TALLINN UNIVERSITY OF TECHNOLOGY

SCHOOL OF ENGINEERING

Department of Electrical Power Engineering and Mechatronics

LOW COST AIOT SYSTEM FOR HUMAN MOTION DEVIATION MONITORING

KULUEFEKTIIVNE TEHISINTELLEKTI TOEGA ASJADE INTERNETI SÜSTEEM LIIKUMISE KÕRVALEKALLETE JÄLGIMISEKS

MASTER THESIS

Student:	Eshgin Guluzade
Student code:	212298MAHM
Supervisor:	Alar Kuusik, PhD, Senior research scientist
Co-supervisor:	Muhammad Usman Naseer, Early-Stage researcher

Tallinn 2024

AUTHOR’S DECLARATION

Hereby I declare, that I have written this thesis independently.

No academic degree has been applied for based on this material. All works, major viewpoints and data of the other authors used in this thesis have been referenced.

“13” May 2024

Author: Eshgin Guluzade

Thesis is in accordance with terms and requirements

“13” May 2024

Supervisor: Alar Kuusik

Accepted for defence

“.......”. .... 20… .

Chairman of theses defence commission: ....

/name /

[1] *The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the University's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.*

Department of Electrical Power Engineering and Mechatronics

THESIS TASK

Student: Eshgin Guluzade, 212298

Study programme: MAHM02/18

Main speciality: Mechatronics

Supervisor(s): Senior research scientist, Alar Kuusik, 620 2166;

Early-Stage researcher, Muhammad Usman Naseer

Thesis topic:

(in English) Low Cost AIoT System for Human Motion Deviation Monitoring

(in Estonian) Kuluefektiivne tehisintellekti toega asjade interneti süsteem liikumise kõrvalekallete jälgimiseks

Thesis main objectives:

1. Collecting IMU data for steppage gait analysis
2. Training of 1D CNN model
3. Implementing 1D CNN model on low-cost, low-power devices, specifically ESP32

Thesis tasks and time schedule:

No	Task description	Deadline
1.	Literature Review	15.02.2024
2.	Adaptation of Existing Algorithm for Real-Time Processing	05.03.2024
3.	Integration the software with ESP32, 30% of thesis textual content prepared	20.03.2024
4.	Training and Optimization of Adapted Algorithm developed	05.04.2024
5.	Real-World Testing completed, 70% of text prepared	20.04.2024
6.	Final Thesis Submission	13.05.2024

Language: English **Deadline for submission of thesis:** "13" May 2024

Student: Eshgin Guluzade .... "......."............................ 20....a

/signature/

Supervisor: Alar Kuusik .... "......."....................20....a

/signature/

Consultant: Muhammad Usman Naseer .... "......."......................20....a

/signature/

Head of study programme: Anton Rassõlkin …............ “.......”.....................20….a

/signature/

TABLE OF CONTENTS

PREFACE

This master thesis has been supported by the project „Increasing the knowledge intensity of Ida-Viru entrepreneurship" („AIoT*5G - Artificial intelligence, edge computing and IoT solutions in distributed systems") co-funded by the European Union.

The thesis examines the use of 1D Convolutional Neural Networks (1D CNNs) on the ESP32 microcontroller—a low-power, cost-effective device—for analyzing gait disorders prevalent in older adults and individuals with neurological conditions. By integrating the ESP32-D0WDQ6 with the MPU6050 sensor modules, the study demonstrates the platform's capability for real-time gait analysis, specifically in identifying and classifying abnormalities like steppage gait.

Despite the limitations posed by resource-constrained hardware, this study shows that even small, efficiently designed 1D CNNs can offer valuable insights into gait disorders. However, the research also identifies the importance of tailoring model architectures specifically for low-power devices. Future work should focus on optimizing the architecture through techniques such as pruning and quantization, improving memory management, and supporting parallel processing across the ESP32's dual-core architecture.

This work establishes a benchmark for deploying machine learning models on the ESP32 platform, demonstrating that with proper model design, low-power devices can be leveraged effectively for advanced gait analysis for clinical, wellness and occupational use.

I would like to express my gratitude to Alar Kuusik and Muhammad Usman Naseer for their unwavering, constant support and guidance throughout my thesis work. Special thanks to Jakob Rostovski, a PhD student at Tallinn University of Technology, for his willingness to share his own research insights and constructive feedback during the project.

LIST OF ABBREVIATIONS AND SYMBOLS

AI – Artificial Intelligence

ADC - Analog-to-Digital Converter

BHI - Biomedical and Health Informatics

BLE - Bluetooth Low Energy

CNN - Convolutional Neural Network

CSV - Comma-Separated Values

CSV - Comma-Separated Values

ECG – Electrocardiogram

EEG – Electroencephalogram

ESP-IDF - Espressif IoT Development Framework

FES - Functional Electrical Stimulation

GPIO - General Purpose Input/Output

IDE - Integrated Development Environment

I2C - Inter-Integrated Circuit

MAC - Medium Access Control

ML - Machine Learning

NN - Neural Network

PSRAM - Pseudo-Static Random-Access Memory

ReLU - Rectified Linear Unit

RNN - Recurrent Neural Network

SPI - Serial Peripheral Interface

SRAM - Static Random-Access Memory

SVM - Support Vector Machine

TFLite - TensorFlow Lite

USB - Universal Serial Bus

WBAN - Wireless Body Area Network

1. INTRODUCTION

Gait disorders, characterized by abnormal patterns in walking, are common in the aging population and in individuals with specific neurological disorders. Gait disorders and deviations are also risk factors for occupational safety. These disorders pose significant risks for dependence, cognitive decline, falls, and even mortality. Studies indicate that after the age of 70, approximately 35% of people exhibit some form of gait abnormality, a figure that rises substantially in those over 85 [1]. The manifestation of these disorders, such as reduced speed and stride length, often suggests underlying pathologies, both in the nervous and non-nervous systems.

A typical gait cycle, initiating with the contact of one foot and ending with the contact of the other, consists of stance and swing phases [2]. Aging affects various elements of the gait cycle: after 70 years, walking speed declines by about 15% per decade, and faster walking sees an even sharper decline [3]. Slow walking speed is a strong predictor of mortality, exceeding even the impact of chronic conditions and hospitalizations in older adults. The muscle weakness, particularly in the calves, contributes significantly to this decline. However, older adults often compensate for this weakness through the use of hip flexors and extensor muscles [4].

The double stance time, a phase where both feet are in contact with the ground, increases with age, accounting for up to 26% of the gait cycle in older individuals [5]. This increase affects stride length and may be more pronounced on uneven surfaces or in fall-prone situations. Changes in walking posture are also notable; older individuals tend to walk with greater anterior pelvic rotation and without forward lean. Contributing factors include increased abdominal fat, weakened abdominal muscles, and rigidity in hip flexors. The expected result of the research is to develop a automatic personalized multi- sensor device which can predict falls with more than 90% accuracy in different environmental conditions and contexts for user mobile applications.

Gait disorders are particularly prevalent in patients with neurological problems, significantly increasing their risk of falls [6]. Understanding and addressing these disorders is essential for fall prevention and enhancing overall mobility in the aging population. Technological advancements like accelerometers for step counting and wearable sensors have greatly improved the assessment and analysis of gait outside traditional laboratory settings, allowing for more accurate and natural observation of gait in everyday environments [7].

In conclusion, gait disorders in older adults and those with neurological conditions are a critical public health concern. Continuous efforts in research, technological innovation, and community design are vital to mitigate their impacts and improve quality of life. In this work, we address these gaps by investigating the feasibility of deploying low-cost and energy efficient AIoT solutions for human motion deviation monitoring. Through extensive testing, we highlight key challenges encountered during the deployment of 1D-CNN models. These contributions will pave the way for more advanced AIoT systems, demonstrating that low-power devices can be leveraged for running ML models when model architectures are appropriately tailored. The proposed approach serves as a benchmark for deploying machine learning on low-power devices.

2. LITERATURE REVIEW

2.1 Gait disorders and patterns

Gait analysis, a critical process in assessing and understanding human locomotion, plays a vital role in healthcare for diagnosing and monitoring gait disorders. Its importance is particularly pronounced in identifying early signs of health issues in the aging population and individuals with neurological conditions [8]. Gait patterns vary significantly, each characterized by unique features that can indicate specific health conditions or functional impairments. Understanding these various gait types is essential in clinical diagnostics and research [9].

Table 1 shows a phenomenological classification of common gait disorders [10].

Table 1. Phenomenological classification of gait disorders (modified from [11])

Type of Gait Abnormality	**Descriptive Characteristics**
Hemispastic gait	One-sided body extension movements
Paraspastic gait	Bilateral stiffness with outward leg movement
Ataxic gait	Wide stance, disordered coordination
Sensory ataxic gait	Worsens without visual cues, cautious
Freezing gait	Sudden halting, especially during turns
Astasia gait	Disturbance in balance while standing
Cautious gait	Broad based, cautious, slow, anxious
Propulsive gait	Forward center of gravity with rapid, short steps
Steppage gait	Elevated thigh lift due to foot muscle weakness
Antalgic gait	Reduced weight-bearing phase due to pain
Choreatic gait	Dance-like movements, lack of stability
Dystonic gait	Foot or leg adopts unusual posture
Waddling gait	Duck-like walking, lateral movement
Vertiginous gait	Unsteady, tilting to one side
Psychogenic gait	Bizarre, atypical gait with infrequent falls

Each gait type offers insights into underlying physiological or neurological issues, making their identification and analysis crucial in healthcare settings. Particularly, the analysis of abnormal gaits like steppage gait can reveal significant information about neuromuscular health and guide treatment decisions.

Steppage gait is a distinctive walking pattern that arises primarily from weakness or paralysis of the dorsiflexor muscles of the foot, which include the tibialis anterior, extensor hallucis longus, and extensor digitorum longus. Individuals with steppage gait exhibit an exaggerated lifting of the knee and hip during walking, as if stepping over an obstacle on the ground. This compensatory action is necessary to prevent the toes from dragging along the ground, which occurs due to the inability to properly flex the ankle and lift the foot (foot drop).

The most common underlying causes of steppage gait include peripheral neuropathies, such as those associated with diabetes mellitus or trauma to the peroneal nerve. Additionally, steppage gait can be seen in neurodegenerative conditions such as Charcot-Marie-Tooth disease, where progressive loss of muscle tissue and touch sensation occurs in the feet and legs [2].

Clinically, steppage gait can be identified by observing the patient walk. The hallmarks include a foot that hangs with the toes pointing down, causing the toes to scrape the ground when walking, and a gait that involves lifting the leg higher than normal at the hip. The sound of foot scuffing or seeing wear on the tip of the shoe may be a first clue. To confirm the diagnosis and assess the severity, gait analysis technologies, including pressure mats, motion capture systems, or electromyography, are utilized to measure and record the walking patterns, muscular activity, and compensatory mechanisms used by the patient.

In the treatment and management of steppage gait, rehabilitation exercises, physical therapy, and sometimes surgical interventions are employed. A key component of therapy is strengthening the muscles responsible for dorsiflexion of the foot and employing orthotic devices to support the foot and improve walking [8]. Recent advancements in functional electrical stimulation (FES) have shown promise in providing temporary improvement in dorsiflexor strength during walking, thus improving the gait pattern in real-time.

