République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي

Djilali Bounaama Khemis Miliana University
جامعة الجلالي بونعامه خميس مليانة

Faculty of Science and Technology Department of Mathematics and Computer Science

Thesis Presented
for obtaining a Master's degree
of Computer Science

Option: Software Engineering and Distributed Systems

Toward training a micro model for embedded Arabic Sign Language Recognizing system (Micro ML-ARSLR)

Realized by:
Miss. Aicha Kaouther
CHERCHALI.

In front of the jury members:

Mr.Nouredinne AZZOUZA:	President
Mr.Djamel BAHLOUL :	Examiner
Mr.Abdelhamid HARICHE:	Supervisor

Academic Year: 2022 / 2023

Dedication

“

I dedicate my dissertation work to the most beloved people in my life. Firstly, I would like to dedicate this work to my grandmother, who sadly left us earlier. I deeply wish she could be here with me, witnessing my growth and the realization of my dreams. May she find eternal peace and happiness in Paradise.

I also extend my heartfelt dedication to my family, who has been a constant source of love and support. To my loving parents, I am forever grateful for their words of encouragement and unwavering belief in my tenacity. Without their help, both morally and financially, pursuing this study would not have been possible. To my sister, Khadidja, who has been my rock, standing by my side and supporting me in every step of this project.

I extend my dedication to my two brothers,Mohamed and Abdelkader whose love, support, and encouragement have been invaluable throughout this journey. To the delightful children of my family, Amira, Aicha, Omar, and Slimane, thank you for bringing joy to my life and inspiring me to work hard.

Lastly, I would like to express my gratitude to my friends, who have been with me through thick and thin. Your presence in my life has made this journey more meaningful and memorable.

To all the people mentioned above, your love, guidance, and support have played an integral role in my academic achievements. This dedication is a testament to the profound impact you have had on my life. Thank you for being my pillars of strength and for believing in me. I dedicate this work to each and every one of you with heartfelt appreciation and love.

”

- Aicha

Acknowledgment

First and foremost, I would like to thank Almighty Allah for granting me the courage and patience to complete this work.

I would like to express my heartfelt gratitude to **Mr Abdelhamid HARICHE**, my teacher and thesis supervisor, for his immense help, quality guidance, and invaluable advice. His unparalleled patience and professionalism have been truly remarkable.

To the members of the jury, I express my deep gratitude for the honor they have bestowed upon me by taking the time to read and evaluate this work.

I would also like to thank the teaching and administrative staff of the Faculty of Science and Technology for their efforts in providing us with an excellent education.

Lastly, I would like to express my gratitude to everyone who has contributed directly or indirectly to the completion of this work.

Abstract

There are over 360 million people globally living with hearing disabilities, relying on sign language for communication. Translating sign language poses significant challenges, requiring data on hand and finger movements that are often unavailable, and existing approaches, such as computer vision, are resource-intensive. Embedded systems offer a potential solution, with recent advancements enabling machine learning algorithms to run on resource-constrained devices. This study develops an Embedded Intelligent System that utilizes sensors to track finger curvatures and hand movements, employing a neural network algorithm for sign language translation.

The research starts with the development of an open-source hardware device integrated with sensors to capture data related to different sign language gestures. A dataset is then compiled by performing these gestures and carefully organizing the acquired data. This dataset is utilized to train a neural network model, enabling on-device inference for real-time recognition of sign language gestures.

The Arduino Nano BLE Sense, combined with flex sensors, is utilized for data collection. As the flex sensor bends during the signing of different letters and numbers, the corresponding data is logged. This data is then used to train the neural network model, enabling the translation and display of sign language sentences on the serial monitor.

The main contribution of this paper is a novel machine learning on the edge approach that leverages open-source resources to create a sign language translation device. This approach enables the translation to be performed on resource-constrained devices without relying on external inference engines. The developed system showcases the potential of embedded systems and provides a foundation for further advancements in sign language translation technology.

Keywords : Sign language translation, embedded systems, machine learning on the edge, open-source hardware, Neural network algorithm, Arduino Nano BLE Sense, flex sensors .

Résumé

Il y a plus de 360 millions de personnes dans le monde vivant avec des déficiences auditives, qui comptent sur la langue des signes pour communiquer. La traduction de la langue des signes pose d'importants défis, nécessitant des données sur les mouvements des mains et des doigts qui sont souvent indisponibles. Les approches existantes, telles que la vision par ordinateur, sont gourmandes en ressources. Les systèmes embarqués offrent une solution potentielle, grâce aux récents progrès permettant l'exécution d'algorithmes d'apprentissage automatique sur des dispositifs aux ressources limitées. Cette étude développe un système embarqué intelligent qui utilise des capteurs pour suivre les courbures des doigts et les mouvements des mains, en utilisant un algorithme de réseau neuronal pour la traduction de la langue des signes.

La recherche commence par la conception et la construction d'un dispositif matériel open-source équipé de capteurs pour collecter des données sur divers gestes de langage des signes. Un ensemble de données est créé en effectuant ces gestes et en organisant soigneusement les données collectées. Cet ensemble de données sert d'exemples d'entraînement pour un modèle de réseau neuronal, facilitant l'inférence embarquée pour la reconnaissance en temps réel des gestes de langage des signes.

L'Arduino Nano BLE Sense, combiné à des capteurs de flexion, est utilisé pour la collecte de données. Lorsque le capteur de flexion se plie pendant la réalisation de différentes lettres et chiffres, les données correspondantes sont enregistrées. Ces données sont ensuite utilisées pour entraîner le modèle de réseau neuronal, permettant la traduction et l'affichage des phrases de langue des signes générées sur le moniteur série.

La principale contribution de cet article est une approche novatrice d'apprentissage automatique embarqué qui utilise des ressources open-source pour créer un dispositif de traduction de la langue des signes. Cette approche permet d'effectuer la traduction sur des dispositifs aux ressources limitées, sans dépendre de moteurs d'inférence externes. Le système développé met en évidence le potentiel des systèmes embarqués et jette les bases de nouvelles avancées dans la technologie de traduction de la langue des signes."

Mots clés : Traduction de la langue des signes, systèmes embarqués, apprentissage automatique intégré, matériel open-source, algorithme de réseau neuronal, Arduino Nano BLE Sense, capteurs de flexion.

ملخص

هناك أكثر من 360 مليون شخصا في جميع أنحاء العالم يعيشون مع إعاقات سمعية، يعتمدون على لغة الإشارة للتواصل. تعتبر ترجمة لغة الإشارة تحديا كبيرا، حيث يتطلب الأمر بيانات عن حركات اليد والأصابع التي غالبا ما تكون غير متاحة، فالنهج الموجودة، مثل الرؤية الحاسوبية، تستهلك موارد كبيرة. تقدم الأنظمة المدمجة الحل المحتمل، مع التطورات الأخيرة التي تمكن من تشغيل خوارزميات التعلم الآلي على أجهزة محدودة الموارد. يهدف هذا البحث إلى تطوير نظام مدمج ذكي يستخدم الحساسات لتتبع انحناءات الأصابع وحركات اليد، بواسطة استخدام خوارزمية شبكة عصبية لترجمة لغة الإشارة.

يبدأ البحث بتصميم وبناء جهاز مفتوح المصدر مزود بحساسات لجمع البيانات حول مختلف حركات لغة الإشارة. يتم إنشاء مجموعة بيانات عن طريق تنفيذ هذه الحركات وتنظيم البيانات المجمعة بعناية. تعمل هذه المجموعة كأمثلة تدريب لنموذج شبكة عصبية، مما يسهل التخمين على الجهاز للتعرف في الوقت الحقيقي على حركات لغة الإشارة.

يتم استخدام جهاز BLE Nano Arduino، Sense بالاشتراك مع حساسات الانحناء، لجمع البيانات. عند انحناء حساس الانحناء أثناء القبام بالإشارات المختلفة، يتم تسجيل البيانات المقابلة. تُستخدم هذه البيانات لتدريب نموذج الشبكة العصبية، مما يمكن من ترجمة وعرض التنبؤات الموقعة على شاشة العرض المدمجة.

المساهمة الرئيسية لهذه الدراسة هي نهج جديد لتعلم الآلة على الشريحة باستخدام موارد مفتوحة المصدر لإنشاء جهاز ترجمة لغة الإشارة. يتيح هذا النهج تنفيذ الترجمة على أجهزة محدودة الموارد دون الاعتماد على محركات الاستدلال الخارجية. يعرض النظام المطور إمكانات الأنظمة المدمجة ويوفر أساسًا لمزيد من التقدم في تكنولوجيا ترجمة لغة الإشارة."

كلمات مفتاحية : ترجمة لغة الإشارة، الأنظمة المدمجة، التعلم الآلي على الشرائح، الأجهزة مفتوحة المصدر، الأجهزة ذات خوارزمية الشبكة العصبية ، BLE Nano Arduino، Sense مستشعرات الانحناء، التصنيف في الوقت الحقيقي.

Table of Contents

List of Figures

List of Tables

List of abbreviations and acronyms

AI	*Artificial Intelligence*
ANN	*Artificial Neural Network*
ASL	*Arabic Sign Language*
ASLR	*Arabic Sign Language Recognition*
ASLR-SP	*Arabic Sign Language Recognition using Signal Processing*
BLE	*Bluetooth Low Energy*
CV	*Computer Vision*
DIP	*Digital Image Processing*
DSP	*Digital Signal Processing*
EIS	*Embedded Intelligent System*
HMM	*Hidden Markov Model*
IDE	*Integrated Development Environment*
IMU	*Inertial Measurement Unit*
LCD	*Liquid Crystal Display*
ML	*Machine Learning*
PCA	*Principal Component Analysis*
SVM	*Support Vector Machine*
SLR	*Sign Language Recognition*

SLRAS *Sign Language Recognition in Arabic Sign Language*

SP *Signal Processing*

SR *Sign Recognition*

TFLite *TensorFlow Lite*

W.H.O. *World Health Organization*

Introduction

Hearing loss, ranging from mild to profound, affects a significant portion of the global population. Individuals with profound hearing loss rely on sign language as their primary means of communication. However, unaddressed hearing loss can lead to loneliness, social isolation, and economic burdens, making it essential to develop effective assistive technologies for sign language recognition.

The challenges of sign language recognition can be addressed with recent advancements in technology, particularly in machine learning and embedded systems. Developing a cost-effective and efficient sign language recognition system for resource-constrained embedded devices could significantly enhance communication capabilities for individuals with hearing disabilities.

The primary objectives of this study are :

1. To develop a sign language recognition system using the Arduino Nano 33 BLE Sense and TensorFlow Lite for Microcontrollers, incorporating flex sensors and an Inertia Measurement Unit (IMU) to capture finger and hand movements.

2. To train machine learning models using collected data for real-time inference on the device itself, eliminating the need for external inference engines.

3. To contribute to the field of assistive technology by providing an affordable and effective sign language recognition solution for resource-constrained embedded systems.

Humans are social beings, and communication is a fundamental human need. For individuals with hearing difficulties or deafness, effective communication can be challenging. Sign Language was developed to enable communication through hand gestures, providing an essential means of expression for the Deaf community.

The projected increase in the number of deaf individuals, with an estimated 1 in every 10 people being deaf by 2050, highlights the urgency of developing a method to interpret sign language for those who do not understand it. Additionally, the majority of the projected deaf population is expected to come from low- and middle-income countries, with a significant proportion being 60 years or older (World Health Organization, 2021). Hence, there is a crucial need to develop affordable and efficient assistive technology to bridge the communication gap.

By specifically focusing on sign language gestures representing Arabic sentences, this study addresses a significant need for effective communication within the Deaf community. The outcomes will benefit developers interested in creating Assistive Technology applications for the Deaf community, and it will lay the groundwork for future research on translating other sign language gestures, including more complex two-handed signing.

The study's emphasis on implementing machine learning algorithms on edge devices is vital, as these devices often have limited resources, including tight time constraints, low power, and limited memory. However, advancements in technologies like TinyML and TensorFlow Lite for Microcontrollers enable the incorporation of machine learning capabilities into these low-resource devices, making it feasible to perform gesture recognition on tiny embedded devices without requiring extensive hardware or fast network connections. This approach ensures data security and privacy, as all processing takes place on the device itself, without the need for data to leave the device for external processing.

This thesis is organized into four chapters :

In this chapter **State of the art** , we emphasize the importance of sign language in communication and inclusion, particularly focusing on the challenges related to Arabic sign language recognition. We outline the research objectives and key concepts that will guide our study.

This chapter **Signal Processing and Machine Learning with Data Glove** delves into the methodology for recognizing Arabic Sign Language. We discuss signal processing techniques, data glove-based approaches, data collection, preprocessing steps, feature extraction, machine learning algorithms, considerations for dataset creation, and the process of model development, training, and evaluation metrics.

Here, in this chapter **Research Design and Methodology** we present the design of the sign language recognition system, covering aspects related to research, system, and architectural considerations. We discuss the necessary hardware and software requirements for implementing the system effectively.

In this chapter **ARSLR System Design and Implementation** , we detail the design of the Arabic Sign Language Recognition (ARSLR) system, providing both conceptual and logical architecture explanations. We also present the implementation details, testing procedures, and results obtained from our experiments. The chapter concludes with key findings and insights derived from the system's performance and overall effectiveness.

Chapter 1

Arabic sign language recognition

Introduction

Sign language is a crucial tool for deaf people, and its recognition has been explored through various approaches, such as rule-based, vision-based, data-driven, and deep learning techniques.

This chapter emphasizes the significance of sign language recognition and serves as a foundational basis for the subsequent chapters focused on Arabic sign language recognition.

1.1 Sign language and its importance

Since the sign language through the time put a big question to its inner and outsider community, this section will show this general impact as follow :

1.1.1 Definition and characteristics of sign language

Sign language is a visual-spatial language system used by deaf and hard-of-hearing individuals to communicate and express themselves. It is a natural language that relies on manual gestures, facial expressions, body movements, and spatial relationships to convey meaning.[49]

- **Visual-Gestural Modality :** Sign language uses visual-gestural communication, including hand shapes, movements, facial expressions, and body postures.
- **Grammar and Structure :** Sign languages have grammatical rules, structures, and phonological systems, including word order, sentence structure, negation, and agreement patterns.
- **Regional Variation :** Sign languages exhibit regional variation, developing within specific deaf communities or cultural groups in different countries or regions, like American Sign Language (ASL).
- **Linguistic Complexity :** Sign languages possess linguistic complexity, enabling nuanced expression and richness comparable to spoken languages, expressing emotions and cultural nuances.
- **Cultural Identity :** Sign languages are crucial for deaf communities' cultural identity, fostering cohesion and heritage transmission through expression, storytelling, and transmission.
- **Accessibility and Inclusion :** Sign languages promote equal access to information, education, employment, and social interactions for deaf individuals in society.

Sign languages are distinct, fully developed linguistic structures with unique history and cultural significance.

1.1.2 Brief History of sign language

There is much debate as to when people first communicated via sign language with some scientists hypothesizing that early humans probably used a basic form to communicate with one another before the advent of speech.

Throughout history, sign language has played a significant role in communication. Its origins can be traced back to ancient times, with early examples found in ancient Greece and Native American tribes in North America.

However, it wasn't until the 1500s that a shift in attitudes occurred, recognizing that deaf individuals could learn and communicate through sign language. This led to the development of educational techniques and the establishment of schools for the deaf, such as **Thomas Braidwood**'s Academy in Britain. Sign language continues to be an essential means of communication for the deaf community.[43]

Fig. 1.1 : History of sign language

1.1.3 Importance of sign language for communication and inclusion

Sign language is vital for communication and inclusion among individuals who are deaf or hard of hearing. It enables interaction within the deaf community and facilitates social inclusion.[29]

Sign language also offers deaf children a means of education, contributing to their self-esteem and overall well-being.

Learning sign language promotes social responsibility and sensitivity in non-deaf individuals. It enhances everyday communication, tasks, and accessibility for deaf individuals.

Additionally, sign language has developmental advantages for deaf children, aiding in language acquisition and cognitive abilities. It is also beneficial for autistic children who struggle with verbal expression.

1.2 Fundamentals of Sign Language Recognition

In this section we explore Sign Language Recognition's fundamentals, techniques, and methodologies for accurately recognizing gestures and movements in sign languages.

1.2.1 Sign Language Recognition

Sign Languages Recognition (SLR) interprets gestures and movements in text or speech using computer vision and machine learning techniques.

Sign language is a full language with unique vocabulary and grammar, using hand gestures, finger configurations, facial expressions, and body traces.

Effective translation requires considering all components of the language. Sign language varies across countries, with localized dialects in Arab countries like Saudi, Yemeni, Jordanian, and Egyptian.

The attentions of most researchers has been shifted to ASLR research, as it is now applied in many domains like machine control in the industrial domain [16, 81] , communication system for deaf and hearing-impaired people [82, 17] , Human-Computer Interaction (HCI) [9], Virtual Reality (VR) [13, 83], and many others. ASLR research is divided into two main groups, namely ;

1) Static sign language.

2) Dynamic sign language [65, 42], as shown in 1.2

The position of the hand and its orientation are usually used to correct the static gestures **which consists of hand poses and postures** and dynamic gestures **which includes hand movements with a certain type like waving** in a given space and time without making any kind of movement.

Static gestures involve single hand orientation without movement[75]. While dynamic gestures use continuous video frames as input.

Moreover,the static gestures depend on finger shape and angular direction, while dynamic gestures involve hand position.

On the order hand,dynamic hand gestures involve continuous motion, with messages based on stroke phase sequence, divided into stroke phase, retraction, and preparation phases[62].

According to the study carried out in [42], static gestures based on orientations, shape, flex angles, position, and context, while dynamic gestures are characterized by orientations, shape, and scale.

