UNIVERSITY OF NAIROBI
SCHOOL OF COMPUTING AND INFORMATICS

SIGN LANGUAGE GESTURE RECOGNITION USING MACHINE LEARNING ON THE EDGE

By
Eliud Ngigi NGARUIYA
P52/33754/2019

Supervisor
Dr. Wanjiku Ng'ang'a

RESEARCH PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE AWARD OF MASTER OF SCIENCE IN COMPUTATIONAL INTELLIGENCE AT THE SCHOOL OF COMPUTING AND INFORMATICS, UNIVERSITY OF NAIROBI

August 17, 2021.

DECLARATION

Researcher's Declaration

This project is my original work and has not been presented in any other institution for the purpose of academic award. All sources, references, literature used or excerpted during elaboration of this work are properly cited and listed in reference to the respective sources.

SIGNATURE______________________ DATE 17th August '2021

Eliud Ngigi NGARUIYA

Registration Number: P52/33754/2019

Supervisor's Approval

This project report has been submitted in partial fulfilment for the requirements of the award of the Degree of Master of Science in Computational Intelligence in the University of Nairobi with my approval as the University Supervisor.

SIGNATURE______________________ DATE 17.08.2021

Dr. Wanjiku Ng'ang'a

School of Computing and Informatics

University of Nairobi

ACKNOWLEDGEMENTS

I would like to acknowledge and thank my parents, without their help, moral and financial support, pursuing this study would not have been possible for me. Many thanks to my supervisor, Dr. Wanjiku Ng’ang’a who’s insights and guidance I had through every step from developing a proposal through to finalizing the project.

ABSTRACT

There are approximately over 360 million people globally living with a hearing disability; partially or completely deaf. In order to communicate, these people use sign language. Like any spoken language, sign language has its own grammar rules and can be translated from signed language to spoken language by a sign language interpreter. However, unlike translating spoken languages, translating sign language is a challenging task. Humans have been doing the translation. But with recent advancement in machine learning and artificial intelligence algorithms, these translation tasks are being taken up by machines. Despite the progress, machine translation of sign language is still faced with challenges since it requires data that is largely unavailable of hand and finger movements for words and phrases signing. Furthermore, the approaches adopted such as computer vision are resource intensive.

Embedded systems have evolved from simple transistor circuits to complex microprocessor and microcontroller systems. Though still resource constrained in terms of the processing power, memory and power consumption, these embedded systems can now perform tasks previously not possible on earlier versions. Recently, we have had machine learning algorithms that can run on resource-constrained embedded systems like TensorFlow Lite for Microcontrollers. With these algorithms, we can now perform machine learning inferences locally on-device.

In this study, a machine learning on the edge algorithm was used to translate sign language gestures to spoken language by developing an Embedded Intelligent System that uses sensors to track finger curvatures and hand movement. The device is first designed and built using open-source hardware. Then used to create a dataset by collecting data. Data is collected by performing the signing of various sign language gestures. The data collected is curated and used to provide examples for various classes to the k-nearest neighbour. This algorithm is then used to perform on-device inferencing to classify new gestures as they are signed. Arduino nano BLE sense is used together with flex sensors. As flex sensor bends with signing of different letters and numbers, the data is collected and logged in a file. This data is then used to train a model using a K-nearest neighbour (KNN) algorithm. New signed numbers are translated and displayed on the serial monitor.

The main contribution of the paper is a new machine learning on the edge approach using open-source resources to create a sign language translating device that works on resource constrained devices without the reliance of external inference engines

Table of Contents

DECLARATION i

ABSTRACT ii

ACKNOWLEDGEMENTS iii

LIST OF FIGURES vi

LIST OF ABBREVIATIONS vii

1. CHAPTER ONE: INTRODUCTION 1

1.1. Background 1

1.2. Problem Statement 3

1.3. Objectives 4

1.4. Justification 4

2. CHAPTER TWO: LITERATURE REVIEW 5

2.1. Introduction 5

2.2. Related Works 5

2.3. Conclusion 8

3. CHAPTER THREE: RESEARCH DEISGN AND METHODOLOGY 9

3.1. Research Design 9

3.2. System Design 10

3.3. Architectural Design 11

3.4. System Analysis 12

3.4.1. System Requirements 12

Overview 12

Objectives 13

Process 13

System requirements 14

3.4.2. Sign Language 14

3.4.3. Hardware Requirements and Analysis 15

Flex Sensor 15

Microcontroller 16

3.4.4. Software Requirements and Analysis 18

Embedded Software 18

Machine Learning & Deep Learning Algorithm 18

K-Nearest Neighbour (kNN) 19

3.5. Design 20
3.5.1. Hardware Design 20
Hardware Architectural Design 20
3.5.2. Software Design 22
Embedded Software Design 22
KNN Algorithm Design 24
3.6. Implementation 26
3.6.1. Hardware Implementation 26
3.6.2. Embedded Software Implementation 27
4. CHAPTER FOUR: RESULTS AND DISCUSSION 33
4.1. Introduction 33
4.2. Achievements & Discussion 33
4.3. Conclusions 33
4.4. Recommendations & Future Work 34
References 35

LIST OF FIGURES

Figure 1: Research Design 9
Figure 2: V - model design methodology adopted for embedded software development 10
Figure 3: System Architecture block diagram 11
Figure 4: Data Collection and Model Training Block Diagram 12
Figure 5: American Sign Language for numbers 0 to 9 14
Figure 6:Flex Sensor 15
Figure 7: Flex Sensor working principle 15
Figure 8: Flex Sensor - Voltage divider configuration 16
Figure 9: System Data Flow Diagram 20
Figure 10: Hardware Conceptual Architecture block diagram for data collection 20
Figure 11: System Hardware Architecture 21
Figure 12: System Hardware logical architecture block diagram 21
Figure 13: System software block diagram for data collection 22
Figure 14: Data collection algorithm design 23
Figure 15: Data management process 24
Figure 16: Machine Learning Model Creation Process 24
Figure 17: Machine learning workflow for sign language gesture recognition glove 25
Figure 18: High Level end-to-end gesture classification process block diagram 25
Figure 19: SL Glove Implementation 26
Figure 20: MCU - Voltage-divider - Flex Sensor Interface 27
Figure 21: Microcontroller Wiring 27
Figure 22: Testing Flex Sensors 27
Figure 23: Logging data from Serial Communication line to file 28
Figure 24: Starting the sensor data logging to file 29
Figure 25: Serial output on putty. 29
Figure 26: Dataset created from Logged output on file 30

LIST OF ABBREVIATIONS

AI	:	Artificial Intelligence
ASL	:	American Sign Language
BLE	:	Bluetooth Low Energy
CV	:	Computer Vision
DNN	:	Deep Neural Network
EIS	:	Embedded Intelligent System
IMU	:	Inertia Measurement Unit
IoT	:	Internet of Things
KNN	:	K-Nearest Neighbour
LCD	:	Liquid Crystal Display
ML	:	Machine Learning
SL	:	Sign Language
TFLite	:	TensorFlow Lite
W.H.O.	:	World Health Organization
WSN	:	Wireless Sensor Networks

1. CHAPTER ONE: INTRODUCTION

1.1. Background

Hearing or auditory perception is defined by (Plack, 2014) as the ability to perceive sounds by detecting vibrations, the changes in air pressure surrounding a medium through time, through an organ such as the ear. The loss of ability to perceive sound in humans is known as hearing loss. Hearing loss may be categorized as mild, moderate, severe or profound and may affect one or both ears. A person is classified as having hearing loss if they are not able to hear sound in the normal hearing threshold of 20dB or better in both ears. Factors leading to hearing loss are several as indicated by (World Health Organisation, 2021) and may occur in different period in one's lifetime. These includes Prenatal period with factors such as genetic factors and intrauterine infections. Other periods are Perinatal period, childhood and adolescence, adulthood and old age.

According to (World Health Organisation, 2021), over 5% of the world population requires rehabilitation to address their disabling hearing loss. They further estimate that by 2050, over 700 million people – or a tenth of the world population will have a disabling hearing loss.

Unaddressed hearing loss has impact on the daily living for the people involved both to the deaf person and the people they have to interact with on daily basis. The impact could be classified as individual impact or social impact which ranges from loneliness, social isolation and stigma. W.H.O estimates that unaddressed hearing loss poses an annual global cost of US $980 Billion in health sector costs, educational support, productivity loss and societal cost. The cost does not include hearing aid device costs.

People with hearing loss or hearing disability are considered as deaf. These people have profound hearing problem with very little to no hearing at all. The method often used by these people to communicate is the sign language. Also known as signed languages, these languages use visual-manual modality to convey meaning. (Sandler, 2006) says sign languages are full-fledged natural languages with their own grammar and lexicon.

In Kenya, a Kenyan Sign Language dictionary was published in 1991. More recently an online dictionary and a mobile application have been published since 2014 (Dictionary, 2015). In 1990s, Kenya National Association of the Deaf spearheaded a government program to train and employ teachers based in Machakos Teachers College to increase literacy in English and Swahili among the deaf community which was largely lacking despite the schools being established as early as 1960s.

The advancement in technology in recent times is a great enabler in achieving tasks that previously were difficult and near impossible. In particular machine learning technology, which is a part of artificial intelligence based on the idea that machines can learn from large amounts of data, identify patterns and make decisions independently or semi-independently with minimal human intervention. Embedding this kind of capability on embedded devices makes it possible to achieve what traditional embedded devices were unable to achieve.

Embedded Systems have been around and been used in various industries for control and automation of processes and for monitoring (Peckol, 2019). With rise of Wireless Sensor

Networks (WSN) (Akyildiz, I.F., Su, Sankarasubramaniam, & Cayirci, 2002), these embedded systems were able to connect and communicate with each other remotely and consequently, this have given rise to the now ubiquitous technology of Internet of Things (IoT) (Presser, 2016) whereby large numbers of such embedded systems (ES) interconnects and connects to the worldwide web. Furthermore, the development of light-weight machine learning algorithms that can run on resource-constrained embedded devices have seen rise of what is now called Embedded Intelligent Systems (EIS) as a result of embedding intelligent behavior into the traditional embedded devices using these Machine Learning (ML) algorithms such as Deep Neural Networks (DNN), referred to as Edge AI or Machine Learning on the Edge (Lee, 2018).

Despite the advancement on these technologies, their application on assistive technology applications remains limited. There are notable applications in other assistive technologies including Microsoft's Soundscape (Microsoft, 2017) and GesturePod (Shishir G. Patil, 2019). The availability of Open-Source hardware and Software in recent times has also accelerated the development of hardware and software solutions in timely and cost-effective manner.

Using these technologies, the problem of sign language sign recognition can be tackled with relative ease. The process would require collecting sufficient data on sign language gestures, training machine learning algorithms with this data and running the models in embedded device which performs inference and determine what the gesture made means.

In this study, Arduino Nano 33 BLE Sense -an open-source hardware platform from Arduino and TensorFlow Lite for Microcontrollers from Google is used to develop a device that will be able to detect and accurately categorize gestures made in sign language. The microcontroller chip is the nRF52840 running at 64MHz, has 1MB Flash Memory and 256KB SRAM. The nano 33 BLE Sense has several on-board sensors including a 9 axis Inertia measurement Unit (IMU). Other sensors to be used is 10 flex sensors, one for each finger. The TensorFlow Lite is an open-source framework that enable creating quantized machine learning models efficient and small enough to run on low-resourced platforms such as microcontrollers.

1.2. Problem Statement

Humans are social being and as so there is a lot of interaction amongst each other. Communication is one such interaction and arguably one of the most basic human need. For effective communication to occur, the people involved need to understand each other. The problem arises when one person or both in an interaction have a hearing difficulty or are deaf. We have invented Sign Language for such occasions whereby one signs using hands and the deaf can understand and sign back thus allowing effective communication without depending on hearing. With projected 1 in every 10 people being deaf by 2050, there is a need to develop a means of interpreting the hand signals to be able to communicate with people who do not understand the sign language. Moreover, 80% of the projected number coming from low-and middle-income countries, and over 25% being 60 years and older (World Health Organisation, 2021), the solution need be practical, low-cost for affordability, light-weight and scalable.

This study follows and is limited to using the American Sign Language (ASL) Numbers

The main objective of this study is to develop a sign language translation device and answer the following questions:

i) Is it possible to develop a machine learning model for sign language gesture recognition for numbers 0 – 9 using a cheap hardware microcontroller, flex sensors and a machine learning algorithm?

ii) What accuracy can be achieved by a machine learning model using sensor data from flex sensors?

