NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

RICARDO CAMACHO QUADRIO DA CUNHA MONTEIRO

Master in Electrotechnical and Computer Engineering

BASKETBALL MOTIONS RECOGNITION AND INJURY PREVENTION

APPLICATION OF A NON DISRUPTIVE SMART WEARABLE IN BASKETBALL

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

SPECIALIZATION IN SPECIALIZATION NAME

NOVA University Lisbon

Draft: March 30, 2026

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

BASKETBALL MOTIONS RECOGNITION AND INJURY PREVENTION

APPLICATION OF A NON DISRUPTIVE SMART WEARABLE IN BASKETBALL

RICARDO CAMACHO QUADRIO DA CUNHA MONTEIRO

Master in Electrotechnical and Computer Engineering

Adviser: Anikó Katalin Horváth da Costa
Assistant Professor, NOVA University Lisbon

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING
SPECIALIZATION IN SPECIALIZATION NAME

NOVA University Lisbon
Draft: *March 30, 2026*

Abstract

Standard sports wearables rely predominantly on kinematic data from Inertial Measurement Units (Inertia Measurement Units (IMUs)), missing the physiological context of the muscles to classify complex, high-intensity athletic movements. Addressing this limitation, this dissertation presents the design, implementation, and evaluation of a multi-modal wearable ecosystem tailored for basketball analytics. The custom hardware architecture fuses 9-axis IMU kinematics with Electromyography (Electromyography (EMG)) sensors' data transmitting via Arduino Bluetooth Low Energy (BLE) microcontrollers at 50Hz.

To process this high-frequency data, a custom Android application was engineered to serve as a decentralized edge-computing hub. The software synchronizes the bilateral sensor streams into a 300ms sliding window and feeds a dual-tier machine learning pipeline: a cloud-based Random Forest (Random Forest (RF)) classifier for deep post-session analysis, and a quantized TinyML neural network executing locally for ultra-low latency inference.

Empirical evaluation mathematically validated the core sensor fusion hypothesis. Feature importance analysis revealed that localized EMG variance and Root Mean Square (RMS) outranked standard accelerometer kinematics as the primary contributors for differentiating similar basketball movements. Ultimately, the successful deployment of Edge Machine Learning (ML) completely eliminated network-induced latency, enabling instantaneous, active injury prevention alerts. This research demonstrates that harmonizing mechanical sensors, electrical physiology, and embedded machine learning provides a good approach to safeguarding athlete health.

Keywords: Sports Wearables · Sensor Fusion · Electromyography (EMG) · Inertial Measurement Unit (IMU) · Edge ML · TinyML · Machine Learning (ML)

Resumo

Independentemente da língua em que a dissertação está escrita, geralmente esta contém pelo menos dois resumos: um resumo na mesma língua do texto principal e outro resumo numa outra língua.

A ordem dos resumos varia de acordo com a escola. Se a sua escola tiver regulamentos específicos sobre a ordem dos resumos, o template (LaTeX) NOVAthesis LaTeX (novathesis) irá respeitá-los. Caso contrário, a regra padrão no template novathesis é ter em primeiro lugar o resumo *no mesmo idioma do texto principal* e depois o resumo *no outro idioma*. Por exemplo, se a dissertação for escrita em português, a ordem dos resumos será primeiro o português e depois o inglês, seguido do texto principal em português. Se a dissertação for escrita em inglês, a ordem dos resumos será primeiro em inglês e depois em português, seguida do texto principal em inglês.

```
\abstractorder(<MAIN_LANG>):={<LANG_1>,...,<LANG_N>}
```

Por exemplo, para um documento escrito em Alemão com resumos em Alemão, Inglês e Italiano (por esta ordem), pode usar-se:

```
\ntsetup{abstractorder={de={de,en,it}}}
```

Relativamente ao seu conteúdo, os resumos não devem ultrapassar uma página e frequentemente tentam responder às seguintes questões (é imprescindível a adaptação às práticas habituais da sua área científica):

Palavras-chave: Primeira palavra-chave · Outra palavra-chave · Mais uma palavra-chave · A última palavra-chave

Contents

Acronyms **ix**

1 Introduction **1**
1.1 Objectives 2
1.2 Motivation 2
1.2.1 Document Structure 2

2 State Of The Art **4**
2.1 Wearables 5
2.1.1 Applications 5
2.1.2 Wearables in Basketball 7
2.2 Movement Recognition Sensors 7
2.2.1 IMU - Inertial Measurement Units 8
2.2.2 EMG - Electromyography 8
2.2.3 Plantar Pressure Wearables 8
2.3 Restraints 10
2.4 Communication Protocols 11
2.4.1 BLE - Bluetooth Low Energy 11
2.4.2 WiFI 11
2.5 Gait Cycle 12
2.5.1 Basketball Movements 13
2.6 Related Work 15
2.7 Relevant data for analysis 16
2.7.1 Average Pressure 16
2.7.2 Peak Pressure (PP), Pressure Time Integral (PTI) and Mean Peak Pressure (mPP) 16
2.8 Data Analysis 17
2.8.1 Algorithms 17
2.8.2 EMG Features 21

2.9 Injury prevention and Fatigue Monitoring 25
2.9.1 Muscle Cramps and Pre-Cramp Anomalies 26

3 Work Proposal **27**
3.1 Project Description 28
3.1.1 Prototype 28
3.1.2 Data Acquisition 31
3.1.3 Data Processing 31

4 Implementation **33**
4.1 Prototype Hardware 33
4.1.1 Arduino 34
4.2 Arduino Code 34
4.2.1 Calibration 34
4.2.2 Sensors' Data Readings And Feature Calculations 35
4.2.3 Bluetooth Low Energy 36
4.3 Android Application 38
4.3.1 User Interface 38
4.3.2 Arduino BLE Communication 42
4.3.3 Communication with InfluxDB 43
4.3.4 User Database 44
4.3.5 Machine Learning Predictions 45
4.3.6 Cramping Prevention and Fatigue Monitoring 46
4.4 Machine Learning Code 47
4.4.1 Models' Training 47
4.4.2 Random Forest 47
4.4.3 TinyML 48
4.4.4 Model Evaluation 48

5 Verification and Testing **50**
5.1 Data Collection 50
5.1.1 Prototype Placement and Configuration 50
5.2 Machine Learning Model Evaluation 51
5.2.1 Training Metrics and Performance Curves 51
5.2.2 Feature Importance and Sensor Fusion 53
5.2.3 ML Training - Confusion Matrices and Error Analysis 55
5.3 System Performance 55
5.3.1 Overall System Accuracy 57
5.3.2 Real Test - Confusion Matrices and Error Analysis 57
5.4 Discussion of Results 60
5.4.1 Known Problems and System Limitations 60

6 Conclusion **62**
6.1 Summary of Achievements 62
6.2 Future Work 63
6.2.1 Hardware Evolution and Textile Integration 63
6.2.2 Dataset Expansion and Automated Labeling 63
6.2.3 Clinical Validation of Fatigue Algorithms 63
6.3 Final Remarks 64

Bibliography **65**

List of Figures

2.1 Different wearables [5]. 5
2.2 [8, Timeline of sensing technology intertwined with basketball sport]. . . . 7
2.3 The gait cycle of a leg with with a cycle time of 0.92s. 40 ms interval between lines [22]. 12
2.4 The three foot's directions [22]. 13
2.5 Some basketball stances. 13
2.6 K-nearest-neighbours example - adapted from [20] 18
2.7 Neural Network Example 19

3.1 System Overview 28
3.2 Prototype Photos 28
3.3 Arduino Nano BLE Rev 2 - Pinout [44] 29
3.4 BITalino EMG sensor [**<empty citation>**] 30
3.5 Halving the Voltage at the Common Collector (VCC) 30

4.1 Physical System Overview 33
4.2 Android Application Overview 39
4.3 Request message to turn on Bluetooth 40
4.4 Start Menu 40
4.5 Profile's Menu and User Creation 41
4.6 Data Menu 41
4.7 Calibration Menu 42
4.8 Calibration Menu 42

5.1 An athlete wearing the prototype 51
5.2 Gastrocnemius Lateralis Positioning [45] 52
5.3 TinyMLAccuracy 52
5.4 TinyMLLoss 53
5.5 RF ROC Curve 54
5.6 RF Precision Recall Curve 54

5.7 Aggregated Feature Importance Graph 55
5.8 TinyML Confusion Matrix 56
5.9 Random Forest Confusion Matrix 56
5.10 Overall Classification Accuracy: Local TinyML vs Cloud Random Forest (Raw and Smoothed) 57
5.11 Classification Accuracy by Movement 58
5.12 Confusion Matrix for the Random Forest Model before Smoothing 58
5.13 Confusion Matrix for the Random Forest Model after Smoothing 59
5.14 Confusion Matrix for the Raw TinyML Model 59
5.15 Confusion Matrix for the Smoothed TinyML Model 60

List of Tables

4.1 Bluetooth Characteristic Configuration 37

Acronyms

BLE Bluetooth Low Energy *(pp. i, iv, vi, 10, 28, 29, 33–39, 41–45, 50, 61, 62)*

EMG Electromyography *(pp. i, vi, 1, 2, 7, 8, 21, 23, 24, 26, 28–31, 33, 35–37, 43, 44, 46, 48, 50, 51, 53, 55, 60, 61, 63, 64)*

EU European Union *(p. 51)*

GATT Generic Attribute Profile *(pp. 42, 43)*

GND Ground *(p. 29)*

IMU Inertia Measurement Unit *(pp. i, 2, 7, 8, 28, 34, 35, 37, 41, 50, 53, 55, 60, 61)*

IoT Internet of Things *(pp. 50, 62)*

MAV Mean Absolute Value *(pp. 25, 26, 31, 36)*

ML Machine Learning *(pp. i, iv, 2, 3, 19, 21, 27, 32, 35, 36, 40, 45, 48, 50, 51, 55, 62)*

NN Neural Network *(p. 48)*

novathesis NOVAthesis LaTeX *(p. ii)*

RF Random Forest *(pp. i, 47, 48, 53, 57, 58)*

RMS Root Mean Square *(pp. i, 25, 26, 31, 36, 46, 55, 60, 61)*

sEMG Surface Electromyography *(pp. 25, 26)*

SQL Structured Query Language *(p. 44)*

SSC Slope Sign Changes *(pp. 25, 26, 31, 36)*

USB Universal Serial Bus *(p. 61)*

VAR Variance *(pp. 31, 36, 55, 60)*

VCC Voltage at the Common Collector *(pp. vi, 29, 30)*

VOUT Output Voltage *(p. 29)*

WL Waveform Length *(pp. 31, 36)*

ZC Zero Crossing *(pp. 25, 26, 31, 36)*

1

Introduction

The constant pursuit of efficiency and durability extends not only to technology but also to the human body. Wearable technology is an example of this research and its popularity is increasing over time. These devices have a great wide-range of functions that can help in medical diagnosis, injury prevention, workout detection, sports performance, sleep analysis, and much more.

In recent years, Wearables have advanced significantly in every possible way. These devices, when paired with smartphones, have revolutionized how we as a society live our day to day lives and how we improve sports. It's no secret that statistics play a big part in today's sports landscape and this it is only getting more complex with the progression of time.

Compared to single-activity recognition, team sports present greater challenges for analytics due to their unpredictability, competitiveness, and constant interaction between players. They are characterized by a lot of high intensity short periods of time and stress induced by some game situations specially at game climax [1].

Basketball, through the global reach of the NBA, has become more popular increasing its visibility and data availability. The most common method of motion capture is via video recording, but high-quality systems are often expensive and only accessible to teams with high budgets [2]. The NBA uses SportVU, a system composed of several high-definition 3D cameras capable of recording the entire court which was introduced in 2013 [3].

This project uses two devices composed by an Arduino and an EMG sensor to collect data. This is an initial design that can be improved by implementing these sensors into an insole or a sock making it as comfortable and as unobtrusive as possible. An android application will be developed as an interface for the user and the database used.

Although other devices may offer higher precision or additional data types, basketball regulations only allow unobtrusive devices that can be concealed and do not interfere with gameplay or player safety.

1.1 Objectives

The primary objective of this dissertation is to design, implement, and evaluate a wearable system tailored for high-intensity sports analytics, specifically focusing on basketball mechanics. Moving beyond the limitations of traditional commercial fitness trackers that rely predominantly on passive kinematic data logging, this project aims to create an active, real-time monitoring ecosystem capable of simultaneous movement classification and localized fatigue detection. In other words, develop everything from the software of the arduinos themselves and the interface to the database management.

The research seeks to prove that combining standard IMU with EMG sensors provides a fundamentally richer, multi-dimensional dataset by collecting data with the developed system and training capable ML models. Another major goal is to demonstrate the viability of a lightweight Edge model deployed onto a mobile device and compare it to a "normal" ML model ran on a cloud service.

Ultimately, the objective is to equip athletes and coaches with a wearable tool that not only categorizes athletic performance, but actively safeguards player health by pre-emptively identifying hypertensive muscle activity before cramping occurs.

1.2 Motivation

I've been playing basketball since I was five years old and I've never lost my passion for the sport. As a player that has dealt with cramping before, I know that after it happens, you need a lot of rest and massaging to get back to a proper state and that takes time. It is very difficult to keep plating at a high level after cramping. With the opportunity of combining both my studies and this hobby, I knew I had to research it and contribute to the betterment of these technologies.

1.2.1 Document Structure

The remainder of this dissertation is organized into five main chapters, each detailing a specific phase of the research, development, and evaluation of the proposed wearable system:

- **chapter 2: State Of The Art** - Reviews the existing literature and technological landscape regarding sports wearables, specifically tailored to basketball. It explores the foundational concepts of movement recognition sensors (such as IMU and EMG), wireless communication protocols, the biomechanics of the gait cycle, and ML algorithms.

- **chapter 3: Work Proposal** - Outlines the architecture of the proposed solution. It defines the core project description, the initial design of the hardware prototype, and the methodologies planned for high-frequency data acquisition and processing.

- **chapter 4** - Provides a comprehensive, highly technical breakdown of the system's development. This chapter details the embedded firmware written for the Arduino microcontrollers, the architecture of the Kotlin-based Android application, the integration with the InfluxDB database, and the complete development for both the cloud and edge ML models.
- **chapter 5** - Presents the evaluation of the developed system. It details the physical data collection protocol and rigorously assesses the ML models using metrics such as feature importance and confusion matrices. The chapter concludes with a critical discussion of the system's real-world performance and an analysis of known limitations.
- **chapter 6** - Summarizes the achievements of the dissertation. It revisits the initial objectives and proposes concrete avenues for future work.

2

State Of The Art

2.1 Wearables

A Wearable is any smart device with sensors that can be worn, embedded in clothing, accessories or in the body or even adhered to or tattooed on the skin. There are many different types of Wearables such as Smart Socks, Smart Insoles, Smartwatches, Smart Rings amongst others. These devices are able to measure many different types of data such as position, pressure, brightness, velocity, heart rate and many others [4].

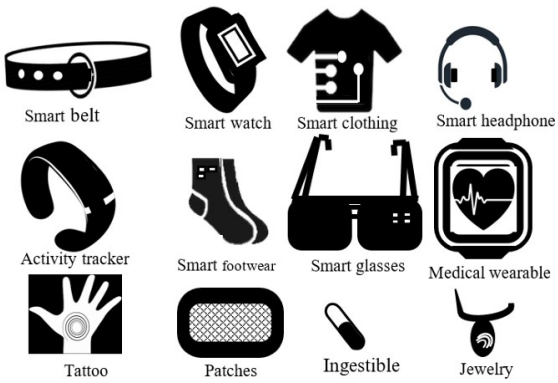


Figure 2.1: Different wearables [5].

Nowadays there are many different wearables that can collect the same type of data. With this in mind, the choice for the right device doesn't only revolve around what type of information it can collect. Portability, processing power, accessibility, battery life are some other important characteristics.

2.1.1 Applications

Wearables provide data useful for many applications. Their portability, affordability, easy connection to a mobile phone and non-prescription nature of these devices makes them an easy choice for several uses. Amongst others the main ones are Health, Activity Recognition and Sports and Tracking/Localization [5].

2.1.1.1 Health

In the health sector wearables have gained a lot of importance over the last few years. With smart devices, doctors and patients can get a continuous health monitoring which lead to better diagnostics and disease and injury prevention. The data provided by the devices is also used to design insoles, shoes and other objects that aid health care and injury prevention [6]. Besides this, wearables are also being used by people to track their own health without any prescription.

Construction companies have also been investing in wearables to prevent injuries in their staff when working with heavy loads. In 2020 there were 619 million people affected by lower back pain globally and the number is estimated to increase to 843 million by 2050 [7].

2.1.1.2 Tracking and Localization

Knowing the position of a person or an animal is important in many occasions. From tracking animals to understand their behaviour to simply finding where a lost elder is, this category has many useful applications.

This devices tend to have a great battery life efficiency since they have to last long periods of time without being charged.

2.1.1.3 Activity Recognition and Sports

Even though this section has a lot of overlapping technology with the health segment, its use cases go beyond the health category. Besides the day-to-day monitoring these devices can also recognize and monitor your physical activities.

With information about timings, amount of force, position and others, coaches and players can prevent injuries and improve their skills.

2.1.1.4 Others

Wearables are also used for education, gaming, virtual and augmented reality, law enforcement besides other applications.

2.1.2 Wearables in Basketball

Basketball was created in 1891 by a Canadian physical education professor named James Naismith in Springfield, Massachusetts as an indoor game to avoid the rain.

Sensors initially became part of the game through the ball manufacturing process in the 1950s to improve safety and standardization of its pressure. In the 1980s the players started using blood oxygen and hear rate monitors in their training to improve strength and conditioning. The development of image sensors led to the recording and live broadcasting of basketball games in the 1990s and the 2000s. Displacement and acceleration sensors starting being used in the 2010s to monitor the players' shooting form and other movements. As mentioned in section 2.1 wearable technology is rapidly evolving and it is the next big step for sports statistics and performance improvement. [8].

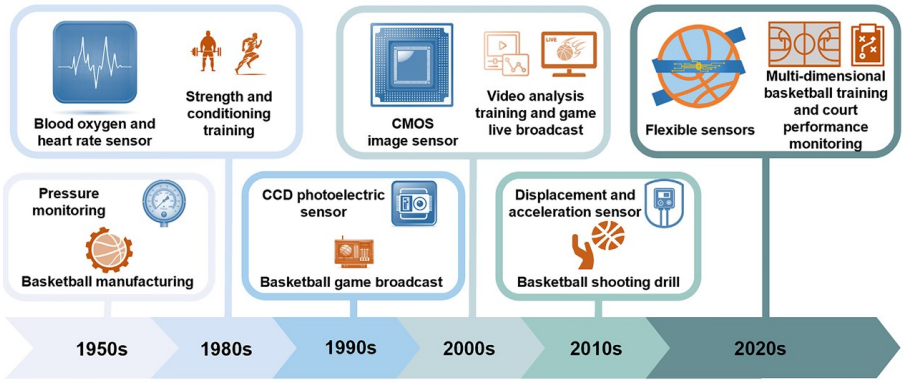


Figure 2.2: [8, Timeline of sensing technology intertwined with basketball sport].

Dave Brailsford popularized the concept "Marginal gains" which consists in improving every single detail, no matter how small in order to get better. The biggest obstacle in this philosophy is quantifying and identifying what we can improve. The answer to this distress are Wearables [9].

For this purpose the technology used consists mostly of micro-electromechanical systems such as inertial measurement units (Inertia Measurement Unit (IMU)'s), networked sensor suites and exoskeletons. Risk assessment shares some of its data with performance enhancement devices but it also works with pressure sensors and IMU's that host several sensors (e.g., accelerometers, gyroscopes, and magnetometers) [10].

2.2 Movement Recognition Sensors

There are many different technologies capable of capturing similar data through different mechanisms. For acquiring movement related data the usual methods are through an IMU, image capture or some sort of pressure sensors.

Another way to collect information about player movements is through the muscles, for example, with an Electromyography (EMG) sensor.

2.2.1 IMU - Inertial Measurement Units

An IMU is a sensor system design to capture motion and orientation usually consisting of an accelerometer, a gyroscope and sometimes a magnetometer. These components allow the measurement of acceleration, angular rate and magnetic field respectively. When combined, they provide a good estimation of position, orientation and movement patterns. However these approximations suffer from drift over time, meaning that small errors accumulate and lead towards increasingly inaccurate readings. Due to its low cost, compact size, high sample rate and versatility, IMUs are especially valuable for wearable devices for activity/movement recognition practices.

