time a JBL speaker was configured to play the ambience of noisy parks and crowded cafes. These 3 ambient noise data made up 1 hour of recording and was subsequently sliced and duplicated into 15000 data clips. In total, 30000 clips were used for the non-Wake Word dataset.

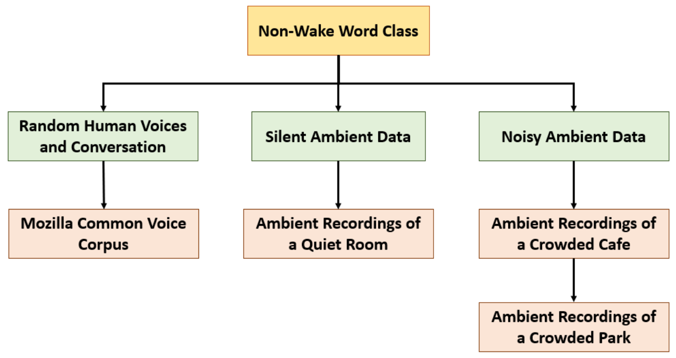
Non-Wake Word Class
Random Human Voices and Conversation
Silent Ambient Data
Noisy Ambient Data
Mozilla Common Voice Corpus
Ambient Recordings of a Quiet Room
Ambient Recordings of a Crowded Cafe
Ambient Recordings of a Crowded Park

Fig. 6. Non-wake word class dataset breakdown.

As before, the wake word class was also subdivided into 3 categories as in Fig. 7. For the self-recording category, the data was collected by taking 500 voice samples of self-recordings with varying pronunciations and pitches. This was done directly using the Raspberry Pi's microphone and without the use of speaker playback.

For the Text-to-Speech (TTS) recordings category, several TTS websites were visited to collect the wake word recordings. Some TTS websites were able to provide different accents and different intonations, which was collected to offer more variety in the dataset. The real-life recording samples category represents recordings of various people outside of the research group which also offer varying intonations and pitches for the dataset.

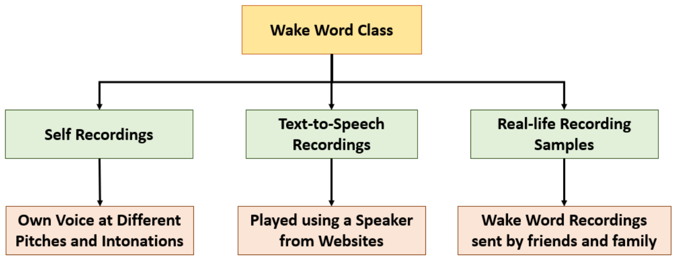
Wake Word Class
Self Recordings
Text-to-Speech Recordings
Real-life Recording Samples
Own Voice at Different Pitches and Intonations
Played using a Speaker from Websites
Wake Word Recordings sent by friends and family

Fig. 7. Wake word class dataset breakdown.

E. Training Process

The flowchart in Fig. 8 shows the full training process, with the orange flowchart representing the initialization phase and the blue flowchart representing the training and evaluation phase. During initialization, the program will first read through the .json files to understand the labels and locations of the dataset samples. Once this is done, each sample will then be converted into mel-frequency cepstral coefficients (MFCC), which is the picture representation of sound.

These MFCC samples then undergo a procedure called spectrogram augmentation. This procedure will modify each MFCC sample by warping it in the time domain, masking blocks of consecutive frequency channels, and masking blocks of utterance in time. Doing this will cause random losses of information to each dataset sample; however, this is done purposely so that the model will be forced to learn harder, which will allow it to become more robust against sample deformations after training.

Next, the augmented dataset is converted into a dataset object in Pytorch, where it is then inserted into a data loader. The Model's parameters will then be defined based on the arguments that had been inputted into the code. At this point, the LSTM class is then created by using the template from the torch.nn module and the specifications from the defined parameters. The loss function will also be defined by using the "BCEWithLogitsLoss" function from the torch.nn module to complete the initialization.

