BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

SOFTWARE ENGINEERING

MASTER’S THESIS

IoT and Fog Computing for Wellness Solutions

Supervisor

Lect. Dr. Ioan Lazar

Author

Rareș Tudor Rus

2024

Abstract

This dissertation explores the integration of Internet of Things (IoT) and Fog Computing technologies to create innovative wellness solutions. The main objective is to design and implement a system capable of monitoring both the exercises inside a gym and the state of the devices in real time, ensuring data confidentiality and security, optimising energy efficiency and maintaining scalability and flexibility. The system architecture includes hardware components such as Arduino for sensor data collection, Raspberry Pi for central processing, a server for data storage, and an Android application for user interaction. By using Fog Computing, the system reduces latency and improves data processing efficiency by bringing computing resources closer to the data source. This approach is particularly beneficial in handling large volumes of data generated by IoT devices in fitness and wellness applications. The study demonstrates how these technologies can be effectively combined to promote an active and healthy lifestyle, providing a practical case study for technology developers and researchers.

Table of contents:

Chapter 1

1. Introduction

1.1. Project Context

The main goal of this work is to explore new horizons in the implementation of Fog Computing using IoT technology. In recent years, the integration of IoT with Fog Computing, Cloud Computing, and Edge Computing has been able to successfully revolutionise various real-world fields such as healthcare, smart cities, and industrial automation. We propose to expand these areas with a new one, that of fitness and well-being, considering that an active and healthy lifestyle is as important as the benefits offered by the other areas mentioned.

Fog computing, which extends the capabilities of cloud computing by bringing data processing closer to the source, plays a crucial role in addressing the latency and bandwidth challenges caused by the large amount of data generated by IoT devices. This approach not only improves response times and reduces network congestion, but also improves data security by processing sensitive information locally.

As technology constantly evolves at a rapid pace, IoT devices are becoming more powerful, the volume of data they transmit increases, the speed at which they transmit data accelerates, and the accuracy with which they record data improves. This factor, together with the ability to optimise a system through the implementation of Fog Computing, guarantees the success of a system that manages large volumes of data that require processing and pre-processing. In this context, we will implement a system that helps users maintain their best conditions for an active lifestyle, including exercise and sports.

Despite the rapid technological advancements and increasing power and precision of IoT devices and various applications of Fog Computing, there is still a problem as the application of these technologies in the fitness and wellness industry remains underexplored. This lack of applications focused on this part highlights the need for innovative solutions that can capitalise on the strengths of both IoT and fog computing to create an active and healthy lifestyle.

The main audience for this project includes people who are keen to maintain an active and healthy lifestyle through regulated and practical health exercises. In addition, the special places set up for sports by the administration of a city and private sport centres, but also fitness trainers who seek to provide more precise and data-based guidance privately will also benefit from this system. Furthermore, the project addresses technology developers and researchers in the field of IoT and fog computing, providing them with a case study of how these technologies can be integrated into the fitness and wellness sector.

1.2. Project Objectives

The primary objective of this thesis is to design and implement an IoT and fog computing-based wellness solution that addresses the following key goals:

- **Real-time Health Monitoring:** The development of a system constantly monitors various health parameters but also various events that happen especially arranged for sports, places, sensors and IoT devices. The system should be able to collect, process and analyse data in real time to provide immediate feedback to the users regarding the requirements they have for the system developed by us.
- **Data Privacy and Security:** We assure you that both the health data collected from users and the data collected from various facilities in specially designed sports venues are processed locally using fog computing principles to improve data privacy and security. The system should implement robust encryption and authentication mechanisms to protect sensitive information from threats from malicious actors.
- **Energy Efficiency:** Optimise the system to be energy-efficient, ensuring that IoT devices and fog nodes operate with minimal power consumption. This includes implementing energy-saving algorithms and utilising low-power communication protocols. Because the number of fog elements could increase drastically and that would be very power consuming

- **Scalability and Flexibility:** Design the system architecture to be scalable and flexible, enabling easy integration of new sensors, devices and functionality. This is necessary because the basic principles of fog computing support adding as many elements as possible to a system. The system should be able to adapt to varying user needs and match future advancements in IoT and fog computing technologies. It is necessary to anticipate the directions that the evolution of hardware and software are taking, better options may appear at any time.
- **User-friendly Interface:** Creați o interfață ușor de utilizat, care să permită utilizatorilor să interacționeze fără efort cu sistemul de wellness. Aceasta include proiectarea de tablouri de bord intuitive, aplicații mobile și interfețe pentru dispozitive portabile care afișează valori de sănătate și recomandări personalizate.

Chapter 2

2. Analysis and Theoretical Foundation

This chapter is one of the busiest in this thesis since, in order to conduct clear and successful research, the project's theoretical underpinning must be adequately designed and thoroughly understood. The major purpose of this chapter is to introduce the reader to the libraries, frameworks, and concepts utilised to create the final application.

At the same time, this chapter will deal with creating an overview of the technical specifications needed to implement a fog computing system, in which the focus is on security and on optimising energy consumption according to the explanations in Chapter 1.

2.1. Internet of Things

The Internet of Things (IoT) refers to the complete network of devices that may gather data from their surroundings and communicate with other devices of the same or more complicated type. This means that with the help received from the communication networks and from the hardware and software available nowadays we can have devices such as coffee machines, cars, vacuum cleaners, televisions that use their integrated sensors to interact with the needs of the user. The term "Internet of Things" was first used in a presentation, {1}, in the summer of 1999, in this presentation, Kevin Ashton stated a combination between Radio Frequency Identification (RFID) and the Internet. In the years that followed, new types of connecting them to the Internet appeared, such as RFID, and Near Field Communication (NFC), but also WiFi and Bluetooth. IoT protocols such as MQTT, CoAP and HTTP/HTTPS enable devices to communicate efficiently and securely. In terms of security, the use of encryption methods and security protocols help protect these devices from cyber attacks.

These IoT devices are classified into many categories based on their applicability and complexity, which were identified in the following human demands for integration in various circumstances. In {Fig 1} we can see all these categories, and with light blue we can see the field of interest of these works.

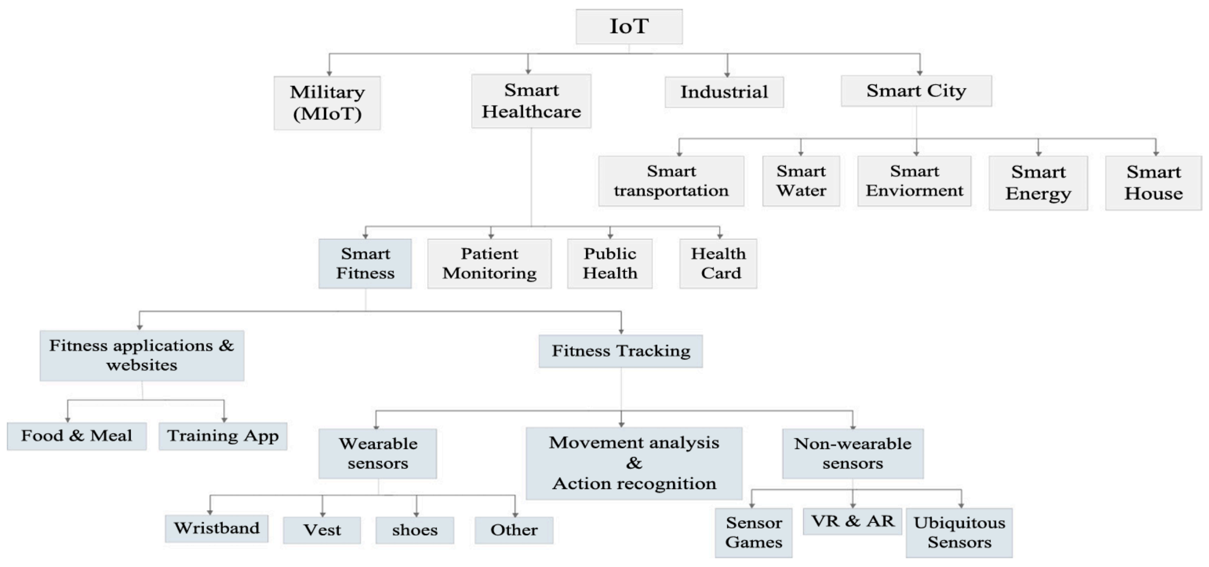


Figure 1 - IoT solutions & categories

IoT systems are composed of multiple interconnected layers, each serving a unique purpose and collectively enabling smart functionalities across various devices and applications. The initial layer, called Sensory Interface, is composed of the physical components (devices and sensors) that interact directly with the environment. These components are crucial for capturing real-time data on everything from environmental conditions to user interactions. Next layer ensures that data collected by sensors is transmitted to further processing points or to the cloud. It encompasses the networks and protocols that maintain data flow, ensuring both efficiency and security in data transmission, and is called Connectivity Infrastructure Layer. Often referred to as the middleware layer, Service Management Layer, this segment handles the complex tasks of data aggregation, device management, and software-based message routing. It acts as a critical bridge, connecting the hardware aspects with the application-specific components of the IoT system. Next, is the Application Spectrum Layer where the processed data is utilised in tailored applications that meet specific end-user or operational needs. Last layer is Enterprise Considerations situated at the top of the architecture, this layer focuses on integrating IoT insights into business strategies. It involves data analysis and the application of these insights to improve business efficiency, decision-making, and market responsiveness.