2.2.2 EMG - Electromyography

EMG is a technique for evaluating and measure electrical activity generated by muscles. With the generated data researchers and practitioners can assess the muscles contribution to different movements, evaluate neuromuscular coordination and identify abnormal patterns that may indicate fatigue, compensation or injury risk. There are two main types of EMG:

- Needle EMG - A technique commonly used in neurology that where fine-wire electrodes are inserted into the muscle tissue leading to very precise recordings but is very invasive.
- Surface EMG - A non-invasive technique where the electrodes are placed over the skin.

2.2.3 Plantar Pressure Wearables

Plantar pressure wearables are innovative devices designed to monitor and analyze the distribution of pressure across the soles of the feet. By embedding sensors into insoles, smart socks, or even shoes, these wearables capture detailed data on foot loading patterns, which is critical for understanding gait, balance, and overall foot health. Plantar pressure data offers valuable insights for applications in sports performance, injury prevention, rehabilitation, and the management of conditions such as diabetes, where pressure distribution can indicate areas at risk for ulcers or other complications. They are also useful for better footwear designs or orthosis [11–14].

For gait monitoring through pressure there are several different types of sensors used in wearables that can be separated in five different categories [15]:

1. Capacitive Sensors - Consisting in two conductive plates separated by an insulating layer. When force is applied, they develop a voltage variation.
2. Resistive Sensors - These are the most common sensors. Their resistance varies with the applied pressure.

3. Optoelectronic Sensors - Composed by a transmitter and a receiver (a laser or a LED and a photodiode). When a force is applied, the light that reaches the photodiode varies proportionally.

4. Piezoresistive and Piezoelectric Sensors - These sensors use the piezoelectric effect. They convert the changes in applied pressure to electrical current.

As of today there are four different kinds of devices that can measure plantar pressure [15]:

1. Platform Systems - Known as the most reliable method for measuring plantar pressure, platform systems offer high accuracy and detailed spatial resolution. Their main limitation is a lack of portability, restricting their use to controlled environments like labs and hospitals.

2. In-Shoe Systems: These systems provide greater mobility and flexibility, with optimized circuit design, power efficiency, and communication technology, and they're more affordable than platform systems. They record plantar pressure inside the shoe and allow for extensive data collection across multiple steps, enabling long-term gait analysis indoors and outdoors. However, they are less accurate than platform-based measurements.

3. Smart Wireless Insoles: Unlike platform and in-shoe systems, smart wireless insoles are cordless, eliminating the need for wires to connect sensors or data acquisition units. While in-shoe systems aren't ideal for extended outdoor use, smart insoles can be used both indoors and outdoors. They often include Bluetooth or WiFi for data transmission and a power source. However, the added elastic layer they create inside the shoe can be several millimetres thick, potentially distorting the accuracy of foot pressure data. They are also relatively expensive and not suited for daily wear.

4. Smart Socks: These fabric-based systems integrate sensors directly into the textile, which avoids the challenges of the previous systems. However, they require handmade or complex manufacturing techniques, which is a drawback.

In basketball, smart insoles and smart socks are particularly well-suited wearables due to their portability, flexibility, and adaptability to both indoor and outdoor environments. Unlike larger, stationary measurement systems, these wearable devices are attached to the athlete, providing continuous and real-time data regardless of the setting.

Smart insoles and socks, when embedded with pressure sensors, can track a player's foot pressure distribution, stability, and impact forces with each step, jump, or landing. This information is crucial for basketball players, as it helps to analyze their movement patterns, improve balance, reduce injury risk, and enhance overall performance. The insole's thin profile and ability to fit inside regular shoes also ensure they don't interfere

with the player's comfort or mobility. Similarly, smart socks, which incorporate sensors directly into the fabric, offer flexibility and comfort while capturing dynamic foot pressure and movement data. Their lightweight design and close contact with the foot provide detailed insights into the nuances of foot pressure changes during the quick movements in basketball. Both smart insoles and socks also benefit from wireless connectivity, allowing data to be transmitted to a smartphone or tablet, enabling real-time monitoring and analysis. This flexibility and data accessibility are particularly valuable in basketball training and performance analysis, as players and coaches can assess and adjust techniques in real time, whether in an indoor or outdoor court.

2.3 Restraints

Wearables have to be light and comfortable since they should be unnoticeable and not disturb the user's activities. This means that these gadgets have low processing capacities and their sensors monitor the data at lower frequencies then non wearable devices. Due to their size they often use miniature sensors that can be sensitive to placement and alignment. Inaccurate placement, external interference, or device movement can cause errors, reducing the reliability of collected data. Additionally, in applications where precise measurements are essential, the smaller sensors in wearables can't match the accuracy of larger, lab-based equipment.

Wearables are often exposed to a wide range of environmental conditions, including sweat, water, dust, and physical impacts. Ensuring these devices remain durable and function properly under such conditions is a challenge, as components like sensors and circuits need to be robust enough to withstand daily use and environmental exposure.

Due to their lack of memory and processing power, the data collected from the devices has to be sent to another device for storage or further processing. The main communication protocols with low energy consumption are Bluetooth Low Energy (BLE) and WiFi for close data transmission. These methods can be impacted by limited range, interference, and connectivity issues, especially in crowded or signal-dense environments, which can disrupt real-time data streaming.

Since these devices are quite small, one of their main restraints is the battery life. With every action taken by the device, whether it's collecting data with a sensor, processing that data or sending it somewhere else, energy is always spent. Depending on their purpose, wearables may have to last up until one year without being charged [16]. To overcome this challenge some devices rely on energy harvesting components to charge the device such as solar power, motion energy, etc [12].

Minding the others problems already mentioned, wearables have low security so they are relatively easy targets due to their poor encryption and protection [5]. Due to the lack of privacy of the athlete's data and since wearable technology can possibly obstruct and injury someone, wearables still are'nt accepted in many sports (in federated play),

basketball included. Even tho they are't allowed in games, there aren't any rules for practices and there are some teams that are taking advantage of these devices [17, 18].

2.4 Communication Protocols

Wearable technology relies heavily on efficient communication protocols to ensure seamless data exchange between devices, applications, and networks. These protocols define the rules for data transmission, enabling wearables to monitor health, track activity, and connect with other devices in real time. Given the constraints of wearables, communication protocols play a critical role in balancing performance with power efficiency. For this project two kinds of protocols will be discussed: Bluetooth Low Energy and WiFi.

2.4.1 BLE - Bluetooth Low Energy

BLE is a short-range communication technology known for its low power consumption. Classified as a Wireless Personal Area Network (WPAN), BLE is widely used in various wearable applications, from diabetes monitoring devices to smartwatches. The original insole design employed BLE for communication between the insole and the paired mobile application. BLE's primary design objectives are to enable low-power communication, particularly for peripheral devices. It is optimized for affordability and simplicity, aligning well with many developer requirements and contributing to its swift adoption. The main difference between BLE and the classic Bluetooth is their transmission rates. BLE transmits at around 2000 Kbps while Bluetooth Classic transmits at around 2-3 Mbps making its energy consumption around two times bigger than BLE. This is the reason BLE is the usual choice for IOT devices. Besides this difference BLE works the same way as Bluetooth Classic [19, 20].

2.4.2 WiFI

WiFi is a widely used wireless Local Area Network (LAN) technology, primarily aimed at connecting mobile devices to the internet. Its broad adoption means it's available in most homes, businesses, and many public spaces. WiFi uses variations of the IEEE 802.11 protocol depending on the generation, with the latest generation, WiFi 6 (802.11ax), approved in September 2024. Operating on the 2.4 GHz or 5 GHz frequency bands, WiFi 6 also supports wider channels, optimized mainly for video streaming. WiFi delivers high data speeds and easy setup for access points, making it a mature and well-proven technology. However, its range is somewhat limited indoors due to high signal absorption, though range improves in open, line-of-sight environments. Due to its relatively high power consumption, WiFi is not ideal for direct communication in IoT devices [19, 20].

2.5 Gait Cycle

The Gait Cycle is the sequence of movements that one leg takes in each step in walking or running. It is the whole moving process from the first point of contact between a foot and the ground until it reaches the ground again (see figureFigure 2.3) [19, 21].

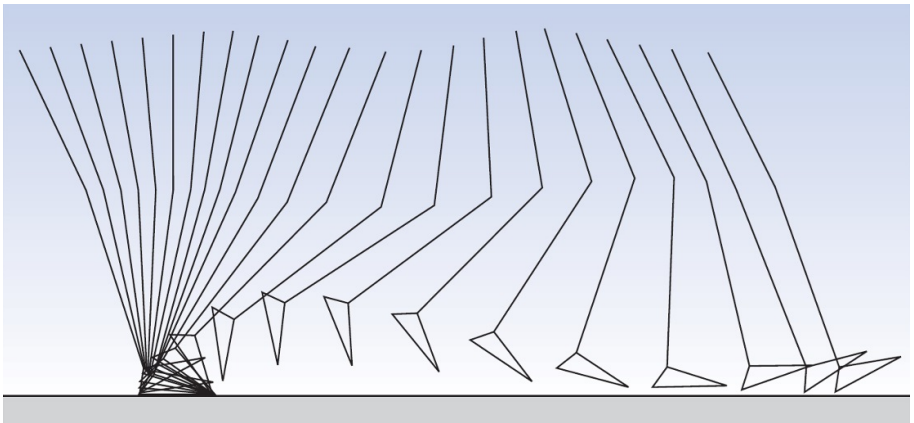


Figure 2.3: The gait cycle of a leg with with a cycle time of 0.92s. 40 ms interval between lines [22].

The usual method for measuring Gait Cycle is with a platform and a multi-camera setup since it is the most precise and reliable. However, this method is very expensive and can only be applicable indoors in a secure environment. An insole or a smart sock aren't as reliable but they compensate by being cheaper, more comfortable for the user and can be worn in their activities.

Although the Gait Cycle varies from person to person (minding age, body type and others) and even in each step each person takes, there are many tendencies for normal gait. With this in mind these patterns can be used to predict and understand what forces are applied in a gait cycle [19].

There are three foot's directions(see figureFigure 2.4): Vertical which is the direction perpendicular to the foot's palm. Fore-aft that follows the line between the toe to the heel. And the lateral direction which goes from one side of the foot to the other [19].

There are two different stages for this motion in slower gaits and an extra stage in faster gaits [23]:

- Stance Phase - time period when the foot is on the ground. This phase represents about 60% of the normal gait cycle.
- Swing Phase - when the foot is in the air. It normally represents the remaining 40% of the gait cycle.
- Flight Phase - this phase only occurs in faster gaits such as running. It refers to the moments where neither foot is in contact with the ground.

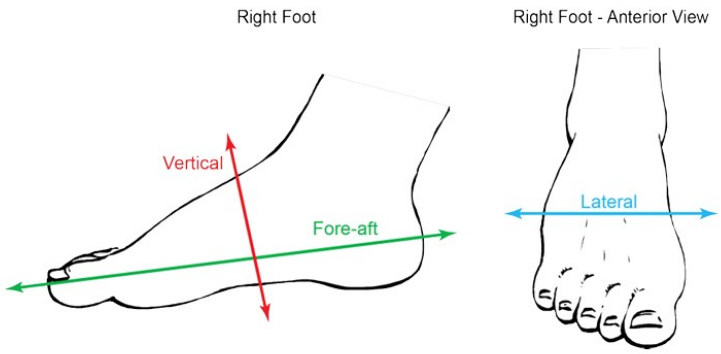


Figure 2.4: The three foot's directions [22].

2.5.1 Basketball Movements

Basketball involves many other distinct movements besides gait. These motions are very complex making them difficult to accurately identify but they are essential for injury prevention and for the performance enhancement analysis. In this project data will be collected from three groups of movements with increasing complexity.

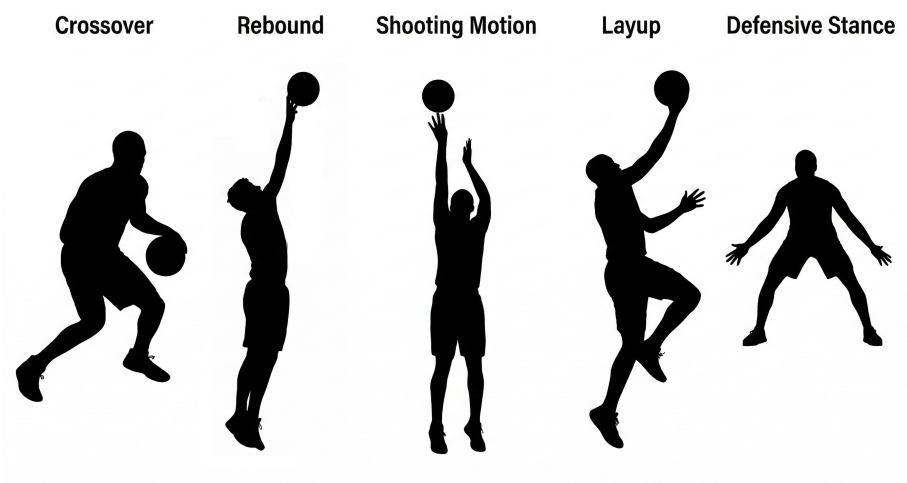


Figure 2.5: Some basketball stances.

2.5.1.1 Basic Movements

- Sprint(fast gait) - Basketball players frequently sprint down the court, which involves a pattern of quick, alternating footfalls with consistent gait cycles on each foot.
- Defence Slide/Stance - The defensive slide requires the player to move laterally while maintaining a low stance.
- Rebound Jump - Rebounding requires explosive jumping to reach the ball, sometimes with minimal preparatory movement, other times with a box out.

2.5.1.2 Intermediate Movements

- Pivot - A pivot involves planting one foot firmly while the other foot moves to change direction.
- Crossover Dribble - In a crossover dribble, the player shifts their weight rapidly from one foot to the other, using pressure changes to decelerate in one direction and quickly accelerate in the opposite direction.
- Change of Direction in Defence - A player changes his direction while sliding.
- Boxout - The act of securing a positing usually near the basket to deny the rebound for the opponent.
- Screen - Similar to the boxout but it is done anywhere on the court to force a defensive player to change his path.
- Layup - A layup is normally a two step shot and it requires a step-by-step increase in pressure, usually peaking in the final step before jumping.
- Cutting - Cutting involves a sharp change in direction with a great speed shift. It is an off ball movement.

2.5.1.3 Advanced Movements

- Jumpshot - The act of shooting the ball. The jump shot is a fundamental shooting technique in basketball, characterized by a quick dip in pressure as the player lowers into a squat before takeoff, followed by a burst of force as they push off the ground to jump.
- Eurostep - This movement is a specific type of layup where the player takes the first step in a direction and the second step to another direction.
- Step Back - The step back is a Jump Shot whe the player takes a step back before shooting the ball.
- Failed Landing (Incomplete layup) - Any type of fall characterized by the sudden fall of pressure and following.

2.6 Related Work

There are many studies on the topic of measuring plantar pressure through an insole. Most of them mostly study the gait cycle while running or walking with similar objectives. Baitong Wang et al. [21] developed an insole with 8 pressure sensors and motion track sensors and with this system managed to clearly identify gait with only the pressure sensors.

Elizabeth S. Chumanov et al. [24] tracked the global position of the pressure on the soles of the feet throughout steps in walking and running. This data can be used for improvement in foot-floor contact models used to simulate human locomotion and in least squares inverse dynamics.

For sports like basketball, the available motion recognition research was done through video recording or smart wearable devices such has wristbands. There is very few studies on the use of insoles for this purpose.

Thomas Holleczek et al. [25] design a pair of smart socks with three pressure sensors each for the use case of snowboard. Even tho it isn't a sport as dynamic and unpredictable as basketball, it still is very relevant for this project. They worked on activity recognition focusing on recognizing if the user was riding backside or riding frontside (with his back to the bottom of the hill or the top respectively). It is pointed that the recognizability of three basic standing postures was explored and that they were distinguished with perfect accuracy mentioning that it can be interesting for gait analysis.

Qing F. Zhou et al. [2] utilized insoles with a motion sensor with the purpose of getting a high accuracy for recognizing three basketball motions: Dribbling, jumping and turning around. For this data processing the K-means algorithm was utilized. Two years later they continued their work and managed to identify five new different footwork movements. In this version a multiple sensor system was proposed and three different classification methods were discussed [3]: K-Nearest neighbours (KNN), Support vector machine (SVM) and Convolutional Neural Networks (CNN).

Catarina Amaro et al. [11] studied the differences in five different movements in basketball between two different moments in a season. The study presented no significant changes in mean peak pressure between the two different times but it did specify that experienced players had lower plantar pressure values in their movements and lower symmetry index. This means that there is room for improvement based on the values read for the less experienced players. It was also mentioned that wear of the shoes may have influenced the slight changes observed in the study. Each player wore their own shoes and this was pointed as a variable for the plantar pressure measured.

2.7 Relevant data for analysis

Wearable devices capture large volumes of raw data, often gathering detailed metrics like pressure distribution, movement patterns, and force dynamics. However, to extract meaningful insights, this raw data must first be transformed into a more organized, manageable format suited for analysis. Preprocessing includes filtering out noise, aligning data with relevant time frames, and structuring it into clear categories or features, which simplifies identifying trends and patterns. By converting this data into a cleaner, standardized format, we can more effectively apply analysis techniques, uncovering insights into biomechanics, performance, and even long-term health outcomes.

2.7.1 Average Pressure

Foot therapists currently rely on average pressure to determine if a support sole is effective and where additional relief might be needed. This average provides an interpretable view of pressure distribution during the gait cycle. However, averaging all values across all frames would dilute important information because of the many zero values present throughout the cycle. To avoid this, only non-zero values are averaged for each sensor. Yet, this approach introduces another challenge: a cell with few non-zero readings could appear similar to one with many readings. To improve clarity, cells are excluded from analysis if they don't exceed a minimum threshold of non-zero measurements for that sensor[19].

2.7.2 Peak Pressure (PP), Pressure Time Integral (PTI) and Mean Peak Pressure (mPP)

When assessing plantar pressure, the primary parameters of interest are peak pressure (the highest recorded plantar pressure), pressure time integral (typically defined as the area beneath the peak pressure-time curve), and mean peak pressure (mPP). However, studies have shown that mPP and PTI are closely related, making it unnecessary to measure both.

When there are differences in these measurements between the feet, it can mean that there is an inadequate distribution of forces and it can lead to injury. So a symmetry index defined by the next equation that compares the plantar pressure on both feet will be calculated.

$$SI = \frac{2 \cdot |R - L|}{(R + L)} \times 100 \tag{2.1}$$

If SI is zero then the feet are doing equal work, if it's under 10% it is acceptable, otherwise it means the movement is causing more distress in one leg than the other leading to a loss of bone mass density in the respective leg and joints [11].

2.8 Data Analysis

By leveraging advanced data analysis techniques, such as machine learning algorithms and statistical modelling, pressure insole data can reveal subtle patterns and correlations that inform training adjustments, injury risk, and performance improvements. There are many different algorithms with different objectives. In this section some of them will be explained:

2.8.1 Algorithms

2.8.1.1 K-nearest-neighbours classification

The K-nearest-neighbours is one of the simplest supervised machine learning algorithms. It's a classification algorithm which uses proximity to assigning each point to the class with the most similar elements within the K-nearest-neighbours [20, 23].

As an example observe figure 2.6. If someone would have to assign a class (green triangles, blue squares or red circles) to the cross in the figure, they would probably say it belongs to the green triangles since they are the closest objects. But if instead someone had to assign a class for the star, it would be a much harder task with a naked eye. In this case two factors make that decision: Which are the closest objects (depending on the weight k) and how many of each class are there in that group [20]. In other words imagine a small circle with its center on the object that we want to classify and if it isn't touching other know objects imagine it growing until it does.

2.8.1.2 Dynamic Time Warping

The dynamic Time Warping (DTW) algorithm transforms unsynchronized time series or time series with different lengths allowing a dynamic comparison between them. This algorithm calculates the best possible alignment between two different series dependent on time providing a measurement of how well aligned (or not) they are [20].

