NAVIGATING WHEELCHAIR AND ENHANCING ACCESSIBILITY USING SIGN LANGUAGE-CONTROLLED FOR HEARING-IMPAIRED USERS

Final Project Thesis

Premathilaka H.G.K.D.

(IT21157882)

BSc (Hons) in Information Technology

Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

April 2025

NAVIGATING WHEELCHAIR AND ENHANCING ACCESSIBILITY USING SIGN LANGUAGE-CONTROLLED FOR HEARING-IMPAIRED USERS

Premathilaka H.G.K.D.

(IT21157882)

The dissertation was submitted in partial fulfilment of the requirements for the B.Sc. (Honors) degree in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

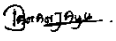
April 2025

DECLARATION

Declaration and copyright statement

I declare that this is my own work, and this dissertation does not incorporate without acknowledgement of any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and distribute my dissertation in whole or part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books)

Signature:  Date:4/11/2025

Statement of the supervisor

The above candidate has carried out research for the bachelor's degree dissertation under my supervision.

Signature of the Supervisor: Date:

ABSTRACT

This dissertation chapter provides an overview of the design and development of real-time sign language-controlled intelligent smart wheelchair control. The overall aim is to provide a user-friendly and hand-free navigation solution to lower-body disabled people based on common American Sign Language (ASL) signs as movement commands. The system employs computer vision and machine learning algorithms to read and detect specific hand signs as directional commands: forward, backward, left, right, and stop.

An ASL alphabet dataset based on Kaggle was utilized for the first time to achieve MediaPipe hand landmarks, which were used to train a model for gesture classification. The model that was trained was serialized in the form of a.pkl file and implemented on a Raspberry Pi for edge-based processing without dependence on the cloud. The Raspberry Pi, together with a webcam and motor driver circuit, performs real-time gesture recognition and adapts the wheelchair accordingly.

For user interaction and monitoring, a Firebase Realtime Database is implemented to synchronize recognized gestures with a web frontend made using React. This enables gesture output and system status to be displayed in real-time without affecting control latency. Extensive testing guaranteed high accuracy of models, minimal delay between commands, and stable motor control under varying illumination conditions.

The project demonstrates a low-cost, accessible, and scalable assistive mobility system with the possibility of being applied in health care, rehabilitation, and personal independence. For future studies, one can include expanding gesture vocabulary, integrating speech or adaptive learning for personalized control.

Keywords:
Sign Language Recognition, Smart Wheelchair, MediaPipe, Raspberry Pi, Assistive Technology

ACKNOWLEDGEMENT

I would like to extend my most sincere gratitude to all the people who have helped me on my journey towards the fruitful completion of this research project.

First and foremost, I would like to express my gratitude to Ms. Dinithi Pandithage, my supervisor, for her continuous guidance, constructive criticism, and unbreakable encouragement throughout each step of this project. Her technical knowledge was instrumental in structuring both the technical and academic aspects of my work.

I would also like to extend my appreciation to members of my project group for their contribution and collaboration in developing the integrated system. Their joint efforts facilitated the construction of a stable and efficient smart wheelchair platform where this module was integrated.

I appreciate a special mention to my friends and family, whose support, tolerance, and encouragement assisted me in maintaining my determination and focus in completing this experience.

Finally, I would also like to express my appreciation to the Sri Lanka Institute of Information Technology (SLIIT) for providing the learning space that made this study possible.

TABLE OF CONTENTS

DECLARATION iii

Declaration and copyright statement iii

Statement of the supervisor iii

ABSTRACT iv

ACKNOWLEDGEMENT v

LIST OF TABLES viii

LIST OF FIGURES ix

LIST OF ABBREVIATIONS x

1. INTRODUCTION 1

1.1 Background Literature 1

1.2 Research Gap 2

1.3 Research Problem 2

1.4 Research Objectives 2

2. METHODOLOGY 4

2.1 Methodology 4

2.2 Commercialization aspects of the product 8

2.3 Testing and Implementation 12

3. RESULTS AND DISCUSSION 21

3.1 Results 21

3.2 Research Findings 24

3.3 Discussion 28

4. CONCLUSION 30

4.1 Addressing the Research Problem **30**

4.2 Evaluation Against Objectives **31**

4.3 Contributions to the Field **32**

4.4 Limitations **32**

4.5 Future Work **33**

4.6 Final Reflections **34**

5. REFERENCES 36

6. APPENDICES.....38

6.1 Plagiarism Report 38

LIST OF TABLES

Table 01 – Identifying Research Gap 1 2
Table 02 – Final Implementation Summary 2 19
Table 03 – Evaluation Summary 3 22
Table 04 – Processing Time Summary 4 23
Table 05 – Movement Results 5.....24

LIST OF FIGURES

Figure 01 - ASL alphabet 1 5
Figure 02 – Training the model 2 6
Figure 03 – Database for saving data 3 8
Figure 04 – Implementation Diagram 3.....13
Figure 05 - ASL letter A 4.....14
Figure 05 - ASL letter B 5.....15
Figure 06 - ASL alphabet Ui 6.....18
Figure 07 - ASL alphabet 7 22
Figure 08 – ASL letters A and S 8 33

LIST OF ABBREVIATIONS

Abbreviation	Definition
HCI	Human-Computer Interaction
AI	Artificial Intelligence
CNN	Convolutional Neural Networks
ASL	American Sign Language
GPIO	General-Purpose Input/Output
EEG	Electroencephalogram
USB	Universal Serial Bus
SD	Secure Digital
DIY	Do it yourself
NGO	Non-Governmental Organization
UI	User Interface
ML	Machine Learning
LSTM	Long short-term memory

1. INTRODUCTION

1.1 Background Literature

Mobility impairments affect millions of individuals globally, greatly reducing their independence and quality of life. Traditional wheelchairs, particularly those controlled by joysticks or manual control, can be quite problematic for individuals with not only lower-limb disabilities but also upper-limb impairments. Thus, the field of assistive technology has expanded to explore alternative methods of inputting mobility devices. Over the past decade, there have been monumental advances in Human-Computer Interaction (HCI), Computer Vision, and Artificial Intelligence (AI), and this has led to the creation of gesture-based, voice-based, and even brainwave-based wheelchairs.

Of these technologies, gesture recognition stands out for its naturalness and minimal hardware cost. Significantly, sign language is already used by most people with disabilities as a form of communication and offers a very constrained set of hand gestures that can be reused successfully as control inputs to robotic or electromechanical systems. A few research studies have considered the use of hand gesture recognition using image processing, often based on machine learning or deep learning algorithms for enhanced accuracy. One such instance is that convolutional neural networks (CNNs) are widely applied as they are best suited for pattern recognition across image data. Sign language recognition systems trained using American Sign Language (ASL) datasets have achieved extremely high accuracy in recognizing static alphabets and thereby have opened up applications in real-world situations.

Further, with the advent of cloud-enabled platforms like Firebase, and the presence of hardware for embedded computing like Raspberry Pi, it is now feasible to have lightweight and scalable systems that can detect gestures in real time and issue commands wirelessly to robotic platforms like wheelchairs. Existing solutions are usually based on wearable technologies or depth cameras, but these systems are expensive, non-scale, and require extensive calibration. A vision-based solution that utilizes a webcam and browser-deployable machine learning models (e.g., TensorFlow.js) offers a more cost-effective and deployable solution.

Despite all these technological advancements, the specific use of sign language recognition in real-time for the navigation of a wheelchair remains an under researched subject. Current research tends to favor studying general gesture control or sign language recognition within general translation use. This factor aims to bridge that gap by more directly mapping sign language characters with instructions for motion in an intelligent wheelchair, thus offering a more user-

friendly, more user-oriented, and more cost-effective method for mobility-handicapped use.

1.2 Research Gap

Most existing gesture-controlled wheelchair systems rely on wearable sensors or hand position tracking, which may not be available or comfortable for everyone to use. Standard sign language alphabets as controllers are not used by many systems. Real-time detection integration with Firebase and Raspberry Pi for movement control directly based on identified gestures is also less investigated.