2.2. MQTT

MQ Telemetry Transport, or MQTT, is a lightweight messaging system that enables effective device-to-device communication in limited settings. It works particularly effectively for applications like Internet of Things (IoT) devices when power and bandwidth are scarce. In the context of this project, MQTT is utilised to facilitate communication between various IoT devices and the central server. Its lightweight nature and efficient messaging capabilities are crucial for real-time data transmission and remote device management. MQTT ensures that the system remains scalable, secure, and reliable, thereby supporting the overall goal of creating an efficient and responsive IoT solution. By integrating MQTT into the system, we leverage its strengths to provide a robust communication layer that can handle the dynamic and diverse needs of IoT applications.

MQTT is an ideal protocol for machine-to-machine communication, especially in the high latency and low network bandwidth environments that are common in the IoT context. Today, almost every electronic device is designed to send and receive information from other devices, simplifying users' tasks. MQTT excels at enabling these devices to communicate efficiently and reliably. One of the key strengths of MQTT is its lightweight nature. It is highly efficient with a small footprint, making it suitable for resource-constrained environments. This efficiency is complemented by support for different levels of quality of service (QoS), which ensures reliable delivery of messages even in unstable environments, such as those using Wi-Fi, Bluetooth or satellite wireless connections.

MQTT provides three levels of QoS to meet different reliability needs. QoS 0, also known as "at most once delivery", delivers the message based on the best effort of the underlying network. There is no acknowledgment of receipt from the recipient, and the message is not stored or retrieved. This level is suitable for scenarios where occasional message loss is acceptable, such as live sensor data streams. QoS 1, or "at least once delivery", guarantees that the message is delivered at least once to the recipient. The sender stores the message until it receives an acknowledgment from the recipient. However, this can lead to duplicate messages if the acknowledgment is lost and the message is present. This level is ideal for applications where message delivery must be ensured but duplicates can be tolerated, such as event or command logging. QoS 2, known as "exactly once delivery", ensures that the message arrives exactly once. This is the most reliable level, involving a two-phase confirmation process to prevent duplication. It ensures that the message is neither lost nor duplicated, making it suitable for critical operations such as financial transactions or control systems.

2.3. Edge Computing

Edge computing is a computing paradigm that processes data from clients closer to them, contrary to the technique presented above. Practically, this practice brings all the resources needed by the system that serves the user to provide him with a result. Edge Computing appeared following the advancements made in the field of IoT devices because they began to transmit large volumes of data to the cloud, and this led to bandwidth blocking and latency issues. These problems were solved by processing data locally and only important data or those necessary to execute more complicated requests that needed resources that were not available locally were sent to the cloud servers. Moreover, a layer of security came along with this type of computing, that of security, because the data is processed locally and no longer passes through the Internet where it can be hit by malicious attacks. As we said, the integration of Edge Computing with that of Cloud Computing is needed to meet the need for data storage and to solve processing problems.

An essential thing for which edge computing still remains a current practice is the integration of modern software practices. Containers and microservices are key, providing modular and lightweight solutions that increase the agility and scalability of Internet edge applications. These technologies enable the rapid deployment and management of applications on numerous devices, making them ideal for the dynamic environments typical of edge computing cases.

According to the study {2}, the main need for Edge Computing is for the efficient management of large volumes of data generated by IoT devices. Even though cloud computing offers huge processing capabilities, the increase in data flow puts enormous demands on network bandwidth, thus creating a bottleneck. For example, a car generates considerable amounts of data, reaching up to 2 Gigabits, which requires fast processing for real-time decisions, which would not be possible by sending the data to the cloud due to delays. In addition, most electrical devices will be part of the IoT and become both producers and consumers of data, making traditional cloud processing insufficient. By processing data directly at the source, Edge Computing not only reduces pressure on networks, but also improves privacy and energy efficiency, adapting to changing needs from consumers to producers of data.

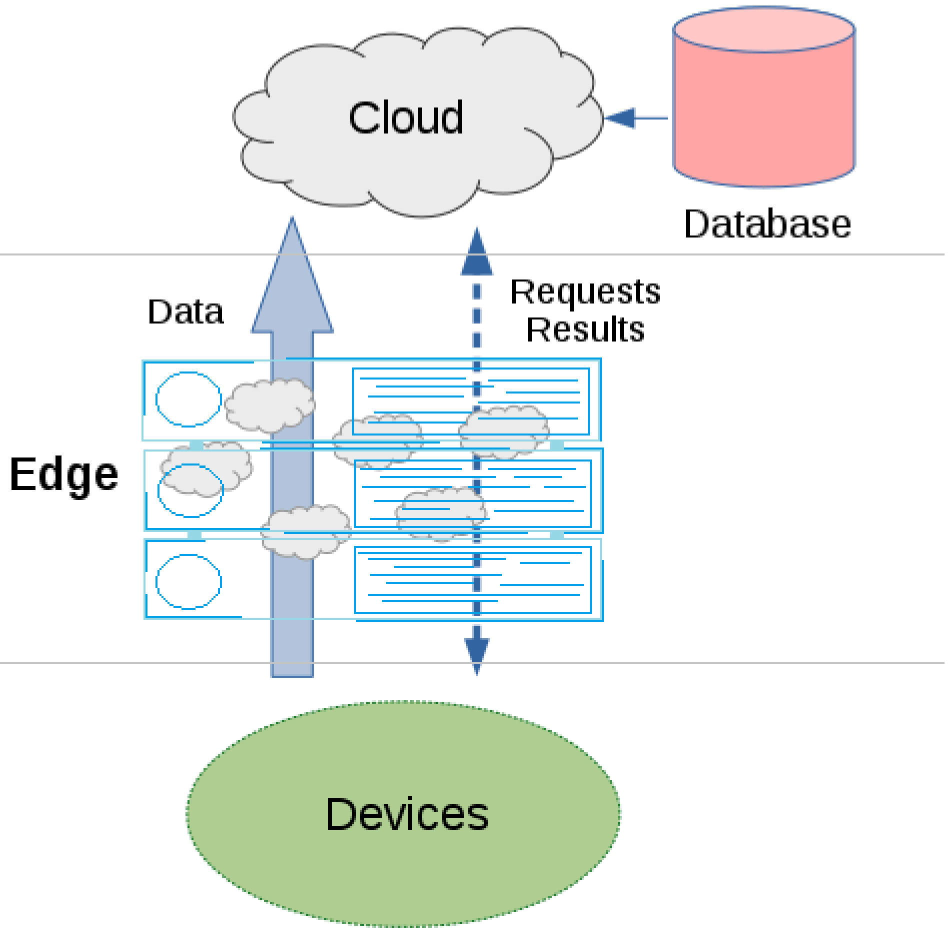


Figure 2 - Edge computing paradigm

2.4. Cloud Computing

Cloud computing is accessing the necessary services or place to store data or necessary software or data analysis,explained in more detail, it allows the use of solution resources that can be easily obtained and configured. This concept was introduced in the 1960s, 40 years later, in the 2000s, Amazon launched Amazon Web Services (AWS), which changed the evolution and development of software. It has several service models, these are Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS), each with a different purpose and coming from different needs. IaaS provides the user with the basic computing infrastructure (servers, storage, network) to which he has full access, which he can use as he wishes and run applications, AWS EC2, Google Compute Engine, Microsoft Azure VMs are some examples that can offer something like this. The next type of service, PaaS, which offers the user a very easy way to implement, test and deploy applications. Some examples of this type are Google App Engine, AWS Elastic Beanstalk, Microsoft Azure App Services, these appeared with the aim of relieving him of the headache necessary for choosing and allocating resources. The last service is SaaS, which offers the user software applications through various means of using the internet, nothing can be changed that means the part of cloud resources. Examples for this are many and well-known, some categories are Google Workspace, Microsoft Office 365, Salesforce.

These services specified above need a way to be hosted and for this we need a method to deploy them, a few have been established so far and these are the 3 most used Public Cloud, Private Cloud and Hybrid Cloud . The first, the public one, where the cloud environment is offered by a general provider that makes it available to certain users, without too many restrictions in their choice, some examples are AWS, GCP and Azure. Private Cloud, is dedicated to a group of people who are part of the same organisation and have a common goal for this virtual environment, it is often used when sensitive data is used in an organisation's project. The last is a combination between the 2, because some resources can be modified and owned by a user or a group of users and they can be used in a private network, for example sensitive data can be on a private cloud and which are not important on a public server to save space. The 2 most famous examples for this are AWS Outposts and Azure Arc.

Servers and storage are examples of physical resources that may be virtualized. By doing this, you may operate several virtual computers on a single physical system, which reduces hardware costs and boosts productivity. Numerous benefits come with virtualization, including better disaster recovery, higher resource efficiency, and improved security through application isolation. Nevertheless, depending on the particular virtualization software utilised, there may also be drawbacks such as possible performance overhead, more complicated maintenance, and vendor lock-in.

In conclusion, cloud computing offers a double-edged sword. In a positive way it's very economical and scalable, letting consumers pay for what they use and access resources whenever they need them. Furthermore, cloud services facilitate remote work and collaboration since they are easily accessible from any location with an internet connection. Additionally, cloud providers manage upgrades and maintenance, freeing up internal IT resources. The drawbacks include dependency on reliable internet access, security issues with keeping sensitive data, and restricted control over infrastructure. In addition, the possibility for vendor lock-in makes provider switching challenging, and data protection regulation compliance adds even another level of complication. Everything is summed up in Figure 3 below.