2.2 Gait and motion monitoring technologies

The domain of gait and motion monitoring has evolved dramatically with advancements in technology, offering profound benefits for diagnosing and managing gait disorders such as steppage gait. Traditional gait analysis relied on visual observation, often with video support, to assess the biomechanics of gait. While effective for basic assessments, these methods were confined to laboratory settings and lacked the ability to capture the dynamism of everyday movements [8]. Advancements brought about sophisticated motion capture systems that use markers to track the three-dimensional movement of the body. These systems provide detailed analyses of gait kinematics and kinetics but require controlled environments and specialized equipment [9]. The proliferation of wearable sensor technology has democratized gait analysis. Devices like accelerometers and gyroscopes allow for the real-time, naturalistic observation of gait, enabling long-term monitoring outside clinical settings [12]. Smartphones and wearable devices have integrated gait analysis capabilities, featuring built-in sensors and applications that process and interpret motion data. This has facilitated remote monitoring and tele-rehabilitation, expanding the reach of gait analysis to everyday settings [13]. In clinical practice, gait analysis technologies have become essential tools for customizing treatment and monitoring rehabilitation. For patients with disorders like steppage gait, these technologies help in creating more effective, patient-specific therapeutic interventions [9]. The following table 2 summarizes the various gait and motion monitoring technologies.

Table 2. Summary of Gait and Motion Monitoring Technologies

Technology Category	**Description**	**Benefits**	**Limitations**	**Applications**
Early Systems	Visual observation and video recording; use of force platforms for kinetic analysis.	Provides foundational gait data; good for initial assessments.	Limited to labs; static data collection.	Baseline measurements; biomechanical studies.
Motion Capture	Marker-based systems tracking 3D body movements; detailed kinematic and kinetic analysis.	High detail and precision.	Requires controlled environment; costly setup.	Advanced gait research; detailed clinical diagnostics.
Wearable Sensors	Portable devices measuring movement; includes accelerometers, gyroscopes, magnetometers	Real-time data; naturalistic settings; continuous monitoring.	Potentially less precise; data volume management.	Daily monitoring; personalized rehab programs.
Clinical Practice	Application of gait analysis for therapeutic evaluation and management.	Tailors treatments; objective progress measurement.	Requires interpretation expertise.	Treatment customization; rehabilitation tracking.

By leveraging these diverse technologies, clinicians and researchers can gather extensive data on gait patterns, offering critical insights into disorders like steppage gait and informing more effective treatment protocols.

2.3 Machine Learning in gait analysis

Machine learning (ML) has emerged as a transformative force in gait analysis, offering powerful tools to interpret complex data and extract meaningful patterns. Machine learning, a subset of artificial intelligence (AI), involves the use of statistical techniques to enable computers to 'learn' from and make predictions or decisions based on data. In gait analysis, ML algorithms process large datasets collected from sensors to identify and classify gait patterns, detect abnormalities, and predict outcomes [14].

ML algorithms are particularly beneficial in diagnosing gait disorders, assessing the risk of falls, and monitoring rehabilitation progress. By learning from data gathered by gait analysis tools, these algorithms can differentiate between various gait types and pinpoint deviations from normal patterns that may indicate a disorder or risk of injury [15]. Convolutional Neural Networks, a class of deep neural networks, are particularly adept at processing data with a grid-like topology, such as time-series data from sensors. CNNs have shown promise in identifying subtle patterns in gait that may not be discernible through traditional analysis methods. For example, a CNN can learn to recognize the characteristic footfall patterns of individuals with steppage gait, aiding in early detection and intervention [16].

The efficacy of ML models in gait analysis is quantified using metrics such as accuracy, precision, sensitivity (recall), specificity, and the F1 score. In [17] research showcased the ability of Support Vector Machines (SVM) to classify gait data with high precision and recall, underlining the potential of ML in identifying gait abnormalities. Recurrent Neural Networks (RNNs), and in particular, Long Short-Term Memory (LSTM) networks, excel at analyzing time-series data and have been shown to be effective in capturing the temporal dependencies of gait cycles, leading to high classification [18].

Challenges in applying ML to gait analysis include managing the variability in data due to differences in sensor placement, individual biomechanics, and environmental factors. As ML algorithms require large volumes of high-quality training data to learn effectively, the collection and annotation of such datasets are non-trivial tasks [19].

2.4 Convolutional Neural Networks (CNNs) and 1D CNNS

Convolutional Neural Networks (CNNs) are a specialized type of deep neural networks renowned for their ability to process and analyze data with grid-like structures, such as images and time-series sequences. These networks employ layers of convolutional filters to extract high-level features from raw data automatically, which is crucial for tasks requiring pattern recognition, such as image classification, voice recognition, and complex signal analysis. The strength of CNNs lies in their architecture, which effectively captures spatial and temporal dependencies in data through a hierarchical learning process [20].

1D Convolutional Neural Networks are tailored to process one-dimensional data. They are ideally suited for analyzing sequential data, such as audio waves, sensor-generated time-series, and physiological signals like ECG and EEG. Unlike traditional deep neural networks that fully connect all inputs and outputs, 1D CNNs utilize convolutional operations across time, allowing them to capture dynamic changes and patterns over intervals. This attribute makes them highly efficient for real-time signal processing and anomaly detection in continuous data streams [21]. The primary advantage of 1D CNNs in applications such as gait analysis lies in their ability to efficiently process temporal data and detect patterns across time. Each layer in a 1D CNN applies a series of convolutional filters to the input data, which are trained to recognize specific temporal features at various scales. These features are then pooled and normalized through subsequent layers to enhance the network's ability to generalize from training data to unseen scenarios. In practice, this means 1D CNNs can identify subtle abnormalities in gait cycles from accelerometer and gyroscope data integrated into wearable devices, providing valuable insights for diagnostic and therapeutic purposes in real-time [16].

The fundamental difference between 1D and 2D CNNs revolves around their respective domains of application. While 2D CNNs are primarily designed for image data, involving convolutions over two-dimensional spatial data, 1D CNNs focus on one-dimensional data streams. This difference affects their internal structure and the nature of the filters used. In 1D CNNs, filters slide across a single dimension, focusing on detecting patterns over time, which is less computationally intensive and more targeted for sequential data analysis, such as in wearable sensor outputs or financial time series [22].

The table 3 outlines the differences between 1D and 2D CNNs across various aspects such as input data type, primary use, convolution operations, complexity, data representation, and typical examples.

Table 3. Comparison of 1D vs. 2D Convolutional Neural Networks

Feature	**1D CNN**	**2D CNN**
Input Data Type	One-dimensional data such as time-series signals	Two-dimensional data such as images or matrices
Primary Use	Audio processing, ECG/EEG signal analysis, motion analysis in time-series data	Image processing, video analysis, spatial pattern recognition
Convolution	Convolution operations are performed across time, focusing on temporal features	Convolution operations are performed over both spatial dimensions, extracting features from a 2D space
Complexity	Typically less computationally intensive due to fewer dimensions	More computationally demanding due to the complexity of spatial data
Data Representation	Often deals with data where time or sequence is a critical axis, e.g., stock prices, physiological signals	Deals with data where spatial relationships are crucial, e.g., recognizing objects in photos
Examples	Monitoring heart rhythms, analyzing speech patterns, detecting anomalies in sensor streams	Facial recognition, autonomous vehicle navigation, medical image diagnosis

The PRG424 research project which is proposed by the Tallinn University of Technology is a significant endeavor in wearable sensor networks, utilizing advanced machine learning techniques to elevate data analysis capabilities. Among the various models evaluated, Convolutional Neural Networks (CNNs), particularly 1D CNNs, showcased superior performance in tasks that involve complex sensor data analysis. CNNs were exemplary in their ability to interpret complex data from body-area networks within the PRG424 project. Their capability to extract meaningful patterns from noise-embedded sensor data set them apart from other evaluated models. This proficiency was crucial for developing systems aimed at reliable and efficient health monitoring.

Technical Achievements Highlighted in PRG424:

- Robust Feature Extraction and Noise Reduction:
  CNNs effectively isolated key features from noisy data streams, a critical function for accurate health diagnostics and monitoring [23].

- Optimized Real-Time Data Processing:
  The models demonstrated real-time processing capabilities, essential for applications that require immediate data analysis to inform health interventions [24].

- Adaptability Across Varied Sensor Inputs:
  The adaptability of CNNs was further validated by their performance across multiple sensor types and configurations, enhancing the system's reliability and applicability [25].

- Performance Metrics in Gait Analysis:
  The "1D-CNN-AD" algorithm, highlighted in the work by Kumar et al. (2024), showcases high F1 scores for detecting Hyperkinetic and Slap gait patterns, achieving 98.1% and 90.8% respectively. This highlights that 1D CNNs can reliably classify specific gait abnormalities. However, the study also points out performance variability in recognizing gait patterns like Ataxic and Hemiplegic due to individual differences. [33].

The outstanding performance of CNNs in the PRG424 project underscores their potential in handling complex, real-world data effectively. This success motivates additional research into CNN-based models, particularly 1D CNNs, for further applications in wearable technology and health monitoring systems. The findings from PRG424 serve as a benchmark, demonstrating that CNNs not only meet but often exceed the performance of traditional machine learning techniques in challenging scenarios.

Methods developed in PRG424 project will be further developed in ongoing project "AIoT*5G - Artificial intelligence, edge computing and IoT solutions in distributed systems" including activities of improving occupational health and safety via smart IoT devices.

3. RESEARCH METHODOLOGY

3.1 Hardware selection

In the development of wearable technology for gait analysis, selecting the right components is crucial for effective monitoring and data collection. The ESP32 microcontroller, which is designed for energy efficiency and MPU6050 sensor have been chosen due to their optimal computational power and functionality for the given task, and cost-effectiveness, making them particularly suitable for this application.

The ESP32-D0WDQ6 is a microcontroller known for its ability to handle lightweight machine learning tasks directly on the device, making it suitable for applications that require quick data processing and immediate decision-making. Its dual-core Tensilica Xtensa LX6 processor, operating at up to 240 MHz, provides enough computational power for analyzing gait patterns in real time [26]. Below are specific reasons why the ESP32-D0WDQ6 is preferred:

- Efficient Processing: The dual-core processor and 520 KB of SRAM allow it to manage basic machine learning algorithms for real-time gait pattern recognition.

- Wireless Connectivity: Built-in Wi-Fi and Bluetooth enable it to communicate wirelessly with other devices, making it possible to transmit data for remote monitoring and analysis.

- Input/Output Flexibility: The ESP32-D0WDQ6 provides various interfaces like SPI, I2C, UART, and ADC inputs, allowing easy integration with different sensors and devices.

