# Smart Glove Hand Gesture Vocalizer for Deaf and Dumb

*By*

Kinza Tanveer

CUI/FA19-EEE-007/ATD

Insia Inayat

CUI/FA19-EEE-030/ATD

Hassan Abbas

CUI/FA19-EEE-038/ATD

BS Thesis

In

Electrical (Electronics) Engineering

COMSATS University Islamabad
Abbottabad Campus-Pakistan

Spring 2023

**COMSATS University Islamabad-Abbottabad Campus**

# Smart Glove Hand Gesture Vocalizer for Deaf and Dumb

A Thesis Presented to

COMSATS University Islamabad- Abbottabad Campus

In partial fulfillment

of the requirement for the degree of

## BS Electrical (Electronics) Engineering

By

Kinza Tanveer

CUI/FA19-EEE-007/ATD

Insia Inayat

CUI/FA19-EEE-030/ATD

Hassan Abbas

CUI/FA19-EEE-038/ATD

Spring 2023

# Smart Glove Hand Gesture Vocalizer for Deaf and Dumb

An Under Graduate Thesis submitted to Electrical and Computer Engineering Department as partial fulfillment of the requirement for the award of Degree of Bachelor of Science in Electrical (Electronics) Engineering.


| Name | Registration Number |
|---|---|
| Kinza Tanveer | CUI/FA19-EEE-007/ATD |
| Insia Inayat | CUI/FA19-EEE-030/ATD |
| Hassan Abbas | CUI/FA19-EEE-038/ATD |


## Supervisor


Dr. Muhammad Bilal Qureshi

Assistant Professor, Department of Electrical and

Computer Engineering

Abbottabad Campus

COMSATS University Islamabad, Abbottabad Campus


July, 2023

# Final Approval

This thesis titled

## Smart Glove Hand Gesture Vocalizer for Deaf and Dumb

By

*Kinza Tanveer*

*CUI/FA19-BEE-007/ATD*

*Insia Inayat*

*CUI/FA19-BEE-030/ATD*

*Hassan Abbas*

*CUI/FA19-BEE-038/ATD*

Has been approved

For the COMSATS University Islamabad, Abbottabad Campus

Supervisor:_____

Dr. Muhammad Bilal Qureshi, Assistant Professor

Department of Electrical and Computer Engineering Engineering/CUI, Abbottabad Campus

HOD: _____

Dr. Owais, Professor

Department of Electrical and Computer Engineering, CUI, Abbottabad Campus

# Declaration

We Kinza Tanveer (CIIT/FA19-EEE-007/ATD), Insia Inayat (CIIT/FA19-EEE-030/ATD), and Hassan Abbas (CIIT/FA18-EEE-038/ATD), hereby declare that we have produced the work presented in this thesis, during the scheduled period of study. We also declare that we have not taken any material/data from any source except referred to wherever due. If a violation of rules has occurred in this thesis, we shall be liable to punishable action.

Date:_____

Signature of the student:

_____

Kinza Tanveer
CUI/FA19-EEE-007/ATD

_____

Insia Inayat
CUI/FA19-EEE-030/ATD

_____

Hassan Abbas
CUI/FA19-EEE-038/ATD

# Certificate

It is certified that Miss Kinza Tanveer, CIIT/FA19-EEE-007/ATD, Miss. Insia Inayat, CIIT/FA19-EEE-030/ATD, and Mr. Hassan Abbas, CIIT/FA19-EEE-46/ATD, have carried out all the work related to this thesis under my supervision at the Department of Electrical and Computer Engineering, COMSATS University, Abbottabad and the work fulfils the requirements for award of BS degree.

Date: _____

Supervisor:

_____

Dr. Muhammad Bilal Qureshi
Assistant Professor
Department of Electrical and Computer Engineering
COMSATS University Islamabad,
Abbottabad Campus

Head of Department:

_____

Dr. Owais,
Professor
Department of Electrical and Computer Engineering
COMSATS University Islamabad,
Abbottabad Campus

# DEDICATION

This thesis is dedicated to our beloved parents, who have always been a source of inspiration and encouragement for us to pursue higher education and face life's complexities with great zeal and enthusiasm.

# ACKNOWLEGDGEMENTS

First and foremost, we are in debt to our project supervisor, Dr. Muhammad Bilal Qureshi, for his unconditional support throughout our BS Final Year Project. His guidance helped us at all times during the project and in the process of writing our thesis. We would also like to express our heartfelt gratitude to the Pakistan Engineering Council (PEC) for their unwavering support and generous funding throughout the duration of this research project. Their commitment to advancing scientific knowledge has been instrumental in the successful completion of this thesis. We are truly thankful for their invaluable contributions to our work.

**Kinza Tanveer**

**CIIT/FA19-EEE-007/ATD**

**Insia Inayat**

**CIIT/FA19-EEE-030/ATD**

**Hassan Abbas**

**CIIT/FA19-EEE-038/ATD**

# ABSTRACT


## Smart Glove Hand Gesture Vocalizer for Deaf and Dumb

There has been an exponential increase in the population of hearing impaired and speech impaired people in recent years due to oral disease, congenital defects, or accidents. As humans, we communicate and get to know each another through our thoughts and ideas. Speech is the best approach to communicate our thoughts. However, for those who are disabled or deaf and dumb, the only means to interact with normal people is through sign language. The problem with sign language is that it is only confined to people who are also unable to speak, so only a few people are familiar with it.

Therefore, we aim to design a Smart Glove that will allow those who have speech impairments of any kind of speech defect to communicate through gestures; specifically, the user will perform hand gesture. The glove will capture every gesture the user makes and translate those gestures into both visual and audible form. In this way we will be able to break down communication barriers between normal people and special ones who are unable to conduct a normal conversation.

The glove will be fully equipped with sensors such as a flex sensor, an accelerometer, and a pressure sensor that will detect various sign language gestures. Flex sensors will be placed on fingers to measure finger bending in response to gestures. An accelerometer will be placed on the palm to measure hand position in the X, Y, and Z axes. Pressure sensors will be used to determine whether there is any contact between the fingers and with what force the fingers are bending. The data will be sent to the Arduino UNO board for further processing before being transferred to an Android phone via Bluetooth. The information obtained will be in the form of text. This text will then convert into speech using Google's text-to-speech converter.



**Keywords:** Glove, Flex Sensor, Accelerometer, Arduino Uno, Bluetooth Module

# TABLE OF CONTENTS

## LIST OF FIGURES

# LIST OF TABLES

# Chapter 1

# Introduction

## 1.1 Overview

About nine billion people in the world are deaf and dumb. Human beings have a natural ability to see, listen and interact with their external environment. Unfortunately, some people with disabilities are unable to use their senses to the best extent possible. Such individuals rely on alternative forms of communication, such as sign language. The language used by the deaf and the mute people is termed as sign language.

Individuals with hearing and speech impairment suffer discrimination and barriers that restrict their participation in various community activities. Due to the lack of proper communication, these individuals are deprived of their right to live, move, or even work independently. The phonological structure of SL generally has five elements, articulation point, configuration of the hands, type of motions, hand orientation, and facial expressions. These are essential elements of SL and can be exploited for sign recognition by automated intelligent systems.

Sign language recognition system (SLRS) is intended to bridge the gap between people with hearing impairment and those around them by creating a common medium among them such as a translator. The translator translates the signs into text or speech. In this regard, the academic literature suggests two types of translators: sensor-based sign language recognition and vision-based sign language recognition (VBSLR).

VBSLR solution to hand-gesture recognition problems depends on cameras and image processing techniques along with artificial intelligence (AI) methods. Single or multiple cameras can be used to localize the signer's body using saliency maps and skin color. For a long time, the VBSLRS approach outperformed the sensor-based approach. VGR techniques, on the other hand, can suffer from occlusion issues, changes in illumination conditions, changes in the distance between the signer and the camera, and high computation complexities. As a result of the high computational complexities, implementing real-time VBSLR is difficult.

An alternate approach to collecting gesture-related data is using instrumented gloves that are fitted with specific sensors such as flex, accelerometer and pressure sensors.

Glove-based systems, as opposed to vision-based systems (e.g., degree of bend, ordination, motion, and others), frequently have the advantage of being able to collect data directly, thereby eliminating the need to pre-process raw data. Figure 1.1 illustrates the various hand gestures used in sign language.


**Figure 1.1: Hand Gestures for Sign Language**

## 1.2 Purpose of the project

The major goal of this project is to break down communication barriers between normal people and special ones who are unable to conduct a normal conversation, whether they are disabled or deaf and dumb. Therefore, we aim to design a Smart Glove that will have the ability to convert hand gestures into both visual and audible form. It is a social initiative step which is taken to help uplift the deaf and dumb community and will also help to eliminate the social stigmas and norms that exist and thrive in our society regarding the deaf and dumb community.

## 1.3 Problem Statement

There has been an exponential increase in the population of hearing impaired and speech disabled people in recent years. When a disabled person, such as a deaf or dumb person tries to interact with a normal person, it becomes quite difficult for a normal person to understand what the disabled person is trying to convey. Because the only means of communication for a mute person is sign language.

The problem with sign language is that it is only confined to people who are also unable to speak. Therefore, a disabled person has to face numerous challenges while communicating with his normal surroundings. Moreover, one of the biggest issues that the disabled people face is lack of academic and social facilities and due to their condition, they also receive less attention than normal people and are often neglected. So, a technology is required to close this gap using systems that translate sign language into speech.

## 1.4 Objective

As communication between a normal person and a person who has lack of speech can be problematic and inconvenient. To avoid these problems, we will create an IoT-based smart glove with systems for converting hand gestures to speech. The glove will be equipped with at least two types of sensors i.e. Flex sensor and Accelerometer . Arduino / Raspberry pi will be used for hand gesture recognition. Sound module will be used to convert hand gestures into audible form. We can also use ML techniques i.e. CNN.

Our proposed idea will comprise of two stages in order to get an effective result.

1. Initially, we will use a smart glove to predict various gestures and then translate them into audible form using a sound/text module on cell phone.
2. Later we hope to make our design portable, sensor free where we will be using various image processing algorithms for feature extraction with the help of machine learning to predict and classify the gestures in video/image.

The main objective of our proposed design is to facilitate effective communication between the dead and dumb community and the rest of the world as it will have the following features:

**I. Gesture Recognition:**

The smart glove will accurately recognize and interpret hand gestures made by the user. This would enable the individual to convey messages and express themselves through gestures, which can be translated into speech/text.

**II. Speech Conversion:**

The smart glove will convert the recognized hand gestures into audible speech/text that would allow the individual to communicate with others who are not familiar with sign language.

**III. Real-time Communication:**

The smart glove will provide real-time communication capabilities, ensuring that the translation of gestures to speech or text happens quickly and efficiently. This would enable smooth and natural interactions with others.

**IV. Customization and Adaptability:**

The smart glove will be customizable to cater to individual preferences and needs. It will allow users to define and personalize their own set of gestures, making the system adaptable to their specific communication style.

**V. Education and Skill Development:**

Our design will serve as a tool for educational purposes, helping individuals learn sign language or enhancing their communication skills.

**VI. Portability and Integration:**

Our smart device will be portable and light weighted, allowing users to wear it comfortably in various settings and will be able to integrate with other devices or communication platforms, such as smartphones or video conferencing systems, to enable seamless communication across different mediums.

**Figure 1.2: Smart Glove**

## 1.5 Future Scope

This project will provide new opportunities for handicapped people, motivating them to improve their abilities so that they can keep up with the speed of the era. Deaf and dumb people will be able to do their regular work and meet their regular needs. The project is built with an efficient algorithm that can work with people of various physiques and anatomy. This project is focused on making communication between handicapped people and normal people easier and faster. Our scope includes three primary goals.

1. Choosing and comprehending an efficient algorithm.
2. Hardware design and implementation, as well as program code; and
3. Practical results with various gestures.

Our project has a lot of future recommendation for instance we can extend the usability of hand gestures. For instance, we can use actual hand gestures to operate video games instead of the usual input method, which is the joystick so that the user can experience the game in a completely new way and feel more grounded. We can create a hand gesture robot control system that can direct various machine functions using hand gestures in sensitive situations, such as a tele-operator robot that can support the surgeon while they perform operations. Moreover, the system that we have proposed can also be upgraded to include wireless data transmission capabilities.

## 1.6 Structure of Thesis

The outline of this thesis is shown as follows.

In Chapter 2, Information on different glove based technologies, different gesture recognition for sign language methods and literature review of some of previous related studies are provided.

In Chapter 3, the methodology of the proposed system is provided.

In Chapter 4, Hardware implementation is provided.

In Chapter 5, Results are discussed.