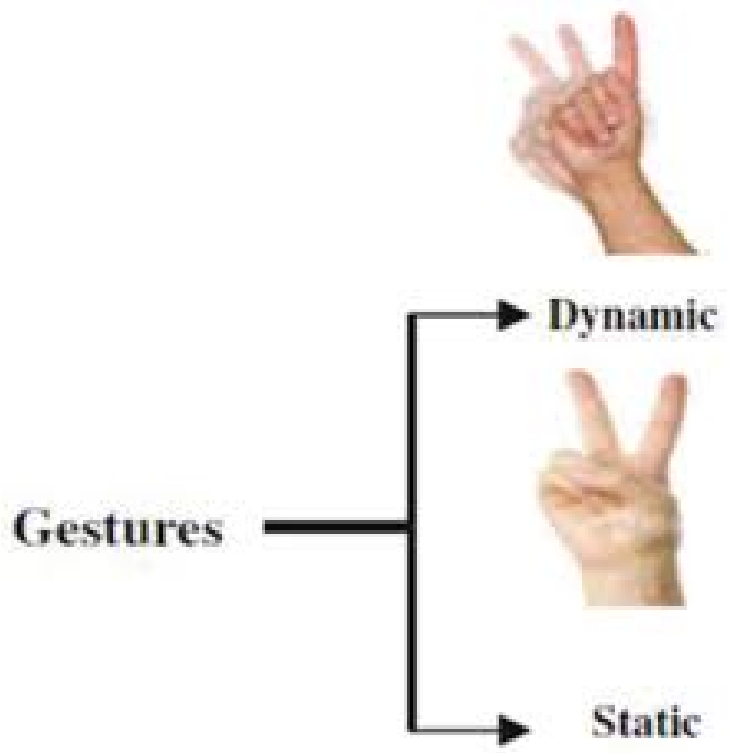


Fig. 1.2 : Example for static and dynamic gesture.

1.2.2 Challenges in Sign Language Recognition

Sign language recognition is a complex task that involves several challenges[34]. Here are some of the key challenges in sign language recognition :

The variation in sign language across countries and regions, as well as individual signing styles.

Limited data availability hinders effective model training due to the small number of signers and difficulties in recording and annotating videos.

Intra-class variability arises when signs are performed differently by different individuals, making accurate recognition challenging.

Occlusion, where body parts obstruct each other, further complicates tracking movements.

Real-time processing is necessary for sign language interpretation, requiring fast and accurate recognition systems.

Data imbalance, with certain signs being more common than others, can introduce biases in machine learning models.

Incorporating non-manual features like facial expressions and body posture into the recognition system is also a challenging aspect.

1. Levels of Sign Language Recognition

Sign language recognition can be categorized into three levels : word-level, sentence-level, and discourse-level recognition.

Word-level recognition focuses on identifying individual signs and mapping them to corresponding words or phrases in spoken language.

Sentence-level recognition aims to understand the meaning of complete sign language sentences, considering context and grammar.

Discourse-level recognition involves interpreting the larger conversation context, including speaker intentions and nonverbal cues.

Accurate sign language recognition systems are crucial for effective communication between hearing and deaf individuals, with advancements in machine learning and computer vision contributing to improving system performance and bridging the communication gap.[1]

2. Architecture of Automatic Sign Language Recognition

ASLR involves four primary phases to accurately recognize gestures : data acquisition, pre-processing and segmentation, feature extraction, and classification.

These phases are illustrated in 1.3 An additional example of the processing model or cycle model of SLR and ArSLR can be found in [2].

The hand image is segmented to determine its position within the body, followed by processing to remove noise and create a model.

Features are extracted, including shape position, orientation, movements, and hand location, for classification purposes.

The captured images are identified as appropriate gestures through analysis and modeling [5, 73] .

SLR operates at three levels : alphabets, isolated words, and continuous sentences.[2]

Please see Figure 1.3 and the relevant section for a more thorough explanation of the ASLR Architectural stages.

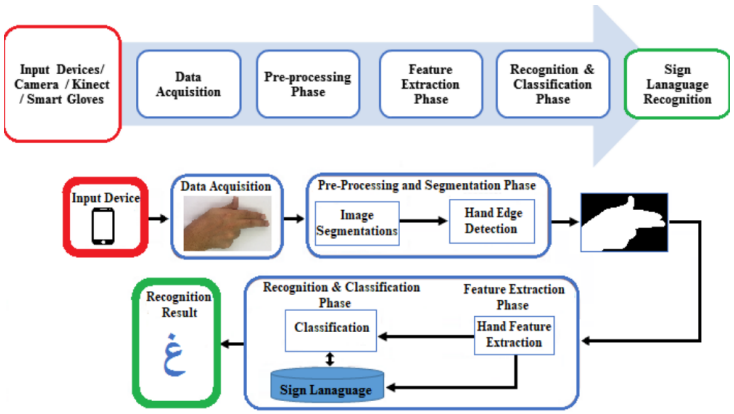


FIG. 1.3 : Architectural design of the Automatic Sign Language Recognition.

1.2.3 Data Acquisition Phase

Hand gesture recognition relies on accurate data acquisition ([76, 18]), using frames from various cameras like stereo, webcam, thermal, or video cameras. 3D cameras, like Kinect, can capture depth information [14].

Selecting the right input device is crucial for successful data acquisition, with options including hand images, Kinect 3D sensor, markers, data gloves, stereo cameras, and webcams.

Arabic sign language recognition uses vision-based and sensor-based approaches.[73, 52, 12].

In the context of Arabic sign language recognition, data can be acquired through two different approaches : vision-based and sensor-based.

A. Vision-based approach

The vision-based approach uses video cameras to capture hand gesture images, using techniques [14] like invasive techniques, active techniques, single cameras, and stereo cameras.

These technologies enable the analysis and recognition of hand gestures, providing richer depth perception and visual data. The 3D hand model approach is another method used for capturing hand gestures.

These technologies are employed within the vision-based approach to capture hand gestures for further analysis and recognition.

B. Sensor-based approach

The sensor-based approach captures hand gestures using various sensors and instruments to gather information about hand location, motion, and velocity[14].

Technologies include inertial measurement units (IMU) [67], Wi-Fi and Radar, electromyography (EMG), haptic technologies, mechanical sensors, electromagnetic sensors, ultrasonic sensors, and flex sensors.

These technologies enable precise information about hand movement and gestures, enabling accurate recognition and analysis.

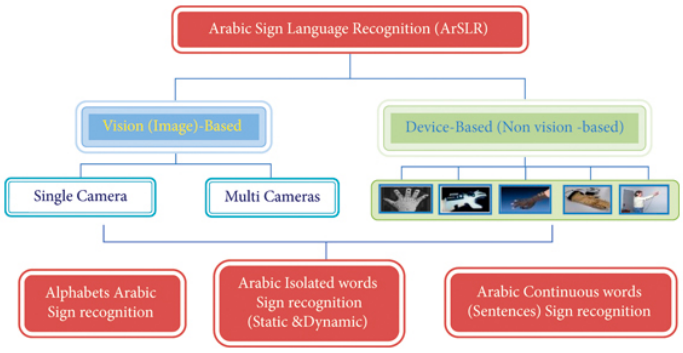


Fig. 1.4 : Approaches of Arabic sign language recognition.

1.2.4 Pre-processing and Segmentation phase (Gesture Modeling Phase)

Successful gesture recognition requires proper modeling of device data through preprocessing, segmentation, noise removal, contour detection, and normalization[86]. Image pre-processing in gesture recognition involves filters, morphological operations, pixel representation reduction, and histogram equalization to enhance performance, resolution, and brightness in images or videos [14].

Segmentation in gesture recognition involves partitioning images into distinct forms, separating regions of interest from background.

There are two types : contextual segmentation, considering geometric relationships, and non-contextual segmentation, with hand segmentation crucial in vision-based ASLR.

1.2.5 Feature Extraction Phase

Feature extraction in gesture recognition relies on segmentation [6, 44], providing essential elements for identifying and categorizing hand gestures, distinguishing them from other body parts.

Hand gesture recognition uses various features like shape, distance, motion, contour, textures, center of gravity, orientation, and velocity. Geometric features like finger detection and fingertips are commonly used, but illumination variations and occlusions affect their reliability [73, 55].

Sign language extraction techniques include SIFT, PCA, SURF, LDA.

1.2.6 Classification Phase

The final phase of recognition involves hand gesture classification, requiring effective techniques and algorithms [73, 55].

Rule-based and machine learning-based approaches are crucial for accurately recognizing and categorizing gestures in pattern recognition and machine learning.

A. Rule-based Approach

This approach develops manually encoded rules between feature inputs, extracting features from gestures and comparing them with encoded rules, identifying matched rules as corresponding gestures [55].

B. Machine Learning-based Approach

Researchers use machine learning techniques to capture mappings between gestures and high-dimensional feature sets, treating gestures as stochastic processes and employing classification algorithms[75] .

These algorithms include Conditional random fields (CRF) [75, 47] , K-means [75, 40] , K-nearest neighbor (K-NN)[75, 79] , Mean-shift clustering model [75, 20], support vector machine (SVM) [75, 10] , Hidden Markov Models (HMMs) [75, 64] , Dynamic Time Warping (DTW) [75, 74] , Time-delay neural network (TDNN)[75, 85] , Finite-State Machine (FSM)[75, 31], Artificial Neural Networks (ANNs) [75], as well as Gaussian mixture distribution and Euclidean distance measure [55, 11, 15]]. For a more comprehensive understanding of each algorithm and detailed discussions and comparisons, refer to [75].

1.2.7 Corpus of Main World Standard Sign Languages

Over 5% of the global population has deaf-mute and hearing disabilities [58], requiring hand, head, and body gestures for expression [76].

A comprehensive corpus of signs is crucial for designing and developing Automatic Sign Language Recognition (ASLR) systems. Table 1.2 provides an overview of the main standard sign language corpora available in different countries.

Tab. 1.2 : Sign Language Databases

Language	Abbrev	SIL code	Library/Db Name	Description
American Sign Language	ASL	Purdue RVL-SLLL	Database	Comprehensive database of ASL gestures, movements, words, and sentences

Continued on next page

Table 1.2 continued from previous page

Language	Abbrev	SIL code	Library/Db Name	Description
Australian Sign Language	Auslan	Auslan Sign bank	Auslan-Signbank	Corpus comprises 7415 words in Auslan, with over 1000 video clips performed by one hundred signers
German Sign Language	DGS	The SIGNUM Database	The SIGNUM Database	Database containing four hundred fifty basic gestures and seven hundred eighty sentences performed by twenty-five signers
Arabic Sign Language	ArSL	Arabic Sign Language Database	Arabic Sign Language Database	Corpus contains one example sentence performed by one signer. It is fully segmented and labeled for continuous ArSL

1.3 Difficulties in Arabic Sign Language Recognition

Arabic Sign Language Recognition (ASLR) poses several unique challenges due to the specific characteristics of sign languages and the Arabic language. Some of the difficulties in ASLR include [33] :

Variation and Dialects : Arabic Sign Language has regional variations and dialects across different countries in the Arab world. Each dialect may have variations in signs, handshapes, and gestures, making it challenging to create a unified recognition system that can handle all these variations accurately.

Vocabulary Size : Sign languages typically have a large vocabulary with a considerable number of signs. Building a comprehensive database of signs and their variations is time-consuming and resource-intensive.

Data Sparsity : Collecting large amounts of sign language data is often difficult due to the relatively small number of signers compared to spoken languages. This data sparsity can affect the performance of machine learning models used for recognition.

Sign Motion Dynamics : The dynamic nature of signs, including hand movements, facial expressions, and body postures, adds complexity to recognition algorithms. Capturing and interpreting these dynamic features accurately requires sophisticated computer vision and machine learning techniques.

Non-Manual Signals : Sign languages use non-manual signals, such as facial expressions and body movements, to convey meaning and grammatical information. Integrating non-manual signals into recognition systems adds complexity to the recognition process.

Occlusion and Background Noise : In real-world scenarios, occlusions (e.g., hands overlapping) and background noise can interfere with the accuracy of sign language recognition, making it challenging to correctly interpret signs.

Real-time Processing : For practical applications like sign language interpretation systems, real-time processing is essential. Achieving low-latency recognition while maintai-

ning accuracy is a significant challenge.

User Dependency : Sign language recognition systems often perform better when trained and fine-tuned for specific users or a small group of signers, limiting their generalizability across the entire signer population.

1.4 Arabic Sign Language Recognition Approaches

Recent research on Arabic sign language recognition (ArSLR) has grown, employing various methods, frameworks, and techniques, categorized into two main categories [33] .

1) Vision-Based Recognition (VBR)

2) Sensor-Based Recognition (SBR)

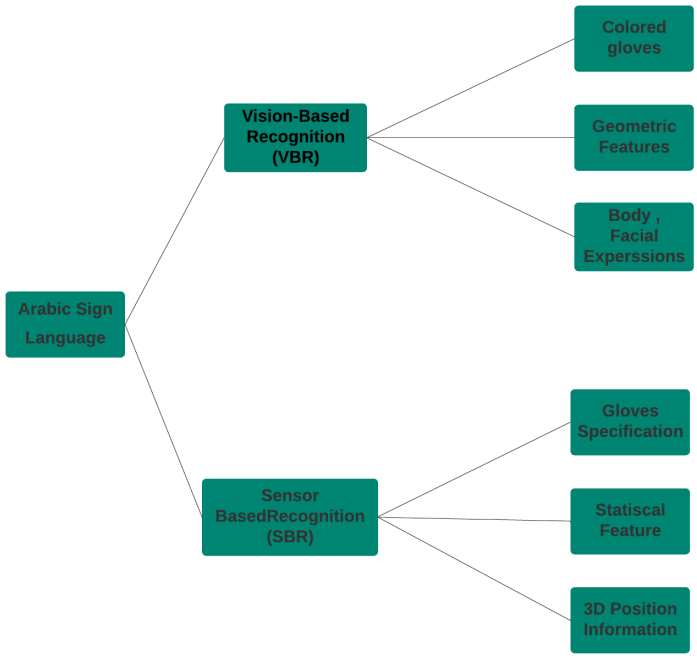


Fig. 1.5 : Taxonomy of Arabic sign language recognition approaches

The illustration is shown in 1.6 and 1.3 provides a comparison of the vision-based and sensor-based approaches based on the design concepts.

Vision-based approach uses images or videos of signers, eliminating the need for heavy equipment like DataGloves. However, challenges like background interference, segmentation, and lighting conditions need to be addressed.

Sensor-based approach uses Smart Gloves' sensors for sign recognition, extracting features for classification algorithms.

Both approaches face challenges but hold potential for further development.

Tab. 1.3 : Comparison of the vision-based and sensor-based approaches based on the design concepts

concept	Sensor-based approach	Vision-based approach
World wide availability	Low likely	High likely
Feature extraction	Relatively easier	Challenging
User experience	Inconvenient	Good
Cost	High	Low
Calibration	Required but stable	Environment Dependen
User dependency	Less prone	Highly prone

Fig. 1.6 : Several types of smart gloves : Power Glove (left), DT-Data Glove (Center), Cyber Glove (right)

Arabic sign language recognition research can be categorized into alphabet, isolated words, and continuous areas.

This classification helps understand the progression and complexity of ArSLR research, providing insights into its development.

1.4.1 Alphabet Sign Language Recognition

Research on Arabic alphabet sign language recognition using vision-based approaches focuses on 39 signs, representing 28 alphabets.

No specific research exists on sensor-based recognition for Arabic alphabet recognition[33].

1.4.2 Isolated Words Sign Language Recognition

The literature investigation reveals numerous research efforts in Arabic isolated words sign language recognition, using vision-based and sensor-based approaches.

Vision-based recognition analyzes isolated words individually, while sensor-based recognition uses data from smart gloves or sensors[4].

1.4.3 Continuous Sign Language Recognition

Limited research on Arabic continuous sign language recognition uses vision-based and sensor-based approaches. This gap presents opportunities for further exploration. Continuous ASLR systems are practical for impaired hearing and Deaf communities, but

challenges include detecting transitional movements, accurate recognition, and effective modeling techniques. Further research is needed to advance the field[80].

1.5 Open issues and challenges for Arabic Sign Language Recognition

A taxonomy of this review shown in 1.7 is developed for open issues and challenges inherent in ArSLR research.

The issues and the challenges according to the taxonomy are classified based on Language, System, Environment, and Gesture which are further described in the subsection below.

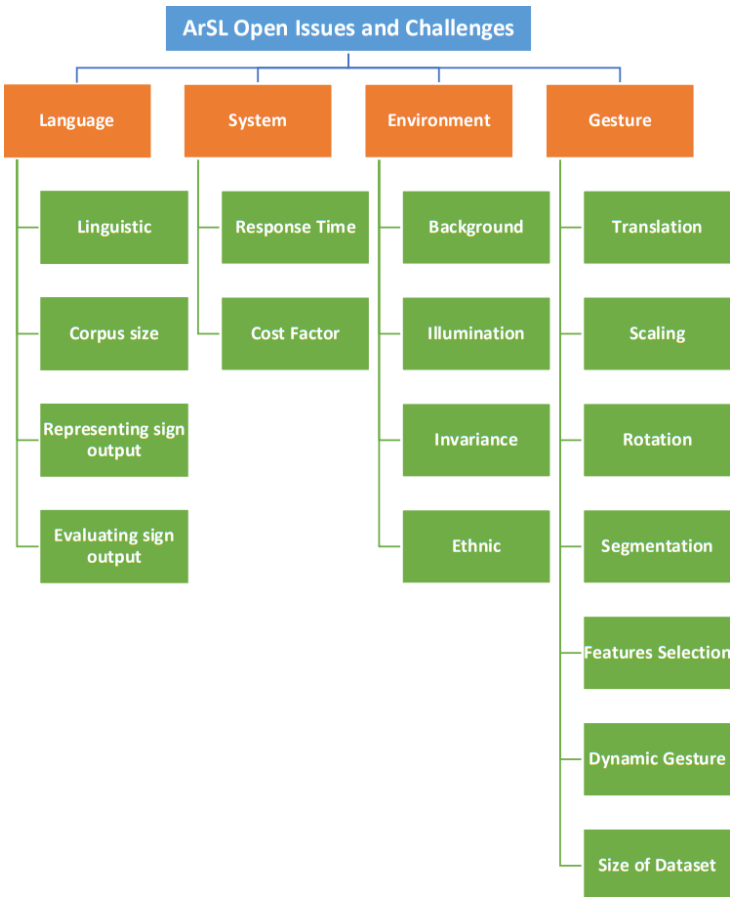


FIG. 1.7 : classification of Arabic ASLR

Conclusion

The chapter emphasizes the significance of sign language recognition for individuals with hearing disabilities, highlighting advancements in machine learning and artificial intelligence. It discusses challenges and approaches, highlighting the potential of innovative

methodologies and technologies to bridge communication gaps and empower individuals with hearing disabilities.