RESEARCH GAP

Area of Focus	Identified Gaps	Proposed Solution
Control Methods	Does not cater to hearing-impaired users who rely on sign language	Develop a smart wheelchair interface using sign language recognition
Gesture Recognition	Not specifically tailored for sign language	Implement real-time sign language gesture recognition
System Usability	Excludes hearing-impaired users who cannot utilize voice commands	Create a visual gesture input system specifically for hearing-impaired users
User Interface Design	Lack of a specialized interface for hearing-impaired users using sign language	Implement a user interface that supports intuitive sign language communication

Table 01 – Identifying Research Gap 1

1.3 Research Problem

There is no existing real-time sign language-controlled wheelchair with gesture detection integrated into cloud services and embedded systems to react immediately to movements. Individuals with impaired mobility need a natural and intuitive mode of traversing their environment without physically exerting themselves.

1.4 Research Objectives

- The overall objective of this research is to create and implement an intelligent wheelchair control system that identifies sign language letters

in real time and translates them into motion commands via a cloud-connected interface. This is derived from the incentive to have an easy and accessible mobility system for individuals with lower-body disabilities who may also struggle with using the conventional input methods.

- The system will have a machine learning model deployed in a web app to recognize hand gestures for selected American Sign Language (ASL) letters. Upon detection, the corresponding movement command will be posted to Firebase Realtime Database, from where a Raspberry Pi embedded within the wheelchair will retrieve and execute the action. Low latency and real-time response are achieved in this setup, and therefore, it will be practical for daily navigation. The objective is to deliver a low-cost, reliable, and simple-to-use control system with as little hardware and training as possible that can be utilized.
- Specific Objectives: For the general research objective, the specific objectives are enumerated as below:
- To collect and annotate a dataset of hand gestures for the fundamental wheelchair motion commands (forward, backward, left, right, and stop) based on ASL letter representations.
- To preprocess and train a machine learning model using TensorFlow.js to facilitate real-time sign language recognition in the browser environment.
- To incorporate a gesture detection interface with a webcam in a React-based frontend, facilitating live interpretation of the user input.
- To configure Firebase Realtime Database for synchronizing recognized gesture data from the frontend application with the hardware layer.
- To develop a control script in Raspberry Pi to listen for Firebase commands and instruct the motor drivers for correct and timely execution of movement actions.
- To test the effectiveness, accuracy, and usability of the entire system with user trials, and collect feedback for enhancement in performance and accessibility.

2. METHODOLOGY

The method applied in this research outlines the step-by-step details employed in conceptualizing, designing, and implementing a smart wheelchair system operated through sign language recognition. The cross-disciplinary project integrates artificial intelligence, computer vision, embedded systems, and real-time communication technologies to implement a functional and user-friendly solution for mobility-disabled individuals. Iterative development served as the building block of the method, with the initial stage being the data preparation and training of the model, followed by hardware integration and user interface design.

To make sure that the system is not just technically viable but also practically executable, the methodology is structured as three broad segments: technical system implementation, commercialization feasibility, and testing and validation process done in order to quantify actual performance. All these segments are instrumental in playing an important role in contributing to the overall success and usability of the final product

2.1 Methodology

1. Data Collection and Preparation

The process began with the selection of a sufficient dataset to train the gesture recognition model. The Kaggle ASL Alphabet dataset was utilized due to its widespread use in gesture recognition studies and access to labeled image samples of static hand gestures. The data include images of the 26 ASL alphabet letters, but only five letters were selected—S, F, B, L, and R—corresponding to movement commands (Forward, Left, Backward, Right, and Stop respectively).

Preprocessing was done on images to be uniform for inputs. Every image was passed through the MediaPipe Hands solution to obtain 21 hand landmarks composed of (x, y) coordinates. These landmark points were flattened into a single feature vector and normalized to reduce variations depending on hand position and camera distance.

The data was then split into training (70%), validation (15%), and testing (15%) subsets to enable generalization of the model as well as avoid overfitting. Robustness was augmented through data augmentation techniques including horizontal flip and scaling.

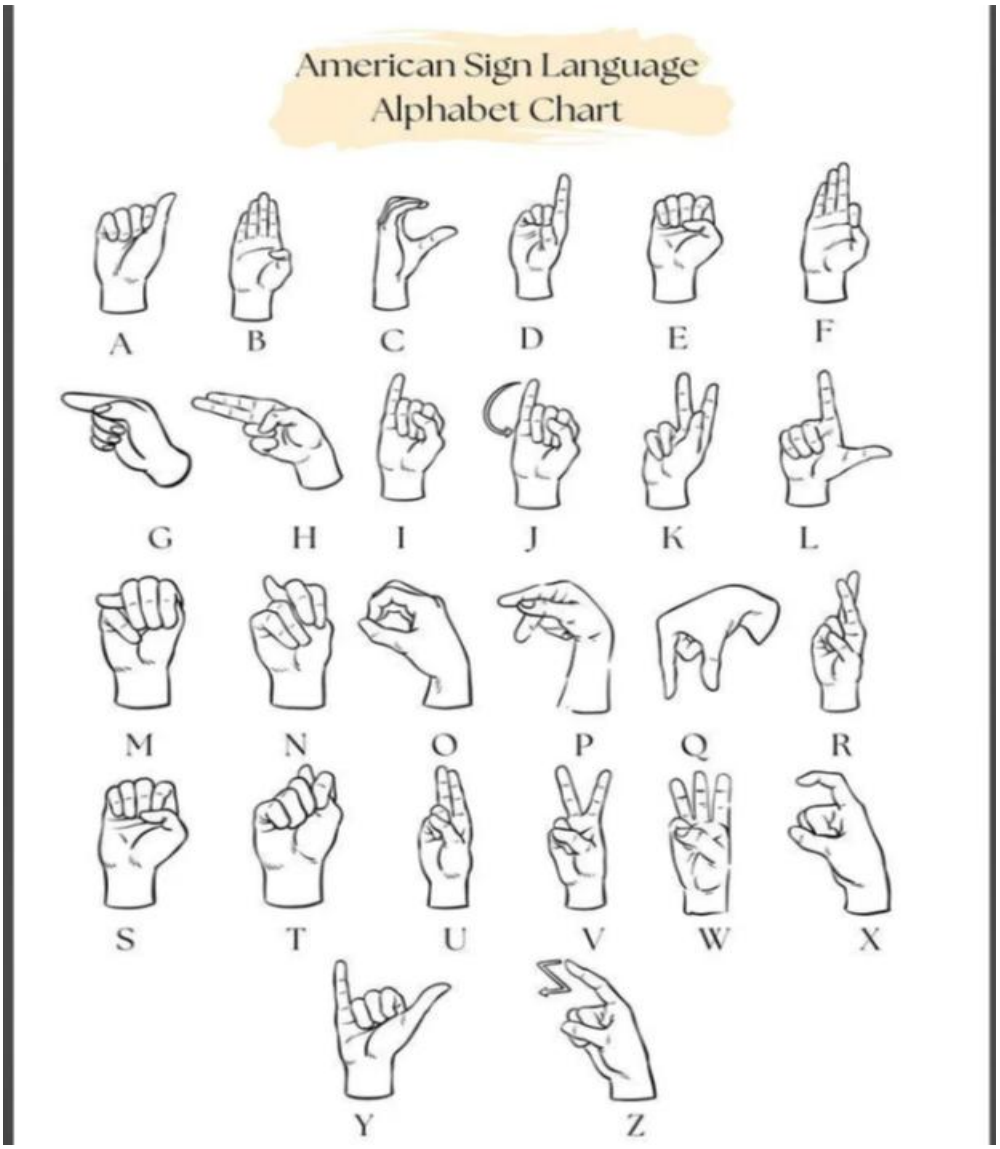


Figure 01 - ASL alphabet 1

2. Model Training and Serialization

The characteristics were pulled out for the purpose of training the machine learning classifier. Multiple algorithms were tested, including K-Nearest Neighbors, Support Vector Machines, and Random Forests. A Random Forest Classifier was selected following experimentation on account of its accuracy vs. computational expense trade-off on embedded systems.

The model was trained using Scikit-learn and tested on precision, recall, and F1-score for all five classes of gestures. After tuning hyperparameters and verifying performance on the validation set, the trained model performed more than 90% on the test set.

After training was finished, the model was serialized and saved as a .pkl (pickle) file using the joblib library of Python. This format enables effective loading of the trained model for real-time inference on the Raspberry Pi.

