

DEAF, BLIND & DUMB VERBALIZER COMMUNICATION SYSTEM

**A Project Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of**

BACHELOR IN TECHNOLOGY

by

**Abhijeet Anand (1612010003)
Mohammad Umair (1612010052)
Mohd Shahrukh (1612010053)**

Under the Supervision of

Dr. Shachi Mall

Mr. Vijendra Pratap Singh

Institute of Technology & Management, Gorakhpur



to the

Faculty of Department of Computer Science & Engineering

**DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY
(Formerly Uttar Pradesh Technical University, Lucknow)**

June, 2020

DECLARATION

I hereby declare that the work presented in this report entitled "**Deaf, Blind & Dumb Verbalizer Communication System**", was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

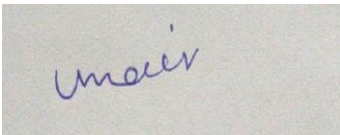
Name: **Abhijeet Anand** (1612010003)

Branch: Computer Science & Engg.



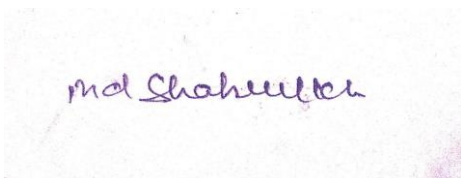
Name: **Mohammad Umair** (1612010052)

Branch: Computer Science & Engg.



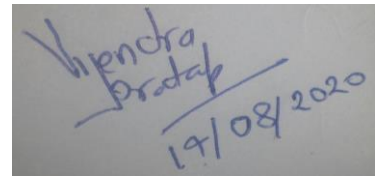
Name: **Mohd Shahrukh** (1612010053)

Branch: Computer Science & Engg.



CERTIFICATE

Certified that **Abhijeet Anand** (1612010003), **Mohammad Umair** (1612010052) and **Mohd Shahrukh** (1612010053) have carried out the project work presented in this project entitled “**Deaf, Blind & Dumb Verbalizer Communication System**” for the award of Bachelor of Technology in Computer Science and Engineering from Dr. A. P. J. Abdul Kalam Technical University (AKTU), Lucknow. The project embodies results of original work, and studies are carried out by the student himself and the contents of the project do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.



Mr. Vijendra Pratap Singh

Assistance Professor
Deptt. of Computer Sc. & Engg.
Institute of Technology and Management,
GIDA Gorakhpur

Date: 14/08/2020

ABSTRACT

Gesturing is an instinctive way of communicating to present a specific meaning or intent. Therefore, research into sign language interpretation using gestures has been explored progressively during recent decades to serve as an auxiliary tool for deaf and mute people to blend into society without barriers. Sign language is a natural way for communication between normal and dumb people, but often they find difficulty in communicating with normal people as we don't understand their sign language. Therefore, there always exists a language barrier. To minimize this barrier, we propose a device which can convert their hand gestures into voice which a normal person can understand. This device consists of a Wireless Glove, consisting of flex sensors and accelerometer. These sensors sense the movement of hands and fingers. This system consists of a speech synthesizer circuit which converts these movements of hand into real time speech output and a display will give the text for the corresponding gesture.

Index Terms—Wireless Glove, Flex Sensor, Accelerometer, Gesture recognition, machine learning, sign language recognition

ACKNOWLEDGEMENTS

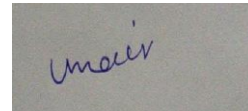
Whenever a module of work is completed, there is always a source of inspiration. We always find my parents as my torch bearers. While completing this task, we realized from our inner core that Rome was not built in day. We found a stack of project reports in the library of ITM Gorakhpur library. These reports are the landmarks for us on the way of this task. The presented report is an effort of day and night works. Selection is always tough; undoubtedly, we are accepting this fact. Initially there were a lot of doubts in my mind regarding the selection of topic for the project. Finally, we have selected this topic as it is from my favorite stream and the core of this branch. We are sincerely thankful to **Dr. Shachi Mall** for her support. In the same connection we would like to thanks **Mr. Vijendra Pratap Singh** and **Mr. Satyam Saini**.

We express our gratitude and thanks to all the faculties and staff member of Computer Science & Engineering department for their sincere cooperation in furnishing relevant information to complete this project report well in time successfully.

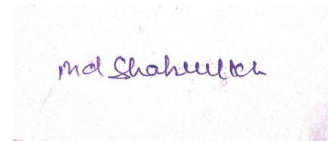
Finally, our greatest debt is to our parents, our family for their enduring love, support and forbearance during our project work.



Abhijeet Anand (1612010003)



Mohammad Umair (1612010052)



Mohd Shahrukh (1612010053)

TABLE OF CONTENTS

	Page No.
DECLARATION	ii
CERTIFICATE	iii
ABSTRACT	iv
ACKNOWLEDGEMENTS	v
TABLE OF CONTENTS	vi-vii
LIST OF TABLES	viii
LIST OF FIGURES	ix- x
CHAPTER 1 INTRODUCTION	1-7
1.1 INTRODUCTION	1
1.2 METHODOLOGY	5
1.3 EXPECTED OUTCOME OF THE PROJECT	6
1.4 MAJOR INPUTS (INFRASTRUCTURE REQUIRED)	6
1.5 USER REQUIREMENTS AND DESIGN SPECIFICATION	6
CHAPTER 2 LITERATURE SURVEY	8-13
2.1 LITERATURE SURVEY	8
2.2 REVIEW OF THE PREVIOUS WORK	8
2.3 SIGNIFICANCE OF RESEARCH WORK	10
2.4 RELATED WORK	10
CHAPTER 3 MODULE DESCRIPTION	14-28
3.1 DATA COLLECTION MODULE	14
3.2 PROCESSING MODULE	15
3.3 APPLICATION MODULE	15
3.4 ALGORITHM: SIGN CLASSIFIER	16
3.5 INTRODUCTION TO SVM	16
3.5.1 Classification	17

3.6	FLOWCHART	18
3.7	ER DIAGRAM	19
3.8	USE CASE DIAGRAM	20
3.9	DATA FLOW DIAGRAM	21
3.10	DECISION TREE	23
3.11	DECISION TABLE	24
3.12	CLASS DIAGRAM	26
3.13	ACTIVITY DIAGRAM	27
3.14	COMPONENT DIAGRAM	27
CHAPTER 4	DATA COLLECTION MODULE	29-52
4.1	INTRODUCTION: DATA COLLECTION MODULE	29
4.2	APPROACH	29
	4.2.1 Flex Sensor	29
	4.2.2 Three –Axis Accelerometer	32
	4.2.3 Wi-Fi Module	36
	4.2.4 Microcontroller (Arduino Nano)	39
4.3	INTRODUCTION TO ARDUINO IDE	43
4.4	DATA COLLECTION CODE	46
4.5	OUTPUT	47
CHAPTER 5	DATA PROCESSING MODULE	53-65
5.1	INTRODUCTION: DATA PROCESSING MODULE	53
5.2	INTRODUCTION TO MATLAB	54
5.3	INTRODUCTION TO ANACONDA	56
5.4	TRAINING CODE	58
5.5	PRINTOUT	61
CHAPTER 6	CONCLUSION AND FUTURE SCOPE	66
6.1	CONCLUSION	66
6.2	FUTURE SCOPE	66
	REFERENCES	67

LIST OF TABLES

	Table Name	Page No.
2.1	Review of the Previous Work	8
3.1	Representation of Components	22
3.2	Decision Table	25
4.1	Pin Configuration	40
4.2	Arduino Nano Technical Specifications	41
4.3	Data Set	47

LIST OF FIGURES

	Figure Name	Page No.
1.1	The ASL Alphabet, according to the Indiana Institute of Disability and Community	4
1.2	Disabled Population by sex and residence India, 2011	5
1.3	Disabled Population by Type of Disability India, 2011	5
2.1	Flowchart of Sign Language Recognition System using CMOS	11
2.2	The Experimental Setup	12
2.3	Alphabet displayed on the screen	13
3.1	Classification	17
3.2	Flowchart construction	18
3.3	FlowChart	19
3.3	ER Diagram	20
3.4	Use-Case Diagram	21
3.5	DFD Level-0	23
3.6	Decision Tree	24
3.7	Class Diagram	26
3.8	Activity Diagram	27
3.9	Component Diagram	28
4.1	Flex Sensor	29
4.2	Flex Sensor pins	30
4.3	Working of Flex Sensor	31
4.4	Three-Axis Accelerometer	32
4.5	ADXL345 Breakout Board	33
4.6	Axes of measurement for a triple axis accelerometer	33
4.7	Working of Accelerometers	34
4.8	The ADXL362 Triple Axis Accelerometer can measure $\pm 2g$, $\pm 4g$, and $\pm 8g$.	35
4.9	Wi-fi Module	36

4.10	ESP8266	37
4.11	ESP8266 Pinout	38
4.12	Microcontroller	40
4.13	Arduino NANO Pinout	41
4.14	Arduino IDE	44
4.15	Arduino IDE Home Page layout	45
4.16	Example of Arduino IDE : Blink	46
4.17	Snap of data being collected	51
4.18	Simulating the data	52
5.1	Training data	53
5.2	Home page of MATLAB	55
5.3	Anaconda Navigator	58

CHAPTER 1

INTRODUCTION

1.1 Introduction

The Sign language is very important for people who have hearing and speaking deficiency generally called Deaf and Mute. It is the only mode of communication for such people to convey their messages and it becomes very important for people to understand their language.

Sign language plays a vital role for person with disability (deaf and dumb) to communicate among themselves or with normal people in a non-verbal manner. Gestures are the primary method to convey messages, which are usually conducted in a three-dimensional space, known as a signing space [1], through an integration of manual and non-manual signals. Manual signals commonly correspond to hand motions and hand posturing, whereas non-manual signals correspond to an external appearance such as mouth movements, facial expressions, and body orientation [2]. Nevertheless, sign language has not been standardized globally. Each nation has developed its own sign language, such as the American Sign Language (ASL) and Germany Sign Language (GSL). However, each sign language varies slightly within different regions of the same country. Hence, it can be a challenge to develop a standardized sign language interpretation system for use worldwide. In a previous study, sign languages have been recognized using two major techniques, i.e., vision and non-vision approaches [3].

In fact, vision method is the major technique applied for sign recognition in the past decades. A system that uses a camera to observe the information obtained through hand and finger motions is the most widely adopted visual-based approach [4]. Tremendous effort and study has gone into the development of vision-based sign recognition systems worldwide. Indeed, vision-based gesture recognition systems can be subdivided into direct and indirect approaches.

A direct approach detects the hand gestures based on the RGB color spaces of the skin color. For instance, Goyal et al. [5] identified Indian Sign Language (ISL) using the scale invariance Fourier transform (SIFT) algorithm by searching the matched key points between the input image and images stored in a database.

A similar method was also applied by More et al. [8] using the SIFT algorithm, which further reduces the dimensions of the feature vector using a principal component analysis (PCA) algorithm for speeding up the processing time. To detect the dynamic hand gestures used in Japanese Sign Language (JSL), Murakami et al. [9] proposed the use of recurrent neural networks capable of recognizing the JSL finger alphabet, which has 42 symbols. In contrast, Chowdary et al. [10] used a simple scanning method to compute the orientation and movement of fingers in binary converted images captured from a web camera. Khan et al. [11] proposed a more sophisticated gesture recognition system using digital images, including

image filtering (pre-processing), image segmentation, color segmentation, skin detection (finger and hand detection using binary images), and template matching.

Meanwhile, an indirect approach identifies the fingers and hand gestures based on the RGB color spaces segmented based on different colors for each finger using a data glove. A possible segmentation method using RGB color spaces along with a hand glove for gesture detection was proposed by Siby et al. [12]. This method utilizes the RGB color space values extracted from a captured hand gesture image of a data glove, and compares the values with those stored in a database.

The exploitation of a vision-based method is greatly affected by the processing of the images, such as image filtering, background cancellation, color segmentation, and boundary detection. For instance, diverse and uncontrolled background images can influence the skin color segmentation or movement detection. Indeed, many researchers have failed to address these complications, and no solid solutions have yet been proposed. Consequently, a non-vision based method is an alternative approach. This method typically utilizes flex and motion sensors to measure the flexion of fingers and the orientation of the hand, respectively.

Dawane et al. [13] attached five flex sensors on a glove with respect to each finger to identify hand gestures by matching the motions with those in a stored motion database. Preetham et al. [14] also proposed a similar technique of using flex sensors, but mapped the sensor data to a character set, which was implemented using a minimum mean square error (MMSE) algorithm for gesture recognition. The results are displayed as text on an LCD screen. This technique was improved by Patil et al. [15]; here, the bending of each sensor is further divided into three flexions, namely, a complete bend (finger close), partial bend, and straightening (finger open). Each ASL alphabet is then mapped according to the bend flexions to be used for template matching.

On the other hand, inertial motions of hand or fingers are alternative approach for gestures recognition. Kim et al. [16] developed a glove-based 3-D hand motion tracking and gesture system that consists of three tri-axis accelerometer sensors placed on the thumb, middle finger, and back of the hand, respectively. The glove is able to detect simple rule-based hand gestures (e.g., scissor, rock, and paper) based on two different angular positions of the sensors: horizontal (z-axis) and vertical (x-axis) gestures. Lu et al. [17] implemented a YoBu glove with total of 18 inertial motion unit (IMU) sensors. Each finger consists of three IMMUs at each joint with a total of 15 for the five fingers. The remaining IMU sensors are placed on the arm, forearm, and upper arm. An extreme kernel-based learning machine is implemented to identify specific gestures based on a total of 54-dimension extracted features. On the other hand, Lim et al. [18] proposed a novel method that uses a small-sized infrared optic sensor, developed as a virtual button, to observe finger flexion patterns on the wrist caused by moving fingers. Five dynamic gestures were proposed: “bye,” “hi,” “hold,” “release,” and “wave,” all of which are detected using a hidden Markov model (HMM) algorithm. Xie et al. [19] developed an accelerometer based smart ring to detect eight basic and twelve complex gestures in 2-D space. A segmentation algorithm is utilized to identify

subject gestures which are further encoded by a Johnson code. Hsu et al. [20] presented an inertial-based digital pen to recognize the handwriting and gestures with dynamic time warping (DTW) method. The pen is hold by the user when writing the numerals or English lowercase letters and further transmitted wirelessly to a computer for online recognition. The similar hand gesture recognition approach with DTW method is proposed by Yin et al. [21] that adopted training-free method which did not require training samples. The gesture recognition process is carried out by first extracting the features followed by a robust template matching method. Besides that, Galka et al. [22] proposed an accelerometer glove for sign language recognition which consisted of seven active three-axis acceleration sensors with five located on the fingers (one sensor on each finger), one on the arm and one on the wrist. The sign recognition model is defined by a HMM and a parallel HMM approach. Cai et al. [23] also developed a wireless data glove type with four fingers button. The raw sensor data are converted into movement, acceleration, rotation and other features for gesture recognition. Meanwhile, Liu et al. [24] presented an interesting idea to integrate inertial and vision depth sensors with HMM model to recognize six hand gestures, including “wave”, “hammer”, “punch”, “draw X”, “circle” and “other” gestures. However, the proposed system is not feasible in practical usage as both sensors needed to present for high accuracy gestures recognition. Nevertheless, a simple data glove with three-axis accelerometer, magnetometers and gyroscopes is proposed by Kim et al. [25] which converted the sensor data into angle data for sign language recognition. Sousa et al. [26] presented a GyGSLA system, a wearable glove that aimed to help inexperienced people in learning the new Portuguese sign language alphabet which is tested with three completely inexperienced sign language subjects. The similar study is also performed by Caporusso et al. [27] by introducing dbGLOVE, a wearable device for supporting deaf-blind people to communicate with others. Meanwhile, physiological sensors such as sEMG are also another popular gesture recognition technique. Lu et al. [28] proposed a score-based sensor fusion scheme using four sEMG sensors and a three-axis accelerometer connected to a mobile application to realize gestures-based real-time interaction. A set of 4 small-scale gestures, 15 large-scale gestures and user defined personalized gestures are proposed in the study. A similar method is also proposed by Wu et al. [29] with four sEMG and an inertial sensor that placed on the wrist to detect 40 most commonly used ASL words. The study observed that only a single channel of sEMG located on the wrist is sufficient for the ASL recognition. Likewise, Wu et al. [30] fused the information from an inertial sensor and sEMG sensor which are placed on a wearable system to recognize 80 commonly ASL signs with selected feature subset and processed by a support vector machine classifier.

This paper aims to lower the barrier in communication with normal people. The main aim of the proposed system is to develop a cost effective system which can give voice to voiceless person with the help of **Smart Gloves**. It means that using smart gloves communication will not be a barrier between two different communities and they will be able to communicate easily with the normal person. This study aims at the development of a sign language interpretation system by analyzing hand and finger gestures from a smart wearable device. The finger gestures are observed through the flexion of the flex sensors, whereas the hand

gestures are examined based on the hand motion through the orientation derived from an inertial motion sensor. The gestures are recognized using a support vector machine (SVM) model implemented in the wearable device.

In the sign language alphabet, most of the letters are defined purely by the flexion of each finger

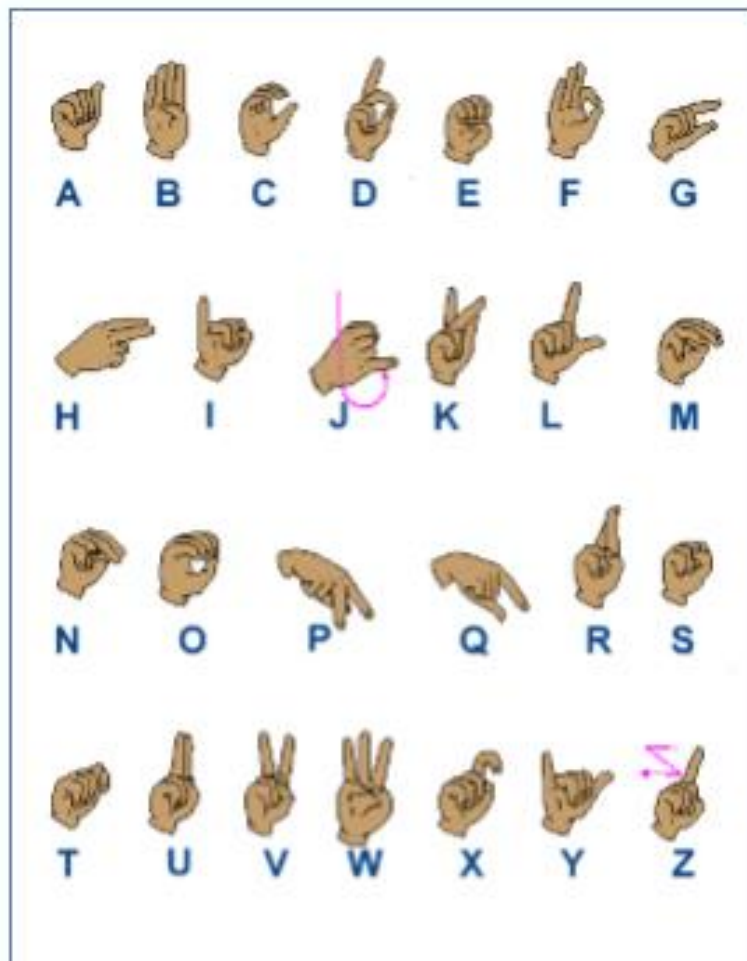


Fig. 1.1 The ASL Alphabet, according to the Indiana Institute of Disability and Community

India being the one of the largest population about 133.92 crores unfortunately consists of 40 to 80 million persons with disability, and it's a big challenging problem to overcome the standard of living of those people with the help of technology

The following data represents the population of person with disability:

Disabled Population by Sex and Residence India, 2011			
Residence	Persons	Males	Females
Total	26,810,557	14,986,202	11,824,355
Rural	18,631,921	10,408,168	8,223,753
Urban	8,178,636	4,578,034	3,600,602

Fig. 1.2 Disabled Population by sex and residence India, 2011

Disabled Population by Type of Disability India : 2011			
Type of Disability	Persons	Males	Females
Total	26,810,557	14,986,202	11,824,355
In Seeing	5,032,463	2,638,516	2,393,947
In Hearing	5,071,007	2,677,544	2,393,463
In Speech	1,998,535	1,122,896	875,639
In Movement	5,436,604	3,370,374	2,066,230
Mental Retardation	1,505,624	870,708	634,916
Mental Illness	722,826	415,732	307,094
Any Other	4,927,011	2,727,828	2,199,183
Multiple Disability	2,116,487	1,162,604	953,883

Fig. 1.3 Disabled Populations by Type of Disability India, 2011

1.2 Methodology

To understand the implications of Indian Sign Language, it is important to have a sense of the scope of the language, as well as the current methods of communicating between ISL users and non-users and how these methods accurately reflect the grammar and mechanics of the language. These factors influenced technological components of the glove throughout development of this proof-of concept. This includes the underlying theory, practices, and methods of using relevant sensors, digital and analog signal conversion, processing, and output methods. By understanding the behaviors and applications of the different components at a fundamental level, one can more easily understand how they can be optimally used for the creation of this product.