After initialization, the training process then begins with a user-defined learning rate and epoch. The training accuracy, loss, precision, and F1-score of the training is then calculated per batch and printed out onto the terminal. After training, the model is then validated using a separate validation dataset; and similarly, the model's accuracy, loss, precision, and F1-score is also calculated and printed out.

To prevent saving an unnecessary amount of models, the code was written so that it compared its most recent validation scores to its previous scores, and it only saved the model's weights to a checkpoint if there had been an improvement to the scores. This process was placed in a loop which ran continuously according to the given epoch. For the training, we implemented an early stopping approach, and halted the training once the model obtains an F1-score of 0.99 for more than 5 continous epochs. However, if the scores remained below 0.99 throughout the epochs, early stopping would not be implemented. Early stopping was implemented as a precaution to prevent the model from overfitting.

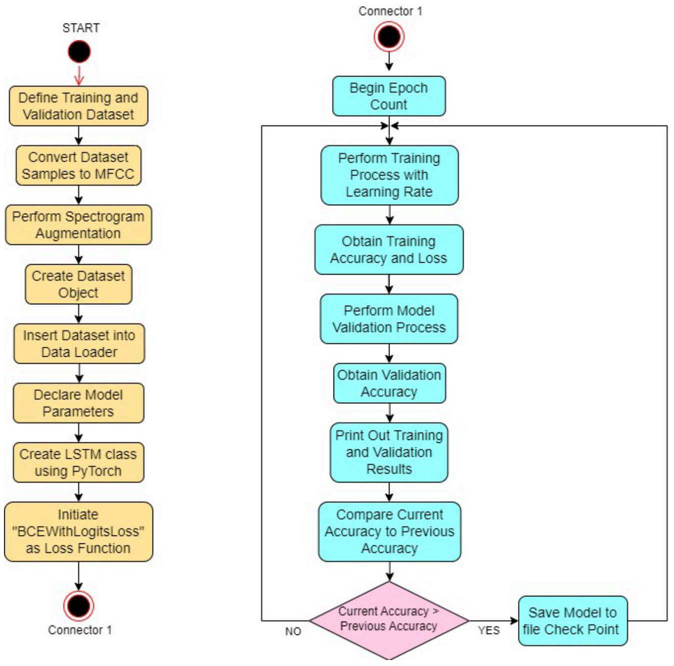
START
Define Training and Validation Dataset
Convert Dataset Samples to MFCC
Perform Spectrogram Augmentation
Create Dataset Object
Insert Dataset into Data Loader
Declare Model Parameters
Create LSTM class using PyTorch
Initiate "BCEWithLogitsLoss" as Loss Function
Connector 1
Connector 1
Begin Epoch Count
Perform Training Process with Learning Rate
Obtain Training Accuracy and Loss
Perform Model Validation Process
Obtain Validation Accuracy
Print Out Training and Validation Results
Compare Current Accuracy to Previous Accuracy
Current Accuracy > Previous Accuracy
NO
YES
Save Model to file Check Point

Fig. 8. Training process flowchart.

F. Voice Command

The integration of Googles speech API into our system is shown in Fig. 9. Firstly, a user will need to provide voice inputs so that the API can use its speech-to-text functionality to convert recognized words into strings. The strings of the recognized words are then stored into a variable which undergoes a logical comparison to see if it matches specific command strings that had been predefined. This whole process is a continuous loop which exits once a correct match is obtained. When a match is found, the action which corresponds to the matched command will be executed. For example, the wheelchair's motors will be programmed to move forward if the command string of "move forward" is matched.