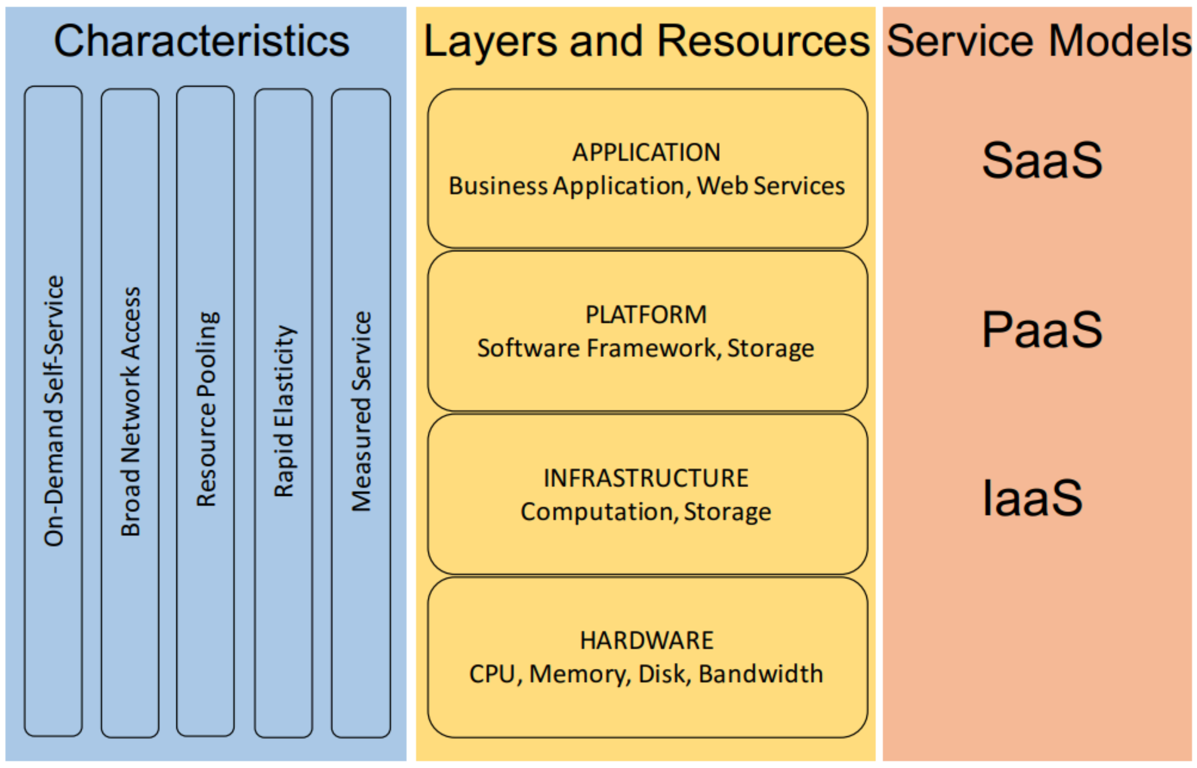


Figure 3 - Cloud Computing: Characteristics, Layers, and Service Models

2.5. Fog Computing

Fog computing is an architecture that uses edge devices to take care of an important part of processing, storage and local communication and then send it via the Internet to a cloud platform. This principle was developed with the aim of moving the expensive processes that were done on the cloud and required important resources to the edge of the Internet, where an edge device can take care of a specific type of process, thus obtaining a general performance especially for applications that process a lot of data. In {3}, the foundations of this type of computation are laid to facilitate the use of IoT devices and to develop a method by which to obtain what we said above. Therefore, the authors presented a new type of edge component, called fog cell, which aims to mimic an already existing cloud, but with a more specific purpose, so the computing and storage power is lower. This element is between cloud and edge and aims to decrease latency and execution time. Moreover, in this paper the importance of the distribution of this type of cell is presented, for example for a smart city it is more important that an element of the kind that helps generate the best route to a destination should be in areas where more foreigners of that city arrive, such as in airports or train stations.

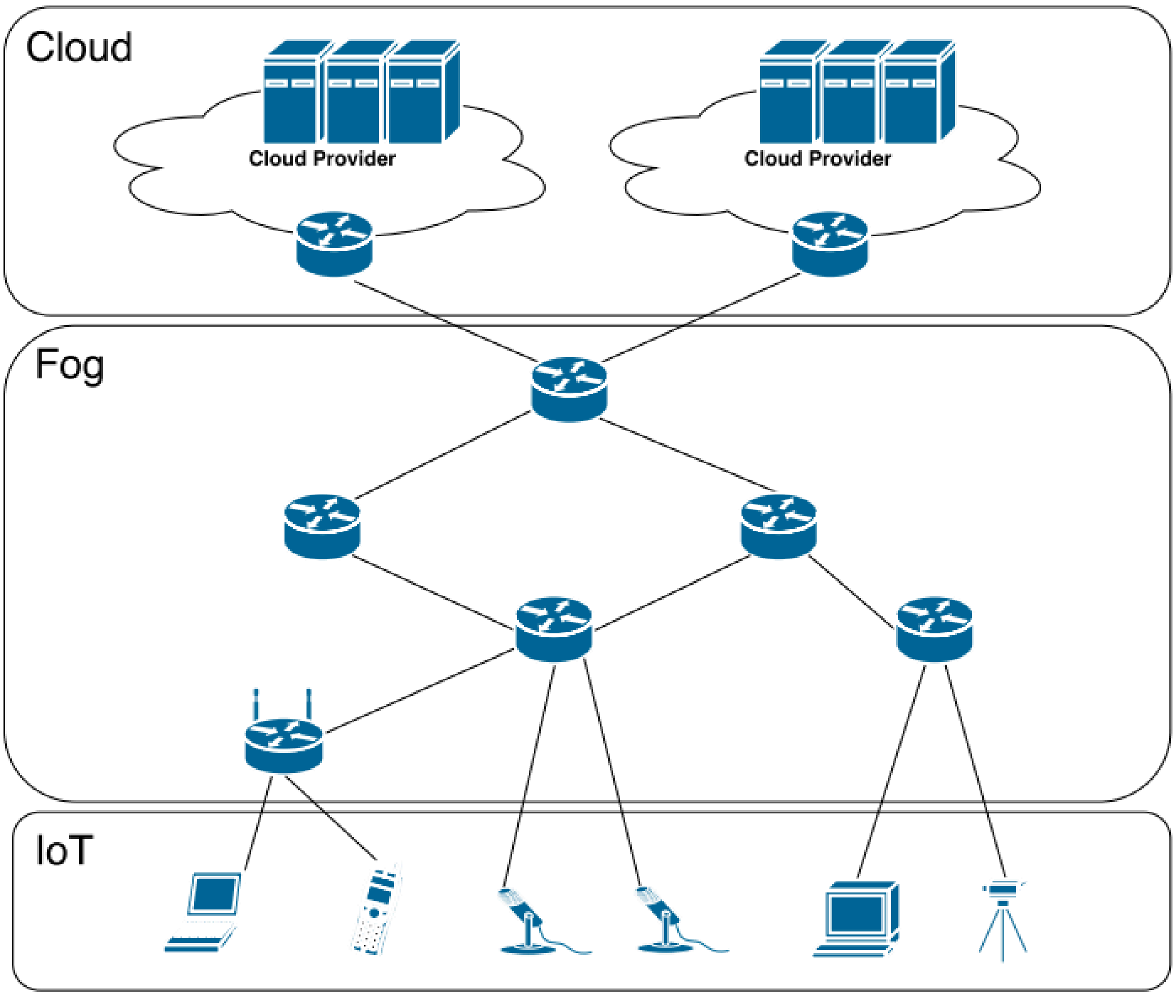


Figure 4 - Fog Computing Layers

In Figure 4 we can see the conceptual architecture of this principle and we can get an idea of how it can be applied in the target areas, such as health, smart cities, automation and so on. A very important thing that these fog nodes offer is the fact that communication between them and implicitly with the edge ones can be done through several methods. Both the Internet and Bluetooth can be used, thus offering enormous flexibility when implementing different systems.

The fact that these cells each handle a specific type and number of edge elements only benefits the system in which they are used. For example, if sensors are used to transmit a large flow of data quickly, we can assign a single element to deal with this sensor, and only the important data is sent to another actor, which could be a fog node or even a cloud server. This approach also provides an aspect of greater security since sensitive data is transported to a location closer to its source and thus processed nearby, making the task of encoding and safeguarding processing or preprocessing results easier. As a result, the main benefits of this type of computing are that data reaches its destination faster, whether it is the cloud centre or the user, that less bandwidth is consumed by processing and filtering data at the edge, that only relevant and processed data is sent to the cloud, and, last but not least, security.

Let's not forget that everywhere there are disadvantages, and this is also the case here, here are the main problems with Fog Computing. The management of an infrastructure of this kind introduces a higher degree of complexity than with centralised systems, where resources are concentrated in much fewer places. Another aspect of this strategy that is disliked is the management component, because because there are so many parts in the game, they must know more or less about each other and communicate with one another, and improving an element necessitates modifications to the waterfall. At the same time, monitors require more labour than other types of systems to ensure error resolution and balance of calculation volume. The kind of data transmission must also be considered since you must carefully pick between practically every element, as opposed to other areas where it is sufficient to select a form of communication and a communication protocol. As a conclusion, when the number of units doing calculations or collecting data grows, management issues emerge.

2.6. Software Frameworks

In the following part of this chapter, a slightly more theoretical part of each program used to create the software part will be presented. Each framework used is chosen with the aim of fulfilling the purpose of a part of the complete system presented in this work. The aspects that these frameworks were based on are the following: other case studies presented in Chapter 3, ease of integration and solving problems and bugs that appear after implementation and running, how future proof are these development tools and not in the future giving feedback from the users to determine the overall fitness of them. Each sub part of what is next in this chapter will provide only the reasons of why the framework was used and for what part of the system.