Chapter 2

Signal Processing and Machine Learning with Data Glove

Introduction

This chapter discusses the significance of signal processing, feature extraction, and machine learning in Arabic sign language recognition using data gloves.

Signal processing extracts information from sensor data, such as finger flexion, hand orientation, and motion, while feature extraction identifies key patterns and characteristics of gestures.

Machine learning algorithms train models to classify and recognize Arabic sign language gestures based on extracted features. Combining these techniques leads to accurate and meaningful recognition of Arabic sign language, improving communication and inclusivity for the deaf and hard-of-hearing community.

2.1 Introduction to SP in ARSLR

In recent years, there has been a growing interest in developing technologies for recognizing and interpreting sign language, particularly Arabic Sign Language (ArSL).

This complex task requires advanced signal processing techniques, such as data glove-based approaches, which capture fine-grained hand movements and gestures with high precision.

In this section, we will discuss the fundamental concepts and techniques of signal processing applied in Arabic Sign Language Recognition.

2.1.1 Fundamentals of Signal Processing

Signal processing involves a range of techniques used to manipulate and analyze signals to extract valuable information.

In the context of Arabic Sign Language recognition, signal processing techniques are employed to process the data captured by the data glove. These techniques include signal acquisition, preprocessing, feature extraction, classification, and post-processing.

Signal acquisition involves converting analog signals into digital format using ADCs, followed by preprocessing techniques like filtering, normalization, and denoising.

Feature extraction is crucial for representing hand gestures in ArSL, with techniques like feature selection and dimensionality reduction improving recognition efficiency.

Classification uses machine learning algorithms for accurate predictions, and post-processing refines recognition results by incorporating temporal, context-aware, or language models.[30]

2.1.2 Challenges and Considerations in SP for Data Glove-Based Approach

Signal processing techniques are crucial for Arabic Sign Language recognition, analyzing and interpreting captured signals, extracting features, and recognizing complex hand gestures.

This helps researchers build accurate systems and facilitates effective communication between deaf and broader society[39].

Data glove-based approaches capture hand movements and gestures with high precision, but signal processing challenges include noise, artifacts, sensor calibration, synchronization, variations in signing styles, and real-time processing requirements.

To overcome these, algorithm design, optimization, and validation through extensive testing and evaluation are crucial[38].

2.2 Categories of SLRS

Over the years, researchers have pursued sign language recognition (SLR) using two distinct approaches : the digital image processing (DIP) based approach and the data glove based approach.

In both approaches, the fundamental concept remains consistent - gathering a substantial amount of data and employing supervised learning algorithms to classify that data.

DIP-based systems use images of sign gestures for recognition, while data glove-based systems use glove sensor readings to capture hand and finger movements values.

While the core principle remains similar, there are noteworthy differences between the two approaches, which we will elaborate on in the following sections.

2.2.1 DIP based Approach for SLR

The DIP-based approach involves collecting a large dataset of images using a generic or specialized depth camera [46]. Machine learning algorithms are applied to train a model for classifying sign gestures based on these images.

The data collection process is cumbersome and image quality may be compromised by ambient lighting conditions[72]. Creating a portable system using this vision-based approach is challenging, as users must remain within the camera's field of vision, restricting their distance and motion.

Researchers have explored DIP-based techniques, such as Microsoft Kinect for sign language recognition [22], neural networks like backpropagation [59], and Hidden Markov

Models for character interpretation [51]. Techniques like histogram equalization, background rejection, skin color extraction, thresholding, morphological filtering, and template matching have also been applied [72].

Researchers have explored hybrid classification methods using K-Nearest Neighbor (KNN) and SVM algorithms [66], Lexicon-based approaches for finger-spelled words [66], and consumer depth cameras for real-time sign language recognition. These methods include rotation, translation, and scale invariant features[46].

The DIP-based approach uses images, machine learning algorithms, and techniques like neural networks, HMMs, and depth cameras to recognize sign language gestures. However, challenges like data collection, portability, and image quality remain significant.

2.3 Data Glove-Based Approach for ASLR

The data glove-based approach is a promising technique for Arabic Sign Language recognition, capturing accurate hand movements, finger positions, and gestures.

This section focuses on the data glove-based approach and its application in Arabic Sign Language recognition[3].

The data glove is a wearable device equipped with sensors that capture and measure the movements and positions of the wearer's hand and fingers. It offers a natural and intuitive way to track and record sign language gestures.

The data glove typically consists of flex sensors, accelerometers, gyroscopes, and sometimes additional sensors such as magnetometers [8].

Fig. 2.1 : Wearable-tech glove translates sign language into speech in real time.

2.3.1 Data Glove based Approach for SLR

Data gloves are sensors that capture hand movement features like bending, orientation, rotational motion, and contact. However, some commercial gloves are unsuitable for sign language recognition due to high costs or lack of specific sensors[72, 72]. Researchers often design custom gloves, which are more cost-effective and incorporate only the necessary sensors.

Research on sign language recognition using data gloves has limited focus on a comprehensive solution that includes glove construction, continuous detection, and alphabet recognition [23, 84].

Patil et al[72] attempted to classify all 26 ASL alphabet characters based on finger bending, but this approach is not feasible due to unclear guidelines on determining ranges of flex sensor values. Elmahgiubi et al[23] described a data glove capable of detecting 20 ASL alphabet letters, but did not specify boundary values for ranged queries. None of these studies addressed sign language modeling, which involves deriving meaningful messages from data streams.

Mehdi et al[53] utilized a 5DT 7-sensor glove for data collection and used an Artificial Neural Network for sign classification. They focused on detecting 24 letters of the ASL alphabet, omitting two dynamic letters. They discussed sampling rate and sign language modeling, but did not elaborate further.

SLARTI, developed by the University of Tasmania, recognizes Australian Sign Language [66],Using four manual features : handshape, orientation, place of articulation, and motion. It uses four feature-extraction neural networks for each feature.

2.3.2 Advantages and Applications of Data Glove in SLR

Data gloves offer several advantages in Arabic Sign Language recognition, including high precision, natural interaction, real-time recognition, and multi-modal input.

They capture fine-grained hand movements, enabling signers to express themselves without additional equipment or markers. [60]

These gloves also enable immediate interpretation and response, enhancing recognition accuracy. They also find applications in virtual reality, human-computer interaction, and rehabilitation, providing a means to track and interpret hand movements for immersive experiences, gesture-based interfaces, and therapeutic interventions[60].

2.3.3 Data Collection and Preprocessing with Data Gloves

Data collection is crucial for developing robust Arabic Sign Language recognition systems. Data gloves capture native signer gestures, providing valuable resources for training and evaluating machine learning models [54].

Preprocessing is essential for improving glove data quality using techniques like :

1. Sensor calibration : involves adjusting data gloves for hand sizes and joint range.

2. Gesture segmentation : Segmenting individual sign language gestures in captured data.

3. Noise reduction : involves filtering or smoothing techniques to remove sensor noise and enhance signal quality.

2.3.4 Signal Processing Techniques for Data Glove-Based Approach

Signal processing techniques extract features from data captured by data gloves, including finger bending angles, joint angles, and hand velocities.

These techniques aid in gesture recognition, configuration and movement capture, and estimation of hand movements using accelerometer and gyroscope data.

Extracted features aid machine learning algorithms for gesture classification and recognition, enabling accurate interpretation of Arabic Sign Language gestures with data gloves, combining signal processing and machine learning techniques[69].

2.4 Feature Extraction and Representation in Data Glove-Based Approach

Feature extraction is crucial for recognizing sign language gestures using data glove-based approaches. This section discusses commonly used features, methods, and algorithms used to extract meaningful information from data glove sensor data[68].

Feature extraction is crucial for sign language recognition using the data glove-based approach, as it reduces data dimensionality and highlights patterns, enabling accurate and efficient recognition of gestures[56].

Commonly used features for data glove-based sign language recognition include finger flexion, hand orientation, hand shape, and hand motion. These features help understand gestures, capture unique characteristics of signs, and track hand movement for accurate recognition, ensuring meaningful and meaningful sign language recognition[70].

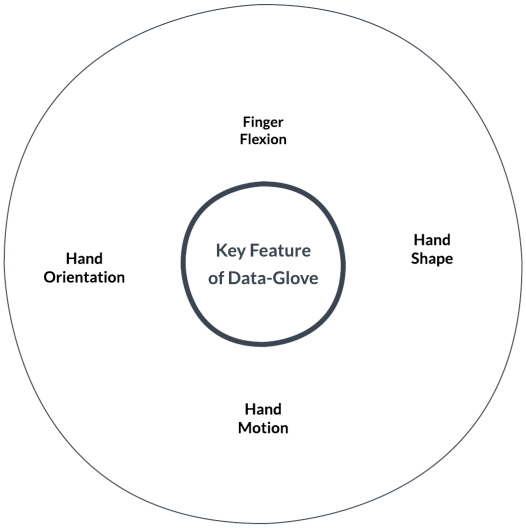


Fig. 2.2 : Exploring Key Features of Data Glove-Based Approach.

2.4.1 Feature Extraction Methods and Algorithms

Data glove-based approach employs various methods and algorithms for feature extraction, aiming to extract meaningful information from sensor data [27], including :

Statistical Features : Calculating measures like mean, standard deviation, or variance to capture important properties of sensor data.

Time-domain : Analysis analyzes sensor signals' temporal characteristics to capture dynamic hand movements, including zero-crossing rate and energy.

Frequency-domain : Analysis extracts information on gesture components using techniques like FFT and power spectral density analysis.

Wavelet analysis : Decomposes sensor data using transforms to capture time and frequency information for multi-resolution hand movement analysis.

Principal Component Analysis (PCA) : Reduces sensor data's dimensionality by focusing on significant variations in lower-dimensional space.

2.5 Machine Learning Algorithms for ARSLR

Machine learning algorithms are crucial for Arabic Sign Language recognition, enabling automatic interpretation and classification of hand movements.

Machine learning plays a vital role in Arabic sign language recognition using the data glove-based approach. It enables the system to learn from the collected sensor data and make predictions or classifications based on the learned patterns[77].

Machine learning algorithms are employed to train models that can accurately recognize Arabic sign language gestures.

2.5.1 Supervised and Unsupervised Learning Algorithms

In Arabic sign language recognition, both supervised and unsupervised learning algorithms are utilized.

Supervised Learning Algorithms : These algorithms learn from labeled data, where each data sample is associated with a specific sign language gesture.

They are trained using a large corpus of labeled sensor data to learn the patterns and relationships between the features and corresponding gestures[78] .

Common supervised learning algorithms used in Arabic sign language recognition include :

1. Support Vector Machines (SVM)

SVM (Support Vector Machine) is such a popular ML technique that it can constitute an entire group on its own. It uses a separation hyperplane or decision boundary to define the decision boundaries among a set of data points classified with different labels.

It is a strictly supervised classification algorithm, meaning that the algorithm develops an optimal hyperplane using input data or training data, and this decision boundary, in turn, classifies new examples. Depending on the kernel used, SVM can perform both linear and non-linear classification.

This algorithm has strong regularization and can be used for both classification and regression tasks. It is characterized by the use of kernels, solution sparsity, and capacity control obtained by adjusting the margin or the number of support vectors...[57].

2. Random Forests

It is an enhanced version of the decision tree used for supervised learning. This classifier creates a certain number of decision trees and then uses majority voting to determine the final prediction.

This algorithm is more efficient than decision trees because they work together as a group and correct each other's errors, while in the random forest algorithm, the trees are not linked to each other. This algorithm can be used for both classification and regression tasks [50].

3. K-Nearest Neighbors (KNN)

The KNN algorithm (K-Nearest Neighbors) is a fundamental and straightforward classification technique when there is little or no prior knowledge about the data distribution.

It can be used for both regression and classification tasks, but it is primarily used for classification problems.

KNN allows the classification of instances based on their nearest neighboring instances in the feature space [48].

4. Neural Networks

A recurrent neural network (RNN) involves sequential processing of data for learning. This process is justified by its ability to remember what has occurred in the preceding sequence being processed. It is called recurrent because the output at each time step is used as input for the next time step, achieved by remembering the output from the previous time step. This, in turn, allows us to learn long-term dependencies in the training data.

RNN is composed of layers with memory cells. There are different types of memory cells to use in RNN, one of which is the Long Short-Term Memory (LSTM) unit or cell.

LSTM consists of a cell state and a carry along with the current word vector being processed when the sequence is processed at each time step. The carry is responsible for ensuring that there is no loss of information during the sequential process [87].

The anatomy of an LSTM cell is shown in Figure 2.3. An LSTM cell is composed of weights and three different gates for learning. At each time step, there is an input gate for the current input, an output gate for predicting values, and a forget gate used to discard irrelevant information.

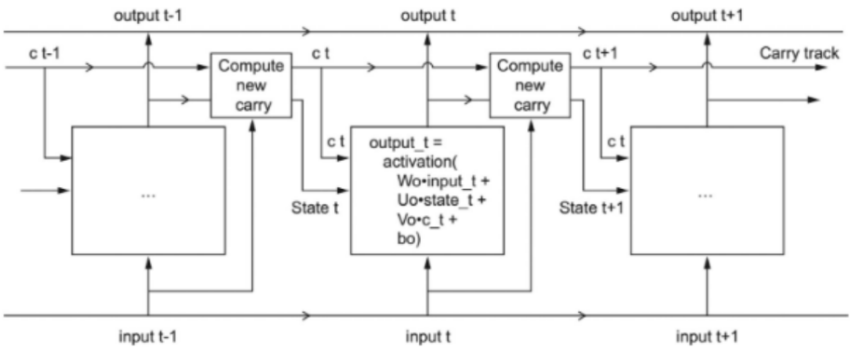


Fig. 2.3 : The anatomy of an LSTM cell.

LSTM networks have been found to be more effective than RNNs when information needs to be preserved over a longer period than a typical RNN can handle [35].

Unsupervised Learning Algorithms : These algorithms work with unlabeled data,

aiming to discover hidden patterns or structures in the data without prior knowledge of the sign language gestures.

Unsupervised learning can be used for tasks like clustering similar gestures or dimensionality reduction [37].

Popular unsupervised learning algorithms used in Arabic sign language recognition include :

1. K-Means Clustering

The K-Means Clustering algorithm is a widely used unsupervised machine learning technique for clustering data points into distinct groups. It partitions data into K clusters, with K representing the number of predefined clusters specified by the user.

The algorithm randomly initializes K centroids, assigns each data point to the nearest centroid, and recalculates the centroids until convergent. The goal is to minimize the sum of squared distances between data points and their assigned centroid, ensuring similarity and differences between clusters. K-Means is widely used in applications like image segmentation, customer segmentation, anomaly detection, and pattern recognition.

However, it has limitations, such as sensitivity to initial centroid selection and difficulty handling non-linearly separable data.[36]

2. Hierarchical Clustering

The Hierarchical Clustering algorithm is a widely used unsupervised machine learning technique for clustering data points into nested clusters. It merges or agglomerates clusters based on distance metric, and continues until all data points are part of a single cluster or a stopping criterion is met. There are two main types :Agglomerative and Divisive.

Agglomerative clustering merges similar clusters iteratively, while Divisive clustering divides clusters recursively, forming smaller clusters.

The result is typically represented as a dendrogram, illustrating hierarchical relationships between clusters. Although widely used in various domains, it can be computationally expensive and sensitive to distance metric and linkage criteria[41].

3. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a widely used dimensionality reduction technique in data analysis and machine learning. It transforms high-dimensional datasets into lower-dimensional spaces while retaining important information or patterns.

PCA identifies principal components, which capture the most variation in the data. The steps involve standardizing the data, computing the covariance matrix, computing

eigenvectors and eigenvalues, selecting principal components, and transforming the data. It helps to understand data structure and improves machine learning algorithms by reducing the dimensionality of the feature space[21].

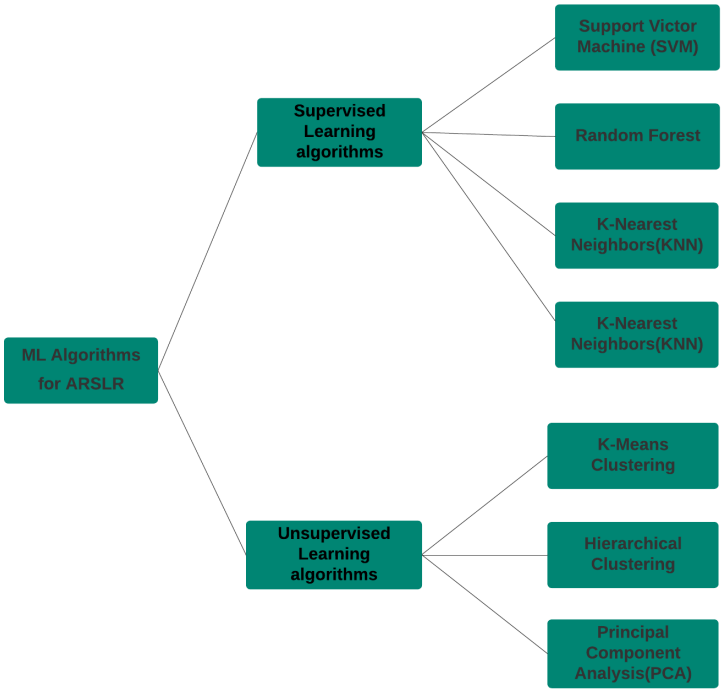


FIG. 2.4 : ML algorithms for ArSLR.