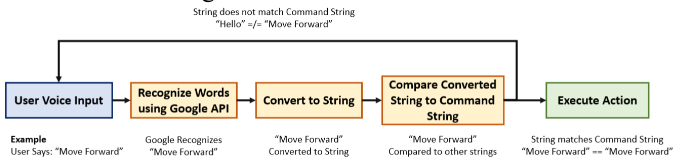
String does not match Command String
"Hello" =/= "Move Forward"
User Voice Input
Recognize Words using Google API
Convert to String
Compare Converted String to Command String
Execute Action
Example
User Says: "Move Forward"
Google Recognizes "Move Forward"
"Move Forward" Converted to String
"Move Forward" Compared to other strings
String matches Command String
"Move Forward" == "Move Forward"

Fig. 9. Voice command system flowchart.

A total of 11 commands were prepared. These commands can be found from the following list:

1. Turn Right
2. Turn Left
3. Go Forwards
4. Go Backwards
5. From Bedroom to Kitchen
6. From Kitchen to Living Room
7. From Bedroom to Living Room
8. From Living Room to Kitchen
9. From Kitchen to Bedroom
10. From Living Room to Bedroom
11. Exit

Each command is self-explanatory, with the motor's movements prewritten and mapped to the command. It should be noted that commands 5 to 10 will represent the locational commands, which will be used to demonstrate the prototype's ability to move to specific positions without further guidance.

G. Overall System

The flow chart in Fig. 10 describes the prototype's working principle in full, as it combines the wake word model, the voice command system, and the motorized hardware. The system will first start off by initializing the microphone. A 2-second continuous recording stream will then be started, and each sample will be stored into a recording queue. At this point, the samples will then be converted into their MFCC form, before being fed into the trained wake word model for prediction.

If the model does not predict the wake word, the system will return to its listening stage; however, if the model predicts that the MFCC sample is a wake word, the software will then move to the voice command stage. This is when the Google Speech API will be initialized and will wait for the user's input. If the user does indeed utter a valid command, then the software will perform the action; however, if not, the software will simply exit the voice command system and go back to wake word listening.

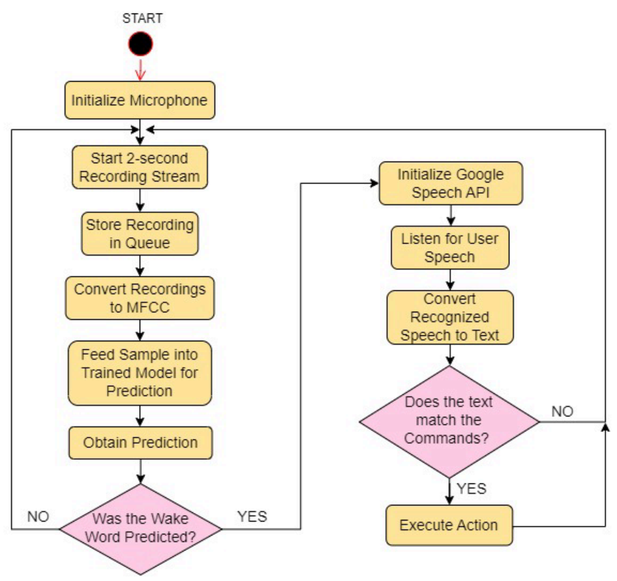
START
Initialize Microphone
Start 2-second Recording Stream
Store Recording in Queue
Convert Recordings to MFCC
Feed Sample into Trained Model for Prediction
Obtain Prediction
Was the Wake Word Predicted?
NO
YES
Initialize Google Speech API
Listen for User Speech
Convert Recognized Speech to Text
Does the text match the Commands?
NO
YES
Execute Action

Fig. 10. Overall System Flowchart.

H. Evaluation Methods

To evaluate the Wake Word Model and Voice Command system, tests in different scenarios and environments were conducted. In each test, the system was evaluated multiple times so that an accuracy value could be obtained. This value was calculated with the formula refer to (1) and was implemented in all tests that involved accuracy evaluations.

$$Accuracy = \frac{Number\ of\ Successful\ Attempts}{Total\ Number\ of\ Attemps} \times 100\% \qquad (1)$$

1. Detection Range

In this test, the Wake Word model and Voice Command System’s accuracy will be tested at different distances of input. Details of the tests can be found in Table I.