The main purpose is to design a smart wearable glove, which can be used in hand. The proposed wearable system outperforms the existing method, for instance, although background lights, and other factors are crucial to a vision-based processing method, they are not for the proposed system [6].

- With the help of sensors, the actual movement of figures will be recorded.
- Corresponding patterns will be concluded.
- Accordingly, the conversion into voice is done.

1.3 Expected Outcome of the Project

Projects have aimed to develop technology to facilitate communication between hearing and non-hearing people. The outcome is system generated voice after analyzing the data obtained from the sensors which monitors the following:

- Movements of the hand in 3D space.
- The resistance in the flex sensors on bending the figures at specific angles.

1.4 Major Inputs (infrastructure) required

The projects are a combination of various hardware to accept the data and software which manipulates the input to produce the output.

Hardware

- Flex Sensors, Triple Axis Accelerometer, Wi-Fi Module, Push Button, LEDs, Speaker, Battery, Arduino Nano, Raspberry Pi, Wires.

Software

- Matlab, Anaconda, Arduino
- OpenCV: It is an open source computer vision programming functions library aimed at developing applications based on real time computer vision technologies.

1.5 User Requirements and Design Specifications

Several difficulties are faced by Indian-born ISL users when communicating with those of the hearing community. ISL speaking people understand these difficulties best, so requirements that they explicitly state take priority. These requirements will later be taken into consideration when deciding on product specifications.

Explicit Requirements

Based on multiple interviews with ISL users, there are several explicitly-stated requirements of a sign-language translating glove. These include the speed of translation and the accuracy

of the device. This and the glove accuracy, or if and how often it can detect and translate the correct gesture as the user signs it, determine the reliability of the glove. The device accuracy also involves the device's ability to incorporate the various motions that ISL signs involve. This includes not only the rotation and flexion of hands and fingers, but also the orientation of the hand.

Implicit Requirements

In addition to the requirements specified by ISL users, the team has defined five requirements of the device that increase the prototype's usefulness to ISL users. Several aspects and interactions of the device with its users and environment were taken into consideration. The requirements are as follows:

- **Easy to Use** - Any complications in its user interface would inhibit the glove's use in everyday life. The user should be able to begin translation without much difficulty or delay. Each translation should be done without any unnecessary button-pressing or other interfacing.
- **Portable** - The system should not be dependent on a computer or other attached system. It should be able to be brought almost anywhere the user goes, with the possible exception for underwater.
- **Affordable** - Not much financial aid is available for assistive devices. This device should be accessible by the average person by practical and affordable means.
- **Reliable** - There is a certain degree of accuracy that the device should maintain. This threshold has been decided by our team to be of about 90% accuracy. If the device does not accurately and consistently translate signs, then the user will resort to time consuming alternatives such as writing on pen and paper and the device will have no use.
- **Aesthetically Pleasing** - For marketability purposes, the device shall be aesthetically pleasing and easily wearable without any factors that hinder convenience during extended periods of time. This includes a smooth, professional appearance without any components that irritate, cut, bruise, or otherwise cause discomfort for the user.

CHAPTER 2

LITERATURE SURVEY

2.1 Literature Survey

Sign language plays a vital role for person with disability (deaf and dumb) to communicate among themselves or with normal people in a non-verbal manner. Gestures are the primary method to convey messages, which are usually conducted in a three-dimensional space, known as a signing space [1], through an integration of manual and non-manual signals. Manual signals commonly correspond to hand motions and hand posturing, whereas non-manual signals correspond to an external appearance such as mouth movements, facial expressions, and body orientation [2]. Nevertheless, sign language has not been standardized globally. Each nation has developed its own sign language, such as the American Sign Language (ASL) and Germany Sign Language (GSL). However, each sign language varies slightly within different regions of the same country. Hence, it can be a challenge to develop a standardized sign language interpretation system for use worldwide. In a previous study, sign languages have been recognized using two major techniques, i.e., vision and non-vision approaches [3].

In fact, vision method is the major technique applied for sign recognition in the past decades. A system that uses a camera to observe the information obtained through hand and finger motions is the most widely adopted visual-based approach [4].

A direct approach detects the hand gestures based on the RGB color spaces of the skin color. For instance, Goyal et al. [5] identified Indian Sign Language (ISL) using the scale invariance Fourier transform (SIFT) algorithm by searching the matched key points between the input image and images stored in a database.

This paper aims to lower this barrier in communication with normal person. The main aim of the proposed system is to develop a cost effective system which can give voice to voiceless person with the help of **Smart Gloves**. It means that using smart gloves communication will not be barrier between two different communities and they will be able to communicate easily with the normal person.

2.2 Review of the previous work

Table 2.1 Review of the previous work

Reference	Methodology	Conclusion	Limitation
[1] S. Goyal, I. Sharma and S. Sharma, "Sign	1. A direct approach detects the hand	1. Based on the RGB color spaces of the skin	1. Requirement of color space. 2. For every

language recognition system for deaf and dumb people,” Int. J. Eng. R. Technol., vol. 2, no. 4, pp. 382-387, Apr. 2013.	gestures. 2. Indian Sign Language (ISL) using the scale invariance Fourier transform (SIFT) algorithm by searching the matched key points between the input image and images stored in a database	color, detects hand gestures.	alphabet approximately 5 images at different angles and distances need to be stored
[2] S. P. More and A. Sattar, “Hand gesture recognition system using image processing,” presented at Int. Conf. Electrical, Electronics, Opt. Techniq., Chennai, India, Mar. 2016	1. A method was applied by More et al. using the SIFT algorithm. 2. It reduces the dimensions of the feature vector using a principal component analysis (PCA) algorithm for speeding up the processing time.	1. Similar to RGB based color space, which reduces dimension using PCA Algorithm	1. Large database is required in order to store the images.
[3] K. Murakami and H. Taguchi, “Gesture recognition using recurrent neural network,” in Proc. SIGCHI Conf. Human Factors Comput. Syst., New York, USA, 1991.	1. To detect the dynamic hand gestures used in Japanese Sign Language (JSL). 2. Murakami et al proposed the use of recurrent neural networks. 3. It was capable of recognizing	1. Use of Recurrent Neural Networks.	1. Mainly used for JSL. 2. Limited to 42 symbols only.

	the JSL finger alphabet (42 symbols).		
[4] T. Khan and A. H. Pathan, "Hand gesture recognition based on digital image processing using matlab," Int. J. Sci. Eng. R., vol. 6, no. 9, pp. 338-346, Sep. 2015.	1. Khan et al. proposed a more sophisticated gesture recognition system using digital images, image filtering (pre-processing), segmentation, skin detection (finger and hand detection using binary images), and template matching.	1. Uses image filtering, segmentation, color segmentation, and skin detection 2. Binary conversion is done.	1. It's incapable of measuring the motions of hands or figures.

2.3 Significance of Research Work

The main significance of the smart gloves is to convert the sign language into voice. It will help the person with disability to communicate easily with others. Indian sign language consists of various directions and combinations of figures which represent the different alphabets. With the help of hand gestures, data is taken as input and then analyzed.

In this paper, a smart sign language interpretation system using a wearable hand device is proposed to meet this purpose. This wearable system utilizes ten flex-sensors, and a three-axis inertial motion sensor to distinguish the characters in the Indian Sign Language alphabet. The entire system mainly consists of three modules: a wearable device with a sensor module and a processing module, and a display unit [6].

2.4 Related Work

Wherever using the concept of gestures, few systems had been developed in the past to recognize the gestures made using hands but with limitations of recognition rate and time which include:

1. **Using CMOS camera:** The proposed algorithm consisted of four major steps which are namely Image Acquisition, Feature Extraction, Orientation Detection and Gesture Recognition which is also shown in the below given Fig. 2.1. While deciding on the following algorithm it was observed that pre-processing steps that are to be applied on the

images for removal of noise in the background was not at all required and the approach was concluded to be simple and easy to implement. The steps of the methodology are further explained in details:

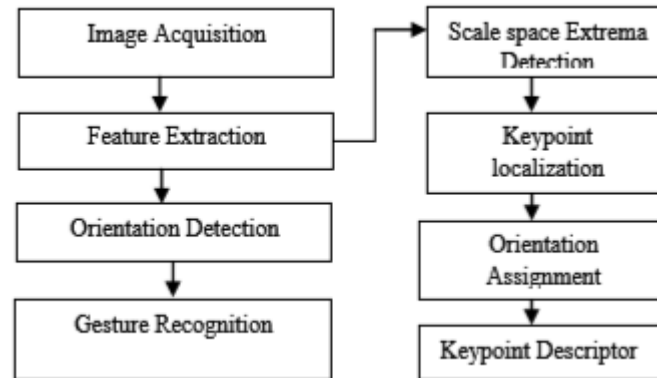


Fig. 2.1 Flowchart of Sign Language Recognition System using CMOS

2. **Using image processing technique:** This approach use 3D model description [3] for modeling and analysis of the hand shape. It searches for kinematic parameters and are requires by making 2D projection from 3D model of the hand to correspond edge images of the hand. But lot of hand features might be lost in 2d projection. Different types of methods are there including Volumetric and Skeletal type. Volumetric model deals with 3D visual appearance of human hand and usually used in real time applications. The problem is it deals with all the parameters of the hand which are huge dimensionality. To overcome volumetric hand parameter problem, skeletal method can be used. It limits the set of parameters to model the hand shape from 3D structure.

3. **Flex sensor based glove:** When a particular alphabet is shown using sign language, it involves bending of the fingers thereby resulting in the bending of the flex sensors; there occurs a change in resistance of the flex sensor. A range of voltage values are given for each bend of the flex sensor for every finger. Thus, for every alphabet, each flex sensor is given a particular range of voltage values so as to distinguish it from other alphabets. The tilt sensor is used to measure orientation of the hand and fingers. The angle at which the hand is bent is given a range of values to distinguish alphabets that have similar bends so as to reduce ambiguities. The voltage values generated by flex and tilt sensors are all analog in nature. These analog values are converted into digital format using the ADC of ARM 7. The digital voltage values generated are compared and analyzed with previously stored values in the ARM flash memory. The set of values that matches with the input values is selected and the alphabet corresponding to those values are given as the output of the ARM. This process is known as template matching, where raw input data collected is compared with predefined data to give the appropriate output. The alphabet so generated is transmitted to an Android device using a Wi-Fi module. The alphabet is displayed on the

Android device using the Blynk App where different widgets can be dragged and dropped based on the application of the user.



Fig. 2.2 The Experimental Setup



Fig. 2.3 Alphabet displayed on the screen

These techniques required lots of devices, database and collection of image sets. For every alphabet approximately 5 images at different angles and distances need to be stored in order to complete analysis of the dataset.

CHAPTER 3

MODULE DESCRIPTION

3.1 Data collection Module

For this study, a custom-made wearable device was taken to hold the hardware. The wearable device holder is printed using flexible filaments with good elasticity. These filaments enable functional hinges, joints, and shaped parts, allowing the device to fit different hand sizes. Five finger holders were also designed using a flexible filament placed on the first joint of each finger to hold the flex sensors. Similarly, these flexible holders can also accommodate different finger sizes of different users.

Different components will be used in order to collect the data of hand gesture:

Flex Sensor

Flex sensors and strain gauges are two types of sensors that measure the degree by which an object is bent. These devices take advantage of a material's resistance and how this changes as the material is subject to mechanical strain. Resistance is influenced by the length of a particular material (L), as well as its cross sectional area (A), charge carrier mobility (μ), and charge carrier density (N). Using these properties, the resistance of a material can be found using the formula in Equation 1 where q represents the number of Coulombs per carrier.

Equation for the resistance of a material:

$$R = L / qe * N * A * \mu \quad (3.1)$$

This formula can be applied to use a material's resistance like a sensor. By stretching a material, the length L increases, resulting in a larger resistance, while compressing or decreasing the value of L causes the resistance to become smaller.

The flex sensor consists of an internal resistor. The flex sensors are bidirectional in nature and hence it can detect change in resistance in both positive and negative directions. The internal circuitry of the flex sensor is a voltage divider circuit which gives different values of voltage based on the amount of bending of the fingers, due to the change in resistance inside the flex sensor.

Three-axis Accelerometer

The three-axis accelerometer is used to measure the orientation of the hand and fingers. The internal circuit of a three-axis accelerometer consists of one fixed capacitor and one variable capacitor. Due to the orientation or tilt of the hand a change in capacitance results in proportional voltage change.

Wi-Fi Module

ESP8266 is a 16 pin self- contained networking unit. These are low power devices which work on UART protocols and are programmed using AT commands. Serially the data is sent from the processor to this unit with the help of UART protocol.

When a particular alphabet is shown using sign language, it involves bending of the fingers thereby resulting in the bending of the flex sensors; there occurs a change in resistance of the flex sensor. A range of voltage values are given for each bend of the flex sensor for every finger. Thus, for every alphabet, each flex sensor is given a particular range of voltage values so as to distinguish it from other alphabets.

Microcontroller (Arduino Nano)

Arduino Nano is the main controller used in this project. It is a microcontroller based board on the Atmega 328k. It has 20 digital input and output pins with 7 pins for PWM.

3.2 Processing Module

To simplify and optimize the coding implementation, features are extracted from the sensor data and serve as inputs to the built-in SVM classifier to determine the sign language alphabet letters [6].

In this study, the signs are classified into 28 classes using a support vector machine (SVM) [7]. An SVM is a binary supervised learning classifier, that is, the class labels can only take the values of +1 and -1. The training procedure used a quadratic optimization algorithm to derive structural axes to separate the training dataset into n numbers of a hyperplane.

The classified sign gestures from the proposed smart wearable device are transmitted to the sign interpretation system.

3.3 Application Module

The service or system converts the received text into an output, which is played back concurrently by the mobile or laptop device speaker. In this study, the Android-based sign interpretation application is merely utilized for receiving classified sign gestures from the smart wearable device and further displaying the results on the screen. Also converting to speech simultaneously.

3.4 Algorithm: Sign Classifier

In this study, the signs are classified into 28 classes using a support vector machine (SVM) [7]. An SVM is a binary supervised learning classifier, that is, the class labels can only take the values of +1 and -1. The training procedure used a quadratic optimization algorithm to derive structural axes to separate the training dataset into nnumbers of a hyperplane. Assume the i -th training sample using

$$(x_i, y_i), y_i \in \{-1, +1\}, i=1, 2, 3, \dots, n, \quad (3.2)$$

Where x_i is the feature vector and y_i is the training label in accordance to the feature vectors of the i -th training datasets. The decision boundary is defined as

$$f(x) = w \cdot x - b, \quad (3.3)$$

Where the i -th feature is classified as positive (+1) if $f(x) > 0$, and negative (-1) if $f(x) < 0$. The separating hyperplane line is structured at $f(x) = 0$. The points positioned around the separating hyperplane line are known as support vectors (SVs) and their distance to the hyperplane line is known as the margin. Optimization of the SVM is calculated by finding the smallest distance among all SVs, as shown in (3), which is subject to (4).

$$\text{Min} |w|/2 \quad (3.4)$$

w, b

$$y_i(w \cdot x_i - b) \geq 1 \quad (3.5)$$

The values of +1 and -1 are expressed as correctly classified alphabet and incorrectly classified alphabet respectively.

The values of +1 and -1 are expressed as correctly classified alphabet and incorrectly classified alphabet respectively.

3.5 Introduction to Support Vector Machines

Support Vector Machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.

The advantages of support vector machines are:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- **Versatile:** different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

The support vector machines in scikit-learn support both dense (numpy.ndarray and convertible to that by numpy.asarray) and sparse (any scipy.sparse) sample vectors as input. However, to use an SVM to make predictions for sparse data, it must have been fit on such data. For optimal performance, use C-ordered numpy.ndarray (dense) or scipy.sparse.csr_matrix (sparse) with dtype=float64.

3.5.1 Classification

SVC, **NuSVC** and **LinearSVC** are classes capable of performing multi-class classification on a dataset.

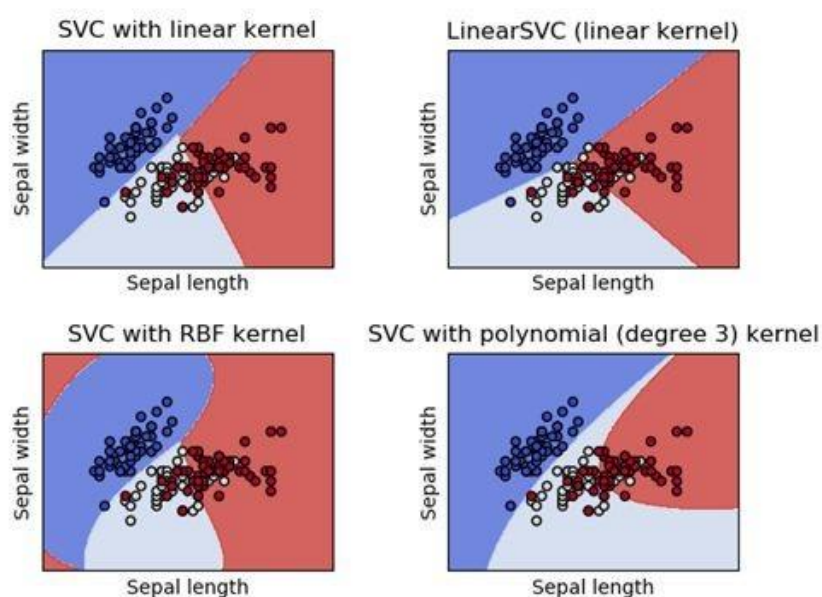


Fig. 3.1 Classification

SVC and NuSVC are similar methods, but accept slightly different sets of parameters and have different mathematical formulations (see section Mathematical formulation). On the other hand, LinearSVC is another implementation of Support Vector Classification for the case of a linear kernel. Note that LinearSVC does not accept keyword kernel, as this is assumed to be linear. It also lacks some of the members of SVC and NuSVC, like support_.