2.8.1.3 Regression analysis

Regression analysis' purpose is to find a continuous outcome variable, usually y, through one or more predictor variables, normally x (if only one). The simplest of these models is the linear regression model that relates a linear relationship between x and y. This equation can be expressed as [20]:

$$y = \beta + \alpha \cdot x + \epsilon \tag{2.2}$$

In this equation y is the output variable, β is the starting point (x=0), α is the regression coefficient associated with the predictor variable x and ϵ is the residual error. An extended version of this equation with n predictor variables is:

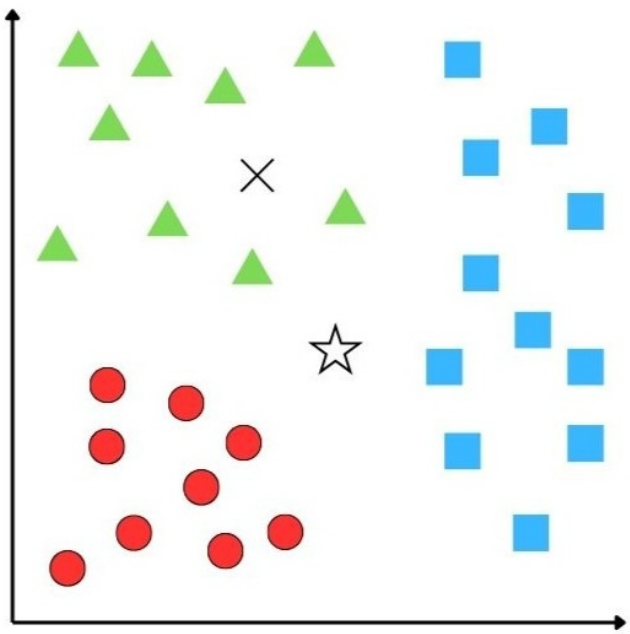


Figure 2.6: K-nearest-neighbours example - adapted from [20]

$$y = \beta + \alpha_1 \cdot x_1 + \alpha_2 \cdot x_2 + \alpha_n \cdot x_n + \epsilon \tag{2.3}$$

2.8.1.4 Recursive Feature Elimination

Recursive Feature Elimination (RFE) selects features by progressively narrowing down the set of available features. Each feature is assigned a weight, and those with the lowest absolute weights are removed from the set. In each iteration, a single feature is removed, and this process continues on the reduced set until the target number of features is reached [23].

2.8.1.5 Principal Component Analysis

Principal Component Analysis (PCA) reduces the dimensionality of data by applying a linear transformation based on singular value decomposition. This transformation projects the data into a lower-dimensional space, creating a new set of variables that are ordered by their importance [23].

2.8.1.6 Linear Discriminant Analysis

The Linear Discriminant Analysis (LDA) model fits a Gaussian distribution to each class, assuming a shared covariance matrix across all classes. This model reduces input dimensionality by projecting data onto the directions that best separate the classes. Its goal is to enhance separation between classes by maximizing the scatter between them while minimizing scatter within each class [23, 26].

2.8.1.7 Neural Network - NN

A NN is a Machine Learning (ML) model that imitates the structure and functions of biological neural networks. A NN utilizes many connected nodes called "artificial neurons" organized by layers where each node connects to others from the previous or the next layer (Figure Figure 2.7). Each of these connections have corresponding weights determined trough the training algorithm. For this process the data has to be labelled.

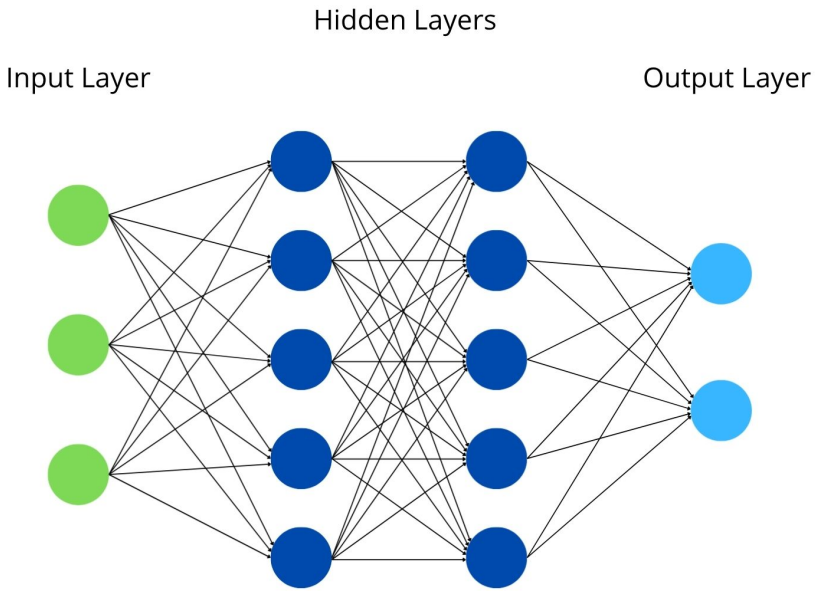


Figure 2.7: Neural Network Example

There are three types of layers: input, hidden and output. A NN must have one layer of each, though it may contain multiple hidden layers. When a network has two or more hidden layers it is considered a Deep Neural Network (DNN). This ML technique is highly effective for analysing large datasets with non-linear relationships. However, toa achieve accurate and precise results it requires a substantial amount of complete data, as missing entries can significantly affect performance. Despite its effectiveness, one major limitation of neural networks is their lack of interpretability, making it difficult to clearly understand the underlying classification boundaries [26].

2.8.1.8 Convolutional Neural Network - CNN

A Convolutional Neural Network is a sort of Deap NN designed to process spatial or temporal structures. These models, before the fully connected layers, use convolutional filters to automatically extract local features, reducing the number of parameters compared with fully connected networks and improving efficiency. CNNs perform well in large-scale classification tasks and require minimal manual feature engineering. However, they often need substantial computational resources and can be difficult to interpret. Despite these limitations, CNNs remain one of the most effective tools for pattern recognition in modern machine learning [27].

2.8.1.9 Fuzzy Logic - FL

Fuzzy Logic is a form of multi-valued logic in which its values are real and range between 0 and 1. Providing a mathematical framework capable of incorporating and balancing both precision and significance in decision-making processes.

A FL system operates through two main processes: fuzzification and defuzzification. Fuzzification converts crisp input values into fuzzy values by assigning them degrees of membership within a fuzzy set, typically represented by an arbitrary curve that defines the relationship between the input values and their corresponding membership levels. A fuzzy set A can be expressed as:

$$A = (x, \mu A(x)) | x \epsilon U \tag{2.4}$$

where A is the fuzzy set, U is the universe of discourse containing the elements x, and $\mu A(x)$ is the membership function.

After fuzzification, the fuzzy sets are evaluated through a series of if-then rules using logical operators to produce an output. This output is then converted back into crisp values through defuzzification, which can be performed using various methods such as the centroid or bisector approaches. The main advantage of FL lies in its flexibility and its ability to be easily modified or combined with other classification techniques. However, FL also presents limitations - its numerous localized parameters and training requirements can make system construction and tuning a time-consuming process [26].

2.8.1.10 Neuro-Fuzzy Hybridization - NF

NF combines the principles of FL and NN creating an intelligent hybrid model that merges human-like reasoning with adaptive learning. Typically, a NF system is built upon a fuzzy logic framework and trained using data-driven methods from NN theory. During training, it adjusts local parameters based on local data and represents knowledge as fuzzy rules, allowing initialization with or without prior information. This hybrid approach leverages the strengths of both processes, interpretability from FL and learning ability from NN,

while reducing their main drawbacks such as dependence on prior knowledge and high computational demands [26].

2.8.2 EMG Features

The raw EMG signal is very noisy and very high frequency so it generates large volumes of data making it very demanding for direct analysis making it inadequate for real-time processing or ML applications. Therefore, relevant features of the signal must be extracted to capture the essential characteristics of the muscle activity to reduce computational load, improve robustness against noise and features also improve the performance of classification algorithms.

2.8.2.1 Mean Absolute Value - MAV

The Mean Absolute Value is one of the most common and simpler features extracted from EMG signals. It corresponds to the average of the absolute values of the signal over a specific period of time, providing a measurement of the muscle's electrical activity level. Mathematically, MAV is calculated as:

$$MAV = \frac{1}{N}\sum_{i=1}^{N}|x_i| \tag{2.5}$$

where N is the number os samples in the time frame and x_i is the EMG signal at sample i [26–33].

2.8.2.2 Root Mean Square - RMS

The Root Mean Square is another common feature in EMG analysis. As the name implies, it is the square root of the mean of the signal amplitude emphasizing larger signal amplitudes than the MAV. It's defined as:

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2} \tag{2.6}$$

Where N is the number os samples in the time frame and x_i represents the EMG signal at each sample [27–35].

2.8.2.3 Variance - VAR

The Variance of a signal measures how much it fluctuates around its mean, reflecting the degree of variability or dispersion in muscle activation.

Mathematically it is expressed as:

$$VAR = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu)^2 \tag{2.7}$$

Where N is the number of samples, x_i represents each signal value, and μ is the mean of the signal.

It is a useful feature for pattern recognition and classification since it captures the power and intensity fluctuations of the signal helping differentiating the many types of muscle activity [26–28, 32, 33, 36].

2.8.2.4 Waveform Length - WL

The Waveform Length represents the cumulative length of the signal waveform over a given time frame. It reflects both the signal's complexity and amplitude variation, capturing information about the frequency and intensity of the muscle activity.

The WL feature is calculated according to:

$$WL = \sum_{i=1}^{N-1} |x_{i+1} - x_i| \tag{2.8}$$

Where x_i represents the i-th sample of the EMG signal, and N is the total number of samples.

WL is a widely used time-domain feature for EMG-based solutions since it efficiently combines amplitude and frequency information [26–30].

2.8.2.5 Peak Frequency - PKF

Peak Frequency is the frequency where the maximum power occurs in the signal's power spectrum. The PKF is obtained by finding the frequency at which the power spectral density reaches its maximum value:

$$PKF = 2 \max_f P(f) \tag{2.9}$$

Where $P(f)$ is the power spectral density of the signal at frequency f. This feature provides valuable information about the dominant frequency of muscle activity, which is often related to the type and intensity of the contraction. Changes in PKF can indicate muscle fatigue, as the dominant frequency tends to shift towards lower values when fatigue increases. Therefore, PKF is commonly used in EMG analysis to assess muscle performance and fatigue levels [27, 28].

2.8.2.6 Median Frequency - MDF

The Median Frequency corresponds to the frequency where the power spectrum is equally divided in two. This can be translated to the following equation:

$$MDF = \frac{1}{2} \sum_{i=1}^{N} P_i \tag{2.10}$$

Where P_i is the power spectral density of the signal at frequency i, and N is the maximum frequency component considered [26–28, 30].

2.8.2.7 Mean Frequency - MNF

Mean Frequency represents the average frequency of the power spectrum at each frequency component, weighted by the power spectral density.

$$MNF = \frac{\sum_{i=1}^{N} f_i P(f_i)}{\sum_{i=1}^{N} P(f_i)} \tag{2.11}$$

Where f_i represents the ith frequency component, $P(f_i)$ is the power spectral density at that frequency, and N is the total number of frequency bins considered.

MNF provides a good insight into the distribution of power across the frequency spectrum and is often used to assess muscle fatigue, since it tends to decrease as fatigue increases. Unlike MDF (subsubsection 2.8.2.7), which divides the spectrum into two equal halves, MNF reflects the overall spectral balance of the signal [26–28, 30, 31, 36].

2.8.2.8 Integrated EMG - IEMG

Integrated EMG is the sum of the rectified EMG signal over a time interval. This feature quantifies the muscle activation level providing information about the effort and muscle intensity during a movement.

Mathematically, the integrated EMG is defined by:

$$IEMG = \sum_{i=1}^{N} |x_i| \tag{2.12}$$

Where x_i is the EMG signal amplitude at each sample, and N is the total number of samples in the analyzed time period. IEMG is particularly useful for assessing the intensity of muscular activity and comparing activation levels across different tasks or conditions [26, 27, 29].

2.8.2.9 Simple Square Integral - SSI

The Simple Square Integral is a feature that expresses the energy of the EMG signal. The equation below expresses the computation of SSI:

$$SSI = \sum_{i=1}^{N} \left|x_i^2\right| \tag{2.13}$$

Where x_i represents the EMG signal amplitude at the $i^t h$ sample and N is the total number of samples within the analysis window[26, 27, 29].

2.8.2.10 Zero Crossing - ZC

Zero Crossings is a feature that measures the amount of times that the EMG signal changes sign as expressed by the following expression:

$$ZC = \left\{ \sum_{n=2}^{N} sgn(x_i \cdot x_{i-1})(x_i - x_{i-1}) \geq \epsilon \right. \tag{2.14}$$

Where sgn(·) is the sign function

$$sgn(x) = \begin{cases} 1 & \text{if } x > \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{2.15}$$

Where x_i is the EMG signal amplitude at each sample, N is the total number of samples and ϵ is a small positive value used to avoid counting zero crossings caused by minor fluctuations or noise[27].

2.8.2.11 Slope Sign Changes - SSC

The Slope Sign Change feature measures the number of times the slope of the EMG signal changes sign within a given time window. It reflects the signal's frequency and complexity, providing information about the motor unit firing pattern and the muscle's dynamic behaviour. A higher number of slope sign changes generally indicates a more complex or higher-frequency muscle activity.

As with the ZC (subsubsection 2.8.2.10) feature, a small threshold is often used to minimize the effect of noise on the calculation.

Mathematically, the SSC can be expressed as:

$$SSC = \sum_{i=2}^{N-1} f[(x_i - x_{i-1}) \cdot (x_i - x_{i+1})] \tag{2.16}$$

Where

$$f(x) = \begin{cases} 1 & \text{if } x \geq \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

Where, x_i represents the EMG signal amplitude at sample i, and the ϵ is a small positive value used to reduce sensitivity to minor fluctuations or electrical noise [27].

2.8.2.12 Willison Amplitude - WAMP

Willison Amplitude is a time-domain feature that assumes the number of times the absolute difference between two consecutive signal samples exceeds a predefined threshold. It is particularly useful for capturing changes in muscle activation intensity while remaining computationally efficient[27, 32, 33, 35, 36].

Mathematically, it is defined by:

$$WAMP = \sum_{n=1}^{N-1} f(|x_i - x_{i+1}|) \tag{2.18}$$

Where

$$f(x) = \begin{cases} 1 & \text{if } x \geq \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{2.19}$$

2.9 Injury prevention and Fatigue Monitoring

As established in the first chapter chapter 1, the pursuit of "marginal gains" in high-intensity team sports like basketball requires the continuous development on the mitigation of related injuries. This research has increasingly shifted from reactive treatments to proactive, data-driven prevention strategies. Historically, the monitoring of localized muscle fatigue during athletic performance has relied on subjective athlete feedback or post-event analysis, both of which lack the temporal resolution required for real-time injury prevention. However, recent advancements in wearable biosensors and microcomputing have enabled the continuous, objective monitoring of neuromuscular and kinematic parameters during dynamic athletic movements.

Current research emphasizes that acute muscular failure is rarely an instantaneous event but rather precedented by a quantifiable period of continuous, localized muscular fatigue and deteriorating neuromuscular efficiency. The best and most common approaches for detecting this fatigue utilize surface Electromyography (Surface Electromyography (sEMG)) to track its temporal evolution. Physiological changes—such as the accumulation of metabolic byproducts slow down muscle fibre conduction velocity, fundamentally altering the sEMG signal [37]. Time-domain features are highly effective markers for this phenomenon. As muscles fatigue, amplitude-based features such as Root Mean Square (RMS) and Mean Absolute Value (MAV) typically exhibit an upward trend as the central nervous system increases motor unit recruitment and synchronization to maintain mechanical force. On the other hand, features that act as proxies for frequency, such as Zero Crossing (ZC) and Slope Sign Changes (SSC) decrease as action potentials slow down. Research has successfully utilized these specific time-domain features to construct objective fatigue indices, proving their efficiency in capturing impending muscle failure [38].

However, relying on isolated sEMG data presents critical limitations in dynamic athletic environments. A high sEMG amplitude indicates elevated electrical demand, which could signify impending fatigue-induced failure, or it could simply reflect an athlete's intentional increase in mechanical output, such as executing a jump or rapidly accelerating. To resolve this ambiguity, modern systems utilize sensor fusion, coupling sEMG with the Inertial Measurement Units (IMUs) discussed previously. By correlating the electrical demand of the muscle (via sEMG) with the actual mechanical output or kinematic stability (via IMU), researchers can accurately map true neuromuscular efficiency. Recent studies confirm that

multimodal fatigue detection systems using both kinematic and neuromuscular signals provide highly robust continuous fatigue classification, achieving accuracies above 95% even in the presence of complex, continuous movements [39, 40].

Edge computing and machine learning algorithms—such as Random Forests and Decision Trees—these multimodal wearable systems represent the cutting edge of real-time monitoring [41]. By identifying the precise moment neuromuscular efficiency begins to decline, these unobtrusive systems provide the foundational architecture necessary to predict and prevent specific, acute fatigue-related conditions.

2.9.1 Muscle Cramps and Pre-Cramp Anomalies

Cramping that occurs in sports or conditioning is clinically called as an Exercise-Associated Muscle Cramp (EAMC). This is a sudden, involuntary, and painful contraction of skeletal muscle during or immediately following physical activity, EAMCs severely disrupt athletic performance and can precipitate further musculoskeletal injury. While historically attributed to systemic factors such as dehydration or electrolyte imbalance, contemporary sports medicine increasingly supports the "altered neuromuscular control" hypothesis. This theory posits that severe, localized muscle fatigue disrupts the neural mechanisms regulating muscle spindle and Golgi tendon organ activity, leading to sustained, involuntary motor unit firing.

Whenever an EAMC happens, the EMG signal demonstrates an extreme and abnormal behaviour. The focal point of the cramp is characterized by massive localized spikes in sEMG amplitude [42]. During the cramping, the RMS and MAV features record very high values when compared to standard movements.

Recent literature shows that these involuntary contractions aren't spontaneous but rather have some distinct pre-cramp anomalies. For instance, in pilot studies observing involuntary nocturnal leg cramps, researchers found that changes in muscle tension patterns could reliably predict the start of the clinical cramp. Specifically, sEMG amplitude was observed spiking significantly—in some cases up to 40 seconds before the painful contraction physically manifested or triggered any movement artifacts [43].

This window of anomalies represents a crucial opportunity for injury prevention systems. By establishing a baseline of the normal activity of an athlete and identifying the rise in amplitude (RMS) and MAV and the drop in ZC and SSC, a threshold can be outlined, representing severe fatigue, to predict future cramping. If an abrupt, localized spike in sEMG amplitude occurs without a corresponding increase in mechanical output (as verified by kinematic IMU data), it strongly indicates the altered neuromuscular control that immediately precedes an EAMC.

3

Work Proposal

This project begins with the prototype design and the development of the arduino's code and of an interface app. Afterwards, data from different movements is collected and the ML models are developed. The last goal is to implement a ML model into a server connected via WiFi and a Tiny ML model into the app giving real-time feedback and analysis of their movements.

Basketball is one of the most popular sports worldwide and it is constantly becoming more complex and more competitive and so does the related technology alongside it. Gathering data from the players is crucial to improve the efficiency of their practices and their game. As of today, coaches rely in video analysis to help their players with tactical movements and technical skills but this method has its limitations such as blocked views or difficult angles for the recording device especially in a five vs five scenario. This is a strong suit of wearable sensors since they can capture data continuously and with the right design, be unnoticeable for the user and other players.

For basketball in particular, the movements consist in a lot of short sprints, quick changes of pace and direction, lateral movement and high jumps (vertical and horizontal) and its high tense moments are more common at the end of games (fourth quarter), specially at the last minutes.

3.1 Project Description

In this project, for each foot, an Arduino and an EMG sensor will be used. The Arduinos have an integrated IMU each that capture acceleration and orientation and they can communicate through BLE. Besides the IMU, the EMG sensors will collect data from the muscles to provide further insight for the movement recognition and injury report.