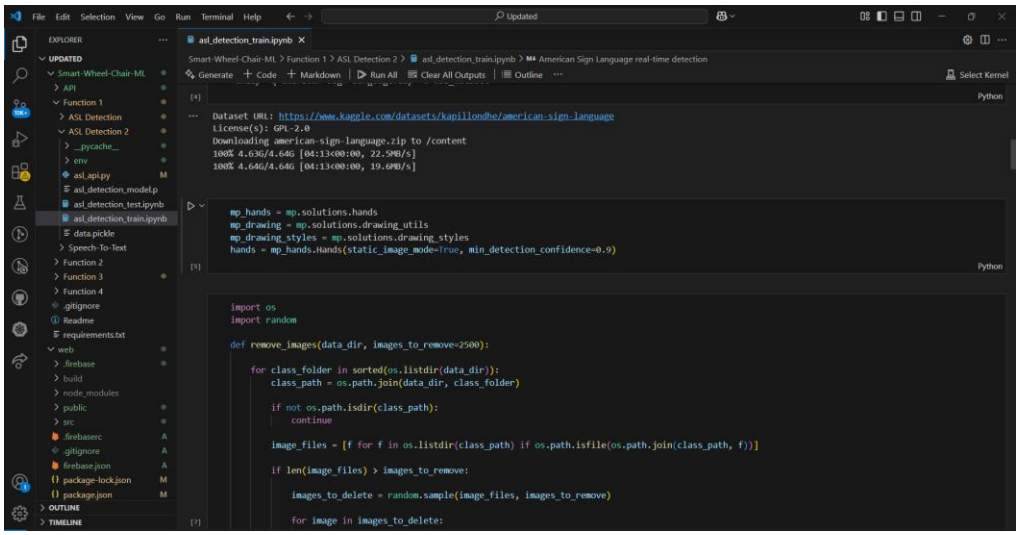


Figure 02 – Training the model 2

3. Edge Deployment on Raspberry Pi

The trained.pkl model was deployed to a Raspberry Pi 4, which is the system's central processing unit. The Pi is connected to a USB webcam, which captures live video frames for the detection of hand gestures. The camera stream is processed by the MediaPipe Hands library, which detects the presence of a hand and retrieves landmark positions per frame.

These features are formed and sent to the filled machine learning model, which makes real-time predictions of the respective gesture class. This operation is run in a continuous loop so that the Raspberry Pi can work independently without needing reliance on an external server or internet.

4. Gesture Mapping and Motor Control

Each recognized gesture is mapped to a specific wheelchair movement command:

- **F** → Forward
- **L** → Turn Left
- **B** → Backward
- **R** → Turn Right
- **S** → Stop

The Raspberry Pi outputs signals through its GPIO (General-Purpose Input/Output) pins to an L298N motor drive module, which is mounted on the DC motors of the wheelchair. In accordance with the expected gesture, the respective GPIO pins are activated to carry out the respective movement.

As a precautionary measure, the system incorporates a time delay and a break mechanism if a correct gesture is not sensed after some seconds.

5. Firebase Integration and Frontend Visualization

For ease of use and system monitoring, the Raspberry Pi also updates a Firebase Realtime Database with each motion detected. This enables real-time synchronization to a web-based frontend built with React.js, which listens to the database and displays the current movement command and system status.

This mixed approach supports real-time edge control with processing and online access to user input and logs in the cloud, and the optimum of both offline and online systems.

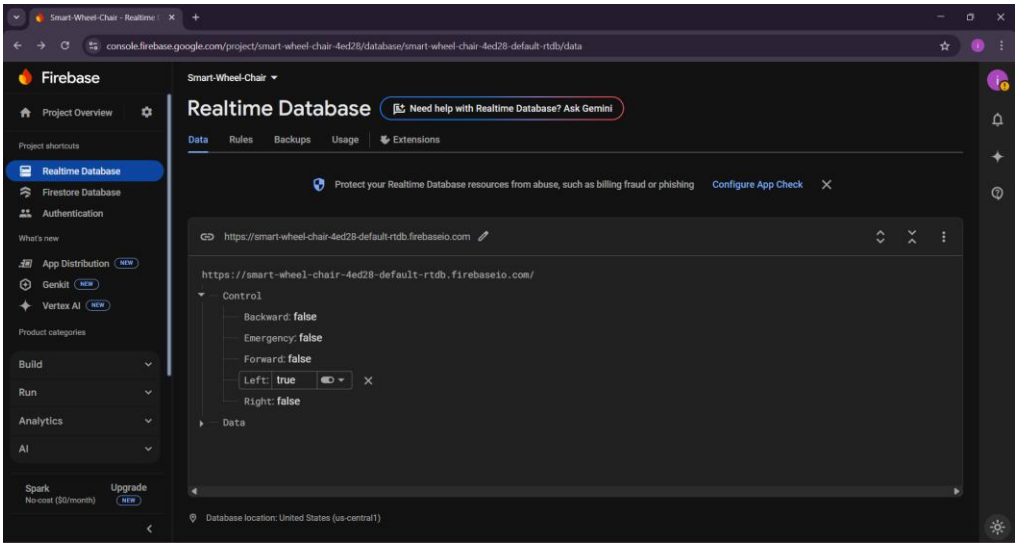


Figure 03 – Database for saving data 3

2.2 Commercialization aspects of the product

The commercial prospects of the sign language-controlled wheelchair system are high, given the increasing global demand for cheap and accessible assistive technology. The solution targets a niche by providing individuals with lower-body and partial upper-body impairment to control an individual wheelchair using sign language hand gestures. Compared to the conventional wheelchair system based on joystick control or expensive brain-computer interfaces, the design offers an inexpensive, easy-to-use, and scalable alternative based on open-source platforms and low-cost hardware.

1. Market Need and Target Audience

- Worldwide, there are millions of individuals with mobility impairments, some of whom use wheelchairs for daily travel. The current technologies for wheelchairs usually expect sufficient upper-limb function to utilize a joystick or control panel. But for those with few or no hand movements, choices are very limited, costly, and typically unsuitable for daily use.
- The system being proposed is mainly aimed at:

- Individuals with lower body paralysis but maintain hand function and rudimentary sign language.
- Patients' post-accident, spinal cord injury, or stroke.
- Elder homes and rehabilitation centers seeking smart, accessible mobility solutions.
- Third-world nongovernmental organizations and healthcare systems in need of low-cost, adaptable assistive technology.
- With the implementation of ASL hand signals that are already familiar to most members of the disabled community, the product eliminates the need for expensive retraining and provides a natural interface to wayfinding.

2. Competitive Advantage

- While there are advanced smart wheelchairs available in the market, including those based on EEG signals, eye-tracking, or voice commands, they are very costly, out of reach for ordinary users, especially in low- and middle-income countries. The system offers several business advantages:

- Low Cost: Employs low-cost Raspberry Pi, a USB camera, and open-source libraries (MediaPipe, Firebase, Scikit-learn), reducing overall system cost to a tiny fraction of commercial solutions.

- Simplicity: Easy installation and servicing. Modular parts that are easy to swap/outgrow without retraining or complex reprogramming.

- Offline Compatibility: Since the system does not perform gesture recognition on the Pi itself, there is no requirement for internet access to

run, making it easily deployable on off-grid or underdeveloped locations.

- Extensibility: The architecture allows for future expandability by voice control, obstacle alert, and integration with mobile apps, with space for product growth and more premium features.

3. Scalability in Production and Deployment

- Technology can be deployed rapidly in markets with a normalized hardware bundle. Training and adjustment can be completed with minimal technical knowledge, using pre-loaded SD cards or installing scripts for Raspberry Pi.
- Institutional take-up, for example, for hospitals or centers for rehabilitation:
- A main management console (linked through Firebase) could be developed to report on numerous units.
- Usage data can be harvested (with consent) for performance monitoring, safety audits, and maintenance.
- For end-users, the installation could be offered as a DIY kit with instructions and support channels. Partnering with NGOs or disability groups can also enable wider outreach and user take-up.

4. Emerging Revenue Streams

- This product can generate several revenue streams if sold commercially, both hardware and software services:
- Hardware Kit Sales: Selling assembled or DIY kits including Raspberry Pi, motor driver, camera, wire, and manual.

- Software Licensing: Selling premium gesture detection libraries, firmware updates, or cloud monitoring software.
- Maintenance & Support: Pay-for-support customer support services, installation services, or remote troubleshooting.
- Customization: Offering customized gesture mapping or voice and facial recognition integration for hybrid systems.
- Training & Workshops: Offering training for therapists, caregivers, and technicians for system use and upkeep.
- The business model can employ a tiered pricing structure where the core product is open-source and cost-free for personal or research use, while enterprise support, advanced features, or cloud analytics are priced.