In Chapter 6, The conclusion and recommendations are provided.

# CHAPTER 2

## 2 Literature Review

Individuals with hearing and speech impairment suffer discrimination and language barriers that restrict their participation in various community activities. Due to the lack of proper communication, these individuals are deprived of their right to live, move, or even work independently. Language is a means of communication among humans. Different communities used different languages. Sign language (SL) is a means of communication among people who cannot hear or speak normally.

SL is a visual-spatial language based on positional and visual components, such as finger and hand gestures, position, and orientation alongside arm and body movements. These elements serve to convey the meaning of an idea together. The phonological structure of SL generally has five elements, articulation point, configuration of the hands, and type of motions, hand orientation, and facial expressions. Each gesture in SL is a combination of the five blocks mentioned. These five blocks are essential elements of SL and can be exploited for sign recognition by automated intelligent systems for instance: a smart glove.

Looking at the academic literature from the view of critical analysis, we have found that none of the presented articles has developed an approach of Data Glove SLR that is capable of handling a wide range of gestures. However, certain developments have covered several aspects related to hand attributes.

*(Vijay-alakshmi and Aarthi 2016)* [1] applied a sensor-based approach for English alphabet gesture recognition by developing a module to convert gesture to text using statistical template matching. This approach converts the output into text and finally to voice using HMM, which has been used to construct the speech corresponding to the text.

However, the glove did not include a mechanism for reading thumb movements, which has an active role in shaping many of the signs. Another factor is the small size of the training data (eight characters only). Nevertheless, the recognition accuracy ranged from 80 to 89%.

A wireless DataGlove was designed and developed *(Tanyawiwat and Thiemjarus 2012*) [2] for American Sign Language. A combination of five contact sensors and five flex sensors in addition to 3D accelerometer was used in this glove for fingerspelling gesture recognition. The gesture recognition engine performed statistical template matching. Gestures were collected from six deaf subjects and one healthy subject. However, low recognition accuracy (76.1%) was recorded with the 21 features from the new sensor glove. This result is due to the large number of misclassifications of letters such as D, H, X, Z, and SP.

The work presented by *Elmahgiubi et al. (2015)* [3] produced a sensory glove that captures the signs of American Sign Language (ASL) and converts these postures into text using template matching teaching.Three types of sensors were used for gesture acquisition: five flex sensors along with five force sensors and 6 degrees of freedom MPU6050.

However, the system is designed to recognize 26 ASL alphabet only. Furthermore, the system was able to interpret 20 out of the 26 letters. This condition means that the system misclassified six letters: E, M, N, S, T, and Z.

The electronic portable hand glove was developed by *Arif et al. (2016)* [4]. The glove consisted of five flex sensors and one accelerometer to capture hand form. A contact sensor was also involved in the process of distinguishing some signs. The system was used to translate gestures of ASL through statistical template matching to produce a useful solution for those with hearing impairment.

However, the system has been tested only on the alphabetical data of the ASL. In addition, it had a large system design and heavy equipment weight that made it inconvenient to use. Additionally, the loss of usability privilege as a portable device in public areas is considered as a failure.

Table 2.1 shows the different types of wearable sensors used in the past few years.

| Glove name | Active element (sensor used) | Actuator used | Performance | Evaluated application |
|---|---|---|---|---|
| 5DT Data Glove (commercial) | Fiber-optic sensor | None | Recognition accuracy of 97.4%, 12 static gestures | Rehabilitation, object grasping, and manipulation |
| Talking glove | Five strain gauges | None | N/A | Communication by deafblind people |
| IMU sensor-based electronic goniometric (iSEG) glove | 16 nine-axes IMUs | None | 85.24% accuracy for sensors | Rehabilitation |
| IMMU glove | 18 IMMUs | None | Accuracy of 89.59% for 10 static gestures, accuracy of 82.5% in recognizing 16 dynamic gestures | Defined static gestures |
| reduced graphene oxide (RGO) glove | Ten RGO-coated fibers | None | 98.5% recognition of 10 static and 98.3% for 9 dynamic gestures | Chinese sign language |
| Cyberglove (commercial) | 18 Sensors | None | 90% recognition of American sign language words | Word recognition using American sign language |
| Robost data glove | 14 fluid-based sensors | None | 88.5% for 15 gestures | Random gestures involving flexion–extension and abduction–adduction movements |

Table 2.1 Different Wearable gesture-based smart gloves.

## 2.1 Differentiation between Prevailing and Suggested System

Table 2.2 shows the differences between existing system and suggested system.

| Existing System | Proposed System |
|---|---|
| Sign Language | Smart Glove/ Gesture Recognition Using ML to get rid of the hardware and to make it portable |
| Use of hand gestures | Use of sound module and image processing techniques |
| More man power required | Little to none man power needed |
| Difficult to understand | Easy way of understanding ASL |

**Table 2.2 Difference in Prevailing and Suggested System**

## 2.2 Prevailing Systems

As per our research, some of the prevailing systems regarding sign language recognition are as follows:

### 2.2.1. Computer Vision-Based Systems

Vision-based hand gesture recognition is an area of active current research in computer vision and machine learning. Being a natural way of human interaction, it is an area where many researchers are working on, with the goal of making human computer interaction (HCI) easier and natural, without the need for any extra devices.

These systems use cameras to capture video of sign language gestures and then apply computer vision algorithms to analyze and recognize the gestures. Various techniques such as hand tracking, motion analysis, and feature extraction are employed to interpret the signs. Deep learning approaches, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are commonly used for recognition.

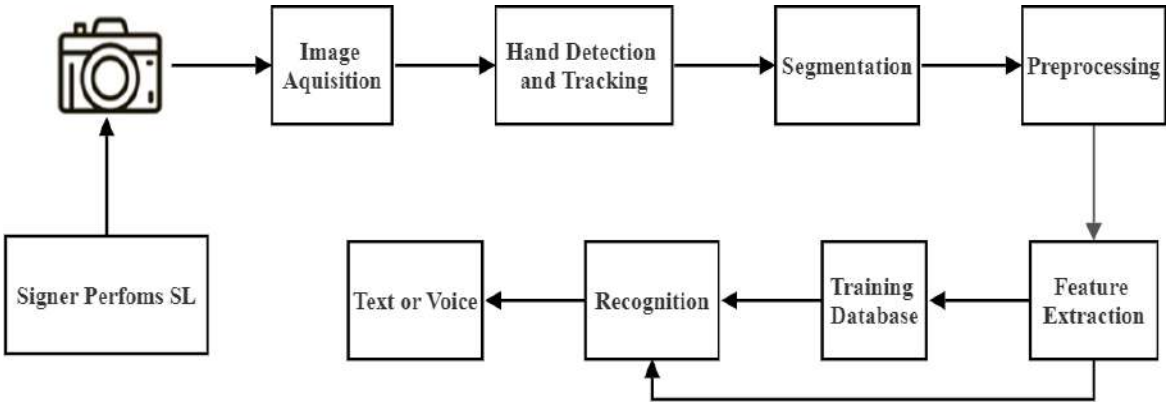



**Figure 2.1: Working of computer vision-based systems**

### 2.2.2. Working of Computer Vision-Based Systems

**i. Data Acquisition:**

The system captures video data using one or multiple cameras. The cameras are typically positioned to have a clear view of the signer's hands and body, capturing the sign language gestures from different angles if possible. The video data serves as input for subsequent processing.

**ii. Preprocessing:**

Before analyzing the video data, preprocessing techniques are applied to enhance the quality of the footage. These techniques can include noise reduction, image stabilization, and normalization to adjust for varying lighting conditions and camera perspectives.

**iii. Hand and Gesture Detection:**

Computer vision algorithms are used to detect and track the signer's hands in the video frames. Techniques such as background subtraction, skin color modeling, or machine learning-based approaches (e.g., cascade classifiers) can be employed to identify the regions of interest corresponding to the hands.

**iv. Feature Extraction:**

Once the hands are detected, various features are extracted from the video frames to represent the sign language gestures. These features can include hand shape, hand orientation, hand trajectory, and motion characteristics. Different computer vision techniques like edge detection, optical flow, or deep learning-based feature extraction methods (e.g., convolutional neural networks) can be utilized.

**v. Gesture Recognition:**

The extracted features are then fed into a recognition model or algorithm to classify and recognize the sign language gestures. This can involve pattern recognition, machine learning, or deep learning techniques. The model may have been pre-trained on a large dataset of sign language gestures or adapted specifically for the target sign language.

**vi. Output Generation:**

Once the sign language gesture is recognized, the system generates an appropriate output, which can vary depending on the application. It could be displayed as a text translation, synthesized speech, or even visual animations of the gestures.

### 2.2.3. Marker-Based Systems

Marker-based systems for sign language recognition involve the use of markers placed on the hands and body of sign language users. These markers can be reflective or colored, and they serve as reference points for tracking the movements of the hands and body during sign language communication.

### 2.2.4. Working of Marker-Based Systems

**i. Marker Placement:**

The markers are strategically placed on specific points of the signer's hands and body that are crucial for sign language recognition. For example, markers can be attached to the fingertips, palm, wrist, and other key joints.

**ii. Marker Tracking:**

Cameras or depth sensors are used to capture video footage of the signer. The captured video contains information about the movement of the markers as the signer performs sign language gestures. Computer vision algorithms are applied to track the markers' positions and orientations throughout the signing sequence.

**iii. Gesture Recognition:**

Once the marker positions are tracked, the system processes this data to recognize the sign language gestures. The tracked marker data can be analyzed using pattern recognition techniques or machine learning algorithms. The system compares the tracked marker movements against a pre-defined gesture database or a trained model to identify the corresponding sign gesture.

**iv. Output Generation:**

After recognizing the sign language gesture, the system generates an appropriate output, which can vary depending on the application. It could be displayed as a text translation of the sign gesture, synthesized speech, or even avatar-based animations that mimic the gestures in real-time.

### 2.2.5. Vision-Based Sensor Arrays

These systems employ a combination of cameras and depth sensors, such as Microsoft Kinect or Intel Real Sense, to capture both 2D video and 3D depth information. This additional depth data enhances the recognition accuracy by providing spatial information about the sign language gestures.

### 2.2.6. Working of Vision-Based Sensor Arrays

**i. Camera and Depth Sensor Setup:**

The system incorporates multiple cameras and depth sensors, such as Microsoft Kinect or Intel Real Sense, positioned strategically to capture the signer's movements from different angles. The cameras capture regular 2D video footage, while the depth sensors provide depth information, creating a 3D representation of the signer and their gestures.

**ii. Data Acquisition:**

Simultaneously, the cameras and depth sensors capture synchronized video frames and depth maps of the signer performing sign language gestures. The depth maps provide information about the distance of objects from the sensors, allowing for the reconstruction of a 3D representation of the signer's hands and body.

**iii. Fusion of Video and Depth Data:**

The captured 2D video frames and depth maps are then combined or fused to create a unified representation that contains both the visual appearance and spatial depth information of the signer's gestures. This fusion process aligns the corresponding 2D and 3D data points to create a more comprehensive understanding of the sign language gestures.

**iv. Hand and Gesture Tracking:**

Computer vision algorithms analyze the fused data to track the movements of the signer's hands and body in 3D space. Techniques such as feature tracking, optical flow, or depth-based hand tracking algorithms are applied to determine the positions, orientations, and trajectories of the hands and body parts.

**v. Gesture Recognition:**

Once the hand and body movements are tracked, the system applies gesture recognition algorithms to recognize the sign language gestures. These algorithms can utilize both the visual appearance and spatial depth information to analyze the movement patterns, hand shapes, and dynamics of the gestures. Machine learning or deep learning techniques can be employed to classify and recognize the gestures based on the captured 2D and 3D data.

**vi. Output Generation:**

After the sign language gestures are recognized, the system generates an appropriate output, such as text translation, synthesized speech, or visual animations, depending on the specific application or user requirements.

## 2.3 Suggested System

Although the existing systems in sign language recognition have made significant strides, our proposed system/design represents a notable advancement that surpasses their limitations and sets a new standard for accuracy, efficiency, and user experience.

Initially, our design would consists of primarily two sections:

**1.** Transmitter Section

**2.** Receiver Section.

The devices contained in the transmitter section are:

**1.** Flex sensor.
**2.** Accelerometer.

The devices contained in the receiving section are:

1. Arduino UNO.
2. HC- 05 (Bluetooth Module).
3. Cell Phone.

### 2.3.1. Sensor Glove Based System

We have developed a glove-based system for sign language recognition. The primary goal of this system is to accurately capture and interpret hand movements and gestures in order to facilitate effective communication between the mute community and normal people.