TABLE I. DETAILS FOR DETECTION RANGE TEST

Location	Quiet Controlled Room
Distances Tested	10 cm, 20 cm, 50 cm, 100 cm, 150 cm, 200 cm, 300 cm

For the test, each distance will be tested in separate sets of 5. Each set will consist of 20 prompting attempts, which adds up to 100 total attempts at each distance. After performing the 100 attempts, the accuracy will then be calculated using formula mentioned in section H.

2. Accuracy in Simulated Environments

For this test, the Wake Word model and Voice Command System’s accuracy will be tested at different levels (decibels) and types of noise. Details of the tests can be found in Table II.

TABLE I. DETAILS FOR NOISY ENVIRONMENT TEST

Location	Simulated Noisy Park Environment, Simulated Noisy Cafe Environment
Distance	Fixed 50 cm
Decibel Tested	40 dB – 75dB

To simulate noisy Park and Café environments, a high-quality surround sound JBL speaker will be utilized to play royalty free ambience samples of these environments.

The system will then evaluated in these simulated environments but at 3 different noise levels; with the noise levels being controlled using the volume toggle of the speaker. Additionally, Google Assitance’s Wake Word system on an android phone will also tested similarly so that its performance can be used as a benchmark.

3. Accuracy for Different Individuals

In this test, the Wake Word Model’s universal recognition property will be evaluated. Details of the tests can be found in Table III.

TABLE III. DETAILS FOR DIFFERENT INDIVIDUAL TEST

Location	Quiet Controlled Environment
Distance	Fixed 50 cm
Number of People Tested	10

A total of 10 (50% Male, 50% Female) candidates will be selected to take part in the test. Each candidate will be instructed to say the Wake Word, and the model’s subsequent response will be recorded down. Each candidate will also be asked to test the model 3 times, to make sure that results are consistent. Accuracy will also be calculated using the formula in section H.

4. Voice Command Accuracy and Compliances.

This test will evaluate the accuracy of every implemented voice command in the system and will also test its compliance. Details of the tests can be found in Table IV.

TABLE IV. DETAILS FOR VOICE COMMAND ACCURACY AND COMPLIANCE TEST

Location	Quiet Controlled Room
Distance	Fixed 50 cm

For this test, the system’s command recognition accuracy will first be tested. Each command will be tested in 5 sets of 20 attempts, totalling up to 100 prompting attempts.

Next, compliance testing will also be performed in a similar 5 sets of 20 manner. However, the compliance test is meant to evaluate the system’s compliance towards a given voice command. For example, if given command “Turn Right” the system is marked as successful if it performs a right turn, but mark as unsuccessful if it performs an incorrect movement.

5. Latency

In this test, the latency of the Wake Word Model and each command will be evaluated. Details of the tests can be found in Table V.

TABLE V. DETAILS FOR LATENCY TEST

Location	Quiet Controlled Room
Distance	Fixed 50 cm

In this latency evaluation, a timer will be used to time the response time for the system. Each test will be repeated 10 times, and the average will be recorded as the latency result.

III. Results and Discussions

A. *Detection Range (Controlled)*

The results of the detection accuracy versus distance are shown in Fig. 11. The wake word model was able to achieve around 90% accuracy when it was tested between the distances of 10 cm to 200 cm (user to microphone) in a controlled lab environment. For the voice command recognition, the accuracy is above 80% with the same test setup. The loudness of the input voice is as if greetings or commands are naturally given to another person.

B. *Accuracy in Simulated Environment*

For validation, the detection of the proposed wake word model is compared to Google Assistant. Different noise levels were generated by a speaker. From the results in Fig. 12, it is shown that the proposed model was able to perform relatively well (above 80% accuracy) under 55 dB of simulated park noise. It started to struggle once the noise amplitude level increased. It is noted that the wake word model performed better than Google as the model has been trained to specifically trigger on the “Hey SellTron” keyword.