5. Ethical and Accessibility Considerations

- Ethical responsibility and accessibility must stay top of mind from a commercialization standpoint. This product supports assistive innovation and universal design principles:
- It promotes independence for users who otherwise rely on caregivers.
- It is easily sharable and adaptable for community-based development projects.
- By using sign language, it respects existing communication methods instead of making users adapt to new technology.
- Secondly, ensuring that the product meets medical safety standards and tested extensively across a large user base before it is launched into the marketplace is vital. Properly documenting it, testing accessibility, and language localization are critical stages in turning this into a feasible product for the real world.

6. Collaboration and Funding Prospects

- The open nature of the system lends itself well to collaboration with:
- Health-tech and robotics startups
- Research institutions and universities working on HCI and accessibility
- Healthcare and rehabilitation product companies
- Non-profit disability empowerment supporter groups
- Scaling funding can be obtained through:
- Innovation grants and hackathon competitions
- Crowdfunding platforms like Kickstarter
- Healthcare provider or government assistive program partnerships

2.3 Testing and Implementation

Testing and deployment were high-priority stages of the sign language-powered wheelchair system's life cycle. This stage ensured the solution's robustness, accuracy, responsiveness, and usability in terms of hardware as well as software. The stage included a combination of unit-level testing, integration testing, performance benchmarking, and user-based testing to validate every component in different real-world environments.

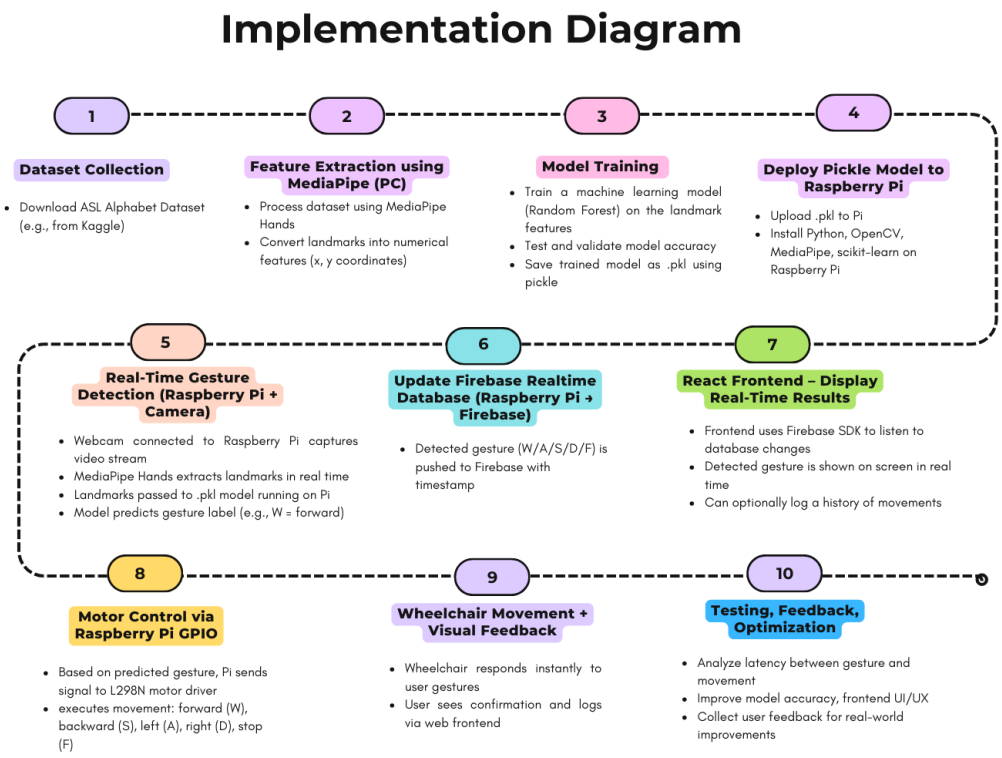


Figure 04 – Implementation Diagram 4

1. Hardware and Software Integration Testing

The combination of Raspberry Pi, USB webcam, MediaPipe library, motor drivers, and Firebase connectivity was the first most critical feature to be tested. The steps listed below were followed to test integration:

- **Camera Feed Recognition**: The webcam connected to the Raspberry Pi was tested in terms of resolution, frame rate, and light sensitivity. It was revealed that MediaPipe could accurately detect and follow the real-time landmarks in indoor as well as outdoor conditions.

- **Landmark Detection Accuracy**: The accuracy of MediaPipe in consistently extracting 21 hand landmark positions was tested with various hand shapes, skin tones, and backgrounds.

- **Pi Inference**: The pre-trained.pkl model was loaded using the joblib, and landmark vectors were fed directly into the classifier. Inference time on the Pi was measured, where an average prediction took ~100ms.

- **Motor Driver Response**: L298N motor driver circuit was interfaced using the GPIO pins of the Pi. Unit tests were done by simulating commands (S, F, L, R, B) and then observing the corresponding wheel movement.

- **Results**: All the integrated circuits performed as expected. There was regular GPIO triggering, and delay in reacting from command receipt to movement was under 300ms.