2.6 Dataset Considerations in Arabic Sign Language Recognition

Datasets are crucial for training and evaluating Arabic sign language recognition systems.

High-quality, representative datasets are essential for accurate models and assessing recognition algorithms' performance.

Well-designed datasets enable generalization to unseen data and real-world scenarios.

Existing datasets for Arabic sign language recognition may be limited compared to other languages .[28], but include :

2.6.1 Challenges and Limitations of Dataset Collection

Collecting datasets for Arabic sign language recognition poses several challenges and limitations [28], including :

Tab. 2.1 : Datasets for Arabic Sign Language Recognition

Dataset	Description
ASLG-PC12	This dataset contains video recordings of 12 signers performing Arabic sign language gestures. It covers a set of predefined signs and includes both isolated and continuous signing sequences.
Arabic Sign Language (ArSL) Corpus	This corpus consists of video recordings of multiple signers performing a wide range of Arabic sign language gestures. It provides a comprehensive collection of different signs and variations.
ArabiCorpus	ArabiCorpus is a large-scale Arabic multi-modal corpus that includes Arabic sign language data along with other modalities. It covers various domains, including news, interviews, and conversations.
Custom Collected Datasets	Researchers and developers may also create their own datasets by collecting data from native signers. These custom datasets can be tailored to specific requirements and address the limitations of existing datasets.

2.7 Model Development and Training

Model development and training are essential stages in Arabic Sign Language recognition systems.

This section discusses architecture, design, training, optimization techniques, and evaluation metrics.

Model architecture and design are crucial in sign language recognition, affecting performance by choosing deep learning architectures, layers [26], and recurrent or convolutional layers.

The training process involves feeding the dataset and updating parameters to minimize loss function. Optimization techniques like stochastic gradient descent, Adam, or RMSprop can improve convergence speed and overall performance.[61]

Evaluation metrics like accuracy, precision, recall, and F1 score provide insights into the model's ability to correctly classify sign language gestures[7].

Conclusion

In this chapter, we have delved into the domain of Arabic Sign Language (ArSL) recognition utilizing a data glove-based approach. Our exploration provides a solid foundation for future research and development in the field of ArSL recognition using data

Tab. 2.2 : Challenges and Limitations of Dataset Collection

Challenges	**Description**
Limited Availability	Compared to widely studied sign languages like American Sign Language (ASL) or British Sign Language (BSL), datasets for Arabic sign language may be relatively scarce. This scarcity can hinder the development and evaluation of Arabic sign language recognition systems.
Cultural and Linguistic Variations	Arabic sign language exhibits variations across different regions and dialects. It is essential to consider these variations when collecting datasets to ensure the coverage of diverse gestures and linguistic features.
Annotation and Standardization	Annotating and standardizing large-scale sign language datasets can be challenging due to the complex nature of sign language gestures. Ensuring accurate and consistent annotations across different signers and variations is crucial for reliable training and evaluation.
Data Collection Setup	Collecting high-quality data requires careful consideration of the recording setup, including camera angles, lighting conditions, and background noise reduction. These factors can impact the quality and reliability of the collected dataset.

glove-based approaches.

Chapter 3

Design of ARSLR System

Introduction

This chapter provides a comprehensive overview of the research design and methodology adopted for the Arabic Sign Language Recognition (ARSLR) project. It encompasses various aspects, including the research approach, experimental setup, system design, architectural design, system analysis, and hardware and software requirements.

The chapter aims to establish a solid foundation for the subsequent chapters by outlining the overall structure and organization of the ARSLR system.

3.1 A comprhensive Research Design for ARSLR

This study employs a design and creation approach, aiming to develop and analyze an artifact. As described by [71], design science research involves the creation of new knowledge, showcasing academic qualities, and focusing on improvement, invention, and exaptation through the use of an artifact.

In this case, the artifact being developed is an embedded intelligent device. Consequently, the development process will adhere to the phases involved in embedded systems development.

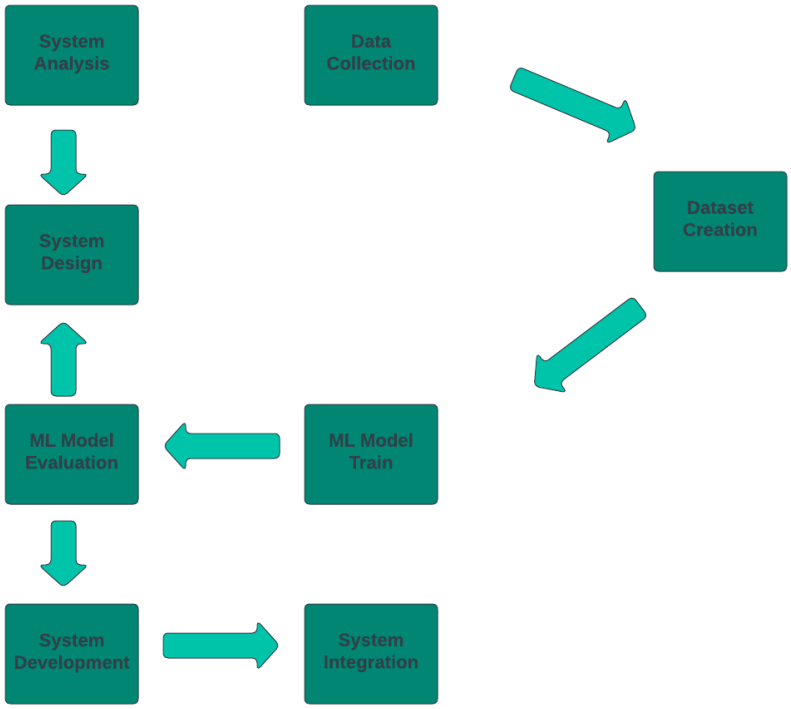


Fig. 3.1 : Research Design

The development process will follow the steps outlined in Figure3.1 , as depicted in the research and the Figure 3.2 outlines the design and creation process .

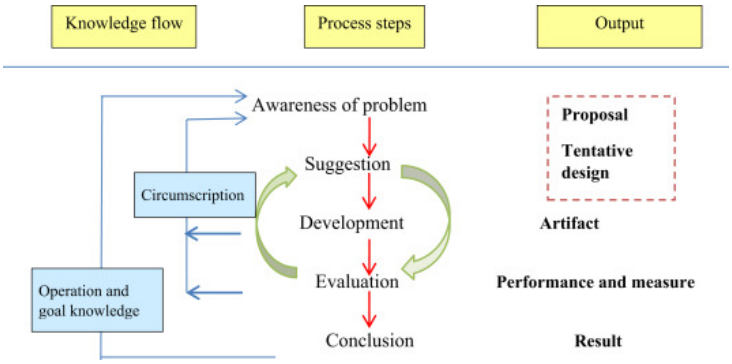


FIG. 3.2 : Design Science Research Process Model as adopted from Saltuks' Design and Creation

3.2 ARSLR System Design

The design goals for this study are as follows :

i) Development of a sign language gesture recognition device based on Arduino Nano 33 BLE Sense and Flex Sensors.

ii) Scanning and analysis of inputs from Flex Sensors.

iii) Outputting the data in CSV format.

iv) The Arduino Nano 33 BLE Sense uses 64MHz clock, 1MB Flash memory, 256kB SRAM, and various interfaces for basic hardware.

In this study, the V-Model development methodology is employed.

The V-Model is a sequential software development life cycle (SDLC) model that follows a V-shaped structure [25]. It is also referred to as the Verification and Validation model.

In this approach, each development stage is associated with a corresponding testing phase. Each phase must be successfully completed before progressing to the next phase. Figure 3.3 illustrates the different steps involved in the development, verification, and validation processes.

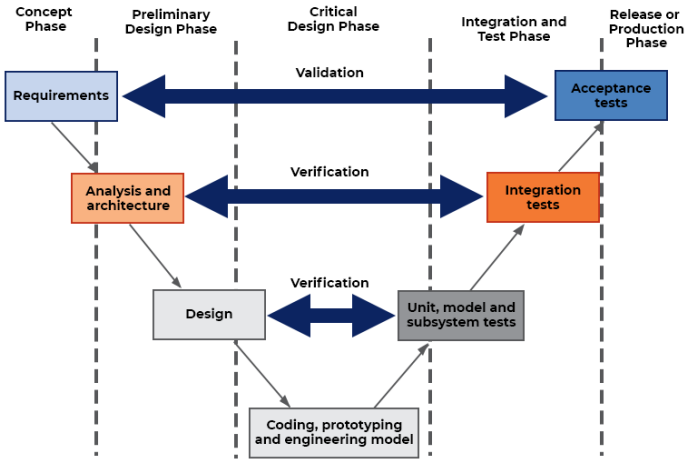


FIG. 3.3 : V - model design methodology adopted for embedded software development

3.3 Architectural Design of ARSLR

The system design is divided into separate modules, each handling different functions. It encompasses data transfer and communication between internal modules and external systems.

The provided block diagram illustrates the architecture of the system, which serves the purposes of data collection and performing inference on the embedded device.

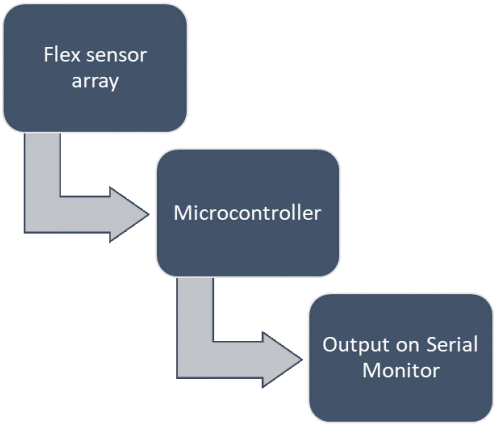


FIG. 3.4 : System Architecture block diagram

3.3.1 Coding

Coding involves the creation of embedded software for the components utilized in constructing a system or embedded device. In the case of a larger system, this process is commonly referred to as software development.

The Arduino platform, an integrated development platform equipped with numerous

libraries for a wide range of sensors available in the market, serves as the chosen platform for this development. Utilizing these libraries during the development process significantly reduces both the time and overall cost associated with developing an embedded system.

Throughout the development phases, three program files are employed :

1. A program for the data collection phase, which employs sensors to collect and log data in CSV format.

2. The machine learning phase, where the gathered data is utilized to train a model that functions as a classifier.

3. The classifier program, which is uploaded onto the embedded device to capture new data and leverage the loaded model for classification purposes.

3.3.2 Data Collection

Instead of relying on a pre-existing dataset, data collection is conducted to create a new dataset specifically intended for training the classifier.

We opt to collect two separately dataset one for the left hand and other for the right one cause we have sentences that use the both hands, each hand contain one arduino nano 33 BLE SENS to measure the gesture and five flexes sensors to measure the bending of fingers .

We choosed 103 sentences that use only hands without touching any part of the body .

The provided diagram 3.5 illustrates the comprehensive system design for the data collection process.

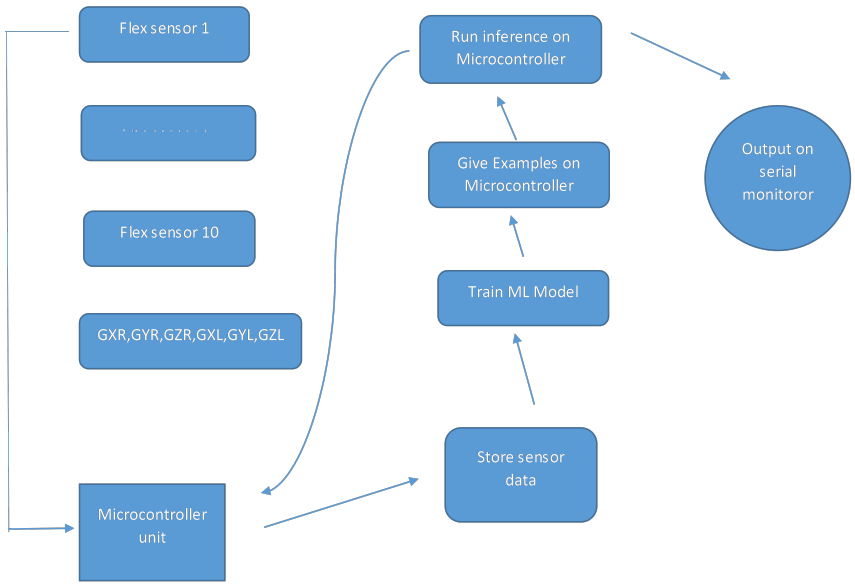


Fig. 3.5 : Data Collection and Model Training Block Diagram

The training and testing data for this study will be acquired from the ten flex sensors. To create the model, TensorFlow Lite for Microcontrollers framework will be employed on a PC.

Prior to being converted and uploaded onto the Microcontroller Unit (MCU), testing and validation will be conducted on the PC. Once the model is successfully uploaded onto the MCU, the inference process will be executed directly on the device, eliminating the need for PC dependency.

3.4 ARSLR System Analysis

3.4.1 System Requirements

1. Overview

During the analysis phase, the proposed system's requirements and constraints, which define the operational boundaries of the system, are examined.

The requirements consist of specific parameters that the system needs to fulfill. Below are the outlined requirements, which are subsequently translated into a more comprehensive specification.

2. Objectives

The primary goal of this project is to create a Sign-Glove (SG) capable of capturing and interpreting the gestures and finger movements used in sign language. The SG will be worn by the individual performing the signing, and it will accurately translate the captured gestures.

More specifically, this study focuses on interpreting phrases using the Arabic Sign Language (ArSL).

3. Process

To facilitate the development of this system, multiple factors are taken into consideration, and in-depth discussions regarding hardware and software requirements are presented in the subsequent sections.

In this development project, specific factors are prioritized. The device should be lightweight, consume minimal power , and maintain a high level of accuracy when recognizing and translating gestures.

The following list outlines the measures that have been considered for analysis :

1. Accuracy : The accuracy of the device is determined by calculating the disparity between the expected values and the recorded values. The desired level of accuracy for the device is set at a minimum of 80

2. Precision : For the purpose of this project, a total of one hundred distinguishable measurements are needed. These measurements correspond to each sentence or gesture captured during the system's operation.

3. Size and Weight : The physical dimensions of the device are essential considerations, particularly regarding the space it occupies. As the device will be worn on both hands, it is important for the package to be compact in order to fit comfortably behind the glove.

4. Power : Power consumption is an important requirement for the device since it is intended to be a wearable that operates on battery power. However, for the purpose of prototype development, this requirement will not be strictly adhered to as the device will be powered by a USB connection from a PC.

5. 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 .

3.5 Hardware Requirements and Analysis

This section delves into the specific hardware components employed to capture gestures, convert them into electrical signals, and interpret them.

In the context of this project, the hardware, particularly the sensors and microcontroller, assume a critical role in capturing analog data. This data is utilized for two key purposes : training a machine learning algorithm and testing the resulting model to assess metrics such as accuracy levels, precision, and resolution.

The sensors and microcontroller are essential elements in the gesture recognition system, as they enable the acquisition of precise analog data necessary for both the training and evaluation stages of the machine learning process.

The subsequent subsections provide a detailed discussion on the necessary hardware components, encompassing both sensors and the microcontroller unit.

3.5.1 Flex Sensors

Flex sensor measures deflection or bending, gaining popularity in the 1990s as a gaming interface in Nintendo Power Glove. It is a low-cost, user-friendly sensor.

The technology has various applications, including measuring joint movement, detecting doors, walls, and sensing pressure on robotic grippers, as well as door sensors and bumper switches.

1. Flex sensor overview

A flex sensor serves as a variable resistor that adjusts its resistance when it is bent. Its proportional relationship between resistance and bending has earned it the name "Flexible Potentiometer".

These sensors come in two main sizes : one that is 2.2 inches (5.588 cm) long, and another that measures 4.5 inches (11.43 cm) in length.

A flex sensor is composed of a phenolic resin substrate with a layer of conductive ink applied to it, as shown in Figure 3.6

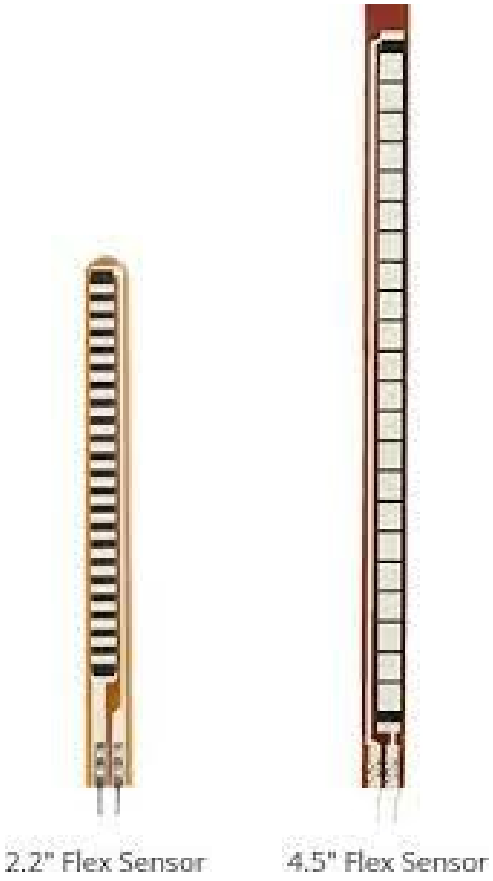


Fig. 3.6 : Flex Sensor

A segmented conductor is then placed on top of the ink layer to create a flexible potentiometer that changes resistance with deflection.

However, it's important to note that flex sensors are designed to bend in only one direction, away from the ink layer. Bending the sensor in the opposite direction can cause damage.

Additionally, care should be taken not to bend the sensor too close to the base, as the lower part of the sensor where the pins are crimped on is delicate and can break easily when bent.