3.6 Flowchart

"A picture is worth a thousand words," but it's rather important to know which picture and which 1000 words. There is no question that graphical tools, such as the flowchart or box diagram, provide useful pictorial patterns that readily depict procedural detail. However, if graphical tools are misused, the wrong picture may lead to the wrong software. A flowchart is quite simple pictorially. A box is used to indicate a processing step. A diamond represents a logical condition, and arrows show the flow of control. Fig. 3.2 illustrates three structured constructs. The sequence is represented as two processing boxes connected by an line (arrow) of control. Condition, also called if then-else, is depicted as a decision diamond that if true, causes then-part processing to occur, and if false, invokes else-part processing. Repetition is represented using two slightly different forms. The do while tests a condition and executes a loop task repetitively as long as the condition holds true. A repeat until executes the loop task first, then tests a condition and repeats the task until the condition fails. The selection (or select-case) construct shown in the figure is actually an extension of the if-then-else. A parameter is tested by successive decisions until a true condition occurs and a case part processing path is executed.

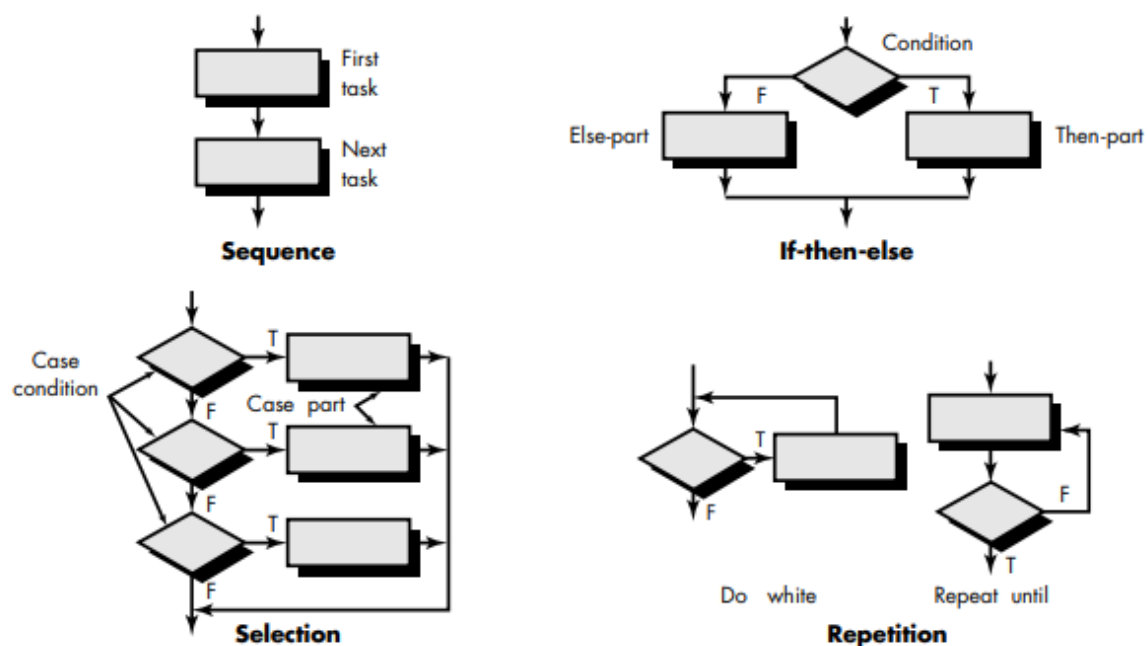


Fig. 3.2 Flowchart constructs.

Another graphical design tool, the box diagram, evolved from a desire to develop a procedural design representation that would not allow violation of the structured constructs. Developed by Nassi and Shneiderman [NAS73] and extended by Chapin [CHA74], the diagrams (also called Nassi-Shneiderman charts, N-S charts, or Chapin charts) have the following characteristics: (1) functional domain (that is, the scope of repetition or if-then-else) is well defined and clearly visible as a pictorial representation, (2) arbitrary transfer of

control is impossible, (3) the scope of local and/or global data can be easily determined, (4) recursion is easy to represent.

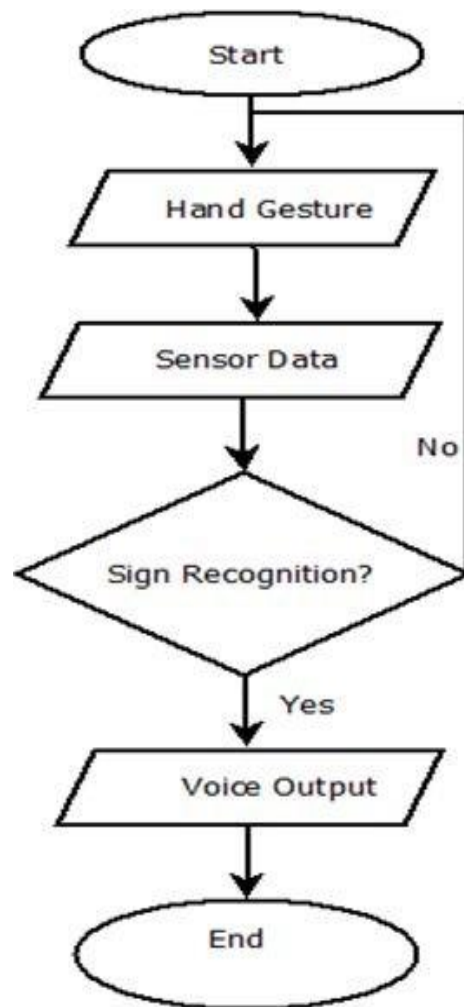


Fig. 3.3 Flowchart

3.7 ER Diagram

An Entity Relationship Diagram (ERD) shows the relationship of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities have attributes that define its properties.

It illustrates how "*entities*" such as people, objects, and concept related to each other with in a system. ER diagrams are most often used to design or debug relational databases in the fields of software engineering business information systems, educations and research. ER diagram, also known as ERD'S or ER models, they are used a defined set of symbols such as rectangles, diamonds ovals as connecting lines to depicts the interconnectedness of entities, relationship and their attributes.

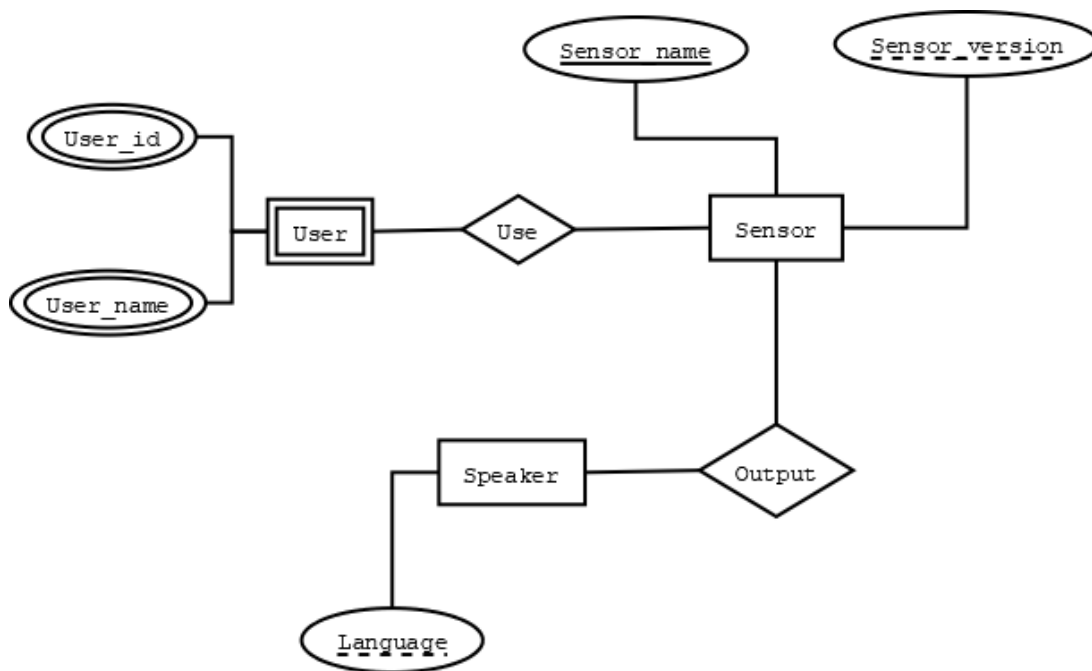


Fig. 3.3 ER Diagram

3.8 Use-Case Diagram

A use case diagram is usually simple. It does not show the details of the use cases.

- It only summarizes some of the relationships between use cases, actor and systems.
- It does not show the order of steps being performed.

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. It can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction. These internal and external agents are known as **actors**. So use case diagrams are consists of **actors, use cases and their relationships**. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

Purpose

The purpose of use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose.

So in brief, the purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.
- Show the interacting among the requirements are actors.

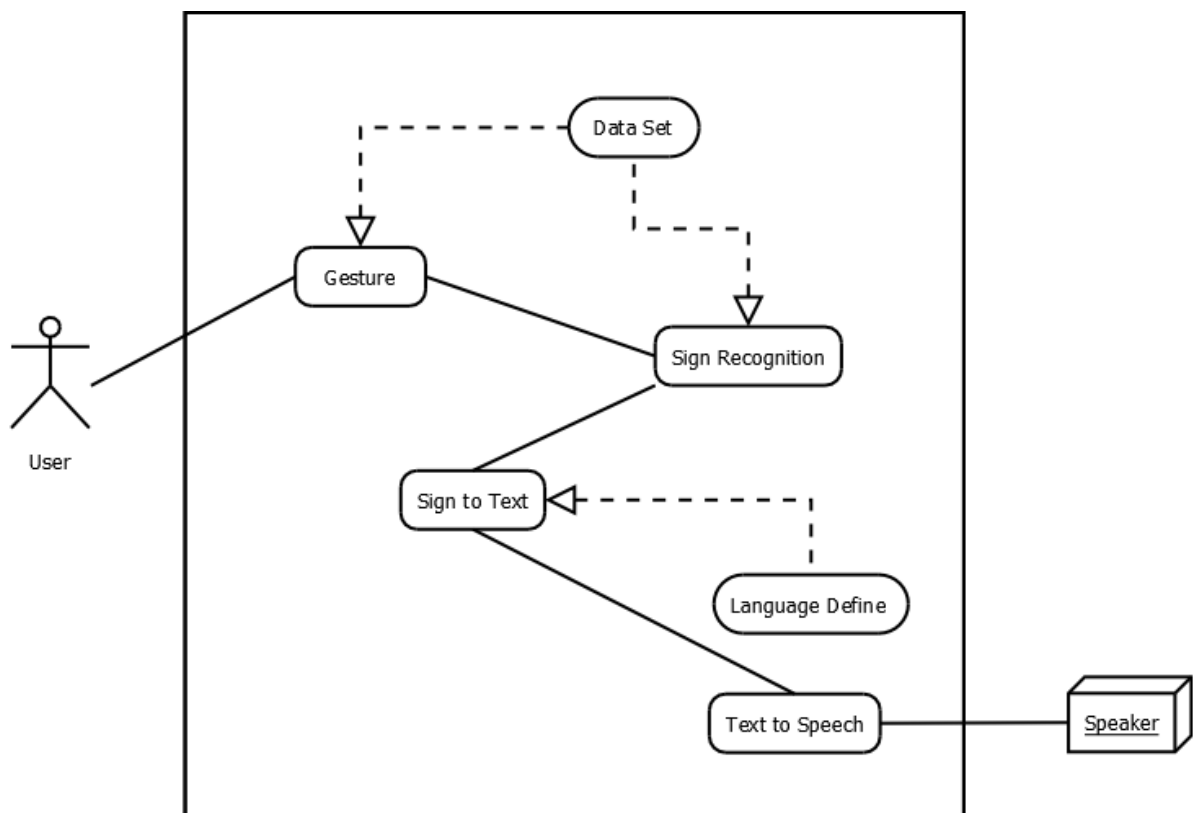


Fig. 3.4 Use Case Diagram for DFD

3.9 Data Flow Diagram (DFD)

A **DFD** illustrates how data is processed by a system in terms of inputs and outputs. As the name suggests, it focuses on the flow of information, where data comes from, where it goes and how it gets stored.

It represents the flows of data between different processes in a business. It is a graphical technique that depicts information flow and the transforms that are applied as data move from input to output. It provides a simple, intuitive method for describing business processes.

without focusing on the details of computer systems. DFDs are attractive technique because they provide what users do rather than what computers do.

Types of DFD

Data Flow Diagrams are either Logical or Physical.

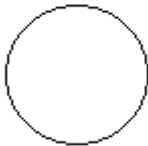
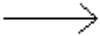
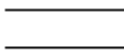

- **Logical DFD** - This type of DFD concentrates on the system process and flow of data in the system. For example in a Banking software system, how data is moved between different entities.
- **Physical DFD** - This type of DFD shows how the data flow is actually implemented in the system. It is more specific and close to the implementation

Representation of Components

DFDs only involve four symbols. They are:

- Process
- Data Object
- Data Store
- External entity

Table 3.1 Representation of Components

Shape	Description
	Process Transform incoming data flow(s) to outgoing flow(s).
	DataFlow Movement of data in the system.
	DataStore Data repositories for data that are not moving. It may be as simple as a buffer or a queue or as sophisticated as a relational database.
	ExternalEntity Sources of destinations outside the specified system boundary

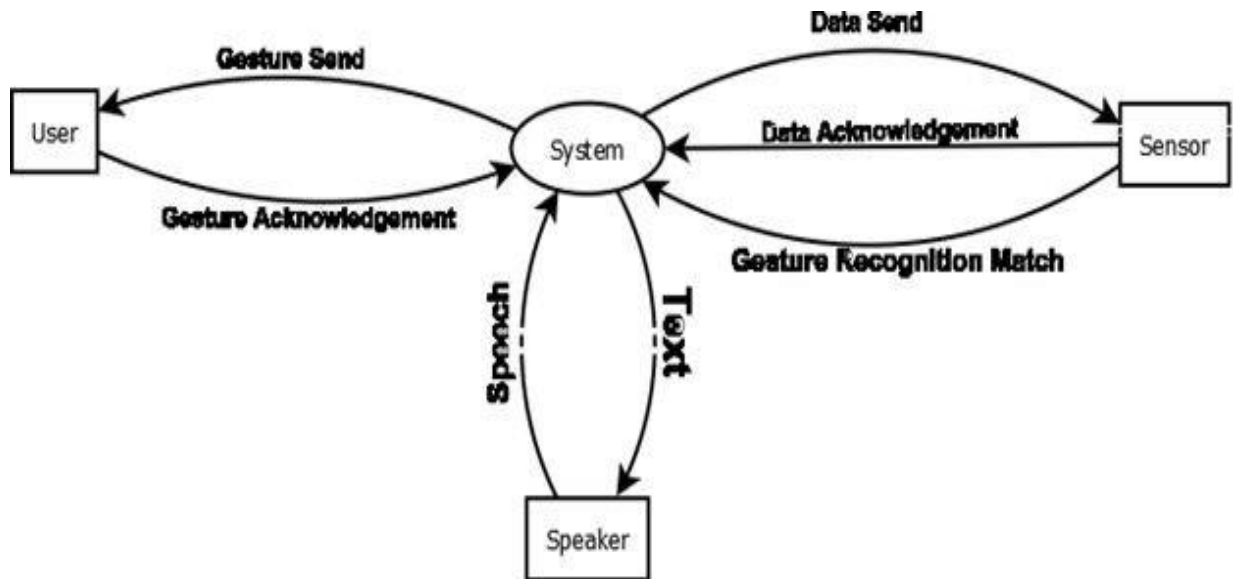


Fig. 3.5 DFD Level-0

3.10 Decision Tree

A decision tree gives a graphic view of the processing logic involved in decision making and the corresponding actions taken. The edges of a decision tree represent conditions and the leaf nodes represent the actions to be performed depending on the outcome of testing the condition.

Decision trees are helpful, not only because they are graphics that help us to 'see' what we are thinking, but also because making a decision tree requires a systematic, documented thought process. Often, the biggest limitation of our decision making is that we can only select from the known alternatives. Decision trees help formalize the brainstorming process so we can identify more potential Decision trees are commonly used in operations research and operations management. If, in practice, decisions have to be taken online with no recall under incomplete knowledge, a decision tree should be paralleled by a probability model as a best choice model or online selection model algorithm. Another use of decision trees is as a descriptive means for calculating conditional probabilities.

Decision trees, influence diagrams, utility functions, and other decision analysis tools and methods are taught to undergraduate students in schools of business, health economics, and public health, and are examples of operations research or management science methods.

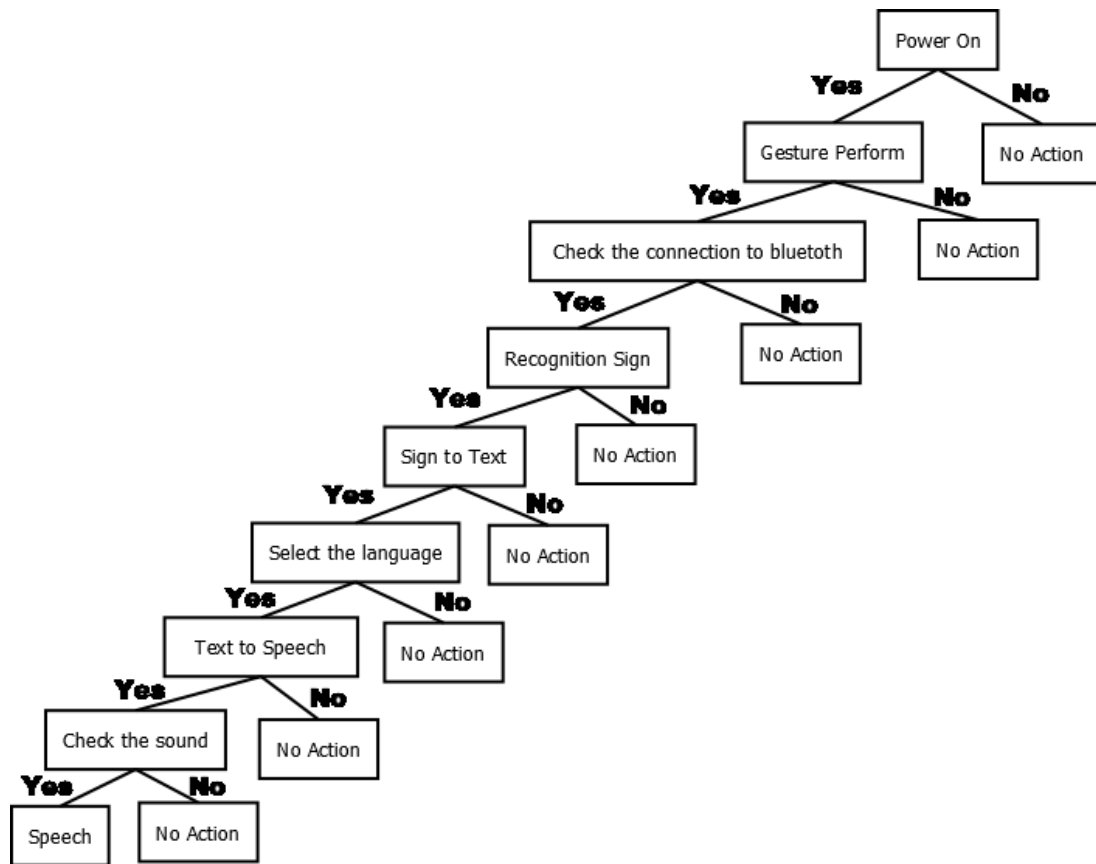


Fig. 3.6 Decision Tree

3.11 Decision Table

Decision tables are a concise visual representation for specifying which actions to perform depending on given conditions. They are algorithms whose output is a set of actions.