Android Studio

Android Studio is specially developed to create applications that run on Android operating systems, it is designed in such a way that it is built in with all the resources that a user needs to develop an application.It has a visual layout editor, allowing the user customise the application's interface by dragging and dropping User Interface (UI) elements into an editor that generate XML files by hand. The XML files are the files where the code for UI pages is written. Also, an Application Package Android (APK) Analyzer, is a smart feature that reduces the size of the application and also the time spent on debugging. Android Emulator, letting the user to test an application on multiple virtual devices so the application can work well across many different devices. So as a result, it becomes very helpful when somebody who doesn't have access to multiple devices. Flexible build system, is a feature that lets the user configure different builds of the same project for different devices, helping with the management of the dependencies and configurations.

Arduino IDE

Arduino Integrated Development Environment is used most often when it comes to programming boards with microcontrollers and which have connection pins, it is used to write, compile, run and debug for these boards. It has a very simple and easy-to-navigate interface, whose main components are the code editor, a toolbar and a place for messages after debugging. It comes with numerous libraries that help in solving common problems such as controlling sensors and communication modules. The serial monitor allows communication with the Arduino board via serial communication, useful for debugging and monitoring data. The main feature for which this IDE is so well known, besides being easy to use, is the Board

Manager, which makes it easy to add support for different Arduino and third-party boards.

Node Red

Node Red is a flow based programming tool that makes it easy to create event driven applications in a simple way. It is built on top of NodeJS and provides a browser based editor for wiring flows together. We chose this framework due to the fact that it can run on devices such as Raspberry Pi, Android devices, Arduino and even in cloud environments. It is used in scenarios spanning a wide range of industries from manufacturing and utilities, healthcare and agriculture, home and industrial automation. Node-RED's browser-based visual editor allows users to connect devices, APIs, and online services. The visual nature of this editor simplifies the development process, allowing users to create complex workflows without extensive coding knowledge. This makes it an excellent choice for rapid prototyping and development, especially in environments where rapid iteration and flexibility are crucial. Node-RED also supports a wide range of communication protocols, including MQTT, HTTP, WebSockets, and more. This makes it an ideal hub for IoT applications where devices may need to communicate using different protocols.

Visual Studio

Visual Studio is an integrated development environment (IDE) developed by Microsoft, designed to create a wide range of applications on multiple platforms, including Windows, web, mobile and cloud. It was chosen for this project for its comprehensive development toolset, ease of integration with multiple programming languages, and robust debugging and testing capabilities. It also includes full support for version control systems such as Git and Team Foundation Version Control (TFVC), making it easy to manage code changes and collaborate with team members. Its integration with Azure DevOps services further facilitates continuous integration and continuous deployment (CI/CD) pipelines, enabling automated build, test and deployment processes.

Visual Studio is used in the context of this project to develop and maintain backend services, the API and other critical components of the central server. Its robust feature set ensures that our development process is efficient and that the final product is reliable and maintainable.

SQL Server Management Studio

SQL Server Management Studio (SSMS) is a comprehensive tool for managing, configuring, and administering Microsoft SQL Server. It is chosen for this project because of its powerful database management features, ease of use, and seamless integration with SQL Server. SSMS provides a graphical interface that simplifies complex database administration tasks, including creating, configuring, querying, and managing database security. Its easy-to-use interface allows database administrators and developers to manage databases with minimal effort, ensuring that data operations are efficient and accurate. SSMS provides comprehensive performance tuning and monitoring tools that are critical to maintaining database health and efficiency. Features such as execution plan analysis, indexing recommendations, and real-time performance metrics help optimise query performance and ensure normal database operation.

2.7. Hardware Components

In this section of the subchapter, we introduce the hardware components—the physical elements employed to accomplish the objectives outlined at the outset of this documentation. These components primarily function to capture and transmit data from their sensors, as well as to process and analyse specific data sets. The selected components are tailored to meet the minimum requirements of the application, illustrating that this implementation can be adapted for use on a wide range of devices. Additionally, the versatility and efficiency of these components ensure that the system remains scalable and adaptable to various operational contexts.

Arduino

The Arduino hardware component is somewhat complex, as it functions as an open-source mini computer capable of controlling various components, such as sensors and buttons. These boards feature a microcontroller, acting as the processor where command lines are executed. Additionally, they come with pins that allow the connection of other components (e.g., motors, LEDs, sensors) and require a power supply. The programs running on these boards are written in Arduino IDE, a simplified version of C++. In this paper, we use these programs to capture data from sensors placed on devices within the room and transmit this data to a more powerful component for processing. The specific model employed in this research is the Arduino Uno, which is the most basic version. It includes 14 digital pins, suitable for LEDs, buttons, and sensors, which suffice for our current needs. Furthermore, it has 6 analog pins designed to read data from sensors that provide analog signals, such as temperature sensors.

Raspberry Pi

This component is used as a fog node, which aims to interconnect devices with sensors that capture data and with the cloud server where they are stored and processed. At the same time, data is processed using this device. This model B of Raspberry Pi, version 2, was chosen, because it allows connection to the Internet or bluetooth using auxiliary parts. These types of connection are essential to achieve the purpose of this fog node, also essential is the processing power that the Quad-core ARM processor, with a frequency of 900 MHz results. This model has 1 GB LPDDR2 SDRAM which is more than enough.

Smartphone and Smart Devices

The smart devices used had the purpose of taking data from the user and presenting a graphic interface where the user can interact with the system. Sensors are presented below in detail, because they represent the main source of data, and we need to study them to understand how we can detect a source of human movement. The phone model used to implement and develop the practical part of this work is a Samsung FE, which has all the features we need. To demonstrate the portability of the application and its ease of use, we also chose a smartwatch that runs the Android system and allows applications developed in its own regime, this is a Samsung Watch 4.

Sensors

The devices mentioned above contain a variety of fascinating and useful functions, but this thesis concentrates on the sensors. In particular, the accelerometer and gyroscope sensors are responsible for providing raw data from the user that is analysed and utilised to forecast human behaviour. Both of the sensors stated above are inertial sensors, which rely on inertia and appropriate measurement principles. Sensor devices can be divided into two types: piezoelectric sensors and microelectromechanical systems (MEMS)-based. Piezoelectric devices feature a crystal layer composed of positive and negative ions, known as Piezoelectric material, sandwiched between two electrodes, one with a positive charge and the other with a negative charge. When a force is applied, this structure distorts and the electrons of these ions shift about, resulting in an electric charge (output voltage). This charge is proportional to the force being applied, therefore it can measure the force specified. This method is illustrated in Figure 5 below.

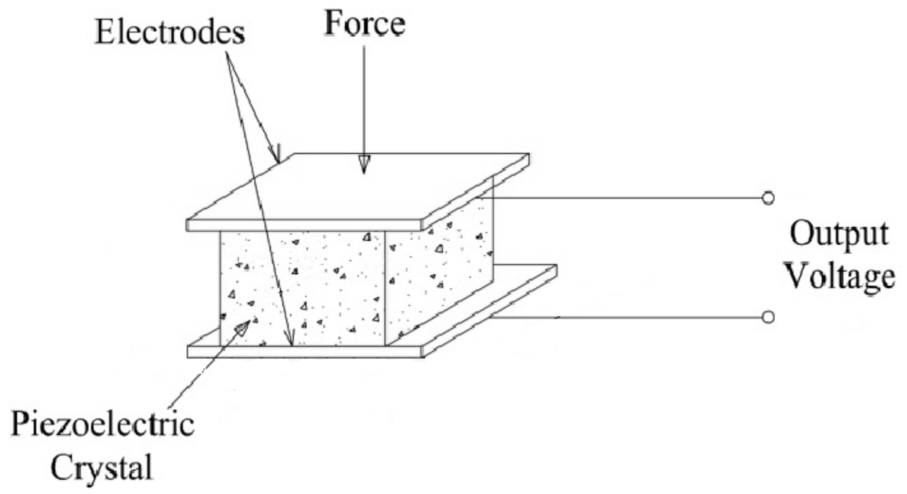


Figure 5 - Arrangement of piezoelectric material between electrodes.

Microelectromechanical systems (MEMS) are tiny machines that quantify force via capacitance changes. It is made up of two sets of plates, one fixed called "Anchor" and one spring-loaded called "Mass" that may move in one direction at a time and respond to acceleration. When an acceleration is given in a specific direction, the mass moves and the capacitance between the Anchor plates and the Mass plates changes. This shift between the plates is measured and processed to get the initial acceleration. Figure 6 shows the device in action by exhibiting a MEMS accelerometer when no acceleration is applied and subsequently when acceleration is used.

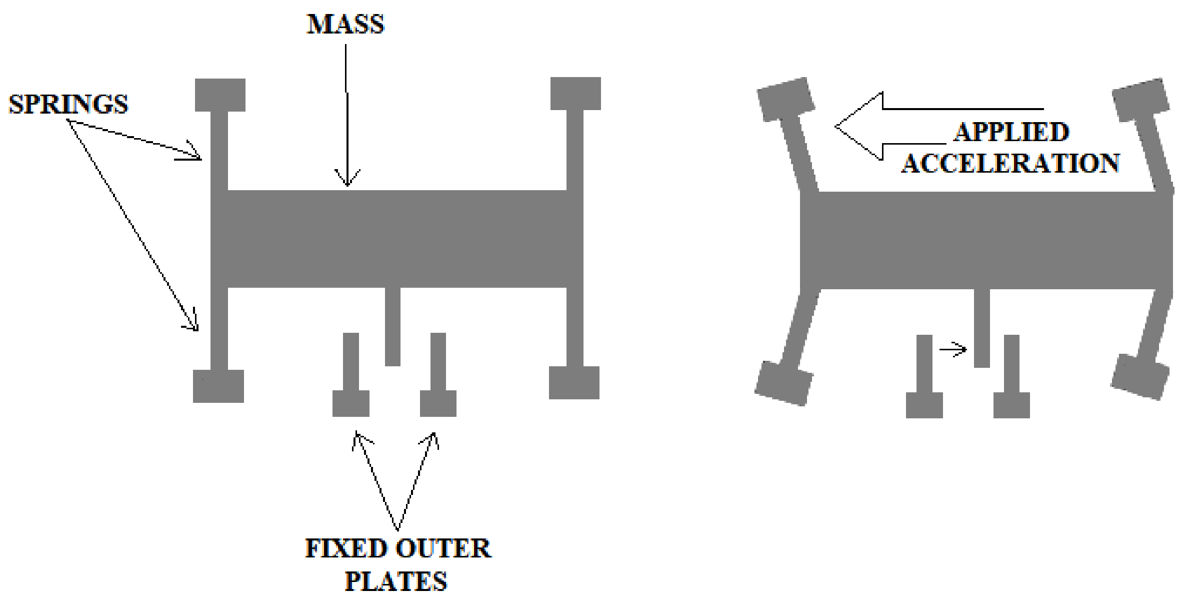


Figure 6 - MEMS accelerometer structure.