1.3. Objectives

The goal of this research is to develop a power-efficient, light-weight and low-cost sign language gesture recognition using open-source hardware, specifically the Arduino nano 33 Ble Sense and flex sensors. In particular, the study has the following sub-objectives:

1. Assess the accuracy of using Machine Learning Frameworks in specific TinyML ArduinoKNN are suited for sign language gesture recognition on embedded device
2. Develop a sign language gesture recognition device for numbers 0 to 9 based on on-device inferencing

Further, the study focuses and is limited to gestures that represents numbers (0 to 9) which form a basis for counting. The outcome of this research will be of great value to software and hardware developers interested in developing Assistive Technology applications for the Deaf people. Moreover, it will form a basis for further research work on translating other sign language gestures by running on-device inference and can be extended to translate alphabets, words and phrases that require full hand movement and two hands signing. For the deaf user the device will be an efficient means of communication with other people including those who do not understand the sign language.

1.4. Justification

This study focuses on performing machine learning on edge devices. Most of the edge devices are low-resource with tight time constraints, low-power and small amount of memory. Due to these limitations, it has been very difficult to have machine learning models that can run on them and therefore most embedded devices have always relied on rule-based programs. With introduction of TinyML and TensorFlow Lite for Microcontrollers, machine learning capability, however limited, has been brought to these low-resourced devices. Having these abilities on microcontrollers means inferencing can be done right there on the device without having to rely on a network connection to a cloud service. This in turn translates to having better data security and privacy since no data ever leaves the device. Therefore, worth experimenting for this particular application of gesture recognition, being able to recognize gestures on tiny embedded devices using the power of machine learning without depending on hefty hardware and fast network connections.

2. CHAPTER TWO: LITERATURE REVIEW

2.1. Introduction

Communication is a two-way process for sharing information. It is an essential part of getting along with others and getting things done. The communication modes can take different forms including writing, verbal and body language. It is simply straight forward when people involved can talk and both understand the language being used.

Sign language is the language used by the mute and the deaf. People who have lost their ability to hear uses signs to communicate amongst themselves and to communicate to people who can hear normally. Sign language is the basic means of communication for these people. However, majority of the population do not understand sign language and therefore needs a translator (Dabre & Dholay, 2014).

2.2. Related Works

Many solutions have been proposed and explored for sign language translation. There are two major approaches to this problem. Glove-based solutions and using the computer vision approach.

i) Computer Vision (CV) approach

Computer vision is defined as a field of artificial intelligence (AI) that trains computers to interpret visual world using digital images and videos, deep learning algorithms (SAS, 2021).

In their study (Dabre & Dholay, 2014), uses a camera system to capture images which are then used to train a model to translate the gestures. A similar study by (Bantupalli & Xie, 2019) uses the ASL dataset together with CV techniques to extract temporal and spatial features from video sequences using a Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) respectively. These approaches achieve high accuracy on the training data for example (Kartik, Sumanth, Ram, & Prakash, 2020) achieved an accuracy of 98.67% from a dataset of 87,002 images trained on 78,300 images and test size of 8,700 images. Despite the high accuracy, CV approaches rely on expensive and cumbersome hardware. A computer is always required to run inference making it impractical to use.

ii) Glove Approach.

The Oxford dictionary defines a glove as a covering worn on the hand having separate parts for each finger and thumb. Sign-language gloves have been explored since 1980s when researches started to explore ways humans can interact with computers using gestures. In a study carried out in 1988, Stanford University researchers (Kramer & Leifer, 1988) proposed to develop “the talking glove” a device that could analyse a non-vocal person’s finger spelling and hand formation and output the spelled word with synthesized voice. Further, using a voice recognition equipment, the deaf user could read an incoming text on a miniatured LCD screen on a modified wrist watch and deaf-blind could read on a portable braille display. The complete system was cumbersome and very expensive. The system cost without the CyberGlove itself was US $3,500. (David J & Zeltzer, 1994). Several other translation gloves have been proposed over the years and some won awards like Ryan Patterson in 2001, a high school student from Colorado that used a leather glove with 10 sensors that monitored each finger position, relayed that information to a computer for display as text. In 2015, two Mexican researchers at Mexico's National Polytechnic Institute developed a similar glove that used a Bluetooth module to communicate with an Android phone.

The most recent publication by (Zhou, et al., 2020) translation glove published in June 2020 uses assisted stretchable sensor arrays and achieved an accuracy of 98.63% with a recognition time of less than 1s. In their study, (Zhou, et al., 2020) used proprietary technology including integrated wearable sensor arrays and nanotechnology.

Another glove approach as explored by (Vutinuntakasame, Jaijongrak, & Thiemjarus, 2011) was using Hall Effect Magnetic Sensors (HEMS) to detect applied magnetic field determining voltage variations in an electrical conductor. The approach places the HEMS at the tip of the fingers and a magnet at the palm of the hand. It generates a voltage of between 0.1v and 0.4v that is then passed on to a microcontroller. This approach however depends on an external hardware, the Odroid XU4 minicomputer for gesture recognition and classification making it quite a large device that is bulky, impractical and is power inefficient to power bot the minicomputer and the Arduino.

For the training datasets, most of the literature that exist has used a sample size of between 80 samples to as many as 1400 samples. The following table covers some of the details about the devices, gestures and samples per gesture when signing for American Sign Language.

Gestures	Components	Size of the sample	Author
4 gestures	5 flex sensors		(Praveen, Karant, & Mega, 2014)
Alphabets A -H	5 flex sensors Accelerometer Contact sensor	80 samples (10 samples each letter	(Sharma, Verma, & Khetarpal, 2015)
26 Alphabets	5DT Glove	234 samples (3 for each letter)	(Iwasako, Soga, & Taki, 2014)
Alphabets A-Z and numbers 0 -9	8 touch sensors	1080 samples (30 for each)	(Fu & Ho, 2008)
Alphabets A – Z	5 flex sensors Accelerometer Contact Sensors	260 samples (10 entries for each alphabet)	(Elmahgiubi, Ennajar, Drawin, & Elbuni, 2015)
120 static gestures	Flex sensor Contact sensor	3600 samples (100 for each)	(Ahmed, S.M.B., & Qureshi, 2010)

29 letters	10 flex sensors 3-axis accelerometer	1450 samples	(Vijayalakshmi & Aarthi, 2016)

All the listed studies have been conducted using different sample sizes. The larger the sample sizes are used since classification is done off the device using an extra hardware device, minicomputers. The larger the sample size, the more memory is used up. For the low memory microcontrollers this would be a problem and thus the need to have an efficient system that classifies gestures using a small sample size of less than 10 for each gesture without compromise on the accuracy.

According to (Ahmed, Z., Aws, Mahmood, & Muammad, 2018), the SLR systems have been plagued with low accuracy, not able to keep up with real-time recognition and unable to track the human hand that has multiple degrees of freedom. In their study, (Abdulla, Abdulla, & Manaf, 2016) highlights the challenges in obtaining a high precision for fast movements when signing naturally like in an actual conversation. Also lack of quality datasets (Arif, Rizvi, Jawaid, Waleed, & Shakeel, 2016) for sign language is another problem facing gesture recognition systems. This is critical problem especially for machine learning applications that often needs large datasets to learn from.

To develop a sign language translating device, there are considerations to make for the various stakeholders. The researcher could put an emphasis on creating quality datasets, analyse the various sign language variations in use, the sensor types and numbers and also put an emphasis on hybrid systems that includes body movements, facial expressions and two hands signing (Ahmed, Z., Aws, Mahmood, & Muammad, 2018). The developer should focus on developing an inexpensive system that is affordable to the people most afflicted by hearing disability. According to (Bajpai, Porov, Srivastav, & Sanchan, 2015) most people afflicted by this disability are poor and live below the poverty level and thus affordability should be one of the top considerations while developing the solution (Vijay, Suhas, Chandrashekhar, & Dhananjay, 2012). Another critical consideration according to the study by (Gupta, Singh, Pandey, & Solanki, 2015) and (Bajpai, Porov, Srivastav, & Sanchan, 2015) is real-time recognition, this is to ensure the gestures are being recognized and interpreted as they happen for a natural and fluid communication. Portability is another thing to consider as the solution needs to be moved around by the user in order to be usable even when not connected to external hardware like personal computers. In several studies, notably by (Tanyawiwat & Thiemjarus, 2012), (Vijayalakshmi & Aarthi, 2016) and (Trottier-Lapointe, et al., 2012) have listed portability as one of the features to consider for a practical sign language translation device.

From the Literature review, most of the work on sign language translation is either based on computer vision approach, that is computing resources intensive or the glove approach which relies on classic programming paradigm or using mobile phones for running the inference. Further, considering the highlighted considerations by (Ahmed, Z., Aws, Mahmood, & Muammad, 2018) on device costs, accuracy and portability, most of the reviewed work falls short in one property or the other. Research on running inference on the edge, on the glove itself without relying on PC or Mobile phones is still scarce.

2.3. Conclusion

The mainstream approaches to sign language translation are using the resource intensive Computer Vision approach or the network dependent glove approach. In order to achieve ubiquitous, low-cost, low-resource machine translation of signed language, there is a need to shift from the traditional approaches of using resource intensive techniques like CV that requires model development, training and running inference to be performed on the PC. With recent development of Machine Learning on the Edge, like the release of Machine Learning Frameworks like TinyML and TensorFlow Lite for Microcontrollers, it is essential to explore the possibilities of running inference for machine translation right there on the device without relying on any additional hardware. Moreover, exploring open-source, low-cost hardware and software, will make it possible for relatively fast and easy development and affordable solutions. This will also ensure that the most critical properties of cost considerations, accuracy, real-time and portability highlighted in the study are met.

3. CHAPTER THREE: RESEARCH DEISGN AND METHODOLOGY

3.1. Research Design

This study adopts design and creation strategy whereby an artefact will be developed and analysed. According to (Saltuk & Kosan, 2014) design science research "creates new interesting knowledge, demonstrates academic qualities and focuses on improvement, invention and exaptation" using an artifact. Furthermore, the device is an embedded intelligent device and therefore will include the phases of embedded systems development. The development will follow process in figure 1.