Each decision corresponds to a variable, relation or predicate whose possible values are listed among the condition alternatives. Each action is a procedure or operation to perform, and the entries specify whether (or in what order) the action is to be performed for the set of condition alternatives the entry corresponds to.

In other words, a decision table is an excellent tool to use in both testing and requirements management. Essentially it is a structured exercise to formulate requirements when dealing with complex business rules. Decision tables are used to model complicated logic. They can make it easy to see that all possible combinations of conditions have been considered and when conditions are missed, it is easy to see this.

There are three things in Decision tree:

- Conditions
- Rules
- Actions

Conditions in Decision Table explain the lead role playing actors in the project. It is the main component of the table

Actions work on the basis of conditions which is required for projects. It shows the working behavior of lead role playing terms.

Rules are expressed in Decision table by “Y” or “N” for YES and NO respectively.

The following steps are applied to develop a decision table:

1. List all actions that can be associated with a specific procedure (or module).
2. List all conditions (or decisions made) during execution of the procedure.
3. Associate specific sets of conditions with specific actions, eliminating impossible combinations of conditions; alternatively, develop every possible permutation of conditions.
4. Define rules by indicating what action(s) occurs for a set of conditions.

Table 3.2 Decision Table

Rules					
Condition	Power On	N	Y	Y	Y
	Sensor Connection	N	N	Y	Y
	Gesture	N	N	N	Y
Action	Sign To text		Y	Y	Y
	Language for speech		Y		
	Text to speech		Y	Y	

3.12 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

It provides a structural view of systems. It captures the static structure of Object-Oriented systems, or how they are structured rather than how they behave.

Class diagrams support architectural design. It represents the basics of Object-Oriented systems. They identify what classes there are, how they interrelate and how they interact.

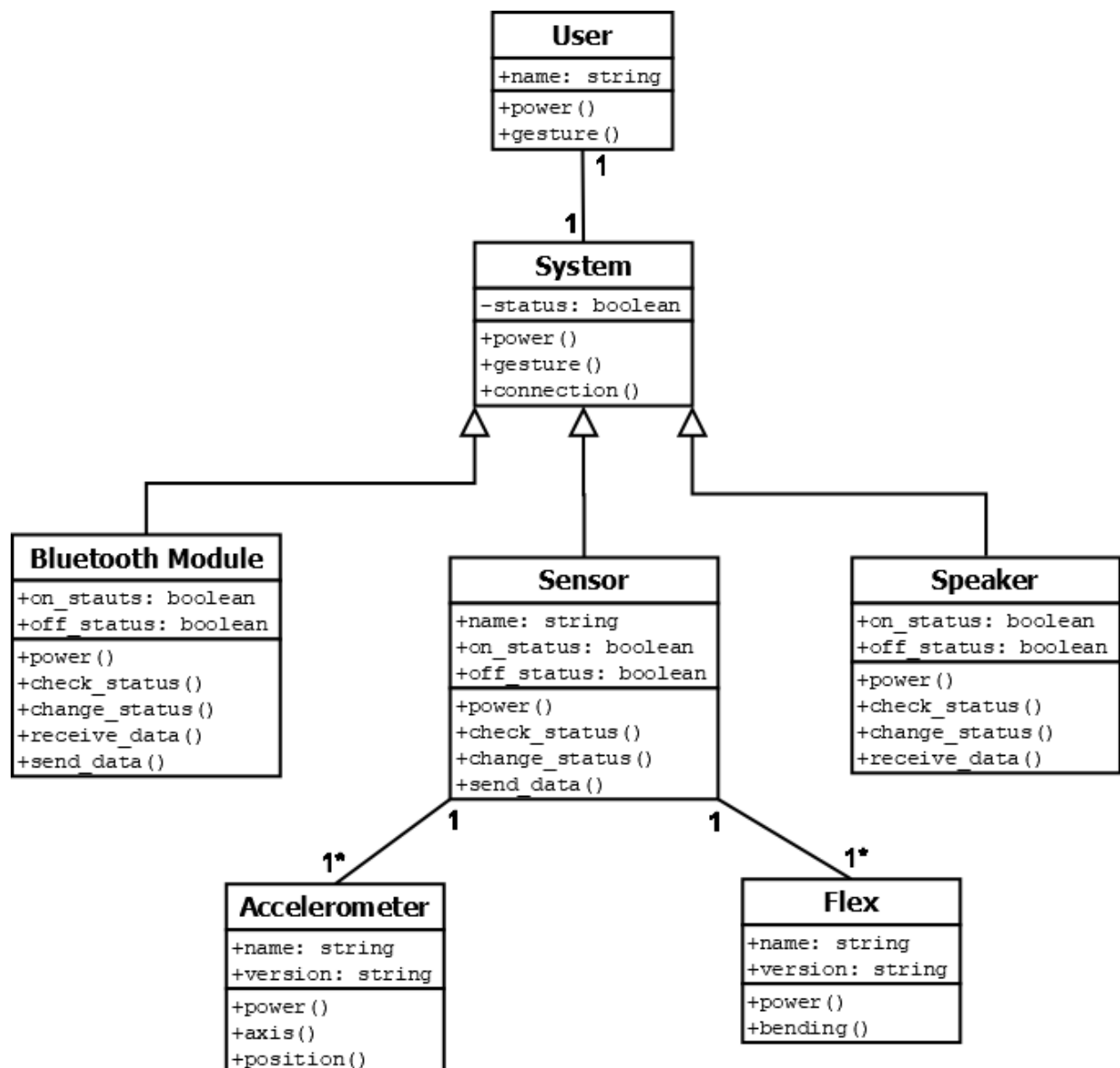


Fig. 3.7 Class Diagram

3.13 Activity Diagram

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagrams are the object-oriented equivalent of flow charts and data-flow diagrams from structured development.

Activity diagrams describe the work flow behavior of a system. The process flows in the system are captured in the activity diagram. It illustrates the dynamic nature of a system by modeling the flow of control from activity to activity.

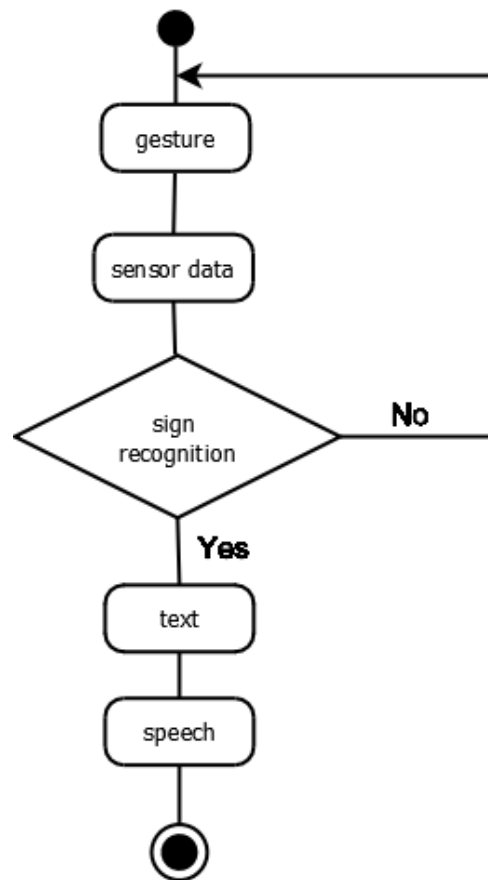


Fig. 3.8 Activity Diagram

3.14 Component Diagram

Component diagrams are different in terms of nature and behavior. Component diagrams are used to model the physical aspects of a system. Now the question is, what are these physical aspects? Physical aspects are the elements such as executables, libraries, files, documents, etc. which reside in a node.

Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

Purpose of Component Diagrams

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system, but it describes the components used to make those functionalities.

The purpose of the component diagram can be summarized as:

- Visualize the components of a system.
- Construct executables by using forward and reverse engineering.
- Describe the organization and relationships of the components.

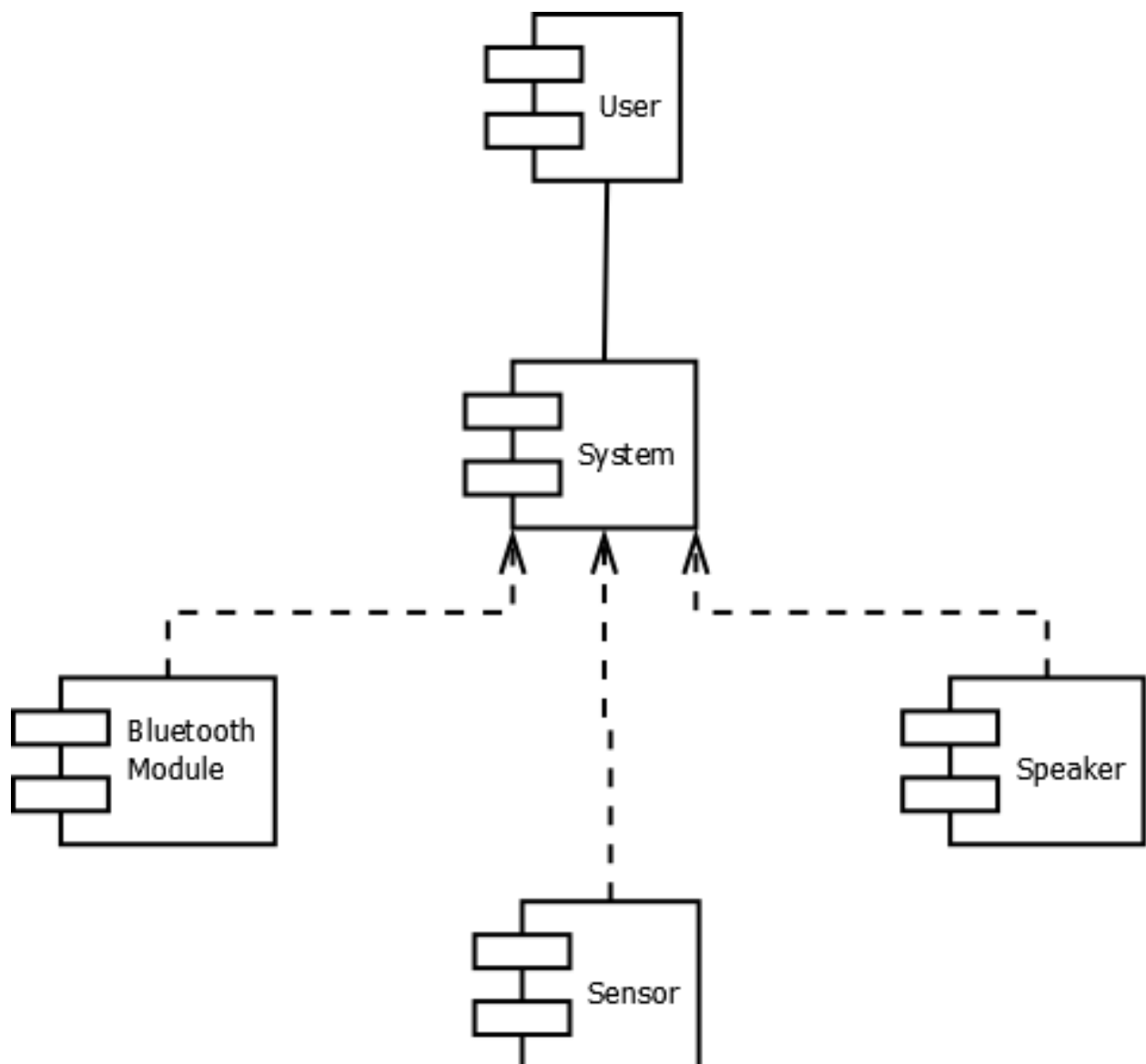


Fig. 3.9 Component Diagram

CHAPTER 4

DATA COLLECTION MODULE

4.1 Introduction: Data collection module

For this study, a custom-made wearable device was taken to hold the hardware. The wearable device holder is printed using flexible filaments with good elasticity. These filaments enable functional hinges, joints, and shaped parts, allowing the device to fit different hand sizes. Five finger holders were also designed using a flexible filament placed on the first joint of each finger to hold the flex sensors. Similarly, these flexible holders can also accommodate different finger sizes of different users.

Different components will be used in order to collect the data of hand gesture:

4.2 Approach

Following are the basic devices responsible for collecting data.

4.2.1 Flex Sensor

The flex sensor consists of an internal resistor. The flex sensors are bidirectional in nature and hence it can detect change in resistance in both positive and negative directions. The internal circuitry of the flex sensor is a voltage divider circuit which gives different values of voltage based on the amount of bending of the fingers, due to the change in resistance inside the flex sensor.

The flex sensor like most of the sensors measures the physical quantity but do so in the unique way. The flex sensor measures the amount of stress that is applied on the sensor while it is being bent that is when the flex sensor is made to bent the resistive film present inside the flex changes its resistance due to the mechanical stress that is applied to it while being bent. The more the sensor is bent the more will be change in resistance. So we can say that flex sensor measures the amount of force or bending angle by exploiting the change in its resistance.



Fig. 4.1 Flex Sensor

Types of Flex Sensor

These sensors are classified into two types based on its size namely 2.2-inch flex sensor & 4.5-inch flex sensor. The size, as well as the resistance of these sensors, is dissimilar except the working principle.

Therefore the suitable size can be preferred based on the necessity. Here this article discusses an overview of 2.2-inch flex-sensor. This type of sensor is used in various applications like computer interface, rehabilitation, servo motor control, security system, music interface, intensity control, and wherever the consumer needs to modify the resistance throughout bending.

In fact, the flex sensor consists of both **omnidirectional** and **bidirectional** types. The omnidirectional flex sensor changes its resistance when it bends in one direction only, whereas the bidirectional type changes its resistance when it bends in both the upward and downward directions.

Pin Configuration of flex sensor

The pin configuration of the flex sensor is shown below. It is a two-terminal device, and the terminals are like p1 & p2. This sensor doesn't contain any polarized terminal such as diode otherwise capacitor, which means there is no positive & negative terminal. The required voltage of this sensor to activate the sensor ranges from 3.3V -5V DC which can be gained from any type of interfacing.

Pin P1: This pin is generally connected to the +ve terminal of the power source.

Pin P2: This pin is generally connected to GND pin of the power source.

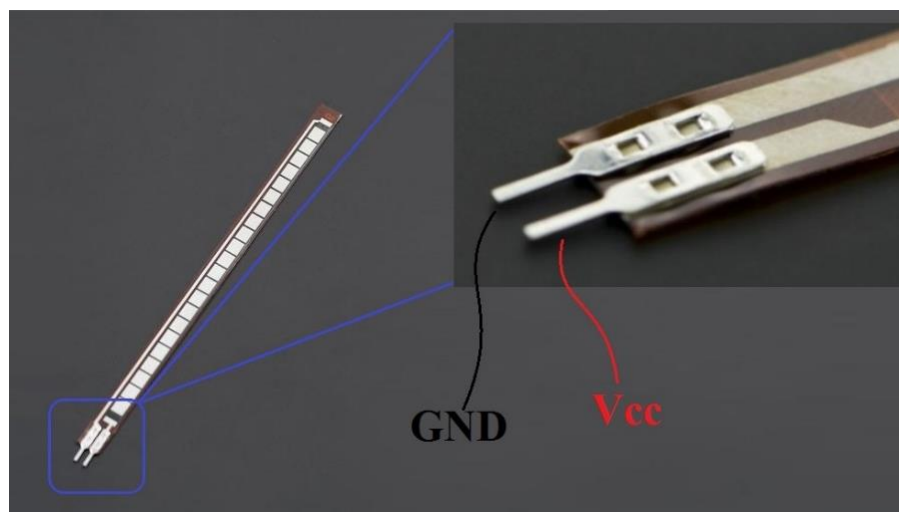


Fig. 4.2 Flex Sensor pins

Working of the Flex Sensor

Let us now see how the Flex sensor can be used in the circuit. As we have seen in the previous discussion that the Flex sensor measures the bending or the stress applied in that way by altering its resistance correspondingly. So we can say that the Flex sensor is basically the variable resistor whose resistance depends upon the amount of bend. Also notice that the Flex sensor is the analog sensor so in order to measure the change in resistance corresponding to the amount that the flex sensor is bent, we need a simple voltage divider circuit. The circuit for reading the change in voltage as a result of the change in resistance is shown in the figure below:

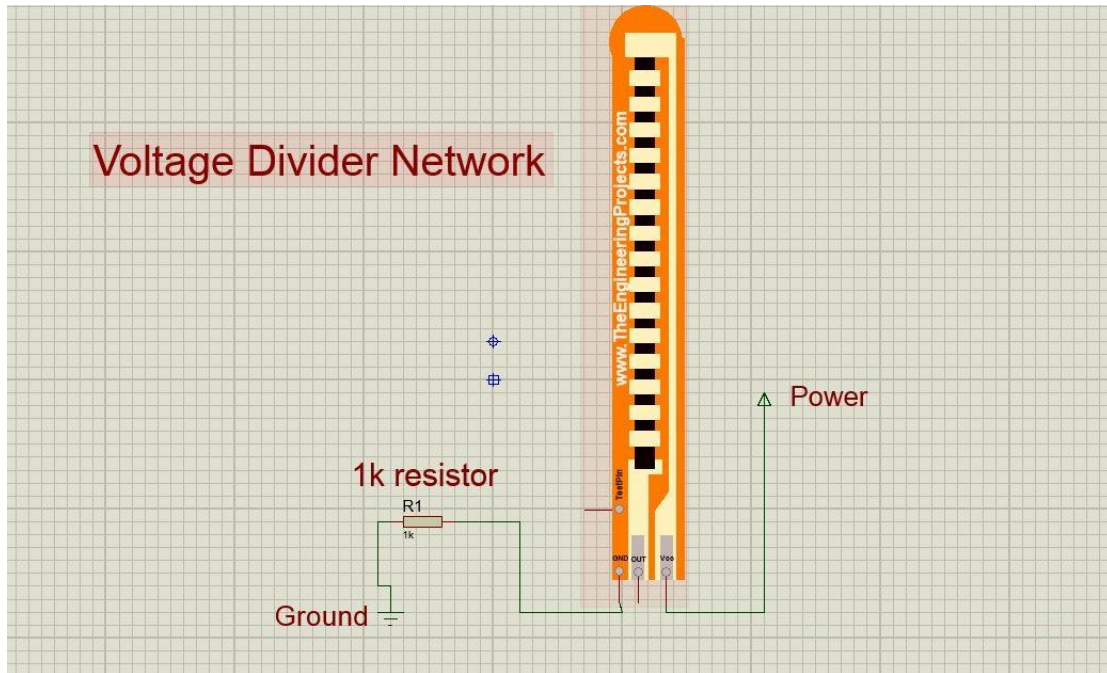


Fig. 4.3 Working of Flex Sensor

Now let us understand the working of the above circuit. As can be seen in the figure that the Flex sensor is connected to the 1 kilo ohm resistor in voltage divider configuration. So if the resistance of the Flex sensor varies with the amount of the bending the voltage drop as governed by the voltage divider equation varies across the 1 kilo resistor and the flex sensor itself. The variation of the voltage level can then easily be read on the analog pin of the Arduino microcontroller development board. So this is the way the flex sensor works.