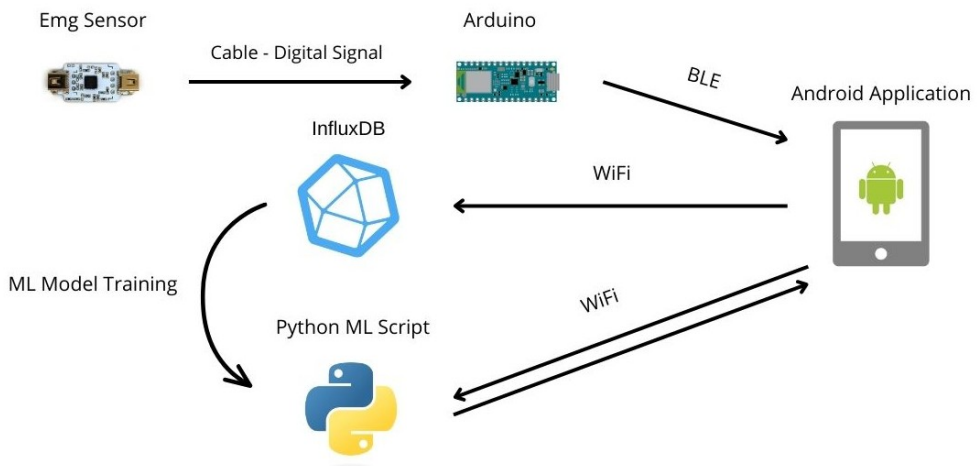
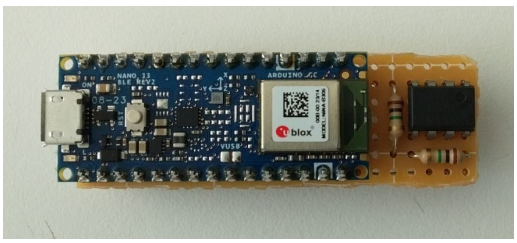


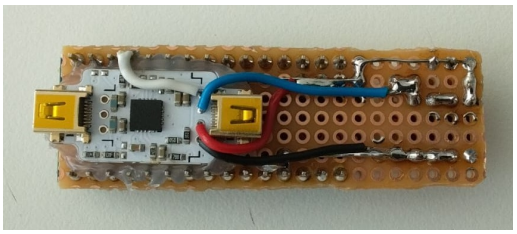
Figure 3.1: System Overview

3.1.1 Prototype

The Arduino chosen was the Nano 33 BLE Rev 2 for its size, 9 axis IMU, BLE connection and versatility. The choice for the EMG sensor was the BITalino Electromyography designed for surface EMG. This sensors were soldered to the Arduinos for power and communication.



(a) Top side - Arduino



(b) Bottom side - EMG sensor

Figure 3.2: Prototype Photos

3.1.1.1 Arduino Specifications

This Arduino runs at 3.3V and the IMU sensors are the BMI270 (3-axis accelerometer + 3-axis gyroscope) and the BMM150 (3-axis Magnetometer) sensors. It has a width of 18mm and length of 45mm weighing only 5g [44]. The Arduino IDE will be used for the Arduinos' programming.

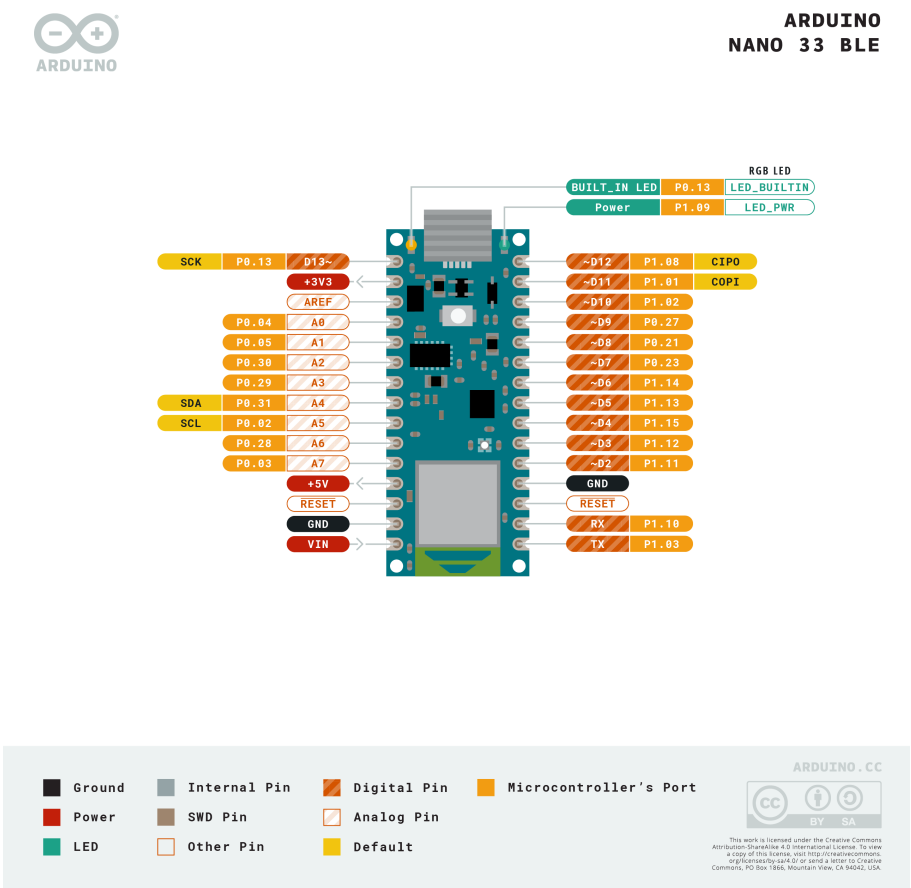


Figure 3.3: Arduino Nano BLE Rev 2 - Pinout [44]

3.1.1.2 BITalino Electromyography

The sensor has a gain of 1009 and an input of 2.0-3.5V making the output of 3.3V of the Arduino ideal for it. As seen in figure 3.4, The emg has an input channel for the electrode cables on the left side and 4 connection points on the right side:

- Output Voltage (VOUT) - Connected through the white wire to the respective input pin of the Arduino
- Voltage at the Common Collector (VCC) - The red wire is connecting this point to the 3.3V pin.
- VCC/2 - From the same pin that the VCC point is connected to, we have a small voltage divider with an Ampop for stabilization to halve the tension (blue wire).
- Ground (GND) - The black wire connect the Arduino's ground to this point.

A circuit was designed to obtain a stable reference for the EMG sensor, implemented using a resistive divider followed by an operational amplifier. As shown in figure Figure 3.5, two identical resistors are connected between the 3.3V supply from the arduino and ground

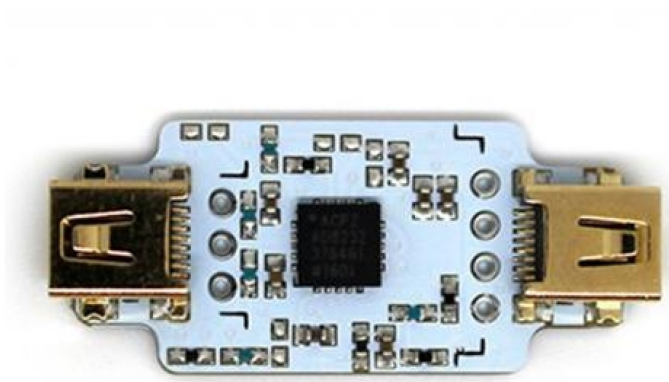


Figure 3.4: BITalino EMG sensor [**<empty citation>**]

resulting on the node in the middle settling at half the voltage. The op-amp's purpose is to stabilize the circuit by lowering the impedance at the output.

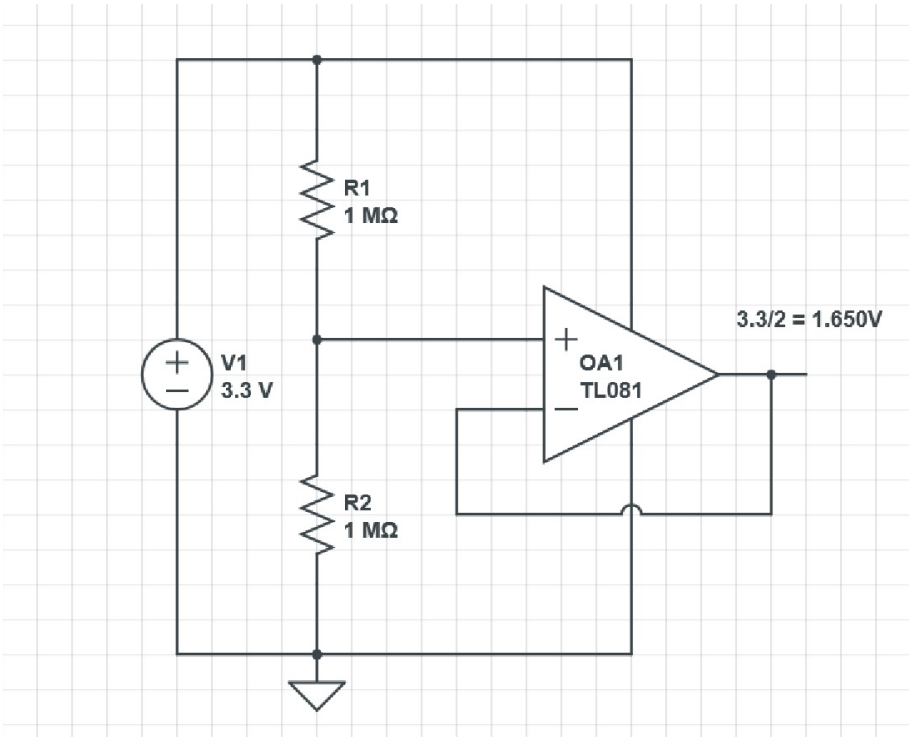


Figure 3.5: Halving the VCC

The analog sensor data are converted into digital values ranged between 0 and $2^10 - 1$ and for practical reasons we need to convert it back to voltage as show in the following

expression:

$$EMG(V) = \frac{(\frac{ADC}{2^10} - \frac{1}{2} * VCC)}{G} \tag{3.1}$$

$$EMG(mV) = EMG(V) * 1000 \tag{3.2}$$

Where ADC is the digital value from the sensor, VCC is the 3.3V provided by the arduino, G is the sensor gain which is 1009 and EMG(V) and EMG(mV) is the signal in Volt and millivolt respectively.

3.1.2 Data Acquisition

A fundamental step in this project is the acquisition of reliable and representative data, as the performance of subsequent analysis methods directly depend on the data quality. The EMG sensors will be attached to the Gastrocnemius Lateralis (see **??**) to capture the muscle activation patterns in basketball movements.

This process will be carried out under a controlled environment ensuring that the movements being made are correctly categorized. This labeling of activities is essential for the machine learning algorithm. For a richer dataset the players will be asked to do the movements with different directions, intensities, speed and other variations that can be applied depending on the motion.

Special attention will be given to minimizing signal noise and motion artifacts during acquisition, using appropriate electrode placement and filtering techniques. This ensures that the raw signals preserve as much physiological information as possible, thereby improving the robustness of later processing and analysis stages.

3.1.3 Data Processing

First the arduino converts the digital value of the EMG to mV and afterwards it calculates six different features before sending them to the app:

- Root Mean Square - RMS
- Mean Absolute Value - MAV
- Waveform Length - Waveform Length (WL)
- Zero Crossing - ZC
- Slope Sign Changes - SSC
- Variance - Variance (VAR)

Following the feature calculations, the arduinos send the data to the android device that forwards it to the database where it gets stored and to the server that predicts which movement the user is doing. With the stored data, two ML models are trained and then used for basketball movements recognition.

4

Implementation

This section is split into different sections explaining the project's developed software.

4.1 Prototype Hardware

Two prototypes were developed composed by an Arduino 33 nano BLE Rev 2, an EMG sensor and a small power bank to provide power (see 3.1.1). Standard power banks have a modern system to idle if not enough power is drawn. Since the Arduinos don't break that threshold, to keep the power banks active, a keep-alive circuit was utilized to periodically pulse the current draw.

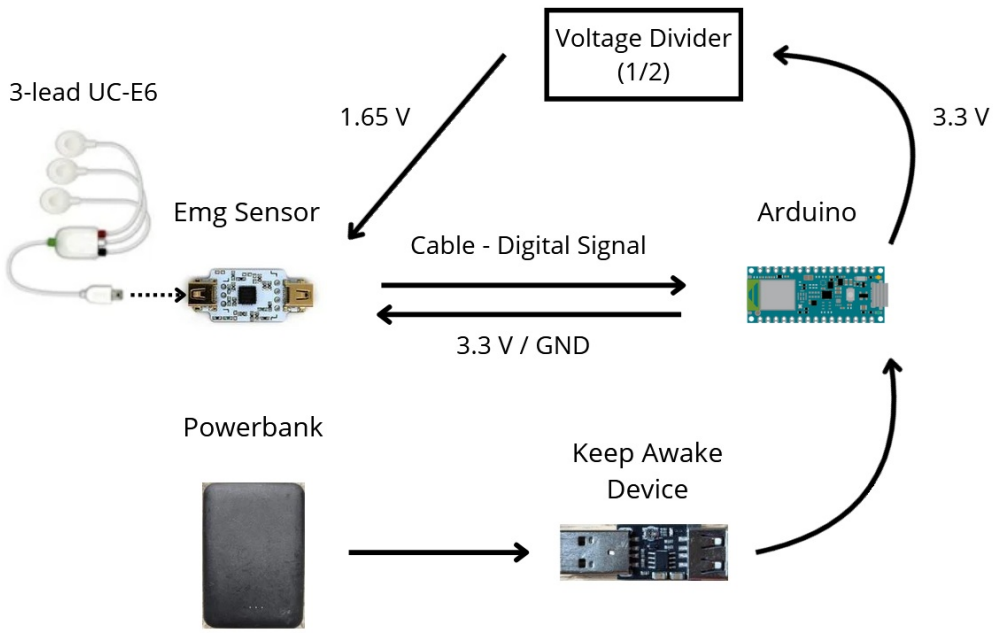


Figure 4.1: Physical System Overview

4.1.1 Arduino

As mentioned in the introduction of this chapter, the Arduinos used are two Arduino 33 nano BLE Rev 2 which is a 3.3V board of only 45x18mm of area. It features a powerful processor, the nRF52840 from Nordic Semiconductors, a 32-bit ARM® Cortex®-M4 CPU running at 64 MHz. In terms of sensors, the Arduino has a 9 axis IMU meaning it includes an accelerometer, a gyroscope, and a magnetometer with 3-axis resolution each. As its name mentions it also has BLE communication which is vital for this project [44].

4.2 Arduino Code

The Arduino programming language is a modified C/C++ language that simplifies the board coding that is run on the Arduino's IDE. Initially there were two code files for the Arduinos, one for the sensors calibration (specifically the magnetometer) and the other one is run for reading and sending the data. The calibration was eventually integrated in the main program and only a single program is needed.

4.2.1 Calibration

To ensure accurate and reliable measurements, the magnetometer and the gyroscope undergo a calibration process. Since the magnetometer's values depend heavily on the surrounding environment, a calibration routine must be executed before using the device. During this process, the program records the magnetometer's minimum and maximum values while the device is moved following a continuous motion resembling an infinity symbol (∞) and rotated around its axis until the user decides it's enough.

The offset for each axis is then determined by the following equation:

$$offset = minValue + (maxValue - minValue)/2 \tag{4.1}$$

And the scale is calculated through the next equation:

$$scale = (maxValue - minValue)/2 \tag{4.2}$$

In these expressions the minValue and the maxValue correspond to the lowest and highest recorded values for each axis. The corrected measurements for each axis can then be expressed as:

$$value = (rawValue - offset)/scale \tag{4.3}$$

The gyroscope also requires calibration to eliminate zero-rate bias (drift) that occurs even when the sensor is stationary. This process consists on leaving the device completely still for 5 seconds while the raw angular velocity is measured and stored. At the end, the average of these readings for each axis is the offset used in the main program:

$$value = rawValue - offset \tag{4.4}$$

After connecting both devices, in the Data Menu (subsubsection 4.3.1.3), there is a button that takes us to the Calibration Screen (subsubsection 4.3.1.4) where the user can proceed with that process. This should always be done before collecting data in a new location or time. In case of a disconnection, when the device reconnects, the app automatically loads the last saves calibration values. When the arduino receives the calibration command (either gyroscope or magnetometer), it starts the respective operation until, in the magnetometer's case, the user ends it or, in the gyroscope case, it finishes automatically. Afterwards the values are set as offsets and they're sent to the app where they are saved for later upload.

4.2.2 Sensors' Data Readings And Feature Calculations

To achieve consistent gait analysis and movement recognition, the system requires a high frequency of EMG and IMU data collection. However BLE has strict restrictions on the packet output frequency specially since we're dealing with two devices at high frequencies (they may collide with each other).

4.2.2.1 Sampling and Windowing

Raw EMG signals contain significant noise and offset so, before feature extraction, a High-Pass Filter is applied with a cut-off frequency of approximately 20 Hz (α =0.888). The IMU data is fed to the ML trainings raw. The microcontrollers operate in a non-blocking loop architecture designed to handle both the EMG and IMU acquisitions and different rates:

- High-Frequency EMG Readings (1000 Hz) - To obtain relevant features from the sensor it is read every 1 ms.
- Kinematic (IMU) & Transmission (50 Hz) - Every 20 ms the IMU data is measured right before the ble communication since no calculations are done to that data.

The coordination of both timings is managed through a windowing approach. A sliding window of 20 EMG samples is filled and once full (after 20 ms), the system triggers an event:

1. The IMU sensors are read.
2. The time-domain features are calculated from the last 200 EMG samples (see subsubsection 4.2.2.2)
3. All the data is bundled in a packet and waits to be sent (see subsubsection 4.2.3.2)

4.2.2.2 EMG Features Calculation

As mentioned before, EMG signals are non-stationary and contain significant noise making them unsuitable for direct raw transmission via BLE due to bandwidth constraints. Furthermore, many studies indicate that raw EMG underperforms on a ML classifier environment when compared against EMG features. The microcontroller extracts time-domain features since they are computationally more efficient than the frequency-domain-features that have the heavy processing load of the Fast Fourier Transforms (FFT). Each calculation processes a window of 200 samples (200 ms) resulting in six measurements:

1. RMS subsubsection 2.8.2.2
2. MAV subsubsection 2.8.2.1
3. VAR subsubsection 2.8.2.3
4. ZC subsubsection 2.8.2.10
5. SSC subsubsection 2.8.2.11
6. WL subsubsection 2.8.2.4

4.2.3 Bluetooth Low Energy

For the Bluetooth connection we first have to assign a name to the device and a unique address for its service and characteristics. In this project three characteristic are used. Then the service is advertised to nearby devices and, when there's a connection, the last saved calibration values are uploaded, the sensors' data collection is started and the characteristic is updated.

4.2.3.1 GATT Service

The system defines a unique service identified by a 128-bit UUID. Within this service, three specific Characteristics are utilized to handle data streaming and bi-directional control command flow.

4.2.3.2 Payload Optimization

A standard 32-bit floating-point representation would require 60 bytes per sample, which is inefficient for BLE bandwidth so in order to reach the objective of 50 Hz transmission rate for the nine IMU axis and six EMG features the payload has to be highly optimized. With this in mind, all the float sensor values were converted to 16 bit signed integers reducing the packets significantly while maintaining enough precision for analysis.

Each sensor frame consists of:

1. Accelerometer Data (6 bytes): X, Y, Z (×100 scaling)

Table 4.1: Bluetooth Characteristic Configuration

Characteristic Configuration			
Characteristic Names	Properties	UUID Prefix	Function
Data Stream	READ & NOTIFY	1b4867e2	The android app listens to this characteristic and is notified whenever the devices upload data
Calibration Command	WRITE	1b4867e3	When the user starts calibrating the android app sends commands via this characteristic
Calibration Data	READ & NOTIFY	1b4867e4	Here the devices upload the progress of the calibrations and the final results

2. Gyroscope Data (6 bytes): X, Y, Z (×100 scaling)
3. Magnetometer Data (6 bytes): X, Y, Z (×100 scaling)
4. EMG Features (12 bytes): glsRMS, glsWL, glsZC, glsMAV, glsSSC, glsVAR (Mixed scaling: ×10000, ×10000$or \times 10$)

$$TotalSize = 18(IMU) + 12(EMG) = 30Bytes \tag{4.5}$$

Although the packet size has been reduced, transmitting an individual packet every 20 ms from two separate device still introduces a BLE protocol overload. To get around this problem, the packets are bundled in groups of 5 ($30bytes \times 5packets = 150bytes$). This strategy effectively utilizes the large MTU (Maximum Transmission Unit) negotiated with the Android device, reducing the communication frequency to exactly once every 100 ms (for each device) reducing cpu usage and radio congestion. To make the system even more robust, a FIFO (First In First Out) buffer with a capacity of 150 packets was implemented. Whenever the communication with the Android device fails, that data is then saved in the buffer and the device tries to send it again later. One other challenge introduced by the BLE transmission and buffering implemented is the blocking nature of the communication operations. Pushing a 150 Byte bundle to the Bluetooth stack will momentarily block the main processing loop. While the microprocessor waits for the radio to finish, it will miss some of the 1 ms window defined for the EMG sampling leading to gaps in the timeline. To circumvent this, the system tracks the elapsed time strictly via a microsecond hardware timer. Immediately before constructing and transmitting the next BLE bundle, the firmware compares the current time against the scheduled EMG sampling interval.