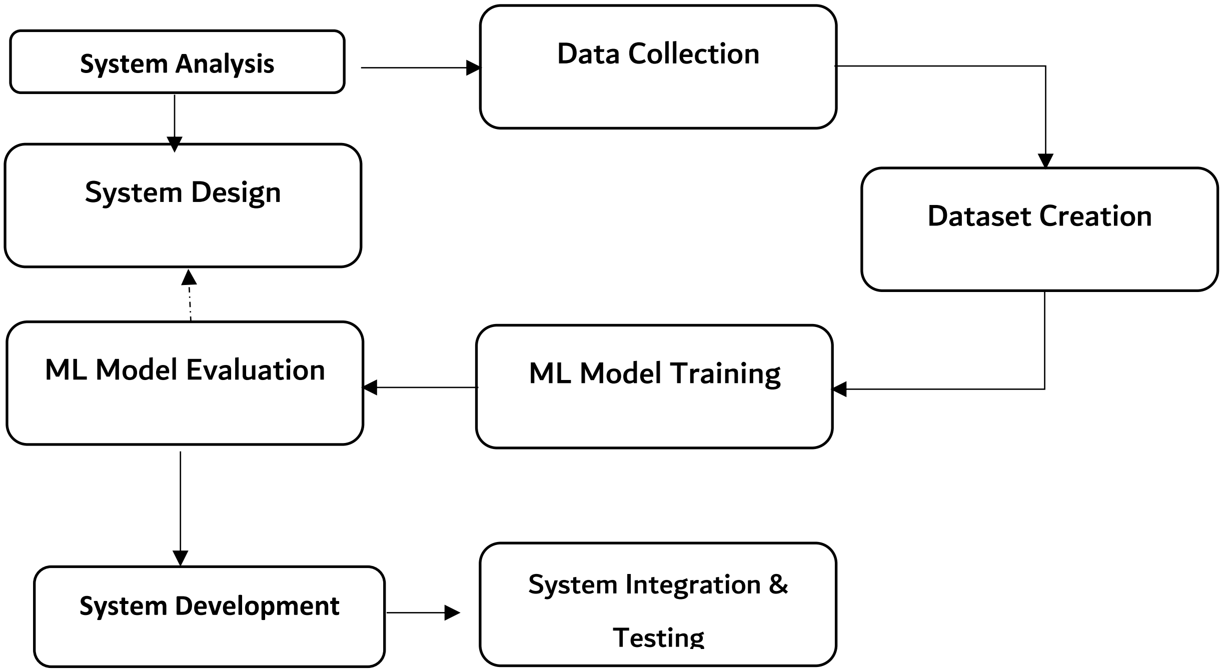


Figure 1: Research Design

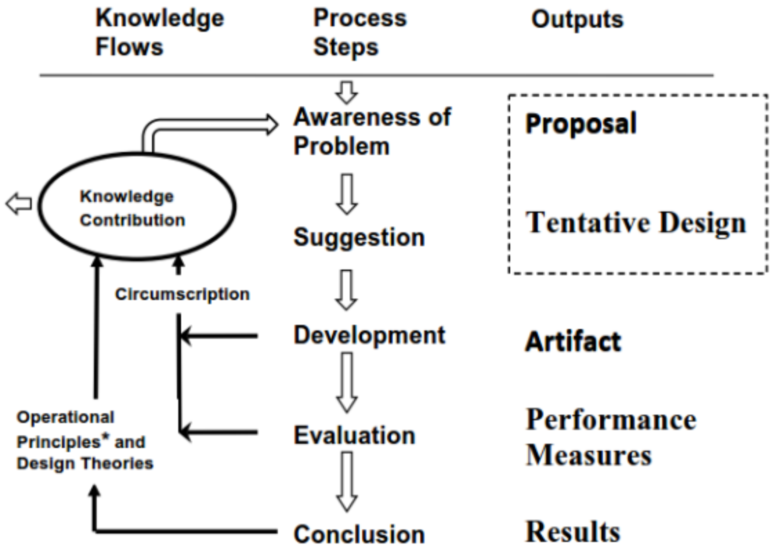


Figure 2: Design Science Research Process Model as adopted from Saltuks' Design and Creation

This study adopts the Design Science Research Process model by (Saltuk & Kosan, 2014). The figure 2 outlines the design and creation process.

3.2. System Design

Design goals are:

i) Arduino nano 33 BLE Sense and Flex Sensor based Sign Language Gesture recognition device

ii) Scan and analyse Flex sensors inputs

iii) Output data in csv format

iv) Basic Hardware (Arduino Nano 33 BLE Sense with 64MHz clock, 1MB Flash memory, 256kB SRAM, ADC/DAC and I2C, SPI, USB, I2S and UART Interfaces.

In this study, the V-Model development methodology is adopted. The V-model is a software development life cycle (SDLC) model where the process executes in a sequential manner in V-shape (Forsberg & Mooz, 1991). It is also known as the Verification and Validation model. In this method, each development stage is associated with a testing phase. Each phase must be completed before moving on to the next phase. The figure 2 below illustrates the various steps undertaken for development, verification and validation.

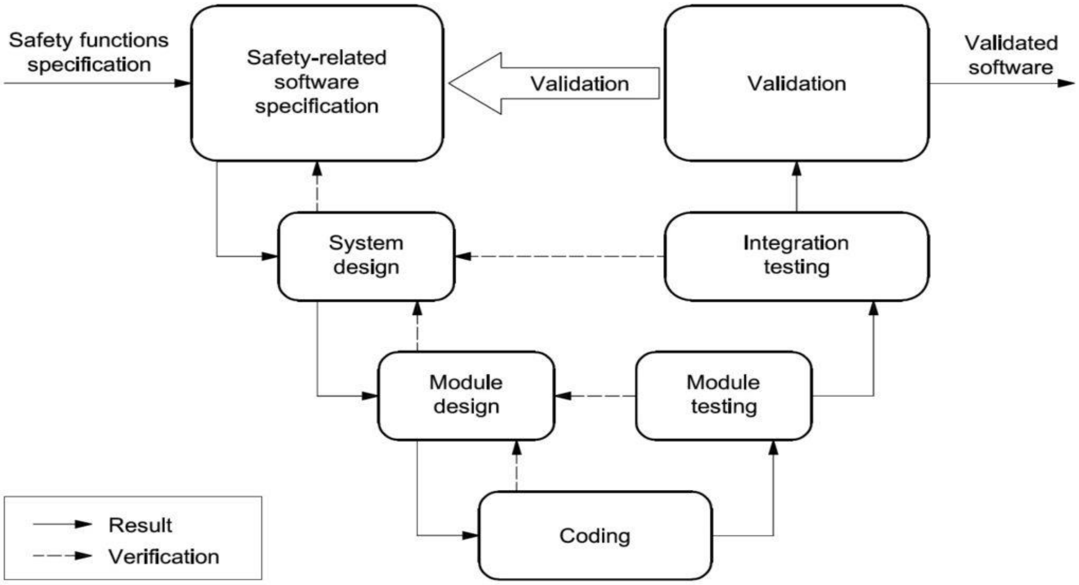


Figure 3: V - model design methodology adopted for embedded software development

3.3. Architectural Design

System design is broken down further into modules taking up different functionalities. The data transfer and communication between internal modules with other systems is understood (Kumar, 2019).

The following block diagram shows the system architecture. This architecture is used for both data collection and for running inference on the embedded device.

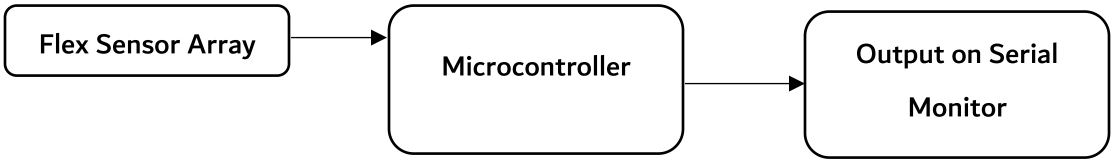


Figure 4: System Architecture block diagram

Coding

Coding is the process of writing embedded software for the components used to build the system or embedded device. For a larger system it is known as software development.

The platform used for the purpose of this development is the Arduino platform which is an integrated development platform with many libraries for most of the sensors existing in the market. Using this library in the development process saves on time and overall development cost of an embedded system.

There are 3 program files for the 3 development phases:

1) Data collection phase program that uses the sensors to collect and log data in csv format
2) Machine learning phase where the data is used to train a model that is used as a classifier and
3) The classifier program that is uploaded to the embedded device to capture new data and use the loaded model to classify it.

Data Collection

Data collection to form a dataset that will be used to train the classifier instead of using a ready-made dataset. The following diagram shows the overall system design for data collection.

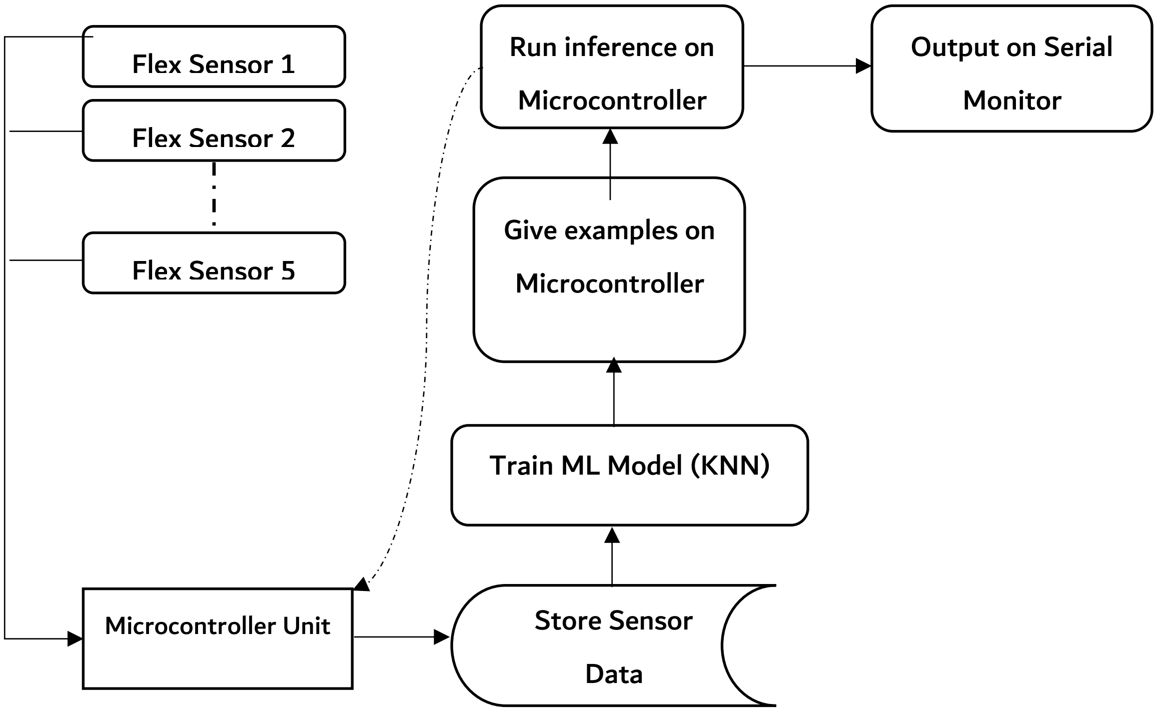


Figure 5: Data Collection and Model Training Block Diagram

Training and Testing data used in this study will be obtained from the 5 flex sensors.

Model will be created using TensorFlow Lite for Microcontrollers framework on PC. Testing and Validation will be done on PC before being converted and uploaded to the Microcontroller Unit (MCU). Once the model is uploaded into the MCU, the inference will be performed on the device itself without relying on the PC.

3.4. System Analysis

3.4.1. System Requirements

Overview

In the analysis phase, the requirements and constraints -the limits within which the system must operate- of the proposed system are explored. The requirements are the specific parameters that the system must satisfy. These requirements are outlined below and further translated into a more detailed specification.

Objectives

The main objective of this project is to develop a Sign-Glove (SG) that captures the gestures, finger movement of the person signing using sign language. The SG will be worn by the person doing the signing, the gestures are captured and translated appropriately. Specifically, this study will deal with interpreting numbers 0 to 9 signed using the American Sign Language (ASL)

Process

To develop this system, several factors are considered, the hardware and software requirements are discussed in-depth in the following subsections.

In this development project, several factors are considered. This device needs to be light weight, use less power (less than 200mW) and be accurate while recognizing and translating gestures. These measures considered for analysis are listed below.

i) Accuracy

It is the difference between the expected values and the actual recorded values. The accuracy of the device should be above 80%

ii) Precision

This is the number of distinguishable measurements. For the purpose of this project, only ten distinguishable measurements are required. Each measurement for each number.

iii) Size and Weight

These qualities represent the physical space occupied by the device. Since it’s a hand worn device, the package should be small so as to fit comfortably behind the glove. The physical dimensions should not exceed 50mm by 60mm

iv) Power

Power is the total amount of energy required to run a system. This is a crucial requirement since the device is to be used as a wearable and therefore running on battery power. This requirement however, is not strictly followed for this prototype development since it will be running on USB power from the PC.

v) Time-to-prototype

This is the total time required to design, build and test a prototype system. For this project, the device should be complete within the duration stipulated for the purpose of research project (March to July).

System requirements

The system should be able to

i) Track the finger movements as numbers are signed

ii) Accurately recognize the number signed

iii) Accurately categorize the number signed using a machine learning algorithm

3.4.2. Sign Language

Sign Language is different and distinct from other spoken languages like English. It contains its own fundamental rules on pronunciation, word formation and expressions. However, just like other languages, sign language, specific ways of expressing ideas vary from one region to another and just as spoken languages, expression can change depending on other factors such as age variation and context.

Only signing numbers has been selected to reduce complexity of the design given the limited time I have to work with. Moreover, the more gestures I incorporate, the larger the ML model will be generated and therefore running the risk of not fitting on the limited flash space and RAM on the nano board. To prevent this, and to demonstrate that ML on the edge is possible for gesture translation, only numbers have been chosen for the proof of concept. Further, the ASL has been chosen since it signs numbers using only one hand unlike Kenyan Sign Language that uses both hands to sign numbers. The diagram below illustrates how to sign numbers using ASL. These are the gestures we will be looking to translate using a machine learning model running on Arduino nano.