Specifications & Features of flex sensor

The specifications and features of this sensor include the following.

- Operating voltage of this sensor ranges from 0V to 5V
- It can function on low-voltages.

- Power rating is 1 Watt for peak & 0.5 Watt for continuous.
- Operating temperature ranges from -45°C to +80°C
- Flat resistance is 25K Ω
- The tolerance of resistance will be $\pm 30\%$
- The range of bend resistance will range from 45K -125K Ohms

Application of flex sensor

The applications of the flex-sensor include the following.

- Medical Instruments
- Peripherals of Computer
- Robotics
- Physical Therapy
- Virtual Motion (Gaming)
- Musical Instruments

4.2.2 Three-axis Accelerometer

The three-axis accelerometer is used to measure the orientation of the hand and fingers. The internal circuit of a three-axis accelerometer consists of one fixed capacitor and one variable capacitor. Due to the orientation or tilt of the hand a change in capacitance results in proportional voltage change.

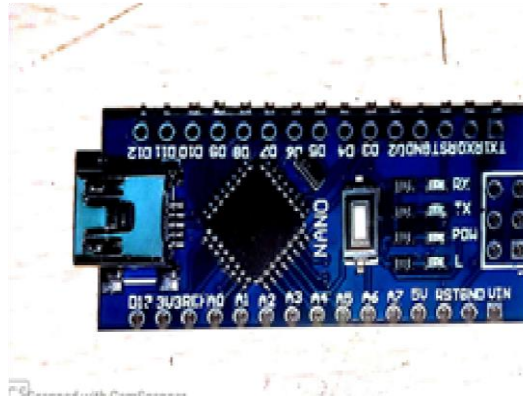


Fig. 4.4 Three-Axis Accelerometer

Accelerometers are devices that measure acceleration, which is the rate of change of the velocity of an object. They measure in meters per second squared (m/s^2) or in G-forces (g). A single G-force for us here on planet Earth is equivalent to 9.8 m/s^2 , but this does vary slightly with elevation (and will be a different value on different planets due to variations in gravitational pull). Accelerometers are useful for sensing vibrations in systems or for orientation applications.

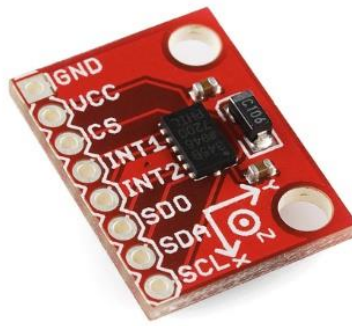


Fig. 4.5 ADXL345 Breakout Board

How an Accelerometer Works

Accelerometers are electromechanical devices that sense either static or dynamic forces of acceleration. Static forces include gravity, while dynamic forces can include vibrations and movement.

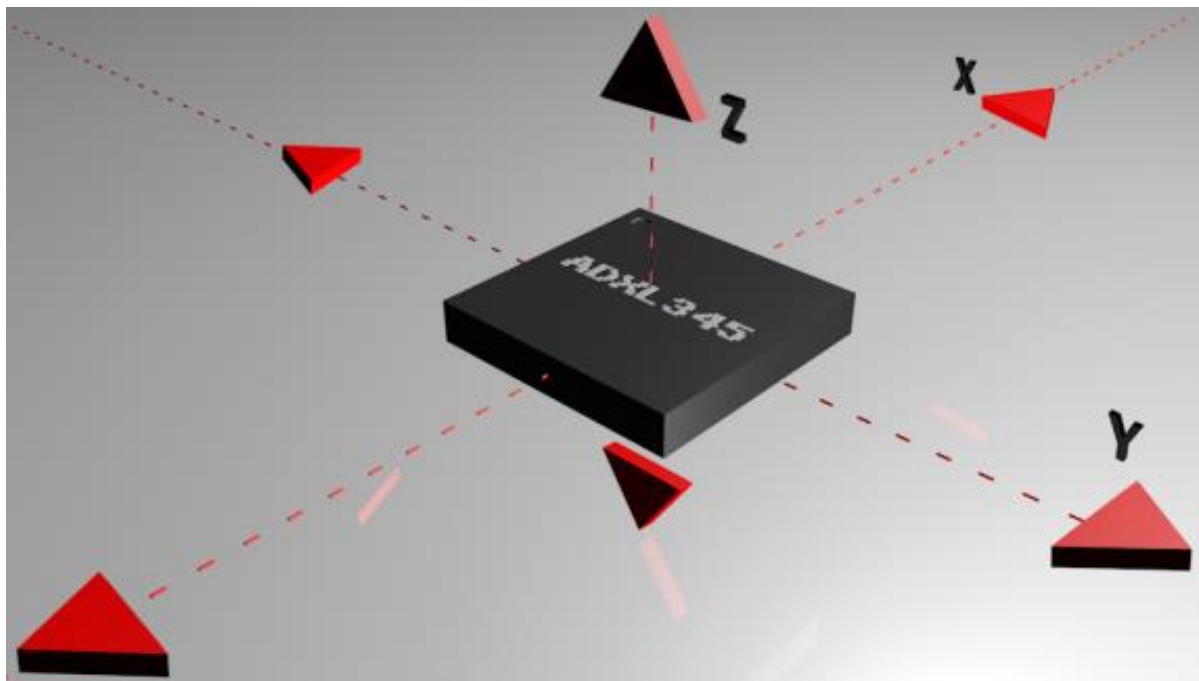


Fig. 4.6 Axes of measurement for a triple axis accelerometer

Accelerometers can measure acceleration on one, two, or three axes. 3-axis units are becoming more common as the cost of development for them decreases.

Generally, accelerometers contain capacitive plates internally. Some of these are fixed, while others are attached to minuscule springs that move internally as acceleration forces act upon the sensor. As these plates move in relation to each other, the capacitance between them changes. From these changes in capacitance, the acceleration can be determined.

Other accelerometers can be centered around piezoelectric materials. These tiny crystal structures output electrical charge when placed under mechanical stress (e.g. acceleration).

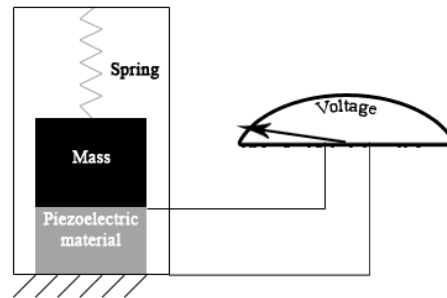


Fig.4.7 Working of Accelerometers

How to Connect to an Accelerometer

For most accelerometers, the basic connections required for operation are power and the communication lines. As always, read the datasheet to ensure proper connections are made.

Communication Interface

Accelerometers will communicate over an analog, digital, or pulse-width modulated connection interface.

Analog - Accelerometers with an analog interface show accelerations through varying voltage levels. These values generally fluctuate between ground and the supply voltage level. An ADC on a microcontroller can then be used to read this value. These are generally less expensive than digital accelerometers.

Digital - Accelerometers with a digital interface can either communicate over SPI or I²C communication protocols. These tend to have more functionality and be less susceptible to noise than analog accelerometers.

Pulse-Width Modulation (PWM) - Accelerometers that output data over pulse-width modulation (PWM) output square waves with a known period, but a duty cycle that varies with changes in acceleration.

Power

Accelerometers are generally low-power devices. The required current typically falls in the micro (μ) or milli-amp range, with a supply voltage of 5V or less. The current consumption can vary depending on the settings (e.g., power saving mode versus standard operating

mode). These different modes can make accelerometers well suited for battery powered applications.

Make sure that proper logic levels are matched, especially with the digital interfaces.

Selection of Accelerometer

When choosing which accelerometer to use, several features are important to consider including power requirements and communication interfaces as discussed previously. Additional features for consideration are below.

Range

Most accelerometers will have a selectable range of forces they can measure. These ranges can vary from $\pm 1g$ up to $\pm 250g$. Typically, the smaller the range, the more sensitive the readings will be from the accelerometer. For example, to measure small vibrations on a tabletop, using a small-range accelerometer will provide more detailed data than using a 250g range (which is more suited for rockets).

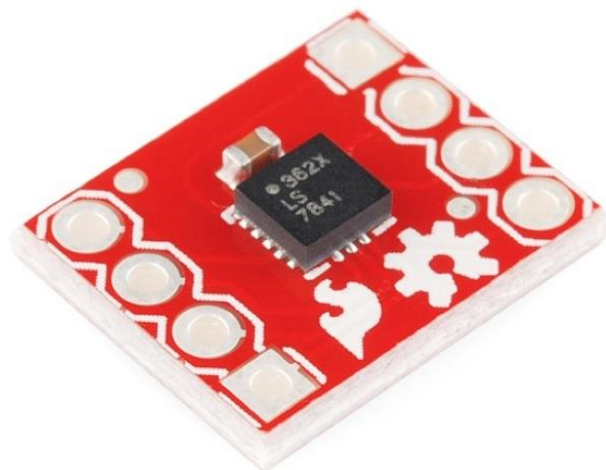


Fig. 4.8 The ADXL362 Triple Axis Accelerometer can measure $\pm 2g$, $\pm 4g$, and $\pm 8g$.

Additional Features

Some accelerometers include features such as tap detection (useful for low-power applications), free-fall detection (used for Active Hard Drive Protection), temperature compensation (to increase accuracy in dead reckoning situations) and 0-g range sensing, which are other features to take into consideration when purchasing an accelerometer. The need for these types of features on the accelerometer will be determined by the application in which the accelerometer is incorporated.

There are also IMUs (Inertial Measurement Units) available, which can include accelerometers, gyroscopes and even, occasionally, magnetometers into a single IC package or board. Some examples of this include the MPU6050 and MPU9150. These are commonly

used in motion tracking applications and UAV guidance systems, where location and orientation of an object is important.

4.2.3 Wi-Fi Module

ESP8266 is a 16 pin self- contained networking unit. These are low power devices which work on UART protocols and are programmed using AT commands. Serially the data is sent from the processor to this unit with the help of UART protocol.

When a particular alphabet is shown using sign language, it involves bending of the fingers thereby resulting in the bending of the flex sensors; there occurs a change in resistance of the flex sensor. A range of voltage values are given for each bend of the flex sensor for every finger. Thus, for every alphabet, each flex sensor is given a particular range of voltage values so as to distinguish it from other alphabets.



Fig. 4.9 Wi-Fi Module

A Wi-Fi microchip module, introduced by Espressif Systems, that comes with both TCP/IP and Microcontroller capability. ESP8266 is very user-friendly, features low cost and develops a simple TCP/IP connection by connecting microcontrollers with Wi-Fi. It has ability to hosting or offloading all Wi-Fi function to other processors. The first chip in this series was ESP-01 that gained a sheer attention in the market but created language barrier as it came with Chinese documentation. Later many features are added to this device that mainly comes with English documentation. It is easy to use and even an average person can make their feet wet with the learning of this device. In this tutorial, we discuss ESP8266 Wi-Fi module, its features, specifications, applications and everything you need to know to make it run in a real-time.

Introduction to ESP8266

ESP8266 is a cost-effective Wi-Fi module that supports both TCP/IP and microcontrollers. It runs at 3V with maximum voltage range around 3.6V. More often than not, it also comes under name ESP8266 Wireless Transceiver.

This module stays ahead of its predecessor in terms of processing speed and storage capability. It can be interfaced with the sensors and other devices and requires very little modification and development to make it compatible with other devices.

Components and GPIO pins interfaced on the little chip are very compact that makes it suitable for hard to reach places.

It covers little space and everything is laid out on the PCB board quite precisely that no external circuitry is required to put this device in the running condition.

No external RF circuitry is required as this module comes with self-calibrated RF capability that makes it suitable to work under all operating conditions.

It is a very useful device for wireless networking, however, there are some limitations i.e. external logic level converter is needed as it doesn't support 5-3V logic shifting.

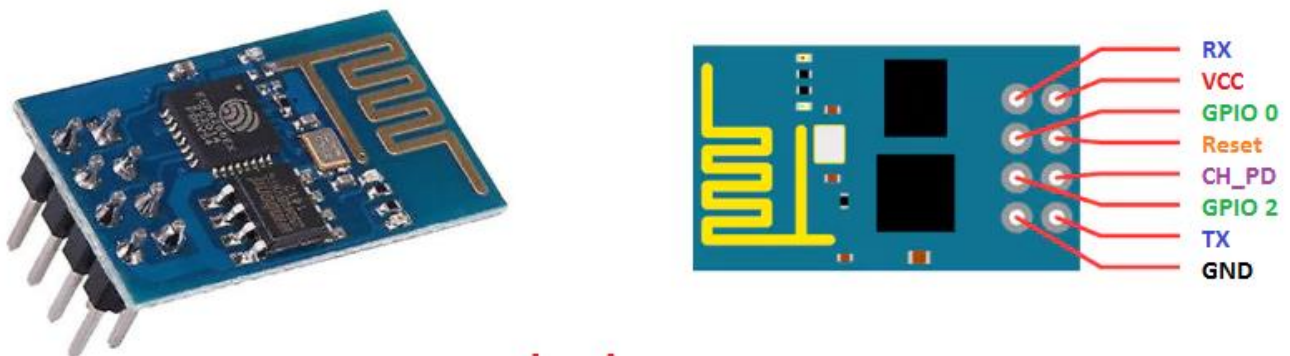


Fig. 4.10 ESP8266

Technical Specifications

- It is also known as a system-on-chip (SoC) and comes with a 32-bit Tensilica microcontroller, antenna switches, RF balun, power amplifier, standard digital peripheral interfaces, low noise receive amplifier, power management module and filter capability.
- The processor is based on Tensilica Xtensa Diamond Standard 106Micro and runs at 80 MHz.

- It incorporates 64 KiB boot ROM, 80 KiB user data RAM and 32 KiB instruction RAM.
- It supports Wi-Fi 802.11 b/g/n around 2.4 GHz and other features including 16 GPIO, Inter-Integrated Circuit (I²C), Serial Peripheral Interface (SPI), 10-bit ADC, and I²S interfaces with DMA.
- External QSPI flash memory is accessed through SPI and supports up to 16 MiB and 512 KiB to 4 MiB is initially included in the module.
- It is a major development in terms of wireless communication with little circuitry, and contains onboard regulator that helps in providing 3.3V consistent power to the board.
- It supports APSD which makes it an ideal choice for VoIP applications and Bluetooth interfaces.

How to Power Up the Module

We can power up the device with PC port using USB to Serial adaptor. The 2 AA and LIPO batteries are equally handy for powering up the device.

It is advised to not power this device directly with 5V dev board. Doing so can severely affect the quality and overall performance of the device.

ESP8266 Pinout

ESP8266 comes with eight pins named:

- RX
- VCC (+3.3 V; can handle up to 3.6 V)
- GPIO 0 General-purpose I/O No. 0
- RST, Reset
- CH_PD (Chip power-down)
- GPIO 2 General-purpose I/O No. 2
- TX
- GND

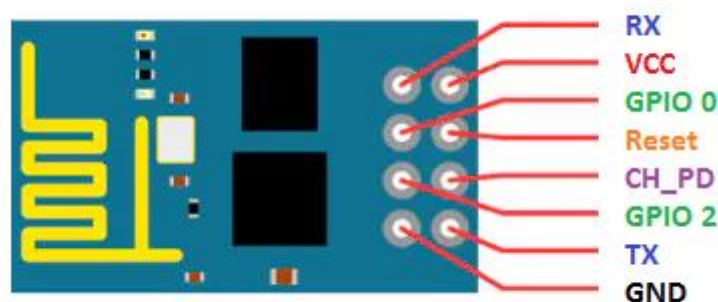


Fig. 4.11 ESP8266 Pinout

Each pin comes with a specific function associated with it where Vcc and GND are voltage source and ground respectively.

RX and TX are used for communication where TX is dedicated for data transmission and RX is used receiving data.

Applications

This module is widely used in many projects with the intention of Wi-Fi capability, however following are the main applications.

- Wireless Web Server
- Geolocation using ESP8266
- Pressure Sensors on Railway Tracks
- Air Pollution Meter
- Temperature logging system
- World's smallest IoT project
- Wi-Fi controlled robot
- Humidity and temperature monitoring
- M2M using ESP8266
- Home appliances
- Home automation
- Smart plugs and lights
- Industrial wireless control
- Baby monitors
- IP cameras
- Sensor networks
- Wearable electronics Security ID tags

4.2.4 Microcontroller (Arduino Nano)

Arduino Nano is the main controller used in this project. It is a microcontroller based board on the Atmega 328k. It has 20 digital input and output pins with 7 pins for PWM.

It is the open-source microcontroller development board based on the ATMEGA328P microcontroller IC. The microcontroller IC on which the Arduino UNO and Arduino NANO is based is usually the same by the way sometimes the difference lies in the package type of the microcontroller IC. Having same microcontroller IC it follows that the crucial specifications of both the Arduino UNO and Arduino NANO are essentially the same. The Arduino NANO is sometimes preferred over the Arduino UNO when there is limitation on the space constraint. Arduino NANO is quite small in size as compared to the Arduino UNO and can easily be mounted on the Breadboard making it useful in Breadboard based prototypes. Arduino NANO has 14 Digital Input / Output pins and 8 analog pins. The Arduino NANO has two additional Analog to Digital converters as compare to the Arduino UNO so

that NANO has two additional Analog pins. Arduino NANO has one UART, one Inter-Integrated Circuit (I2C) computer bus and one Serial Peripheral Interface (SPI) computer bus. Arduino UNO also has one UART, one SPI and one I2C interface on board. Out of the 14 digital input / output pins 5 pins are PWM (Pulse Width Modulation) enabled. The discussion on the PWM phenomenon and the peculiar use of these PWM enabled pins will be discussed later in the posts. Some differences that exist between the Arduino UNO and Arduino NANO will be pointed out later in the post. The Arduino NANO looks like the one in the following figure:



Fig. 4.12 Microcontroller

A. Arduino Nano Pin Configuration

Table 4.1 Pin configuration

Pin Category	Pin Name	Details
Power	Vin, 3.3V, 5V, GND	<p>Vin: Input voltage to Arduino when using an external power source (6-12V).</p> <p>5V: Regulated power supply used to power microcontroller and other components on the board.</p> <p>3.3V: 3.3V supply generated by on-board voltage regulator. Maximum current draw is 50mA.</p> <p>GND: Ground pins.</p>
Reset	Reset	Resets the microcontroller.
Analog Pins	A0 – A7	Used to measure analog voltage in the range of 0-5V

Input/Output Pins	Digital Pins D0 - D13	Can be used as input or output pins. 0V (low) and 5V (high)
Serial	Rx,Tx	Used to receive and transmit TTL serial data.
External Interrupts	2, 3	To trigger an interrupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuilt LED.
IIC	A4 (SDA), A5 (SCA)	Used for TWI communication.
AREF	AREF	To provide reference voltage for input voltage.