Chapter 3

3. Bibliographic Research

3.1. Internet of Things (IoT) Technologies

One of the most important problems in the development of technology for the medical system is its integration with the legal side and the respect of the laws everywhere. {1.1} paper studies the health monitoring system improved by the transition from 4G to the 5th generation of technologies for wireless cellular networks and the use of specific IoT technologies. The significant advances that are brought by this transition are: the improvement of communication between processes and the power of real-time data collection, with the aim of improving patient care. All this, while legal issues regarding data confidentiality and their protection are carefully handled. The main debate in this article is between the benefits that technology brings to the medical system, but at the same time the legal and moral issues that must be respected in order to be used brilliantly. A product that will serve a hospital, pharmacy or so on must strictly respect the data security part in order not to put sensitive information about the diseases that a patient may have.

Moreover, this product must be permanently stable because any kind of error, no matter how small, can be vital for a user or a patient, therefore any piece of software or hardware in such a system must be studied and chosen with care, all these legal parts must be respected. At the same time, after these legal barriers, the possibilities of developing health monitoring systems that already exist can be infinite, because the differences between 4G and 5G are immense. The continuous development of technology brings more and more improvements to the IoT part, which means a safe evolution of this field.

In conclusion, a continuous collaboration between technology and legal frameworks that protect the privacy and rights of patients, while encouraging innovation in the field of health, is necessary.

Once these bureaucratic and legal thresholds are crossed, the possibilities are endless, this is presented in the paper {1.3}. In this paper, IoT is combined with 5G technology to emphasise their advantages in different applications with application in the medical field and in that of smart cities. It enchants the difference between 5G and older versions of this type of communication through a five-layer architecture for IoT systems in 5G environments, consisting of the IoT sensor layer, the network layer, the communication layer, the architecture layer and the application layer . At the same time, the issue of security is also discussed, which is sensitive due to the wide connectivity offered by 5G technology. The authors claim that one of the improvements is that in the future it is necessary to decrease energy consumption. The main application areas in which 5g and Iot can be integrated and bring improvements, which are discussed in this paper, are smart cities, healthcare, automated industries and transport systems.

The study claims that joining IoT technology with 5G networks will usher in a new era in connectivity, characterised by higher speeds, increased efficiency and extended coverage. At the same time, this integration requires a rigorous analysis of security, privacy and regulatory issues.

In {1.5}, researchers integrate IoT devices with modern fitness to enhance life and reduce the need for medical treatment. As a consequence of my study, I've discovered that an increasing number of IoT-based solutions, particularly fitness trackers, activity analysis tools, and fitness applications, boost fitness by revealing how to maximise the effectiveness of an exercise. One thing that is continually applied in today's activities is the use of artificial intelligence (AI) in data analysis and prediction. The authors of this article employed artificial intelligence to improve training regimens, forecast injuries, and personalise user experiences based on individual health data.In this article, the main problem of fitness is emphasised, which is that most of the practices are not professionals and most of the time even the professionals do not know the details about the benefits of each movement performed in different exercises. A new concept is introduced in this paper "Social IoT" which is communication between IoT devices to improve data sharing, which can lead to more integrated and community-focused fitness solutions.

At the same time, the writers of article {1.2} discuss the importance of IoT devices for fitness and health monitoring in enhancing human health, similar to prior papers on the subject. They propose that by combining a network of such sensors with an architecture used in environmental monitoring, patient health data may be better integrated and understood. This article discusses a 3-layer Sensors, Integration and Association, Application and Diagnostic design, which is comparable to the well-known architecture used in forest fire monitoring.

The 3 layers are the “Sensors” layer that describes advanced equipment and sensors used to obtain information, the second layer, “Integration and Association”, proposes an ontology and a potential method for integrating sensor data from the device to the cloud. Finally, the third layer presents an implementation of intelligent learning methods for disease detection and labelling.

3.2. Fog Computing

It is very important to study fog computing because technology is constantly advancing and this means that the devices for retrieving data are becoming more and more efficient and accurate and the methods of sending data are becoming faster. Therefore, the article {2.1} discusses the importance of fog computing to cover the limitations of cloud computing when it comes to very large volumes of data that come in real time. In this paper, one layer is defined for each part of the system, the first layer is the one that contains the physical devices that contain the registers that record data that must be processed. The next layer is the one that contains fog nodes, they are made up of edge devices that are capable of providing computing and storage resources. Follows the fog server layer which is a mediator between data processing and sending important results to the cloud, at the same time this layer handles the aggregation of data from multiple fog nodes and further processes it before sending it to the cloud or back to the nodes for real-time decision -making. The last is the Cloud layer which is used for data storage and analysis, at the same time it deals with non-urgent tasks such as those handled by Fog Nodes. In {2.2}, we also study the benefits that fog computing brings to the modern world, benefits that we have presented throughout the work in various chapters and subchapters. What's new in this work is the fact that the authors highlighted how important a mechanic is to synchronise the elements present in a fog environment. In the work it is emphasised how at the moment there are enough problems that need to be fixed for fog computing to surpass the methods currently used, a major problem is the fact that there are not enough adaptive scheduling solutions that can respond in real-time to change the user's requirements and the conditions of the applications and the internet. Another problem identified by them is scalability, but this problem is unavoidable because no application that uses such a thing can anticipate the number of IoT devices appearing in the system and their data transmission power. In conclusion, the text highlights the importance of having well-coherent programming solutions that are problem-free, flexible and scalable, that can effectively manage the complex and ever-changing interactions within a fog computing network to maintain performance optimum and the fulfilment of its purpose.

3.3. Fog Computing in Healthcare Solutions

In {s40747-021-00582-9} is presented how the rapid growth of science and technology has always been shaped by advancements in medical and healthcare applications. It's evident that cloud technology is crucial in the healthcare system. Traditional cloud computing consists of two layers: the cloud layer, which provides substantial storage and computing power, and the end-user layer, which includes all the connected devices. However, cloud computing has some limitations, such as high service response times, making it less suitable for real-time IoT applications. To solve this, fog computing has been introduced. It brings cloud capabilities closer to the users, reducing latency and improving real-time responses. Also in {5337733}, the authors use fog computing as a new approach that brings cloud computing abilities closer to end users. Cloud computing provides various IoT services, and when used in fog computing, it can support sensors and devices that monitor health parameters like body temperature, ECG, and blood pressure. This setup significantly reduces latency and improves the speed of service delivery, which is very beneficial in healthcare.The objective of this project is to develop a cloud-based healthcare monitoring system that enables real-time notifications and reduces latency. The proposed system comprises three layers. Sensor network layer - this layer collects data via embedded sensors. Fog layer: This layer processes data locally, provides real-time notifications and ensures security. Cloud layer: This layer stores and analyses long-term data for future use. This architecture aims to improve the efficiency and responsiveness of healthcare monitoring. Papper {j teler} also underlines how fog computing systems are very efficient and the most important aspect that is highlighted is that it reduces latency, but further studies and improvements are needed to address security issues.

However, in addition to the benefits brought by this type of computing, there are also challenges that can be encountered when developing such a system {s40747-021-00582-9 }{j teler}: Efficient compute offloading schemes are needed to optimise performance and to allocate tasks between fog and cloud computing systems. Scheduling algorithms to achieve energy efficiency and reliable management of fog servers in heterogeneous networks. Providing security on fog servers to protect sensitive healthcare data from potential threats. The integration of fog computing in IoHT applications is still in its early stages, with many challenges to be addressed in future research. Security and privacy: Fog nodes are susceptible to various attacks due to their distributed nature. Data management: Efficient data management and storage are essential for successful fog computing. Operational issues: Ensuring reliability and efficiency in a heterogeneous and geo-distributed environment. Service issues: Balancing service quality and resource allocation.

Chapter 4

4. Detailed Design and Implementation

This chapter outlines the development process of the application discussed in this study. It focuses on several key aspects: the system architecture, implementation, and design patterns. Additionally, it includes important code snippets that illustrate specific solutions. The decisions made during the development were informed by the analysis and theoretical foundations presented in Chapter 4, as well as by the knowledge gained throughout the university coursework. The next paragraphs will cover all of the important components and offer thorough explanations for each. Additionally, ideas, approaches, and resources were discussed and planned. The explanation will be written in such a way that anyone with a basic grasp of the subject may reproduce and comprehend how this approach works. The intended design is represented in the figure below.

4.1. System Architecture

Component Details

The gym monitoring system comprises various hardware and software components that interact seamlessly to collect, process, and display data. The key components include Arduino for sensor data collection, Raspberry Pi as the central processing unit, a server for data storage and API handling, and an Android application for user interaction.