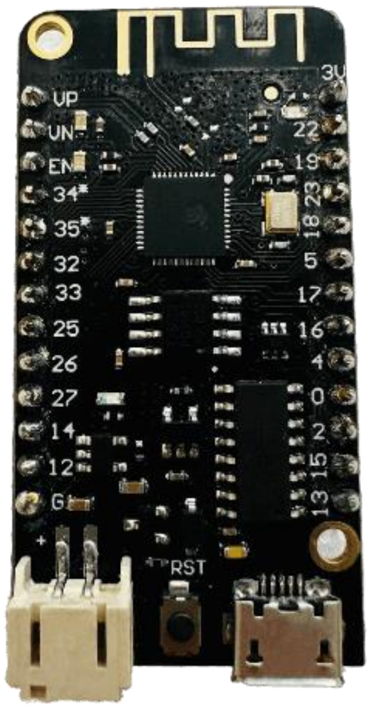


Figure 1. ESP32-D0WDQ6

To visually summarize the differences among the ESP32 variants, the following table compares the key features of the ESP32-D0WDQ6 with other models like the ESP32-S0WD, ESP32-D2WD, ESP32-WROOM, ESP32-WROVER, and ESP32-C3:

Table 4. Comparison of ESP32 Variants

Feature	**ESP32-D0WDQ6**	**ESP32-S0WD**	**ESP32-D2WD**	**ESP32-WROOM**	**ESP32-WROVER**	**ESP32-C3**
Core Type	Dual-core LX6	Single-core LX6	Dual-core LX6	Single-core LX6	Dual-core LX6 with PSRAM	Single-core RISC-V
Max Clock Speed	Up to 240 MHz	Up to 160 MHz	Up to 240 MHz	Up to 160 MHz	Up to 240 MHz	Up to 160 MHz
Wi-Fi	Dual-mode	Single-mode	Dual-mode	Single-mode	Dual-mode	Single-mode
Bluetooth	BT 4.2, BLE	BT 4.2, BLE	BT 4.2, BLE	BT 4.2, BLE	BT 4.2, BLE	BLE only
RAM	520 KB SRAM	520 KB SRAM	520 KB SRAM	520 KB SRAM	520 KB SRAM plus PSRAM	400 KB SRAM
Security Features	Basic	Basic	Basic	Basic	Enhanced	Enhanced with RISC-V TrustZone
Cost	Medium	Low	Medium	Low	Medium	Low
Use Case Suitability	Best for high-performance applications needing dual-core efficiency	Suitable for less complex applications	Similar performance but less popular, fewer community resources	Suitable for less complex applications	Best for applications needing additional memory	Best for cost-sensitive, security-enhanced applications

In addition to its dual-core processor, high clock speed, enhanced connectivity, and sufficient RAM and expandability, the ESP32-D0WDQ6 is notably favored for its robust adoption within the developer community. This widespread usage guarantees access to extensive support and a rich library of resources, which are crucial for effective troubleshooting and the enhancement of AIoT applications.

Moreover, the ESP32-D0WDQ6 offers an excellent balance of performance and cost (5-6 Euro). While it is not the least expensive model in the market—such as the ESP32-S0WD or ESP32-C3—its moderate price is well justified by superior processing power and connectivity options. These features make the ESP32-D0WDQ6 an ideal choice for applications requiring advanced capabilities without compromising on budget.

The MPU6050 is a micromechanical inertial motion sensor that integrates a 3-axis gyroscope and a 3-axis accelerometer on a single chip. This sensor is widely used in applications requiring accurate and efficient motion tracking, such as gait analysis. The following points detail the MPU6050's specifications and justify its selection for gait analysis [27].

- Comprehensive Motion Tracking: The MPU6050 provides critical gyroscopic and accelerometric inputs essential for capturing the complex dynamics of human gait. The sensor’s dual functionality enables it to accurately measure both angular rate and linear acceleration, offering a detailed profile of body movements.

- High Precision and Sensitivity: The MPU6050 features a 16-bit Analog-to-Digital Converter (ADC) for each 6-axis motion tracking channel. This high-resolution sensing allows for the detection of minute variations in movement, essential for detailed and precise gait analysis [27].

- Low Power Consumption: Like the ESP32, the MPU6050 is designed for low power consumption, drawing as little as 3.9 mA in standard measurement modes. This feature makes it ideal for prolonged monitoring in wearable devices, where battery life is a critical factor.

- Ease of Integration: It communicates over I2C protocol, which simplifies its integration with popular microcontrollers, such as the ESP32. This ease of connection supports the rapid development and deployment of complex systems like those used in gait analysis. The sensor also includes built-in pull-up resistors on the I2C lines, which are crucial for ensuring reliable data transmission by maintaining the integrity of the signal during high or floating states.

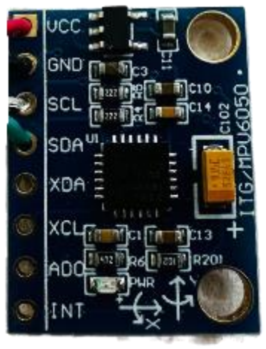


Figure 2. Breakout board with BMU6050 IMU

The combination of ESP32 and MPU6050 provides a powerful, efficient, and cost-effective solution for gait analysis. The ESP32's processing power and connectivity options allow it to quickly process and transmit the data collected by the MPU6050, supporting not only the monitoring and analysis of gait patterns but also enabling the detection of gait abnormalities in real-time.

3.2 Software selection

3.2.1 Hardware selection

The primary options considered were C, C++, Arduino, and Rust. While highly efficient, C requires more complex boilerplate code for hardware interaction, which can increase development time and complexity, making it less suitable for rapid prototyping. Although C++ offers extensive control and efficiency, the complexity of setting up and managing C++ environments for microcontroller programming, compared to Arduino's integrated setup, can be cumbersome in projects needing quick iterations and hardware integrations. Rust is known for its safety and concurrency features, Rust is a compelling choice for high-reliability systems.

However, its ecosystem for embedded development is less mature compared to Arduino’s, with fewer libraries specifically designed for quick implementation and less community support for beginner to intermediate level projects focusing on AIoT. Arduino, a platform and an Integrated Development Environment (IDE) that uses a simplified version of C++, as selected as the programming environment for its ease of use, rapid prototyping capabilities, and extensive library support, making it ideal for integrating complex devices like the MPU6050. Its user-friendly Arduino IDE simplifies coding and debugging, which is advantageous for projects with tight deadlines. Additionally, Arduino's vast global community offers substantial resources and support, which are vital for troubleshooting and refining AIoT applications. As an open-source platform, Arduino provides cost-effectiveness and broad hardware compatibility, ensuring minimal integration issues with devices such as the ESP32.

To save the data in some file formats, Python is selected, I use it to continuously read sensor readings from the serial monitor or Bluetooth port, save it in csv file format and do needed manipulation on the data.

3.2.2 For Running CNN

Since we use the TensorFlow library to train the model, Python was selected as the programming language because it seamlessly integrates with TensorFlow. To deploy and run the 1D CNN model on the ESP32, Arduino was initially selected due to its existing use in collecting data from the MPU6050 sensor. However, issues with TensorFlow Lite Arduino libraries, discussed in Chapter 4.4, led to exploring other deployment methods.

The official option provided by Espressif is to use ESP-IDF, but an even better solution was found in PlatformIO [30]. PlatformIO is an open-source, cross-platform development environment designed for embedded systems, offering a comprehensive suite of tools for simplifying the development process across various platforms, including the ESP32. I chose to use PlatformIO integrated with Visual Studio Code IDE.

3.3 Designing ML model for ESP32

3.3.1 Model Architecture and Design

Deploying a CNN on an ESP32 device presents various challenges, including the constraints of computational power and memory. Despite these challenges, I designed and implemented models to simulate a practical CNN, starting with simpler models and gradually increasing complexity.

Target CNN Model Architecture:
The real research project utilizes a 1D CNN with two convolutional layers (max 256 neurons per layer), followed by a max-pooling layer and two dense layers:

- Input: 480 features (256 Hz sampling rate)
- Layer 1: Conv1D (256 neurons)
- Max-Pooling Layer
- Dense Layers: 100 neurons and 2 neurons respectively

Given the challenges highlighted in various literature sources regarding running CNNs on the ESP32, I began with basic architectures to understand resource limitations and progressively advanced toward the final model.

1. Given First Model: Feedforward Neural Network (Multilayer Perceptron)
   This simple model aimed to provide a baseline:
   - Input Layer: 2 features
   - Hidden Layer: 5 neurons, ReLU activation
   - Output Layer: 1 neuron, Sigmoid activation (binary classification)

2. Second Model: 1D CNN
   This model incorporated convolutional operations for enhanced feature extraction:
   - Input Shape: (2, 1) for two steps with one feature per step
   - Conv1D Layer: 64 filters, kernel size 2, ReLU activation
   - Flatten Layer: Converts 2D output into a single-dimensional vector
   - Dense Layer: 50 neurons, ReLU activation
   - Output Layer: 1 neuron, Sigmoid activation (binary classification)

3. Third Model: Enhanced 1D CNN
   A more complex model with multiple convolutional layers:
   - Input Layer: Shape (20, 1)
   - Conv1D Layer 1: 32 filters, kernel size 5, ReLU activation

- Conv1D Layer 2: 32 filters, kernel size 3, ReLU activation
- MaxPooling Layer: Pool size 2
- Flatten Layer: Converts multi-dimensional feature maps into a single vector
- Dense Layer: 50 neurons, ReLU activation
- Output Layer: 1 neuron, Sigmoid activation (binary classification)

4. Final Model: Simulating Steppage Gait Analysis (1D CNN)
   This model aligns closely with the target application, aiming to classify three types of movement: turn, walk, and abnormal.
   - Input Layer: Shape (20, 6) for accelerometer and gyroscope data
   - Conv1D Layer 1: 32 filters, kernel size 5, ReLU activation
   - Conv1D Layer 2: 32 filters, kernel size 3, ReLU activation
   - MaxPooling Layer: Pool size 2
   - Flatten Layer: Converts feature maps into a flat feature vector
   - Dense Layer: 50 neurons, ReLU activation
   - Output Layer: 3 neurons (Softmax activation to classify movement types)

Table 5 provides a quick reference to the architecture, complexity, and use case of each model.

Table 5. Comparison of Neural Network Models for Gait Analysis on ESP32

Feature	**First Model (Feedforward NN)**	**Second Model (1D CNN)**	**Third Model (Enhanced 1D CNN)**	**Fourth Model (Steppage Gait Analysis - 1D CNN)**
Input Shape	(2,)	(2, 1)	(20, 1)	(20, 6)
Layers	- Input: 2 features - Hidden: 5 neurons - Output: 1 neuron	- Conv1D: 64 filters - Flatten - Dense: 50 neurons - Output: 1 neuron	- Conv1D: 32 filters - Conv1D 1: 32 filters - MaxPooling: pool size 2 - Flatten - Dense: 50 neurons - Output: 1 neuron	- Conv1D 1: 32 filters - Conv1D 2: 32 filters - MaxPooling: pool size 2 - Flatten - Dense: 50 neurons - Output: 3 neurons
Activation Functions	ReLU (hidden) Sigmoid (output)	ReLU (Conv1D, Dense) Sigmoid (output)	ReLU (Conv1D, Dense) Sigmoid (output)	ReLU (Conv1D, Dense) Softmax (output)
Pooling	None	None	MaxPooling (pool size 2)	MaxPooling (pool size 2)
Target Task	Binary classification	Binary classification	Binary classification	Multi-class classification (turn, walk, abnormal)
Complexity	Low	Moderate	High	Very High
Suitable Use Case	Simple binary classification	Time-series classification	Time-series classification	Multi-feature, multi-class gait analysis

3.3.2 Model training and conversion

Training Setup:

- **Libraries:** TensorFlow and Keras were used for model implementation due to their comprehensive neural network layers and model optimization. Tensorflow is also used in the PRG424 project. In this project, I have used the latest version of Tensorflow as of now is 2.16.1.Runs on different platforms: Linux, macOS, Windows

- **Data Generation:** For the first three models in Table 3, training scripts utilized a custom data generator function to simulate training data dynamically. These generators produced sequences of varying lengths and feature sets that were fed into the training pipeline via TensorFlow datasets. For the last model, I have used the datasets which are collected using the MPU6050 sensor - this is discussed in chapter 4.3 in detail.