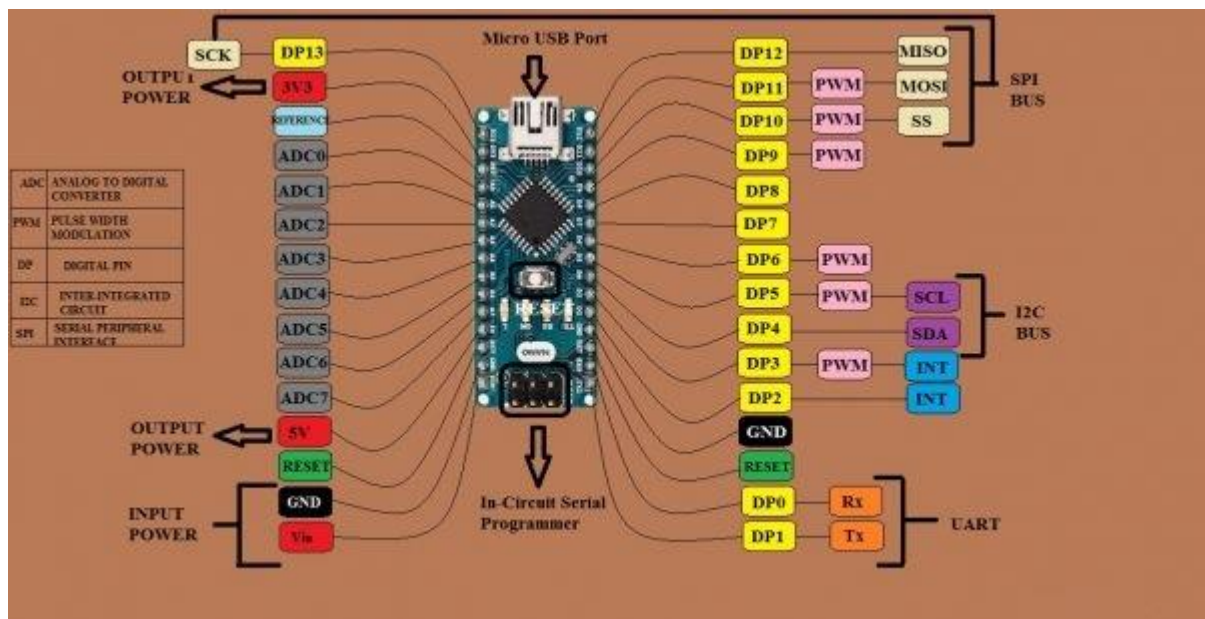


Fig. 4.13 Arduino NANO Pinout

B. Arduino Nano Technical Specifications

Table 4.2 Arduino Nano Technical Specifications

Microcontroller	ATmega328P – 8 bit AVR family microcontroller
Operating Voltage	5V

Recommended Input Voltage for Vin pin	7-12V
Analog Input Pins	6 (A0 – A5)
Digital I/O Pins	14 (Out of which 6 provide PWM output)
DC Current on I/O Pins	40 mA
DC Current on 3.3V Pin	50 mA
Flash Memory	32 KB (2 KB is used for Bootloader)
SRAM	2 KB
EEPROM	1 KB
Frequency (Clock Speed)	16 MHz
Communication	IIC, SPI, USART

C. Understanding Arduino Nano

The Arduino board is designed in such a way that it is very easy for beginners to get started with microcontrollers. This board especially is breadboard friendly is very easy to handle the connections. Let's start with powering the Board.

Powering the Arduino Nano

There are totally three ways by which we can power our Nano.

USB Jack: Connect the mini USB jack to a phone charger or computer through a cable and it will draw power required for the board to function

Vin Pin: The Vin pin can be supplied with a unregulated 6-12V to power the board. The on-board voltage regulator regulates it to +5V

+5V Pin: If you have a regulated +5V supply then you can directly provide this o the +5V pin of the Arduino.

Input/output:

There are totally 14 digital Pins and 8 Analog pins on your Nano board. The digital pins can be used to interface sensors by using them as input pins or drive loads by using them as output pins. A simple function like pinMode() and digitalWrite() can be used to control their operation. The operating voltage is 0V and 5V for digital pins. The analog pins can measure

analog voltage from 0V to 5V using any of the 8 Analog pins using a simple function like `analogRead()`

These pins apart from serving their purpose can also be used for special purposes which are discussed below:

- **Serial Pins 0 (Rx) and 1 (Tx):** Rx and Tx pins are used to receive and transmit TTL serial data. They are connected with the corresponding ATmega328P USB to TTL serial chip.
- **External Interrupt Pins 2 and 3:** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM Pins 3, 5, 6, 9 and 11:** These pins provide an 8-bit PWM output by using `analogWrite()` function.
- **SPI Pins 10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK):** These pins are used for SPI communication.
- **In-built LED Pin 13:** This pin is connected with an built-in LED, when pin 13 is HIGH – LED is on and when pin 13 is LOW, its off.
- **I2C A4 (SDA) and A5 (SCA):** Used for IIC communication using Wire library.
- **AREF:** Used to provide reference voltage for analog inputs with `analogReference()` function.
- **Reset Pin:** Making this pin LOW, resets the microcontroller.

4.3 Introduction to Arduino IDE

Arduino IDE is an open source software that is mainly used for writing and compiling the code into the Arduino Module. It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process.

It is easily available for operating systems like MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role for debugging, editing and compiling the code in the environment.

A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro and many more.

Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code.

The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board.

The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module.

This environment supports both C and C++ languages.

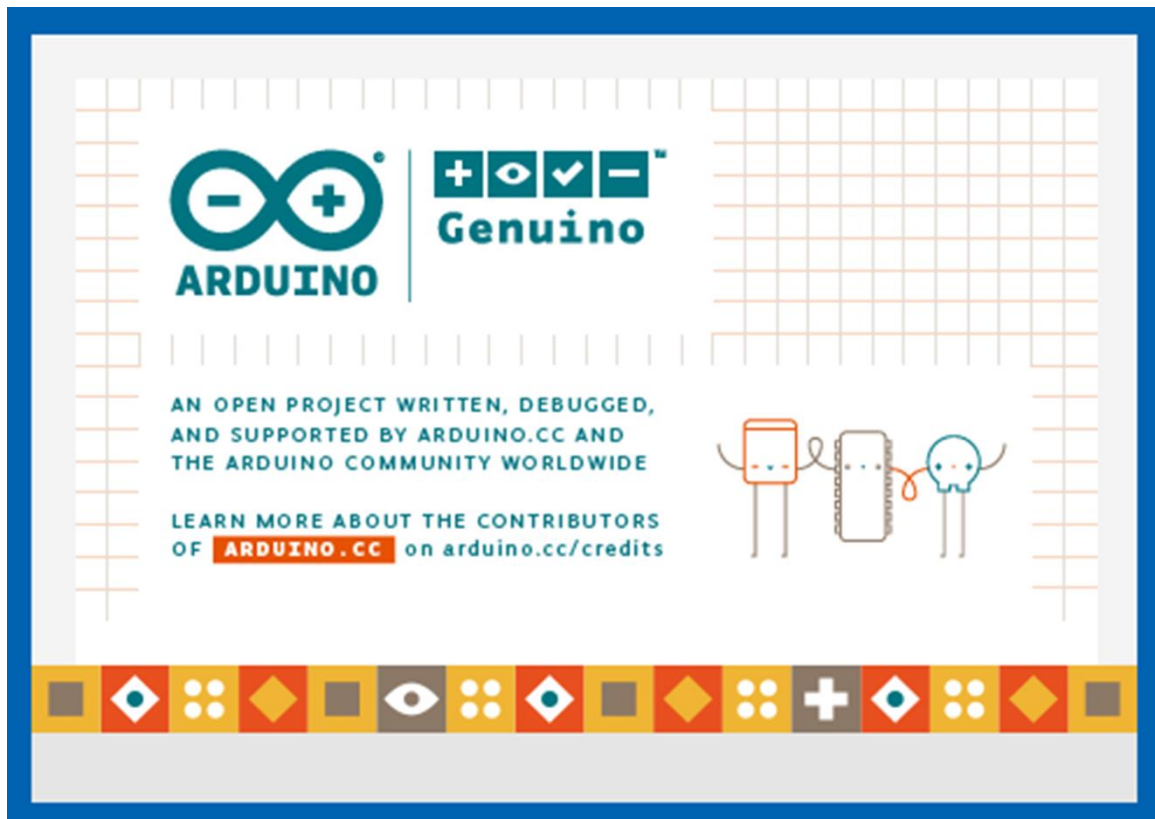


Fig. 4.14 Arduino IDE

The IDE environment is mainly distributed into three sections

1. Menu Bar
2. Text Editor
3. Output Pane

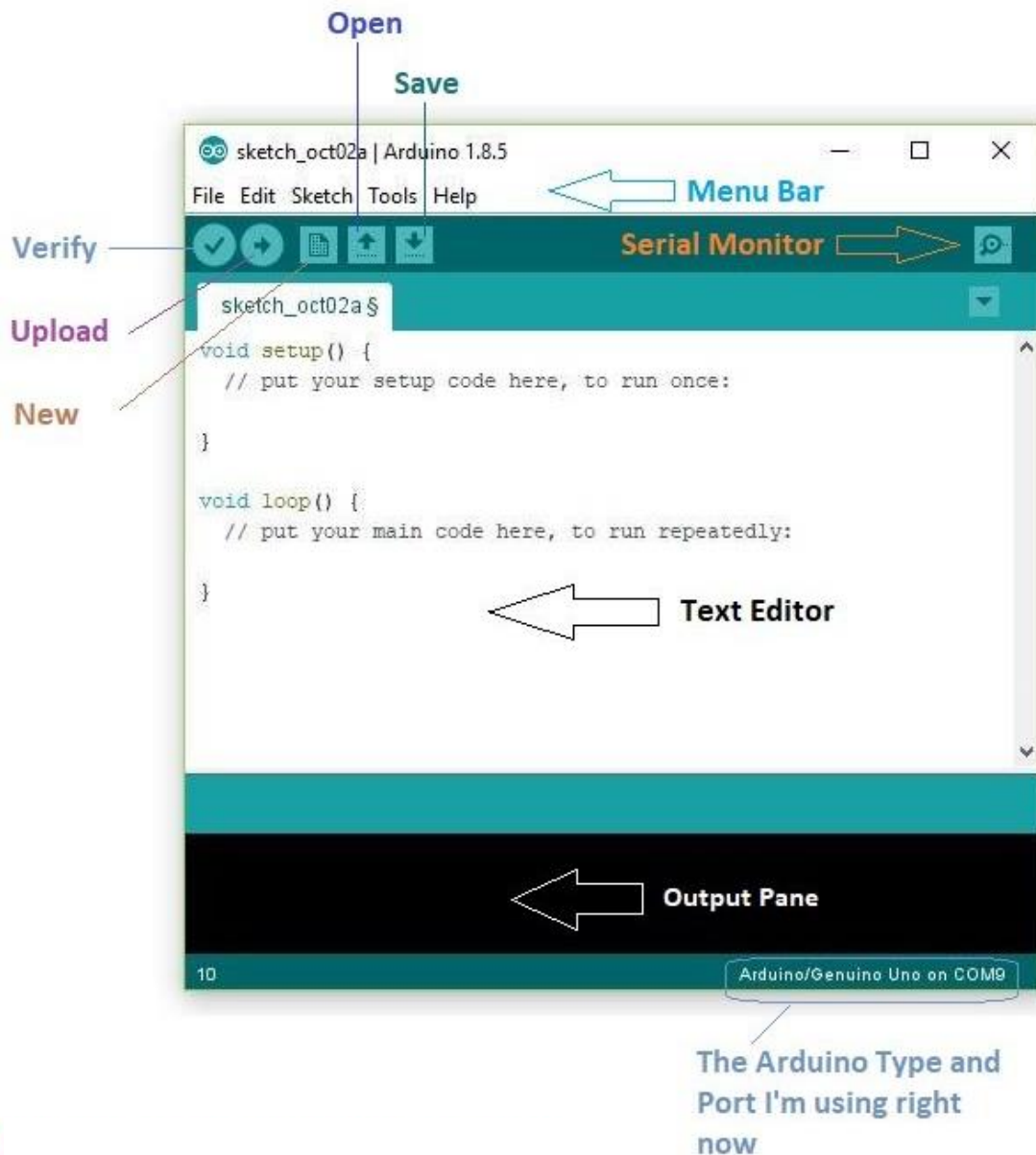


Fig. 4.15 Arduino IDE Home Page layout

Blink example

Most Arduino boards contain a light-emitting diode (LED) and a current limiting resistor connected between pin 13 and ground, which is a convenient feature for many tests and program functions. A typical program used by beginners, akin to Hello, World!, is "blink", which repeatedly blinks the on-board LED integrated into the Arduino board. This program uses the functions `pinMode()`, `digitalWrite()`, and `delay()`, which are provided by the internal libraries included in the IDE environment. This program is usually loaded into a new Arduino board by the manufacturer.



Fig. 4.16 Example of Arduino IDE: Blink

4.4 Data collection Code

flextest4.ino

```
int flexpin1 = A0; int
flexpin2 = A1; int
flexpin3 = A6; int
flexpin4 = A2; int
flexpin5 = A5;
```

```
void setup(){
  Serial.begin(9600);
  Serial.println("CLEARDATA");
  Serial.println("LABEL,Date,Time,Flex1,Flex2,Flex3,Flex4,Flex5");
  Serial.println("RESETTIMER");
}
```

```
void loop(){
  int flexVal1 = analogRead(flexpin1);
  int flexVal2 = analogRead(flexpin2);
  int flexVal3 = analogRead(flexpin3);
  int flexVal4 = analogRead(flexpin4);
```



```
int flexVal5 = analogRead(flexpin5);
```

```
/*Serial.print("Flex1:");  
Serial.println(flexVal1);  
Serial.print("Flex2:");  
Serial.println(flexVal2);  
Serial.print("Flex3:");  
Serial.println(flexVal3);  
Serial.print("Flex4:");  
Serial.println(flexVal4);  
Serial.print("Flex5:");  
Serial.println(flexVal5);  
delay(1000);*/
```

```
Serial.print("DATA,DATE,TIME,");  
Serial.print(flexVal1); Serial.print(",");  
Serial.print(flexVal2); Serial.print(",");
```

```
Serial.print(flexVal3);  
Serial.print(",");
```

```
Serial.print(flexVal4);  
Serial.print(",");  
Serial.println(flexVal5); delay(1000);  
}
```

4.5 Output

Following is the set of data being taken from the flex sensors:

Table 4.3 Data Set

Date	Time	Flex1	Flex2	Flex3	Flex4	Flex5
31-03-20	7:47:23 PM	8	80	35	100	31
31-03-20	7:47:23 PM	8	80	34	100	31
31-03-20	7:47:24 PM	8	80	34	100	31
31-03-20	7:47:25 PM	8	80	34	100	31
31-03-20	7:47:26 PM	7	80	34	100	31
31-03-20	7:47:27 PM	9	81	34	100	32

31-03-20	7:47:28 PM	5	79	34	100	31
31-03-20	7:47:29 PM	7	79	34	100	32
31-03-20	7:47:30 PM	7	79	32	100	32
31-03-20	7:47:31 PM	7	79	34	100	33
31-03-20	7:47:32 PM	0	73	33	100	32
31-03-20	7:47:33 PM	0	74	34	100	32
31-03-20	7:47:34 PM	5	78	33	99	30
31-03-20	7:47:35 PM	0	74	33	99	31
31-03-20	7:47:36 PM	7	79	32	98	31
31-03-20	7:47:37 PM	0	74	32	99	32
31-03-20	7:47:38 PM	6	78	33	99	33
31-03-20	7:47:39 PM	6	79	33	99	32
31-03-20	7:47:40 PM	7	79	33	99	32
31-03-20	7:47:41 PM	7	79	32	98	32
31-03-20	7:47:42 PM	0	74	31	98	31
31-03-20	7:47:43 PM	0	73	32	98	33
31-03-20	7:47:44 PM	0	73	32	98	32
31-03-20	7:47:45 PM	0	74	31	98	33
31-03-20	7:47:46 PM	0	74	32	98	34
31-03-20	7:47:47 PM	4	77	32	98	34
31-03-20	7:47:48 PM	0	73	32	98	33
31-03-20	7:47:49 PM	0	74	32	98	34
31-03-20	7:47:50 PM	5	78	31	98	34
31-03-20	7:47:51 PM	0	73	31	98	34
31-03-20	7:47:52 PM	0	74	31	98	32
31-03-20	7:47:53 PM	1	76	32	98	34
31-03-20	7:47:54 PM	5	78	32	98	33
31-03-20	7:47:55 PM	4	77	32	98	33
31-03-20	7:47:56 PM	4	77	32	98	33
31-03-20	7:47:57 PM	7	79	32	98	33
31-03-20	7:47:58 PM	6	78	30	97	34
31-03-20	7:47:59 PM	7	79	30	97	34
31-03-20	7:48:00 PM	7	79	30	97	34
31-03-20	7:48:01 PM	7	79	30	97	34
31-03-20	7:48:02 PM	7	79	30	97	34
31-03-20	7:48:03 PM	7	79	31	97	34
31-03-20	7:48:04 PM	7	79	31	97	34
31-03-20	7:48:05 PM	7	79	31	97	34
31-03-20	7:48:06 PM	7	79	31	97	34
31-03-20	7:48:07 PM	7	79	31	97	33
31-03-20	7:48:08 PM	7	79	32	98	34
31-03-20	7:48:09 PM	7	79	32	98	35
31-03-20	7:48:10 PM	0	73	32	98	35
31-03-20	7:48:11 PM	0	73	33	98	35
31-03-20	7:48:12 PM	0	73	33	98	35
31-03-20	7:48:13 PM	0	73	33	98	35
31-03-20	7:48:14 PM	0	73	33	98	35
31-03-20	7:48:15 PM	0	73	33	99	35
31-03-20	7:48:16 PM	0	73	33	98	35
31-03-20	7:48:17 PM	0	73	33	98	35
31-03-20	7:48:18 PM	7	79	33	98	36