Arduino

- **Role**: The Arduino is responsible for capturing data from the sensors and transmitting it via Wi-Fi to the Raspberry Pi.
- **Components**: Equipped with an ESP8266 Wi-Fi module to enable wireless communication.

- **Functionality**: Collects sensor data, processes initial readings, and sends the data to the Raspberry Pi for further processing.

Communication Modules

- **Wi-Fi Module (ESP8266)**: Enables the Arduino to transmit data wirelessly to the Raspberry Pi, facilitating real-time data transfer.

Raspberry Pi

- **Role**: Serves as the central processing and communication hub of the system.
- **Components**: Raspberry Pi 2 Model B with an external Wi-Fi module.
- **Functionality**: Receives data from the Arduino, processes it locally, and communicates with the server for data storage and retrieval. Hosts an MQTT broker for real-time data communication with the Android application.

Server

- **Role**: Hosts the database and provides RESTful APIs for data storage and retrieval.
- **Components**: SQL Server for database management.
- **Functionality**: Stores user profiles, workout history, and gym machine details. Handles API requests from the Android application to fetch and update data.

Android Application

- **Role**: Provides a user-friendly interface for interacting with the system.
- **Pages**:
    - **Welcome Page**: Greets the user and provides navigation options.
    - **Profile Page**: Allows users to view and edit their profiles.
    - **Workouts Page**: Displays the user's workout history.
    - **Machine Details Page**: Provides detailed information about gym machines.
- **Functionality**: Communicates with the server to fetch user data and with the Raspberry Pi for real-time updates via MQTT.

MQTT Broker

- **Role**: Manages real-time data communication between the Raspberry Pi and the Android application.
- **Functionality**: Ensures that data sent from the Android application is promptly received by the Raspberry Pi for immediate processing.

Detailed Diagram

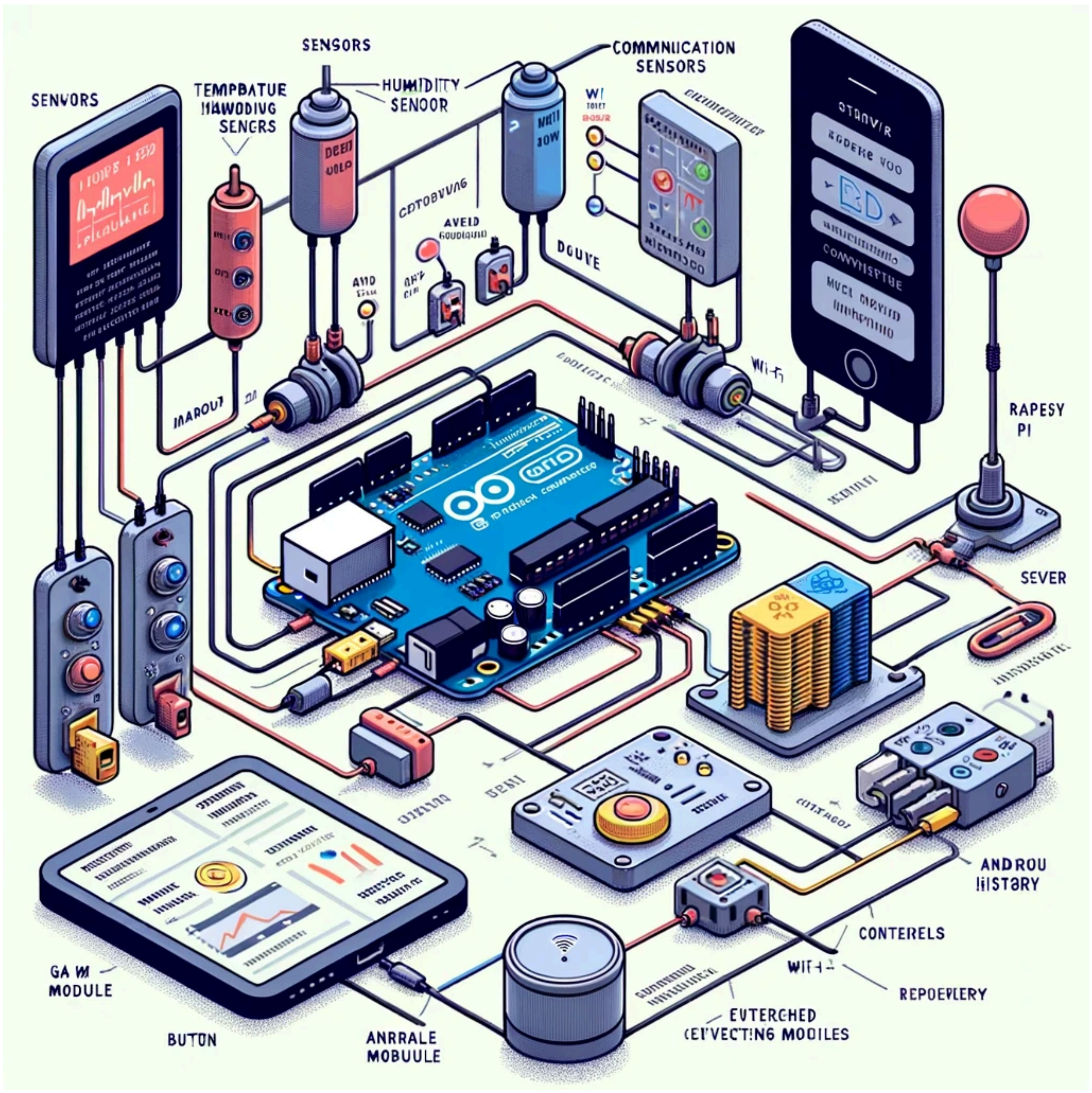


Figure 7 - Detailed Diagram

4.2. Hardware Configuration

Arduino

This component of the system has the role of capturing data from the various sensors present on the development board. Aest is a key component in our project, acting as the primary data capture agent for the environmental and proximity sensors installed in the room. Its role extends beyond simple data collection; this setup allows the Arduino to efficiently manage incoming data streams and ensure timely communication with the Raspberry Pi for further processing and analysis.

In the following, we will present each hardware component used to obtain a configuration at home like the one used in the practical part of this work.

Block Diagram

We present the block diagram outlining the architecture of our gym monitoring system. This diagram visually represents the connections and interactions between the various hardware components. It provides a clear overview of how data flows through the system, from initial data collection to final output and control mechanisms.

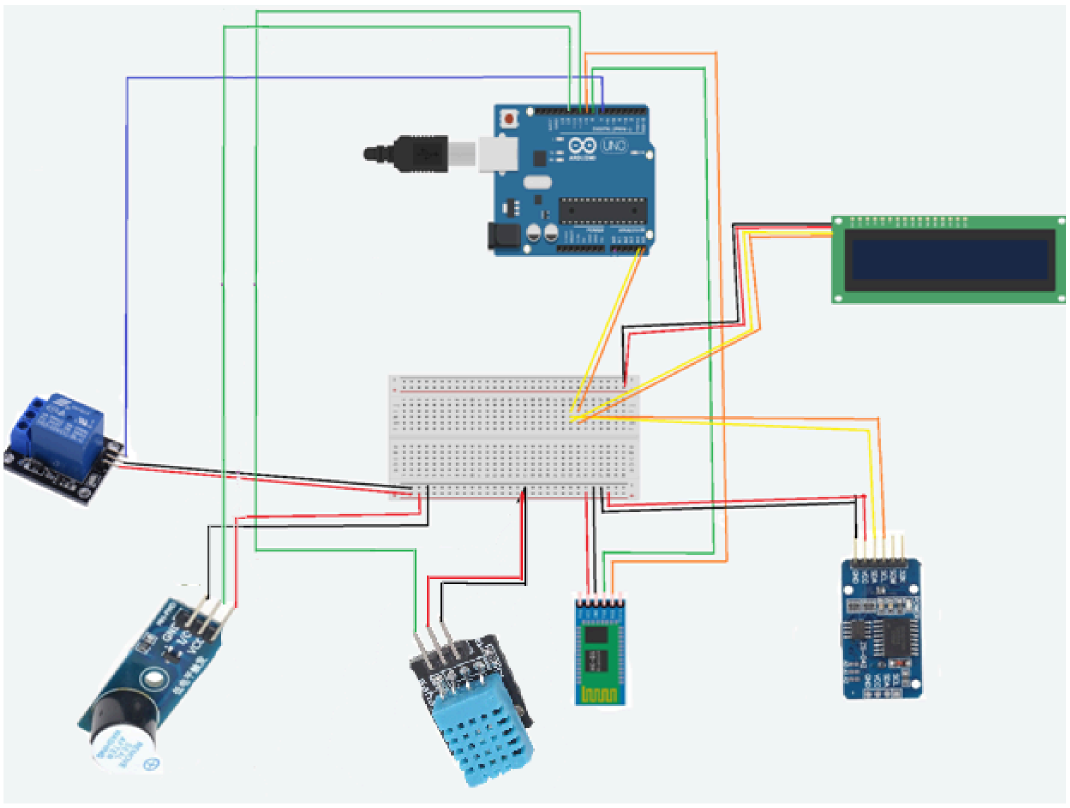


Figure 15 - Block Diagram of Fog Cell

Board

The WeMos D1 R2 WiFi Development Board is designed for various IoT applications. It uses an ESP8266 microcontroller that integrates Wi-Fi connectivity, allowing the board to connect to the Internet without the need for external Wi-Fi modules. The microprocessor is capable of running either at 80 MHz or at a higher frequency of 160 MHz when more processing power is required. The board structure is reminiscent of traditional Arduino designs such as the Uno and Leonardo, making it compatible with many Arduino shields and peripherals.It comes with 4 Mb of flash memory, providing space for storing code and essential data. Built-in 802.11 Wi-Fi compatibility allows the board to connect directly to the internet and communicate with web servers or other networked devices, facilitating remote monitoring and control applications.