Model Training:

- The models increased in complexity across the scripts, starting with a simple binary classifier and advancing to multi-class models with diverse feature sets. Each model was trained over 10 epochs with the Adam optimizer and appropriate loss functions (BinaryCrossentropy or SparseCategoricalCrossentropy)

Model Conversion to TFLite:

- When targeting embedded systems like the ESP32, it is important to convert Tensorflow models to Tensorflow Lite as it ensures that the model remains lightweight, compatible, and efficient while delivering the desired inference performance on devices like the ESP32. However, during conversion, some operations in the TensorFlow model couldn't be converted to TensorFlow Lite equivalents. The presence of unconverted operations in your model affects the generated TensorFlow Lite model in the following ways:

  Compatibility Issues: Some parts of the model may not run natively on devices, potentially affecting efficiency.

  Fallback to Reference Ops: The non-converted operations might be executed using reference implementations, which can be slower or less optimized than fully converted operations.

Inference Speed and Size: The use of non-optimized operations could slow down inference speed and increase the model size, reducing the efficiency expected from a TFLite model.

To fix this issue, I tried to simplify the architecture and verify the Tensorflow Lite compatibility, however, some of the operations are still not converted. This can be seen in Table 6.

Table 6. Model Characteristics

Feature	First Model (Feedforward NN)	Second Model (1D CNN)	Third Model (Enhanced 1D CNN)	Fourth Model (Steppage Gait Analysis - 1D CNN)
Training Time (seconds)	58	76	88	84
Converted Model Size (Kilobyte)	4	8.7	62	65
Total Operations	14	17	28	28
Non-Converted Operations	3	4	11	11

TFLite Conversion to Data Array:

- The ESP32 microcontroller lacks a file system, making it challenging to directly load TensorFlow Lite models. To embed the model, it is converted to a data array using the Linux command-line tool xxd. This tool converts the .tflite model file into a C-style array:

```
xxd-i model.tflite>modelData.cc
```

The resulting .cc file contains an array of weights and parameters. By including it in the project as source code, the model data can be accessed directly, simplifying deployment.

4. PROJECT IMPLEMENTATION

4.1 Hardware setup

Connecting an MPU6050 sensor to an ESP32 is fairly straightforward, as the MPU6050 uses the I2C communication protocol.

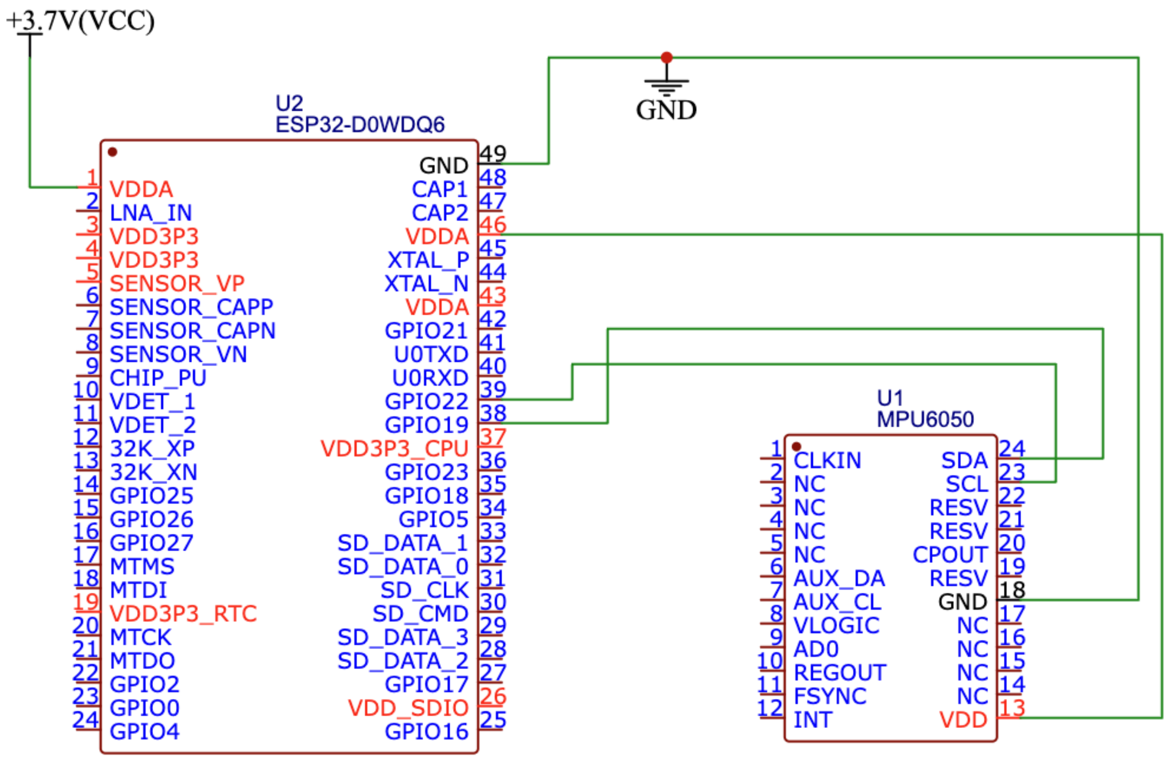


Figure 3. Wiring Diagram of main components

- SDA (I2C Serial Data Line) - Connected the SDA pin of the MPU6050 to GPIO 19 on the ESP32. This pin serves as the data line for I2C communication.

- SCL (I2C Serial Clock Line) - Connected the SCL pin of the MPU6050 to GPIO 22 on the ESP32. This pin provides the clock signal for synchronizing I2C communication.

4.2 Sensor placement

To record data for steppage gait accurately using the MPU6050 sensor, which captures both acceleration and angular velocity, the sensor should be placed where it can best measure the leg's kinematics. For steppage gait, which is often characterized by an exaggerated lifting of the knee and foot due to muscle weakness or nerve damage, key points of interest could be:

- On the Shin: Placing the sensor on the shin, close to the tibia, can help record the forward motion and the elevated lifting of the foot that occurs with each step.

- Top of the Foot: This can give you information about the foot's tilt and lift as the person tries to prevent the toe from dragging.

- Ankle: Since steppage gait involves a higher foot lift, placing the sensor near the ankle can help monitor the ankle's angle during the gait cycle.

The steppage gait is characterized by an exaggerated flexion of the ankle to lift the foot higher off the ground, preventing the toes from scraping the floor. Placing the sensor on top of the foot directly captures this motion. The top of the foot is a pivotal point in gait analysis as it can provide data on both the vertical lift (important in steppage gait) and the foot's orientation during stride. Other points, like the shin or thigh, may not offer as direct a measure of these specific movements. Therefore, it is decided to place the sensor on top of the foot which is shown in Figure 4.

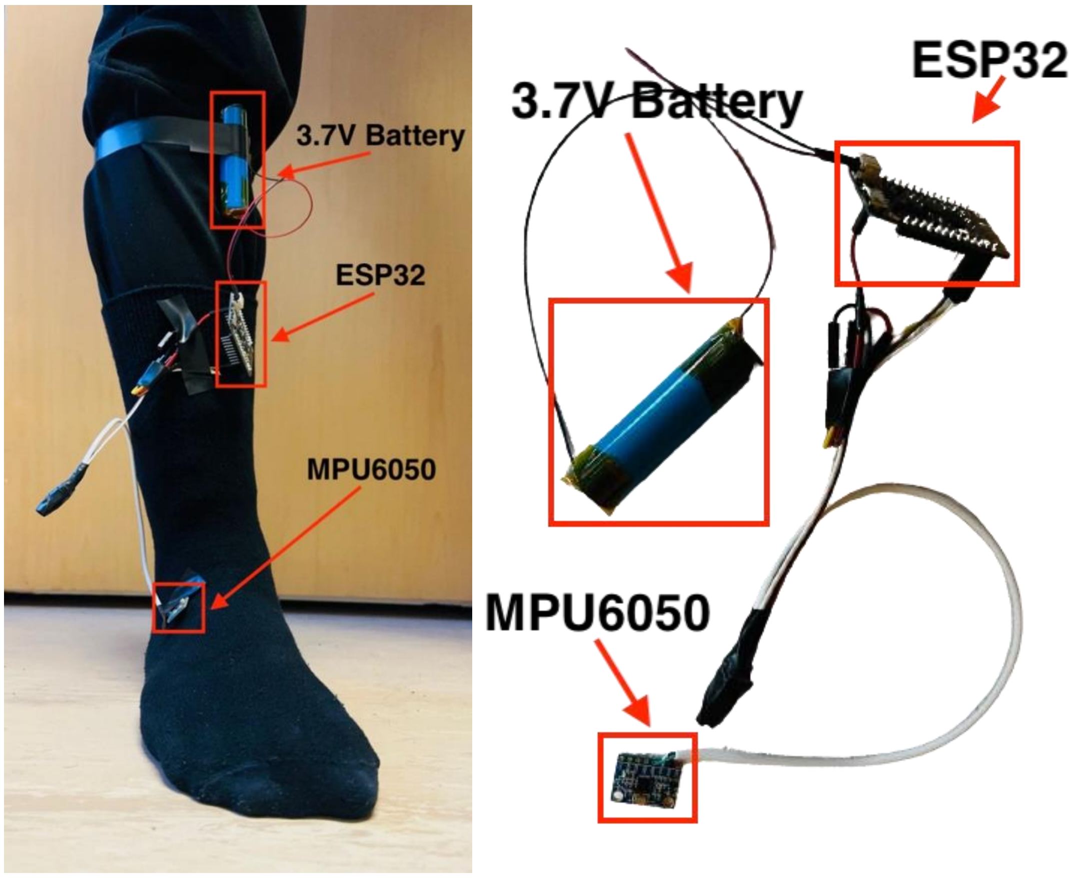


Figure 4. Placement of MPU6050 and ESP32 for Steppage Gait Analysis

4.3 Data collection

In order to train ML model for steppage gait, at least two sets of data should be collected. For the initial tests, I have created a simple Arduino sketch which is tailored to gather motion data from an MPU6050 sensor using an ESP32 microcontroller and send this data to the serial port through Bluetooth. The choice to use Bluetooth for data transmission was driven by the need for remote, wireless communication in the data collection process, essential for gathering extensive datasets across different environments without physical tethering.

1. Initial Setup
    - The script initializes I2C communication using specific GPIO pins for the MPU6050 sensor.
    - It sets up serial communication at a baud rate of 115200 bps for output and checks if the sensor is connected and responsive.
    - The accelerometer and gyroscope measurement ranges, which are respectively ±8g and 500 dps, are configured to allow for capturing a wide range of motions.

2. Sensor Configuration and Testing
    - The MPU6050 sensor is initialized and its full-scale measurement ranges for acceleration and angular velocity are set.
    - The script checks for a successful connection with the sensor and, if failed, it halts further execution.

3. Data Collection and Processing
    - In the main loop, motion data including 3-axis acceleration and 3-axis gyroscope readings are continuously collected.
    - The raw sensor data is converted into real-world units (meters per second squared for acceleration and degrees per second for gyroscopic measurements).