31-03-20	7:48:19 PM	7	79	33	98	36
31-03-20	7:48:20 PM	0	73	33	98	35
31-03-20	7:48:21 PM	0	74	33	98	36
31-03-20	7:48:22 PM	2	76	33	98	36
31-03-20	7:48:23 PM	4	76	33	99	36
31-03-20	7:48:24 PM	5	77	33	99	36
31-03-20	7:48:25 PM	6	78	33	99	36
31-03-20	7:48:26 PM	6	78	33	98	36
31-03-20	7:48:27 PM	0	74	33	98	36
31-03-20	7:48:28 PM	0	73	33	98	36
31-03-20	7:48:29 PM	5	78	33	98	36
31-03-20	7:48:30 PM	7	78	33	98	36
31-03-20	7:48:31 PM	7	78	34	99	37
31-03-20	7:48:32 PM	0	73	33	99	37
31-03-20	7:48:33 PM	0	73	33	99	37
31-03-20	7:48:34 PM	0	73	33	98	37
31-03-20	7:48:35 PM	3	76	30	97	34
31-03-20	7:48:36 PM	5	77	32	98	35
31-03-20	7:48:37 PM	4	77	29	96	34
31-03-20	7:48:38 PM	0	11	32	98	36
31-03-20	7:48:39 PM	0	11	32	99	36
31-03-20	7:48:40 PM	1	7	32	99	36
31-03-20	7:48:41 PM	1	12	31	98	36
31-03-20	7:48:42 PM	0	45	32	99	37
31-03-20	7:48:43 PM	0	71	34	100	37
31-03-20	7:48:44 PM	0	72	35	100	38
31-03-20	7:48:45 PM	0	72	35	99	38
31-03-20	7:48:46 PM	0	73	33	99	37
31-03-20	7:48:47 PM	0	73	34	99	38
31-03-20	7:48:48 PM	0	73	34	99	38
31-03-20	7:48:49 PM	0	72	33	98	34
31-03-20	7:48:50 PM	0	73	33	98	32
31-03-20	7:48:51 PM	0	73	33	98	33
31-03-20	7:48:52 PM	0	8	32	99	36
31-03-20	7:48:53 PM	0	12	33	100	37
31-03-20	7:48:54 PM	0	70	35	100	36
31-03-20	7:48:55 PM	0	72	27	95	34
31-03-20	7:48:56 PM	0	72	23	92	33
31-03-20	7:48:57 PM	0	72	22	92	31
31-03-20	7:48:58 PM	0	72	23	92	32
31-03-20	7:48:59 PM	0	72	22	91	32
31-03-20	7:49:00 PM	7	78	20	90	33
31-03-20	7:49:01 PM	7	78	21	90	32
31-03-20	7:49:02 PM	7	77	21	91	34
31-03-20	7:49:03 PM	6	78	22	91	33
31-03-20	7:49:04 PM	5	77	21	91	33
31-03-20	7:49:05 PM	0	73	24	91	32
31-03-20	7:49:06 PM	0	73	29	95	32
31-03-20	7:49:07 PM	0	72	26	93	32
31-03-20	7:49:08 PM	0	72	26	94	34
31-03-20	7:49:09 PM	0	72	27	94	33

31-03-20	7:49:10 PM	0	72	29	95	33
31-03-20	7:49:11 PM	0	72	31	97	32
31-03-20	7:49:12 PM	0	73	32	98	33
31-03-20	7:49:13 PM	0	72	31	96	33
31-03-20	7:49:14 PM	6	78	26	94	30
31-03-20	7:49:15 PM	7	78	29	95	31
31-03-20	7:49:16 PM	4	76	30	96	32
31-03-20	7:49:17 PM	8	79	30	96	32
31-03-20	7:49:18 PM	8	79	30	96	32
31-03-20	7:49:19 PM	2	75	31	97	32
31-03-20	7:49:20 PM	0	73	32	97	33
31-03-20	7:49:21 PM	0	73	33	98	33
31-03-20	7:49:22 PM	7	78	32	97	33
31-03-20	7:49:23 PM	3	76	33	98	33
31-03-20	7:49:24 PM	0	73	31	97	0
31-03-20	7:49:25 PM	2	75	33	98	0
31-03-20	7:49:26 PM	1	74	34	98	27
31-03-20	7:49:27 PM	2	75	34	99	29
31-03-20	7:49:28 PM	7	78	34	99	32
31-03-20	7:49:29 PM	6	78	34	99	32
31-03-20	7:49:30 PM	4	76	33	98	0
31-03-20	7:49:31 PM	6	78	35	99	20
31-03-20	7:49:32 PM	0	73	35	99	24
31-03-20	7:49:33 PM	0	72	34	99	24
31-03-20	7:49:34 PM	0	73	34	98	25
31-03-20	7:49:35 PM	1	74	34	99	26
31-03-20	7:49:36 PM	0	74	35	99	26
31-03-20	7:49:37 PM	0	72	34	99	27
31-03-20	7:49:38 PM	0	72	34	99	27
31-03-20	7:49:39 PM	0	73	33	99	28
31-03-20	7:49:40 PM	0	72	34	98	27
31-03-20	7:49:41 PM	0	72	34	99	28
31-03-20	7:49:42 PM	0	72	34	99	28
31-03-20	7:49:43 PM	0	72	33	98	28
31-03-20	7:49:44 PM	0	73	32	99	28
31-03-20	7:49:45 PM	0	73	33	99	28
31-03-20	7:49:46 PM	0	73	34	99	28
31-03-20	7:49:47 PM	0	72	33	98	28
31-03-20	7:49:48 PM	5	77	34	99	29
31-03-20	7:49:49 PM	2	75	32	97	28

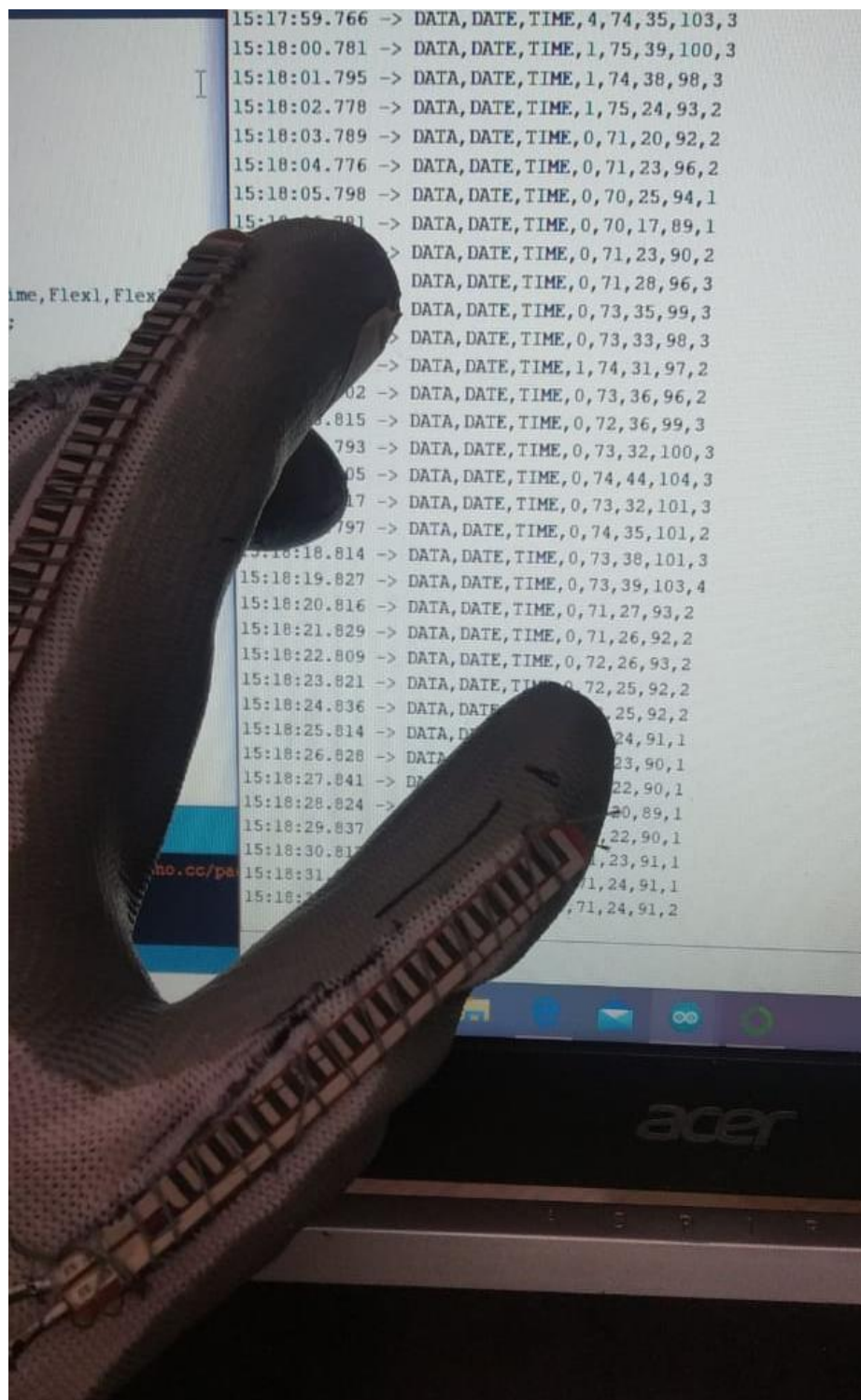


Fig. 4.17 Snap of data being collected

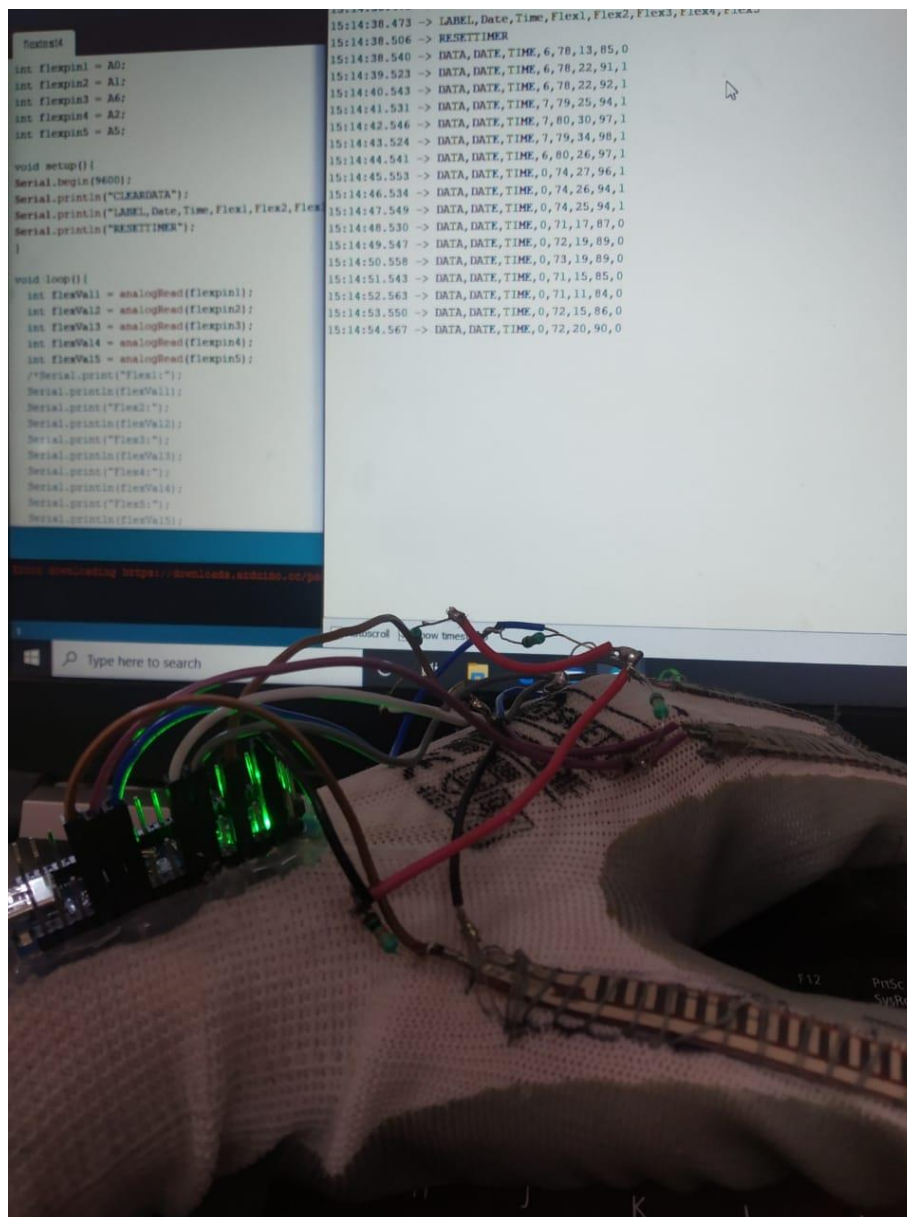


Fig. 4.18 Simulating the data.

CHAPTER 5

DATA PROCESSING MODULE

5.1 Introduction: Data processing module

To simplify and optimize the coding implementation, features are extracted from the sensor data and serve as inputs to the built-in SVM classifier to determine the sign language alphabet letters [6].

In this study, the signs are classified into 28 classes using a support vector machine (SVM) [7]. An SVM is a binary supervised learning classifier, that is, the class labels can only take the values of +1 and -1. The training procedure used a quadratic optimization algorithm to derive structural axes to separate the training dataset into n numbers of a hyperplane.



Fig. 5.1 Training Data

The classified sign gestures from the proposed smart wearable device are transmitted to the sign interpretation system.

5.2 Introduction to MATLAB

MATLAB is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.

The name **MATLAB** stands for **MATrixLABoratory**. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects.

It is a high-performance language for technical computing. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. These factors make MATLAB an excellent tool for teaching and research. MATLAB has many advantages compared to conventional computer languages (e.g., C, FORTRAN) for solving technical problems. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide. It has powerful built-in routines that enable a very wide variety of computations. It also has easy to use graphics commands that make the visualization of results immediately available. Specific applications are collected in packages referred to as toolbox. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering.

Starting MATLAB

After logging into your account, you can enter MATLAB by double-clicking on the MATLAB shortcut icon (MATLAB 7.0.4) on your Windows desktop. When you start MATLAB, a special window called the MATLAB desktop appears. The desktop is a window that contains other windows. The major tools within or accessible from the desktop are:

- The Command Window
- The Command History
- The Workspace
- The Current Directory
- The Help Browser
- The Start button

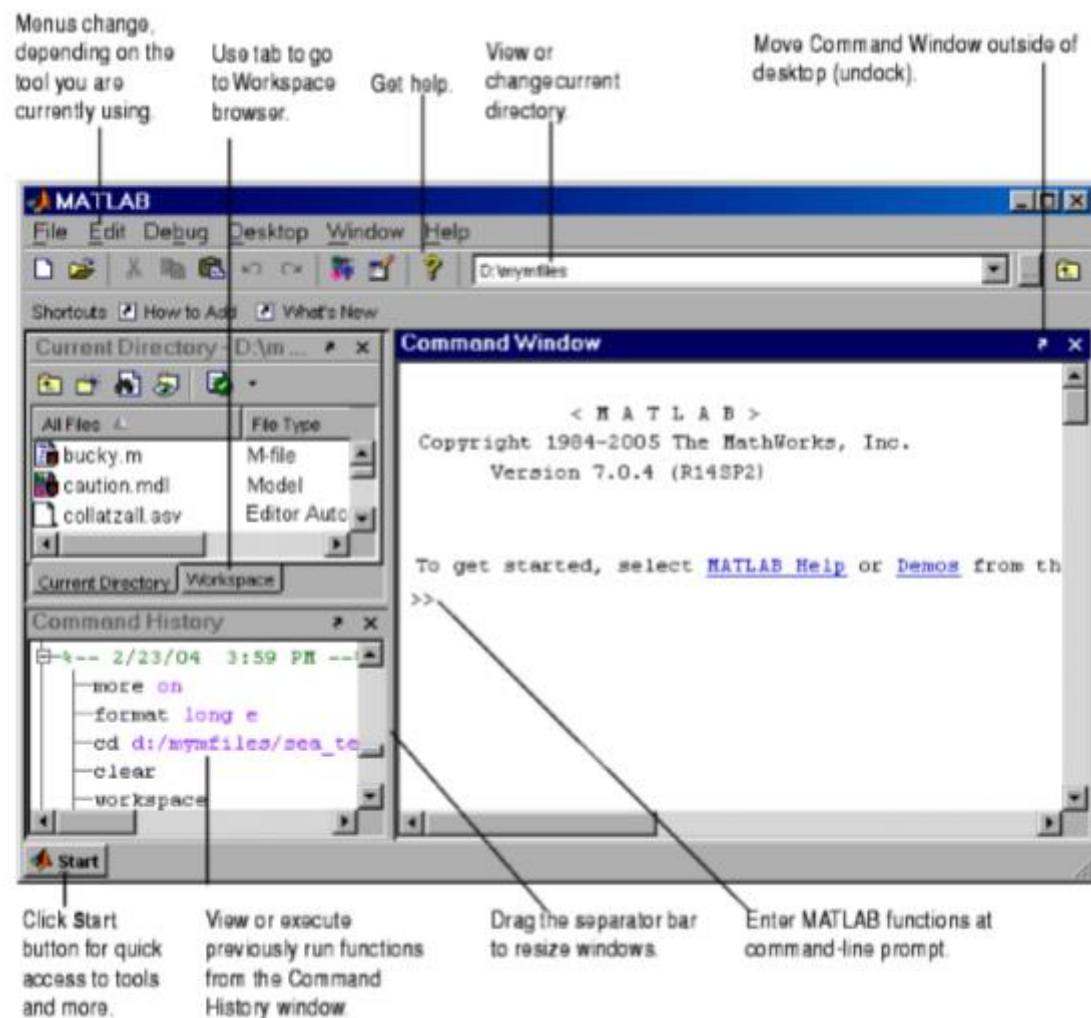


Fig. 5.2 Home page of MATLAB

Writing a MATLAB program

1. Using Command Window:

Only one statement can be typed and executed at a time. It executes the statement when the enter key is pressed. This is mostly used for simple calculations.

2. Using Editor:

Multiple lines of code can be written here and only after pressing the run button (or F5) will the code be executed.

Note: Statements ending with a semicolon will not be displayed in the command window, however, their values will be displayed in the workspace. Any statement followed by % in MATLAB is considered as a comment.

3. Vector Operations:

Operations such as addition, subtraction, multiplication and division can be done using a single command instead of multiple loops

5.3 Introduction to Anaconda

Anaconda® is a package manager, an environment manager, a Python/R data science distribution, and a collection of over 7,500+ open-source packages. Anaconda is free and easy to install, and it offers free community support.

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from PyPI as well as the conda package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the Command Line Interface (CLI).

The big difference between conda and the pip package manager is in how package dependencies are managed, which is a significant challenge for Python data science and the reason conda exists.

When pip installs a package, it automatically installs any dependent Python packages without checking if these conflict with previously installed packages. It will install a package and any of its dependencies regardless of the state of the existing installation. Because of this, a user with a working installation of, for example, Google Tensor flow, can find that it stops working having used pip to install a different package that requires a different version of the dependent numpy library than the one used by Tensor flow. In some cases, the package may appear to work but produce different results in detail.

In contrast, conda analyses the current environment including everything currently installed, and, together with any version limitations specified (e.g. the user may wish to have Tensor flow version 2.0 or higher), works out how to install a compatible set of dependencies, and shows a warning if this cannot be done.