2. Flex Sensor Working

The conductive ink applied to the sensor acts as a resistor. In its straight position, the resistance of the ink is approximately 25k, as illustrated in Figure 3.7

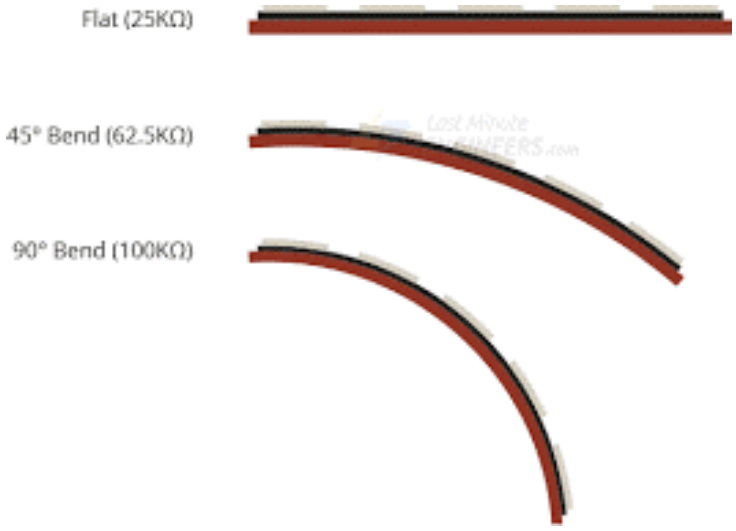


Fig. 3.7 : Flex Sensor working

When the flex sensor is bent, its conductive layer is stretched, causing a reduction in cross-sectional area, similar to stretching a rubber band. This reduction in cross-sectional area leads to an increase in resistance. At a 90-degree angle of deflection, the resistance of the sensor is approximately 100KΩ.

Once the sensor is straightened again, the resistance returns to its original value.

By measuring the resistance, it's possible to determine the degree of bending of the sensor.

3. Reading a Flex Sensor

To obtain readings from a flex sensor, the most straightforward approach is to create a voltage divider circuit by connecting the sensor with a fixed-value resistor (typically 47kΩ).

This involves connecting one end of the sensor to a power source and the other end to a pull-down resistor, as demonstrated below.

Subsequently, the connection point between the fixed-value pull-down resistor and the flex sensor is attached to the ADC input of an Arduino. This produces a variable voltage output that can be detected by the ADC input of the Arduino.

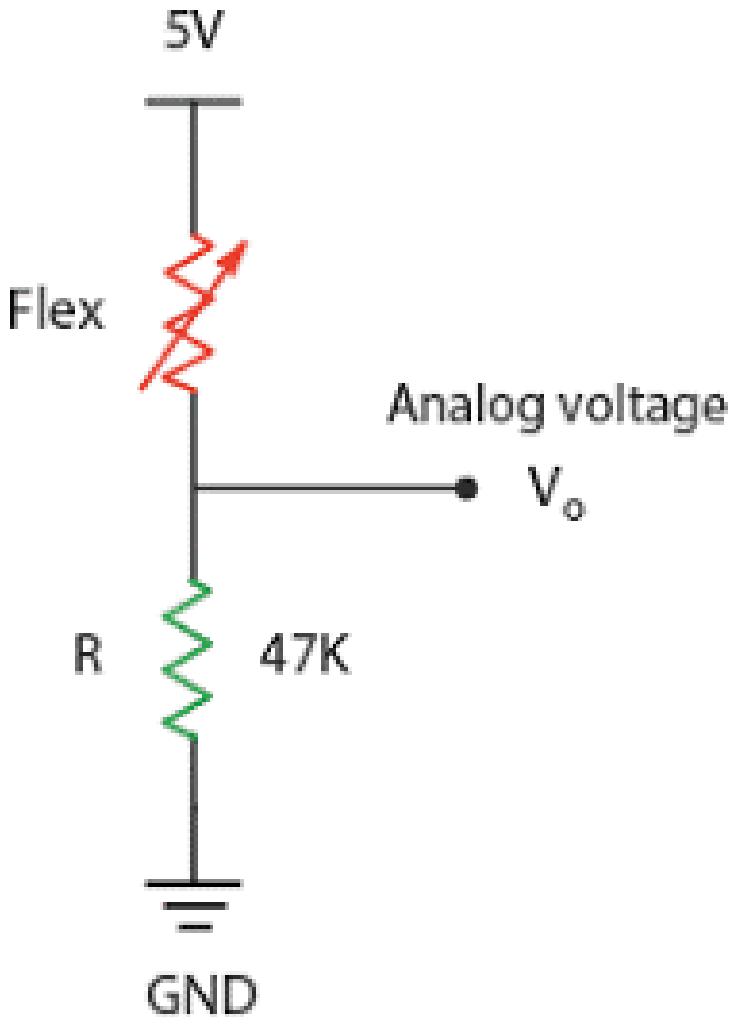


Fig. 3.8 : Reading a Flex Sensor

It's important to note that the voltage measured is the voltage drop across the pull-down resistor, not the flex sensor. In the configuration depicted, the output voltage declines as the bend radius grows larger.

The voltage divider arrangement's output is defined by the following equation :

4. Basic Flex circuit and Characteristics

The circuit diagram in Figure 3.9 illustrates the basic flex sensor comprising two or three interconnected sensors.

The flex sensors' outputs are directed to an op-amp, utilizing a non-inverted configuration to increase their voltage. The output voltage decreases as the degree of bending increases. The output voltage is calculated using the voltage divider rule.

Fig 3.10 displays the characteristics of the flex sensor circuit, where R1 refers to the input resistor connected to the non-inverting terminal.

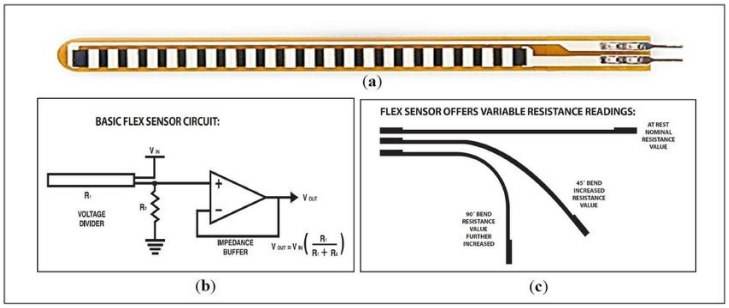


Fig. 3.9 : Basic Flex Circuit

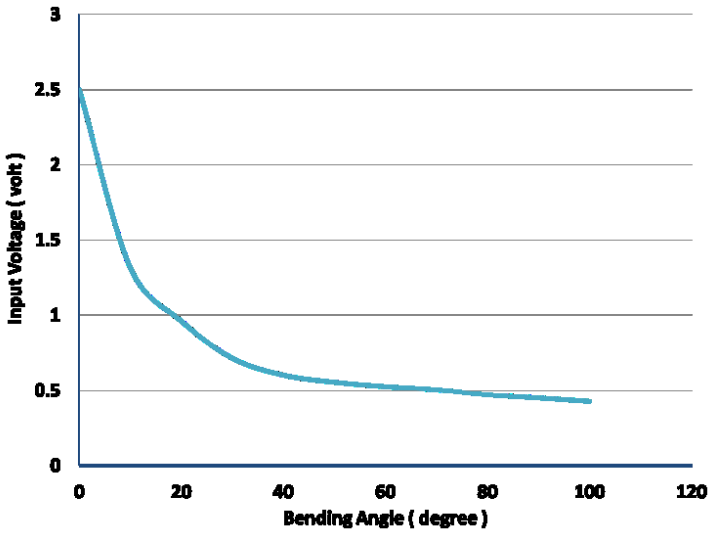


Fig. 3.10 : Characteristics of flex sensor

3.5.2 Microcontroller

The microcontroller serves as the central processing unit, flash memory, and RAM of the entire device, acting as its "brain." For this project, it is crucial to have a microcontroller capable of running TensorFlow Lite for Microcontrollers machine learning models.

Therefore, the Arduino Nano 33 BLE Sense board was selected due to its compatibility with these requirements. The Nano 33 BLE Sense is an AI-enabled development board from Arduino, characterized by its small form factor (45mm x 18mm) and operating at a voltage of 3.3V. This board incorporates various embedded sensors and is equipped with the Nordic Semiconductor nRF52840, a relatively powerful low-power 32-bit ARM Cortex M4 CPU running at 64MHz.

Additionally, it has 128MB of RAM, enabling efficient processing and storage capabilities. Notably, the processor also supports Bluetooth Low Energy (BLE) and Near Field Connectivity (NFC), allowing for seamless data pairing and transmission with external devices.

These sensors include :

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

1. 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)

Tab. 3.1 : Technical specification of Arduino Nano BLE sense

The diagram below shows the nano 33 board and all its onboard sensors [45]

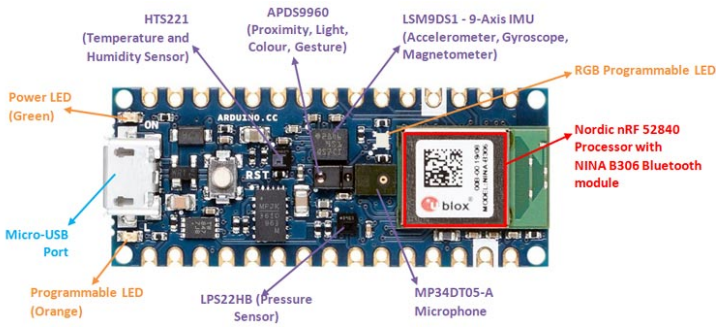


Fig. 3.11 : Arduino nano 33 Ble sense development board

The voltage output (Vout) from the flex sensor - resistor voltage divider is directly connected to five out of the eight available analog inputs of the Nano 33 board.

These specific pins are directly linked to a 12-bit analog-to-digital converter, which performs the crucial tasks of quantizing, sampling, and converting the analog voltage into a digital signal.

This digital signal can then be processed by the microcontroller's CPU for further analysis and utilization.

3.6 Software Requirements and Analysis

According to [32] , software can be defined as a collection of instructions, data, or programs that enable the operation of computers and the execution of specific tasks.

In the following section, we will analyze the different software components necessary for various aspects of the project. This includes software for data acquisition, the machine learning algorithm, and embedded software for running on-device inference.

3.6.1 Embedded Software

Embedded systems differ from typical computers in that they are highly specialized devices with limited resources in terms of time, processing power, and memory.

Consequently, they necessitate the use of embedded software to effectively control and operate these devices.

It's important to note that embedded software is often used interchangeably with firmware, as mentioned by [24].

The term "embedded software" refers to the specific software components designed to run on embedded systems, which require efficient utilization of resources and tailored functionality to meet the unique requirements of these specialized devices.

There are two essential categories of embedded software that are required for this project.

Firstly, we need software that captures data from the sign glove for all the gestures. Each gesture is distinctive, meaning that each sensor on each of the ten fingers will produce a unique sensor voltage value.

The data capture embedded software is responsible for collecting these values and outputting them to a file in a specified format, as illustrated below.

Sensor1	Sensor2	Sensor3	....	Sensor10	Sentence
....	....	....	...	...	...

Tab. 3.2 : Data capture format

Each column header for sensor represents the corresponding finger. E.g., Sensor 1 corresponds to Finger 1 of the left hand which is the small finger, all the way to Sensor 10 on the thumb finger , the other column are for the gesture (GXL,GYL,GZL, GestureL,GXR,GYR,GZR, GestureR).

Finally, on the last column we have the sentence 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 ten sensors is then tabulated on the table above to create a dataset. The dataset created is then used for ML model development.

3.6.2 TensorFlow

According to[19] , TensorFlow is a comprehensive and open-source platform for machine learning. It provides a complete set of tools and libraries for developing and deploying machine learning models.

On the other hand, TensorFlow Lite for microcontrollers is a lightweight version of TensorFlow specifically designed and optimized to operate on devices with limited resources. It does not rely on an operating system, dynamic memory allocation, or standard C/C++ libraries.

The TensorFlow Lite for Microcontrollers is compatible with the Arduino platform, making it suitable for various applications.

In this particular study, the researchers utilized TensorFlow Lite for Microcontrollers to develop and execute a 30kB Neural Net (NN) model. The objective was to recognize and distinguish gestures using flex sensors and an IMU (Inertial Measurement Unit).

The supported Arduino board for this study was the Arduino Nano BLE. By leveraging TFLite, the researchers were able to create an efficient and compact neural network model that could operate effectively on the limited resources available on the Arduino Nano BLE board.

1. Limitations of TFLite for Microcontrollers

TensorFlow Lite for Microcontrollers is specifically engineered to operate on microcontroller development boards with limited resources, unlike the standard TensorFlow Lite, which is designed to run on more powerful Linux-based embedded devices such as NVidia's Jetson Nano and Raspberry Pi.

Using TensorFlow Lite for Microcontrollers entails several specific limitations :

1. It supports only a restricted subset of TensorFlow operations, which impacts the range of architectures that can be executed.

2. Currently, the number of supported devices is limited, meaning that compatibility may be constrained.

3. Manual memory management is necessary when working with TensorFlow Lite for Microcontrollers as it utilizes a Low-level C++ API.

4. On-device training is not supported, thus requiring training to be performed on a more capable environment and subsequently loading the trained model onto the microcontroller.

3.6.3 Recurrent Neural network (RNNs)

RNN is a supervised machine learning algorithm for processing sequential data, like time series or natural language. It uses a network of interconnected nodes with recurrent connections to capture temporal dependencies, unlike k-nearest neighbors (kNN) which relies on distance measures.

This project chose RNNs over kNN for their ability to model and capture complex patterns in sign language data, making them ideal for tasks like gesture recognition. RNNs learn and recognize long-term dependencies, enabling more accurate predictions. The entire implementation process, including training and inference, can be done within the Arduino platform, providing a convenient and integrated solution.

3.7 Simulation

In the initial stages of the project, a simulation using Proteus was created to test the functionality of the Arabic sign language recognition system. Due to the unavailability of the Arduino Nano 33 BLE Sense library in Proteus, the testing was initiated using a single flex sensor to verify the bending capabilities. The flex sensor was connected to the Arduino board, and its response to different degrees of bending was observed and analyzed.

As the project progressed, the focus shifted to incorporating accelerometer data into the system. Since the specific library for the Arduino Nano 33 BLE Sense was not directly available in Proteus, an alternative approach was adopted by utilizing an accelerometer to measure and record the values of ax, ay, and az (acceleration in the X, Y, and Z axes). This step was crucial to understanding the hand's orientation and movement during sign language gestures.

Subsequently, the flex sensors were integrated to detect the bending of individual fingers. By combining the accelerometer data with the readings from the flex sensors, the system was able to interpret and recognize various hand gestures, forming a foundation for Arabic sign language recognition.

The integration of both the accelerometer and flex sensors provided a comprehensive dataset that allowed for the training and validation of the Arabic sign language recognition algorithm. This process was instrumental in achieving accurate recognition of sign language gestures, and the subsequent results will be discussed in further detail in the following sections of this thesis.

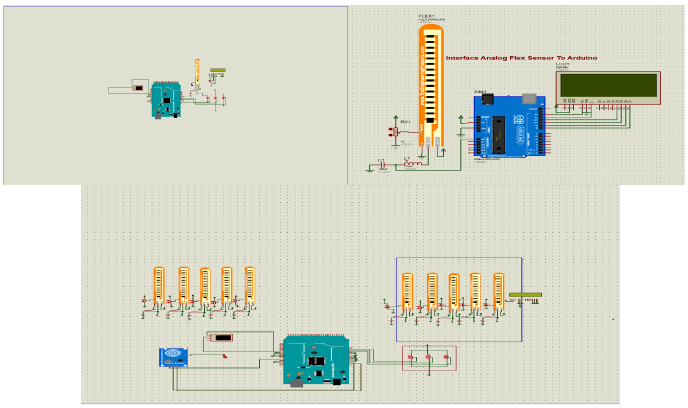


Fig. 3.12 : Simulation Steps of the Arabic Sign Language Recognition (ArSLR) System

Conclusion

In conclusion, Chapter 3 of the ARSLR study presents the system design, architecture, and analysis. It covers the design components, coding, data collection, system requirements, sign language, hardware and software requirements, and analysis. The chapter provides a comprehensive overview of the ARSLR system, its functionality, and the necessary elements for its successful implementation.

Chapter 4

ARSLR System Implementation

Introduction

The design and implementation of the system are covered in Chapter 4 of the ARSLR research, with an emphasis on the hardware architectural design, software design, and machine learning process. It also goes through how hardware and embedded software components are implemented. An extensive description of the system's design, implementation, and testing phases is given in the chapter's last part, which is devoted to testing and outcomes.

4.1 Toward an ARSLR prototyping

During this phase, the construction of the conceptual model for both the hardware and software components is undertaken. To streamline the process, the project is divided into various modules and subcomponents.

Additionally, this phase involves determining the cost, schedule, and anticipated performance of the system. The data flow diagram presented below illustrates the key components of the system and illustrates how data flows from one module to another.

To commence the hardware prototype construction, the initial step involves designing the architecture of the hardware system. This is accomplished by creating a high-level block diagram that outlines the overall structure and components of the prototype.

Subsequently, the subsequent subsections will delve into the conceptual and logical architectures, providing more detailed descriptions and explanations of the hardware system's design.

4.2 Hardware Architectural Design of ARSLR

The conceptual architecture of data collection hardware outlines the identification and allocation of responsibilities for components involved in the data collection process will be discused in the next sections.