Following are the key components and functionality of our proposed design:

**i. Glove Design:**

The glove used in our system is specifically designed for sign language recognition. It is equipped with a various types of sensors, including accelerometers and flex sensors. These sensors are strategically placed on the fingers, palm or wrist to capture the full range of hand movements and positions.

### ii. Sensor Data Acquisition:

As the signer performs sign language gestures, the embedded sensors in the glove detect and measure changes in hand orientation, finger flexion, and overall movement. The sensors generate data that represents the physical attributes of the hand, such as angles, accelerations, and resistances. This data forms the basis for further analysis and recognition.

### iii. Gesture Recognition and Interpretation:

To enable accurate gesture recognition, we have recorded a dataset that includes values at different finger flexion levels corresponding to acceleration and resistances. Leveraging this dataset, we have trained our design using pattern recognition algorithms. As a result, our system is capable of interpreting at least four common signs: Yes, No, Okay, and Victory. This means that when a user performs any of these signs, the system can recognize and interpret them accordingly.

### iv. Bluetooth Integration and Speech Conversion:

In order to provide convenient and real-time feedback, our design incorporates a Bluetooth module. This allows the glove to communicate with a cell phone or other compatible devices. Once the hand gestures are recognized, the corresponding information is transmitted wirelessly to the cell phone. The cell phone then utilizes its capabilities, such as text-to-speech synthesis or pre-recorded speech samples, to convert the recognized hand gestures into audible speech. This feature enables seamless communication between the mute community and the normal people of our society.

Overall, our proposed design offers a practical and innovative solution for sign language recognition. By combining sensor technology, machine learning, and wireless communication, it presents a user-friendly approach to bridging the communication gap and enabling seamless interaction for sign language users.

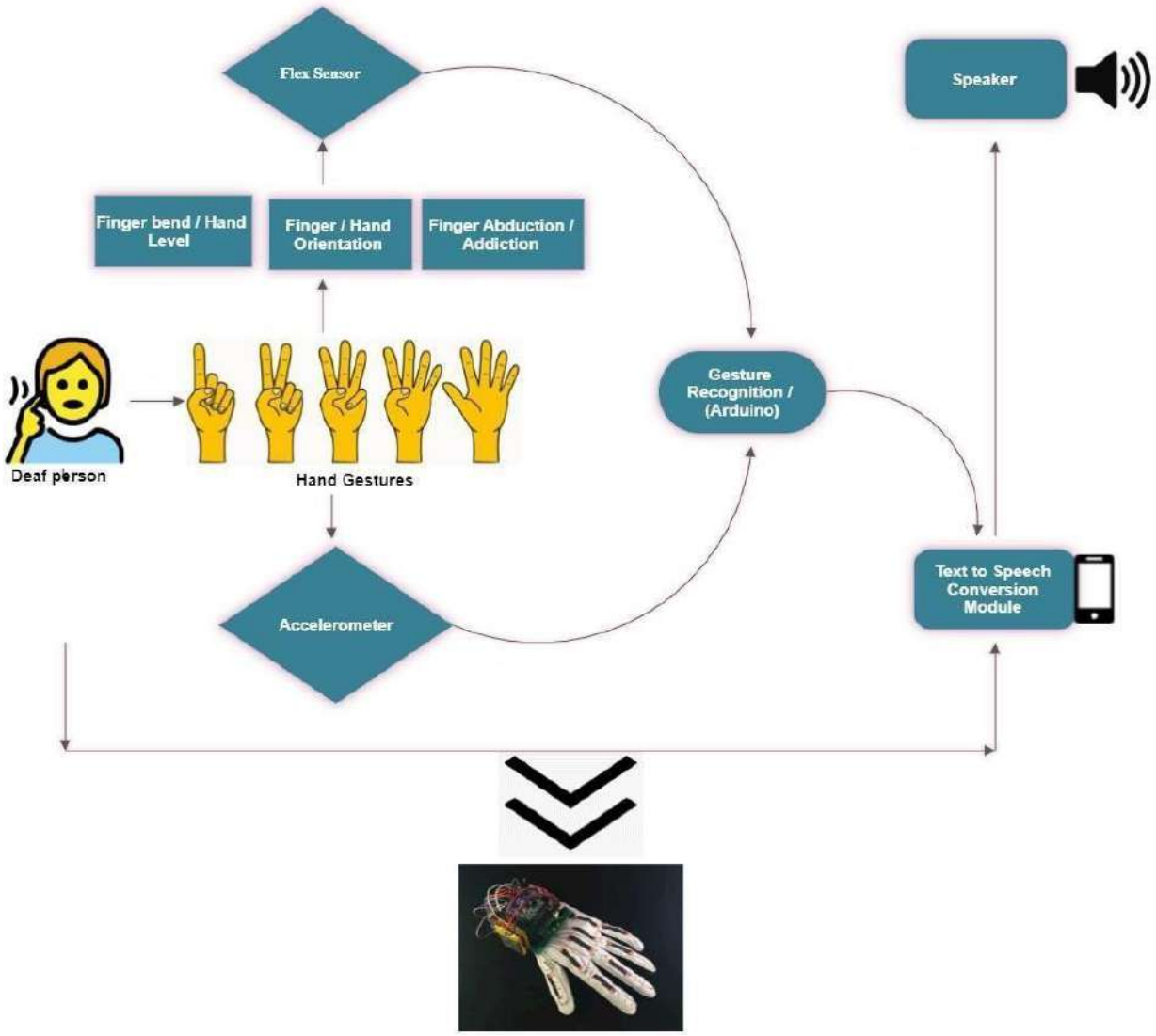



**Figure 2.2: Block diagram depicts the functionality of the proposed design**

### 2.3.2. Sensor Free Approach

Once the hardware development is complete and the algorithms for each sensor have been learned, our focus will shift towards creating a portable and sensor-free approach for sign language recognition. This will involve leveraging machine learning techniques, specifically Convolutional Neural Networks (CNN) for image recognition, combined with the Raspberry Pi and a camera to achieve a more reliable and portable system.

By utilizing CNN-based image recognition, we can train the system to recognize sign language gestures directly from the video input captured by the camera. This eliminates the need for additional sensors, making the system more compact, cost-effective, and easily deployable.

The Raspberry Pi, with its small form factor and computational capabilities, serves as an ideal platform for implementing the portable system. It provides the necessary computing power to run the CNN-based recognition model, process the images, and generate real-time results.

The camera, integrated with the Raspberry Pi, will capture the signer's hand movements and gestures, which will be fed into the trained CNN model for recognition. The model will analyze the captured images, extract relevant features, and classify the sign language gestures accurately.

This transition to a portable and sensor-free approach offers several advantages, including increased flexibility, ease of use, and wider accessibility. Sign language users will no longer be restricted to a specific hardware setup, enabling them to use the system on various devices, including smartphones, tablets, or even wearable devices.

Moreover, the reliance on machine learning techniques allows for continuous improvement and refinement of the recognition model as more data is collected and the system learns from user interactions. This adaptive learning capability enhances the system's accuracy and performance over time.

By incorporating CNN-based image recognition with the Raspberry Pi and a camera, our portable and sensor-free approach aims to revolutionize sign language recognition, making it more versatile, reliable, and user-friendly.

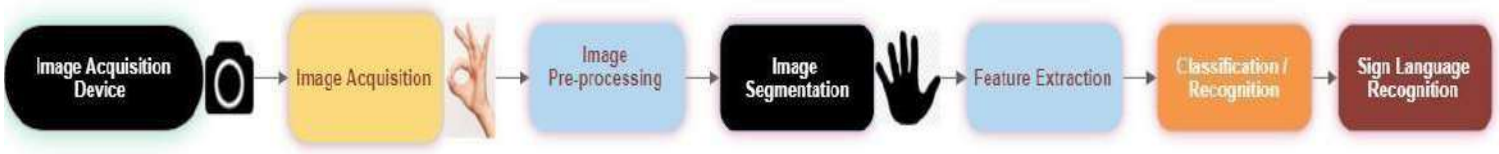



**Figure 2.3: Block diagram depicts the functionality of the sensor free approach**

### 2.3.3 Gesture Recognition Using TensorFlow and OpenCV

In our proposed design, we have also successfully incorporated gesture recognition using TensorFlow and OpenCV via MediaPipe.

**i. TensorFlow:**

It is an open-source machine learning framework developed by Google. It provides a comprehensive ecosystem of tools, libraries, and resources for building and deploying machine learning models. TensorFlow excels in training and deploying deep neural networks and is widely used for tasks like image classification, natural language processing, and gesture recognition. Its flexible architecture and efficient computation make it a popular choice for machine learning projects.

**ii. OpenCV:**

OpenCV, short for Open Source Computer Vision Library, is an open-source computer vision and image processing library. It offers a vast collection of functions and algorithms that aid in various computer vision tasks, including image and video processing, object detection, object tracking, and facial recognition. OpenCV provides extensive support for working with images and videos, making it an excellent tool for analyzing visual data.

In the context of gesture recognition, TensorFlow and OpenCV work together to enable accurate and efficient recognition of human gestures from image or video data.

Following is a brief outline of the process:

**iii. Data acquisition:**

Utilizing a camera or video input, we capture the visual data containing human gestures.

**iv. Preprocessing:**

OpenCV comes into play here, providing a range of functions for image processing tasks such as resizing, cropping, filtering, and noise removal. These operations help prepare the input data for gesture recognition.

**v. Feature extraction:**

OpenCV can extract relevant features from the preprocessed data, such as edges, contours, or keypoints. These features act as meaningful representations of the gestures, capturing essential information for further analysis.

**vi. Training the model:**

TensorFlow's machine learning capabilities allow us to build a gesture recognition model. This involves selecting an appropriate architecture, designing the neural network, and training it on labeled gesture data. TensorFlow's extensive documentation and pre-trained models are valuable resources for this step.

**vii. Inference and recognition:**

Once the model is trained, it can be used for recognizing gestures in real-time. OpenCV provides the means to feed video data to the TensorFlow model, enabling the prediction of gestures. The model analyzes the input frames, classifies the gestures, and provides the corresponding output.

**viii. Integration and application:**

The recognized gestures can be used to control various applications or systems, such as human-computer interfaces, gaming, or robotic control. The specific implementation details depend on the intended use case.

**ix. Media Pipe:**

MediaPipe, developed by Google, is an open-source framework that simplifies the development of cross-platform perceptual computing applications. It provides pre-built components and an easy-to-use pipeline for building complex machine learning pipelines, including gesture recognition. MediaPipe combines the power of TensorFlow and OpenCV with a flexible graph-based framework, enabling the rapid development and deployment of gesture recognition systems.

Over all, by incorporating gesture recognition using TensorFlow and OpenCV in our proposed design, we leverage the strengths of these two powerful libraries. TensorFlow empowers us to create accurate and efficient models for recognizing gestures, while OpenCV provides the necessary image processing and computer vision capabilities.

Together, they enable us to build a robust and effective gesture recognition system, enhancing human-computer interaction in various applications. MediaPipe, on the other hand, offers a streamlined framework for developing such systems, further simplifying the development process.

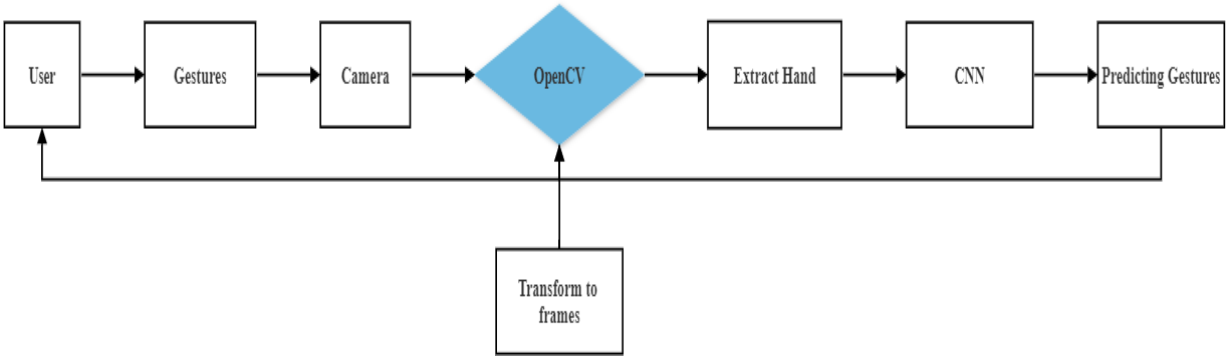



**Figure 2.4: Block diagram depicts the gesture recognition process using OpenCV**

## 2.4 Survey on SLR in context of vision-based and deep learning

This section provides an overview of previous surveys conducted on gesture and sign language recognition (SLR) research, highlighting the techniques employed in various studies. The information, including the applied techniques and performance evaluations, is presented and organized in tabular form within this section.