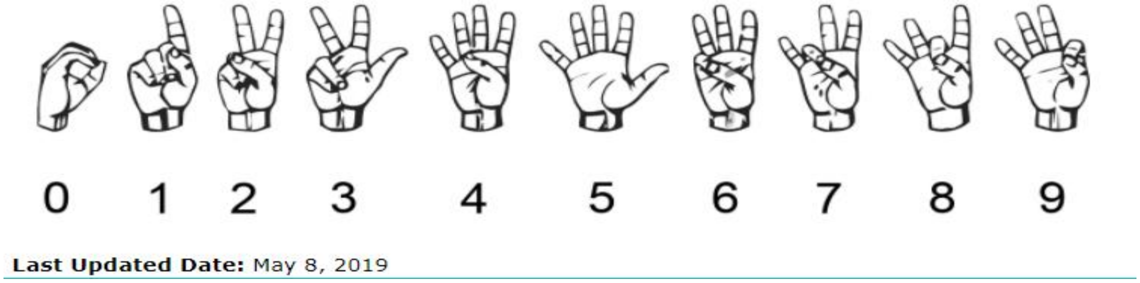


Figure 6: American Sign Language for numbers 0 to 9

In embedded system, design and development is divided into two major categories, Hardware and embedded software. These two must work in tandem, with the development of embedded software being informed by the choice of hardware, i.e., processing unit and sensors.

3.4.3. Hardware Requirements and Analysis

This section goes into details on the different hardware used for the purpose of capturing gestures, converting them into electrical signals and interpreting them. Hardware and specifically sensors and the microcontroller in this project plays a crucial role of capturing analog data that will be used for both training a machine learning algorithm and testing the resulting model to evaluate metrics such as accuracy levels, precision and resolution.

The hardware required, both sensors and microcontroller unit are discussed in the following subsections.

Flex Sensor

In order to capture finger movement during signing, to capture the gesture made by the hand and the fingers, an electromechanical sensor is required. Folding and straightening of fingers is the action required to gesture numbers; unlike other gestures that may require whole arm movement, therefore a flexible sensor that translates these physical movements into electrical signals is required.

A flex sensor, also called a bend sensor. It measures the amount of deflection of the surface on which it is stuck. The working principle of a flex sensor is based on resistivity as material is bent since there is a direct proportionality between bending and resistance. These sensors are used in many applications and areas of research from Human Machine Interface devices (Saggio & Giovanni, 2014) to Rehabilitation research (Sreejan & Yeole, 2017). Flex sensors come in various form like Fibre optic flex sensor, capacitive flex sensor and velostat flex sensor.

Working principle

Velostat flex sensor is made up of polymeric foil that has been impregnated with carbon black. This carbon makes it electrically conductive. It is inexpensive and its ability to change resistance as it bends makes it suitable for our gesture sensing application. The figure below shows a flex sensor

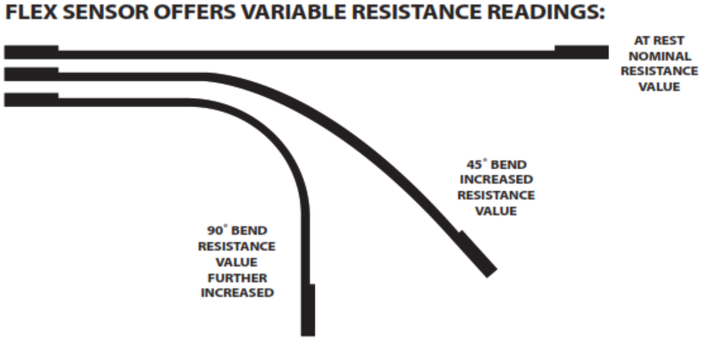

FLEX SENSOR OFFERS VARIABLE RESISTANCE READINGS:
AT REST NOMINAL RESISTANCE VALUE
45° BEND INCREASED RESISTANCE VALUE
90° BEND RESISTANCE VALUE FURTHER INCREASED

Figure 8: Flex Sensor working principle

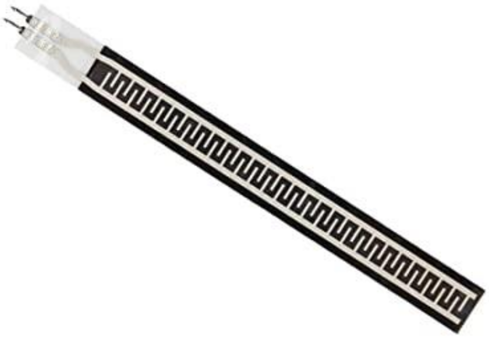


Figure 7:Flex Sensor

The sensor has a nominal resistance when held straight. When the substrate is bent, however, there is a corresponding proportional increase in resistance to the bend radius, the farther you bend the sensor, the higher the resistance values. In order to take advantage of this property, the flex sensor is used as a variable resistor or a potentiometer together with another resistor in a voltage divider configuration as shown in the circuit diagram below.

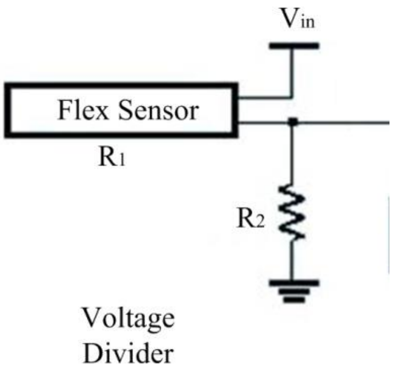


Figure 9: Flex Sensor - Voltage divider configuration

R1: Flex Sensor

R2: Resistor

Vin: Input voltage (3.3v)

Vout: Output voltage from the voltage divider

The output voltage is calculated using the following equation:

$$Vout = Vin(\frac{R1}{R1 + R2})$$

The output voltage (Vout) is an analog voltage that is fed into an analog input pin of our microcontroller.

Microcontroller

The microcontroller is the brain of the whole device. It contains the processing unit, flash memory and RAM. Since for this project we need an MCU capable of running inference of TensorFlow Lite for Microcontrollers machine learning model, a 32-bit processor is required. Therefore, the choice of Arduino nano 33 Ble sense. The Nano 33 BLE Sense is Arduinos 3.3V, small form factor (45mm x 18mm) AI enabled development board. It's a recently released and is loaded with a number of embedded sensors. The board features a relatively powerful low power processor, nRF52840 from Nordic Semiconductor. This processor is a 32-Bit ARM Cortex M4 CPU running at 64MHz, 128MB of RAM. Further, the processor features a Bluetooth pairing with Near Field Connectivity (NFC) and Bluetooth Low Energy (BLE) to pair and send data to external devices

These sensors include:

- Humidity & Temperature sensor
- 9 – axis IMU sensor for motion, vibration and orientation sensing
- Barometric sensor
- Digital Microphone
- Gesture, proximity, light color and light intensity.

Technical Specifications

Microcontroller	**nRF52840**
Operating Voltage	3.3V
Clock Speed	64MHz
Flash Memory	1MB
SRAM	256KB
Digital I/O	14
Analog Inputs	8 (ADC bit 200ksamples)

The possibility of running Edge AI on this tiny, low power device and the array of onboard sensors included is what is appealing and informed its selection among other embedded development boards in the market today.

The diagram below shows the nano 33 board and all its onboard sensors (eTechnophiles, 2019).

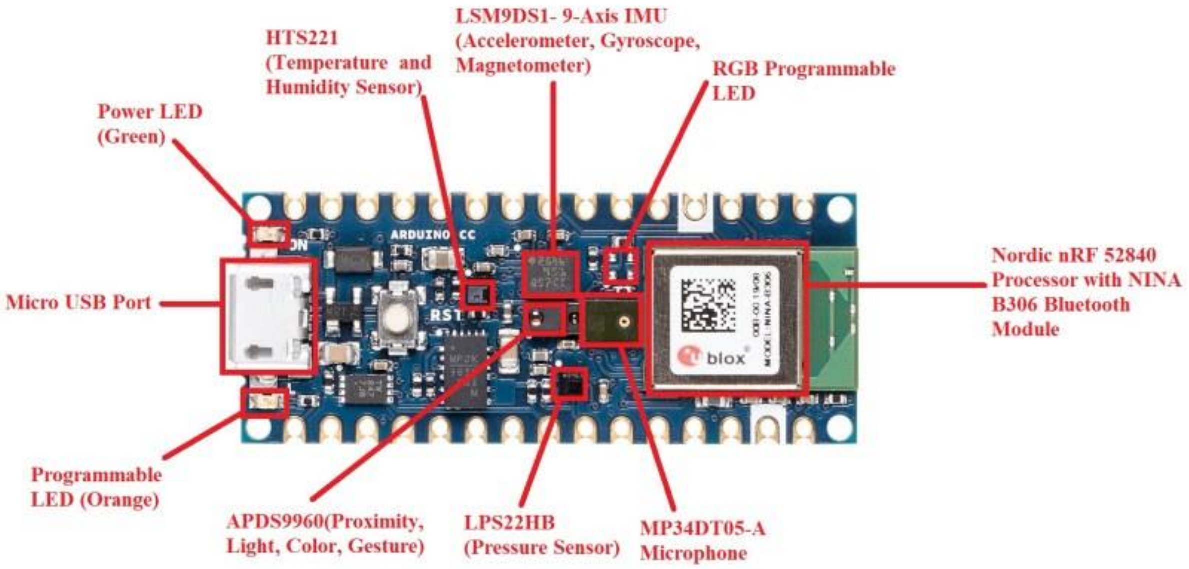

HTS221
(Temperature and Humidity Sensor)
LSM9DS1- 9-Axis IMU
(Accelerometer, Gyroscope, Magnetometer)
RGB Programmable LED
Power LED (Green)
Micro USB Port
Nordic nRF 52840 Processor with NINA B306 Bluetooth Module
Programmable LED (Orange)
APDS9960(Proximity, Light, Color, Gesture)
LPS22HB (Pressure Sensor)
MP34DT05-A Microphone

Figure 10: Arduino nano 33 Ble sense development board

The analog voltage (Vout) from flex sensor – resistor voltage divider is fed directly to 5 of the 8 analog inputs of the nano 33. These pins are directly connected to a 12-bit analog-to-digital convertor that quantizes and samples the voltage and converts it into a digital signal for processing by the CPU.

3.4.4. Software Requirements and Analysis

The software is defined by (Rosencrance, 2019) as a set of instructions, data or programs used to operate computers and execute specific tasks.

This section analyses the software required for data acquisition, machine learning algorithm and embedded software to run on-device inference.

Embedded Software

Embedded systems are not typical computers since they are highly specialized, resource constrained devices both in time, processing power and memory. These requires embedded software to control these devices. The term embedded software is sometimes used interchangeably with firmware (Emilio, 2014).

There are two categories of embedded software that is required. That is, first, the software that captures data from our sign glove for all the ten gestures. Each gesture is unique and therefore each of the sensor on each of the five fingers will have a sensor voltage value. The data capture embedded software needs to capture these values and output them to a file in a specified format shown below.

Table 1: Data capture format

Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Number

Each column header for sensor represents the corresponding finger. E.g., Sensor 1 corresponds to Finger 1 which is the small finger, all the way to Sensor 5 on the thumb finger. Finally, on the last column we have the number (0 – 9) represented by the data entries on the same row.

The first program needs to read the ADC and from those values calculate the voltage and resistance. These resistances for each of the five sensors is then tabulated on the table above to create a dataset. The dataset created is then used for ML model development.

Machine Learning & Deep Learning Algorithm

Machine learning algorithm is defined by (Brownlee, 2020) as a procedure that is run on data to create a machine learning model. These algorithms perform pattern recognition by learning from data. ML algorithms can be put into three distinct classes; supervised, unsupervised and reinforcement learning. They can be further classified as classification, regression or clustering algorithms.

A model is “the output of a machine learning algorithm run on data” and represents what was learnt by the ML algorithm (Brownlee, 2020).

Deep Learning (DL) is defined by (Deng & Yu, 2014) as a class of machine learning algorithms that uses multiple layers to progressively extract higher level features from raw input. The simplest has three layers that is input layer, representational layer and an output layer.

In the case of supervised learning, whereby we have labelled data, deep learning removes redundancy in representation by translating data into compact intermediate representations (IR) similar to Principal Components and thus eliminates the need for feature engineering. The “deep” in deep learning simply means the number of layers that data is to be transformed through from input through the hidden layers all the way to the output. This chain of transformation from input to output is known as credit assignment path (CAP) and describes the causal connection between the input and the output. According to (Shigeki, 2019) CAP of depth 2 has been shown to be a universal approximator since it can emulate any function.