4.2.1 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

FIG. 4.1 : Hardware Conceptual Architecture block diagram for data collection

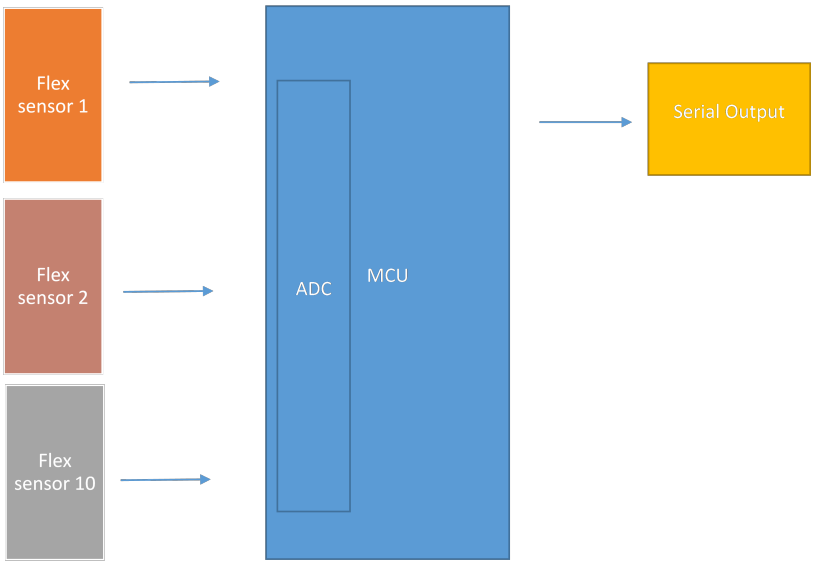


FIG. 4.2 : System Hardware Architecture

4.2.2 Logical Architecture

This architecture primarily emphasizes the interaction between components, the mechanisms and protocols for connecting them, and the design and specification of interfaces.

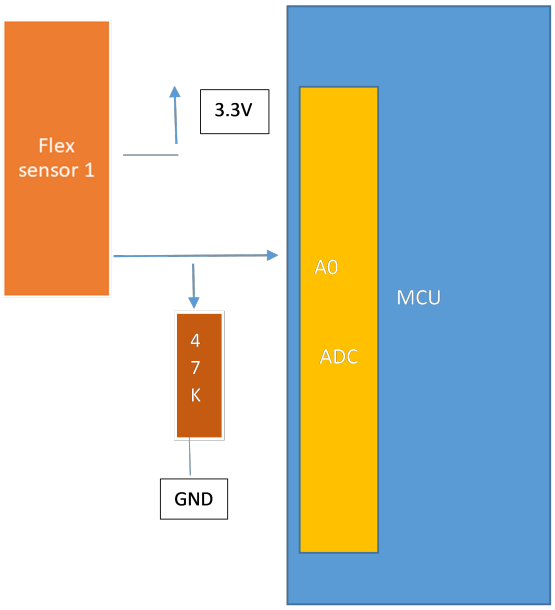


FIG. 4.3 : System Hardware logical architecture block diagram

The hardware configuration remains similar for both data collection and inferencing purposes, with the only distinction lying in the embedded program.

Specifically, for data collection, the source program follows a rule-based approach, adhering to traditional programming practices to read data from the sensors.

However, in the complete prototype, a combination of rule-based programming and machine learning models is employed.

4.3 Software Design

According to [63], software design is the process in which an agent formulates a specification for a software artifact, aiming to achieve specific goals while utilizing a set of primitive components and adhering to certain constraints.

In the context of this project, the software is categorized into two distinct categories.

Category 1 encompasses the embedded software that will be executed on the microcontroller.

On the other hand, Category 2 comprises the Python program, which will be employed in creating a machine learning model using Google Colab. This approach allows for the utilization of the high processing capabilities provided by Google Colab for model development purposes.

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

4.3.1 Embedded Software Design

The embedded software refers to the program specifically designed to run on the Arduino Nano 33 board.

Figure 4.4 depicts a block diagram illustrating the logic of the data collection process for building a dataset.

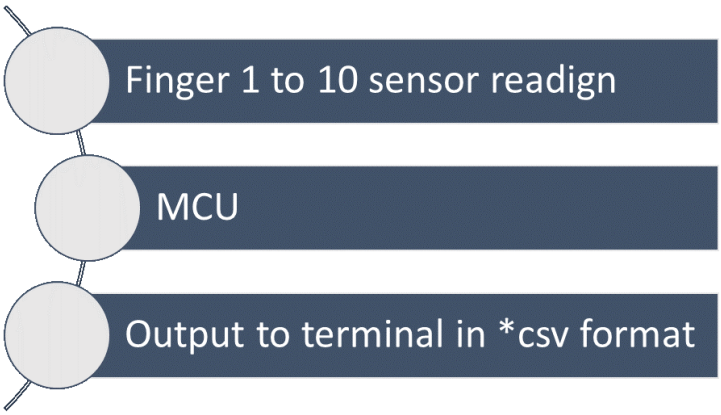


Fig. 4.4 : System software block diagram for data collection

4.4 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 generation
4. Pass the examples
5. Run Inference on device

The completed system is shown in a high-level diagram, figure 4.5 below.

The gestures from flex sensors representing each finger's position are captured as sensor data. This data is then passed on Neural network model. The neural network is trained on a dataset that contains examples of gestures for each class.

The output of the NEURAL network is a sentence or a sequence of words that describes the recognized gesture.

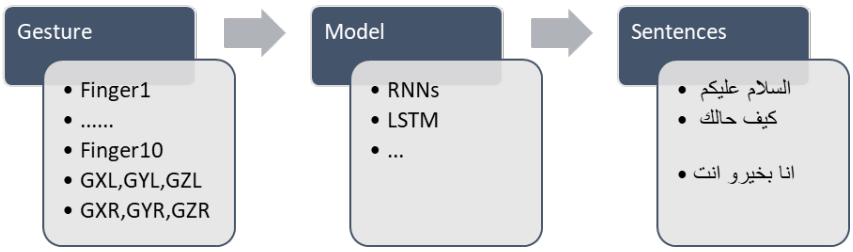


Fig. 4.5 : High Level end-to-end gesture classification process block diagram

4.5 Implementation of ARSLR system

During this phase, the system is developed through the creation of a prototype, which can involve simulating the system or constructing a physical prototype using components.

Both hardware and software debugging are performed at this stage. Debugging embedded devices presents challenges compared to debugging computer systems, mainly due to the absence of a keyboard and mouse.

Additionally, the hardware and embedded software run concurrently in real-time, making traditional debugging techniques like single-stepping and print statements impractical.

To facilitate the debugging process, a cross-compiler and assembler are employed to convert the source code into object code specifically designed for our target system, the nano 33.

The chosen approach for this project was a top-down implementation strategy, which offers the advantage of allowing the system's sub-components to be implemented concurrently.

4.5.1 Hardware Implementation

1. Control Circuit : The control circuit consists of three main components : a microcontroller, the nano 33 BLE development board, and an interface circuit. These elements are interconnected as depicted in the diagram provided

2. Glove :The hardware of the Signing Glove comprises a glove designed to accommodate flex sensors. The flex sensors are arranged in a voltage divider configuration using resistors, and the output from this configuration is connected to the analog pins of the Arduino Nano.

The accompanying figure presents a photograph depicting the Signing Glove.

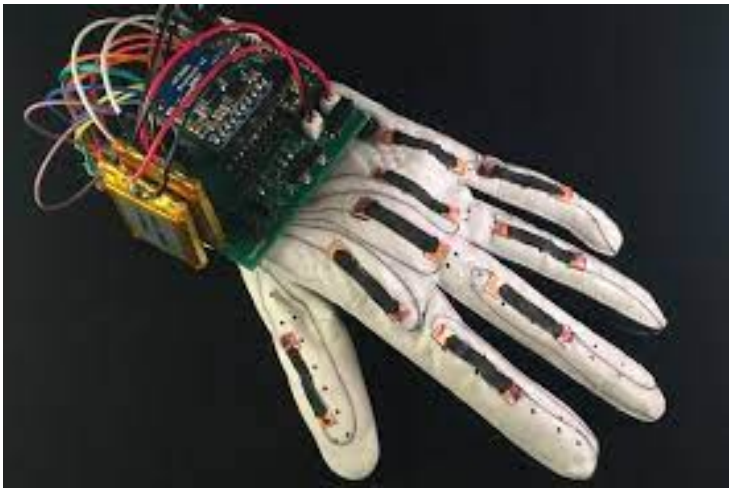


Fig. 4.6 : SL Glove Implementation

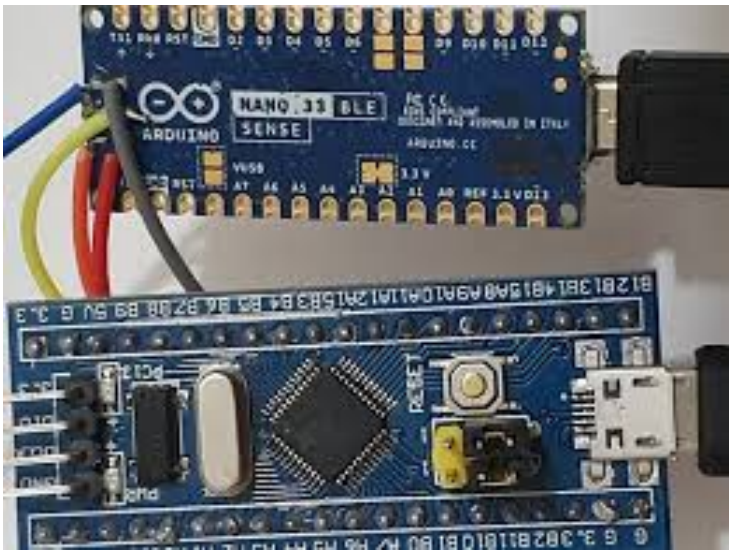


Fig. 4.7 : Microcontroller Wiring

4.5.2 Embedded Software Implementation

There are two embedded software implementations in this project. The first implementation, which is used for data collection, is provided below.

The code snippet presented is specifically for a single flex sensor. The complete program for all the sensors can be found in the appendix.

1. Program for Data Collection

- The project involves two embedded software implementations. The first implementation is used for data collection, while the second implementation focuses on cap-

turing gestures.

- The sensor data is formatted into a CSV format, specifically as follows :

  1. (sensor1, sensor2, sensor3, sensor4, sensor5, sensor6, sensor7, sensor8, sensor9)
  2. (GXL, GYL, GZL,GestureL,GXR,GYR,GZR,GestureR).

- The software logs the sensor data received from the serial communication line and saves it to a file for further analysis.

To perform testing on the flex sensors, the program output is displayed on the Serial Communication port. The provided image illustrates the test results of a flex sensor along with the corresponding gesture.

As the flex sensor is bent, the resistance value increases, and the (X, Y, Z) coordinates change accordingly based on the gesture of the two hands. These changes in sensor readings are logged as demonstrated in the image.

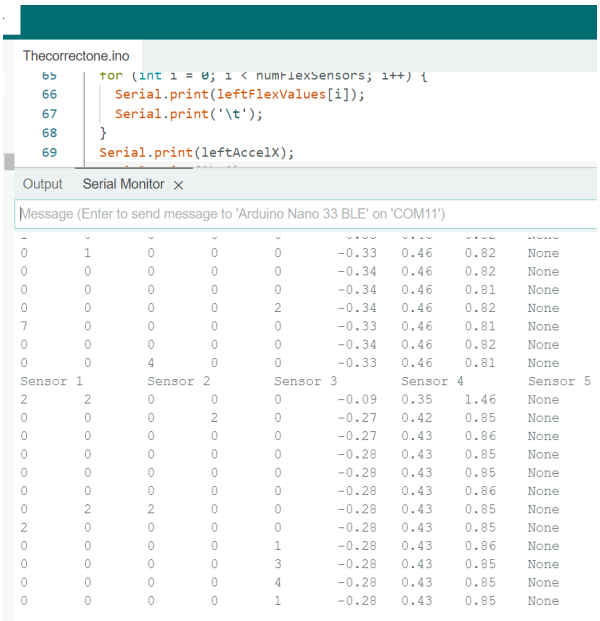


Fig. 4.8 : Testing Flex Sensors

To generate a dataset that includes data from all ten sensors, as well as the (GX, GY, GZ) values for both right and left hands, the following line of code is used :

Serial.print("sensor1, sensor2, sensor3, sensor4, sensor5, sensor6, sensor7, sensor8, sensor9, sensor10, GXL, GYL, GZL, GestureL, GXR,GYR,GZR,GestureR, sentence") ;

This line prints the column headers for the dataset, specifying the names of each sensor (sensor1 to sensor10)

the (GXL, GYL, GZL, GestureL, GXR,GYR,GZR,GestureR) coordinates.

Subsequently, the sensor values and gesture values are printed to the Serial output, separated by semicolons.

This output can be captured using a serial communication software such as COOLTERM, which allows the data to be saved as a separated values (*.csv) file for further analysis.

2. Logging Serial Data to File

a. In order to log Serial data being output we used CoolTerm which helps us to transfer our data from arduino IDE to get a csv file

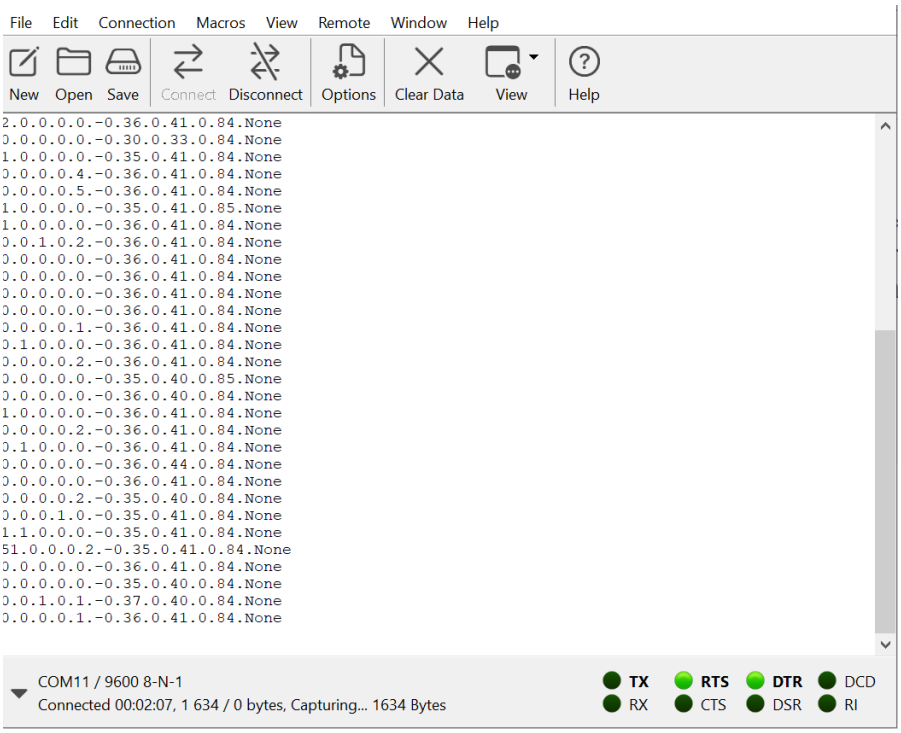
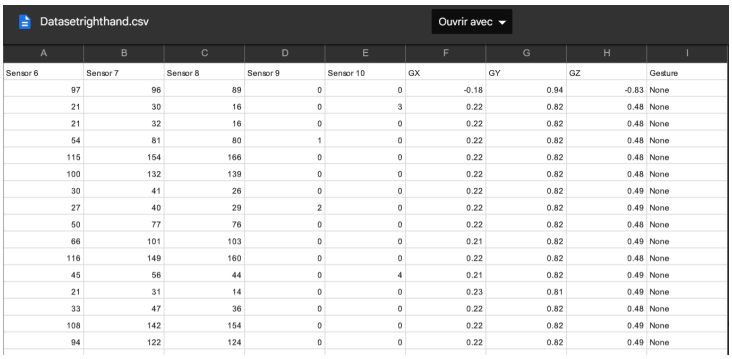


FIG. 4.9 : Serial output on CoolTerm

3. Data Curation Dataset Creation

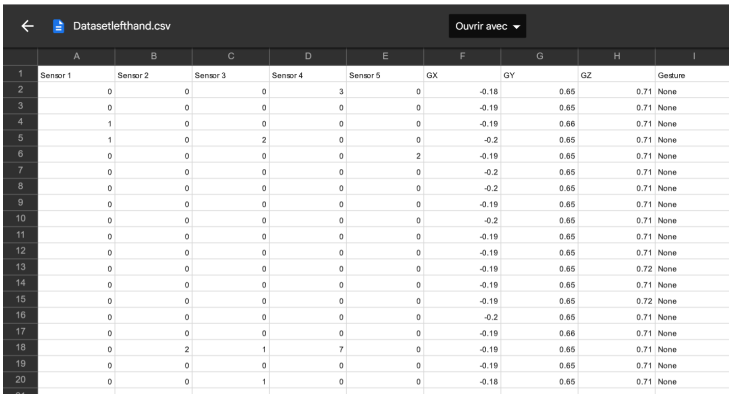
The sensor data is logged in two *.csv files, one for the right side and the other for the left side, containing 103 sentences. To create a dataset, the files were consolidated into a single file, and in the model usage, we have used them separately. The diagram below shows a snippet of the CSV file for the sentences in the curated file.