4. Data Output
    - The converted data, along with a timestamp, is formatted into a CSV string and sent to the serial port.

5. Efficiency and Stability
    - A short delay of 3 milliseconds is incorporated in the loop to help stabilize the rate of data output and sample the data at a rate of 200Hz, equivalent to acquiring data every 5 milliseconds.

While the data is printed continuously on the serial monitor, we need to save the data in some file formats, however, Arduino IDE doesn't provide such an option by default. Therefore, I have made a simple python script which establishes a serial connection with an ESP32 microcontroller, continuously monitors incoming data, decodes it, and systematically logs it to a CSV file for efficient data recording and analysis. The script ensures real-time data capture from the MPU6050 sensor, with each received datum being decoded and saved to a CSV file on the computer, facilitating further data manipulation and study.

After desired data is collected in a csv file, the next step is to analyze the stability of the sampling rate which will indicate how consistent the intervals between samples are. However, Bluetooth communication issues such as signal interference and inconsistent sampling intervals initially posed significant challenges.

Initial Challenges:

External interference during Bluetooth transmission caused inconsistencies in sampling intervals, leading to fluctuating data collection. Transmission delays introduced periodic spikes in sampling intervals, making it difficult to achieve a consistent 200 Hz sampling rate as it is shown in Figure 5.

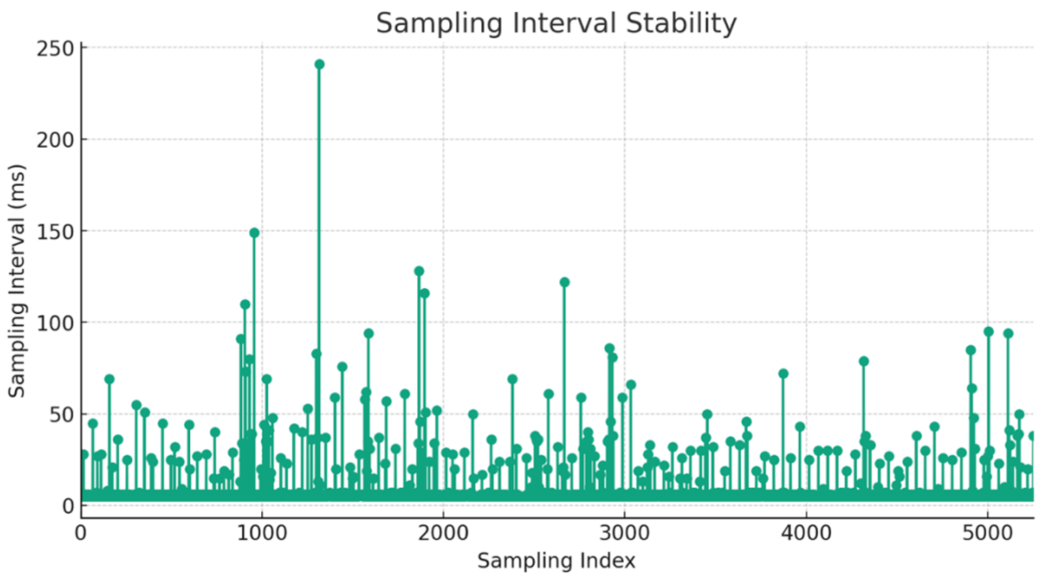


Figure 5. Consistency of Sampling Intervals at 200Hz Sampling Rate (Wireless Bluetooth Communication)

Final Solution: Buffering and Batch Transmission

To overcome these challenges, a solution was designed to incorporate data buffering and batch transmission:

Data collected by the MPU6050 sensor is temporarily stored in an internal buffer. Once the buffer reaches a predefined size, the data is transmitted in a batch over Bluetooth, reducing the impact of intermittent interference. Figure 6 visualizes the architectural diagram of overall process.

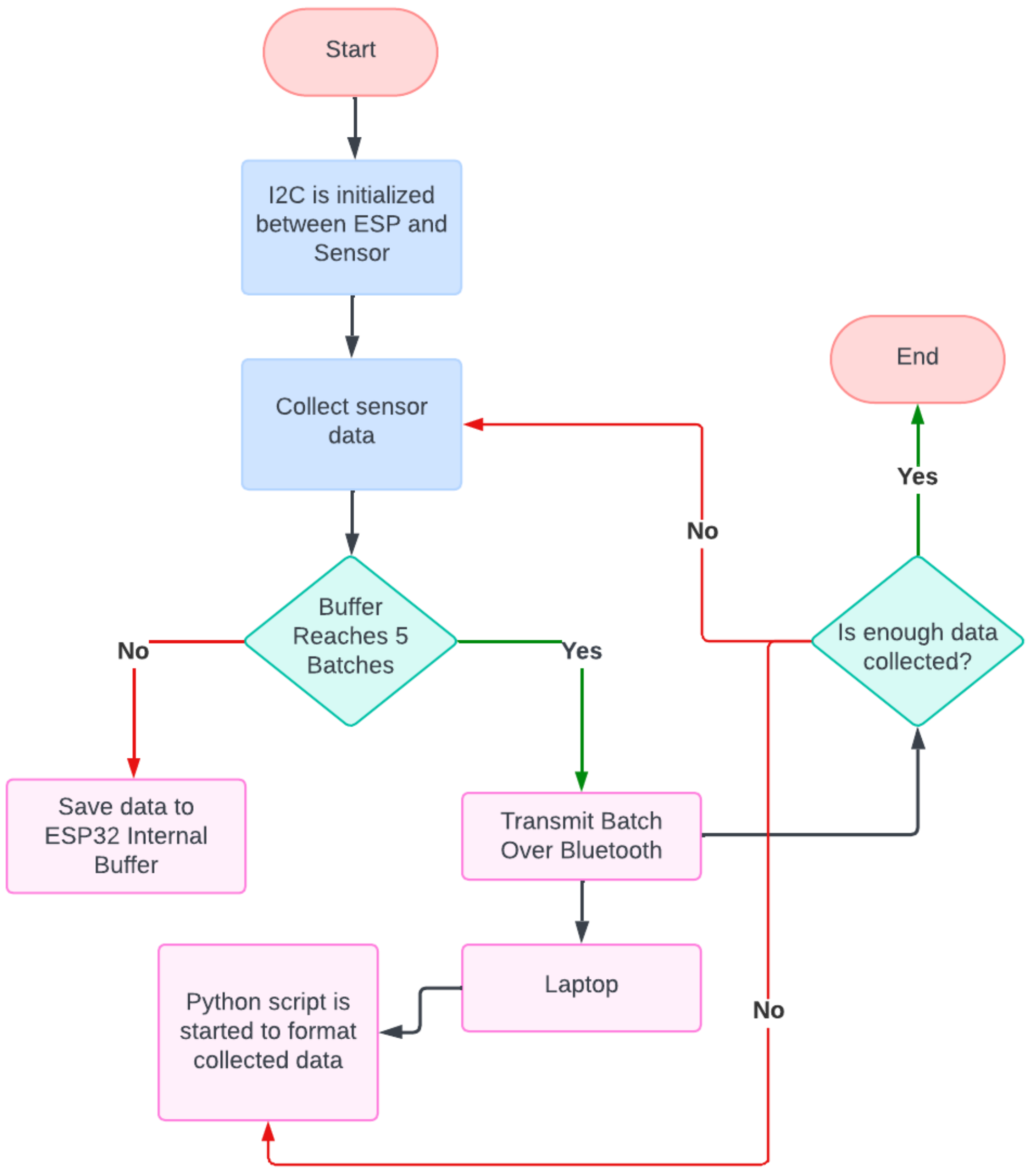

Start
I2C is initialized between ESP and Sensor
Collect sensor data
Buffer Reaches 5 Batches
No
Yes
Save data to ESP32 Internal Buffer
Transmit Batch Over Bluetooth
Laptop
Python script is started to format collected data
Is enough data collected?
No
Yes
End
No

Figure 6. MPU6050 Data Acquisition and Bluetooth Transmission Flowchart

By incrementally adjusting the delay added to stabilize data transmission rates, we achieved an optimal setting of a 4 ms delay with a batch size of 5. This reduced the transmission spikes and achieved a consistent sampling rate of 196 Hz as it is shown in Figure 7 and in Figure 8, suitable for walking motion analysis.

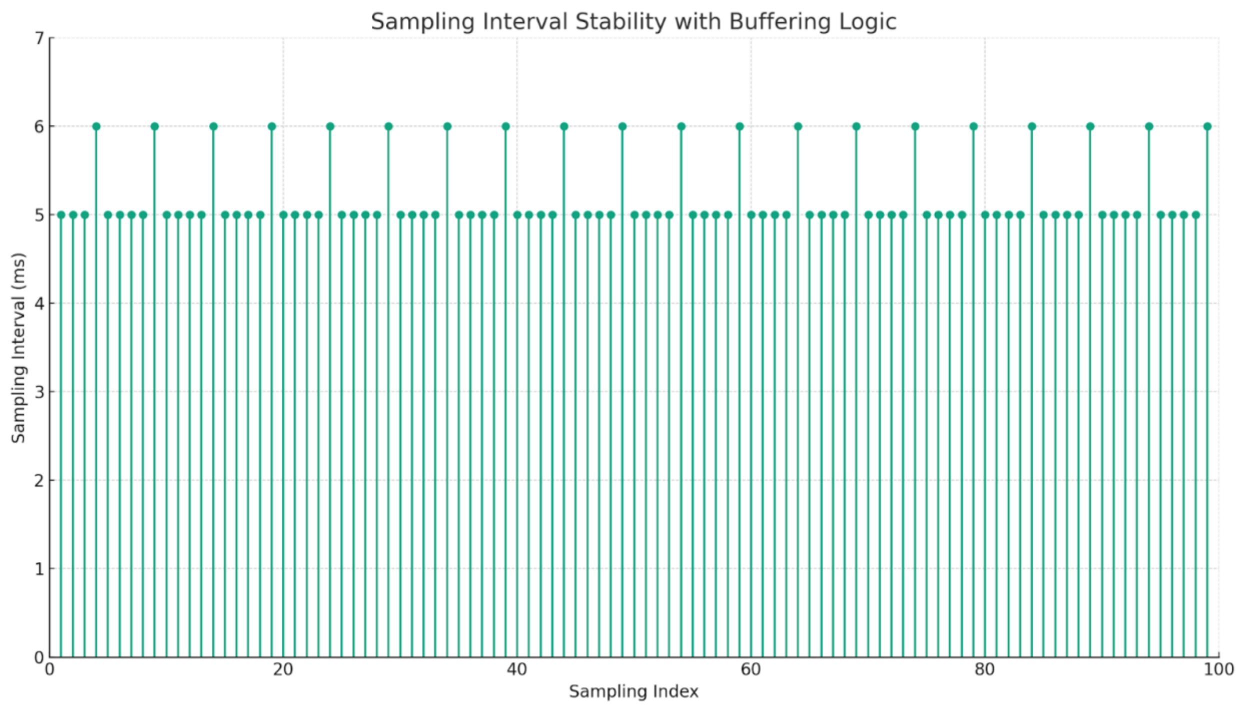


Figure 7. Consistency of Sampling Intervals at 196Hz Sampling Rate (Wireless Bluetooth Communication with Logic of Buffering - batch size = 5, delay = 4, samples = 100)