TensorFlow

TensorFlow is an end-to-end, open-source machine learning platform (TensorFlow, 2021). TensorFlow Lite for microcontrollers is a light weight TensorFlow Library designed and optimized to run on resource constrained devices. It does not require an Operating System to run, dynamic memory allocation nor standard C/C++ libraries. TF is available as TFLite for Microcontrollers for Arduino platform making it suitable to use for this particular application. Arduino nano BLE is the only supported Arduino boards as the time of this study. We shall use TFLite to create and run a 30kB Neural Net (NN) to recognize and identify gestures using flex sensors and IMU.

Limitations of TFLite for Microcontrollers

TensorFlow Lite for Microcontrollers is specifically designed to run on specific resource constrained microcontroller development unlike the standard TensorFlow Lite that runs on more powerful Linux based embedded devices like NVidia’s Jetson Nano and Raspberry Pi.

Working with TensorFlow Lite for Microcontrollers has the following specific limitations:

i) Supports a limited subset of TensorFlow operations, this affects the number of architectures that is possible to run on.

ii) Currently only a limited number of devices are supported.

iii) Requires manual memory management since it uses Low-level C++ API

iv) Does not support on-device training and therefore training needs to be done on a more powerful environment, and load the trained model.

K-Nearest Neighbour (kNN)

kNN algorithm uses distance between datapoints to classify the current data point. It is a supervised machine learning algorithm that is simple to implement and works well with structured data that is dependent on distance measure between the values read from .

For this particular project, an implementation of kNN is more suitable because it makes highly accurate predictions. Furthermore, the quality of predictions is dependent on distance measures. Moreover, it does not require additional tools to implement unlike TFLite that requires training and model creation done on a separate platform. Using kNN in Arduino platform requires only to provide data examples and class, therefore all implementation is done within the same platform.

3.5. Design

In this section the conceptual model of the hardware and software is built. In order to simplify the process, the project is broken down into modules and subcomponents. Also, during this phase, the cost, schedule and expected performance of the system are determined. The data flow diagram below shows the major components of the system and how data moves from one module to the other.

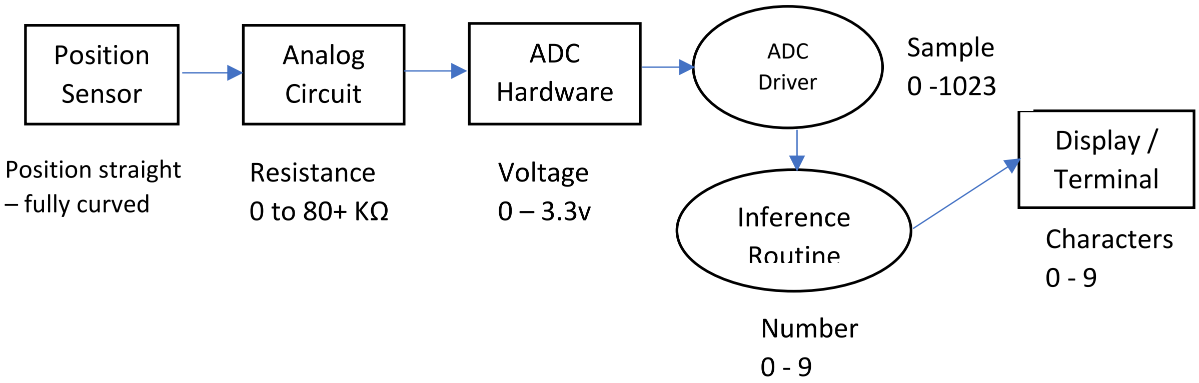


Figure 11: System Data Flow Diagram

In order to build the hardware prototype, we design it first, outlining its architecture using a high-level block diagram. The following subsections will focus on conceptual and logical architectures.

Hardware Architectural Design

i) Conceptual Architecture

The conceptual architecture focuses on identification and allocation of responsibilities to the components.

Data collection hardware conceptual architecture is shown in the figure below

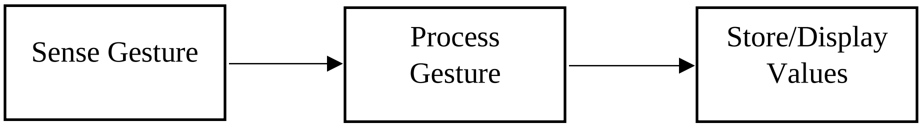


Figure 12: Hardware Conceptual Architecture block diagram for data collection

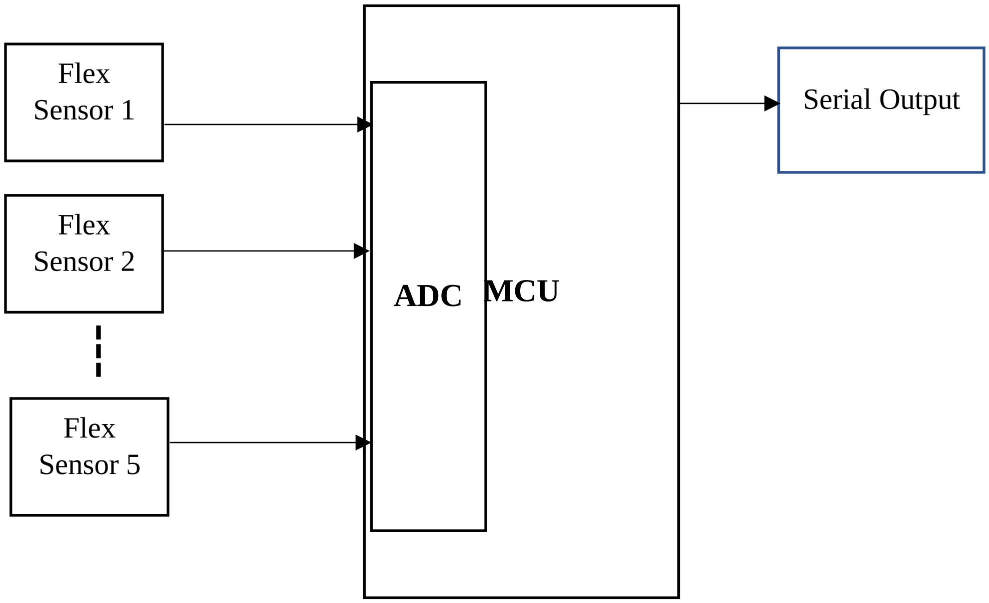


Figure 13: System Hardware Architecture

ii) Logical Architecture

This architecture focuses on components interaction, connection mechanisms and protocols, interface design and specification.

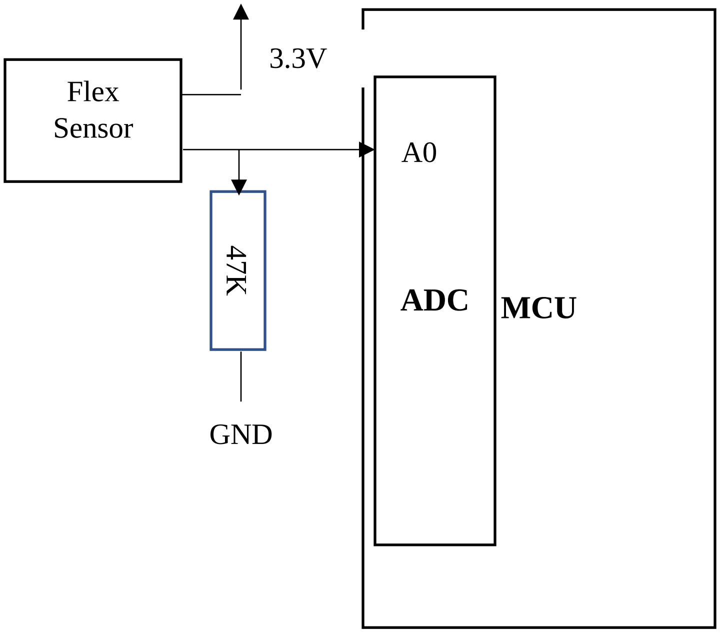


Figure 14: System Hardware logical architecture block diagram

For data collection and for inferencing purposes, the hardware configuration is similar. The only difference is the embedded program. For the purposes of data collection, the source program is purely rule based using the traditional programming practice to read from the sensors. The complete prototype however, uses both rule -based and machine learning model running with classification performed purely using machine learning model.

3.5.2. Software Design

Software design is defined by (Ralph, 2009) as "the process by which an agent creates a specification of a software artifact intended to accomplish goals, using a set primitive component and subject to constraints." The software for the project is divided into two categories. Category 1 is the embedded software that will run on the microcontroller and category 2 is the Python program that will create an ML model using google colaboratory in order to take advantage of the high processing capability offered by Google Colab.

Category 1, the embedded software is further divided into two, first the program for data collection, and second the program that performs gesture classification.

Embedded Software Design

Embedded Software is the program that runs on the nano 33. The figure 13 is a block diagram showing the data collection to build a dataset program logic.

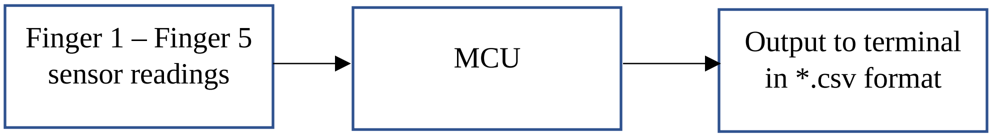

Finger 1 – Finger 5 sensor readings
MCU
Output to terminal in *.csv format

Figure 15: System software block diagram for data collection

```
Input Divider Resistor Value: R_DIV;
Input Voltage:  Vcc;
Input Nominal Resistance: R_FLEX
Input 90 deg bend resistance: R_BEND
Read Sensor 1: s1
Read Sensor 2: s2
Read Sensor 3: s3
Read Sensor 4: s4
Read Sensor 5: s5
VFLEX = sX * Vcc/1023
RFLEX = R_DIV * (Vcc/VFLEX – 1.0)
Output: RFLEX1, RFLEX 2, RFLEX 3, RFLEX 4, RFLEX 5
```

Figure 16: Data collection algorithm design

To collect data, each of the 10 gestures are repeated starting with 0. Every time capture the sensor reading of all the 5 sensors. This data is stored in comma separated files (CSV) in the format shown in table 2 below.

Table 2: Comma Separated Files format for data collected

File-1	**File-2**	**File-3**	**File-4**	**File-5**	**File-6**	**File-7**	**File-8**	**File-9**	**File-10**

Files File-1 to File-10 are the *.csv files (e.g., File-1.csv) holding data for the five fingers for each number represented by each of the 10 gestures. Each files row contains gesture data for each number with the format shown in the table below.

Table 3: Data format inside each file

Finger-1	**Finger-2**	**Finger-3**	**Finger-4**	**Finger-5**
RFLEX1	*RFLEX2*	*RFLEX3*	*RFLEX4*	*RFLEX5*

Row 1 of the table 3 above represents gesture representing number 0 in ASL.
The ten *.csv files holding data with five columns each makes up our dataset with which we later train our ML algorithm to create a model for gesture classification.
The whole process of collecting data and managing it can be describe using a simple block diagram shown below

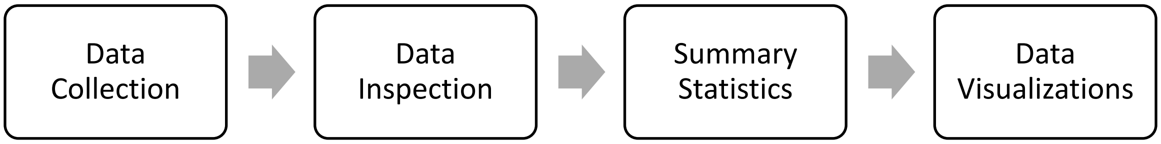


Figure 17: Data management process

KNN Algorithm Design

A machine learning algorithm is a generic program that is made task specific when trained with particular data, thus ML algorithm is a framework used to solve different problems depending on the data the algorithm will be trained with. Model creation is an iterative process that begins with defining the problem as shown in the block diagram below

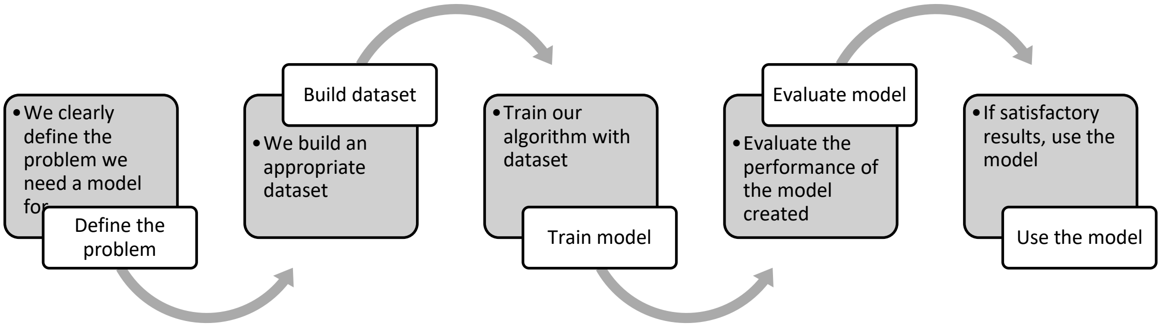

We clearly define the problem we need a model for
Define the problem
Build dataset
We build an appropriate dataset
Train our algorithm with dataset
Train model
Evaluate model
Evaluate the performance of the model created
If satisfactory results, use the model
Use the model

Figure 18: Machine Learning Model Creation Process

This project implements a kNN algorithm to classify gestures. This algorithm is a simple, easy to implement supervised machine learning algorithm. It can be used for both regression and classification problems. In this study, KNN is used for a gesture classification task.

kNN Algorithm Breakdown

a) Specify data examples with each their respective category
b) Initialize the k value, we have chosen 5 examples and therefore we can specify k value as <=5
c) For each example in the data
    a. Calculate the Euclidean distance between the current sensor data (query) and the current example
    b. On the ordered collection, add the distance and the index of the given example.
d) Sort the ordered collection of distance and indices in ascending order.

e) Select the first k entries from the sorted list
f) Return the mode of the k labels

The gesture is classified into the class label that has the highest number of entries.