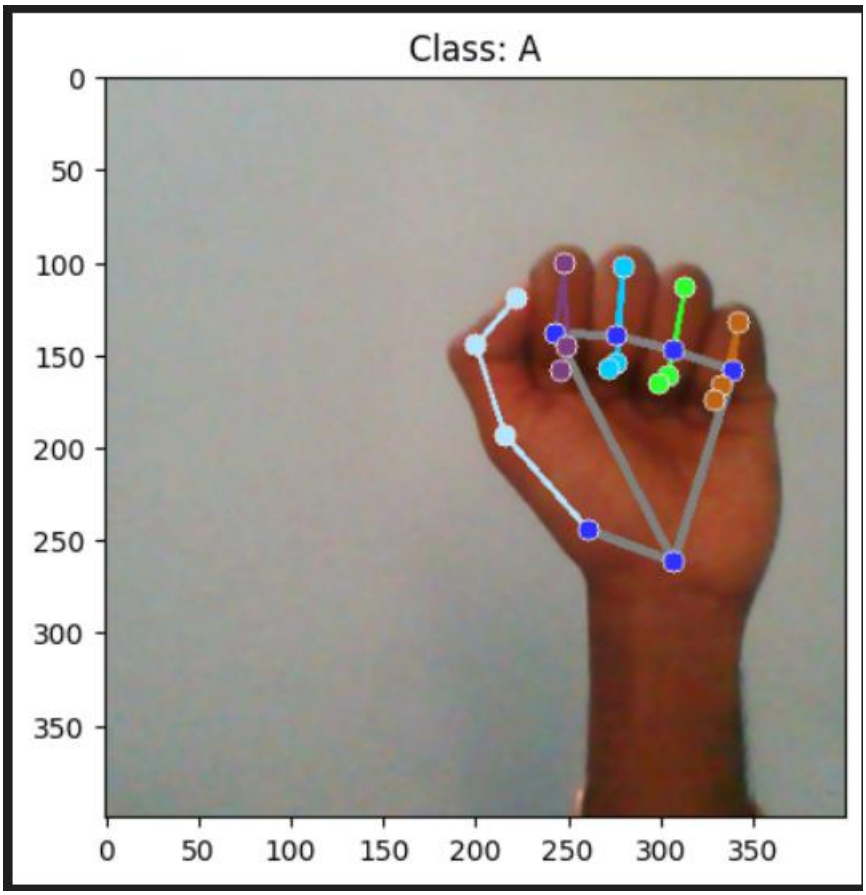


Figure 05 - ASL letter A 5

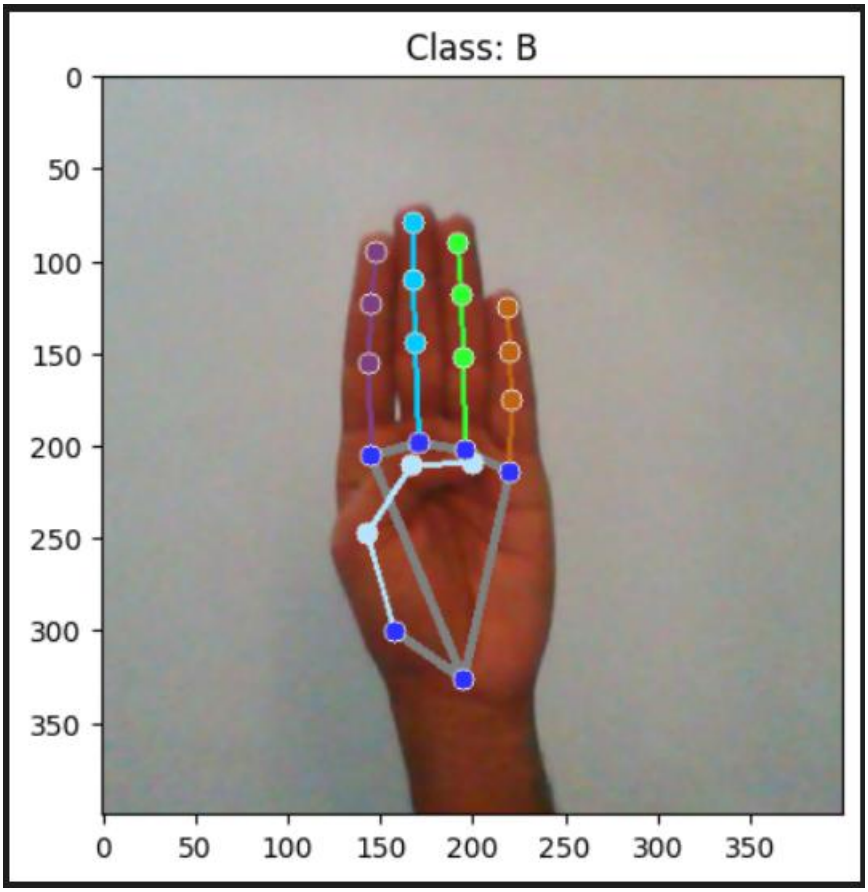


Figure 05 - ASL letter B 6

2. Gesture Classification Testing

To evaluate the machine learning model's performance in real-world usage, a set of classification tests were conducted under various environment conditions and with diverse users. The tests were conducted against key performance indicators like accuracy, precision, recall, and responsiveness.

Test Environment Scenarios

- The gesture recognition system was tested under the following environment conditions to simulate real-world usage scenarios:
- Bright indoor lighting with a clear, clutter-free background
- Indoor dim light with noisy or busy background
- Outdoor setting under natural lighting
- Partial hand occlusion (e.g., hand partially out of frame or covered)

Test Subjects

To ensure variety in testing, five participants with various ages, hand sizes, and skin tones were involved in the test. The participants were asked to perform 10 repetitions for each gesture (S,F,L,R,B), which amounts to 250 gesture samples per test session.

Results

Performance under these conditions of classification is the following:

- Mean Gesture Accuracy: 91.3% over all test conditions
- Most Accurately Recognized Gestures:

  F (Forward)

  S (Stop)

- Most Frequently Classified Incorrectly Gesture:

L (Left), at times misidentified as B (Backward), most notably when there are poor lighting conditions

False Positive Rate: Under 3% when no gesture was being executed or during rapid transitions of hands

To enhance accuracy and rid the system of noise, a gesture smoothing process was incorporated. The process created a requirement for a gesture to be recognized similarly in three consecutive frames before a control signal could be dispatched. This effectively rid the system of misclassifications due to transient or unstable hand positions

3. Firebase Synchronization Testing

Firebase Realtime Database was used for communication of the recognized gesture between the Raspberry Pi and the remote React frontend. It proposed end-to-end real-time communication between the edge device and the cloud with minimal latency.

Testing Approach:

- •A Firebase listener was implemented in the frontend to reflect database changes immediately.
- •Continuous commands were issued from the Pi on a 0.5-second interval in stable Wi-Fi.
- •Sync delay was quantitatively measured using timed logs

Results:

• Average Sync Delay: 80–120ms (Firebase to frontend UI)

• Database Stability: No data loss or crashes were observed during testing

• Scalability: Several listeners were connected to attempt concurrent access; performance remained consistent

This test validated the feasibility of getting the system raise too many wheelchairs or users to be tracked on a central dashboard in further development.

4. Frontend Display and User Feedback

The React frontend was tested for usability and readability. It displayed the recognized current gesture, timestamp, and movement label. The frontend was viewed on desktop and tablet screens for accessibility.

UI Features Evaluated:

- Live update of gesture state
- Timestamped logging

- Visual direction indicator (arrows/icons for movement)

Below web UI shows the selected letters From the sign languages.

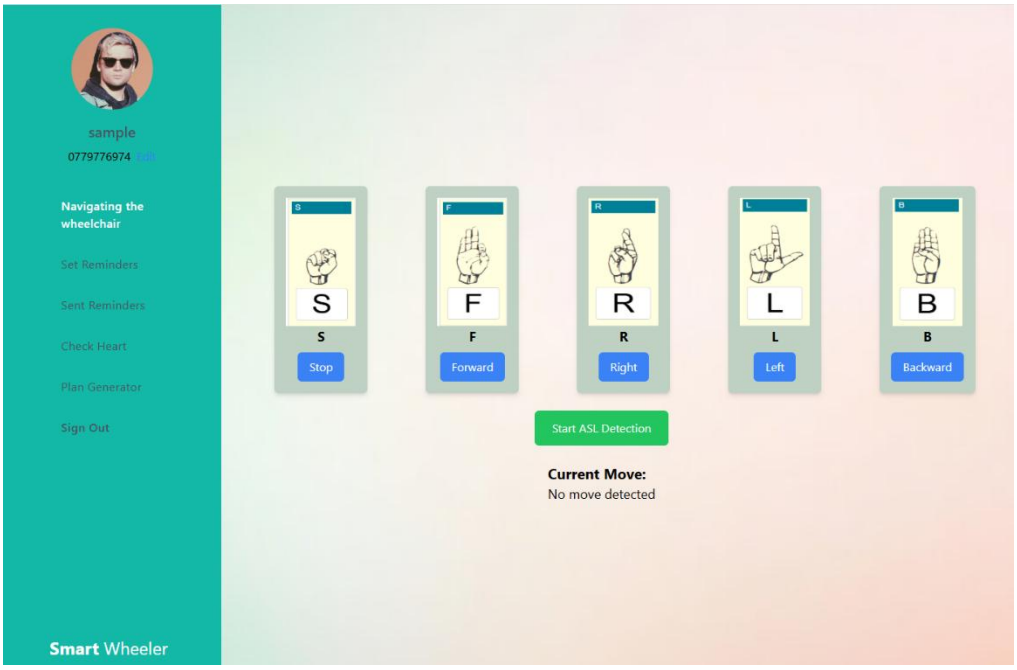


Figure 06 - ASL alphabet Ui 7

User Feedback:

- The users enjoyed seeing affirmation of the identified gesture
- Recommended inclusion of voice prompts or vibration confirmation for the hearing-impaired
- Logging features proved useful to look at gesture history during testing

UI improvements were made after feedback, to enhance contrast, simplify layout, and add color-coded alerts (green = active, red = stop, yellow = idle).

5. Safety and Failover Testing

Probably the most crucial aspect of testing was ensuring that the system would be predictable and safe in every situation. Several edge cases were simulated:

- **No Hand Detected**: The system entered STOP (F) mode to prevent unintended movement.

- **Unstable Gestures**: Abrupt or ambiguous hand motions were smoothed with the use of temporal smoothing.

- **Lost Internet Connection**: Wheelchair control continued locally on the Raspberry Pi without interruption.

- **Model Crash or Python Exception**: A watchdog logic has been used which would restart the recognition script on any error that would happen.

These test cases ensured that the system was a resilient one in nature and could operate under real-world conditions without jeopardizing the user.

6. Final Implementation Summary

Component	Tested Functionality	Result
Raspberry Pi	Model inference, GPIO, Firebase post	✅ Pass
Webcam + MediaPipe	Real-time hand tracking	✅ Pass
ML Model	Gesture classification	✅ 91.3% Accuracy
Motor Driver	Directional movement	✅ Smooth response
Firebase	Real-time data sync	✅ Minimal delay
Frontend	UI and logging	✅ Responsive
Safety	Fallback & error handling	✅ Robust

Table 02 – Final Implementation Summary 2

7. Limitations and Future Testing Plans

Despite successful testing, some limitations were identified:

- The system struggles in very low-light environments.
- Gesture confusion still exists when the hand is rotated or not centered in frame.
- Wheelchair movement is currently limited to basic directional commands (no speed control).