Datasetrighthand.csv — Ouvrir avec

A	B	C	D	E	F	G	H	I
Sensor 6	Sensor 7	Sensor 8	Sensor 9	Sensor 10	GX	GY	GZ	Gesture
97	96	89	0	0	-0.18	0.94	-0.83	None
21	30	16	0	3	0.22	0.82	0.48	None
21	32	16	0	0	0.22	0.82	0.48	None
54	81	80	1	0	0.22	0.82	0.48	None
115	154	166	0	0	0.22	0.82	0.48	None
100	132	139	0	0	0.22	0.82	0.48	None
30	41	26	0	0	0.22	0.82	0.49	None
27	40	29	2	0	0.22	0.82	0.49	None
50	77	76	0	0	0.22	0.82	0.48	None
66	101	103	0	0	0.21	0.82	0.49	None
116	149	160	0	0	0.22	0.82	0.48	None
45	56	44	0	4	0.21	0.82	0.49	None
21	31	14	0	0	0.23	0.81	0.49	None
33	47	36	0	0	0.22	0.82	0.48	None
108	142	154	0	0	0.22	0.82	0.49	None
94	122	124	0	0	0.22	0.82	0.49	None

FIG. 4.10 : Dataset created from Logged output on file



Datasetlefthand.csv — Ouvrir avec

	A	B	C	D	E	F	G	H	I
1	Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	GX	GY	GZ	Gesture
2	0	0	0	3	0	-0.18	0.65	0.71	None
3	0	0	0	0	0	-0.19	0.65	0.71	None
4	1	0	0	0	0	-0.19	0.66	0.71	None
5	1	0	2	0	0	-0.2	0.65	0.71	None
6	0	0	0	0	2	-0.19	0.65	0.71	None
7	0	0	0	0	0	-0.2	0.65	0.71	None
8	0	0	0	0	0	-0.2	0.65	0.71	None
9	0	0	0	0	0	-0.19	0.65	0.71	None
10	0	0	0	0	0	-0.2	0.65	0.71	None
11	0	0	0	0	0	-0.19	0.65	0.71	None
12	0	0	0	0	0	-0.19	0.65	0.71	None
13	0	0	0	0	0	-0.19	0.65	0.72	None
14	0	0	0	0	0	-0.19	0.65	0.71	None
15	0	0	0	0	0	-0.19	0.65	0.72	None
16	0	0	0	0	0	-0.2	0.65	0.71	None
17	0	0	0	0	0	-0.19	0.66	0.71	None
18	0	2	1	7	0	-0.19	0.65	0.71	None
19	0	0	0	0	0	-0.19	0.65	0.71	None
20	0	0	1	0	0	-0.18	0.65	0.71	None

FIG. 4.11 : Dataset created from Logged output on file

4. ML Software Implementation

Due to the simplicity of the dataset, this project utilizes TinyML with a classical machine learning algorithm known as a Neural Network (RNN), rather than the more robust deep learning frameworks such as TFLite for Microcontrollers. One of the key advantages of this approach is its lightweight nature, making it suitable for embedded devices. Moreover, it offers the benefit of easy comprehension and does not require off-device training or the use of additional tools.

4.6 The First Implementation

The implementation of the project using Colab involved several steps to prepare the data, build and train the neural network model, predict gestures, and convert the trained model to TensorFlow Lite for deployment on the Arduino Nano 33 BLE Sense. Here is a breakdown of the implementation process :

1. Parse and prepare the data

First, prepare flex sensor data for left and right hands by cleaning, removing outliers, and formatting it for neural network training.

2. Number of recordings for each gesture

The dataset was analyzed to determine recordings for each gesture in left and right hand datasets, identifying imbalances and biases, and understanding data distribution.

3. Randomize and split the input and output pairs for training

A balanced training dataset was created by randomly splitting input and output pairs into three sets : 60 % for training, 20% for validation, and 20% for testing. This allowed for model evaluation and performance assessment.

4. Build and train the neural network model

A TensorFlow neural network model was trained using a training dataset, with architecture and parameters defined. The model's performance was optimized iteratively by adjusting weights and biases based on the data.

5. Prediction of gestures

The trained model predicted gestures using new input data and evaluated its accuracy and effectiveness in recognizing ASL gestures for left and right hands in validation and testing datasets.

6. Repeated steps for the right hand

Repeated data preparation, model building, and training steps were applied to the right hand dataset to develop a unique model for recognizing gestures.

7. Convert the Trained Model to TensorFlow Lite

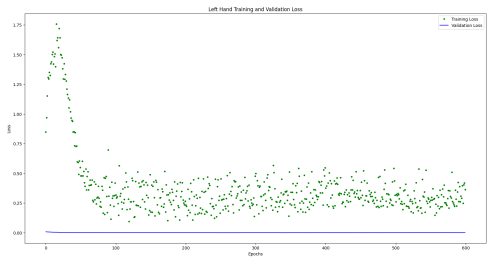
The trained model was converted to TensorFlow Lite format for Arduino Nano 33 BLE Sense deployment, optimizing it for embedded systems and ensuring compatibility with the Arduino platform.

8. Encode the Model in an Arduino Header File

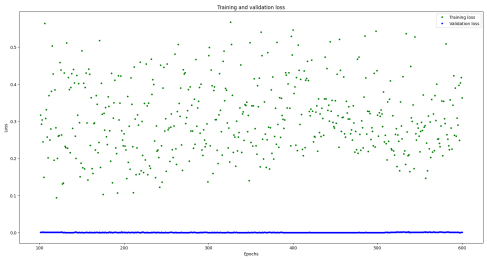
The TensorFlow Lite model was encoded into an Arduino header file, containing information and functions for loading and executing on the Arduino Nano 33 BLE Sense.

4.6.1 First Testing and Results

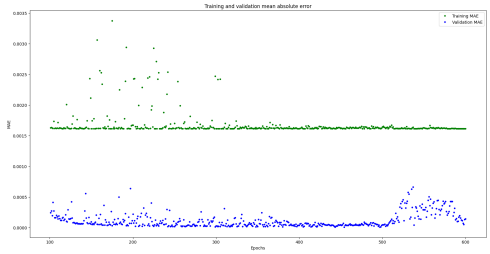
In the testing phase using Colab, two separate datasets for the left and right hand were utilized due to their complexity. It was observed that the model for the left hand did not learn effectively, as it provided the same gesture prediction for all inputs during



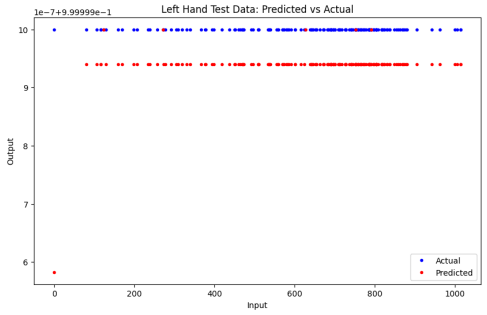
(a) Left Hand Training and Validation Loss



(b) Training and validation loss



(c) Training and validation mean absolute error



(d) Left Hand Test Data Predicted vs Actual

FIG. 4.12 : Training and Validation Loss, Mean Absolute Error, and Predicted vs. Actual Results for Left Hand Gesture Recognition

the prediction phase. This issue can be attributed to the dataset itself, as during the data collection process, there were instances of very large and illogical values.

On the other hand, the model for the right hand performed better in terms of gesture prediction. This can be attributed to the presence of logical values in the dataset for the right hand, which allowed the model to learn and generalize more effectively.

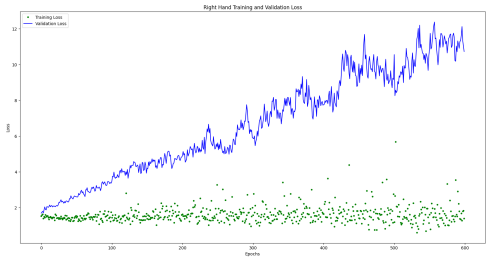
These findings indicate the significance of having a well-prepared and logical dataset for training machine learning models. The quality and consistency of the dataset play a crucial role in the performance and accuracy of the model. In future iterations, efforts should be directed towards collecting a more refined and logically consistent dataset to improve the performance of the left hand model.

It is important to note that the limitations in the dataset and subsequent performance discrepancies between the left and right hand models highlight the challenges faced during the data collection phase. However, these observations provide valuable insights for future improvements and iterations in the development of the Arabic Sign Language recognition system using the Arduino Nano 33 BLE Sense and flex sensors.

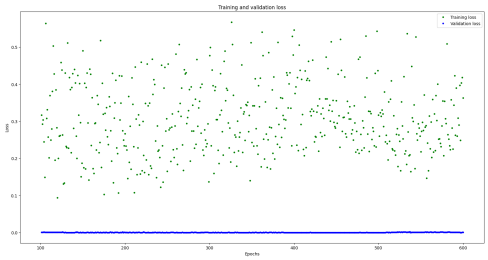
4.7 The second Implementation

In this section, we will discuss the utilization of Edge Impulse for our project.

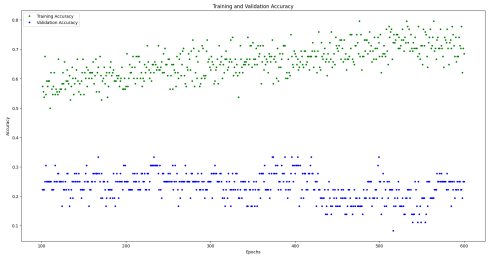
Edge Impulse is a platform specifically designed for developing and deploying machine



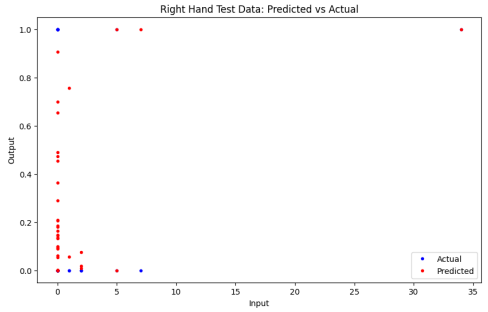
(a) Right Hand Training and Validation Loss



(b) Training and validation loss



(c) Training and Validation Accuracy



(d) Right Hand Test Data Predicted vs Actual

FIG. 4.13 : Training and Validation Loss, Mean Absolute Error, and Predicted vs. Actual Results for Right Hand Gesture Recognition

learning models on embedded systems. Its focus on embedded systems aligns perfectly with our goal of implementing Arabic Sign Language recognition on the Arduino Nano 33 BLE Sense.

We conducted a series of experiments using Edge Impulse, each utilizing a different dataset and approach.

In the first experiment 4.14, we used the right hand dataset without data augmentation and without specifying the order of signs within each sentence. The accuracy achieved in this experiment was 3, indicating poor recognition performance.

Next, we proceeded with the combined dataset4.15, which included both left and right hand gestures. In this experiment, we aimed to leverage the additional data to improve the recognition accuracy. However, the results were still unsatisfactory, with an accuracy of 11%.

Realizing the limitations of the combined dataset, we decided to focus solely on the right hand dataset in our third experiment4.16. This time, we introduced a significant improvement by explicitly specifying the order of signs within each sentence. This modification resulted in a considerable accuracy boost, with the recognition accuracy reaching 41.1%.

Building on this progress, we further enhanced the right hand dataset by applying data augmentation techniques. Although we only augmented 10 sentences in this experiment 4.17, the results were promising, with the recognition accuracy reaching 92%.

Additionally, we experimented with a mixed dataset where the order of lines was

randomly shuffled 4.18. In this scenario, the accuracy obtained was 73%, demonstrating the importance of maintaining a logical order of signs within each sentence for accurate recognition.

Furthermore, we explored gesture variation by introducing slight modifications to the bending readings 4.19. This resulted in six different variations of the same gesture. The recognition accuracy for these six cases varied, with an average accuracy of 43%.

These findings underscore the significance of dataset quality, order specification, and augmentation techniques in improving the performance of the Arabic Sign Language recognition system.

4.7.1 Second Testing and Results

By conducting all that's experiments and exploring the capabilities of Edge Impulse, we gained valuable insights into its potential for developing machine learning models for embedded systems. The platform provides a user-friendly interface for training and deploying models, making it accessible to developers with varying levels of expertise.

The results obtained from Edge Impulse confirmed the importance of dataset quality, organization, and augmentation techniques in improving the accuracy of ASL recognition models. It also highlighted the need for careful selection and preprocessing of the data to ensure optimal performance.

Fig. 4.14 : EDGE IMPULSE Analysis for ARSLR (case1)

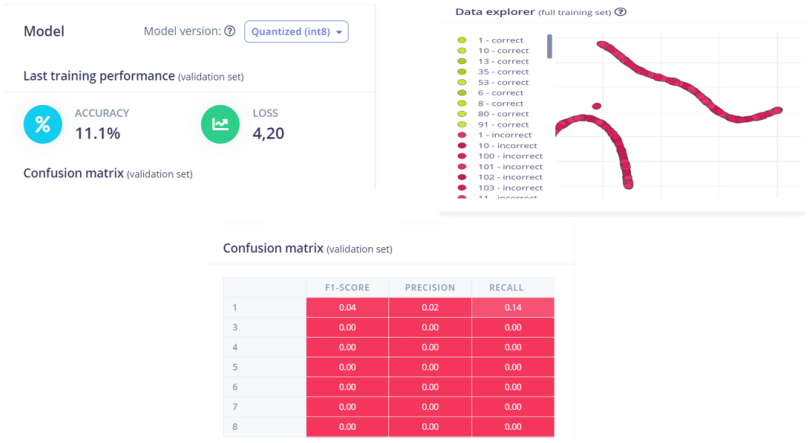


Fig. 4.15 : EDGE IMPULSE Analysis for ARSLR (case2)

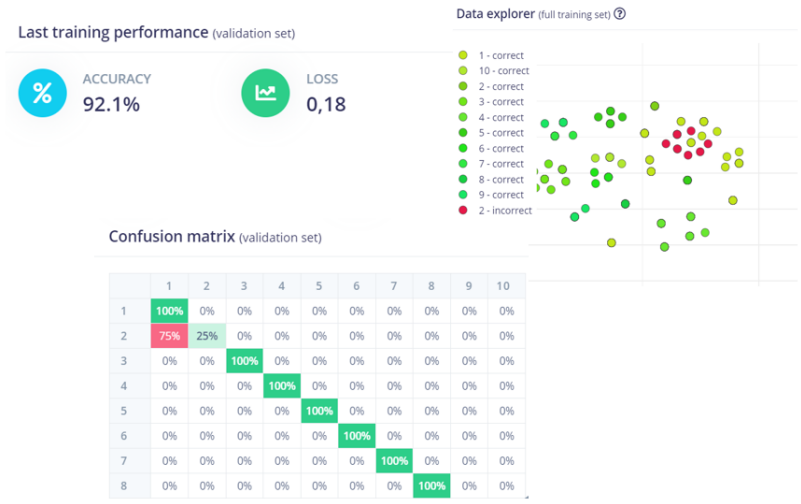


Fig. 4.16 : EDGE IMPULSE Analysis for ARSLR (case3)

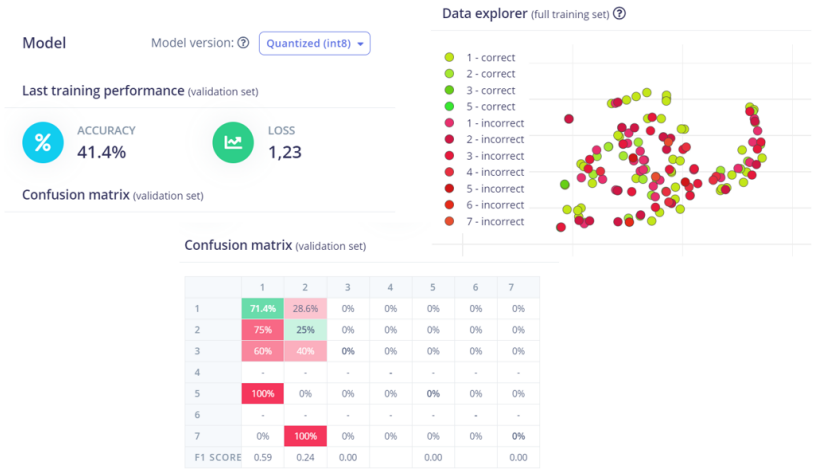


FIG. 4.17 : EDGE IMPULSE Analysis for ARSLR (case4)

FIG. 4.18 : EDGE IMPULSE Analysis for ARSLR (case5)

FIG. 4.19 : EDGE IMPULSE Analysis for ARSLR (case6)

Conclusion

In this chapter, we presented the design and implementation of the project. We discussed the hardware architectural design, software design, and the ML workflow. The implementation phase included both hardware and embedded software implementation.

Overall, this chapter provided a comprehensive overview of the design and implementation process. The testing and results obtained from the implementations serve as valuable feedback for further analysis and enhancement in the subsequent chapters.

Conclusion and Perspectives

Conclusion and future works

This project has focused on the development of a sign language recognition system using the Arduino Nano 33 BLE Sense and flex sensors, with a specific emphasis on Arabic Sign Language (ASL) sentences. The primary objective was to create an affordable, lightweight, and power-efficient solution that enables accurate recognition of ASL gestures using machine learning algorithms.

The contributions that our project was able to make can be summarized as follows :

- We have established a substantial state of the art, which covers all about Arabic sign language recognition using signal processing .
- Development of an affordable and accessible sign language recognition system : By utilizing the Arduino Nano 33 BLE Sense and flex sensors.
- Independent data collection : In order to address the lack of existing data, we took the initiative to collect our own dataset for training and evaluation. This involved capturing an acceptable range of ASL gestures using the gloves prototype. The self-collected data allowed us to create a unique and personalized dataset specifically tailored to our project's objectives.
- The project explores machine learning techniques on edge devices, utilizing technologies like TinyML and TensorFlow Lite for Microcontrollers, to demonstrate gesture recognition feasibility on small embedded devices with limited resources.
- We managed to get close to our topic for the project

Perspectives

Moving forward, there are several key aspects that require further attention and provide opportunities for future development in the field of Arabic Sign Language (ASL) recognition using the Arduino Nano 33 BLE Sense and flex sensors. The challenges encountered during this project, coupled with the time limitations, have shed light on important areas for improvement and exploration :

- Project faced challenges in data collection and labeling due to time constraints. Optimizing processes, using crowdsourcing or ASL experts, can improve dataset quality and representativeness.
- Real-time implementation requires optimization for time constraints, focusing on efficient algorithms, code optimization, and hardware acceleration techniques for near-real-time ASL gesture recognition.
- To improve system performance, consider synchronization techniques and a coordinated approach for Arduino Nano 33 BLE Sense boards within time constraints, utilizing available resources efficiently.

- This project serves as a stepping stone for future advancements, focusing on refining the ASL recognition system, incorporating user feedback, and expanding capabilities through additional sensors and user experiences.