If the previous transmission caused a delay, the system enters a rapid catch-up loop, executing the analog read and filtering functions multiple times sequentially until the "time debt" is cleared ensuring that while the transmission may introduce some jitter in the packet arrival time at the Android device, the sampling period remains consistent within the microcontroller's memory, preserving the temporal integrity required for accurate frequency-domain features.

4.3 Android Application

As mentioned in the Work Proposal (chapter 3), a dedicated Android application was developed in Kotlin, to serve as a bridge between the prototypes and the user/database. This programming language is used to interoperate fully with Java developed by JetBrains and since 2019 it has been the preferred language by Google for Android app development. The application handles the BLE connection with the devices and provides a user interface for researchers, athletes and coaches to visualize the collected data and the model's predictions in real time.

The following section provides a detailed description of the software architecture of the application. Each file is explained according to its function within the system, with emphasis on how data flows from the BLE layer to the machine learning classifier and finally to the user interface. This breakdown clarifies the responsibilities of each component, highlights the interaction between modules, and demonstrates how the application was structured to ensure modularity, maintainability, and extensibility.

This application follows a modular architecture separating responsibilities between the user interface, data handling, and Bluetooth communication. At its core, the program follow the MVVM (Model-View-ViewModel) design pattern, improving testability, maintainability and division of tasks.

The MainActivity.kt is the entry point of the application, initializing the navigation host and providing the top-level container for the user interface. From this activity, the navigation system manages the transitions between the windows (implemented as Jetpack Compose composables), each of which corresponds to a specific functionality of the program. The Android application serves as bridge between the devices, the user and the database.

4.3.1 User Interface

The application consists in 4 different screens that define the layout and interactive elements that the user sees. Each one of these windows has a corresponding ViewModel that acts as a connection point between the user interface and the application logic. These ViewModels handle any calls made by the screens and any changes to relevant operations such as the bluetooth connection.

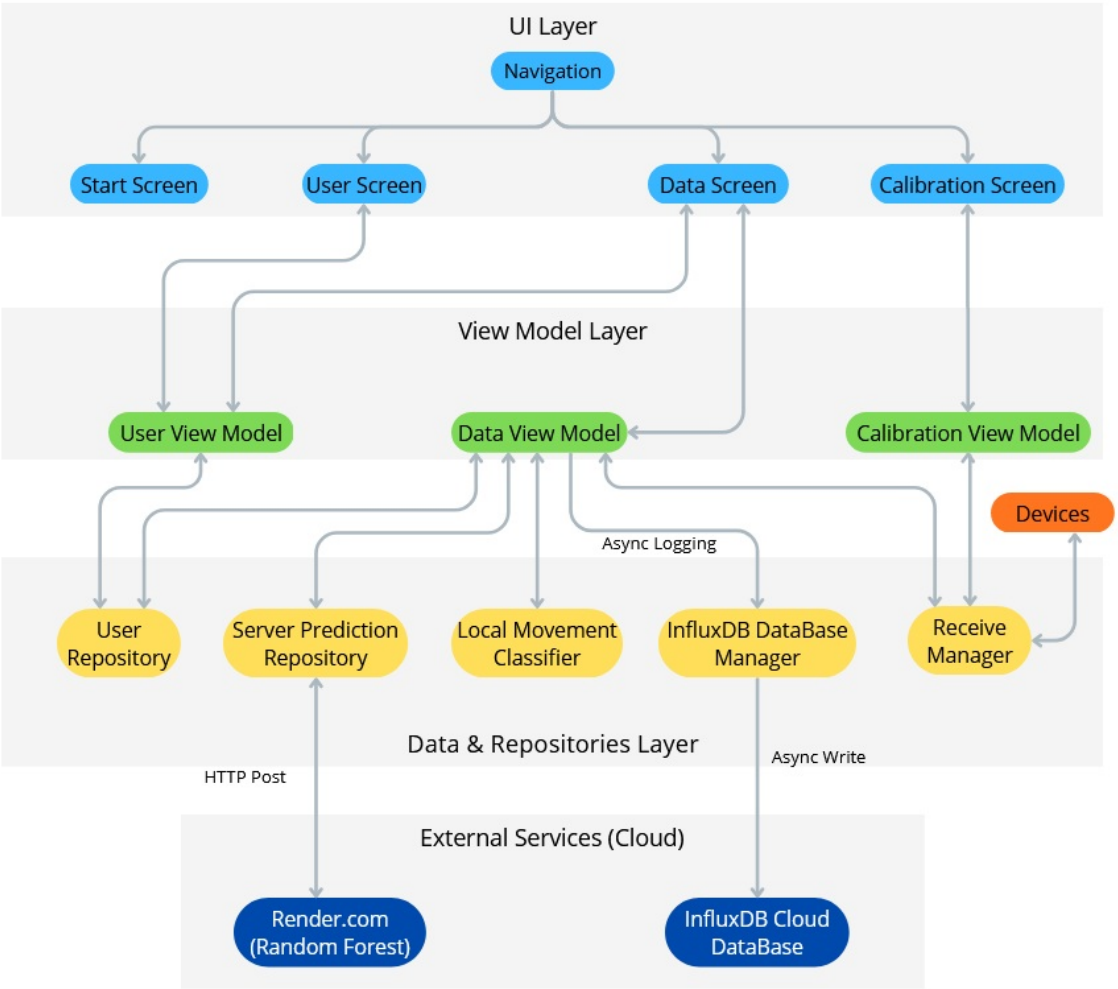


Figure 4.2: Android Application Overview

At first, when the app is launched, some necessary permissions are checked, specifically the Bluetooth (`BLUETOOTH_SCAN` and `BLUETOOTH_CONNECT`), the location (`ACCESS_FINE_LOCATION` and `ACCESS_COARSE_LOCATION`) and the internet access permissions. Even though the app doesn't use the user's location, its permission is required for BLE in Android 12 and higher.

4.3.1.1 Start Menu

This is the first menu shown to the user and it has the simple task of guiding the user through the application's functions. It has one button that takes us to the profile menu (4.3.1.2) and another that takes us to data menu(subsubsection 4.3.1.3).

4.3.1.2 Profile Menu

In this menu every saved user profile is listed and the user is able to create new profiles and delete existing ones. When creating a profile the app generates a unique id and the

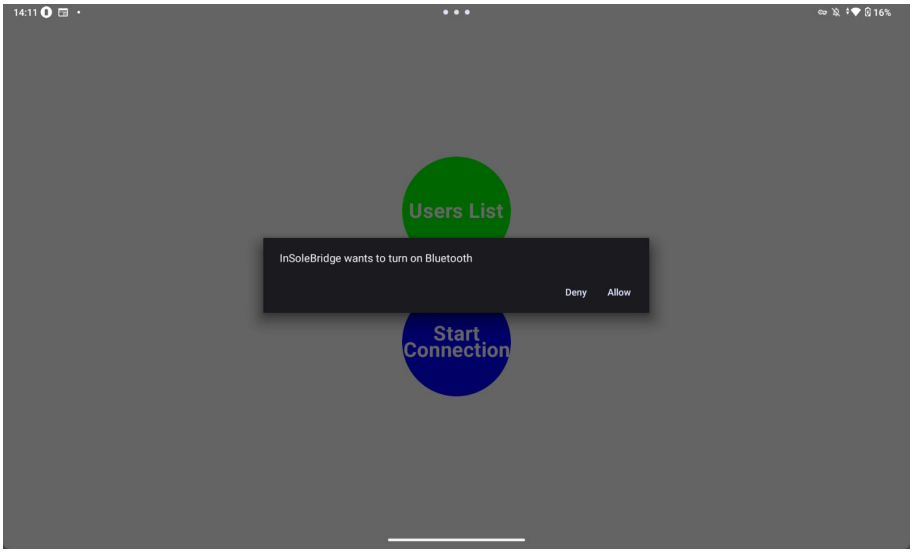


Figure 4.3: Request message to turn on Bluetooth

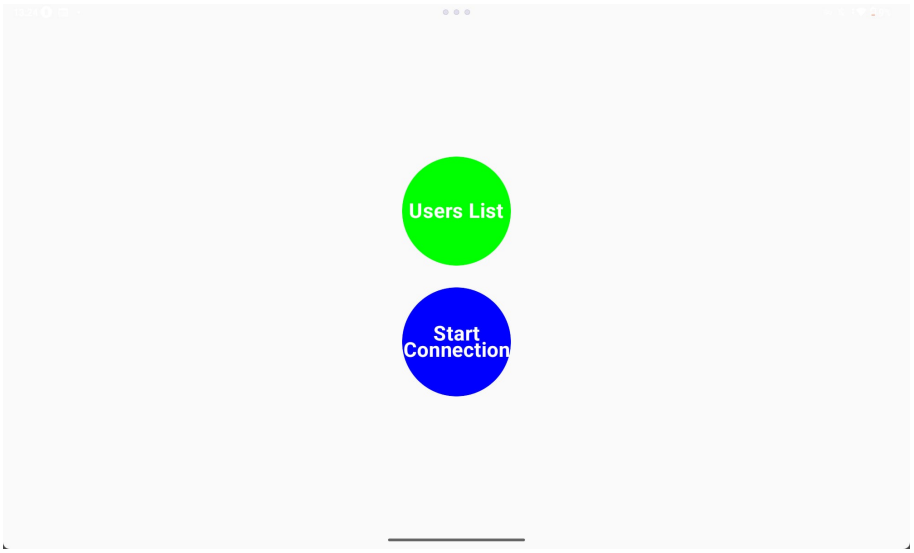
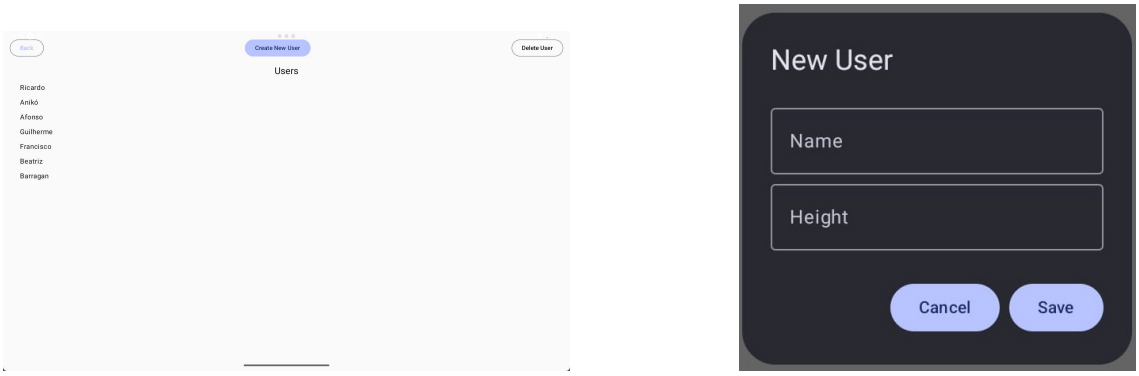


Figure 4.4: Start Menu

user is asked for a name and height. The list of users is stored locally and is available in every session.

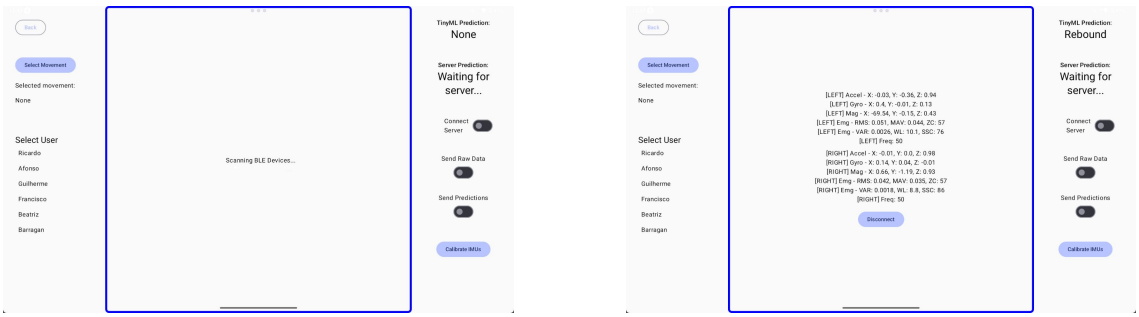
4.3.1.3 Data Menu

When this menu is opened, the app tries to connect to both devices and when able, starts exchanging data. The data is then shown to the user. On the left the user can select an active user profile and a label for the current movement. On the right, from top to bottom, the ML models' predictions are shown (section 4.4). Beneath the models' projections, there are 2 switches - one to start/stop the ML server connection and another one to start/stop sending the collected data to the InfluxDB database. At last there is a button that takes the application to the Calibration Menu (subsubsection 4.3.1.4).



(a) Profiles' Menu (b) New User Window

Figure 4.5: Profile's Menu and User Creation



(a) Without devices' connections (b) When both devices are connected

Figure 4.6: Data Menu

4.3.1.4 Calibration Menu

This window provides a dedicated interface for managing the sensor calibration routines which are critical for ensuring the accuracy of the IMU. As mentioned in the Arduino section (subsection 4.2.1), due to the inherent hardware drift and environmental magnetic interference, the gyroscope and magnetometer have to be recalibrated periodically for better and more accurate results. In this page the user can trigger the calibration methods that send specific commands to the devices with the instructions needed. For the gyroscope, the firmware executes a static calibration by averaging 1000 samples to compute and nullify the offset. For the magnetometer, the system performs a dynamic hard-iron calibration, recording the maximum and minimum magnetic field intensities during a multi-axis rotation to calculate precise spatial offsets and scaling factors.

The gyroscope calibration for both devices is done at the same time while the magnetometers' are executed one at a time. For each operation the app has a respective button and while ongoing calibration, the app display some real-time progress to the user via a dedicated BLE characteristic.

When all the calibration operations are finished, the following screen (Figure 4.8b) is presented and the values are saved to be uploaded when the devices reconnect

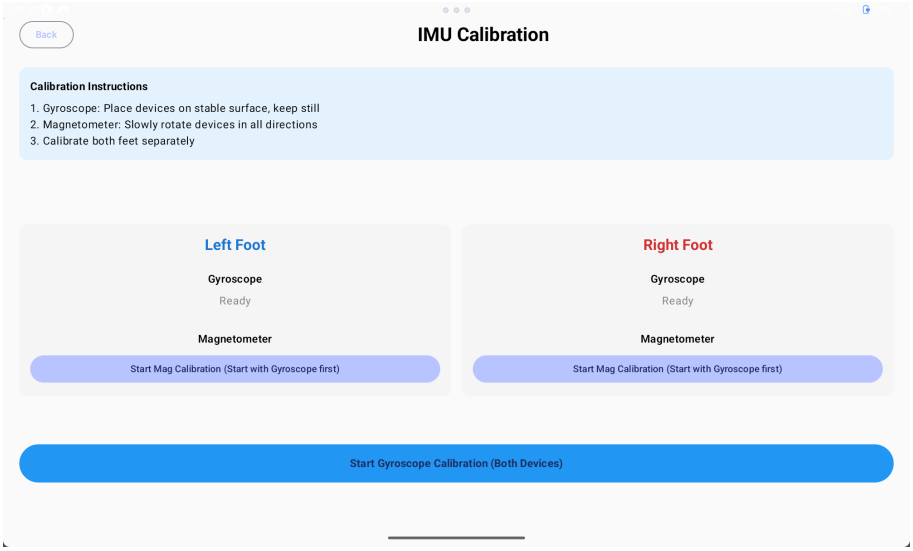
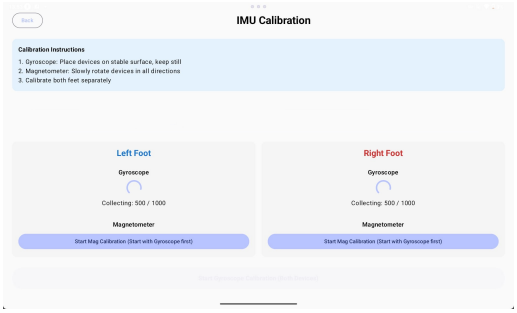
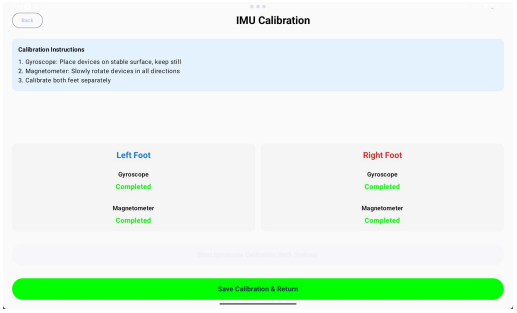


Figure 4.7: Calibration Menu



(a) Gyroscope Calibration in Progress



(b) Calibration Finished

Figure 4.8: Calibration Menu

4.3.2 Arduino BLE Communication

In Android BLE data is handle asynchronously which means that the app receives callbacks which have to be handled. For the BLE communication Android provides a package with a layered architecture:

- BluetoothAdapter - Where every bluetooth connection starts. It allows scanning for nearby available devices and initiating connections.
- BluetoothDevice - Represents another BLE device with one or more services.
- BluetoothGeneric Attribute Profile (GATT) - It's the protocol for BLE services and characteristics. It manages the connection lifecycle and provides callbacks for asynchronous communication.
- BluetoothGATT Service - Each service has a group of associated characteristics and a specific function (e.g., location service)

- BluetoothGATT Characteristic - It's the smallest unit of data in BLE BLE and it can be read, written or subscribed to (to ge notified whenever it changes) depending on its attributes.

Since the are two devices, each with its own uuid (see subsection 4.2.3), each device has its own manager that handle its callbacks (VariabBLEReceiveManager.kt) and these two managers are handled by the DualBLEReceiveManager that coordinates them.

4.3.2.1 Receive Managers

Each receive manager is responsible for managing the lifecycle of the corresponding BLE connection and handling every callback. Depending on the uuid of the connected device, the manager knows with which device it is working with (if left or right).

First, using the BluetoothAdapter (see subsection 4.3.2), the manager searches for the respective device using its name and uuid and once discovered a GATT connection is tried. After establishing the connection, the manager finds the device's services. If a device disconnects the application cleans up its resources and starts trying to reconnect.

In the first approach the system was designed with separate GATT characteristics for different sensor axes and EMG data. However, relying on multiple simultaneous asynchronous notifications caused the Android BLE stack to frequently drop packets and ignore callbacks. To resolve this without resorting to an inefficient sequential polling queue, the system was optimized by bundling all sensor data into a single 150 byte payload transmitted via a single characteristic.

The managers subscribe to the data characteristic of the respective device and rely on the onCharacteristicChanged callback. This allows the microcontrollers to asynchronously push data bundles at 10 Hz. Upon receiving the payload, the managers decode the byte arrays into their respective float and integer values. If the data upload switch is active, this payload is appended with the active userId, the selected movement label, and a timestamp, and is immediately sent to the database.

4.3.3 Communication with InfluxDB

As mentioned before the application uses InfluxDB as its database. The interface between the application and the database is implemented in two files, one containing the function that sends data and the other one has the sensitive client/database info (token, database url and organization and bucket name). This is a coroutine-based execution meaning that it will never stop or delay the main user interface thread and that it can be run multiple times in parallel without affecting the application performance.

Since InfluxDB organizes data using measurements, tags, fields, and timestamps, the application must first parse the incoming BLE payloads into the InfluxDB Line Protocol. This is a text-based format that optimizes ingestion speed by representing the data points in a single string structure:

- Measurement - movement_data
- Tags - sensor(left or right) and userId
- Fields (data values) - accelerometer, gyroscope, magnetometer and EMG data
- Timestamp - Instant.now() (registers that precise moment)

Once formatted, the data is transmitted to the cloud via an asynchronous HTTP POST request to the InfluxDB API, ensuring any network timeouts or errors are caught and handled without freezing the application's user interface.

4.3.4 User Database

To track the origin of the physiological signals, the Android application has a robust local profiling system. Before data collection begins, the user should identify the active subject (otherwise the app send a null for the id to the database). This is managed locally on the Android device using the Android Room Persistence Library, which provides an abstraction layer over an SQLite database to allow fluent database access while harnessing the full power of native Structured Query Language (SQL).