Another test was conducted in a café environment and the result is shown in Fig. 13. Here, the wake word model could only detect the keyword with around 80% accuracy when the background noise is less than 50 dB. This showed that the model is more sensitive towards conversational noise. Google also showed its strength in voice detection in human background noise.

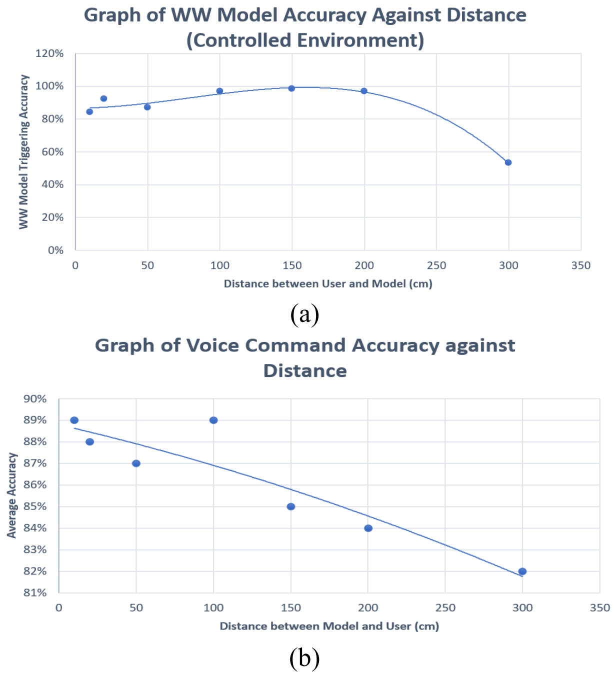

Fig. 11. Accuracy against distance for (a) the wake word and (b) the voice command.

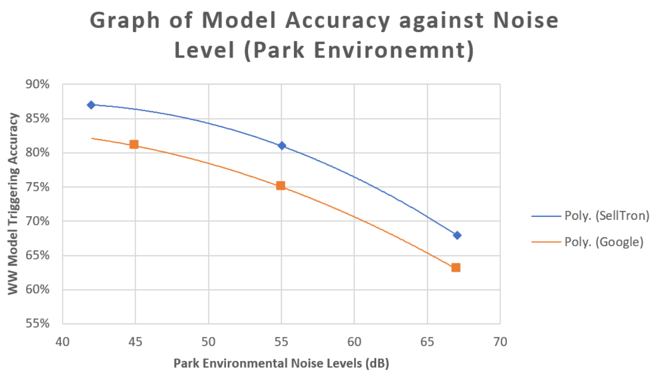

Fig. 12. Accuracy of wake word model vs google assistant in park environment.

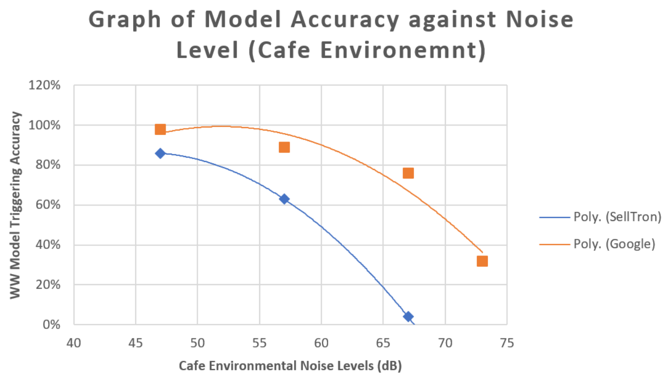

Fig. 13. Wake word model versus google assistant against cafe environment.

Moving on to speech recognition, the results of voice detection in noisy environments are shown in Fig. 14. The system performed well with above 80% accuracy when the background noise is below 65 dB. The speech recognition performed better than the wake word due to Google’s extensively trained model.