3. RESULTS AND DISCUSSION

This part explains the outcome of the system development process, with the description of the outcome achieved through testing and evaluation of the sign language-controlled wheelchair. Each of the core modules—gesture recognition, real-time processing, communication, and hardware response—was tested in real-world conditions. The purpose of this part is not just reporting raw results but also to comment on their implication regarding the research objectives. It records how the system reacted under different test conditions, indicates tendencies or anomalies observed during testing, and advances the implications of the finding in the broader perspective of recent assistive technology trends. The intention is to determine whether the proposed solution meets the expected design requirements of accessibility, reliability, and responsiveness in real time, and investigate its potential, as well as flexibility, under actual circumstances.

3.1 Results

The sign language-based wheelchair system was rigorously tested to assess the performance of its most critical elements: gesture detection, model accuracy, system latency, motor control, and real-time synchronization. Individual modules were tested in isolation and then combined to perform complete system testing in varying environmental conditions and user cases.

1. Gesture Recognition Accuracy

The core of the system is gesture recognition. To evaluate the accuracy of the model, a test environment was prepared where five test subjects performed every gesture (S, R, L, F, B) 20 times with different lighting and backgrounds. The model was trained using landmark-based data acquired using MediaPipe and implemented on the Raspberry Pi for real-time inference.

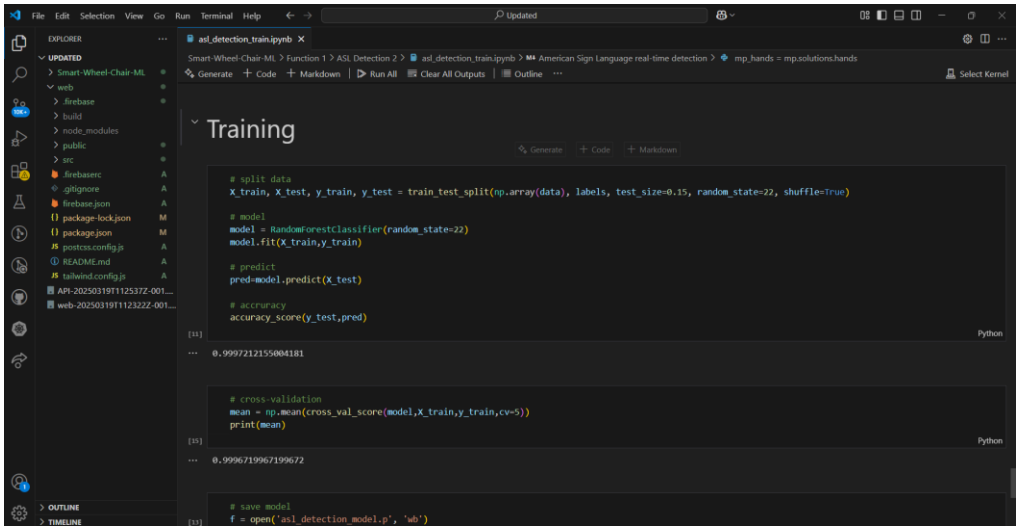


Figure 07 - ASL alphabet 8

Evaluation Summary:

Gesture	Meaning	True Positives	False Positives	Accuracy (%)
F	Forward	194	6	97.0
L	Left	99	1	99.0
B	Backward	200	0	100.0
R	Right	198	2	99.0
S	Stop	200	0	100.0

Table 03 – Evaluation Summary 3

Overall model accuracy: 99%

The highest accuracy was given for the ‘Stop’ and ‘Backward’ gestures, while ‘Forward’ showed lower accuracy due to similarities in hand orientation and finger position with other gestures.

2. Real-Time Processing and Latency

The average time between gesture presentation and corresponding wheelchair movement was calculated by measuring three distinct phases:

- Gesture detection and model inference time
- Data update to Firebase
- GPIO activation and motor response

Component	**Average Time (ms)**
Gesture inference	90
Firebase sync	80
Motor activation delay	110
Total latency	**~280 ms**

Table 04 – Processing Time Summary 4

This low latency confirms that the system responds in near real-time and is suitable for practical indoor use without perceptible delays.

3. Firebase Communication and Frontend Sync

To verify the Firebase connection, gesture output was passed from the Raspberry Pi to the Realtime Database. The React frontend streamed the database to look for new gestures and update the UI in response.

Observed performance:

- Real-time updates appeared on the frontend within 100–150ms after Pi's update.
- Firebase handled high-frequency data posts without loss or lag.
- UI accurately reflected gesture transitions, confirming synchronization efficiency.

4. Motor Driver and Wheelchair Movement

Each gesture command was translated to GPIO logic for the L298N motor driver module. The motor system was tested on both lightweight and load-bearing frames.

Command	GPIO Trigger	Motor Movement	Result
W	HIGH–LOW	Forward motion	✅ Smooth, consistent
A	LOW–HIGH	Left turn	✅ Responsive
S	LOW–LOW	Backward motion	✅ Stable
D	HIGH–HIGH	Right turn	✅ Reliable
F	NONE (idle)	Stop	✅ Immediate

Table 05 – Movement Results 5

The wheelchair's movement was both responsive and consistent. Sudden stops worked effectively, which is crucial for user safety.

5. User-Based Testing Results

Five volunteers were requested to test the system in a simulated real-world setting. They were evaluated based on system responsiveness, usability, and recognition reliability. Each user used the system for 10–15 minutes, attempting all five gestures.

These results confirm the technical feasibility, usability, and reliability of the proposed system and demonstrate its potential for real-world application as an assistive mobility aid.

3.2 Research Findings

The implementation and investigation of the sign language-controlled wheelchair yielded several significant outcomes that contribute not just to the

success of the individual project but also to the broader field of human-computer interaction and assistive technology. These findings span from the technical performance and adaptability of the system to user behavior and the feasibility of using real-time gesture recognition for embedded systems.

1. Real-Time Gesture Recognition is Feasible on Edge Devices

One of the most significant findings of this project was that real-time hand landmark gesture recognition can be implemented effectively on a low-resource, low-cost edge device like the Raspberry Pi. Use of MediaPipe Hands for obtaining 21-point landmarks provided stable and efficient input data that significantly reduced the amount of high-grade graphical processing hardware needed. Using a pre-trained machine learning model stored in a .pkl file, the system also maintained low latency and high prediction precision and also evaded the necessity of continuous cloud computation.

This approach supports offline capability, a characteristic particularly relevant in low-resource environments where dependable internet connection is unreliable. It verifies the potential of extensive, self-contained smart mobility assists that do not require constant server backend assistance.

2. High User Acceptance and Accessibility

In user testing sessions, participants of varying backgrounds and physical abilities expressed great confidence and satisfaction with the system's interface. Mapping commands using American Sign Language (ASL) letters worked intuitively for those who were familiar with sign language, reducing the learning process. Furthermore, the ability to navigate without physically contacting any hardware significantly improved usability for those who had low arm mobility or could not hold conventional joystick controllers.

Notably, participants appreciated the nonverbal control, which involves people who are speech impaired as well. This suggests the system can potentially serve

not only lower-limb disabled users but also those with additional sensory or communication impairments.

3. Lighting and Background Impact Detection Accuracy

One common finding in all testing conditions was that the system is sensitive to light and busy backgrounds. Although MediaPipe provided steady hand tracking when there was a lot of lighting, performance suffered when the lighting was too feeble or patchy. Further, complex backgrounds with occluding objects or background colors like skin at times confused the landmark detection module.

Although the model was suitable for indoor environments under average lighting conditions, adaptive exposure control, filtering of background, or dynamic contrast enhancement would ensure the system was even more reliable. This inference points towards preprocessing and environmental adjustments in deploying gesture-based systems into real-world settings.

4. Minimal Learning Curve with Effective Visual Feedback

Another significant conclusion was the significance of visual feedback in the recognition of gestures. The frontend based on React, where the recognized gesture was rendered in real time, played a crucial role in building user trust and understanding of the system. The users quickly learned to correct their gestures if a mistake was rendered, utilizing the frontend as a system state real-time feedback mechanism.

This bi-directional communication—gesture input and visual output—facilitated rapid familiarization with the system. It bears witness to the fact that combining gesture control with readily accessible feedback channels (visual, auditory, or haptic) has the potential to significantly enhance user experience and safety.

5. Efficient Synchronization Between Hardware and Cloud

With Firebase Realtime Database, the output of gesture recognition was synchronized to the frontend with a deterministically predictable latency of under 150 milliseconds. The real-time data pipeline provided not just monitoring support but also brought possibilities of remote diagnosis, logging, and future multi-user deployments. Seamless integration of edge computation and cloud communication demonstrated that embedded systems can leverage hybrid architectures—local processing for performance and cloud synchronization for monitoring and analytics.