4.3.4.1 User Data Model and Data Access Object

The database system is defined by a data entity (user.kt) that represents a subject. Each subject is assigned an auto-generated primary key (id), a name, and height in centimetres. Interactions with the database are strictly mediated through a Data Access Object (DAO). The UserDao defines the SQL operations required to insert, delete, and retrieve subject profiles. To maintain a fluid user interface and prevent blocking the application's main thread during disk I/O operations, all database queries are executed asynchronously using Kotlin Coroutines (suspend functions).

4.3.4.2 Repository Pattern and Reactive State

To separate the data persistence logic from the User Interface, the architecture employs the Repository Pattern (UserRepository.kt). The repository acts as the single source for user data across the entire application. It abstracts the DAO layer, providing clean functions for the application's ViewModels to fetch lists of available users or create new ones while maintaining recollection of the active user.

The active user's ID is exposed to the rest of the application as a reactive StateFlow and is sent along side the sensors' data to the database. This modern reactive programming paradigm ensures that whenever the active user changes in the User Menu, any observing component in the application (such as the BLE transmission service) is instantly and automatically updated with the new context.

4.3.5 Machine Learning Predictions

The system was implemented with a double ML architecture. This approach utilizes both local computing (executed directly on the Android device) and cloud computing (executed on a remote server). By running two models simultaneously on the exact same data streams, the system is able to guarantee a low latency feedback for the user via the mobile application, while continuously logging a potentially more robust cloud prediction for comparative analysis.

4.3.5.1 Data Processing

To accurately identify every movement the user makes, the models require temporal context, making evaluating single 20ms data points insufficient. To resolve this, a time-series sliding window algorithm was implemented. The Android application synchronizes the BLE packets from the left and right insoles, combining their respective 15 features into a singular 30-feature vector per timestamp. These vectors are accumulated in a First-In-First-Out (FIFO) buffer to create a continuous observation window of 300ms that is then fed into the ML models.

The 300ms window size was specifically selected to capture the concentrated kinematic signature of transient, explosive basketball movements, effectively preventing the "label dilution" that could occur in larger windows. Furthermore, a step size of 100ms was applied, resulting in a 66% window overlap. This configuration compresses the classification rate to a highly efficient 10Hz, maintaining the illusion of instantaneous real-time feedback while preserving smartphone battery life and avoiding network congestion.

Once the 15-frame buffer is filled, the local Edge inference is managed by the MovementClassifier class within the Android application. Since the TensorFlow Lite model resides directly in the application's assets directory, it requires no internet connection to function. The model processes the flattened feature array and yields a probability distribution across the target classification labels. An argmax function is applied to the output tensor to extract the highest confidence prediction. This prediction is immediately pushed to a Kotlin StateFlow, triggering a quick update to the Jetpack Compose User Interface, ensuring the athlete receives uninterrupted feedback even in environments with degraded network connectivity.

4.3.5.2 Communication with the Cloud server

To facilitate the dual-tier prediction, the Android application securely transmits the buffered sensor data to the Flask server using the Retrofit network library. When the 15-frame buffer is filled, the data is serialized into a JSON payload and transmitted via an asynchronous HTTP POST request to the /predict endpoint.

To guarantee that network latency does not block the main thread or interrupt the continuous BLE data ingestion, the API call is offloaded to a background thread utilizing

Kotlin Coroutines (Dispatchers.IO). Upon receiving a successful HTTP 200 OK response from the server, the application deserializes the cloud-computed prediction.

Finally, to enable rigorous post-session analysis, the application utilizes the InfluxDB-Client-Kotlin library to asynchronously log both the TinyML local prediction and the Random Forest cloud prediction to an InfluxDB database. These predictions are uploaded tagged with a timestamp and User ID.

4.3.5.3 Temporal Smoothing and Output Stabilization

During the first live test, a susceptibility to "prediction flickering" was noticed. Since the classifiers evaluate discrete 300ms data windows, anomalies such as an irregular movement or sensor noise could cause isolated 100ms frames to be misclassified. To mitigate this instability, a post-processing Temporal Smoothing algorithm was developed within the Android application. This algorithm functions as a digital low-pass filter utilizing a majority voting logic. A continuous sliding window, implemented via a Kotlin ArrayDeque, was configured to store the last 5 consecutive predictions, effectively granting the system 500ms of temporal short-term memory. To compare the original prediction values with the filtered ones, the system sends both values from both models to a database for future analysis.

4.3.6 Cramping Prevention and Fatigue Monitoring

Besides movement classification, the Android app also has a real-time fatigue monitoring system to prevent cramping. This is achieved through an algorithm that fuses the predictions from the TinyML model with the RMS EMG feature. This script is located within the processLocalPrediction function of the VariabViewModel and it analyses the 300ms observation window raising a flag if the RMS value breaches a pre-defined threshold for either leg while idling or walking (not enough data was collected to monitor all movements). Since the EMG signal varies significantly based on an athlete's muscle density, skin impedance, electrode placement, and baseline fitness, this threshold isn't universal. A calibration should be conducted for every athlete. Since a simple flag could keep firing every processed window, a counter tied to the flag is utilized to send a message to the application when a threshold is hit. This tracker resets whenever the RMS signal drops bellow the defined threshold or the movement changes (from Idle or Walking). When 10 consecutive flags are raised (corresponding to one second), the application sends a notification to the screen notifying the user of the athletes' fatigue and that he should rest. This also helps to prevent false positives and filters out sudden stops that might spike the RMS values.

4.4 Machine Learning Code

The development, training, and exportation of the machine learning models were conducted using Python within a Google Colab environment. This section details the data preprocessing pipeline, the time-series windowing algorithm, and the specific architectures used for both the Cloud and Local classification models.

4.4.1 Models' Training

To train models capable of recognizing dynamic biomechanical movements, the raw sensor data previously logged to InfluxDB was exported as CSV files and ingested into a pandas DataFrame. The initial dataset consisted of discrete kinematic and electromyography measurements mapped to specific target labels (see **??**).

The first phase of preprocessing involved synchronizing the independent left and right insole data streams. Timestamps were parsed and rounded to the nearest 20ms. The dataset was then pivoted from a long format to a wide format, aligning the 15 features from the left insole and the 15 features from the right insole into a singular 30 feature vector for every discrete timestamp.

Since transient basketball movements cannot be accurately classified from a single 20ms static frame, the synchronized data was subjected to a sliding window transformation. The algorithm grouped the continuous data into windows of 300ms. To ensure the models were trained on contiguous sequences without mixing different movement classes, the grouping was strictly partitioned by userId and movement label.

Using a step size of 100ms, overlapping windows were generated. Each 15×30 matrix was then flattened into a 1-dimensional array of 450 features. This flattened array served as the input vector (X) for both models, with the corresponding movement acting as the target label. To ensure a robust evaluation, the generated windows were divided using an 80/20 train-test split, employing stratification to maintain proportional class distributions across both sets.

4.4.2 Random Forest

For the Cloud-Tier classification, a Random Forest (RF) algorithm was implemented using the scikit-learn library. RF was selected due to its ensemble learning architecture, which constructs a multitude of decision trees during training and outputs the mode of the classes for classification. This approach is highly resistant to overfitting and performs exceptionally well on high-dimensional tabular data, such as the 450-feature vectors generated by the sliding window.

The model was instantiated with n_estimators = 100 (utilizing 100 individual decision trees) and a max_depth = 15 to prevent the trees from memorizing localized noise in the training data. The model was fitted on the 80% training split and evaluated against the 20% testing split. Following successful validation, the trained model pipeline was serialized

and exported as a .pkl file using the joblib library. This serialized file was subsequently deployed to the Python Flask backend on Render.com, allowing the server to reconstruct the exact model state for real-time identification.

4.4.3 TinyML

To facilitate ultra-low latency inference directly on the Android edge device, a TinyML approach was employed - developing a lightweight Neural Network (NN) using the TensorFlow and Keras libraries.

Given the strict computational constraints of mobile devices, the network architecture was aggressively optimized. The input layer was designed to dynamically accept the 450-feature flattened array. This was followed by two dense hidden layers consisting of 32 and 16 neurons, respectively, both utilizing the Rectified Linear Unit activation function to introduce non-linearity. The final output layer consisted of neurons corresponding to the total number of target classes, utilizing a softmax activation function to output a normalized probability distribution.

The model was compiled using the Adam optimizer and sparse categorical cross-entropy as the loss function, circumventing the need to manually one-hot encode the target labels. The network was trained over 40 epochs with a batch size of 32. Once validated against the testing split, the Keras model was passed through the TFLiteConverter. Default optimizations were applied to quantize the model weights and reduce the overall memory footprint. The resulting binary file (tinyml_model.tflite) was exported and embedded directly into the Android application's asset directory, granting the mobile device autonomous classification capabilities without relying on external APIs.

4.4.4 Model Evaluation

To rigorously validate the performance and relevance of the trained models, a comprehensive evaluation protocol was established. Going beyond standard accuracy scores, these chosen metrics were designed to provide insight into the models' decision-making processes, sensor reliance, and generalized robustness across the different target classes.

For the Cloud-Tier RF model, a feature importance analysis was configured to quantify the contribution of each sensor. Since the sliding window technique flattened the data into a high-dimensional 450-feature array, the protocol dictated that Gini importance scores extracted from the decision trees be aggregated across the 15 time-frames back to their 30 base sensor components. This aggregation is designed to produce a definitive ranking of the sensors, allowing for a biomechanical assessment of whether the EMG data provided a significant classification interest or if it is insignificant.

Furthermore, to evaluate the multi-class discriminative power of the RF algorithm, the generation of Receiver Operating Characteristic curves was integrated into the testing pipeline. By binarizing the target labels in a One-vs-Rest configuration, the trade-off

between the True Positive Rate and False Positive Rate can be plotted for each discrete movement class (Idle, Walking, Running, Defensive Slide, Rebound).

For the Edge-Tier TinyML model, the evaluation strategy required continuously logging the training history over the 40 epochs to monitor the network's mathematical convergence. Tracking these temporal learning curves (categorical cross-entropy loss and classification accuracy) is critical for neural network architectures to ensure the lightweight model did not overfit the training data or memorize localized noise.

Finally, the pipeline was programmed to compute confusion matrices for both the Cloud-Tier and Edge-Tier models mapping the predicted classifications against the actual target labels on the isolated 20% testing split.

5

Verification and Testing

In this chapter we'll go over the verification and testing procedures conducted to evaluate the proposed wearable monitoring system and the ML models. This evaluation is divided into three sections addressing the full Internet of Things (IoT) system. Starting with the hardware, we examine the position of the prototypes, and how they are attached, and the anatomical placement of the EMG electrodes on the Gastrocnemius Lateralis. Secondly, an in-depth statistical analysis of the trained ML models is conducted. In this section, by leveraging classification metrics, confusion matrices, and feature importance distributions, both developed ML models' performances are compared against each other and evaluated. Here, we also determine the efficacy of fusing IMU kinematics with EMG neuromuscular data for basketball movements classification. In the final section, the system's performance is reviewed by assessing both models' latency and the resilience to asynchronous BLE transmissions.

5.1 Data Collection

The data sets used to train and test the models were collected from several volunteer athletes. They were asked to do each of the five movements - Idle, Walking, Running, Defensive Slide and Rebounding - for some minutes each. The cramping prevention system was only tested on one player that pushed his muscles to near involuntary contractions.

5.1.1 Prototype Placement and Configuration

The prototypes were mounted directly in the athletes' footwear to approximate the ideal positioning (on the bottom of the feet as an insole or a sock). The devices were secured using the shoe laces and the powerbank was secured in the users socks, utilizing the existing infrastructure. With this configuration, the IMU accurately captures the kinematics of the foot, such as accelerations and rotational velocities during the athletes' movements. Furthermore, this setup minimizes the mechanical strain exerted by the power cables on the USB ports, while keeping the EMG leads in close proximity to the chosen muscle group.

Figure 5.1: An athlete wearing the prototype

The muscle chosen to be measured was the Gastrocnemius Lateralis as it is one of the most important muscles for explosive jumping motion. The position for the EMG sensors was chosen as recommended by the European Union (EU)'s SENIAM (Surface ElectroMyoGraphy for the Non-Invasive Assessment of Muscles) project ([45]) ensuring standardization and maximization of the signal-to-noise ratio.

5.2 Machine Learning Model Evaluation

This section goes over an evaluation of the two ML models trained for this study. To effectively assess their performance, stability, and learning behaviour, we will examine the graphics generated throughout the training phase giving us clear insights into how each model converges, identifying potential issues like under or overfitting, and ultimately compare their suitability for the objectives of this project.

5.2.1 Training Metrics and Performance Curves

For the TinyML model, the learning process is monitored iteratively across the training epochs. Figure 5.3 illustrates the model's accuracy progression, comparing the training dataset against the unseen validation dataset. As observed in the accuracy curve, the model learns rapidly during the initial epochs before stabilizing reaching an validation accuracy of approximately 94%. The validation curve remains close to the training line

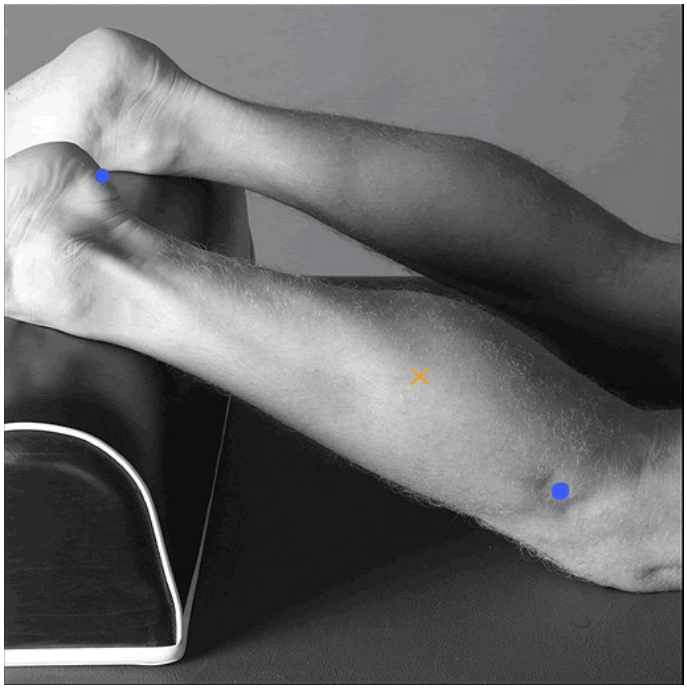


Figure 5.2: Gastrocnemius Lateralis Positioning [45]

throughout the training epochs indicating that the model generalizes well to new data rather than simply memorizing the training samples.

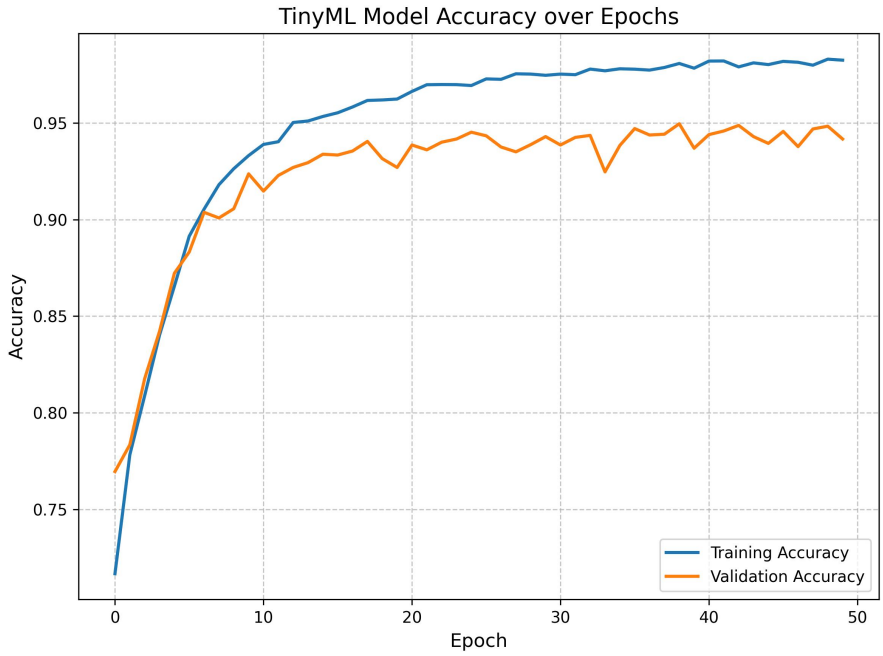


Figure 5.3: TinyMLAccuracy

Figure Figure 5.4 further demonstrates that behaviour. The categorical cross-entropy loss curve shows a synchronized descent of both the training and validation loss curves confirming that the model is successfully minimizing its errors during the training and

attributing high confidence to the correct movement.

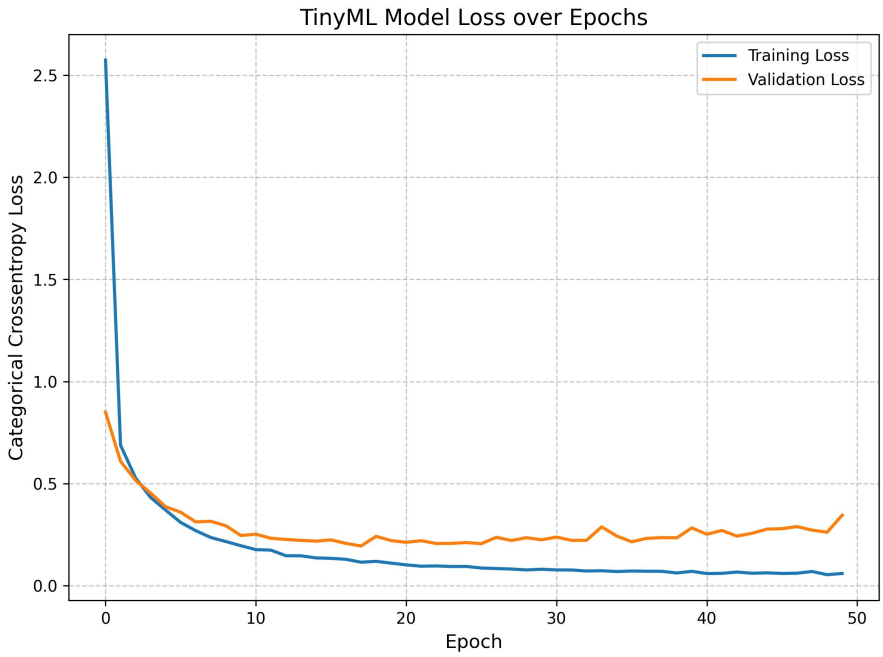


Figure 5.4: TinyMLLoss

While the Neural Network was evaluated via epoch progression, the Cloud-based Random Forest model is evaluated using ensemble classification metrics, specifically the Receiver Operating Characteristic and Precision-Recall curves.

The Receiver Operating Characteristic curve (Figure 5.5) evaluates the true positive rate against the false positive rate for each individual movement class. The Area Under the Curve scores approach 1.0 across all movements, demonstrating that the RF algorithm is highly capable of distinguishing the mathematical boundaries between distinct biomechanical states.

The Precision-Recall curve (Figure 5.6) shows the trade-off between precision and recall for different thresholds. This type of chart is specially crucial in in highly imbalanced sports datasets where the vast majority of movements recorded are Walking, Running or Defensive Sliding while movements like Rebounds, Shooting and Crossover happen in a split second. The resulting curve demonstrates that the RF model is able to detect all movements with high precision and low false positives.

5.2.2 Feature Importance and Sensor Fusion

The RF approach utilizes decision trees so it inherently provides a quantifiable measure of how much each sensor contributes to the final classification, known as Gini Feature Importance.