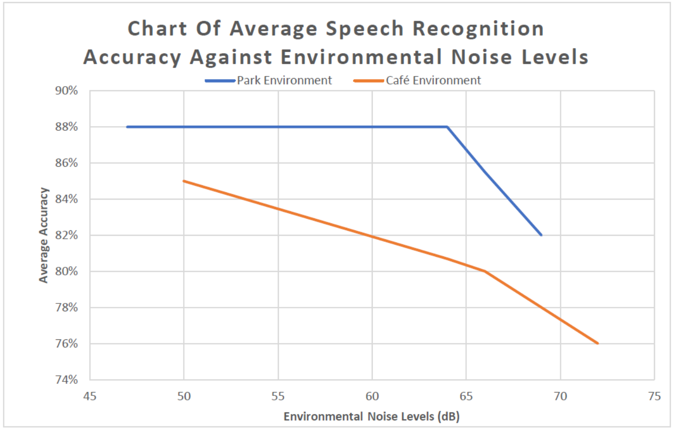

Fig. 14. Speech recognition accuracy in park and café environments.

C. *Accuracy for Different Individuals*

From the results in Table VI, it was found that the accuracy was not entirely consistent, as there were huge variations in accuracy between each individual. This is most probably because everyone had different voices, and there were various distinctions in terms of tone, pitch, and loudness for each person. Since the model was trained on only a small sample size of voices (highly personalized), it meant that not all variety of voices were accounted for. Hence, to improve the wake word model further, the voice dataset will have to be significantly expanded; and a balance between male and female datasets will also have to be kept.

TABLE VI. WAKE WORD MODEL ACCURACY FOR DIFFERENT INDIVIDUALS FOR 5 ATTEMPTS

Person #	Gender	Accuracy (%)
1	Male	20
2	Male	60
3	Male	100
4	Male	60
5	Male	80
6	Female	100
7	Female	60
8	Female	20
9	Female	40
10	Female	60

D. *Accuracy for Voice Commands and Compliances*

The accuracy of the voice commands is above 90%, with the only exception being the "turn left" command which was slightly lower at 89%, as shown in Fig. 15. The reason for its lower accuracy is because the system would misrecognize the word "turn left" as "turn light" during the inaccurate tries, which may be influenced by home automation modules for "turning on the lights". Once the commands were identified, the prototype would perform the task correctly without fail for any of the valid commands given.

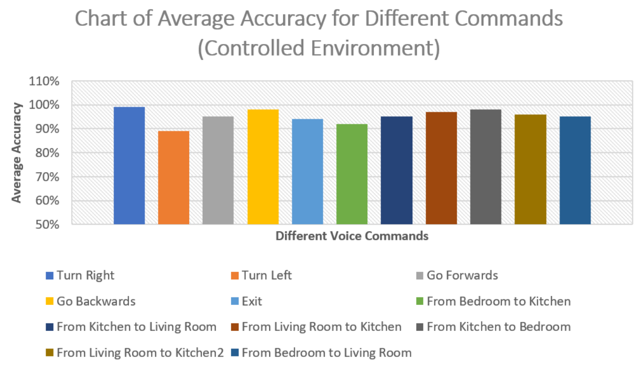

Fig. 15. Accuracy of different voice commands.

E. *Latency*

The wake word activation takes an average of 0.73 s whereby the voice commands averaged around 1.5 s as shown in Table VII and Fig. 16, respectively. These are acceptable response time for wheelchair application.

TABLE VII. WAKE WORLD LATENCY

Office	Average (s)	Min (s)	Max (s)
Response time	0.73	0.33	1.10

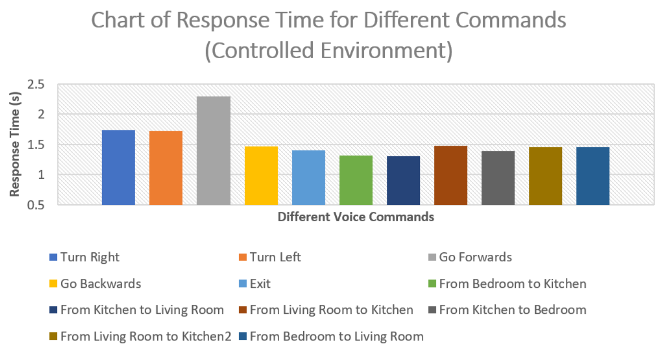

Fig. 16. Voice Command Latency

IV. FUTURE RECOMMENDATIONS

The voice controlled wheelchair is still a very untapped field; thus, this project and report will indubitably add to it. However, other than the implementations that had been demonstrated in this report, it is noted that there are several recommendations that will allow the project to be further elevated.