ML Workflow.
The workflow outlines in details the process undertaken in developing the Sign-Glove ML model starting from data collection through to uploading the created model and running inference on the device's onboard microcontroller. These steps are:

1. Collecting Data
2. Cleaning Data
3. Input data examples and their class
4. Pass the examples to the classifier
5. Run Inference on device

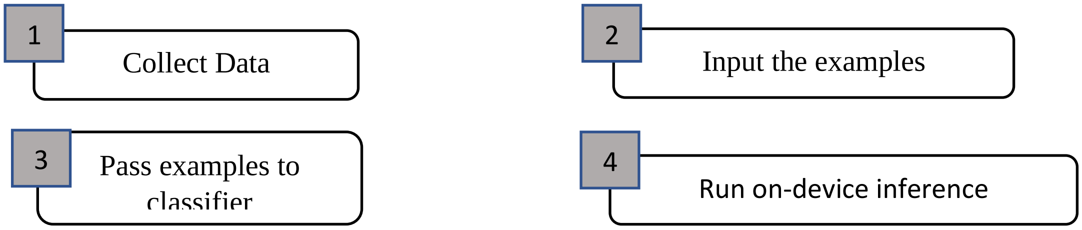


Figure 19: Machine learning workflow for sign language gesture recognition glove

The completed system is shown in a high-level diagram, figure 18 below. The Gestures from flex sensors representing each fingers position, that sensor data is passed on to the kNN algorithm. The algorithm uses the provided examples for each class, calculates the nearest neighbor, and classifies the recognized gesture into one of the 10 classes. The output is a number (0-9) and the confidence level between the lowest confidence of 0.0 and highest confidence level of 1.0.

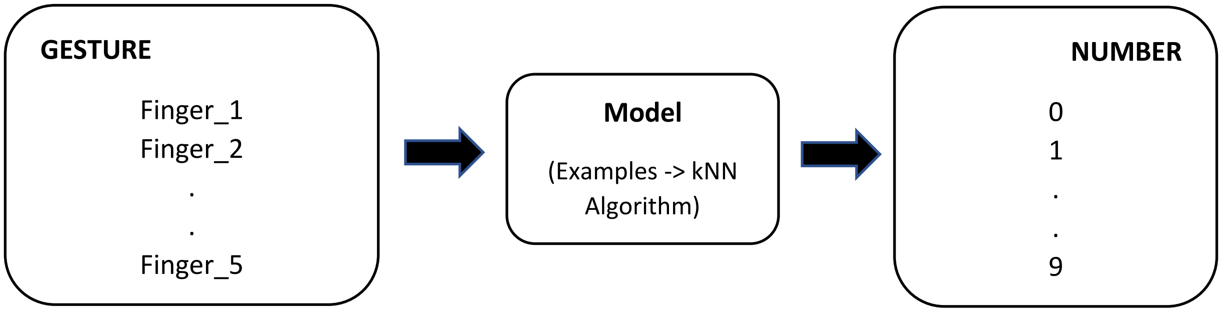


Figure 20: High Level end-to-end gesture classification process block diagram

3.6. Implementation

In this step, the system is implemented by building a prototype. It includes simulation and/or building a prototype with physical components. During this phase both hardware and software debugging was carried out. Due to the limitations of embedded devices, the debugging process is not as straight forward as debugging in computer systems. The lack of keyboard and mouse and also the fact that hardware is working concurrently with embedded software in real time which means it’s not possible to do the usual single-stepping and print statements debugging. To make the process easier, a cross- compiler and assembler was used to convert source code into object code for our target system, the nano 33.

In this project, a top-down implementation was adopted since it has an advantage of having the possibility of implementing the system sub-components simultaneously.

3.6.1. Hardware Implementation

i) **Control Circuit:** The control circuitry is made up of a microcontroller, the nano 33 BLE development board and an interface circuit. The connection is as shown in the diagram below.

ii) **Glove:** The Signing Glove hardware is made of a glove to hold on to the flex sensors, flex sensors in a voltage divider configuration using 47kΩ resistors with the output from the divide connected to analog pins of the Arduino nano. The diagram below shows a photo of the SG

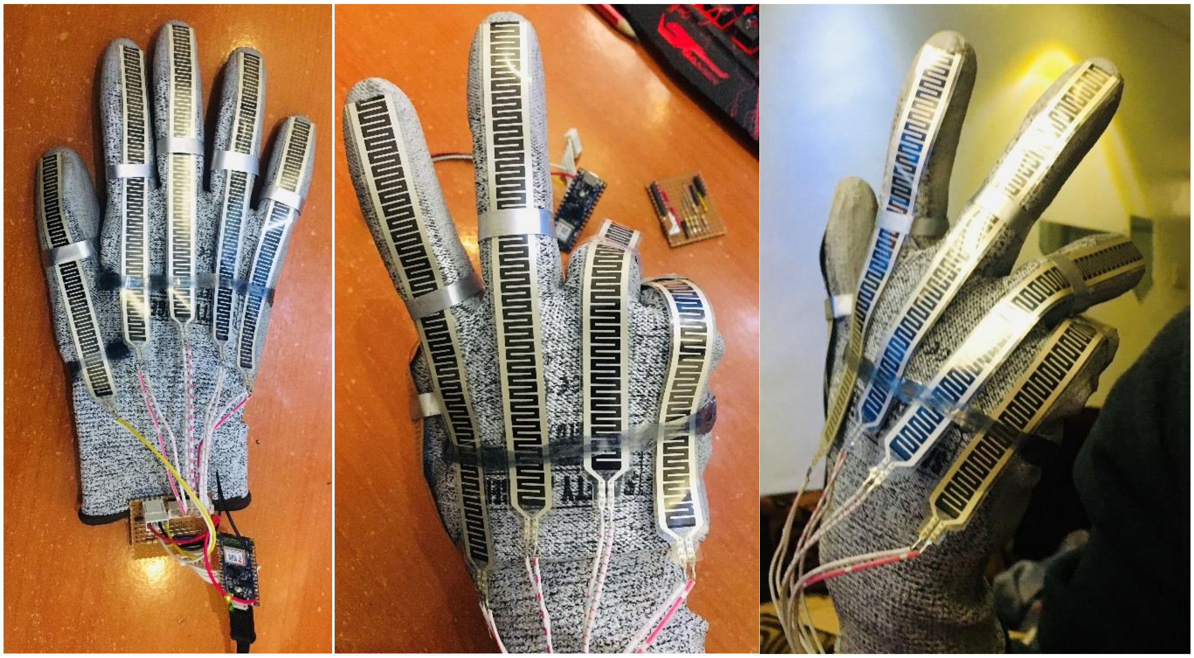


Figure 21: SL Glove Implementation

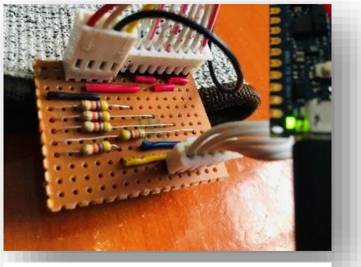


Figure 22: MCU - Voltage-divider - Flex Sensor Interface

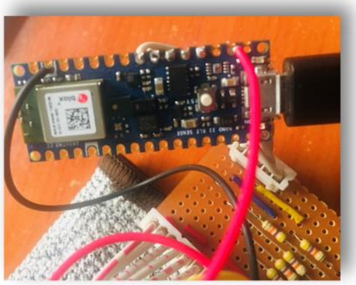


Figure 23: Microcontroller Wiring

This hardware implementation is used for both dataset creation and gesture classification.

3.6.2. Embedded Software Implementation

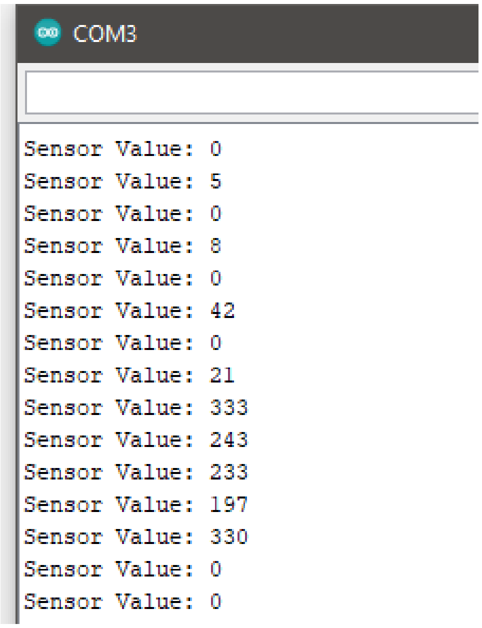
The project has two embedded software implementations. First used to collect data is shown below. The snippet is for a single flex sensor. The complete program for all sensors is on Appendix.

i) Program for Data Collection

- Read from flex sensors
- Format the sensor data into a csv format i.e. (s1, s2, s3, s4, s5)
- Log the sensor data from serial communication line to a file.

To test the flex sensors, program output is displayed on the Serial Communication port. The image below shows the test results of flex sensor 1.

As the flex sensor is bent, the resistance value increases and is logged as shown.



```
Sensor Value: 0
Sensor Value: 5
Sensor Value: 0
Sensor Value: 8
Sensor Value: 0
Sensor Value: 42
Sensor Value: 0
Sensor Value: 21
Sensor Value: 333
Sensor Value: 243
Sensor Value: 233
Sensor Value: 197
Sensor Value: 330
Sensor Value: 0
Sensor Value: 0
```

Figure 24: Testing Flex Sensors

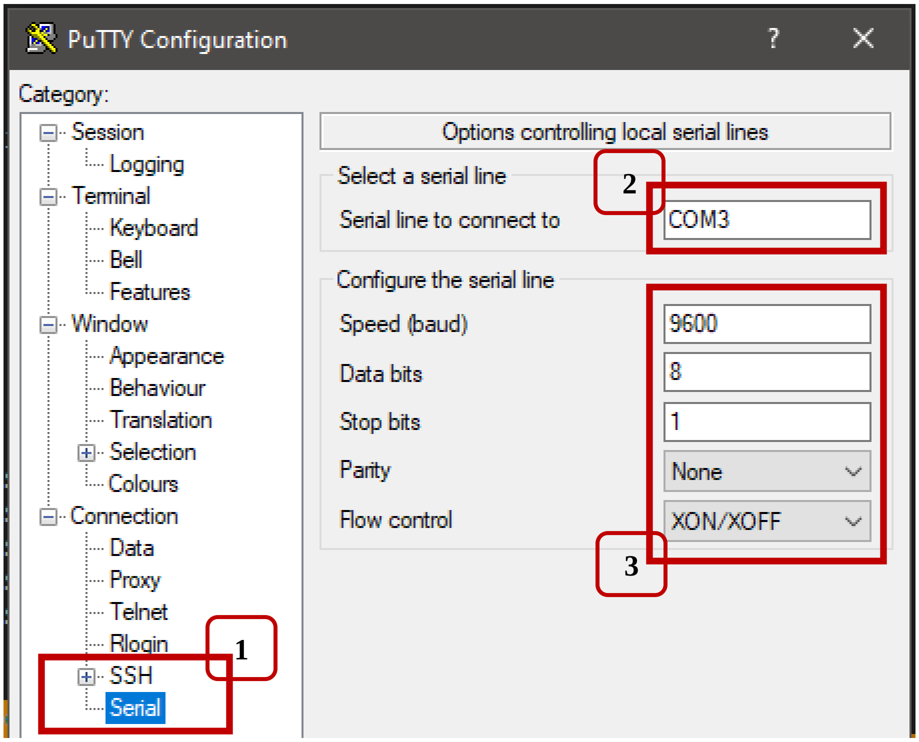
To create a dataset from all the 5 sensors, the following line of code is used. First, we print the column headers followed by sensor values separated by commas to Serial output. This output is then captured using putty as printable output and saved as a comma separated values (*.csv) file.

ii) Logging Serial Data to File

In order to log Serial data being output by nano 33, Putty – a telnet and ssh client – is used. The procedure is as follows

a) On the **Category** section, select **Session**

b) Fill in the **COM** details under "**options controlling local serial lines**" as shown in the image below. In COMx: x is the nano 33 com port.