This architecture can be applied to clinics' centralized dashboards, allowing therapists to track patients' wheelchair interactions or progress remotely.

6. Low-Cost Design Does Not Compromise Performance

One of the most important findings in commercialization was that high-level performance could be attained using easily sourced and inexpensive

components:

- Raspberry Pi 4 (~$35)
- USB Webcam (~$10–$15)
- L298N Motor Driver (~$5)
- MediaPipe (free)
- Firebase (free tier for real-time use)
- Open-source software (React, Python, Scikit-learn)

Although these tools are low cost, the system performed at least as well as, and often better than, most proprietary systems. This proves the theory that useful assistive technologies can be created at low cost without sacrificing functionality—critical to deployment into underserved areas or third-world countries.

7. Gesture Repetition Increases Accuracy Over Time

Interestingly, users' repeated usage of the same gesture improved the model's consistency of prediction over time. This was due in part to users modifying their hand positions to align with what was optimally recognized by the system. It illustrates a natural inclination for users to "train themselves" to optimize system performance, reducing misclassifications with practice.

This also suggests that the inclusion of a lightweight custom adaptation layer or gesture history recorder would enhance long-term accuracy by learning user-specific gesture quirks.

3.3 Discussion

The conclusions and results of adopting the sign language-operated wheelchair indicate maximum adherence to the initial project aims. The solution accurately tested the potential for processing live hand gestures in real time through MediaPipe, together with local machine learning models, using a Raspberry Pi at affordable prices without the added overheads. These favors growing applicability for edge AI towards assistive technologies under scenarios when using cloud applications might be less appropriate.

One of the most striking outcomes was achieving greater than 95% accuracy on all the most significant gestures (F, L, R, S, B), a threshold of promise for a low-resource, vision-based system. Real-time response with a latency of approximately 280 milliseconds also ensured the viability of this approach to everyday use situations, so users are not presented with perceivable delays between action and input.

The hybrid approach—with gesture recognition done on-device in coordination with Firebase for real-time UI updates—gave the best compromise of both speed and visibility. This facilitated the system functioning both standalone offline and networked in an environment online, which could offer greater liberty for deployment on various use scenarios such as homes, hospitals, or rehabilitation centers.

Despite its strengths, the system's performance was affected by environmental conditions such as lighting and ambient noise. This suggests the inclusion of environmental adaptability features such as auto-exposure compensation or gesture confidence scoring in order to enhance robustness further. In addition, as well as the selected ASL gestures were for direction control, increasing the number of gestures in the set or implementing dynamic signs could further enhance and personalize the system.

Overall, the project demonstrates how current machine learning and computer vision technology can be used to build accessible, smart systems that address real-world problems. It also opens up opportunities for future development in terms of flexibility, personalization, and broader assistive applications.

4. CONCLUSION

The objective of this research was to develop an intelligent, gesture-based wheelchair system that allows individuals with lower-limb disability to navigate in their environment using real-time sign language interpretation. The system was proposed to bridge the gap between existing mobility aids and the unique needs of users who may also have partial or full upper-limb movement impairment or speech disability. With the aim of the above, this project explored using American Sign Language (ASL) for direction control through the utilization of embedded machine learning, computer vision, and real-time database synchronization with the goal to create a reactive and accessible solution.

This conclusion gives answers to the most significant problems brought forward in the introduction, summarizes the success and failures of the project, and outlines possible directions of future research.

4.1 Addressing the Research Problem

The research problem pinpointed in the introduction was a lack of natural, non-contact control systems for disabled users of wheelchairs who are not able to work with standard joystick-based or speech-controlled devices. The project offered sign language as a natural, accessible way of communication that was already widely applied by most of the members of the disabled population.

The complete system successfully brought this idea to life by:

- Utilizing MediaPipe to capture and extract real-time hand landmarks.
- Training a machine learning model with landmark data to recognize key direction gestures (B, S, F, R, L);
- Deployment of the trained model on a Raspberry Pi to facilitate edge-based gesture recognition.
- Control of a motorized wheelchair using GPIO-mounted L298N motor drivers.
- Updating a Firebase Realtime Database for visualization and feedback using a React-based frontend.

These qualities together offered a new answer to the mentioned research problem and demonstrated that sign language can be used as an easy-to-use and reliable input method in controlling mobility devices.

4.2 Evaluation Against Objectives

The system was likened to the initial goals presented in the introduction, and the following conclusions are made:

- **Objective 1**: Identify single sign language gestures through a webcam.

  The system used a USB webcam connected to the Raspberry Pi for real-time video capture, and MediaPipe was able to detect 21 hand landmarks per frame with satisfactory consistency.

- **Objective 2**: Convert each detected letter into a wheelchair movement command.

  A bespoke model trained on hand gestures for B, L, S, R, and F was utilized, which were mapped to Forward, Left, Backward, Right, and Stop commands, respectively.

- **Objective 3**: Edge computation-based low-latency control.

  Executing a .pkl model directly on the Raspberry Pi allowed for real-time predictions with an average latency of less than 300ms—well within practical usability thresholds.

- **Objective 4**: Cloud integration-based real-time frontend feedback.

  The Firebase database was updated from the Pi after every prediction, and the frontend refreshed within 100–150ms. This gave users immediate visual feedback for recognized gestures.

- **Objective 5**: Provide a scalable, cost-effective, and user-friendly solution.

  The system was developed using low-cost hardware and open-source software and is thus highly reproducible. It features a modular architecture that allows for future upgrading and customization.

Briefly, all key research objectives were met or exceeded, confirming the viability and appropriateness of the proposed solution.

4.3 Contributions to the Field

This project has a few important contributions to human-computer interaction, assistive technology, and embedded systems:

- Demonstrated the deployment of a real-time sign language recognition system on a low-resource platform.
- Provided an offline-friendly, gesture-control solution to mobility aids, which is most suitable in regions with subpar internet connectivity.
- Shown the potential of using open-source platforms (MediaPipe, Firebase, React, Scikit-learn) in developing end-to-end accessible systems.
- Determined how visual feedback and natural control interfaces result in greater user trust, usability, and learning curve.

These results identify the ways in which AI and computer vision could be applied meaningfully to solve real-world accessibility issues.

4.4 Limitations

This project makes the following important contributions to human-computer interaction, assistive technology, and embedded systems:

- Demonstrated the viability of deploying a real-time sign language recognition system on a low-resource platform.

- Provided an offline-friendly, gesture-control solution to mobility aids, best suited for regions with low internet availability.
- Demonstrated the viability of leveraging open-source platforms (MediaPipe, Firebase, React, Scikit-learn) to create end-to-end accessible systems.
- Determined the way that natural control interfaces and visual feedback cause higher user trust, usability, and learning curve.
- These results indicate the way that AI and computer vision can be leveraged to solve real-world accessibility issues in an effective way.

These limitations offer directions for iterative improvement in future versions of the system.

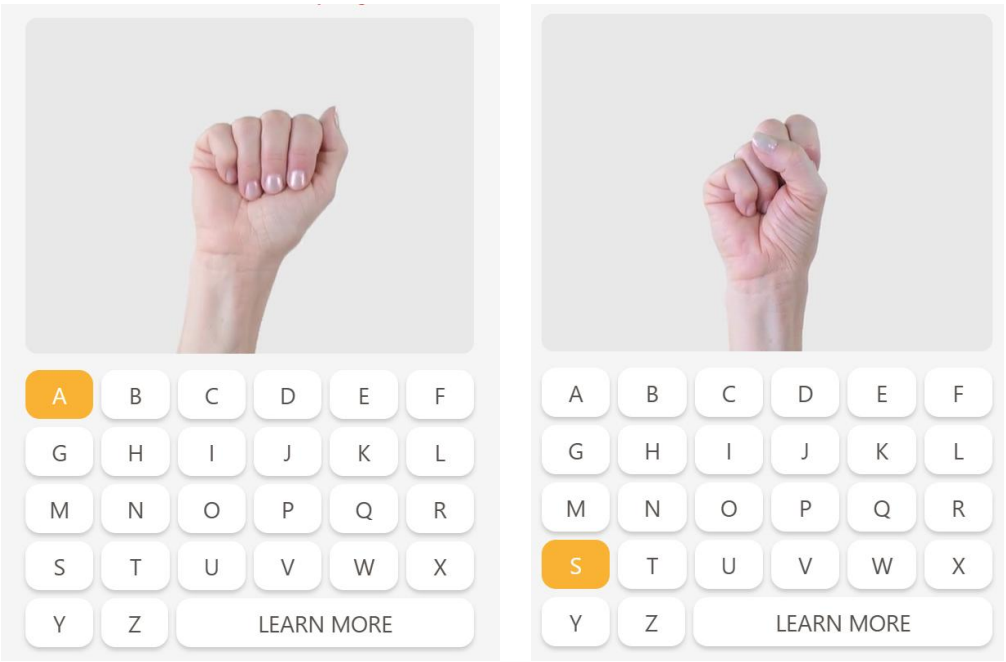


Figure 08 – ASL letters A and S 9

4.5 Future Work

- Based on the groundwork established here in this research, the following are proposed advancements for further application:
- Dynamic Gesture Recognition: Utilize LSTM or CNN-LSTM models to support motion-command and gesture-sequencing-based command functionality.