The first recommendation is to add sensors to the current Wheelchair project. The main sensors that are recommended will be the ultrasonic sensor and LiDAR sensor. This is because these sensors will allow for the implementation of obstacle avoidance features [14]. This is an important feature which was excluded from the present project due to time constraint, but it is a necessity for the Wheelchair if it were to be used commercially. Adding these features will drastically improve the safety of the wheelchair, and it will prevent countless collisions into objects or to walls. Additionally, LiDAR also offers the functionality of mapping out indoor rooms or home interiors [15]. The is one of the most important features that has to be added if the locational voice command concept were to be implemented. This is because this function would allow the Wheelchair to understand where exactly each room or area would be located at; thus, minimizing possible the automation errors that may occur.

Aside from that, it is also recommended that the motor be changed for future implementations, as the one used in this project was only for proof of concept. In a commercial situation, the wheelchair would need to have utmost precision so that accidents do not occur, and the motors will also need to be able to handle the human weight. Other than that, it is also recommended the Wake Word Model be further trained with larger datasets, so that it will be able to understand the Wake Word even when it is said with an entirely different accent. Many large corporations have proven that an accurate Wake Word model is possible as long as enough dataset and computational power is prepared. Presently, a possible candidate model to consider for the future would be OpenAI's whisper speech recognition model, as it was successfully implemented on a Raspberry Pi Model B by Wechsler and was able to perform real-time speech recognition and transcription [16].

V. Conclusion

The overall performance showed the feasibility of implementing both the wake word and speech recognition features on an edge device. For the wake word model, tests showed that it has a detection accuracy of slightly above 80% when the background noise is less than 50 dB. For the speech recognition, the system has above 80% accuracy when the background noise is below 65 dB. Once the command has been identified, the system registers a 100% accuracy for command compliance. This case study would also apply to the literature of the design and development of voice-controlled wheelchairs or similar applications.

Acknowledgment

Our highest gratitude to Motorola Solutions and Career Services of INTI International College Penang for facilitating and supporting this industrial employer project.

References

[1] L. Bennett, “Maintaining arm health in wheelchair users: The need for updated guidelines,” 2015. icord.org. https://icord.org/2015/02/maintaining-arm-health-in-wheelchair-users-the-need-for-updated-guidelines/ (accessed Dec. 2, 2022).

[2] NCHPAD, “Overuse Injuries in Wheelchair Users,” 2013 NCHPAD.org. https://www.nchpad.org/96/713/Overuse~Injuries~in~Wheelchair~Users (accessed Dec. 2, 2022).

[3] V. Khare, K. Meena, and S. Gupta, “Voice Controlled Wheelchair,” *International Journal of Electronics*, *Electrical and Computational System*, pp. 23-27, 2017.

[4] S. Umchid, P. Limhaprasert, S. Chumsoongnern, T. Petthong, and T. Leeudomwong, “Voice Controlled Automatic Wheelchair,” in *The 2018 11th Biomedical Engineering International Conference*, 2018. doi: 10.1109/BMEiCON.2018.8609955.

[5] T. K. Hou, Yagasena, and Chelladurai, “Arduino based voice-controlled wheelchair," *Journal of Physics: Conference Series*, pp. 1-6, 2019.

[6] M. I. Malik, T. Bashir, and O. F. Khan, "Voice-Controlled WheelChair System,” *International Journal of Computer Science and Mobile Computing*, pp. 411-419, 2017.