Despite the challenges and time limitations faced during this project, it is important to recognize that the journey does not end here. In fact, reaching this point provides an opportunity for reflection, learning, and potential for further improvement.

A new comprehensive dataset is crucial for the development of an accurate and robust ASL recognition system. Utilizing the project's knowledge and experience, a new data collection phase can be initiated, ensuring a diverse and high-quality dataset. This will address limitations in previous data collection processes and contribute to the overall improvement of the ASL recognition system.

Building upon lessons learned from this project, focusing on continuous improvement and addressing limitations, will lead to a refined and effective ASL recognition solution.

In conclusion, the completion of this project marks an important milestone in the journey towards developing an Arabic Sign Language recognition system using the Arduino Nano 33 BLE Sense and flex sensors. Despite the challenges faced and the time limitations encountered, it is essential to view this as an opportunity for growth and further development. Collecting a new dataset, coupled with the knowledge gained from this project, will enable the creation of an improved ASL recognition system. The commitment to pushing the boundaries and striving for excellence will drive future iterations, leading to a more accurate, efficient, and impactful solution for individuals with hearing disabilities.

References

[1] Nikolas Adaloglou et al. “A comprehensive study on deep learning-based methods for sign language recognition”. In : *IEEE Transactions on Multimedia* 24 (2021), p. 1750-1762.

[2] Abdelmoty M Ahmed et al. “Automatic translation of Arabic sign to Arabic text (ATASAT) system”. In : *Journal of Computer Science and Information Technology* 6 (2016), p. 109-122.

[3] Mohamed Aktham Ahmed et al. “A review on systems-based sensory gloves for sign language recognition state of the art between 2007 and 2017”. In : *Sensors* 18.7 (2018), p. 2208.

[4] Omar Amin et al. “HMM based automatic Arabic sign language translator using Kinect”. In : *2015 Tenth International Conference on Computer Engineering & Systems (ICCES)*. IEEE. 2015, p. 389-392.

[5] Tamás Aujeszky et Mohamad Eid. “A gesture recogintion architecture for Arabic sign language communication system”. In : *Multimedia Tools and Applications* 75 (2016), p. 8493-8511.

[6] Ashish G Bairagi. “Y.. Kapse,“Survey on Sign language to Speech Conversion,”” in : *Int. J. Innov. Res. Comput. Commun. Eng* 6.1 (2018), p. 267-274.

[7] Noel C Baker et Patrick C Taylor. “A framework for evaluating climate model performance metrics”. In : *Journal of Climate* 29.5 (2016), p. 1773-1782.

[8] Salar Basiri et al. “Dynamic iranian sign language recognition using an optimized deep neural network : an implementation via a robotic-based architecture”. In : *International Journal of Social Robotics* 15.4 (2023), p. 599-619.

[9] Sara Bilal et al. “Hidden Markov model for human to computer interaction : a study on human hand gesture recognition”. In : *Artificial Intelligence Review* 40 (2013), p. 495-516.

[10] Christopher JC Burges. “A tutorial on support vector machines for pattern recognition”. In : *Data mining and knowledge discovery* 2.2 (1998), p. 121-167.

[11] Ankit Chaudhary et al. “A survey on hand gesture recognition in context of soft computing”. In : *Advanced Computing : First International Conference on Computer Science and Information Technology, CCSIT 2011, Bangalore, India, January 2-4, 2011. Proceedings, Part III 1*. Springer. 2011, p. 46-55.

[12] Disi Chen et al. “An interactive image segmentation method in hand gesture recognition”. In : *Sensors* 17.2 (2017), p. 253.

[13] Jun Cheng et al. “Feature fusion for 3D hand gesture recognition by learning a shared hidden space”. In : *Pattern Recognition Letters* 33.4 (2012), p. 476-484.

[14] Ming Jin Cheok, Zaid Omar et Mohamed Hisham Jaward. “A review of hand gesture and sign language recognition techniques”. In : *International Journal of Machine Learning and Cybernetics* 10 (2019), p. 131-153.

[15] Hyo-Rim Choi et TaeYong Kim. “Combined dynamic time warping with multiple sensors for 3D gesture recognition”. In : *Sensors* 17.8 (2017), p. 1893.

[16] Ryszard S Choras. “Hand shape and hand gesture recognition”. In : *2009 IEEE Symposium on Industrial Electronics&Applications*. 2009.

[17] Tushar Chouhan et al. “Smart glove with gesture recognition ability for the hearing and speech impaired”. In : *2014 IEEE Global Humanitarian Technology Conference-South Asia Satellite (GHTC-SAS)*. IEEE. 2014, p. 105-110.

[18] Helen Cooper, Brian Holt et Richard Bowden. “Sign language recognition”. In : *Visual Analysis of Humans : Looking at People* (2011), p. 539-562.

[19] Robert David et al. “Tensorflow lite micro : Embedded machine learning for tinyml systems”. In : *Proceedings of Machine Learning and Systems* 3 (2021), p. 800-811.

[20] Konstantinos G Derpanis. “Mean shift clustering”. In : *Lecture Notes* 32 (2005), p. 1-4.

[21] A Directions. “Principal Component Analysis (PCA) Principal Component Analysis (PCA)”. In : *Statistics, June* (2007), p. 1-12.

[22] Cao Dong, Ming C Leu et Zhaozheng Yin. “American sign language alphabet recognition using microsoft kinect”. In : *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2015, p. 44-52.

[23] Mohammed Elmahgiubi et al. “Sign language translator and gesture recognition”. In : *2015 Global Summit on Computer & Information Technology (GSCIT)*. IEEE. 2015, p. 1-6.

[24] Maurizio Di Paolo Emilio. *Embedded systems design for high-speed data acquisition and control*. Springer, 2015.

[25] Kevin Forsberg et Harold Mooz. “The relationship of system engineering to the project cycle”. In : *INCOSE international symposium*. T. 1. 1. Wiley Online Library. 1991, p. 57-65.

[26] Marguerite Foxon. “Evaluation of training and development programs : A review of the literature”. In : *Australasian Journal of Educational Technology* 5.2 (1989).

[27] Xiaopei Guo et al. “A novel method for data glove-based dynamic gesture recognition”. In : *2017 International conference on virtual reality and visualization (ICVRV)*. IEEE. 2017, p. 43-48.

[28] Mohamed Hassan, Khaled Assaleh et Tamer Shanableh. “Multiple proposals for continuous arabic sign language recognition”. In : *Sensing and Imaging* 20 (2019), p. 1-23.

[29] Omayya M Al-Hassan, Kamal E Bani-Hani et Mu’tasem M Al-Masa’deh. “Sign Language as a Means of Inclusion : A Case Study”. In : *International Journal of Disability, Development and Education* (2023), p. 1-15.

[30] E Llobet Hines, E Llobet et JW Gardner. “Electronic noses : a review of signal processing techniques”. In : *IEE Proceedings-Circuits, Devices and Systems* 146.6 (1999), p. 297-310.

[31] Gerard J Holzmann et William Slattery Lieberman. *Design and validation of computer protocols*. T. 512. Prentice hall Englewood Cliffs, 1991.

[32] Volker Hovestadt et al. “Resolving medulloblastoma cellular architecture by single-cell genomics”. In : *Nature* 572.7767 (2019), p. 74-79.

[33] Nada B Ibrahim, Mazen M Selim et Hala H Zayed. “An automatic Arabic sign language recognition system (ArSLRS)”. In : *Journal of King Saud University-Computer and Information Sciences* 30.4 (2018), p. 470-477.

[34] Nada B Ibrahim, Hala H Zayed et Mazen M Selim. “Advances, challenges and opportunities in continuous sign language recognition”. In : *Journal of Engineering and Applied Sciences* 15.5 (2020), p. 1205-1227.

[35] Dino Ienco et al. “Land cover classification via multitemporal spatial data by deep recurrent neural networks”. In : *IEEE Geoscience and Remote Sensing Letters* 14.10 (2017), p. 1685-1689.

[36] Anil K Jain. “Data clustering : 50 years beyond K-means”. In : *Pattern recognition letters* 31.8 (2010), p. 651-666.

[37] Xianwei Jiang et al. “A survey on artificial intelligence in Chinese sign language recognition”. In : *Arabian Journal for Science and Engineering* 45 (2020), p. 9859-9894.

[38] Boban Joksimoski et al. “Technological solutions for sign language recognition : a scoping review of research trends, challenges, and opportunities”. In : *IEEE Access* 10 (2022), p. 40979-40998.

[39] Mert Kalfa et al. “Reliable extraction of semantic information and rate of innovation estimation for graph signals”. In : *IEEE Journal on Selected Areas in Communications* 41.1 (2022), p. 119-140.

[40] Tapas Kanungo et al. “An efficient k-means clustering algorithm : Analysis and implementation”. In : *IEEE transactions on pattern analysis and machine intelligence* 24.7 (2002), p. 881-892.

[41] Alboukadel Kassambara. *Practical guide to cluster analysis in R : Unsupervised machine learning*. T. 1. Sthda, 2017.

[42] Sumaira Kausar et M Younus Javed. “A survey on sign language recognition”. In : *2011 Frontiers of Information Technology*. IEEE. 2011, p. 95-98.

[43] Arlene Kelly. “A Brief History on the Field of Deaf Studies (1)”. In : (1998).

[44] Rafiqul Zaman Khan et Noor Adnan Ibraheem. “Hand gesture recognition : a literature review”. In : *International journal of artificial Intelligence & Applications* 3.4 (2012), p. 161.

[45] Agus Kurniawan et Agus Kurniawan. “Arduino nano 33 ble sense board development”. In : *IoT Projects with Arduino Nano 33 BLE Sense : Step-By-Step Projects for Beginners* (2021), p. 21-74.

[46] Alina Kuznetsova, Laura Leal-Taixé et Bodo Rosenhahn. "Real-time sign language recognition using a consumer depth camera". In : *Proceedings of the IEEE international conference on computer vision workshops*. 2013, p. 83-90.

[47] John Lafferty, Andrew McCallum et Fernando CN Pereira. "Conditional random fields : Probabilistic models for segmenting and labeling sequence data". In : (2001).

[48] Daniel T Larose et Chantal D Larose. "k-nearest neighbor algorithm". In : (2014).

[49] Minhyuk Lee et Joonbum Bae. "Real-time gesture recognition in the view of repeating characteristics of sign languages". In : *IEEE Transactions on Industrial Informatics* 18.12 (2022), p. 8818-8828.

[50] Yanli Liu, Yourong Wang et Jian Zhang. "New machine learning algorithm : Random forest". In : *Information Computing and Applications : Third International Conference, ICICA 2012, Chengde, China, September 14-16, 2012. Proceedings 3*. Springer. 2012, p. 246-252.

[51] Stephan Liwicki et Mark Everingham. "Automatic recognition of fingerspelled words in british sign language". In : *2009 IEEE computer society conference on computer vision and pattern recognition workshops*. IEEE. 2009, p. 50-57.

[52] Mayyadah R Mahmood et Adnan M Abdulazeez. "A Comparative study of a new hand recognition model based on line of features and other techniques". In : *Recent Trends in Information and Communication Technology : Proceedings of the 2nd International Conference of Reliable Information and Communication Technology (IRICT 2017)*. Springer. 2018, p. 420-432.

[53] Syed Atif Mehdi et Yasir Niaz Khan. "Sign language recognition using sensor gloves". In : *Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP'02.* T. 5. IEEE. 2002, p. 2204-2206.

[54] Noraini Mohamed, Mumtaz Begum Mustafa et Nazean Jomhari. "A review of the hand gesture recognition system : Current progress and future directions". In : *IEEE Access* 9 (2021), p. 157422-157436.

[55] GRS Murthy et RS Jadon. "A review of vision based hand gestures recognition". In : *International Journal of Information Technology and Knowledge Management* 2.2 (2009), p. 405-410.

[56] Muhammad Waqas Nadeem et al. "Deep learning for diabetic retinopathy analysis : A review, research challenges, and future directions". In : *Sensors* 22.18 (2022), p. 6780.

[57] William S Noble. "What is a support vector machine ?" In : *Nature biotechnology* 24.12 (2006), p. 1565-1567.

[58] World Health Organization et al. "Deafness and hearing loss. 2019". In : *URL : https ://www. who. int/news-room/fact-sheets/detail/deafness-and-hearing-loss* (2020).

[59] Pratibha Pandey et Vinay Jain. "An efficient algorithm for sign language recognition". In : *computer* 6 (2015), p. 7.

[60] Theofilos Papadopoulos et al. "Interactions in augmented and mixed reality : An overview". In : *Applied Sciences* 11.18 (2021), p. 8752.

[61] Paul Kang-Hoh Phua et Daohua Ming. “Parallel nonlinear optimization techniques for training neural networks”. In : *IEEE Transactions on Neural Networks* 14.6 (2003), p. 1460-1468.

[62] Pramod Kumar Pisharady et Martin Saerbeck. “Recent methods and databases in vision-based hand gesture recognition : A review”. In : *Computer Vision and Image Understanding* 141 (2015), p. 152-165.

[63] Paul Ralph et Yair Wand. “A proposal for a formal definition of the design concept”. In : *Design Requirements Engineering : A Ten-Year Perspective : Design Requirements Workshop, Cleveland, OH, USA, June 3-6, 2007, Revised and Invited Papers*. Springer. 2009, p. 103-136.

[64] Daniel Ramage. “Hidden Markov models fundamentals”. In : *CS229 Section Notes* 1 (2007).

[65] Siddharth S Rautaray et Anupam Agrawal. “Vision based hand gesture recognition for human computer interaction : a survey”. In : *Artificial intelligence review* 43 (2015), p. 1-54.

[66] J Rekha, J Bhattacharya et S Majumder. “Hand gesture recognition for sign language : A new hybrid approach”. In : *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*. The Steering Committee of The World Congress in Computer Science, Computer … 2011, p. 1.

[67] Simon Ruffieux, Denis Lalanne et Elena Mugellini. “ChAirGest : a challenge for multimodal mid-air gesture recognition for close HCI”. In : *Proceedings of the 15th ACM on International conference on multimodal interaction*. 2013, p. 483-488.

[68] Zinah Raad Saeed et al. “A Systematic Review on Systems-Based Sensory Gloves for Sign Language Pattern Recognition : An Update From 2017 to 2022”. In : *IEEE Access* (2022).

[69] K Martin Sagayam et D Jude Hemanth. “Hand posture and gesture recognition techniques for virtual reality applications : a survey”. In : *Virtual Reality* 21 (2017), p. 91-107.

[70] Giovanni Saggio et al. “Dynamic Measurement Assessments of Sensory Gloves Based on Resistive Flex Sensors and Inertial Measurement Units”. In : *IEEE Transactions on Instrumentation and Measurement* (2023).

[71] Ozan Saltuk et Ismail Kosan. “Design and creation”. In : *Ludwig Maximilians Universitat Munchen* (2014).

[72] Nazmus Saquib et Ashikur Rahman. “Application of machine learning techniques for real-time sign language detection using wearable sensors”. In : *Proceedings of the 11th ACM Multimedia Systems Conference*. 2020, p. 178-189.

[73] Arpita Ray Sarkar, G Sanyal et SJIJOCA Majumder. “Hand gesture recognition systems : a survey”. In : *International Journal of Computer Applications* 71.15 (2013).

[74] Pavel Senin. "Dynamic time warping algorithm review". In : *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA* 855.1-23 (2008), p. 40.

[75] Ahmad Sami Al-Shamayleh et al. "A systematic literature review on vision based gesture recognition techniques". In : *Multimedia Tools and Applications* 77 (2018), p. 28121-28184.

[76] Ahmad Sami Al-Shamayleh et al. "Automatic Arabic sign language recognition : A review, taxonomy, open challenges, research roadmap and future directions". In : *Malaysian Journal of Computer Science* 33.4 (2020), p. 306-343.

[77] Rajat Sharma, Ravi Khapra et Neeraj Dahiya. "Sign language gesture recognition". In : *International Journal of Recent Research Aspects* 7.2 (2020), p. 14-19.

[78] Gamal Tharwat, Abdelmoty M Ahmed et Belgacem Bouallegue. "Arabic sign language recognition system for alphabets using machine learning techniques". In : *Journal of Electrical and Computer Engineering* 2021 (2021), p. 1-17.

[79] Saravanan Thirumuruganathan. "A detailed introduction to K-nearest neighbor (KNN) algorithm". In : *Retrieved March* 20 (2010), p. 2012.

[80] Mohamed Fahmy Tolba, Ahmed Samir et Magdy Aboul-Ela. "Arabic sign language continuous sentences recognition using PCNN and graph matching". In : *Neural Computing and Applications* 23 (2013), p. 999-1010.

[81] Cuong Tran et Mohan Manubhai Trivedi. "3-D posture and gesture recognition for interactivity in smart spaces". In : *IEEE Transactions on Industrial Informatics* 8.1 (2011), p. 178-187.

[82] Noor Tubaiz, Tamer Shanableh et Khaled Assaleh. "Glove-based continuous Arabic sign language recognition in user-dependent mode". In : *IEEE Transactions on Human-Machine Systems* 45.4 (2015), p. 526-533.

[83] Maryam Vafadar et Alireza Behrad. "A vision based system for communicating in virtual reality environments by recognizing human hand gestures". In : *Multimedia Tools and Applications* 74 (2015), p. 7515-7535.

[84] Peter Vamplew et Anthony Adams. "Recognition of sign language gestures using neural networks". In : *European Conference on Disabilities, Virtual Reality and Associated Technologies*. 1996.

[85] Christian Wohler et Joachim K Anlauf. "An adaptable time-delay neural-network algorithm for image sequence analysis". In : *IEEE Transactions on Neural Networks* 10.6 (1999), p. 1531-1536.

[86] Ying Wu et Thomas S Huang. "Hand modeling, analysis and recognition". In : *IEEE Signal Processing Magazine* 18.3 (2001), p. 51-60.

[87] Yong Yu et al. "A review of recurrent neural networks : LSTM cells and network architectures". In : *Neural computation* 31.7 (2019), p. 1235-1270.