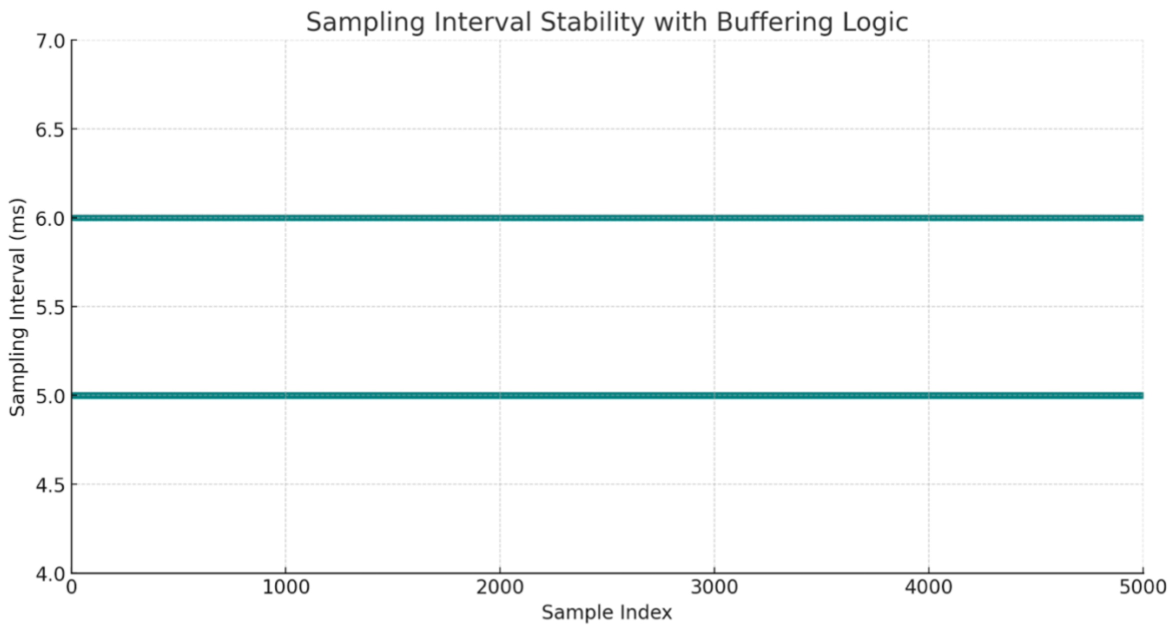


Figure 8. Consistency of Sampling Intervals at 196Hz Sampling Rate (Wireless Bluetooth Communication with Logic of Buffering - batch size = 5, delay = 4, samples = 5000)

To sum up, the Bluetooth data acquisition system successfully achieved a sampling rate of 196 Hz with a buffering and batch transmission approach. However, this solution is not fully reliable for consistent sampling intervals due to inherent limitations. The external interference during Bluetooth transmission causes intermittent delays and periodic spikes, resulting in inconsistent data collection. Although the batch transmission method helped reduce transmission spikes and maintained a stable 196 Hz rate, this approach still struggles with providing stable consistency between samples. Furthermore, obtaining higher sampling rates with reliable consistency becomes increasingly complex due to signal interference and buffering limitations.

4.4 Data labeling

After we collect the data, the next step is to turn it into an audio file. To do this, we used a handy Python script developed by a team member from the PRG424 research group. However, in the generated audio file, there was a kind of distortion – like the signal was being squeezed too much. This problem, known as signal clamping, was because we didn't set the MPU6050 sensor to catch all the data we were throwing at it; it was set to a range of ±2g, which, plainly speaking, wasn't enough. Realizing this, we adjusted the sensor to a wider range of ±4g, thinking it would surely be enough. But, when we made the audio file again, the problem was still there – the sound was still hitting a limit. Figure 9 shows a part of the generated audio file. Even though the range was set to ±4g, we have still experienced signal clamping.

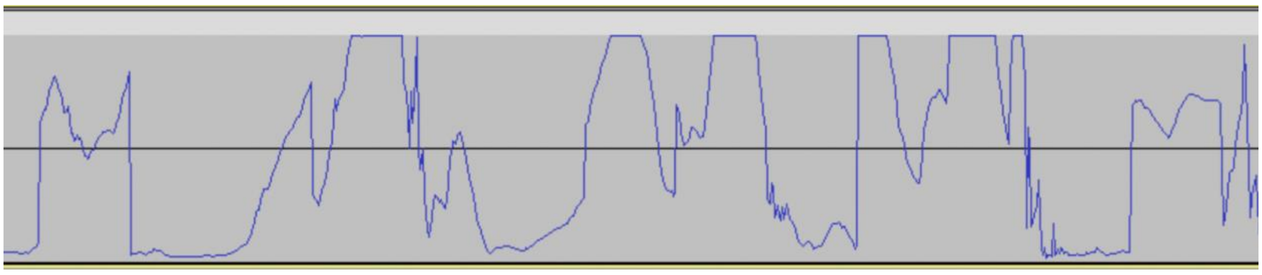


Figure 9. Generated audio file where sensor range is set to ±4g

Therefore, it was decided to increase the range to ±8g and it avoided signal clamping. Figure 10 shows the generated audio file where sensor range was set to ±8g.

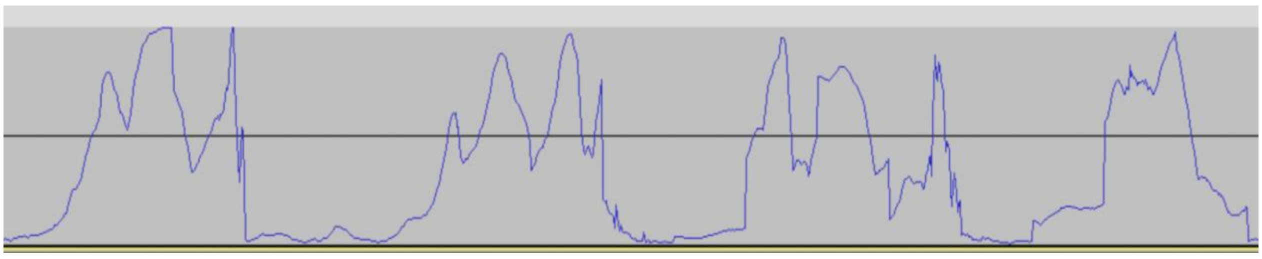


Figure 10. Generated audio file where sensor range is set to ±8g

We've compiled four separate recordings, each capturing approximately 35-40 instances of movement. These include regular walking steps, right and left turns, and abnormal steps designed to mimic the steppage gait pattern. Figure 11 shows labeled audio data which is then used to generate a text file that can be used as an input for training ML model.

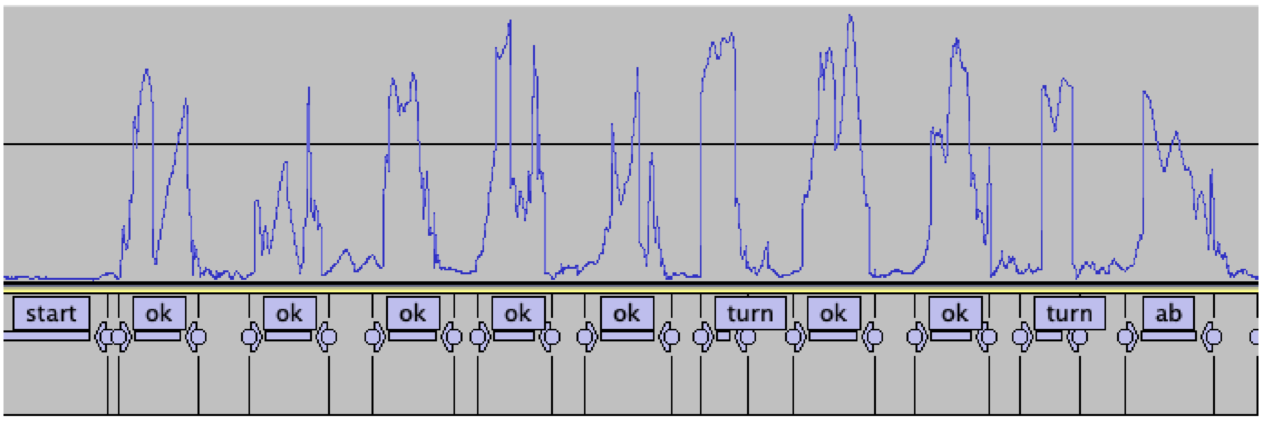


Figure 11. Labeled audio data

4.4 Deploying and running model on esp32

Since Arduino Studio was already used to collect data from the MPU6050 sensor, I explored deploying machine learning models on the ESP32 through Arduino. However, numerous issues arose during the deployment process that took significant time to troubleshoot, ultimately yielding unsatisfactory results. There are mostly 2 libraries recommended by the Arduino community to run ML models through Arduino Studio.

1. TensorFlowLite_ESP32 Library

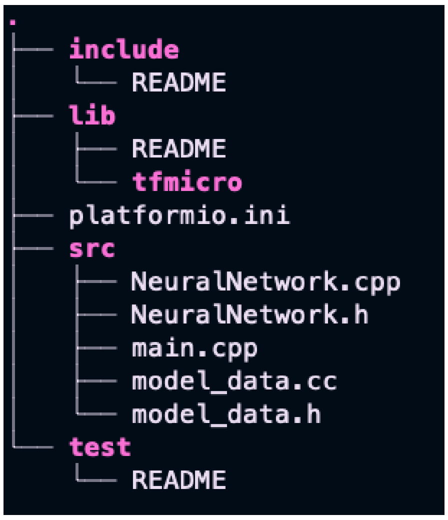
   This library, provided by the TensorFlow authors, hadn't been updated for two years, limiting its compatibility with newer models such as 2.16.1 which is the latest version of TensorFlow. By downgrading the TensorFlow version to 2.5.0, some issues were mitigated, allowing the first two models to function correctly. Unfortunately, this wasn't a complete solution since it failed to support the last two, more complex models [28].

2. EloquentTinyML Library

   Switching to EloquentTinyML (source: EloquentTinyML) seemed promising, as it provided clearer instructions and easy-to-follow examples. All models were initially deployed successfully, but the ESP32 would only run for 3 seconds before ceasing output. Additionally, Arduino's strict compiler settings treated warnings as errors, complicating the build process [29].

Attempts to modify compiler flags to suppress errors led to further issues, prompting the switch from Arduino Studio.

PlatformIO [30], an open-source cross-platform environment, offered a more flexible solution. Integrated with Visual Studio Code, it simplified development and deployment. By building the TensorFlow Lite library with CMake [31], a wrapper was created for easier integration with the Arduino framework on PlatformIO. Project structure is shown in Figure 12.



```
.
├── include
│   └── README
├── lib
│   ├── README
│   └── tfmicro
├── platformio.ini
├── src
│   ├── NeuralNetwork.cpp
│   ├── NeuralNetwork.h
│   ├── main.cpp
│   ├── model_data.cc
│   └── model_data.h
└── test
    └── README
```

Figure 12. PlatformIO project strcuture

Since we have already built the Tensorflow Lite library (tfmicro), we create the NeuralNetwork class around it which loads and manages the pre-trained model (model_data.cc), initializes a TensorFlow Lite interpreter, and registers essential neural network operations. We then allocate 20KB tensor area to provide workspace for the model during inference. The main.cpp file contains the main Arduino sketch that calls the neural network class for predictions. To successfully deploy the model on esp32 we also need to include "Arduino.h" library which is provided by the PlatformIO.