[7] S. Anwer et al., “Eye and Voice-Controlled Human Machine Interface System for Wheelchairs Using Image Gradient Approach,” *Sensors*, pp. 1-13, 2020.

[8] M. A. A. Rakib et al., “Smart Wheelchair with Voice Control for Physically Challenged People,” *European Journal of Engineering and Technology Research*, pp. 97-102, 2021.

[9] S. Sutikno, K. Anam, and A. Saleh, "Voice Controlled Wheelchair for Disabled Patients based on CNN and LSTM," in *4th International Conference on Informatics and Computational Sciences (ICICoS)*, 2020. doi: 10.1109/ICICoS51170.2020.9299007.

[10] *C. Olson*, “New report tackles tough questions on voice and AI,” 2019. Microsoft.com. https://about.ads.microsoft.com/en-us/blog/post/april-2019/new-report-tackles-tough-questions-on-voice-and-ai (accessed Dec. 2, 2022).

[11] M. Phi, “Illustrated Guide to LSTM’s and GRU’s: A step by step explanation,” 2018. towardsdatascience.com. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21. (accessed Dec. 2, 2022).

[12] S. Dobilas, “LSTM Recurrent Neural Networks — How to Teach a Network to Remember the Past,” 2022. towardsdatascience.com . https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e. (accessed Dec. 2, 2022).

[13] C. Yayda and W. Hrauda, “Wake-Up Word Detection Using LSTM Neural Networks,” Master Thesis, Graz University of Technology, 2016.

[14] OTLASERS, "Ultrasonic and Lidar Sensors for Robot Obstacle Avoidance," Otla Sers Magazie, 17 July 2023. [Online]. Available: https://otlasers.com/ultrasonic-and-lidar-sensors-for-robot-obstacle-avoidance/#:~:text=Both%20ultrasonic%20and%20lidar%20sensors%20play%20important%20roles,detection%20and%20the%20ability%20to%20detect%20specific%20materials.. (accessed 16 February 2024).

[15] R. Ziccardi, "The Complete Guide To LIDAR Indoor Mapping," Maptelligent, 15 May 2023. [Online]. Available: https://maptelligent.com/blog/the-complete-guide-to-lidar-indoor-mapping/. (accessed 18 February 2024).

[16] S. Wechsler, "Whisper in Real Time on Raspberry Pi: Run OpenAI's C++ Model," Toolify.ai, 23 November 2023. [Online]. Available: https://www.toolify.ai/ai-news/whisper-in-real-time-on-raspberry-pi-run-openais-c-model-84609. [Accessed 16 February 2024].

[17] N. MR, S. T, S. M and S. MN, "Real Time Speech Emotion Recognition using LSTM and Raspberry Pi," in *International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*, Bangalore, 2023.

[18] M. Bakouri, "Development of Voice Control Algorithm for Robotic Wheelchair Using NIN and LSTM Models," *Computers, Materials & Continua 2022,* vol. 73, no. 2, pp. 2441-2456, 2022.

[19] P. Fromaget, "Raspberry Pi Pros And Cons," RaspberryTips, [Online]. Available: https://raspberrytips.com/raspberry-pi-pros-and-cons/. (accessed 16 February 2024).

[20] A. Harsha, "The Ultimate Showdown: RNN vs LSTM vs GRU – Which is the Best?," Shiksha Online, 22 May 2023. [Online]. Available: https://www.shiksha.com/online-courses/articles/rnn-vs-gru-vs-lstm/#:~:text=RNNs%2C%20LSTMs%2C%20and%20GRUs%20are,of%20memory%20cell%20and%20gates. (accessed 17 February 2024).

[21] S. Saxena, "What is LSTM? Introduction to Long Short-Term Memory," Analytics Vidhya, 4 January 2024. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/. (accessed 17 February 2024).

[22] K. Supriya, A. Divya, B. Vinodkumar and G. R. Sai, "Trigger Word Recognition using LSTM," *International Journal of Engineering Research & Technology (IJERT),* vol. 9, no. 6, pp. 1-8, 2020.