The next Figure 5.7 shows that even the IMU data has greater importance and the EMG provides a very small contribution to the movement identification (+/-10%). This

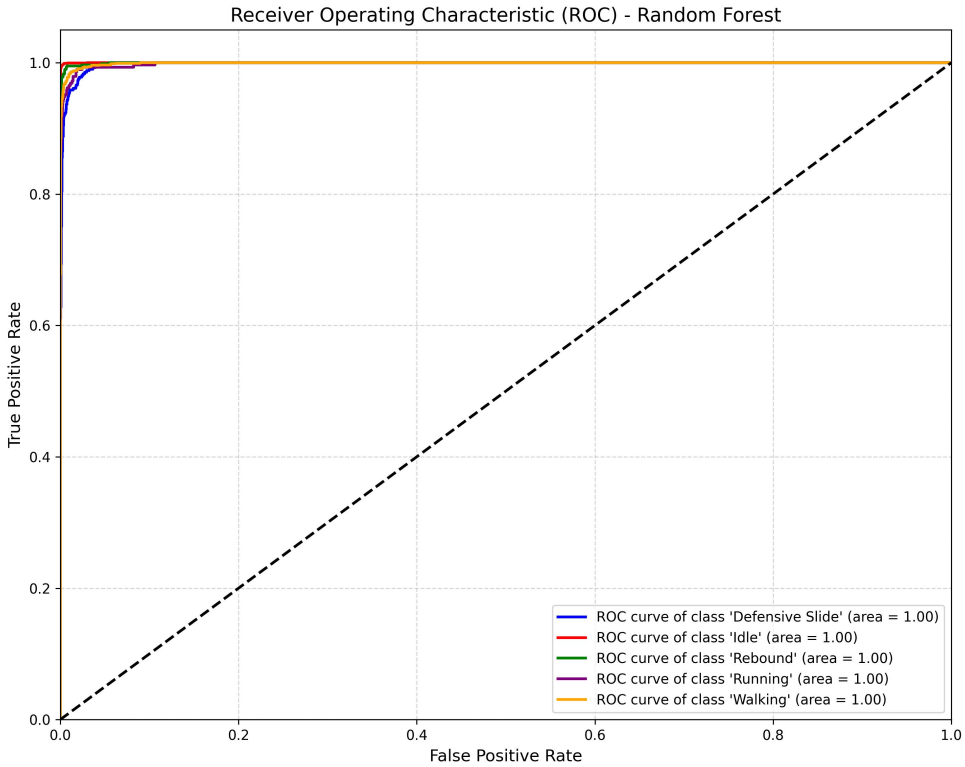


Figure 5.5: RF ROC Curve

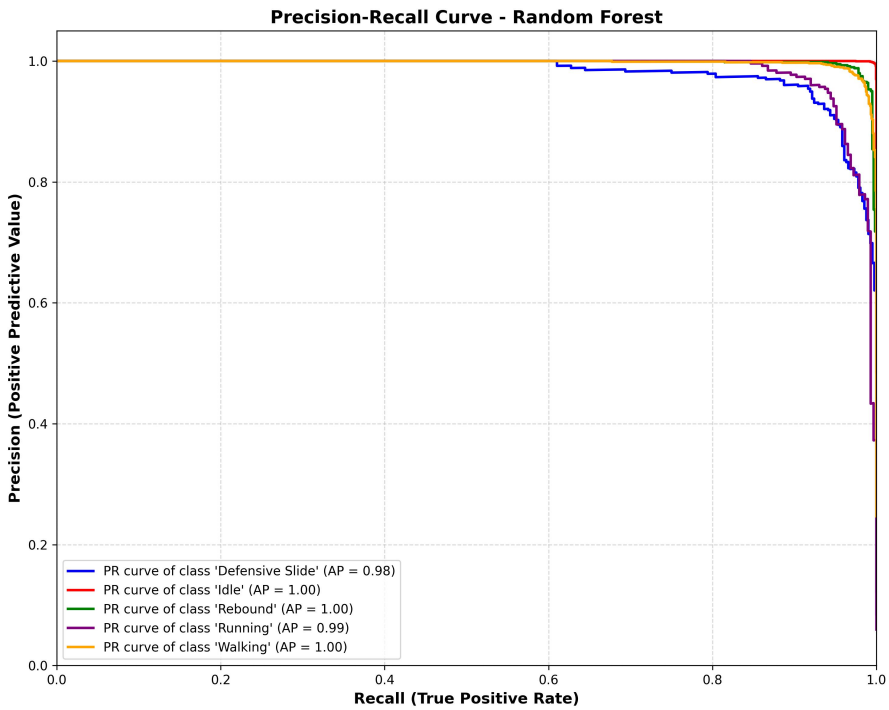


Figure 5.6: RF Precision Recall Curve

means that for better performance the EMG data could be cut and be used only for the injury prevention algorithm.

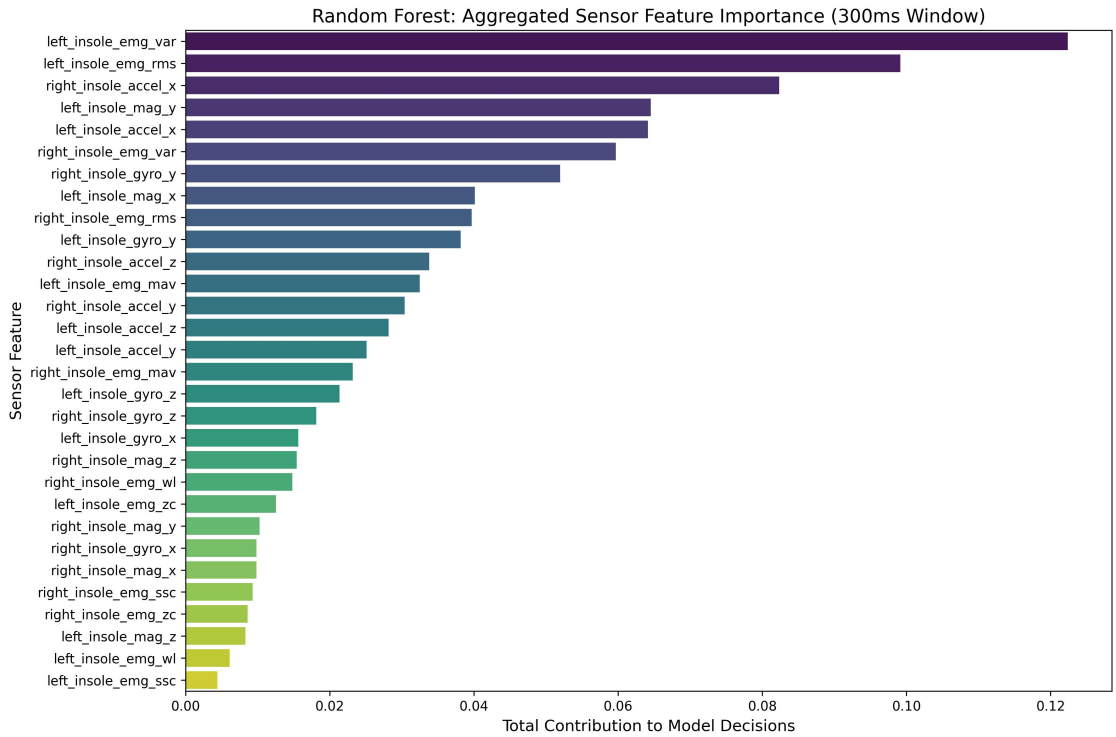


Figure 5.7: Aggregated Feature Importance Graph

As illustrated in Figure 5.7, the EMG features VAR and RMS had the biggest impact. This shows that the EMG is valuable for movement recognition systems alongside the IMU. The graph also shows a preference for the left side suggesting that the model picked up on a dominant players' foot.

5.2.3 ML Training - Confusion Matrices and Error Analysis

To conduct an analysis of the models' predictive flaws, confusion matrices were generated for both the Edge and Cloud classifiers based on the offline test dataset.

?? reveal strong diagonal density confirming high accuracy overall. However, looking at the errors reveal that most of the misclassification occurs between movements that share similar profiles (for example the models sometimes confuse the movement Defensive Sliding with Running and vice-versa or Rebound with Idle).

5.3 System Performance

A live test was performed in similar fashion to the data collection phase with the simple difference of running the server and taking note of the predictions and the movement made by the athlete (using the "Send Predictions" switch). The player also worked intensively until the fatigue warning triggered.

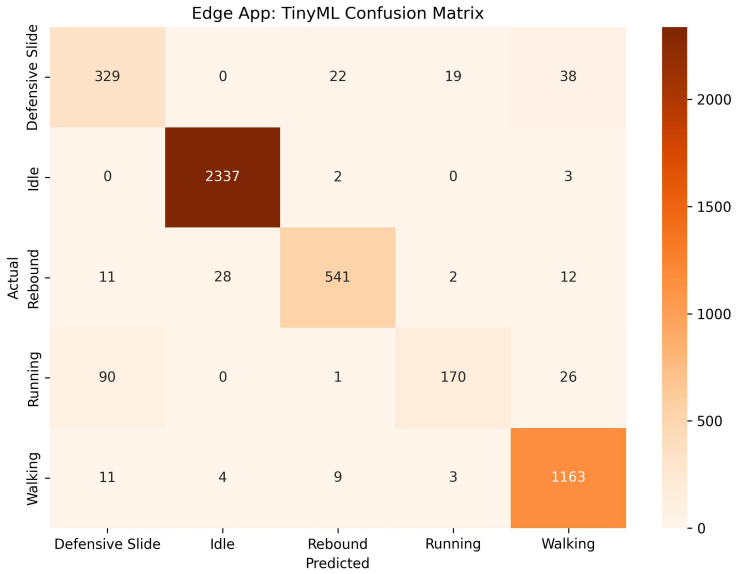


Figure 5.8: TinyML Confusion Matrix

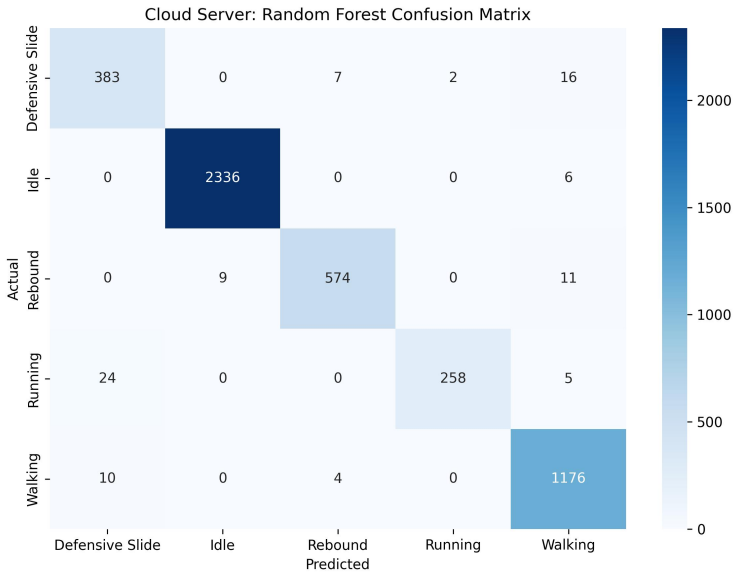


Figure 5.9: Random Forest Confusion Matrix

5.3.1 Overall System Accuracy

The primary objective of this test was to compare the classification accuracy of the ultra-lightweight Edge model against the deeper Cloud model and the variation of smoothed predictions vs raw. In terms of latency, the tinyML model responded instantly while the RF model had a slight delay.

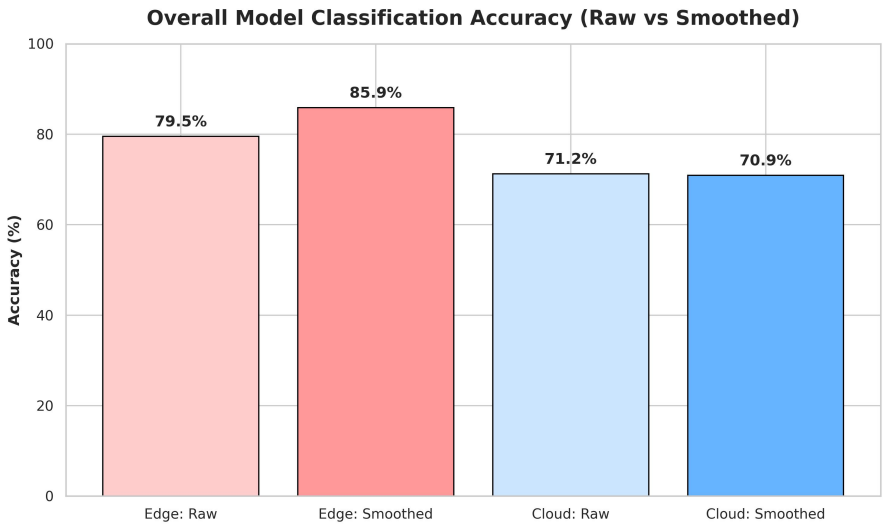


Figure 5.10: Overall Classification Accuracy: Local TinyML vs Cloud Random Forest (Raw and Smoothed)

As illustrated in Figure 5.10, the localized TinyML model successfully outperformed the cloud-based Random Forest model, achieving an overall accuracy of 79.5% compared to the server's 71.2%. Even tho the local model achieved better results, with a greater dataset the server model should outperform it, especially with more confusing movements. The smoothed version of the local model has a 6.4% higher accuracy than its counterpart while the server's is worse by 0.3%. This shows that smoothing the predictions is a viable and worthwhile algorithm if the base model is consistent enough.

5.3.2 Real Test - Confusion Matrices and Error Analysis

In Figure 5.11, we can see that overall, besides Defensive Sliding, the RF model either outperformed the TinyML model or was nearly as accurate. The RF model struggled to correctly identify the movements Rebounding and Defensive Sliding but excelled at the Idle, Walking and Running movements. The improvement from the smoothing algorithm is also prevalent here in the majority of the movements.

Continuing the system's assessment, the confusion matrices for both models were generated:

The RF's matrices (Figure 5.12, Figure 5.13) are very similar showing that when the athlete was Defensive Sliding, the model predicted the movement Running or Walking while only predicting about 12% of the time right. However, the model has a very high

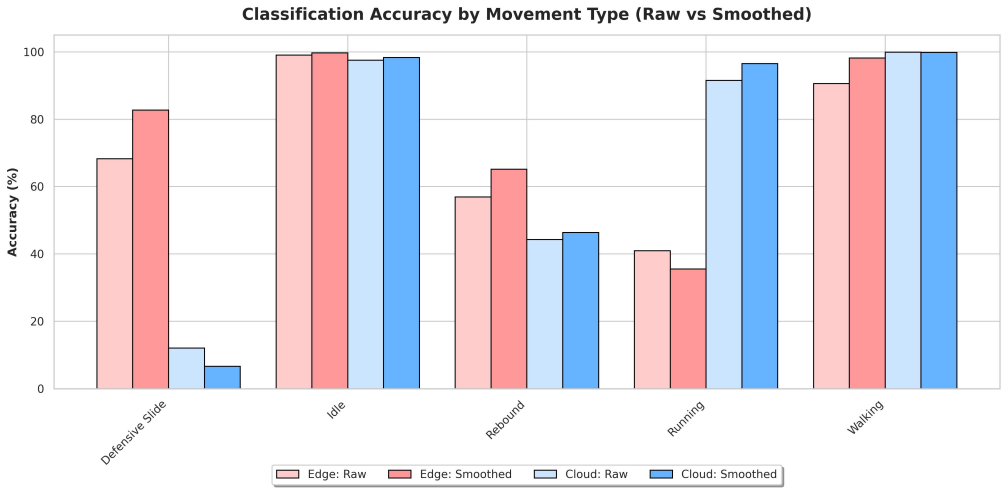


Figure 5.11: Classification Accuracy by Movement

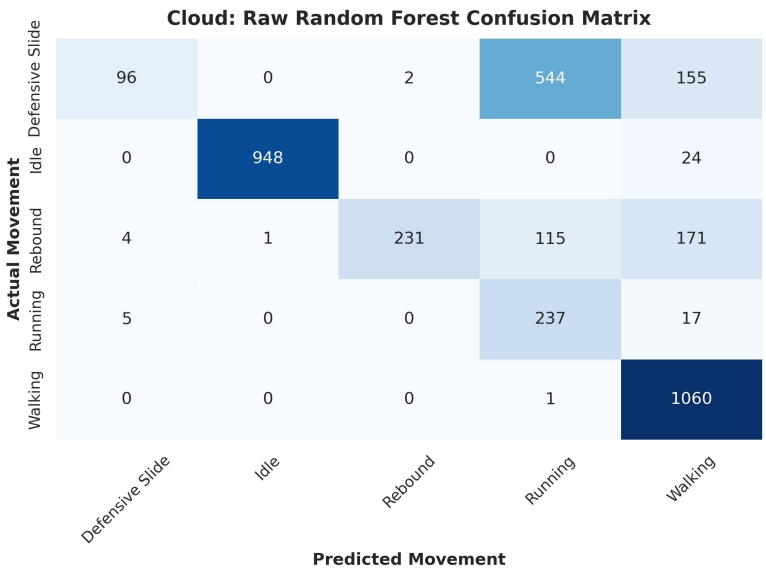


Figure 5.12: Confusion Matrix for the Random Forest Model before Smoothing

accuracy on the movements Walking, Idle and Running while Rebounding was mistaken with Walking or Running half the time.

On the other hand, the TinyML model had a slight accuracy boost by the smoothing algorithm. Analysing Figure 5.14 and in Figure 5.15, the edge model also mistaken Running with Defensive Sliding and vice-versa as the RF model did. Even so, the matrix's diagonal is pretty strong and the movements Walking and Idle and phenomenal having a near perfect accuracy.

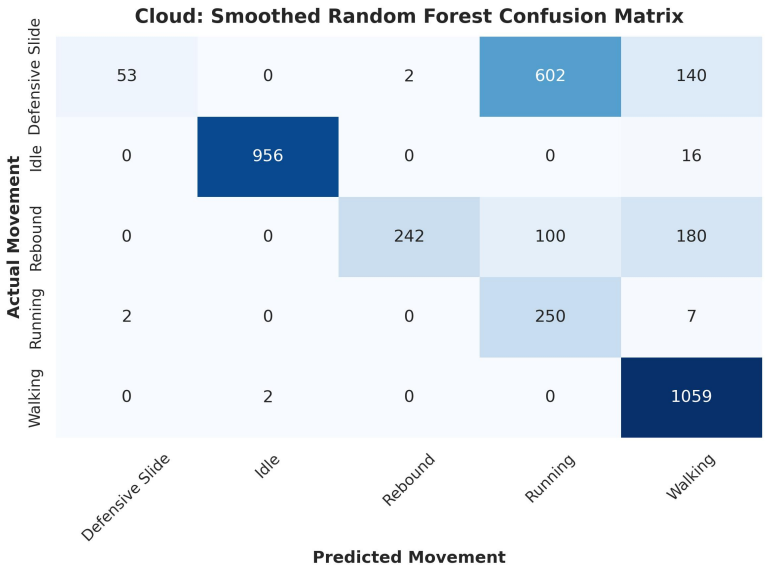


Figure 5.13: Confusion Matrix for the Random Forest Model after Smoothing

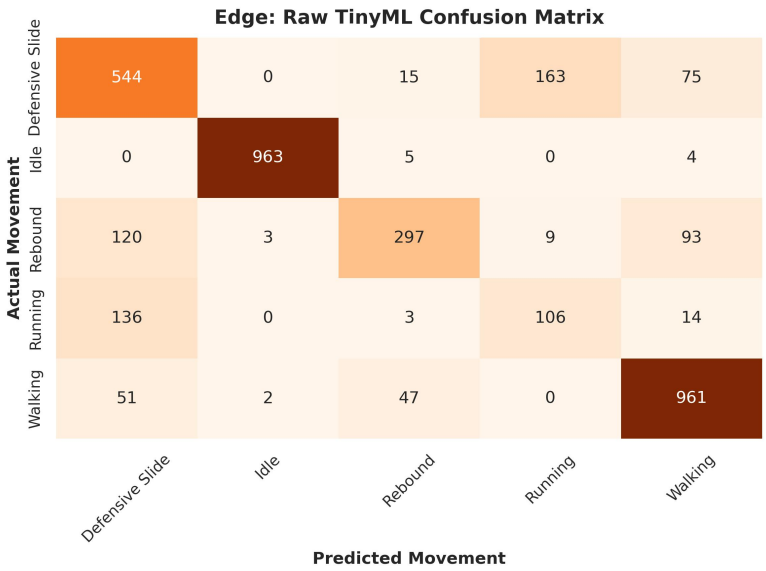


Figure 5.14: Confusion Matrix for the Raw TinyML Model

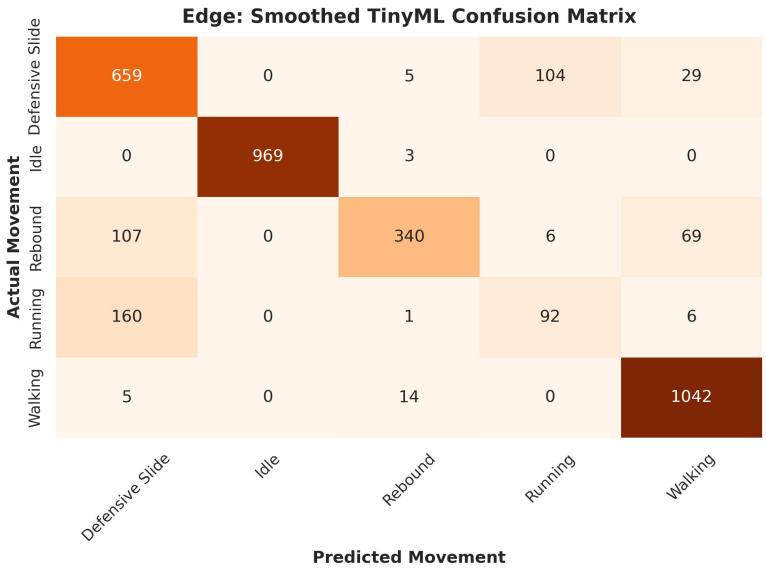


Figure 5.15: Confusion Matrix for the Smoothed TinyML Model

5.4 Discussion of Results

Even tho the ultimate goal of using the system in a real game scenario wasn't conducted in this project, the results validate the core hypothesis of the study that combining IMU with EMG data enhances movement recognition while also highlighting the major obstacles for that objective.