The following configuration is used for the ESP32-D0WDQ6 model:

```
[env:esp32dev]
platform = espressif32
board = esp32dev
framework = arduino
monitor_speed = 115200
```

Figure 13 visualizes the high-level architectural diagram of overall process.

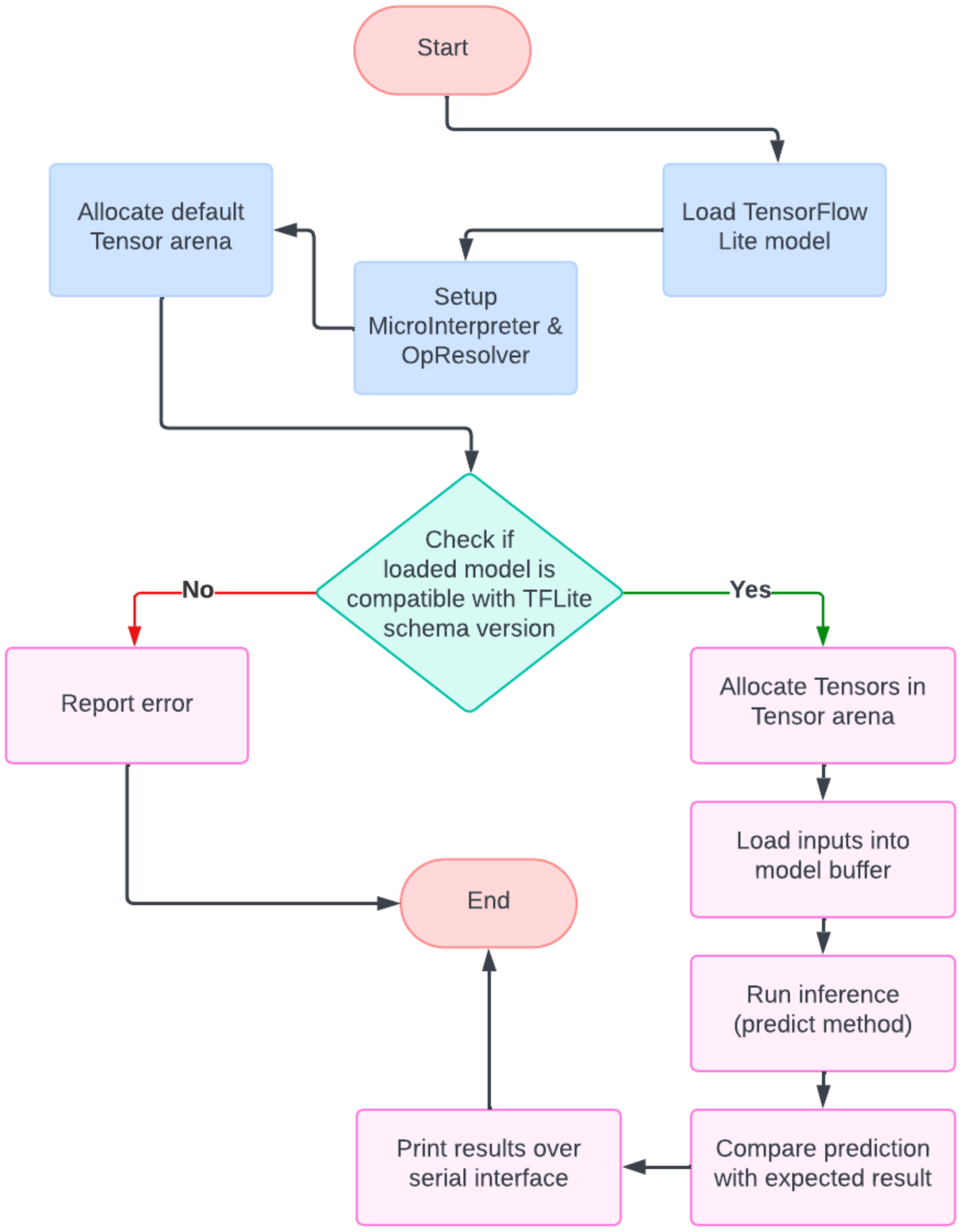
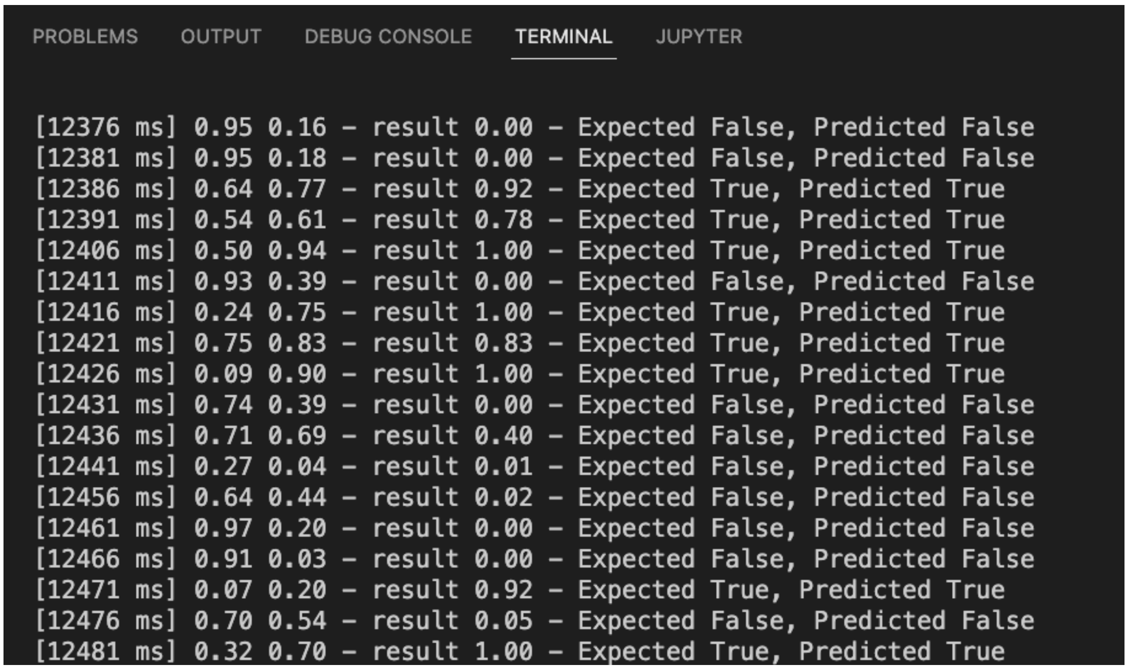

Start
Load TensorFlow Lite model
Setup MicroInterpreter & OpResolver
Allocate default Tensor arena
Check if loaded model is compatible with TFLite schema version
No
Yes
Report error
Allocate Tensors in Tensor arena
Load inputs into model buffer
Run inference (predict method)
Compare prediction with expected result
Print results over serial interface
End

Figure 13. Flowchart of Running ML model on ESP32

For the first 3 models, described in Table 4, I have created a simple binary classification script - every iteration, two random numbers are generated between 0.0 and 1.0. These numbers are set as inputs for the neural network, The network processes these inputs to make a prediction. The code then compares the predicted value with expected value (whether the generated second number is greater that generated first number). I was able to deploy the first 2 models on esp32. Simple binary classification models worked smoothly. Data was fed at a consistent 200 Hz rate, and the results met expectations. A part of output of serial readings provided in Figure 14:



```
[12376 ms] 0.95 0.16 - result 0.00 - Expected False, Predicted False
[12381 ms] 0.95 0.18 - result 0.00 - Expected False, Predicted False
[12386 ms] 0.64 0.77 - result 0.92 - Expected True, Predicted True
[12391 ms] 0.54 0.61 - result 0.78 - Expected True, Predicted True
[12406 ms] 0.50 0.94 - result 1.00 - Expected True, Predicted True
[12411 ms] 0.93 0.39 - result 0.00 - Expected False, Predicted False
[12416 ms] 0.24 0.75 - result 1.00 - Expected True, Predicted True
[12421 ms] 0.75 0.83 - result 0.83 - Expected True, Predicted True
[12426 ms] 0.09 0.90 - result 1.00 - Expected True, Predicted True
[12431 ms] 0.74 0.39 - result 0.00 - Expected False, Predicted False
[12436 ms] 0.71 0.69 - result 0.40 - Expected False, Predicted False
[12441 ms] 0.27 0.04 - result 0.01 - Expected False, Predicted False
[12456 ms] 0.64 0.44 - result 0.02 - Expected False, Predicted False
[12461 ms] 0.97 0.20 - result 0.00 - Expected False, Predicted False
[12466 ms] 0.91 0.03 - result 0.00 - Expected False, Predicted False
[12471 ms] 0.07 0.20 - result 0.92 - Expected True, Predicted True
[12476 ms] 0.70 0.54 - result 0.05 - Expected False, Predicted False
[12481 ms] 0.32 0.70 - result 1.00 - Expected True, Predicted True
```

Figure 14. Output of First 2 models

The Figure 14 displays the Serial Monitor's output. The first column shows the elapsed time since the program started, and the second column lists two randomly generated numbers as inputs. The third column represents the neural network's prediction, which is a floating-point value between 0 and 1 returned by NeuralNetwork->predict(). This value indicates the probability that the second number is greater than the first. In the code (main.cpp), a threshold of 0.5 is used to convert this probability into a binary result:

- 0: The score is less than 0.5, meaning the second number is predicted to be less than or equal to the first.

- 1: The score is 0.5 or higher, meaning the second number is predicted to be greater than the first.

For instance, if the second number is greater than the first, the neural network will likely return a result close to 1, and if not, the result will be close to 0. The last two columns compare the actual and predicted results of the two inputs. It is also observed that actual response time of trained 1D-CNN model (2nd model in Table 5) is smaller than 0.1 ms, however, to get consistency between every prediction, we should have at least 5 ms delay in the loop.

Although I have successfully deployed first 2 simple models from Table 5 on ESP32, the final two models faced deployment challenges, received below errors:

Unsupported Operation (EXPAND_DIMS), Memory Alignment and Tensor Allocation failure, Unhandled Exception (Guru Meditation Error: Core 1 panic'ed (LoadProhibited)).

Errors related to memory allocation which caused an exception in the ESP32 microcontroller that it could not handle, causing a system crash and automatic reboot. Despite extensive efforts, these issues couldn't be fully resolved. Although the "Unsupported Operation" issue was addressed, the exception seemed connected to it, as 11 out of 28 operations failed to convert during the TensorFlow-to-TensorFlowLite conversion for the last two models.

By and large, while the deployment of the first two simpler models on the ESP32 demonstrated effective results, the final two models encountered significant challenges during implementation. These challenges highlight the inherent limitations of deploying intricate machine learning models on resource-constrained embedded devices. The ESP32 is effective for simpler models, but for more demanding tasks, optimizations and model adjustments will be essential.

5. CONCLUSION

This work concentrated on collecting gait data and deploying 1D Convolutional Neural Networks (1D CNNs) on low-power devices, specifically the ESP32 microcontroller. The combination of the ESP32-D0WDQ6 microcontroller and the MPU6050 sensor provided an efficient platform for real-time gait analysis. Despite the computational and memory limitations of the microcontroller, 1D CNN models were successfully implemented, providing benchmark data on running machine learning models on the ESP32.