Table 2.2 shows techniques utilized in appearance-based SLR (Sign language recognition)

| Author/year | Image Acquisition/dataset | Pre-Processing | Hand segmentation | Feature extraction | Classification |
|---|---|---|---|---|---|
| **Badhe** et al. (2015) [1] | Camera | Threshold image | Canny Edge detection (Skin colour with yCbCr) | 2D FFT Fourier Descriptor | Euclidean Distance - Template matching |
| **Nanivadekar** et al.(2014) [2] | Digital camera video recorder | Video converted into frames | Gaussian (skin colour detection) | – | – |
| Nandy et al. (2010) [4] | Normal camera | – | HMM | Orientaion histogram | Euclidean Distance |
| P. V. V. Kishore and et all (2013) [5] | HD Sony camcorder | | Active Contours (AC) extract shape features | Horn Schunck optical flow (HSOF) | backpropagation training algorithm |
| Rao et al. (2018) [6] | Selfie mode video | DCT& Viola jones algorithm (frame pruning) | Adaptive histogram Hand-head segmentation | Feature matrix with n features | Ada boost mulilabel multiclass |
| Kishore et al. (2018) [7] | Sony Cybershot H7 digital camcorder | Gaussion low pass filter (gray scale) | Fusing Discrete wavelet transform & canny edge detector | Elliptical Fourier descriptor | Sugeno type fuzzy inference system |
| Shivansankra et al.(2018) [9] | ASLU | RGB to HSV, HSV to yCbCr colour space | Threshold image Binary erosion | Central mass of region | Roundness values and number of peaks |

| Author/year | Image Acquisition/dataset | Pre-Processing | Hand segmentation | Feature extraction | Classification |
|---|---|---|---|---|---|
| Nagi et al. (2011) [22 | front (CMOS) camera | RGB to YCbCr color space | Single Gaussian Model (SGM) | Erosion and dilation | MPCNN |

**Table 2.3: Techniques used in appearance-based SLR in the past years**

Table 2.3 provides a concise summary of traditional machine learning-based approaches.

| Author/year | Data set/image acquisition | Preprocessing/Pre-trained | NN Model | Loss Function | Optimizer | Classifier | Accuracy |
|---|---|---|---|---|---|---|---|
| Nikhil Kasukurthi and et all(2019) [3] | Intel Real sense P200 Depth camera | SqueezeNet | CNN | Categorical Cross Entropy | Stochastic Gradient Descent | softmax function | 83.29 % |
| B. Shi et al. (2018) [8] | You tube Web cam | AlexNet | Faster R–CNN (CNN-LSTM) | – | Stochastic Gradient Descent (SGD) | Softmax Function(CNN) CTC [68] | 42% |
| Ss Shivashankara and et all (2018) [10] | Iphone 6 60 FPS | Inception model(2014) | CNN-RNN | Categorical Cross Entropy | ADAM | softmax function | 90% |
| Su Yang and et all (2017) [12] | Video 10 FPS | RGB to HSV colour space | CNN-LSTM | log-likelihood function | ADADELTA | softmax function | 95% |
| Xiao and et all (2018) [13] | Microsoft Kinect RGB video | Colour and depth frames | CNN-Dynamic Baysian | – | – | softmax function & (CHMM) | 99.40 % |

| Author/year | Data set/image acquisition | Preprocessing/Pre-trained | NN Model | Loss Function | Optimizer | Classifier | Accuracy |
|---|---|---|---|---|---|---|---|
| Garcia et al. (2016) [46] | Surrey University and Massey University ASL datasets | GoogLeNet (ILSVRC2012) | CNN | Xavier initialization [56] | – | softmax-based loss function: | 97% |
| Liang et al. (2018) [50] | Microsoft Kinect sensors | median filtering, zero mean | 3D-CNN | negative log-likelihood | SGD | Softmax classifier | 83.66 % |
| **Bheda** et al. (2017)55 | Standard camera | image's background-subtraction | CNN | Categorical Cross Entropy Loss | SGD | Softmax classifier | 83.5 % |
| Molchanov et al. (2015) [21] | VIVA challenge's Dataset | nearest neighbor interpolation (NNI) | 3D-CNN (HRN & LRN) | negative log-likelihood | SGD | Softmax classifier | 77.5 % |
| Anantha rao et al. (2018) [23] | Selfie Camera | high-performance computing (HPC) | CNN | – | Sgd | Softmax classifier & RELU | 92.88 % |
| Yang et al. (2017) [36] | Standard camera | Haar feature classifier, | CNN | log-likelihood function | ADAGRAD | softmax function | 99.68 % |

**Table 2.4 Deep Learning based Approaches in the past years.**

# CHAPTER 3

# 3 Methodology

## 3.1 Flex sensor

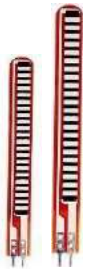
A flex sensor or bend sensor is a low-cost and easy-to-use sensor specifically designed to measure the amount of deflection or bending. It became popular in the 90s due to its use in the Nintendo Power Glove as a gaming interface. Since then people have been using it as a goniometer to determine joint movement, a door sensor, a bumper switch for wall detection or a pressure sensor on robotic grippers.

There are three types of flex sensors:

1. Conductive ink based.
2. Fiber optic based.
3. Conductive fabric based.

Since the most flexible and cost friendly flex sensor is conductive ink based. Therefore we will be incorporating it in our design in order to make it cost efficient.

A flex sensor is basically a variable resistor whose resistance varies with bending. Because the resistance is proportional to the amount of bending, it is commonly referred to as a Flexible Potentiometer. Flex sensors are typically available in two sizes: 2.2″ (5.588cm) long and 4.5″ (11.43cm) long. A flex sensor is made up of a phenolic resin substrate that has conductive ink deposited on it. On top is a segmented conductor that forms a flexible potentiometer whose resistance changes with deflection. Flex sensors are only intended to flex in one direction: away from ink. Bending the sensor in another direction may cause it to be damaged.



2.2″ 4.5″

**Figure 3.1: Flex Sensor**

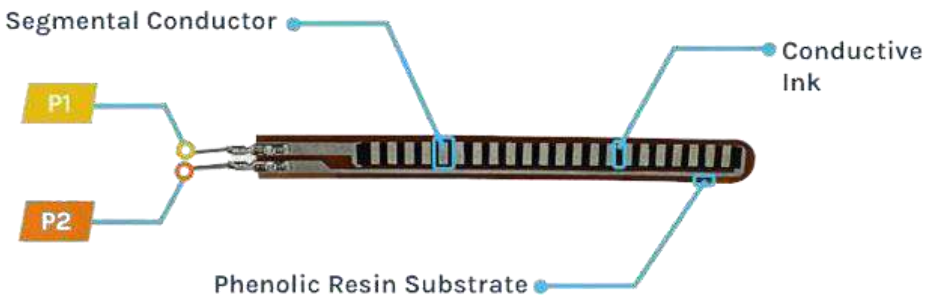



**Figure 3.2: Configuration of Flex Sensor**

### 3.1.1 Flex Sensor Pin-out

The flex sensor has two pins, one is P1 and the other one is P2, these two pins can be used to retrieve data from the flex sensor. The sensor acts more like a variable resistor, whose resistance changes based on how much it is bent, hence just like a resistor, the pins on this sensor are also interchangeable.

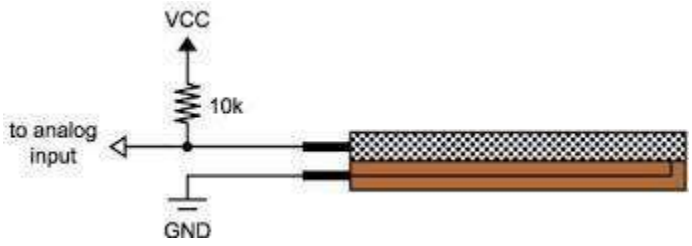



**Figure 3.3: Pin-out of Flex Sensor**

### 3.1.2 Flex Sensor Working

The conductive ink printed on the sensor acts as a resistor. When the sensor is straight, the resistance is about 25k as shown in the figure below:

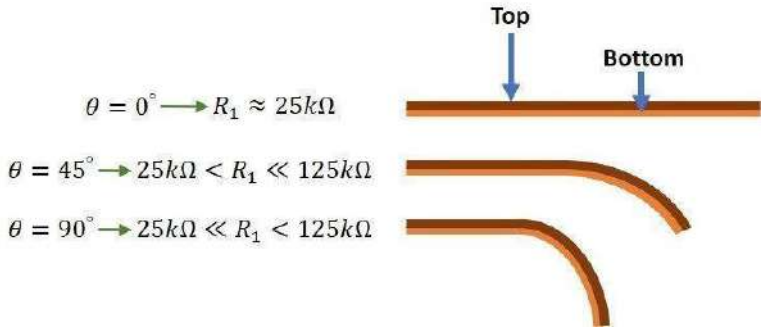



**Figure 3.4: Working of Flex Sensor**

The conductive layer is stretched when the sensor is bent, resulting in a smaller cross section (imagine stretching a rubber band). Because the cross section is reduced, the resistance increases. This resistance is approximately 100K at a 90° angle. The resistance returns to its original value when the sensor is straightened again. We can determine how much the sensor is bent by measuring the resistance.

### 3.1.3. Reading a Flex Sensor

The easiest way to read the flex sensor is to connect it with a fixed value resistor (usually 47kΩ) to create a voltage divider. To do this we connect one end of the sensor to Power and the other to a pull-down resistor as shown in the figure 2.5 . Then the point between the fixed value pull-down resistor and the flex sensor is connected to the ADC input of an Arduino. This way we can create a variable voltage output, which can be read by Arduino's ADC input.

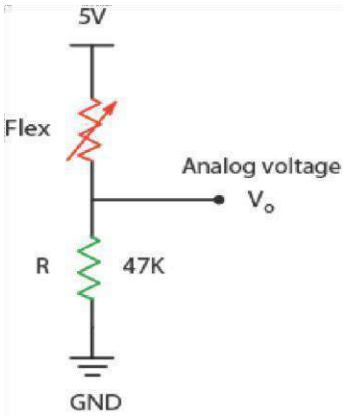



**Figure 3.5: Voltage divider for Flex Sensor**

The output voltage measured is the voltage drop across the pull-down resistor, not across the flex sensor. In the shown configuration, the output voltage decreases with increasing bend radius.

The output of the voltage divider configuration is described by the following equation:

$$Vo = Vcc \frac{R}{R + Rflex}$$

### 3.1.4 Interfacing Flex Sensor with Arduino

Connecting a flex sensor to an Arduino is very simple. All we need is to connect a 47kΩ pull-down resistor in series with the flex sensor to create a voltage divider circuit. The A0 ADC input of an Arduino is then wired to the junction of the pull-down resistor and the flex sensor.

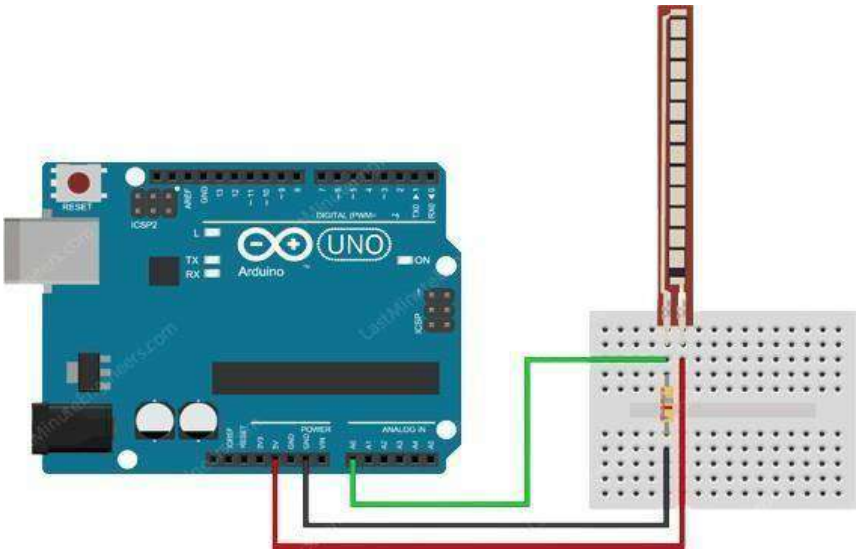



**Figure 3.6: Wiring a Flex Sensor with Arduino**

## 3.2 Accelerometer Sensor: (MPU6050)

In recent years, some crafty engineers successfully made micro machined gyroscopes. These MEMS (micro electromechanical system) gyroscopes have paved the way to a completely new set of innovative applications such as gesture recognition, enhanced gaming, augmented reality, panoramic photo capture, vehicle navigation, fitness monitoring and many more, no doubt the gyroscope and accelerometer are great in their own way. But when we combine them, we can get very accurate information about the orientation of an object. This is where the MPU6050 comes in. The MPU6050 has both a gyroscope and an accelerometer, using which we can measure rotation along all three axes, static acceleration due to gravity, as well as motion, shock, or dynamic acceleration due to vibration.