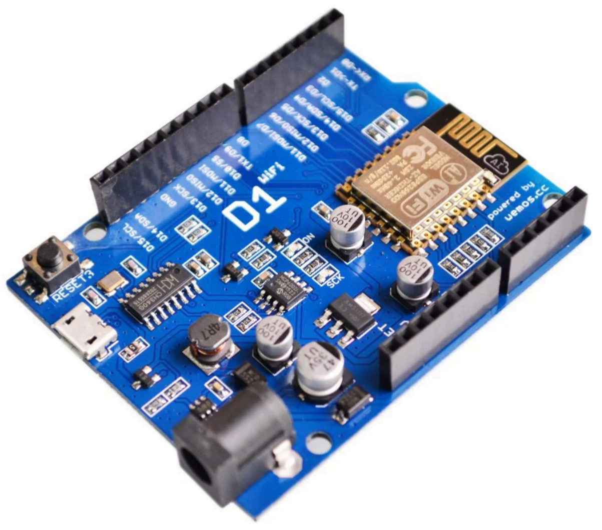


Figure 8 - WeMos D1 R2 WiFi ESP8266 Development Board

Sensors and Modules

In this section, we detail the key components that make up our gym monitoring system. Each module plays a crucial role in ensuring that the system runs smoothly and efficiently. These components include various sensors and devices that enable wireless communication, environmental monitoring, and user interaction.

Below is a table summarising each module, its functionality and an illustrative image for a clearer understanding.

Name	Functionality	Image
Bluetooth HC-05 Module	Enables wireless communication between the Arduino and other Bluetooth-enabled devices.	*Figure 9*
Relay 1 Channel 5V Module	Acts as a digital switch to control high power loads, useful for automating gym equipment.	*Figure 10*
RTC DS3231 AT24C32	Maintains accurate time and date for data logging, essential for timestamping recorded data.	*Figure 11*

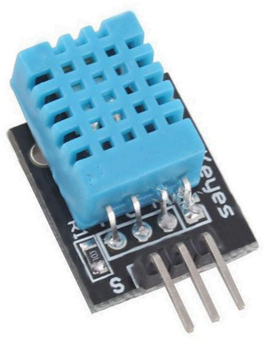
LCD 2004 with Blue I2C	Displays system statuses, sensor values, and warnings, enhancing the system's user interface.	*Figure 12*
Gas Detector Sensor MQ-135	Detects various gases to monitor air quality in the gym, ensuring a healthy environment.	*Figure 13*
DHT11 Temperature and Humidity Sensor	Measures ambient temperature and humidity, crucial for environmental control within the gym.	 *Figure 14*

Table 1 - Table of Sensors and Modules

Raspberry Pi

The Raspberry Pi 2 Model B serves as the central processing and communication hub in our gym monitoring system. It processes data received from Arduino and other Fog Components, performs local data analysis, and communicates with the cloud server and end-user devices. This section details the setup and configuration of the Raspberry Pi 2, highlighting its capabilities and its integration into our system.

It has a Quad-core CPU running at 900 MHz, with a 1GB of RAM and a MicroSD slot for loading the operating system and storing data. Best solution for our needs is using a Raspbian OS as the operating system, which is the official operating system developed specifically for Raspberry Pi devices, known for its lightweight and flexibility, making it suitable for educational, experimental, and industrial applications.

Since the Raspberry Pi 2 Model B does not come with an integrated Wi-Fi module, we have opted to use the SURECOM EP-9001-g, an external module for WLAN connection. This choice allows for reliable and robust wireless connectivity, essential for seamless data communication within our system. In the below Figure is presented the setup used in this paper.

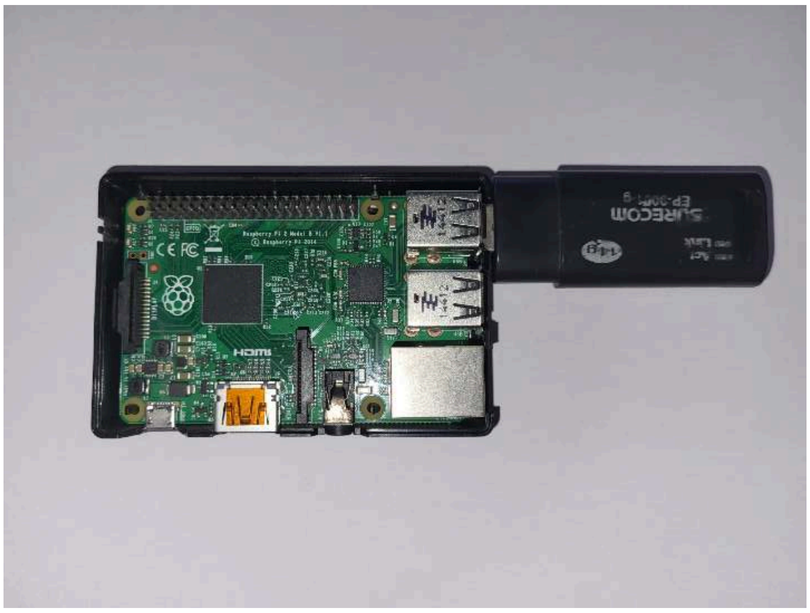


Figure 16 - Raspberry PI 2 model B, and WLAN module

Set Up

It starts by installing an operating system on your Raspberry Pi, start by downloading your preferred operating system from the official Raspberry Pi website or from another vendor if you're using an alternative like Ubuntu. Once the download is complete, use an imaging tool such as BalenaEtcher or Raspberry Pi Imager to write the operating system image to a MicroSD card. This process must be performed on a PC equipped with an SD card reader and ensure that the operating system is correctly configured on the card before inserting it into the Raspberry Pi.

The next step is where the Wi-Fi module needs to be configured to allow communication with the server and other devices. Attach the SURECOM EP-9001-g module to one of the USB ports on the Raspberry Pi. Depending on the module, you may need to install specific drivers or software to enable its functionality. This can be done through command-line tools in the terminal, where you will update your system and install any necessary packages. The final step in configuring the Wi-Fi module involves editing the network configuration files to include the Wi-Fi network details. Modify the *wpa_supplicant.conf* file to ensure that the Raspberry Pi can automatically connect to the network on startup.

These steps prepare the Raspberry Pi to serve as the central processing unit for our system, handling tasks from data processing to communication, crucial to the smooth integration and functionality of our gym monitoring system. Regular updates and maintenance of the system are recommended to ensure it works efficiently and reliably.

4.3. Software Configuration

Arduino

The Arduino IDE is used for programming the Arduino. This environment allows you to write code (sketches), upload it to the Arduino board, and monitor the serial output. In order to have an easy life working with sensors and modules, we need to add libraries to the sketch. In order to do this, we need to: Go to Library Manager from the right side menu and search for each library and then press INSTALL. Here are the following libraries that we need:

- **ESP8266WiFi**: Connects Arduino to Wi-Fi networks.
- **DHT**: Reads data from the DHT11 temperature and humidity sensor.
- **RTClib**: Interfaces with the DS3231 Real-Time Clock module.
- **LiquidCrystal_I2C**: Controls the LCD display.
- **SoftwareSerial**: Enables Bluetooth communication.
- **PubSubClient**: Enables MQTT communcation.

Next we need to initialise our serial and bluetooth connections in order to help with the debug and with data transmission, wi-fi connection in order to connect to a specific wi-fi, need to set up the MQTT server for communication and lastly we need to initialise the following sensors and modules:

- **DHT Sensor**: for temperature and humidity readings.
- **RTC Module**: to set the current time if the module has lost power.
- **LCD Display**: to display and turn on the backlight.

Following we are going to present the main loop, because arduino boards are working over the loop function that starts running immediately after the setup function. Here is what is going on inside this loop function:

- **MQTT Connection Maintenance**: Checks if the MQTT connection is active and attempts to reconnect if it is lost.
- **Sensor Data Reading**: Reads temperature and humidity from the DHT sensor and the state of a button connected to digital pin 3.
- **Error Handling**: Checks for errors in reading sensor data and prints an error message if necessary.
- **Data Display**: Displays the temperature and humidity on the LCD screen.
- **Data Publishing**: Constructs a payload string with the sensor data and button state, and publishes it to the MQTT topic "gym/sensors".
- **Serial Debugging**: Prints the sensor data and button state to the Serial Monitor for debugging.
- **Loop Delay**: Delays the loop for 2 seconds before repeating the process.

We also need to use a function where we want to reconnect the MQTT client, in case the connection is broken, where we attempt to reconnect to the MQTT broker if the connection is lost, with a retry interval of 5 seconds. Prints the connection status and publishes a connection announcement upon successful connection.