By using strategies such as data buffering, batch transmission, and timer interrupts, the system achieved consistent data sampling rates and stable data transmission over Bluetooth to collect data for off-device model training. Deploying complex machine learning models on ESP32 device presented challenges related to memory alignment and tensor allocation.

Nonetheless, the current system showed that even small 1D CNN models can cause memory issues if the model architecture is not explicitly designed for low-power devices like the ESP32. The 1D CNN model designed by the PRG424 research group, which has 480 features, could not be deployed, or run effectively on the ESP32 in its current form. Optimizing the architecture and reducing model size will be crucial steps for enabling efficient deployment and execution of machine learning models on this platform. The work highlights the importance of tailored model design and optimization to make practical use of low-power, resource-constrained devices in advanced gait analysis.

6. FUTURE WORK

There are several improvements that can further advance this research. First, as for data collection simple wired communication can be used rather than Bluetooth communication system because we concluded that this system is not reliable. 1D CNN architectures designed specifically for low-power devices should be optimized. This will involve pruning redundant layers, minimizing the number of filters, and using quantization techniques to reduce the model's memory footprint and computational requirements.

Another important aspect will be exploring more effective methods for converting models to TensorFlow Lite (TFLite) for the ESP32. Ensuring that the operations in the original models remain compatible with the lightweight TFLite framework can be achieved through custom implementations or simplifying the model architecture.

Memory management is also essential, where optimizing memory usage will improve data buffering strategies and ensure proper tensor allocation. Allocating enough memory space for the tensor arena and managing other onboard processes will help prevent system crashes. Supporting parallel processing across the ESP32's dual-core architecture will help distribute computational tasks more evenly, thereby reducing processing time.

These suggestions will help in addressing challenges while using the full potential of 1D CNNs on low-power devices for consistent and scalable gait analysis.

7. SUMMARY

This thesis delves into the prevalence, characteristics, and implications of gait disorders, particularly focusing on their occurrence in older adults and individuals with neurological conditions. It emphasizes how gait abnormalities increase the risk of dependence, cognitive decline, and mortality. The research identifies various gait types and disorders, providing insights into their physiological and neurological origins. Steppage gait, a primary focus, is characterized by an exaggerated lifting of the knee and hip due to dorsiflexor muscle weakness.

Technological advancements in gait analysis, such as motion capture systems and wearable sensors, have improved monitoring and assessment outside of clinical settings. However, these systems still face limitations in precision and data volume management.

The thesis highlights the potential of machine learning, especially convolutional neural networks (CNNs) and their 1D variant (1D CNNs), in gait analysis. These models can effectively identify gait patterns, diagnose disorders, and predict potential risks. The research projects "PRG424" and “AIoT*5G” showcases the efficacy of CNNs in wearable sensor networks for extracting meaningful patterns and providing real-time processing and adaptability across varied sensor inputs.

The research methodology involves selecting the right hardware, particularly the ESP32 microcontroller and MPU6050 sensor, for efficient gait analysis. The ESP32-D0WDQ6 is chosen due to its dual-core processor, high performance, and built-in Wi-Fi and Bluetooth for seamless data transmission. Various neural network models, from simpler to medium complexity, are developed to test how they will perform on low-power devices.

Despite challenges related to memory and computational constraints, the thesis demonstrates the feasibility of using 1D CNNs on the ESP32 platform. The findings emphasize the importance of model optimization and tailored architecture design to maximize the potential of low-power devices in gait analysis.

8. KOKKUVÕTE

See lõputöö keskendub kõnnihäirete levimusele, omadustele ja mõjudele, eriti nende esinemisele eakatel ja neuroloogiliste seisunditega inimestel. Töö rõhutab, kuidas kõnnihäired suurendavad sõltuvuse, kognitiivse languse ja suremuse riski. Uurimus tuvastab mitmesugused kõnnitüübid ja -häired, pakkudes teavet nende füsioloogiliste ja neuroloogiliste päritolude kohta. Erilist tähelepanu pööratakse stepp-kõnnile, mida iseloomustab põlve ja puusa liialdatud tõstmine dorsifleksion lihaste nõrkuse tõttu.

Tehnoloogilised edusammud kõnnianalüüsis, nagu liikumise jäädvustamise süsteemid ja kantavad andurid, on parandanud jälgimist ja hindamist kliinilistest seadetest väljaspool. Siiski on neil süsteemidel endiselt piirangud täpsuses ja andmemahtude haldamises.

Lõputöö toob esile masinõppe, eriti konvolutsioon iliste närvivõrkude (CNN) ja nende 1D variandi (1D CNN), potentsiaali kõnnianalüüsis. Need mudelid suudavad tõhusalt tuvastada kõnnimustreid, diagnoosida häireid ja ennustada potentsiaalseid riske. Uurimisprojektid "PRG424" ja "AIoT*5G" demonstreerivad CNNide tõhusust kantavate andur võrkudega, võimaldades tähendusrikkaid mustreid välja tuua ning pakkuda reaalajas töötlemist ja kohandatavust erinevate andurisignaalidega.

Uurimismetoodika hõlmab sobiva riistvara valimist, eriti ESP32 mikrokontrolleri ja MPU6050 anduri, tõhusaks kõnnianalüüsiks. Valitud on ESP32-D0WDQ6, mis on valitud oma kahe tuumaga protsessori, kõrge jõudluse ning sisseehitatud Wi-Fi ja Bluetoothi tõttu sujuvaks andmeedastuseks. Erinevad närvivõrgu mudelid, lihtsamatest keskmise keerukuse, on arendatud, et testida, kuidas need madala võimsusega seadmetes toimivad.

Hoolimata mäluga ja arvutuslikest piirangutest, näitab lõputöö 1D CNNide kasutamise teostatavust ESP32 platvormil. Leiud rõhutavad mudeli optimeerimise ja kohandatud arhitektuuri kujundamise tähtsust, et maksimeerida madala võimsusega seadmete potentsiaali kõnnianalüüsis.

9. LIST OF REFERENCES

[1] Joe Verghese MD, Aaron LeValley MA, Charles B. Hall PhD, Mindy J. Katz MPH, Anne F. Ambrose MD, Richard B. Lipton MD, "Epidemiology of Gait Disorders in Community-Residing Older Adults", December 2005

[2] Jacquelin Perry and Judith M. Burnfield, "Gait Analysis: Normal and Pathological Function" 2010

[3] Stephanie Studenski, MD, MPH; Subashan Perera, PhD; Kushang Patel, PhD," Gait Speed and Survival in Older Adults", January 2011

[4] Matteo Cesari MD, PhD, Stephen B. Kritchevsky PhD, Brenda W. H. J. Penninx PhD, Barbara J. Nicklas PhD, Eleanor M. Simonsick PhD, Anne B. Newman MD, MPH, Frances A. Tylavsky DrPH, Jennifer S. Brach PT, PhD, GCS, Suzanne Satterfield MD, DrPH, Douglas C. Bauer MD, Marjolein Visser PhD, Susan M. Rubin MPH, Tamara B. Harris MD, MS, Marco Pahor MD, "Prognostic Value of Usual Gait Speed in Well-Functioning Older People—Results from the Health, Aging and Body Composition Study", September 2005

[5] D C Kerrigan 1, M K Todd, U Della Croce, "Gender differences in joint biomechanics during walking: normative study in young adults", 1998

[6] Bastiaan R. Bloema, Jasper E. Visseraand John H.J. Allum, "Posturography" 2003

[7] Rolf Moe-Nilssen, Jorunn L Helbostad, "Estimation of gait cycle characteristics by trunk accelerometry", January 2004

[8] Whittle, M. W., "Gait Analysis: An Introduction. Butterworth-Heinemann", 1996

[9] Perry, J., & Burnfield, J. M., "Gait Analysis: Normal and Pathological Function" in Journal of Sports Science and Medicine, June 2010

[10] Ružička E, Jankovic JJ. Disorders of gait. In: Jankovic JJ, Tolosa E,
"Parkinson's disease and movement disorders" 2002

[11] Walter Pirker, Regina Katzenschlager, “Gait disorders in adults and the elderly”, 2016

[12] Tao, W., Liu, T., Zheng, R., & Feng, H., “Gait analysis using wearable sensors”, February 2012

[13] Muro-de-la-Herran, A., Garcia-Zapirain, B., & Mendez-Zorrilla, A., “Gait analysis methods: An overview of wearable and non-wearable systems, highlighting clinical applications” Sensors, 14(2), 3362-3394, February 2014

[14] Hausdorff, J. M., “Gait dynamics, fractals and falls: finding meaning in the stride-to-stride fluctuations of human walking.” Human Movement Science, 26(4), 555-589, 2007

[15] Dai, H., Lin, H., & Lueth, T. C., “Automatic classification of gait patterns in patients with Parkinson’s disease using a wearable sensor system” IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI), 144-147, 2014

[16] LeCun, Y., Bengio, Y., & Hinton, G., “Deep learning”. Nature, 521(7553), 436-444, 2015

[17] Andrea Mannini 1, Angelo Maria Sabatini, “Machine learning methods for classifying human physical activity from on-body accelerometers', February 2010

[18] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J., “LSTM: A search space odyssey”, IEEE transactions on neural networks and learning systems, 28(10), 2222-2232), 2017

[19] Kidziński, Ł., Delp, S., & Schwartz, M., “Automatic real-time gait event detection in children using deep neural networks”, 2019

[20] Goodfellow, I., Bengio, Y., & Courville, A., “Deep Learning” in MIT Press, 2016

[21] Kiranyaz, S., Ince, T., & Gabbouj, M., “1D Convolutional Neural Networks and Applications: A Survey”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021

[22] Zhang, Q., Yang, L. T., Chen, Z., & Li, P, "A Survey on Deep Learning for Big Data" Information Fusion, 42, 146-157, 2018

[23] Khan, F., Ahmad, R., Ahmed, W., Alam, M.M., & Drieberg, M., "Interference and Priority aware Coexistence (IPC) algorithm for link scheduling in IEEE 802.15.6 based WBANs", 2019

[24] Khan, R.; Alam, M.M.; Kuusik, A., "Channel Prediction based Enhanced Throughput and Channel Aware MAC in SmartBAN Standard", in 16th International Symposium on Wireless Communication Systems, 2019

[25] Saboor, A. et al., "Latest Research Trends in Gait Analysis Using Wearable Sensors and Machine Learning: A Systematic Review", in IEEE, 2020

[26] ESP32 Series Datasheet Version 4.5,

[https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf]

[27] MPU-6000 and MPU-6050 Product Specification Revision 3.4,

https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf

[28] Arduino TensorflowLite ESP32 Library

https://github.com/tanakamasayuki/Arduino_TensorFlowLite_ESP32

[29] EloquentTinyML Library, https://github.com/eloquentarduino/EloquentTinyML

[30] PlatformIO, https://platformio.org/

[31] TensorFlow Lite Builder with CMake,

https://www.tensorflow.org/lite/guide/build_cmake

[32] Github Issue to request missing TensorFlow operations to TFLite

https://github.com/tensorflow/tensorflow/issues/21526

[33] Jakob Rostovski, Mohammad Hasan Ahmadilivani, Andrei Krivošei, Alar Kuusik & Muhammad Mahtab Alam, Real-Time Gait Anomaly Detection Using 1D-CNN and LSTM, May 2024