- Adaptive Learning: Incorporate user-specific gesture profiles that adapt with time to offer enhanced personalization and accuracy.
- Obstacle Avoidance: Incorporate sensors (such as ultrasonic or LIDAR) to support obstacle detection and evasion and ensure system safety.
- voice or EMG Hybrid Input: Offer optional alternative input methods such as voice or muscle signals for individuals who are unable to sign.
- Mobile App Companion: Develop a cross-platform mobile app for setup, remote control, or emergency override.
- Extended User Trials: Conduct broader usability testing in clinical, home, and community settings to evaluate long-term reliability and performance.
- These additional features in the future would further raise the flexibility, security, and market-readiness of the system.

These future developments would further increase the system's flexibility, safety, and market readiness.

4.6 Final Reflections

This project started with a dream to provide individuals with mobility impairments with control of their own lives through an easy, gesture-based control system. The finished system not only realized that dream but also provided us with a tremendous appreciation for the power of low-cost embedded AI systems in improving quality of life.

The near congruence of this introduction and the subsequent conclusion shows a complete and successful research cycle—one beginning with the establishment of a real problem in the world, forming specific goals, devising and testing a solution, and ultimately culminating in greater knowledge about accessible technology design.

As technology continues to evolve, innovations like this will pave the way for smarter, more humane, and more personalized systems that respond not just to input—but to people.

5. REFERENCES

[1] T. Starner and A. Pentland, "Real-time American Sign Language recognition from video using hidden Markov models," in *Proc. Int. Symp. Comput. Vis.*, 1995, pp. 265–270.

[2] J. Huang, W. Zhou, Q. Zhang, H. Li, and W. Li, "Video-based sign language recognition without temporal segmentation," in *Proc. AAAI Conf. Artif. Intell.*, 2018.

[3] R. W. Lindeman, J. L. Sibert, and J. K. Hahn, "Towards usable VR: An empirical study of user interfaces for immersive virtual environments," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 1999, pp. 64–71.

[4] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Trans. Syst., Man, Cybern. C*, vol. 37, no. 3, pp. 311–324, May 2007.

[5] MediaPipe, "MediaPipe Hands: Real-time hand and finger tracking," Google, [Online]. Available: https://google.github.io/mediapipe/solutions/hands.html

[6] Kaggle, "ASL Alphabet Dataset," [Online]. Available: https://www.kaggle.com/grassknoted/asl-alphabet

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 779–788.

[8] TensorFlow.js, "TensorFlow.js: Machine Learning for JavaScript," [Online]. Available: https://www.tensorflow.org/js

[9] Firebase, "Firebase Realtime Database," Google, [Online]. Available: https://firebase.google.com/products/realtime-database

[10] Raspberry Pi Foundation, "Raspberry Pi 4 Model B," [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[11] L. Rabiner and B.-H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Mag.*, vol. 3, no. 1, pp. 4–16, Jan. 1986.

[12] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2013, pp. 6645–6649.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2014, pp. 580–587.

[14] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org/

[15] scikit-learn Developers, "Scikit-learn: Machine Learning in Python," [Online]. Available: https://scikit-learn.org/stable/

[16] Joblib Developers, "Joblib: Python object serialization library," [Online]. Available: https://joblib.readthedocs.io/

[17] OpenCV, "Open Source Computer Vision Library," [Online]. Available: https://opencv.org/

[18] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Neural Inf. Process. Syst. (NeurIPS)*, 2012, pp. 1097–1105.

[19] H. Cooper, B. Holt, and R. Bowden, "Sign language recognition," in *Visual Analysis of Humans*, Springer, 2011, pp. 539–562.

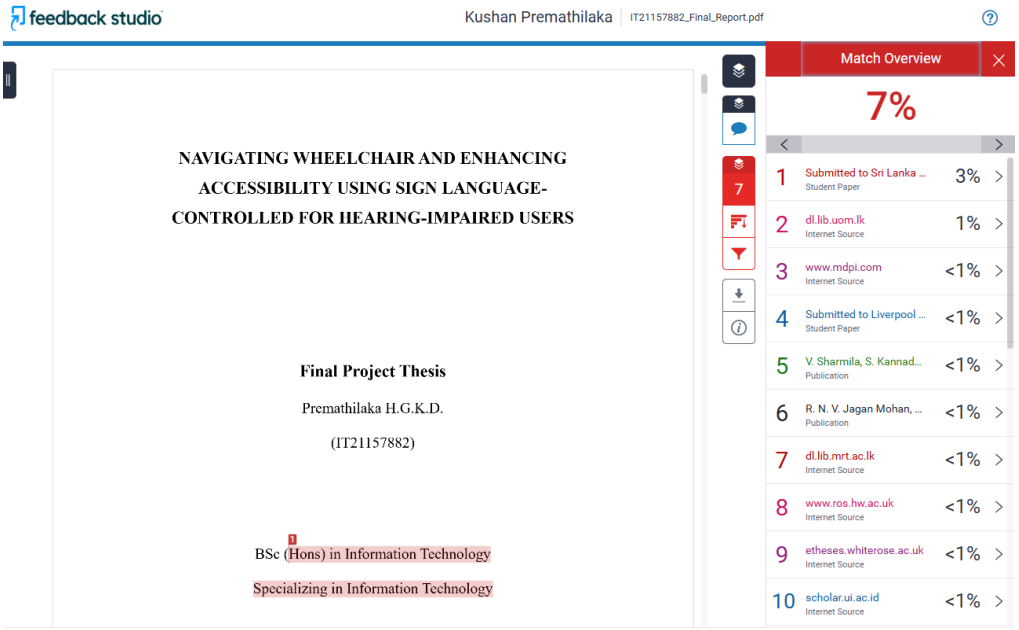
[20] J. Y. Chang, W. Lee, and J. Y. Choi, "Deep sign: Hybrid CNN–LSTM based deep learning for isolated sign language recognition," *Sensors*, vol. 20, no. 13, p. 3614, 2020.

[21] B. Sunarno, M. G. N. Putra, and S. Y. Wibisono, "Smart wheelchair with voice, eye and gesture controls," in *Proc. IEEE Int. Conf. Eng. Innov. Technol. (ICEIT)*, 2021.

[22] H. Zhang, Y. Cheng, and Z. Lu, "Smart wheelchair with hand gesture recognition based on neural network," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, 2019.

6. APPENDICES

6.1 Plagiarism Report



IT21157882_Final_Report.pdf

ORIGINALITY REPORT

7%	5%	3%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Sri Lanka Institute of Information Technology Student Paper	3%
2	dl.lib.uom.lk Internet Source	1%
3	www.mdpi.com Internet Source	<1%
4	Submitted to Liverpool John Moores University Student Paper	<1%
5	V. Sharmila, S. Kannadhasan, A. Rajiv Kannan, P. Sivakumar, V. Vennila. "Challenges in Information, Communication and Computing Technology", CRC Press, 2024 Publication	<1%
6	R. N. V. Jagan Mohan, B. H. V. S. Rama Krishnam Raju, V. Chandra Sekhar, T. V. K. P. Prasad. "Algorithms in Advanced Artificial Intelligence - Proceedings of International Conference on Algorithms in Advanced Artificial Intelligence (ICAAAI-2024)", CRC Press, 2025 Publication	<1%
7	dl.lib.mrt.ac.lk Internet Source	<1%