Raspberry Pi

After the wifi module and the operating system are working well on our Pi board we need to install mosquitto broker and Node-RED. Mosquitto is an MQTT (Message Queuing Telemetry Transport) broker that facilitates communication between IoT devices. Here are the steps to install it on our board:

1. Update your package list: **sudo apt-get update**

2. Install Mosquitto and the client: **sudo apt-get install mosquitto mosquitto-clients**
3. Start the Mosquitto service: **sudo systemctl enable mosquitto**
4. Enable Mosquitto to start on boot: **sudo systemctl enable mosquitto**

Now, we need to instal Node-RED, as following:

1. Install Node-RED using the Node.js package manager: **sudo apt install -y npm sudo npm install -g --unsafe-perm node-red**
2. Start Node-RED: **node-red-start**
3. Enable Node-RED to start on boot: **sudo systemctl enable nodered.service**

To integrate MQTT with Node-RED on the Raspberry Pi, we started by opening Node-RED through a web browser by navigating to http://<your_pi_ip>:1880. Within the Node-RED interface, a variety of nodes are available on the left side, including input and output nodes required for MQTT operations. We added the necessary nodes by dragging and dropping the 'mqtt in' node, which is responsible for subscribing to MQTT topics, and the 'mqtt out' node, which is responsible for publishing messages to MQTT topics, into the workspace. These nodes are crucial for receiving sensor data and sending processed data, respectively. For the 'mqtt in' node, we configured it by double-clicking the node to open its settings. Here, we set the server to localhost because the Mosquitto broker is running on the same Raspberry Pi. We specified the topic at ‘gym/sensors’ to subscribe to sensor data published from the Arduino. After configuring these settings, we saved the configuration by clicking "Done". Similarly, we configured the 'mqtt out' node by setting the server to localhost and the topic to gym/processed, which will be used to publish the processed sensor data. This configuration ensures that processed data can be sent back via the MQTT protocol. We saved the configuration by clicking "Done".

Finally, we deployed the entire flow by clicking the "Deploy" button located in the upper right corner of the Node-RED interface. Deploying the stream enabled the configuration, allowing the system to start receiving messages published to the ‘gym/sensors’ topic and displaying them in the debug panel. This configuration is essential to ensure that sensor data is received, processed correctly and can be monitored in real time.

Android

To create the Android app for our gym monitoring system, we used Android Studio and chose Java as our programming language. The app consists of several pages: a welcome page, a profile page, a page to see previous workouts and a page to see details about the gym equipment. In addition, the application continuously sends data to the Raspberry Pi while it is open and retrieves profile, exercise machine and training data from a server.

First, I set up a new Android project in Android Studio, naming it "GymMonitorApp". I structured the project by creating separate activities for each main function of the application. These activities include the WelcomeActivity for the welcome page, the ProfileActivity for the profile page, the WorkoutsActivity for viewing previous workouts, and the MachineDetailsActivity for viewing details about the exercise machines. In the WelcomeActivity, we designed a layout that includes a welcome message and a button to navigate to the profile page. This gives users an initial greeting and easy access to their profile. For the profile page in ProfileActivity, we created a layout that allows users to view and edit profile information such as name, age, and weight. This page is essential for personalising the app experience and tracking user-specific data. The WorkoutsActivity page displays a list of previous workouts, allowing users to review their exercise history. This feature is crucial for users to track their progress over time. MachineDetailsActivity provides detailed information about various exercise machines, including their names, descriptions, and instructions for use. This helps users understand how to use the equipment effectively and safely. To handle MQTT communication for real-time data exchange with the Raspberry Pi, we implemented an MQTTClient class. This class manages the connection to the MQTT broker, subscribes to relevant topics, and publishes messages. In the WelcomeActivity, we initialised the MQTT client and established a connection to the broker running on the Raspberry Pi. This configuration allows the application to send and receive data as long as it is open. For each activity, we designed the corresponding XML layout files. The activity_welcome.xml layout includes a text view for the welcome message and a button to navigate to the profile page. The activity_profile.xml layout contains editable text fields for the user to enter profile information. The Layouts for WorkoutsActivity and MachineDetailsActivity are designed to display lists and detailed views of workouts and machines, respectively.

By combining these components, we developed a comprehensive Android application that interacts with our gym monitoring system. The app not only provides an easy-to-use interface for managing personal fitness data, but also facilitates real-time communication with the Raspberry Pi, ensuring that all relevant information is synchronised and up-to-date. This integration is essential to provide a seamless and efficient user experience.

Server

To develop the server for our gym monitoring system, we used C# with ASP.NET Core and SQL Server. The server is designed to handle CRUD operations for user profiles, gym machines, and workout data, providing a RESTful API that interfaces with the Android application and the Raspberry Pi. We began by creating a new ASP.NET Core Web API project in Visual Studio , naming it "GymMonitorAPI". Our data model includes three primary entities: UserProfile, GymMachine, and Workout. Each entity is represented by a C# class with properties corresponding to the database columns. In the Startup.cs file, we configured the services and middleware required for our application. This included setting up the database context, enabling controllers, and configuring Swagger for API documentation. We created controllers for each entity (UserProfileController, GymMachineController, and WorkoutController) to handle the API endpoints. These controllers provide methods for creating, reading, updating, and deleting records in the database. Each method interacts with the GymMonitorContext to perform the necessary database operations.

Finally, we ran Entity Framework migrations to create the initial database schema and update the database. This process involved using the Package Manager Console to generate and apply migrations, ensuring that the database structure matched our data model.

Chapter 5

5. Conclusions

In this chapter, we will outline the conclusions drawn upon completing this project, highlighting the insights gained across the various domains we explored and utilised. Given that this project encompassed both the client-side and server-side aspects, we effectively navigated the process of creating a product suitable for presentation in the global IT market. Throughout the development of this application, we encountered numerous challenges such as misunderstandings, bugs, and incompatibilities. However, with the guidance of our supervising teacher, we devised solutions to either circumvent or resolve these issues. Despite being a demanding and arduous journey, it culminated in a final product that we take pride in and through which we gained invaluable knowledge. To provide a clearer understanding of the outcomes of this project, we will present both technical and practical details in the subsequent section of this chapter. Additionally, in the final section, we will discuss potential improvements and the features we aim to implement in the future.

Considering that this project was a complete application that contains both the client side and the server side, we can say that we went through the process of creating a product that can be presented on the IT market in the world. During the development of this application we encountered all sorts of misunderstandings, bugs, incompatibilities and so on, but together with the coordinating teacher, we found ways to avoid this problem or even solve them. It was a difficult and difficult process, but one that led to a final product that we are proud of and through which we acquired very valuable knowledge.

5.1. Obtained Results

The implementation of the wellness solution based on IoT and Fog Computing gave some significant results:

1. Monitoring physical exercises inside the gym in real time: the system has successfully monitored different parameters and provides immediate feedback to users. This real-time capability has been

achieved through efficient data collection, processing and analysis facilitated by fog nodes.

2. Real-time monitoring of the sensors placed on the equipment inside the gym: the system has successfully monitored different seasons and provides relevant information about the sports equipment.
3. Data privacy and security: By locally processing sensitive information using fog computing principles, the system has improved data privacy and security. Robust encryption and authentication mechanisms have been implemented to protect user data from potential threats.
4. Scalability and flexibility: The system architecture has been designed to be highly scalable and flexible, allowing easy integration of new sensors, devices and functionality. This adaptability is essential to accommodate future advances in IoT and fog computing technologies.
5. User-friendly interface: The Android app has provided a user-friendly interface that allows seamless interaction with the system. The app featured intuitive dashboards and mobile interfaces, effectively displaying health metrics and personalised recommendations.

Overall, the project demonstrated the potential of IoT and Fog Computing technologies to improve fitness and wellness applications. Future improvements could focus on incorporating advanced machine learning techniques, such as federated learning, to further improve data privacy and model accuracy on distributed devices. These improvements are presented in the following.

5.2. Future Improvements

Federated Learning

Federated learning is a machine learning technique where the model is trained on multiple distributed devices like smart devices and serves that have local data, without needed to share the local data, as shortly presented from {4.1}.This strategy is especially effective for maintaining privacy and complying with data security rules since it allows diverse individuals to contribute to the development of a model without actually sharing the data.Taking a general look at the workflow required to have such a system to work, we must first initialise a machine learning model on a central server that uses local data. The next step is to transmit this model (small changes can be made to the preprocessing of the data to satisfy differences between the systems on which the training is done) to devices such as smartphones, smart devices, servers and so on. After training, these new models are forwarded without sensitive data being forwarded. The central server aggregates these updates to improve the model, the aggregation strategies differ from one model to another, this improved model is sent again to the devices and the process is repeated until the desired results and predictions are reached. Federated learning is especially prominent in applications where data privacy is critical, such as healthcare, banking, and mobile device usage, allowing for collaborative model training without jeopardising data privacy as also presented in {4.2}. The key idea of {4.3} report is that by pooling and networking data according to a criterion of geographic models and training them separately for each community, learning each community, learning it federally (FL) can create a significant efficiency and accuracy in scenarios where the data distributions are not independently identically -distributed. This approach ensures that the learning process is more adapted to the specific characteristics of each community, which leads to better performance.

In conclusion, federated learning offers a privacy-preserving solution for collaborative machine learning tasks across decentralised data sources. By enabling model training without sharing raw data, federated learning has the potential to revolutionise various domains, including healthcare, by ensuring data privacy, scalability, and efficiency in model training.

Security and Energy Consumption

As the integration of IoT and fog computing continues to evolve, ensuring security measures remain a very important aspect for future improvements. There is currently a need to add more advanced encryption techniques, and future systems should use end-to-end encryption to protect data as it moves from IoT devices to fog nodes and eventually then, to the cloud servers. Device fingerprinting, biometric verification, and multi-factor authentication (MFA) are some of the methods that can be used to strengthen authentication protocols. IoT devices and fog nodes can consume large amounts of energy, drastically affecting both operational costs and environmental footprint. Energy efficiency is another critical aspect that requires attention and continuous improvement. Using low-power hardware specifically designed for IoT applications can help save energy. Selecting energy-efficient sensors, processors, and communication modules will help extend the battery life of IoT devices and reduce the need for frequent recharging or replacement. Developing and implementing energy-efficient algorithms for data processing and transmission can significantly reduce the energy consumption of IoT devices and fog nodes. These algorithms should improve resource usage without compromising performance.

Bibliography