Options controlling local serial lines

Select a serial line

Serial line to connect to: COM3

Configure the serial line

Speed (baud): 9600

Data bits: 8

Stop bits: 1

Parity: None

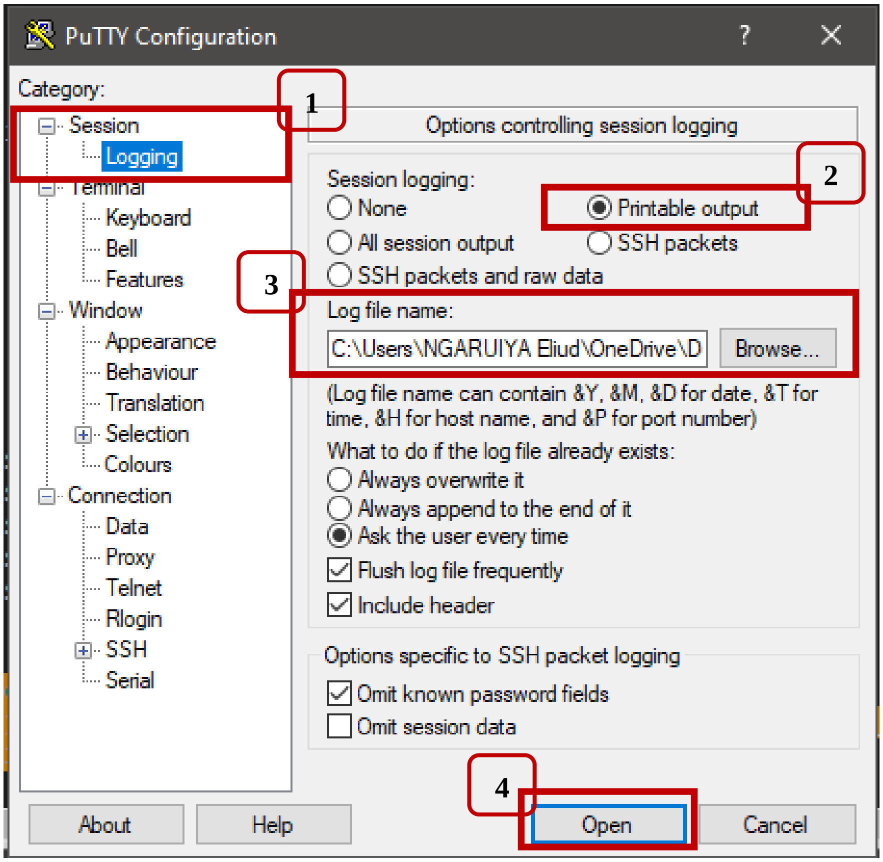
Flow control: XON/XOFF

Figure 25: Logging data from Serial Communication line to file

c) On the **Category** section, select **Logging**

d) On **Session logging** select **printable output** radio button

e) Specify the **log file name** as shown in the figure below



Options controlling session logging

Session logging:

○ None ● Printable output

○ All session output ○ SSH packets

○ SSH packets and raw data

Log file name:

C:\Users\NGARUIYA Eliud\OneDrive\D

(Log file name can contain &Y, &M, &D for date, &T for time, &H for host name, and &P for port number)

What to do if the log file already exists:

○ Always overwrite it

○ Always append to the end of it

● Ask the user every time

☑ Flush log file frequently

☑ Include header

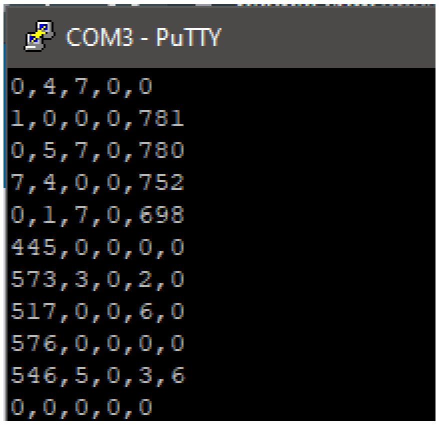
Options specific to SSH packet logging

☑ Omit known password fields

☐ Omit session data

Figure 26: Starting the sensor data logging to file

f) Click **Open.** This starts the logging session capturing serial data and printing it to a file. This is our dataset.



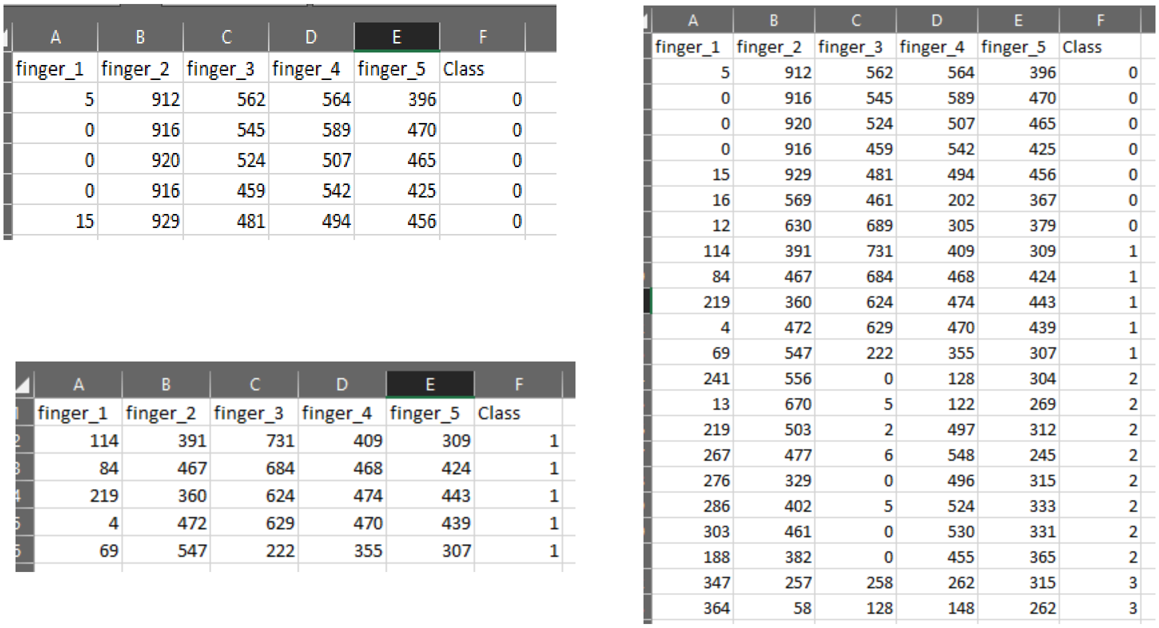
```
0,4,7,0,0
1,0,0,0,781
0,5,7,0,780
7,4,0,0,752
0,1,7,0,698
445,0,0,0,0
573,3,0,2,0
517,0,0,6,0
576,0,0,0,0
546,5,0,3,6
0,0,0,0,0
```

Figure 27: Serial output on putty.

iii) Data Curation & Dataset Creation

The sensor data is logged in multiple *.csv files, one for each number. This translates to 10 files. To create a dataset, the files were consolidated into a single file.

The diagram below shows a snippet of the csv file for numbers 0 & 1 and the curated file.



A	B	C	D	E	F
finger_1	finger_2	finger_3	finger_4	finger_5	Class
5	912	562	564	396	0
0	916	545	589	470	0
0	920	524	507	465	0
0	916	459	542	425	0
15	929	481	494	456	0

A	B	C	D	E	F
finger_1	finger_2	finger_3	finger_4	finger_5	Class
114	391	731	409	309	1
84	467	684	468	424	1
219	360	624	474	443	1
4	472	629	470	439	1
69	547	222	355	307	1

A	B	C	D	E	F
finger_1	finger_2	finger_3	finger_4	finger_5	Class
5	912	562	564	396	0
0	916	545	589	470	0
0	920	524	507	465	0
0	916	459	542	425	0
15	929	481	494	456	0
16	569	461	202	367	0
12	630	689	305	379	0
114	391	731	409	309	1
84	467	684	468	424	1
219	360	624	474	443	1
4	472	629	470	439	1
69	547	222	355	307	1
241	556	0	128	304	2
13	670	5	122	269	2
219	503	2	497	312	2
267	477	6	548	245	2
276	329	0	496	315	2
286	402	5	524	333	2
303	461	0	530	331	2
188	382	0	455	365	2
347	257	258	262	315	3
364	58	128	148	262	3

Figure 28: Dataset created from Logged output on file

Program for Gesture Recognition & Classification

- Pass the examples to the classifier and the number class
  example_1[] = {84,467,684,468,424};
  example_2[] = {219,360,624,474,443};
- Pass the examples to the classifier and the number class
  gestureClassifier.addExample(example_1, 1);
  gestureClassifier.addExample(example_2, 1);
- New sensor data is then passed on to the classifier and classified according to the nearest neighbors by the KNN algorithm.

  Full Source code on Appendix

i) ML Software Implementation

Since this project uses a relatively simple dataset, TinyML using classical machine learning algorithm k nearest neighbor (kNN) is implemented as compared to the more powerful deep learning frameworks like TFLite for Microcontrollers. This has the advantage of being lightweight for embedded devices, easy to understand and employs no off-device training nor additional tools.

Testing and Results

To test the completed sign language gesture recognition device developed, two users were requested to wear the glove and sign random numbers between 0 and 9. And repeat for 20 times to capture each class at least twice. The testing was split into two experiments. Experiment 1 and Experiment 2 that was necessitated by the relatively low confidence levels while classifying numbers 3, 4 and 5.

i) Experiment 1 whereby the KNN algorithm is provided with 5 examples of each class and parameter k = 5.

ii) Experiment 2 whereby the KNN algorithm was provided with 10 examples of each class while maintaining k = 5.

First, the classifier was provided with only five examples of each class, for example, we provided the algorithm with 5 samples of sensor data while signing the number “4”. The results from the first experiment.

Number signed	KNN Prediction	Confidence
4	4	0.8
7	7	1.0
3	3	0.6
9	9	1.0
5	5	0.8
5	5	0.6
1	1	1.0
6	6	1.0
2	2	0.8
8	8	1.0
0	0	1.0
7	7	0.8
5	4	0.6
2	2	1.0
6	6	1.0
3	4	0.6
1	1	0.8
0	0	1.0
4	5	0.3
9	9	1.0

The algorithm correctly classified 17 out of 20 gestures correctly albeit with lower confidence on numbers 4 and 5. That is an accuracy of 85%. This can be attributed to the low sensitivity on the flex sensor attached to the thumb therefore giving out almost similar sensor values.

To test whether providing more examples will increase the accuracy, we doubled the examples from 5 to 10 for each class and conducted the second experiment.

Classes 3, 4 and 5 have the lowest confidence of 0.6, which means 3 out of 5 examples or 60% of the falls within the nearest neighbor with 2 falling out of the range. Classes 0, 6, 8 and 9 have the highest confidence of 1.0.