Another objective of this research was to determine if the inclusion of EMG sensors provided a tangible advantage over traditional IMU-only commercial wearables. The Feature Importance analysis (subsection 5.2.2) definitively answered this question, as the Random Forest classifier relied heavily on the EMG VAR and RMS features to differentiate complex basketball movements.

The implementation of the 500ms Majority Voting sliding window raises the challenge of balancing output stability and sensitivity of the models. As observed in the confusion matrices (**??**), temporal smoothing acted as an effective low-pass filter, diminishing flickering and hardware noise for the predictions increasing its accuracy at the cost of responsiveness. This trade-off is acceptable for a system focused on general fatigue and activity tracking, but in future iterations with different goals it should be avoided.

5.4.1 Known Problems and System Limitations

The transition from a controlled laboratory environment to live, dynamic basketball testing introduced several unforeseen hardware and methodological challenges. Documenting these limitations is critical for contextualizing the dataset's quality and guiding future iterations of the prototype.

5.4.1.1 Hardware Fragility and Ergonomics

The explosive nature of basketball movements places immense mechanical stress on external wearable components. During early testing, the Universal Serial Bus (USB) connections, specifically the left Arduino port and the right EMG port, could not withstand the forces generated by the wires during the movements and were physically ripped from their boards. These accidents required re-soldering to restore functionality. Furthermore, the mechanical coupling of the devices to the athlete was imperfect. The prototypes were not entirely secured under maximum tension, occasionally allowing the modules to shift out of their initial alignment upon heavy foot strikes, which introduced mechanical noise into the IMU readings. This led to some internal hardware damage to the left Arduino's Bluetooth module restricting the antennas' effective range. Since the system pushed heavy data payloads at a high frequency (50Hz), the degraded BLE module dropped packets whenever the Android device wans't in close-range proximity to the foot. This heavily reduced the overall quality of the collected data for the left foot. Ergonomically, the custom keep-awake circuit required to bypass the power banks' auto-shutoff feature caused slight scratching against the athletes' legs, indicating that future prototypes require softer, fully integrated enclosures.

5.4.1.2 Dataset and Labeling Constraints

From a machine learning and data science perspective, the collected dataset faces generalizability constraints. First, the data collection was limited to a sample size of only two individuals (the primary researcher and one additional subject). Consequently, the trained models may be highly tuned to the specific biomechanical signatures of these two users, requiring a much larger demographic pool to achieve universal accuracy. Secondly, the manual labeling process introduced boundary noise. Because recording was triggered manually via the Android application, the initial and final seconds of many data windows inadvertently captured transitional movements. This minor overlap injected small instances of mislabelled data into the training pipeline. Finally, the collection of physiological fatigue data presented a significant challenge. Safely and reliably inducing and capturing a pre-cramp state in a testing environment is inherently difficult. As a result, gathering stable, sustained RMS baseline values for the peak fatigue threshold was challenging, making the deterministic cramping algorithm highly sensitive and difficult to calibrate with the current limited dataset.

6 Conclusion

With the ongoing development of wearable technology, particularly smart insoles and socks, there has been a significant advancement in basketball motion recognition and injury prevention. Even tho wearables have several restraints, this technology is the most appropriate for basketball motion recognition and injury prevention as its biggest step back is the prohibition in official matches.

The primary objective of this dissertation was to design, implement, and evaluate a wearable system capable of high-frequency basketball movement classification and real-time fatigue monitoring. This research demonstrated that, by adding other types of data to the traditional kinematics only sports wearables, it's possible to not only improve recognition models but also add other systems that help the players improve their performance and health.

6.1 Summary of Achievements

The realization of this project required the end-to-end development of a comprehensive IoT and ML ecosystem. Rather than relying on off-the-shelf software or pre-existing tracking platforms, the entire system architecture was engineered from the ground up. This monumental scope included programming the embedded C++ firmware for the Arduino microcontrollers to handle high-frequency sensor polling and optimized BLE transmission. To bridge the hardware to the cloud, a native Kotlin Android application was developed utilizing an MVVM architecture, serving as the central hub for real-time data processing, user interface management, and InfluxDB database synchronization. Furthermore, the backend infrastructure was custom-built, featuring a Python Flask server deployed for cloud inference, alongside a complete machine learning training pipeline developed in Google Colab to preprocess the time-series data and export the finalized classification models.

Through this architecture, the core hypothesis of validating a multi sensor fusion system was definitively proven. The machine learning training pipeline demonstrated that combining mechanical and physiological data results in a better contextual awareness.

Deploying these models into a live environment further validated the viability of decentralized sports analytics. By quantizing a custom TinyML neural network and embedding it directly into the Android application, the system achieved a robust Edge Computing architecture capable of recognizing basic basketball movements.

Finally, beyond passive activity tracking, the developed ecosystem successfully introduced an active, real-time injury prevention mechanism by fusing the smoothed TinyML kinematic predictions with the EMG features. This logic proved capable of identifying pre-cramp hypertensive muscle states during periods of rest, providing coaches and users with real-time warnings offering the option to rest and massage hte muscles preventing cramping.

6.2 Future Work

While the current prototype successfully demonstrates the viability of multi-modal sensor fusion and dual-tier machine learning in sports wearables, many steps for future research and development remain to transition the system from a proof-of-concept to a robust helpful tool.

6.2.1 Hardware Evolution and Textile Integration

The most immediate hardware improvement involves transitioning from the modular, externally strapped prototype to a fully integrated smart sock. By utilizing miniaturized flexible printed circuit boards and conductive woven threads instead of traditional adhesive EMG electrodes, the system would completely eliminate the mechanical obstacles caused by device shifting and cable strain while being as unobtrusive as possible. Furthermore, integrating the power supply directly into the fabric would resolve the current ergonomic issue, ensuring maximum comfort and zero interference with the athlete's natural motions.

6.2.2 Dataset Expansion and Automated Labeling

Future data collection has to be greater in both quantity and quality. Expanding the demographic pool to include variety in height, weight, skill level, gender and play styles will prevent the algorithms from overfitting to a specific user's biomechanical signature. Additionally the data collection methodology must be refined to eliminate the noise caused by manual app-based labeling.

6.2.3 Clinical Validation of Fatigue Algorithms

Finally, the deterministic cramping prevention system requires rigorous clinical validation. Since safely and consistently inducing a pre-cramp state in a live basketball environment is practically difficult, future data collection for this subsystem should be conducted in a controlled sports science laboratory to properly delimit the fatigue levels. By correlating

the different EMG features and with a better recognition system, the cramping prevention algorithm could be upgraded. Rather than relying on manually calibrated static thresholds, future iterations could employ dynamic, auto-calibrating baselines that adapt to the athlete's specific physiological state in real-time before a game or practice while warming up.

6.3 Final Remarks

The transition from passive data logging to active, real-time physiological monitoring represents the next critical evolution in sports science technology. The prototype developed in this study serves as a proof-of-concept for this shift. While challenges regarding hardware ergonomics and individualized calibration remain, the underlying software architecture, specifically the dual-tier inference pipeline establishes a robust foundation for future iterations.

Ultimately, this dissertation demonstrates that the future of athletic wearables does not lie merely in tracking how fast or how far an athlete moves, but in understanding the underlying neuromuscular cost of those movements. By successfully harmonizing mechanical sensors, electrical physiology, and machine learning, this work contributes a meaningful step forward in the pursuit of optimizing athletic performance and safeguarding player health.

Bibliography

[1] J. Lutz et al. "Wearables for Integrative Performance and Tactic Analyses: Opportunities, Challenges, and Future Directions". In: *International Journal of Environmental Research and Public Health* 17.1 (2020). ISSN: 1660-4601. DOI: 10.3390/ijerph17010059 (cit. on p. 1).

[2] S. Shi et al. "Utilize Smart Insole to Recognize Basketball Motions". In: *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*. 2018, pp. 1430–1434. DOI: 10.1109/CompComm.2018.8780718 (cit. on pp. 1, 15).

[3] M. Peng, Z. Zhang, and Q. Zhou. "Basketball Footwork Recognition using Smart Insoles Integrated with Multiple Sensors". In: *2020 IEEE/CIC International Conference on Communications in China (ICCC)*. 2020, pp. 1202–1207. DOI: 10.1109/ICCC49849.2020.9238862 (cit. on pp. 1, 15).

[4] A. Ç. Seçkin, B. Ateş, and M. Seçkin. "Review on Wearable Technology in Sports: Concepts, Challenges and Opportunities". In: *Applied Sciences* 13.18 (2023). ISSN: 2076-3417. DOI: 10.3390/app131810399 (cit. on p. 5).

[5] F. John Dian, R. Vahidnia, and A. Rahmati. "Wearables and the Internet of Things (IoT), Applications, Opportunities, and Challenges: A Survey". In: *IEEE Access* 8 (2020), pp. 69200–69211. DOI: 10.1109/ACCESS.2020.2986329 (cit. on pp. 5, 10).

[6] H. Nagano and R. K. Begg. "Shoe-Insole Technology for Injury Prevention in Walking". In: *Sensors* 18.5 (2018). ISSN: 1424-8220. DOI: 10.3390/s18051468 (cit. on p. 5).

[7] W. H. Organization. *Low back pain*. 2023. URL: https://www.who.int/news-room/fact-sheets/detail/low-back-pain (cit. on p. 5).

[8] H. Zeng et al. "Intelligent health and sport: An interplay between flexible sensors and basketball". In: *iScience* 27.3 (2024), p. 109089. ISSN: 2589-0042. DOI: https://doi.org/10.1016/j.isci.2024.109089 (cit. on p. 7).

[9] G. M. Migliaccio, J. Padulo, and L. Russo. "The Impact of Wearable Technologies on Marginal Gains in Sports Performance: An Integrative Overview on Advances in Sports, Exercise, and Health". In: *Applied Sciences* 14.15 (2024). ISSN: 2076-3417. DOI: 10.3390/app14156649 (cit. on p. 7).

[10] S. McDevitt et al. "Wearables for Biomechanical Performance Optimization and Risk Assessment in Industrial and Sports Applications". In: *Bioengineering* 9.1 (2022). ISSN: 2306-5354. DOI: 10.3390/bioengineering9010033 (cit. on p. 7).

[11] C. Amaro et al. "Plantar Pressure Evaluation during the Season in Five Basketball Movements". In: *Applied Sciences* 10 (2020-12), p. 8691. DOI: 10.3390/app10238691 (cit. on pp. 8, 15, 16).

[12] M. Hubl et al. "Embedding of wearable electronics into smart sensor insole". In: *2016 IEEE 18th Electronics Packaging Technology Conference (EPTC)*. 2016, pp. 597–601. DOI: 10.1109/EPTC.2016.7861550 (cit. on pp. 8, 10).

[13] A. M. Tan et al. "Development of a Smart Insole for Medical and Sports Purposes". In: *Procedia Engineering* 112 (2015). 'The Impact of Technology on Sport VI' 7th Asia-Pacific Congress on Sports Technology, APCST2015, pp. 152–156. ISSN: 1877-7058. DOI: https://doi.org/10.1016/j.proeng.2015.07.191 (cit. on p. 8).

[14] Y. Tian et al. "Deep-learning enabled smart insole system aiming for multifunctional foot-healthcare applications". In: *Exploration* 4 (2023-11). DOI: 10.1002/EXP.20230109 (cit. on p. 8).

[15] A. Drăgulinescu et al. "Smart Socks and In-Shoe Systems: State-of-the-Art for Two Popular Technologies for Foot Motion Analysis, Sports, and Medical Applications". In: *Sensors* 20.15 (2020). ISSN: 1424-8220. DOI: 10.3390/s20154316 (cit. on pp. 8, 9).

[16] J. Williamson et al. "Data sensing and analysis: Challenges for wearables". In: *The 20th Asia and South Pacific Design Automation Conference*. 2015, pp. 136–141. DOI: 10.1109/ASPDAC.2015.7058994 (cit. on p. 10).

[17] *NBA, NBPA Use Wearables To Study Player Health and Wellness*. 2024. URL: https://athletechnews.com/nba-nbpa-use-wearables-to-study-player-health-wellness/ (cit. on p. 11).

[18] *Injury Prevention Technology? Hawks Players Are Wearing It*. 2015. URL: https://www.nba.com/hawks/features/injury-prevention-technology-hawks-players-are-wearing-it (cit. on p. 11).

[19] C. Lundberg. "Data collection, management and visualisation for IoT healthcare devices". MA thesis. Technical University of Denmark, 2020 (cit. on pp. 11, 12, 16).

[20] C. R. H. Kjær and C. B. Henriksen. "Smart Sole- Classi cation of lifts using IoT technology". MA thesis. Technical University of Denmark, 2021 (cit. on pp. 11, 17, 18).

[21] B. Wang et al. "FreeWalker: a smart insole for longitudinal gait analysis". In: *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2015, pp. 3723–3726. DOI: 10.1109/EMBC.2015.7319202 (cit. on pp. 12, 15).

[22] D. Levine, J. Richards, and M. Whittle. *Whittle's gait analysis*. 2012-07. ISBN: 9780702042652 (cit. on pp. 12, 13).

[23] B. F. D. Cardoso. "Sensors fusion and movement analysis for sports performance optimization". MA thesis. Faculdade De Engenharia Da Universidade Do Porto, 2017. URL: https://repositorio-aberto.up.pt/bitstream/10216/108411/2/226423.pdf (cit. on pp. 12, 17–19).

[24] E. Chumanov, C. Remy, and D. Thelen. "Tracking the Position of Insole Pressure Sensors during Walking and Running". In: () (cit. on p. 15).

[25] T. Holleczek et al. "Textile pressure sensors for sports applications". In: *SENSORS, 2010 IEEE*. 2010, pp. 732–737. DOI: 10.1109/ICSENS.2010.5690041 (cit. on p. 15).

[26] C. Spiewak et al. "A Comprehensive Study on EMG Feature Extraction and Classifiers". In: *Open Access Journal of Biomedical Engineering and its Applications* 1 (2018-02). DOI: 10.32474/OAJBEB.2018.01.000104 (cit. on pp. 19–23).

[27] S. M. Sid'El Moctar, I. Rida, and S. Boudaoud. "Comprehensive Review of Feature Extraction Techniques for sEMG Signal Classification: From Handcrafted Features to Deep Learning Approaches". In: *IRBM* (2024-11), p. 100866. DOI: 10.1016/j.irbm.2024.100866 (cit. on pp. 20–24).

[28] X. Chen et al. "Hand Gesture Recognition Based on High-Density Myoelectricity in Forearm Flexors in Humans". In: *Sensors* 24 (2024-06), p. 3970. DOI: 10.3390/s24123970 (cit. on pp. 21–23).

[29] L. Quesada et al. "Emg Feature Extraction and Muscle Selection for Continuous Upper Limb Movement Regression". In: (2024-09). DOI: 10.2139/ssrn.4953765 (cit. on pp. 21–23).

[30] R. Rallan and V. Garg. "REVIEW OF EMG SIGNAL CLASSIFICATION APPROACHES BASED ON VARIOUS FEATURE DOMAINS". In: *MATTER: International Journal of Science and Technology* 6 (2021-01), pp. 123–143. DOI: 10.20319/mijst.2021.63.123143 (cit. on pp. 21–23).

[31] I. Ben Abdallah, Y. Bouteraa, and A. Alotaibi. "Hybrid EMG-NMES control for real-time muscle fatigue reduction in bionic hands". In: *Scientific Reports* 15 (2025-07). DOI: 10.1038/s41598-025-05829-w (cit. on pp. 21, 23).

[32] H. Hellara et al. "Comparative Study of sEMG Feature Evaluation Methods Based on the Hand Gesture Classification Performance". In: *Sensors* 24.11 (2024). ISSN: 1424-8220. DOI: 10.3390/s24113638 (cit. on pp. 21, 22, 24).

[33] P. Qin and X. Shi. "Evaluation of Feature Extraction and Classification for Lower Limb Motion Based on sEMG Signal". In: *Entropy* 22.8 (2020). ISSN: 1099-4300. DOI: 10.3390/e22080852 (cit. on pp. 21, 22, 24).

[34] F. Amitrano et al. "Measuring Surface Electromyography with Textile Electrodes in a Smart Leg Sleeve". In: *Sensors* 24.9 (2024). ISSN: 1424-8220. DOI: 10.3390/s24092763 (cit. on p. 21).

[35] Q. Ai et al. "Research on Lower Limb Motion Recognition Based on Fusion of sEMG and Accelerometer Signals". In: *Symmetry* 9.8 (2017). ISSN: 2073-8994. DOI: 10.3390/sym9080147 (cit. on pp. 21, 24).

[36] X. Xi et al. "Evaluation of Feature Extraction and Recognition for Activity Monitoring and Fall Detection Based on Wearable sEMG Sensors". In: *Sensors* 17.6 (2017). ISSN: 1424-8220. DOI: 10.3390/s17061229 (cit. on pp. 22–24).

[37] A. Kamaruddin, P. I. Khalid, and A. Sha'ameri. "The Use of Surface Electromyography in Muscle Fatigue Assessments-A Review". In: *Jurnal Teknologi* 74 (2015-05), pp. 119–124. DOI: 10.11113/jt.v74.4676 (cit. on p. 25).

[38] H. M. Qassim et al. "Proposed Fatigue Index for the Objective Detection of Muscle Fatigue Using Surface Electromyography and a Double-Step Binary Classifier". In: *Sensors* 22.5 (2022). ISSN: 1424-8220. DOI: 10.3390/s22051900 (cit. on p. 25).

[39] S. Otálora et al. "Data-Driven Approach for Upper Limb Fatigue Estimation Based on Wearable Sensors". In: *Sensors* 23.22 (2023). ISSN: 1424-8220. DOI: 10.3390/s23229291 (cit. on p. 26).

[40] S. Hwang et al. "A Multimodal Fatigue Detection System Using sEMG and IMU Signals with a Hybrid CNN-LSTM-Attention Model". In: *Sensors* 25 (2025-05), p. 3309. DOI: 10.3390/s25113309 (cit. on p. 26).

[41] Z. Baigarayeva et al. "Machine learning-based classification of local muscle fatigue using electromyography signals for enhanced rehabilitation outcomes". In: *International Journal of Electrical and Computer Engineering (IJECE)* 15 (2025-12), p. 5954. DOI: 10.11591/ijece.v15i6.pp5954-5967 (cit. on p. 26).

[42] K. Roeleveld, B. Van Engelen, and D. Stegeman. "Possible mechanisms of muscle cramp from temporal and spatial surface EMG characteristics". In: *Journal of applied physiology (Bethesda, Md. : 1985)* 88 (2000-06), pp. 1698–706. DOI: 10.1152/jappl.2000.88.5.1698 (cit. on p. 26).

[43] J. Chaney and R. Sherman. "Changes in muscle tension patterns predicting the start of nocturnal leg cramps: A pilot study". In: *Annals of Psychophysiology* 10 (2023-06), pp. 08–11. DOI: 10.29052/2412-3188.v10.i1.2022.08-11 (cit. on p. 26).

[44] *Arduino Nano 33 BLE Rev 2 Documentation*. URL: https://docs.arduino.cc/hardware/nano-33-ble-rev2/ (cit. on pp. 28, 29, 34).

[45] *Seniam (Surface ElectroMyoGraphy for the Non-Invasive Assessment of Muscles) Project Website*. URL: http://seniam.org/ (cit. on pp. 51, 52).