Open source packages can be individually installed from the Anaconda repository, Anaconda Cloud (anaconda.org), or the user's own private repository or mirror, using the conda install command. Anaconda, Inc. compiles and builds the packages available in the Anaconda repository itself, and provides binaries for Windows 32/64 bit, Linux 64 bit and MacOS 64-bit. Anything available on PyPI may be installed into a conda environment using pip, and conda will keep track of what it has installed itself and what pip has installed.

Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, [PyPI](https://pypi.org) or other repositories.

The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, it is possible to create new environments that include any version of Python packaged with conda.

Packages available in Anaconda

- Over 250 packages are automatically installed with Anaconda.

- Over 7,500 additional open-source packages (including R) can be individually installed from the Anaconda repository with the conda install command.
- Thousands of other packages are available from Anaconda Cloud.
- You can download other packages using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in some cases they can work together. However, the preference should be to install the conda package if it is available.
- You can also make your own custom packages using the conda build command, and you can share them with others by uploading them to Anaconda Cloud, PyPI, or other repositories.

Anaconda Navigator

Anaconda Navigator is a desktop Graphical User Interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder
- Glue
- Orange
- RStudio
- Visual Studio Code

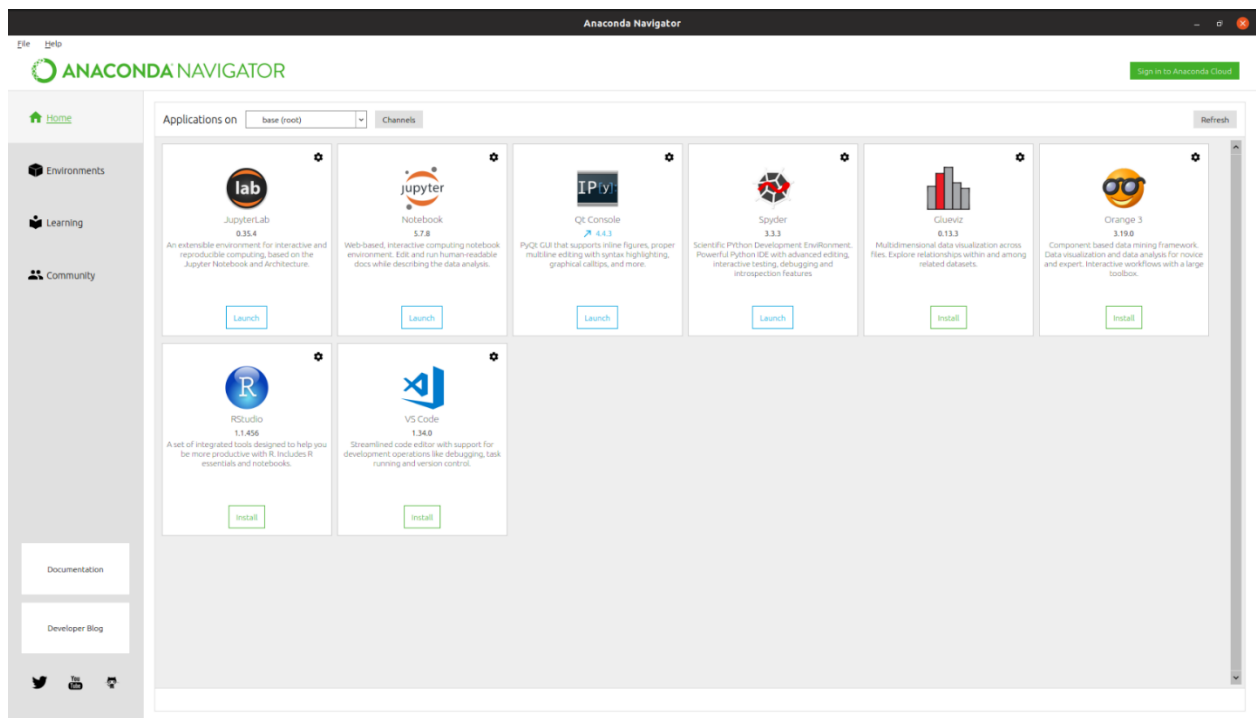


Fig. 5.3 Anaconda Navigator

5.4 Training Code

```
import pandas as pd
import numpy as np
%matplotlib inline

df=pd.read_excel('flextest4.xlsm', index_col=0)
```

```
#preview by default 5 records in the dataset. we can use df.tail()
df.head(500)
```

	Flex2	Flex3	Flex4	Flex5
Flex1				
9	83	35	103	31
8	80	35	100	31
8	80	34	100	31
8	80	34	100	31
8	80	34	100	31

```
#To see the rows and columns and of the data
df.shape

(1311, 4)
```

```
#DATA PROCESSING
#data is divided into attributes and labels
X = df.drop('Flex5', axis=1)
y = df['Flex5']
```

```
#Once the data is divided into attributes and labels,
#the final preprocessing step is to divide data into training and test sets.
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)

#We have divided the data into training and testing sets
```

```
#Now is the time to train our SVM on the training data:::::::::::::Since we are going to perform
a classification task,
#we will use the support vector classifier class,
#which is written as SVC in the Scikit-Learn'ssvm library. This class takes one parameter,
which is the kernel type.
```

```
from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)

#pd.plotting.register_matplotlib_converters()
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
kernel='linear', max_iter=-1, probability=False, random_state=None,
shrinking=True, tol=0.001, verbose=False)
```

```
#Making prediction
y_pred = svclassifier.predict(X_test)
```

```
#Evaluating the Algorithm
#Confusion matrix, precision, recall, and F1 measures are the most commonly used metrics
for classification tasks.
```

```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```

[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0]
[0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1]
[0 0 0 0 0 0 0 0 0 23 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 13 0 2 0 0 0 0 0 0 0 1 2 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 12 0 1 2 0 0 0 0 0 0 1 2 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 8 0 3 0 0 0 0 0 0 0 1 1 1 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 7 0 0 1 0 1 0 0 0 0 2 0 0 0 1 0
0 0]
[0 0 0 0 0 0 0 0 0 0 2 0 0 1 0 0 2 1 2 0 0 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 3 1 3 1 0 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 6 1 0 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 4 0 9 1 0 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 0 0 1 1 0 2 0 0 3 0 12 2 0 0 0 0
0 1]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 21 2 0 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 0 14 7 0 0 1 0
0 0]
[0 0 0 0 0 0 0 0 0 1 2 0 0 0 0 0 0 0 1 0 5 5 0 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 2 0 5 3 0 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 1 1 2 0 0 3 0
0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0
0 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 8 0 0 0 0 0
1 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0]

```

0 0]]

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1
1	0.00	0.00	0.00	0
11	0.00	0.00	0.00	1
13	0.00	0.00	0.00	1
14	0.00	0.00	0.00	1
15	0.00	0.00	0.00	1
17	0.00	0.00	0.00	3
19	0.00	0.00	0.00	10
20	0.29	1.00	0.45	23
21	0.00	0.00	0.00	18
22	0.08	0.06	0.07	18
23	0.00	0.00	0.00	14
24	0.00	0.00	0.00	12
25	0.11	0.12	0.12	8
26	0.00	0.00	0.00	10
27	0.00	0.00	0.00	9
28	0.19	0.25	0.22	16
29	0.00	0.00	0.00	22
30	0.25	0.88	0.39	24
31	0.18	0.28	0.22	25
32	0.00	0.00	0.00	14
33	0.00	0.00	0.00	11
34	0.60	0.33	0.43	9
35	0.00	0.00	0.00	2
36	1.00	0.10	0.18	10
38	0.00	0.00	0.00	0
accuracy		0.23		263
macro avg	0.10	0.12	0.08	263
weighted avg	0.14	0.23	0.14	263

5.5 Printout

/* printOut() * * Simply print every sensor value to the serial monitor display. Returns nothing. */

```
void printOut(){  
  
    Serial.print("You signed gesture ");  
  
    Serial.print(gesture);  
  
    Serial.print(": ");  
  
    Serial.println(gestureName[gesture]);
```

```

    printFCG();

    return;
}

void printFCG(){
    Serial.print("F>>> ");

        Serial.print(fSensor0);

        Serial.print(" ");

            Serial.print(fSensor1);

        Serial.print(" ");

            Serial.print(fSensor2);

        Serial.print(" ");

            Serial.print(fSensor3);

        Serial.print(" ");

            Serial.print(fSensor4);

        Serial.print(" ");

            Serial.print(fSensor5);

        Serial.print(" ");

    Serial.print("C>>> ");

        Serial.print(cSensor0);

        Serial.print(" ");

            Serial.print(cSensor1);

        Serial.print(" ");

            Serial.print(cSensor2);

        Serial.print(" ");

```



```

        Serial.print(cSensor3);

        Serial.print(" ");

        Serial.print(cSensor4);

        Serial.print(" ");

        Serial.println("");

    Serial.print("G>>> ");

    Serial.print(x); //x

    Serial.print(" ");

    Serial.print(y); //y

    Serial.print(" ");

    Serial.print(z); //z

    Serial.println("");

    Serial.println("");

}

```

readSensorValues:

/ readSensorValues() is used by both modes. * When it is called, it does not return anything but it * updates all of the sensor values as their respective global variables. */*

//Returns new cBuffer

```

void readSensorValues(){

    cSensor0 = digitalRead(pinC0);

    cSensor1 = digitalRead(pinC1);

    cSensor2 = digitalRead(pinC2);

    cSensor3 = digitalRead(pinC3);

```

```
cSensor4 = digitalRead(pinC4);
```

```
fSensor0= adc_single_channel_read (adc_single_ch0);
```

```
fSensor1= adc_single_channel_read (adc_single_ch1);
```

```
fSensor2= adc_single_channel_read (adc_single_ch2);
```

```
fSensor3= adc_single_channel_read (adc_single_ch3);
```

```
fSensor4= adc_single_channel_read (adc_single_ch4);
```

```
fSensor5= adc_single_channel_read (adc_single_ch5);
```

WriteTrainingData:

/* writeTrainingData() is used for training mode. It uses readSensorValues(), * then stores these globals into the PreData[][] training data array. */

```
void writeTrainingData(inti){
```

```
    readSensorValues();
```

```
    PreData[i][0] = fSensor0;
```

```
    PreData[i][1] = fSensor1;
```

```
    PreData[i][2] = fSensor2;
```

```
    PreData[i][3] = fSensor3;
```

```
    PreData[i][4] = fSensor4;
```

```
    PreData[i][5] = fSensor5;
```

```
    PreData[i][8] = cSensor0;
```

```
    PreData[i][9] = cSensor1;
```

```
    PreData[i][10]= cSensor2;
```

```
    PreData[i][11] = cSensor3;
```

```
PreData[i][12] = cSensor4;  
  
return;  
  
}
```

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

In this study, we successfully designed and implemented a novel and smart wearable hand device as a sign interpretation system using a built-in SVM classifier. A system is taken to demonstrate the usability of the proposed smart wearable device with an available text-to-speech service. The participating subjects gave a high rating to the proposed smart wearable sign interpretation system in terms of its comfort, flexibility, and portability. The device holders were simple gloves using a flexible nature, so that it can fit different hand and finger sizes, thus eliminating the necessity of custom-made devices.

6.2 Future Scope

Future work on the proposed smart wearable hand device will consider the design of a smaller sized printed circuit board, the inclusion of words and sentences at the sign language level, and instantly audible voice output components.

Our main aim is to reach every person who feels desperate due to their disability at the utmost cost effective solution. This prototype can be further modified to accommodate the sign language gesture alongside with the already existing alphabets. The Android Application can be customized to add further features like incorporation of voice for the sign language gestures and alongside display the history of texts for previously executed gestures.

References

- 1 J. Wang and T. Zhang, "An ARM-based embedded gesture recognition system using a data glove," presented at the 26th Chinese Control and Decision Conf., Changsa, China, May 31 - June2, 2014.
- 2 A. Z. Shukor, M. F. Miskon, M. H. Jamaluddin, F. A. Ibrahim, M. F. Asyraf and M. B. Bahar, "A new data glove approach for Malaysian sign language detection," *Procedia Comput. Sci.*, vol. 76, no. 1, pp. 60-67, Dec. 2015.
- 3 N. Sriram and M. Nithiyanandham, "A hand gesture recognition based communication system for silent speakers," presented at the Int. Conf. Human Comput. Interact., Chennai, India, Aug. 23-24, 2013.
- 4 S. V. Matiwade and M. R. Dixit, "Electronic support system for deaf and dumb to interpret sign language of communication," *Int. J. Innov. Research Sci. Eng. Technol.*, vol. 5, no. 5, pp. 8683-8689, May 2016.
- 5 S. Goyal, I. Sharma and S. Sharma, "Sign language recognition system for deaf and dumb people," *Int. J. Eng. R. Technol.*, vol. 2, no. 4, pp. 382-387, Apr. 2013
- 6 B. G. Lee, Member, IEEE, and S. M. Lee "Smart Wearable Hand Device for Sign Language Interpretation System with Sensors Fusion" DOI 10.1109/JSEN.2017.2779466, *IEEE Sensors Journal*.
- 7 C. L. Lim, C. Rennie, R. J. Barry, H. Bahramali, I. Lazzaro, B. Manor and E. Gordon, "Decomposing skin conductance into tonic and phasic components," *Int. J. Psychophys.*, vol.25, no. 2, pp. 97-109, Feb. 1997.
- 8 S. P. More and A. Sattar, "Hand gesture recognition system using image processing," presented at Int. C onf. Electrical, Electronics, Opt. Techniq., Chennai, India, Mar. 2016.
- 9 K. Murakami and H. Taguchi, "Gesture recognition using recurrent neural network," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, New York, USA, 1991
- 10 P. R. V. Chowdary, M. N. Babu, T. V. Subbareddy, B. M. Reddy and V. Elamaran, "Image processing algorithms for gesture recognition using matlab," presented at Int. Conf. Adv. Comm. Control Comput. Technol., Ramanathapuram, India, Jan. 2015.
- 11 T. Khan and A. H. Pathan, "Hand gesture recognition based on digital image processing using matlab," *Int. J. Sci. Eng. R.*, vol. 6, no. 9, pp. 338-346, Sep. 2015.

- 12 J. Siby, H. Kader and J. Jose, "Hand gesture recognition," *Int. J. Innov. Technol. R.*, vol. 3, no. 2, pp. 1946-1949, March 2015.
- 13 S. P. Dawane and H. G. A. Sayyed, "Hand gesture recognition for deaf and dumb people using gsm module," *Int. J. Sci. R.*, vol. 6, no. 5, pp. 2226-2230, May 2017.
- 14 C. Preetham, G. Ramakrishnan, S. Kumar and A. Tamse, "Hand talk-implementation of a gesture recognizing glove," presented at 2013 Texas Instruments India Educators' Conf., Bangalore, India, Apr. 4-6, 2013.
- 15 K. Patil, G. Pendharkar and G. N. Gaikwad, "American sign language detection," *Int. J. Sci. R. Pub.*, vol. 4, no. 11, pp. 1-6, Nov. 2014.
- 16 J. Kim, N. D. Thang and T. Kim, "3-D hand motion tracking and gesture recognition using a data glove," presented at IEEE Int. Symp. Industrial Elec. 2009, Seoul, South Korea, Jul. 5-8, 2009.
- 17 D. Lu, Y. Yu and H. Liu, "Gesture recognition using data glove: an extreme learning machine method," in *Proc. 2016 IEEE Int. Conf. Robotics and Biomimetics*, Qingdao, China, Dec. 3-7, 2016.
- 18 J. Lm, D. Lee, B. Kim, I. Cho and J. Ryou, "Recognizing hand gestures using wrist shapes," presented at 2010 Digest Technical Papers Int. Conf. Consumer Elec., Las Vegas, USA, Jan. 9-13, 2010.
- 19 R. Xie, X. Sun, X. Xia and J. Cao, "Similarity matching-based extensible hand gesture recognition," *IEEE Sensors J.*, vol. 15, no. 6, pp. 3475-3483, Jun. 2015.
- 20 Y. L. Hsu, C. L. Chu, Y. J. Tsai and J. S. Wang, "An inertial pen with dynamic time warping recognizer for handwriting and gesture recognition," *IEEE Sensors J.*, vol. 15, no. 1, pp. 154-163, Jan. 2015.
- 21 L. Yin, M. Dong, Y. Duan, W. Deng, K. Zhao and J. Guo, "A high-performance training-free approach for hand gesture recognition with accelerometer," *Mult. Tools App.*, vol. 72, no. 1, pp. 843-864, Sep. 2014.
- 22 J. Galka, M. Masior, M. Zaborski, K. Barczewska, "Inertial motion sensing glove for sign language gesture acquisition and recognition," *IEEE Sensors J.*, vol. 16, no. 16, pp. 6310-6316, Aug. 2016.
- 23 X. Cai, T. Guo, X. Wu and H. Sun, "Gesture recognition method based on wireless data glove with sensors," *Sensor Letters*, vol. 13, no. 2, pp. 134-137, Feb. 2015.

- 24 K. Liu, C. Chen, R. Jafari and N. Kehtarnavaz, "Fusion of inertial and depth sensor data for robust hand gesture recognition," *IEEE Sensors J.*, vol. 14, no. 6, pp. 1898-1903, Jun. 2014.
 - 25 K. W. Kim, M. S. Lee, B. R. Soon, M. H. Ryu and J. N. Kim, "Recognition of sign language with an inertial sensor-based data glove," *Technol. Health Care*, vol. 24, no. 1, pp. 223-230, 2016.
 - 26 L. Sousa, J. M. F. Rodrigues, J. Monteiro, P. J. S. Cardoso and R. Lam, "GyGSLA: a portable glove system for learning sign language alphabet," presented at *Int. Conf. Universal Access in Human-Comput. Interact.*, Toronto, ON, Canada, Jul. 17-22, 2016.
 - 27 N. Caporusso, L. Biasi, G. Cinquepalmi, G. F. Trotta and A. Brunetti, "A wearable device supporting multiple touch- and gesture-based languages for the deaf-blind," presented at *Int. Conf. Appl. Human Factors and Ergonomics*, LA, CA, USA, Jul. 17-21 2017.
 - 28 Z. Yu, X. Chen, Q. Li, X. Zhang and P. Zhou, "A hand gesture recognition framework and wearable gesture-based interaction prototype for mobile devices," *IEEE Trans. Human-Mach. Syst.*, vol. 44, no. 2, pp. 293-299, Apr. 2014.
 - 29 J. Wu, Z. Tian, L. Sun, L. Estevez and R. Jafari, "Real-time American sign language recognition using wrist-worn motion and surface emg sensors," presented at *IEEE 12th Int. Conf. Wearable Implantable Body Sensor Net.*, Cambridge, USA, Jun. 9-12, 2015.
 - 30 J. Wu, L. Sun and R. Jafari, "A wearable system for recognizing American sign language in real-time using imu and surface emg sensors," *IEEE J. Biomedic. Health Info.*, vol. 20, no. 5, pp. 1281-1290, Sept. 2016.
- [NAS73] Nassi, I. and B. Shneiderman, "Flowchart Techniques for Structured Programming," *SIGPLAN Notices*, ACM, August 1973.
- [CHA74] Chapin, N., "A New Format for Flowcharts," *Software—Practice and Experience*, vol. 4, no. 4, 1974, pp. 341–357.