During experiment 2, the KNN algorithm was provided with more examples. Increased from 5 to 10.

The signed numbers are shown in the table below.

Number Signed	KNN Prediction	Confidence
2	2	1.0
7	7	1.0
4	4	1.0
9	9	1.0
8	8	1.0
6	6	1.0
1	1	1.0
4	4	0.8
9	9	1.0
5	5	1.0
1	1	1.0
3	3	1.0
0	0	1.0
7	7	1.0
3	3	1.0
6	6	1.0
8	8	1.0
5	5	1.0
0	0	1.0
2	2	1.0

When the examples provided to the KNN was increased from 5 to 10 and the k=5 maintained, the accuracy increased from the 85% achieved on experiment 1 to 100% classifying accurately all the classes of the signed numbers. The confidence level also increased for all classes to 1.0 with only a single 0.8 for number 4.

4. CHAPTER FOUR: RESULTS AND DISCUSSION

4.1. Introduction

In this chapter we evaluate the outcome of the project as outlined in chapter one and how the outcome relates to the main objectives of the study, that is developing a sign language gesture recognition device using open-source resources that accurately identifies and classifies numbers signed in American Sign Language.

4.2. Achievements & Discussion

The project successfully achieves all the main objectives of the study. The sign language gesture recognition glove has been developed based on the Arduino open-source platform. Edge computing aspect of the device has been achieved by running the machine learning algorithm, kNN, on the resource constrained Arduino nano board. By providing 5 examples of each class of the numbers (0-9), confidence of up to 1.0 and accuracy of 100% have been achieved. By providing more examples (increased the examples to 10 from 5) increased the algorithms accuracy and confidence to 100% and 1 respectively for all number classes

From system testing, the sign language gesture classifier classifies all 8 & 9 categories of numbers correctly with an accuracy of 100% with highest confidence on signing number 9 and lowest confidence on number 4 (0.6) using 5 examples. Increasing the number of examples to 10 while retaining k at K=5, the accuracy increased to 100% and confidence 1 for all classes.

With low number of examples, despite the low confidence on signing numbers 3, 4 and 5 is due to almost similar position of the thumb in signing those numbers, the accuracy remains relatively high at over 85% for all the three classes. With a more sensitive sensors to track the fingers, the accuracy of the kNN algorithm would increase even with fewer examples, this is informed by the high confidence and accuracy on signing numbers 6 to 9 due to their distinctive fingers positioning.

4.3. Conclusions

Considering all the population living with hearing disability and the projected numbers rise by 2050 to about 1 person in 10, there is a need to use technology in order to address the communication challenge posed. Despite the progress made in addressing the use of technology for the communication challenge, the existing solutions are either too naïve and not scalable such as rule-based gloves or too complex and dependent on robust hardware resources like in computer vision approach. Therefore, more practical inexpensive solution that depend on machine learning on the edge, small in size and more power efficient is needed.

This study has successfully demonstrated one such solution using glove approach, affordable sensors and open-source hardware. Moreover, it has met the two main objectives of the study. That is to assess the accuracy of using Machine Learning Frameworks in specific TinyML ArduinoKNN are suited for sign language gesture recognition on embedded device and to develop a sign language gesture recognition device for numbers 0 to 9 based on on-device inferencing. In addition, the device achieves the properties considerations from the literature review. It is real-time, accurate, low-cost and portable.

4.4. Recommendations & Future Work

This study was solely focused on and limited to identifying and classifying numbers 0 to 9 due to the project time constraints. Due to this reason, only one machine learning algorithm was explored, the KNN algorithm. For more complex sign language gestures like phrases that require whole arm movement in conjunction with finger movement, a more robust machine learning algorithm like convolutional neural networks (CNN) would be required. For microcontrollers such as one used in this project, implementing a CNN algorithm is possible. Platforms such as TensorFlow are open-source and have algorithms customized to run on low resourced devices in form of TensorFlow Lite for Microcontrollers.

This glove can be extended to classify letters and phrases using the onboard IMU sensor which has a 3-axis gyroscope, an accelerometer and a magnetometer that would track the whole hand movement in addition to fingers. This would require much more data from multiple sensors and thus would require a more robust and scalable ML algorithm like TensorFlow Lite for Microcontrollers while maintaining the same hardware configuration.

References

Abdulla, D., Abdulla, S., & Manaf, R. (2016, December). Design and Implemetation of a sign-to-speec/text system for deaf and dumb people. *In Proceedings of the 2016 5th International Conference on Electronic Devices, Systems and Applications*, 1-4.

Ahmed, M. A., Z., B. B., Aws, A. Z., Mahmood, M., & Muammad, M. (2018). A Review on Systems-Based Sensory Gloves for Sign Language Recognition State of the Art between 2007 and 2017. *Sensors*.

Ahmed, S., S.M.B., & Qureshi, S. (2010, November). Electronic speaking glove for speechless patients, a tongue to a dumb. *In Proceedings of the 2010 IEEE Conference on Sustainable Utilization and Development in ENgineering and Technology (STUDENT)*, 56-60.

Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer networks*, pp.393-422.

Arif, A., Rizvi, S., Jawaid, I., Waleed, M., & Shakeel, M. (2016, April). Techno-Talk: An American Sign Language (ASL) Translator. *In Proceedings of the 2016 International Conference on Control, Decision and Information Technologies (CoDIT),*, 665-670.

Bajpai, D., Porov, U., Srivastav, G., & Sanchan, N. (2015, April). Two Way Wireless Data Communication and American Sign Language Translator Glove for Images Text and Speech Dispaly on Mobile Phone. *In Proceedings of the 2015 Fift Internatiunal Conference on Communication Systems and Network Technologies (CSNT)*, 578-585.

Bantupalli, K., & Xie, Y. (2019). *American Sign Langugae Recognition Using Machine Learning and Computer Vision.* Retrieved from Researchgate.

Brownlee, J. (2020, August 19). *Difference Between Algorithm and Model in Machine Learning.* Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/difference-between-algorithm-and-model-in-machine-learning/

Dabre, K., & Dholay, S. (2014). Machine Learning Model for Sign Language Interpretation using webcam images. *International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*, pp 317-321. doi:10.1109/CSCITA.2014.6839279

David J, S., & Zeltzer, D. (1994). A Survey of Glove-based Input. pp 30 - 39. doi:10.1109/38.250916

Deng, L., & Yu, D. (2014). Deep Learning: Methods and Applications. *Foundation and Trends in Signal Processing*, pp 199 - 200.

Dictionary, K. (2015). *KSL Dictionary*. Retrieved April 2021, from https://www.ksldictionary.com

Elmahgiubi, M., Ennajar, M., Drawin, N., & Elbuni, M. (2015, June). Sign Language translator and gesture recognition. *In Proceedings of the 2015 Global Summit on Computer & Information Technology (GSCIT)*, 1-6.

Emilio, M. D. (2014). *Embedded Systems Design for High Speed Data Acquisition and Control.* Springer. Retrieved June 2021 eTechnophiles. (2019). *etechnophiles*. Retrieved from Arduino Nano 33 BLE Sense Pinout, Introduction & Specifications: https://www.etechnophiles.com/arduino-nano-33-ble-sense-pinout-introduction-specifications/

Forsberg, K., & Mooz, H. (1991). The relationship of System Engineering to the Project Cycle. *In Proceedings of the First Annual Symposium of National Council on System Engineering*, 57-65.

Fu, Y., & Ho, C. (2008). Development of a programmable digital glove. *Smart Mater. Struct*, 526-533.

Gupta, D., Singh, P., Pandey, K., & Solanki, J. (2015, March). Design and development of a low-cost Electronic Hand Glove for deaf and blind. *In Proceedings of the 2015 2nd International Conference on Computing for Sustainable Global Development*, 1607 - 1611.

Iwasako, K., Soga, M., & Taki, H. (2014). Development of finger motion skill learning support system based on data gloves. *Procedia Comput. Sci*, 1307-1314.

Kartik, P., Sumanth, K., Ram, V., & Prakash, P. (2020). Sign Language to Text Conversion Using Deep Learning. *In book: Inventive communication and Computational Technologies, Proceedings of ICICCT,* pp 219 - 227.

Kramer, J., & Leifer, L. (1988, April). The Talking Glove. *ACM SIGCAPH Computers and the physically Handicapped*(39). doi:10.1145/47938

Kumar, D. (2019, April 21). *Software Engineering (SDLC) V-Model*. Retrieved October 30, 2020, from https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/

Lee, Y. T. (2018). Techology trend of edge AI. *International Symposium on VLSI Design, Automation and Test (VLSI-DAT),* (pp. 1-2). IEEE.

Microsoft. (2017). *Microsoft Soundscape*. Retrieved from Microsoft: https://www.microsoft.com/en-us/research/product/soundscape

Peckol, J. (2019). *Embedded systems: a contemporary design tool.* John Wiley & Sons.

Plack, C. J. (2014). The Sense of Hearing. Hove: Psychology Press Limited.

Praveen, N., Karant, N., & Mega, M. (2014, October). Sign Language interpreter using a smart glove. *In Proceedings of the 2014 International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, 1 - 5.

Presser, M. (2016). The rise of IoT–why today.

Ralph, P. a. (2009). A proposal for a formal definition of the design concept. *Design Requirements Workshop (LNBIP 14)*, pp 103 - 136.

Rosencrance, L. (2019). *Tech Target*. Retrieved from Software: https://searchapparchitecture.techtarget.com/definition/software

Saggio, & Giovanni. (2014). A novel array of flex sensors for a goniometric glove. *Sensors and Actuators A: Pysical*, pp 119 - 125.

Saltuk, O., & Kosan, I. (2014, May 7). Design and Creation - Motivation. *Design and Creation*, p. 4.

Sandler, W. &.-M. (2006). *Sign Language and Linguistics Universals.* Cambridge: Cambridge University Press.

SAS. (2021). *SAS Insights*. Retrieved from Analytics Insights: https://www.sas.com/en_us/insights/analytics/computer-vision.html

Sharma, D., Verma, D., & Khetarpal, P. (2015, December). Labview based sign language trainer cum portable display uit for the speech impaired. *In Proceedings of the 2015 Annual IEEE India Conference (INDICON)*, 1-6.

Shigeki, S. (2019). *Human Behavior and Another Kind in Conciousness:Emerging Research and Opportunities.* IGI Global.

Shishir G. Patil, D. K. (2019). GesturePod: Enabling On-device Gesture-based Interaction for White Cane Users. *In Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19).*, pp 403 - 415. doi:https://doi.org/10.1145/3332165.3347881

Sreejan, A., & Yeole, S. (2017). A Review on Applications of Flex Sensors. *International Journal of Emerging Technology and Advanced Engineering*, pp 97 - 100.

Tanyawiwat, N., & Thiemjarus, S. (2012, May). Design of an assistive communication glove using combined sensory channels. *In Proceedings of the 2012 9th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, 34-39.

TensorFlow. (2021). *TensorFlow*. Retrieved from An end-to-end open source machine learning platform: https://www.tensorflow.org/

Trottier-Lapointe, W., Majeau, L., El-Iraki, Y., Loranger, S., Chabot-Nobert, G., Bordaus, J., . . . Lapointe, J. (2012, July). Signal Procesing for low cost optical data glove. *In Proceedings of the 2012 11th International Conference of Information Science, Signal Processing and teir Applications (ISSPA)*, 501-504.

Vijay, P., Suhas, N., Chandrashekhar, C., & Dhananjay, D. (2012). Recent Developments in sign language recognition: A review. *International Journal of Advance Computing, Engineering, Communication Technology*, 21-26.

Vijayalakshmi, P., & Aarthi, M. (2016, April). Sign Language to speech conversion. *In Proceedings of the 2016 International Conference on Recent Trends in Information Technology*, 1-6.

Vutinuntakasame, S., Jaijongrak, V., & Thiemjarus, S. (2011, May). An assistive body sensor network glove for speech-and-hearing-impaired disabilities. *In Proceedings of the 2011 International Conference on Body Sensor Networks (BSN)*, 7-12.

World Health Organisation. (2021, 04 01). *Deafness and Hearing Loss.* Retrieved from World Health Organization: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss

Zhou, Z., Chen, K., Li, X., Zhang, S., Wu, Y., Zhou, Y., . . . Chen, J. (2020, September). Sign-to-Speech translation using machine-learning-assisted stretchable sensor arrays. *Nature Electronics*. doi:10.1038/s41928-020-0428-6