### 3.2.1. MPU6050 Module Overview

A low-power, low-cost 6-axis Motion Tracking chip combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor (DMP) in a small 4mm x 4mm package, as shown in the figure below. It can measure angular momentum or rotation along all three axes, as well as static acceleration caused by gravity and dynamic acceleration caused by motion, shock, or vibration.


**Figure 3.7: MPU6050 Accelerometer**

The module comes with an on-board LD3985 3.3V regulator, so you can use it with a 5V logic microcontroller like Arduino without worry. The MPU6050 consumes less than 3.6mA during measurements and only 5μA during idle. This low power consumption

allows the implementation in battery driven devices. In addition, the module has a power LED that lights up when the module is powered.

### 3.2.2. MPU6050 Module Pin out

The pin diagram of the module is as shown in figure below:

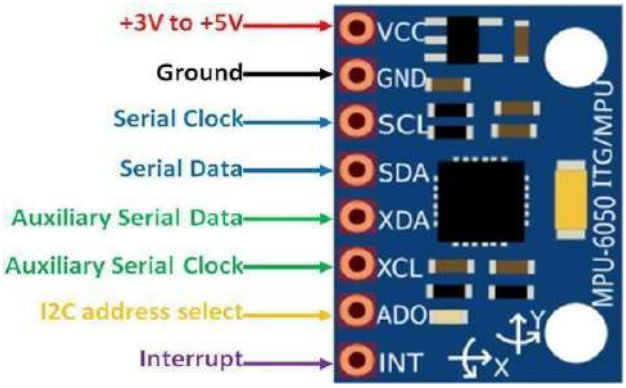



**Figure 3.8: MPU6050 Module Pin out**

VCC is the power supply for the module. Connect it to the 5V output of the Arduino.

GND should be connected to the ground of Arduino.

SCL is a I2C Clock pin. This is a timing signal supplied by the Bus Master device. Connect to the SCL pin on the Arduino.

SDA is a I2C Data pin. This line is used for both transmit and receive. Connect to the SDA pin on the Arduino.

XDA is the external I2C data line. The external I2C bus is for connecting external sensors.

XCL is the external I2C clock line.

AD0 allows you to change the internal I2C address of the MPU6050 module. It can be used if the module is conflicting with another I2C device, or if you wish to use two MPU6050s on the same I2C bus. When you leave the ADO pin unconnected, the default I2C address is $0x68_{HEX}$ and when you connect it to 3.3V, the I2C address becomes $0x69_{HEX}$.

INT is the Interrupt Output. MPU6050 can be programmed to raise interrupt on gesture detection, panning, zooming, scrolling, tap detection, and shake detection.

### 3.2.3 Measuring Acceleration

The MPU6050 can measure acceleration using its on-chip accelerometer with four programmable full-scale ranges of ±2g, ±4g, ±8g and ±16g.

The MPU6050 has three 16-bit analog-to-digital converters that simultaneously sample the 3 axes of movement (along X, Y and Z axis) as shown in fig.

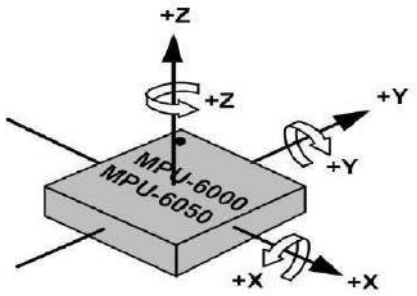



**Figure 3.9: Accelerometer MPU6050**

### 3.2.4. Measuring Rotation

The MPU6050 can measure angular rotation using its on-chip gyroscope with four programmable full-scale ranges of ±250°/s, ±500°/s, ±1000°/s and ±2000°/s.

The MPU6050 has another three 16-bit analog-to-digital converters that simultaneously samples 3 axes of rotation (around X, Y and Z axis). The sampling rate can be adjusted from 3.9 to 8000 samples per second.

### 3.2.5. Interfacing Accelerometer with Arduino

The MPU 6050 communicates with the Arduino through the I2C protocol. If MPU 6050 module has a 5V pin, then we can connect it to Arduino's 5V pin. If not, we will have to connect it to the 3.3V pin. Next, the GND of the Arduino is connected to the GND of the MPU 6050. Then we would have to connect Arduino's digital pin 2 (interrupt pin 0) to the pin labeled as INT on the MPU 6050.

Next, we would need to set up the I2C lines. To do this, we will connect the pin labeled SDA on the MPU 6050 to the Arduino's analog pin 4 (SDA), and the pin labeled as SCL on the MPU 6050 to the Arduino's analog pin 5 (SCL).

Following figure illustrates the wiring of an arduino with accelerometer:

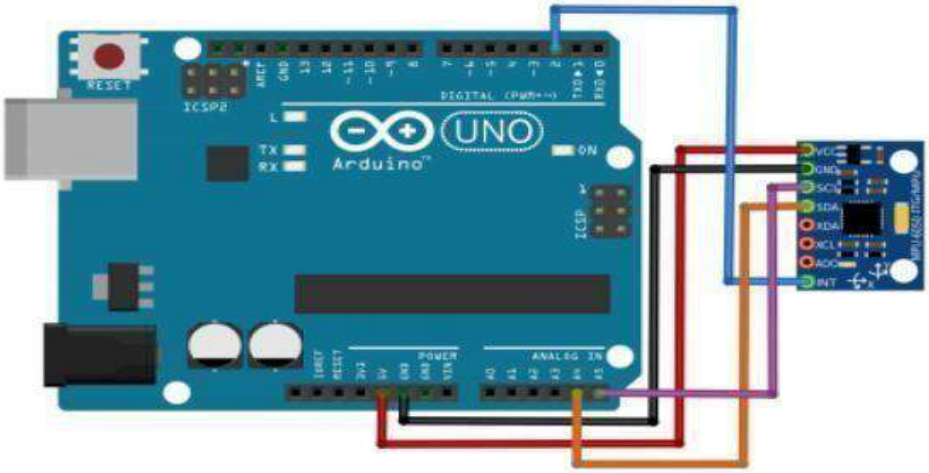


**Figure 3.10: Wiring an accelerometer with Arduino**

## 3.3 Algorithm of the Project

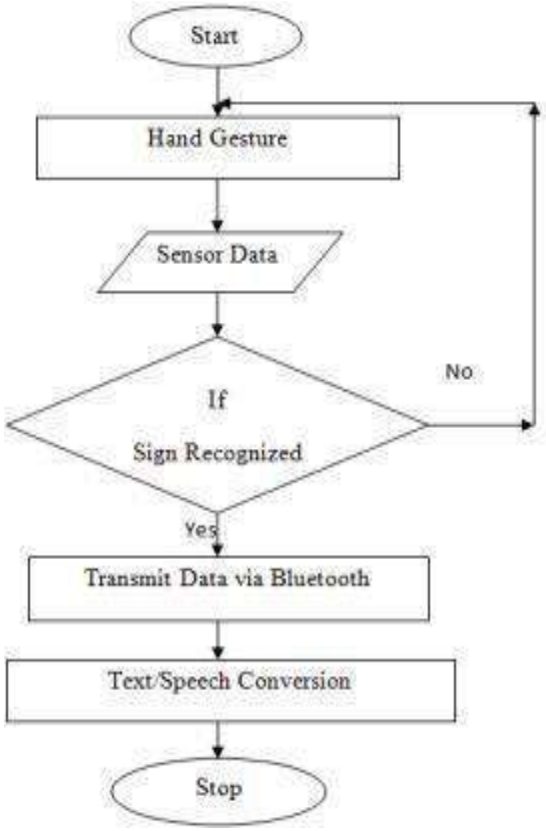



**Figure 3.11: Flow Chart of the proposed hardware model**

The flow of algorithmic program employed in the system is as shown in figure 3.7. The glove is attached with four flex sensors over the thumb, index, middle, ring finger on the hand, Arduino uno and an accelerometer. The accelerometer acknowledges the precise movement of the fingers. If the sign is recognized by the module, the data will be transmitted to an Android app via a Bluetooth module (HC-05), and the gesture made will be converted into text/speech. Otherwise, an error indicating that the sign is not recognized will be displayed.

# CHAPTER 4

# 4 Hardware Implementation

## 4.1 Hardware Details

### 4.1.1. Arduino Uno

The Arduino Uno is a widely used microcontroller board that plays a crucial role in the hardware implementation of the hand gesture vocalizer for the deaf and dumb. It serves as the central processing unit, responsible for controlling and coordinating the various components of the system. It is based on the ATmega328P microcontroller, which offers a range of features suitable for interactive and embedded projects. It provides a user-friendly development environment and a rich ecosystem of libraries, making it a popular choice among hobbyists and professionals alike. It also offers several essential features for the hand gesture vocalizer project. It has a sufficient number of digital and analog input/output pins, allowing connection to various sensors and actuators. These pins can be utilized to interface with the gesture recognition sensor, microphone, speaker, display, and other components of the system.

The board also includes built-in communication interfaces such as UART, SPI, and I2C, which enable seamless communication with external devices or modules. This capability is particularly useful when integrating wireless modules or connecting to a computer or smartphone for data exchange.it is powered via USB or an external power source, providing flexibility in terms of power supply options. This allows the hand gesture vocalizer to be powered using a computer's USB port, a power bank, or a dedicated power adapter.


**Figure 4.1: Arduino Uno**

- **Technical Specifications:**

1. Microcontroller: ATmega328P
2. Operating Voltage: 5 volts
3. Input Voltage (recommended): 7-12 volts
4. Digital I/O Pins: 14 (of which 6 provide PWM output)
5. Analog Input Pins: 6
6. Flash Memory: 32KB (ATmega328P, of which 0.5KB is used by the bootloader)
7. SRAM: 2KB (ATmega328P)
8. EEPROM: 1KB (ATmega328P)
9. Clock Speed: 16 MHz
10. Communication Interfaces: UART, SPI, I2C, USB
11. Power Consumption: 50mA (operating), 20mA (standby)
12. Dimensions: 68.6mm x 53.4mm

- **Characteristics**

The Arduino Uno is based on the ATmega328P microcontroller, operating at 16 MHz. It provides a balance between performance and power consumption.

**i. Digital Input/Output Pins:**

The board has a total of 14 digital input/output (I/O) pins. These pins can be individually configured as either inputs or outputs, allowing for digital communication and control.

**ii. Analog Input Pins:**

Arduino Uno features 6 analog input pins. These pins can measure analog voltages ranging from 0 to 5 volts, enabling the connection of analog sensors or devices.

**iii. Flash Memory:**

The ATmega328P microcontroller on the Arduino Uno has a flash memory capacity of 32KB. This memory is used for storing the program code that runs on the microcontroller.

**iv. SRAM:**

The Arduino Uno has 2KB of Static Random Access Memory (SRAM). SRAM is used for storing variables and data during program execution.

**v. EEPROM:**

It includes 1KB of Electrically Erasable Programmable Read-Only Memory (EEPROM). This non-volatile memory can store data that needs to be retained even when the power is disconnected.

**vi. Operating Voltage:**

The board operates at a voltage of 5 volts. Most of its digital and analog components operate at this voltage level.

**vii. Input Voltage:**

The Arduino Uno can be powered via USB or an external power supply. The recommended input voltage range is 7 to 12 volts. It includes a voltage regulator that ensures stable operation even with varying input voltages.

**viii. Communication Interfaces:**

The Arduino Uno offers several communication interfaces. It has a built-in USB interface for programming and serial communication with a computer. It also supports UART (Universal Asynchronous Receiver-Transmitter), SPI (Serial Peripheral Interface), and I2C (Inter-Integrated Circuit) for communication with external devices.

**ix. Dimensions:**

The Arduino Uno has a compact form factor, with dimensions of approximately 68.6mm x 53.4mm. This size makes it easy to incorporate into various projects, including those with limited space.

**x. Development Environment:**

Arduino Uno is supported by the Arduino Software (IDE). The IDE provides a user-friendly interface for writing, compiling, and uploading sketches (programs) to the board.

**xi. Open-Source:**

The Arduino Uno is an open-source platform. It's hardware design, schematics, and software libraries are freely available. This open nature allows for customization, modification, and community-driven development, fostering a vibrant ecosystem of projects and ideas.

These characteristics highlight the capabilities and features of the Arduino Uno, making it a versatile and widely used microcontroller board for various applications, including the implementation of the hand gesture vocalizer.

## 4.1.2 MPU6050

By incorporating the MPU6050 into the hand gesture vocalizer system, it becomes possible to capture and interpret the user's hand movements accurately. This enables real-time translation of sign language gestures into spoken or written communication, facilitating effective communication between deaf and dumb individuals and non-sign language users.

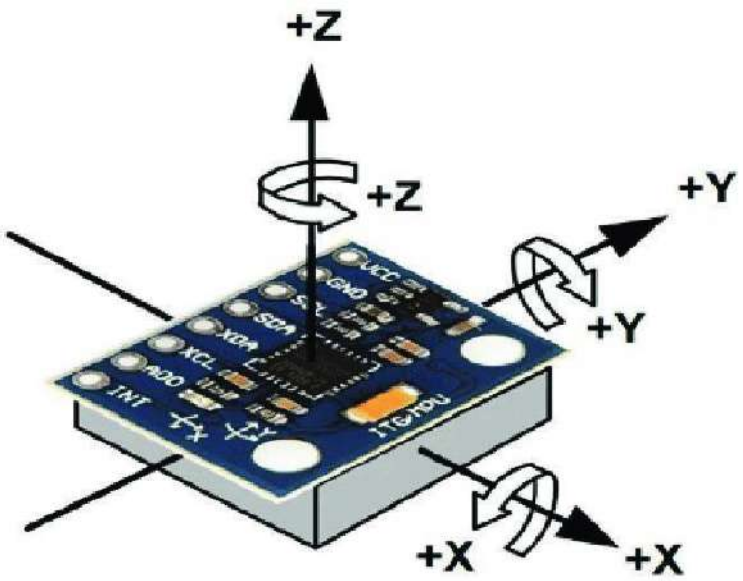



**Figure 4.1: Accelerometer MPU6050**

It combines a 3-axis accelerometer and a 3-axis gyroscope in a single chip, providing accurate motion detection and measurement capabilities.

- **Technical specifications and characteristics of the MPU6050:**

**Accelerometer:**

1. 3-axis accelerometer with programmable full-scale range.
2. Sensitivity range: ±2g, ±4g, ±8g, ±16g.
3. Resolution: 16 bits.
4. Measurement range: -/+ 2g, -/+ 4g, -/+ 8g, -/+ 16g.

**Gyroscope:**

1. 3-axis gyroscope with programmable full-scale range.
2. Sensitivity range: ±250°/s, ±500°/s, ±1000°/s, ±2000°/s.
3. Resolution: 16 bits.
4. Measurement range: -/+ 250°/s, -/+ 500°/s, -/+ 1000°/s, -/+ 2000°/s.

**Power Supply**:

1. Operating voltage: 2.3V to 3.4V.
2. Low-power consumption: typically 3.9mA in full operating mode.

**Temperature Sensor:**

1. Built-in temperature sensor with a measurement range of -40°C to +85°C.
2. Resolution: 16 bits.

**Digital Output:**

1. 16-bit digital output for accelerometer and gyroscope data.
2. Supports both raw and processed data output modes.

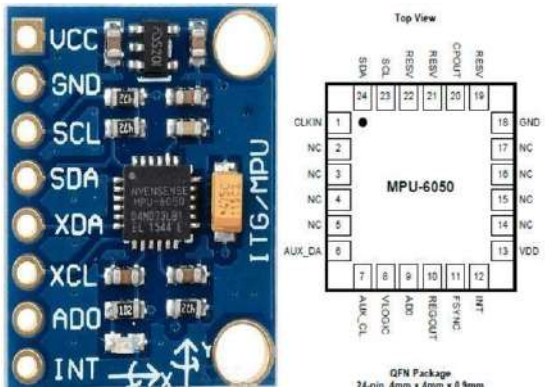



**Figure 4.2: Pin Configuration of MPU6050**

### 4.1.3. Working Principle of MPU6050

The MPU6050 combines a 3-axis accelerometer and a 3-axis gyroscope in a single chip. The accelerometer measures linear acceleration in three directions (X, Y, and Z), while the gyroscope measures angular velocity or rotational motion around those axes.

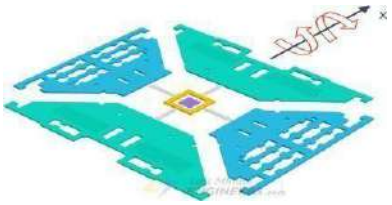


**Fig 4.4: Accelerometer in X plane**

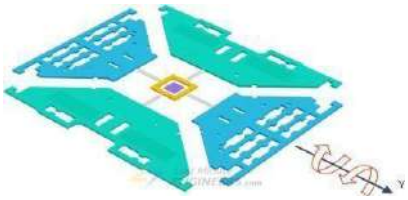


**Fig 4.5: Accelerometer in Y plane**

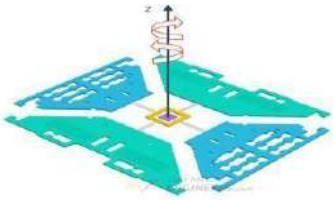


**Fig 4.6: Accelerometer in Z plane**

The MPU6050 uses microelectromechanical systems (MEMS) technology to detect motion. The accelerometer and gyroscope sensors consist of tiny mechanical structures that sense changes in capacitance or resistance due to acceleration or rotation. These changes are then converted into electrical signals proportional to the motion.

The MPU6050 integrates a Digital Motion Processor (DMP), which performs sensor fusion. It combines the accelerometer and gyroscope data to provide accurate and stable orientation estimation, compensating for the strengths and limitations of each sensor. This sensor fusion enhances motion tracking accuracy and reduces drift over time.

**Communication and Integration:**

The MPU6050 communicates with microcontrollers or other devices using the I2C (Inter-Integrated Circuit) serial interface. It supports a programmable I2C address, allowing multiple MPU6050 modules to be connected on the same bus.

To integrate the MPU6050 into a project, you typically connect it to a microcontroller that reads the sensor data via the I2C interface. The microcontroller then processes the data, applies algorithms if needed, and utilizes it for motion tracking, orientation estimation, or gesture recognition.

Overall, the MPU6050 provides a compact and efficient solution for motion sensing applications, combining accelerometer and gyroscope data for accurate motion tracking and orientation estimation. Its versatility and ease of integration make it a popular choice in various fields.

**Applications:**

1. Motion tracking
2. Gesture recognition
3. Inertial Measurement Units (IMUs)
4. Robotics
5. Orientation tracking
6. Activity monitoring
7. Gaming and virtual reality
8. Health monitoring
9. Sports analysis
10. Human-machine interaction

### 4.1.4 Flex Sensor

The flex sensor is a bendable, resistive sensor that changes its resistance based on the amount of bending or flexing applied to it. It is often used to measure the degree of flexion or bend

- **Working Principle:**

The flex sensor is constructed using a flexible material that has conductive elements embedded within it. When the sensor is bent or flexed, the distance between these conductive elements changes, altering the resistance of the sensor. The resistance is directly proportional to the degree of bending or flexing.

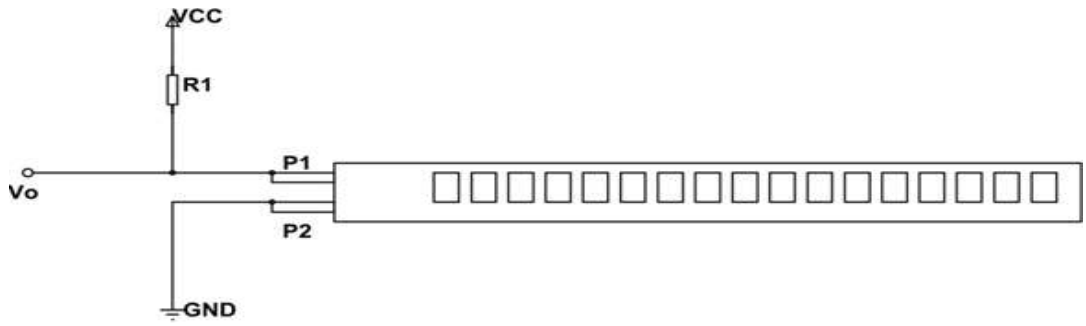



**Figure 4.7: Flex sensor working.**

- **Sensing Range:**

A flex sensor has a range from about ~10K to ~35K, that means it won't give us a full 0-5 volt range (or 0-1023 analog value). Try to use the serial monitor below to find out what analog value you will take while you bending the sensor. Flex sensors are available in different lengths and have a specific sensing range, which indicates the maximum angle of flexion they can detect. Common sensing ranges are 45 degrees, 60 degrees, or 90 degrees.

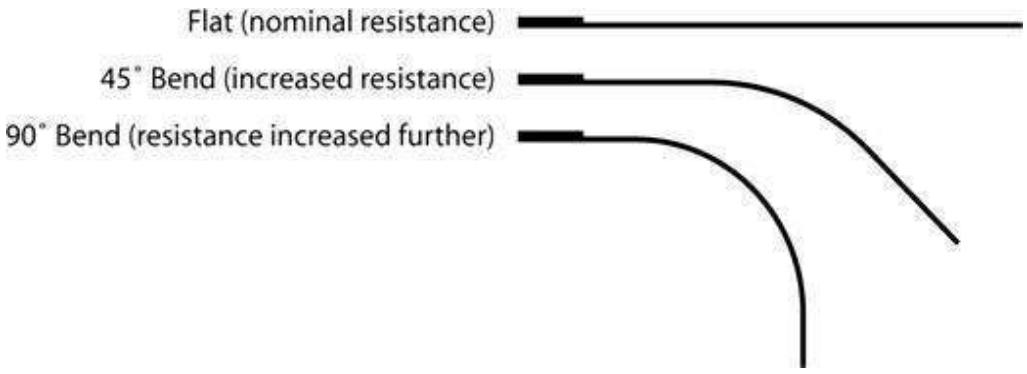



**Figure 4.8 Sensing Range of Flex sensor.**

- **Analog Output:**

  Flex sensors provide analog output. As the resistance changes with bending, the sensor's output voltage or current also changes proportionally. This analog signal can be read by microcontrollers or other analog-to-digital conversion (ADC) devices for further processing.

- **Calibration:**

  Flex sensors may require calibration to establish a baseline resistance value corresponding to no bending. This calibration process ensures accurate measurements relative to the neutral or resting position of the sensor.

  By integrating the flex sensor into hand gesture vocalizer hardware, we capture specific finger flexion patterns, enhance the range of recognized gestures, and provide a more comprehensive and intuitive interaction with the system. The flex sensor adds an extra dimension to gesture recognition, allowing for a more nuanced and expressive communication experience for the user.

- **Finger Flexion Detection:**

  Attach the flex sensor to each finger or specific hand joints of the user. As the user flexes or bends their fingers, the resistance of the flex sensor will change proportionally. Measure this resistance using an analog-to-digital converter (ADC) to determine the degree of finger flexion.

- **Gesture Mapping:**

  Associate different finger flexion patterns or gestures with specific commands, phonetic elements, or words in your hand gesture vocalizer system. For example, a specific finger flexion pattern could represent the letter "A" or a common word like "OK."

- **Signal Processing:**

  Process the analog output from the flex sensor using appropriate algorithms or filters to ensure accurate and reliable gesture recognition. You may need to calibrate the flex sensor to establish a baseline resistance value for neutral finger positions.

- **Integration with Other Sensors**:

  Combine the data from the flex sensor with other sensors like the MPU6050 (mentioned earlier) to capture a more comprehensive hand gesture. The MPU6050 can provide additional motion data, such as hand orientation or movement, while the flex sensor specifically captures finger flexion.

- **Microcontroller Interface**:

  Connect the flex sensor to a microcontroller board (Arduino) using an analog input pin. The microcontroller will read the analog values from the flex sensor and perform the necessary processing to interpret the gestures.

- **Gesture Recognition and Output:**

  Implement a gesture recognition algorithm in our microcontroller or external processing unit to interpret the flex sensor data and trigger the corresponding vocalization or output action. This involve mapping recognized gestures to pre-recorded audio files or synthesizing speech output based on the recognized gestures.

**Applications:**

1. Robotics
2. Prosthetics
3. Wearable Technology
4. Musical Instruments
5. Biomechanics and Sports
6. Human-Computer Interaction
7. Health Monitoring
8. Virtual Reality and Gaming
9. Industrial Automation
10. Safety and Ergonomics

### 4.1.5 Bluetooth Module HC-05

The HC-05 Bluetooth module provides a convenient and reliable solution for establishing wireless communication between devices. Its ease of integration, flexibility, and support for serial communication make it a popular choice for various wireless projects.

- **Bluetooth Version:**

  The HC-05 module is based on Bluetooth version 2.0+EDR (Enhanced Data Rate), which provides reliable and secure wireless communication.

- **Communication Range:**

  The module has a communication range of approximately 10 meters (33 feet) in an open space environment, allowing for short-range wireless connectivity.

- **Serial Communication:**

  The HC-05 module supports serial communication through UART (Universal Asynchronous Receiver-Transmitter) interface. It can be connected to a microcontroller or other devices using serial communication protocols like UART, making it easy to integrate into projects.

- **Master/Slave Mode**:

  The HC-05 module can operate in either master or slave mode. In slave mode, it can connect to a master device such as a smartphone, tablet, or computer. In master mode, it can initiate connections with other slave devices.

- **Compatibility:**

  The HC-05 module is compatible with various devices and platforms, including Arduino boards, Raspberry Pi, and other microcontrollers or development boards that support serial communication.

- **Configuration:**

  The module has AT commands that allow for configuration and customization of settings, such as changing the device name, baud rate, security settings, and pairing mode. These commands are sent through the serial interface to configure the module.

- **Power Supply:**

  The HC-05 module typically operates at 3.3V power supply, but some versions can support a wider range, such as 3.6V to 6V.

- **Application:**

  The HC-05 module is commonly used in projects that require wireless communication, such as remote control systems, home automation, wireless sensor networks, and data logging applications.

- **Pairing and Security:**

  The module supports pairing with other Bluetooth devices using a passkey. It offers basic security features such as authentication and encryption to ensure secure data transmission.

- **Data Transfer Rate:**

  The HC-05 module supports a maximum data transfer rate of 2.1 Mbps, allowing for fast and efficient wireless data transmission.

**Figure 4.9: HC-05**

### 4.1.7 Raspberry Pi Implementation

- **Technical Specifications:**

- **Processor:**
  Broadcom BCM2711 quad-core Cortex-A72 (ARMv8-A) 64-bit SoC
  1.5GHz clock speed
- **Memory:**
  2GB, 4GB, or 8GB LPDDR4-3200 SDRAM options
- **Storage:**
  MicroSD card slot for storage
- **Video Output:**
  2 × micro-HDMI ports (supports up to 4K at 60Hz)
- **Audio Output:**
  3.5mm audio jack
  HDMI audio output
- **USB Ports:**
  2 × USB 3.0 ports
  2 × USB 2.0 ports
- **Network:**
  Gigabit Ethernet (RJ-45)
  Dual-band 802.11ac wireless LAN
  Bluetooth 5.0
- **GPIO Pins:**
  40 GPIO pins
- **Power Supply:**
  5V/3A via USB-C connector
- **Operating System:**
  Raspberry Pi OS (formerly known as Raspbian)
  Various other Linux distributions are also supported
- **Dimensions:**
  85.6mm × 56.5mm

- Support for hardware-accelerated video playback and graphics rendering

- Improved thermal management compared to earlier Raspberry Pi models
- Support for multiple displays through dual HDMI ports.

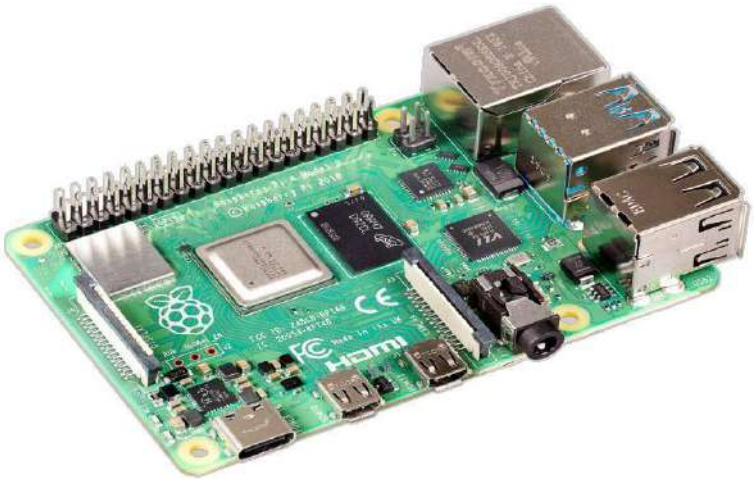


**Figure 4.10 Raspberry Pi 4B**

**1. Hardware Setup:**

- Raspberry Pi 4B .
- Raspberry Pi Camera Module or a USB webcam for capturing gestures.
- Microphone and speaker for audio input/output.
- Power supply and required cables.

**2. Software Setup:**

- Install Raspberry Pi OS (Bullseye) on our Raspberry Pi.
- Update the system and install necessary dependencies using **apt**:

**3. Package Import:**

- Import necessary Python packages, including OpenCV and TensorFlow for hand gesture recognition.

**4. Initialize Models:**

- Configure machine learning models for hand and gesture recognition.
- Use MediaPipe to set up the hand recognition model.
- Load a trained TensorFlow model for gesture recognition.

**5. Environment Setup:**

- Ensure that the Raspberry Pi OS Bullseye is installed and properly configured.
- Verify the compatibility of libraries and packages with the Bullseye release.

6. **Webcam Initialization:**
    - Connect and initialize a compatible webcam or camera module for capturing video frames.
    - Configure the camera source, resolution, and frame rate.
7. **Detect Hand Keypoints:**
    - Convert captured frames to the appropriate format, usually RGB, using OpenCV.
    - Use the MediaPipe or similar library to detect and locate hands within the video frames.
    - Extract hand landmarks and coordinates for further analysis.
8. **Gesture Recognition:**
    - Utilize the TensorFlow model to recognize specific hand gestures.
    - Implement logic to map detected landmarks to recognized gestures.
    - Consider optimizing the model for edge devices to accommodate the Raspberry Pi's limited computational resources.
9. **Speech Synthesis Integration:**
    - When a hand gesture is successfully recognized, trigger the text-to-speech (TTS) component.
    - Use library gTTS and convert the recognized gestures into spoken language.
10. **Testing and Fine-Tuning:**
    - Thoroughly test the system to ensure accurate hand gesture recognition and vocalization.
    - Fine-tune the models and algorithms to achieve desired performance levels.

## 4.2 Wiring Diagram

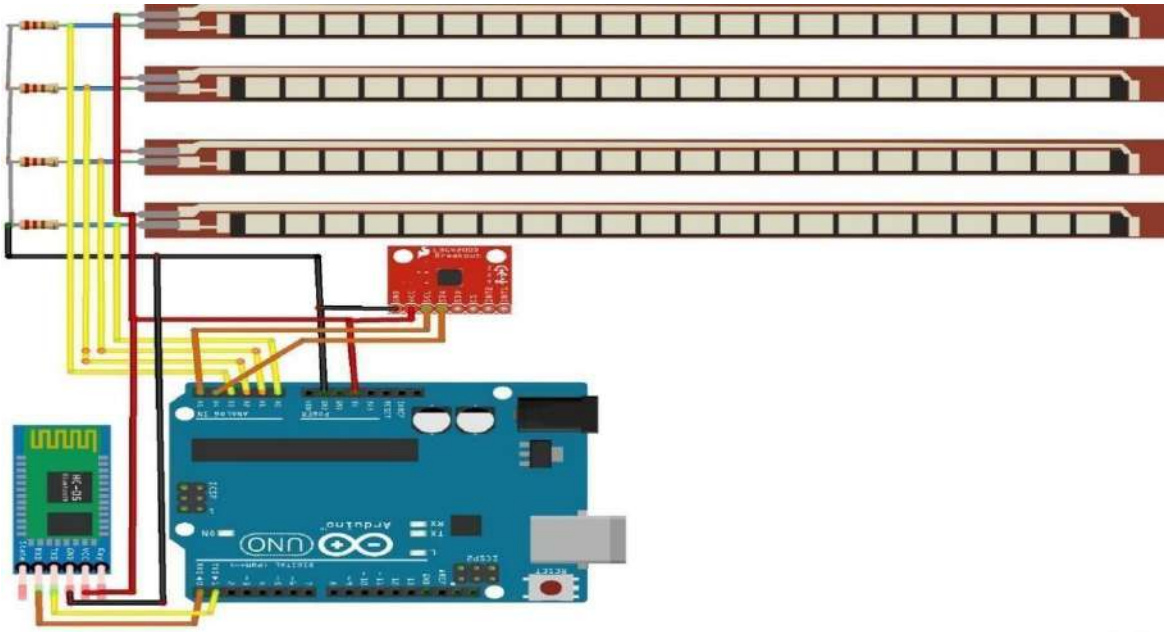


**Figure 4.11: Wiring Diagram of the prepared hardware**

## 4.3 Prepared Hardware Model

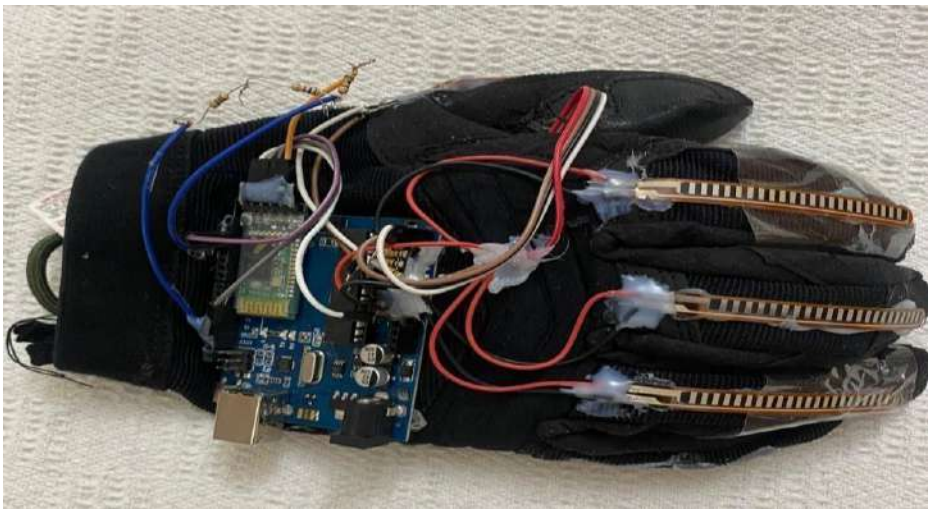


**Figure 4.11.1: Prepared hardware**

## 4.4 Software Implementation

Following are the steps to implement a code in PyCharm using OpenCv and TensorFlow to produce a hand gesture recognition algorithm:

- **Step 1- Import necessary packages:**

To build a Hand Gesture Recognition algorithm, we will need four packages. So we will first import those necessary packages.

```
5  import cv2
6  import numpy as np
7  import mediapipe as mp
8  import tensorflow as tf
9  from tensorflow.keras.models import load_model
```

**Figure 4.11: Importing Necessary Packages**

- **Step 2 – Initialize models:**

```
12 mpHands = mp.solutions.hands
13 hands = mpHands.Hands(max_num_hands=1, min_detection_confidence=0.7)
14 mpDraw = mp.solutions.drawing_utils
```

**Figure 4.12: Initializing MediaPipe**

Here;

1. Mp.solution.hands module performs the hand recognition algorithm. So we create the object and store it in mpHands.
2. Using mpHands.Hands method we configured the model. The first argument is max_num_hands, that means the maximum number of hand will be detected by the model in a single frame. MediaPipe can detect multiple hands in a single frame, but we'll detect only one hand at a time in our project.
3. Mp.solutions.drawing_utils will draw the detected key points for us so that we don't have to draw them manually.

```
17  model = load_model('mp_hand_gesture')
18
19  # Load class names
20  f = open('gesture.names', 'r')
21  classNames = f.read().split('\n')
22  f.close()
23  print(classNames)
```

**Figure 4.13: Initializing Tensorflow**

Here;

1. Using the load_model function we load the TensorFlow pre-trained model.
2. Gesture.names file contains the name of the gesture classes. So first we **open** the file using python's inbuilt open function and then read the file.
3. After that, we read the file using the read() function.

**Our proposed model can recognize 10 different gestures including:**

1. Okay
2. Peace
3. Thumbs Up
4. Thumbs Down
5. Call me
6. Stop
7. Rock
8. Live Long
9. Fist
10. Smile

- **Step 3 – Read frames from a webcam:**

```
cap = cv2.VideoCapture(0)

while True:
    # Read each frame from the webcam
    _, frame = cap.read()

    x, y, c = frame.shape
```

```
    # Flip the frame vertically
    frame = cv2.flip(frame, 1)
    framergb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

```
    # Show the final output
    cv2.imshow("Output", frame)

    if cv2.waitKey(1) == ord('q'):
        break

# release the webcam and destroy all active windows
cap.release()

cv2.destroyAllWindows()
```

**Figure 4.14: Initializing the Webcam**

Here;

1. We create a VideoCapture object and pass an argument '0'. It is the camera ID of the system. In this case, we have 1 webcam connected with the system. If you have multiple webcams then change the argument according to your camera ID. Otherwise, leave it default.

2. The cap.read() function reads each frame from the webcam.
3. cv2.flip() function flips the frame.
4. cv2.imshow() shows frame on a new openCV window.
5. The cv2.waitKey() function keeps the window open until the key 'q' is pressed.

- **Step 4 – Detect hand keypoints:**

```
    # Get hand landmark prediction
    result = hands.process(framergb)

    # print(result)

    className = ''

    # post process the result
    if result.multi_hand_landmarks:
        landmarks = []
        for handslms in result.multi_hand_landmarks:
            for lm in handslms.landmark:
                # print(id, lm)
                lmx = int(lm.x * x)
                lmy = int(lm.y * y)

                landmarks.append([lmx, lmy])

            # Drawing landmarks on frames
            mpDraw.draw_landmarks(frame, handslms, mpHands.HAND_CONNECTIONS)

            # Predict gesture
            prediction = model.predict([landmarks])
            # print(prediction)
            classID = np.argmax(prediction)
            className = classNames[classID]

    # show the prediction on the frame
    cv2.putText(frame, className, (10, 50), cv2.FONT_HERSHEY_SIMPLEX,
                   1, (0,0,255), 2, cv2.LINE_AA)
```

**Figure 4.15: Detecting Hand Gestures**

Here;

1. MediaPipe works with RGB images but OpenCV reads images in BGR format. So, using cv2.cvtCOLOR() function we convert the frame to RGB format.
2. The process function takes an RGB frame and returns a result class.

3. Then we check if any hand is detected or not, using result.multi_hand_landmarks method.
4. After that, we loop through each detection and store the coordinate on a list called landmarks.
5. Here image height (y) and image width(x) are multiplied with the result because the model returns a normalized result. This means each value in the result is between 0 and 1.
6. And finally using mpDraw.draw_landmarks() function we draw all the landmarks in the frame.

# CHAPTER 5

# 5 Results

## 5.1 Hardware Results

The recorded values through which each sign has been recognized by our proposed model are as follows:

| Yes | No |
|---|---|
| **thumbValue >= 8 && <= 9.** | **thumbValue >= 8 && <= 9** |
| **indexValue >= 4 && <= 5** | **indexValue >= 4 && <= 5** |
| **middleValue >= 3 && <= 4** | **middleValue >= 3 && <= 5** |
| **ringValue >= 4 &&  <= 5** | **ringValue >= 4 && <= 5** |
| **acceleration.x >= -5 && x <= 6** | **acceleration.x >= 3 && <= 9** |
| **acceleration.y >= 5 &&.y <= 11** | **acceleration.y >= -10 && y <= -3** |
| **acceleration.z >= -6 &&.z <= 6** | **acceleration.z >= -6 && z <= 3** |

**Table 5.0 Recorded Values for Yes and No Sign**

**Figure 5.1: Yes Sign**

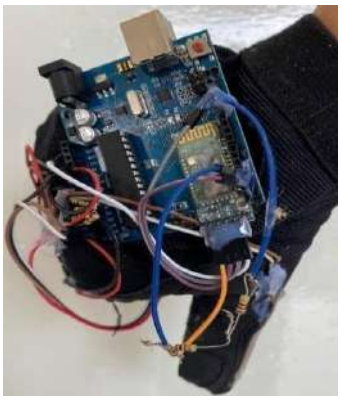


**Figure 5.2: No Sign**

| Okay | Victory |
|---|---|
| thumbValue >= 6 && <= 8 | thumbValue >= 5 && <= 7 |
| indexValue >= 7 && <= 9 | indexValue >= 7 && <= 10 |
| middleValue >= 6 && <= 9 | middleValue >=5 && <= 8 |
| ringValue >= 7 && <= 10 | ringValue >= 5 && <= 7 |
| acceleration.x >= -9 && <= 0 | acceleration.x >= -10 && <= -6 |
| acceleration.y >= -8 && <= 3 | acceleration.y >= -4 &&y <= 6 |
| acceleration.z >= -9 &&z <= 0 | acceleration.z >= -6 && z <= 2 |

**Table 5.1 Recorded Values for Okay and Victory Sign**

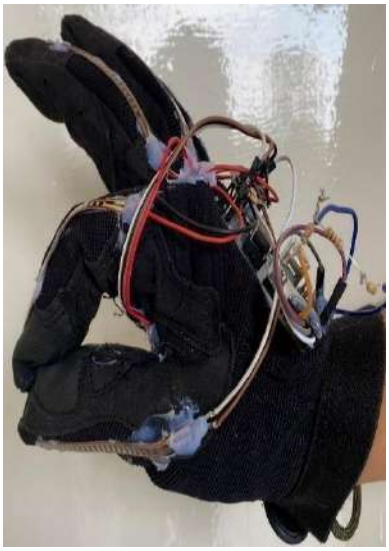


**Figure 5.3: Okay**

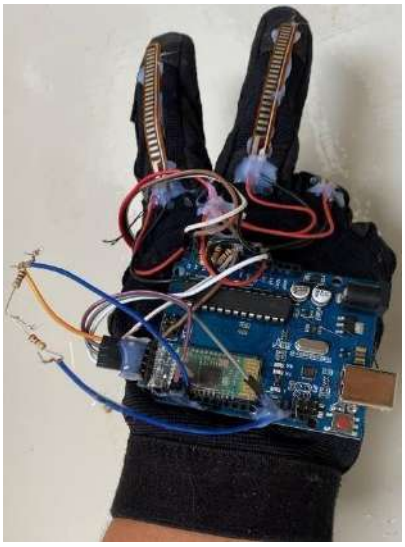


**Figure 5.4: Victory**

## 5.2 Software Results

To make our gesture recognition approach sensor-free, we have utilized PyCharm software for the implementation of the gesture recognition algorithm. The following images display the results of our implemented algorithm:


**Figure 5.5: Image depicts fist Sign**


**Figure 5.6: Image depicts Victory Sign**


**Figure 5.7: Image depicts thumbs up Sign**

# CHAPTER 6

# 6 Conclusion and future work

## 6.1 Thesis Work

Hand Gesture Vocalizer system represents a significant advancement in assistive technology, offering a transformative solution for individuals who are deaf and dumb to communicate effectively. By leveraging the power of hand gesture recognition and vocal synthesis, the system bridges the gap between sign language and audible speech, empowering users to express themselves with greater ease and clarity. The system's architecture, encompassing the hand gesture recognition module, vocal synthesis module, and user interface, provides a seamless and intuitive user experience. The hand gesture recognition module utilizes state-of-the-art computer vision techniques and machine learning algorithms to accurately capture and interpret a wide range of hand gestures. This enables users to convey their thoughts, emotions, and needs through intuitive movements, which are then translated into synthesized speech by the vocal synthesis module.

The implementation of the Hand Gesture Vocalizer system holds tremendous promise for enhancing the quality of life for individuals who are deaf and dumb. It opens up new avenues for communication in various settings, such as educational institutions, workplaces, and social interactions. With this technology, users can participate more actively in conversations, engage in real-time interactions, and access information more efficiently.While the current system has demonstrated its capabilities, there are opportunities for future development and improvement. Ongoing research can focus on refining the gesture recognition algorithms to enhance accuracy, robustness, and adaptability to different user profiles. Additionally, expanding the gesture vocabulary and exploring real-time feedback mechanisms would further enhance the system's usability and user experience.Furthermore, the integration of the Hand Gesture Vocalizer system with mobile devices would significantly increase its accessibility and portability. This would allow users to carry the system with them, empowering them to communicate effectively wherever they go.

Additionally, the integration of Raspberry Pi, a versatile single-board computer, has added another layer of functionality and portability to the hand gesture vocalizer. The compact

and energy-efficient nature of Raspberry Pi makes it an ideal choice for deploying the application in real-world scenarios. Its ability to handle complex computations and interact with external devices has been instrumental in achieving the desired performance and functionality of the system.By harnessing the power of PyCharm and Raspberry Pi, the hand gesture vocalizer has successfully addressed the challenges faced by individuals with hearing and speech impairments. The application's ability to interpret hand gestures, convert them into vocalized speech, and facilitate seamless communication represents a significant milestone in assistive technology.

This thesis work aims to address the communication challenges faced by individuals who are deaf and dumb through the design and implementation of the Hand Gesture Vocalizer system. By combining hand gesture recognition with vocal synthesis, this innovative solution has the potential to revolutionize communication and empower individuals with hearing and speech.The Hand Gesture Vocalizer system has the potential to break barriers and enable meaningful communication for individuals who are deaf and dumb. It represents a significant step towards inclusivity and empowerment, transforming the lives of countless individuals by giving them a voice and the ability to express themselves fully. As technology continues to evolve, we can anticipate even more sophisticated and user-friendly iterations of the Hand Gesture Vocalizer system, further enhancing the communication experience and fostering greater inclusivity in society.

## 6.2 Future Work

There are several avenues for future research and improvement. The following recommendations outline potential directions for further study:

1. **Expansion of Gesture Vocabulary:** The current system recognizes a predefined set of hand gestures. Future work can focus on expanding the gesture vocabulary to encompass a wider range of gestures, allowing users to express themselves more comprehensively.
2. **Improving Accuracy:** Enhancements in computer vision algorithms and machine learning techniques can be explored to improve the accuracy of gesture recognition.

Advanced deep learning models and data augmentation techniques can be employed to achieve higher recognition rates.

3. **Integration with Natural Language Processing:** Integrating natural language processing techniques can enable the system to understand and generate more complex and contextually relevant vocalizations. This would enhance the user experience and facilitate more meaningful communication.
4. **Mobile Application Development:** The hand gesture vocalizer can be further developed into a mobile application, making it more accessible and portable. Integration with smartphones and tablets would allow users to carry the system with them, enabling real-time communication in various settings.
5. **User Feedback and Iterative Design:** Conducting user studies and gathering feedback from the deaf and dumb community is crucial for refining and tailoring the system to their specific needs. Continuous iteration based on user input will ensure the development of a more user-friendly and effective solution.

# References


**1.** https://www.ijprse.com/2020/Vol1_Iss8_November20/IJPRSE_V1I8_6.pdf.

**2.** Ahmed, S. F., Ali, S. M. B., & Qureshi, S. S. M. (2010, November). Electronic speaking glove for speechless patients, a tongue to a dumb. In *2010 IEEE conference on sustainable utilization and development in engineering and technology* (pp. 56-60). IEEE.

**3.** Verma, M. P., Shimi, S. L., & Chatterji, S. (2014). Design of smart gloves. *International Journal of Engineering Research and Technology (IJERT) ISSN*, 2278-0181. .

**4.** https://techvidvan.com/tutorials/hand-gesture-recognition-tensorflow-opencv/

**5.** https://maker.pro/arduino/tutorial/how-to-interface-arduino-and-the-mpu-6050-sensor

***6.*** Shrivastava, A., Mishra, A., & Pandey, M. (2018). "American Sign Language recognition using image processing and machine learning*." 2018 IEEE 4th International Conference On Big Data Analytics (ICBDA), 67-72.*

**7.** Yang, Y., Gong, Y., & Yang, Y. (2019). "Real-time sign language recognition using a convolutional neural network-based multi-modal system." Sensors, 19(17), 3631.

**8.** Husnain, M., Han, J., & Gu, Y. (2020). "A robust hand gesture recognition system based on convolutional neural network and deep learning." IEEE Access, 8, 60773-60784.

**9.** Kim, J. J., Kim, Y. S., & Kim, J. H. (2019). "A real-time Korean sign language recognition system using a convolutional neural network." Electronics, 8(12), 1446.

**10.** https://www.raspberrypi.org/documentation/hardware/camera/

**11.** Arya, V., & Patni, P. (2020). "Sign Language Recognition Using Machine Learning." 2020 IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES), 1-5.

**12.** Zhang, J., Cheng, S., Hu, L., & Wang, Y. (2018). "Real-time sign language recognition using convolutional neural networks from leap motion controller data." IEEE Access, 6, 66662-66670.

**13.** Venkataramanan, A., Sardar, M., & Singh, H. (2019). "Real-time sign language recognition using deep learning and computer vision." Procedia computer science, 165, 365-372.

**14.** Erol, H., Bebis, G., & Nicolescu, M. (2007). "Vision-based hand pose estimation: A review." Computer Vision and Image Understanding, 108(1-2), 52-73.