INTRODUCTION

The primarily goal of the paper "IoT for Smart Engine Health Monitoring System" is to allow individuals to safely and effectively monitor their production hardware and assure a consistent data management for the most important types of engine measurements. Furthermore, the data can be used in the fields of predictive maintenance, resulting in increasing the engine lifespan, boosting overall productivity and reducing maintenance costs.

In today's era, the sweeping shift from traditional industrial practices to an economy based on digitization and self-working machine technologies pushed the society to drastically evolve and adapt to a modern way of living. This transformation impacted every sector and changed it's core, propelling improvements in communication, infrastructure and manufacturing. Nowadays, industries use IoT (Internet of Things) technologies, robots, servers and AI (Artificial Intelligence) algorithms to improve the efficiency and simplify production, thus reducing the costs of manufacturing products and increasing the enterprise sales. The digital revolution reshaped the industrial landscape and created a new link between humans and computers that will allow future technology development and innovation in all domains.

Heavy industry is a type of industry that involves working with large and heavy equipment tools and machines in order to complete complex processes. Because of this factors, heavy industries branches like automotive, manufacturing, agriculture and mining implemented IoT systems that are meant to engage machines to work in specific environments, manipulating them with software tools and digital protocols.

Following the evolution of the industry sector in Romania, this paper aims to present the development of an IoT system for monitoring engines health during production tasks and to expose the advantages of predictive and preventive maintenance. In order to fully cover the functionality, the project will focus on two interconnected parts: the hardware part which is composed of an engine, sensors and devices that process and transmit measurement data and the software part, a mobile application that enables users to monitor data remotely.

The scope of the project is to allow users to easily visualize the state of their engine in the application, having the advantage to receive alert notifications in real time in order to take adequate actions. This implementation will reduce total maintenance and predict failure, thereby reducing the costs and the chance of critical problems in the production line.

CHAPTER 1 - ANALYZING INTERNET OF THINGS SYSTEMS

1.1 History and evolution of Internet of Things systems

An Internet of Things also known as IoT is a sophisticated framework defined by a network in which multiple devices like sensors, micro-controllers, processors, routers and other gadgets are linked together and exchange information throughout the session using secure protocols such as HTTP, HTTPS, MQTT, etc. The data is collected, processed and then sent to one or multiple end points for visualization or further processing, eliminating the need of human interaction with the system until the output is provided. This operation is instrumented by the software which dictate how the IoT should work, establishing multiple ways and paths for data to be transfered, stored and displayed by employing complex algorithms.

The origin of the Internet of Things phenomenon dates back to 1960 and is considered to be the telegraph because of its ability to deliver information over large distances throughout signals. The name of "Internet of Things" was defined during a scientific presentation in 1999 by an innovator and sensor expert Kevin Ashton: "The Internet of Things has the potential to change the world, just as the Internet did. Maybe even more so" [1]. Ashton, co-founder of the Auto-ID Center at MIT, began researching how humans and machines could better connect in order to solve tasks faster and safer and how the devices can be linked together using microchips and wireless signals. Mamdakam describes IoT as "an open and comprehensive network of intelligent objects which are capable of self-organizing and data sharing and those intelligent objects which have ability of reacting in any changes happening in our environment" [2].

Another huge breakthrough happened when Walmart, one of the gigantic retailers is the United Stated decided to implement RFID (radio-frequency identification) in 2003, an advanced technology at that time that was meant to identify certain objects wirelessly, without any physical contact with the items in question. This technology enabled large amount of data to be stored remotely on tags that could be accessed from relatively long distances. The RFID enabled Walmart to track and monitor large quantities of products as well as cutting waiting times because employees were not supposed to sort and scan the products manually anymore, creating a smart and efficient inventory system that would revolutionize the traditional supply chain approaches. As from the IoT point of view, the RFID simply consists of a memory tag that is located onto an antenna, better known as a transmitter. The RFID reader (interrogator) sends radio waves to the tag, which responds with the information stored inside the chip that can be visualized or send further for processing [3] [28].

After this period, as a consequence of massive improvements in digitization and software algorithms as well as data processing enhancements, more advanced sensors were able to measure temperature, electrical tension, touch, moisture, sound, light, vibration or other measurable condition fast and efficiently as well as transmit these information in the network to other devices. Concurrently, the first smartphones were available on the market, allowing individuals to interconnect and create a huge web structure of unrelated devices. A good example of an IoT system was the Ring doorbell developed by Jamie Siminoff in 2011 which allowed homeowners to link the device to the personal smartphone in order to visualize and speak with whoever presses the doorbell button.

Table 1. Short example of IoT history

Innovation	**Creator**	**Year**
IoT Smart Vending Machine	Carnegie Melon University	1982
IoT Smart Toaster	John Romkey	1990
IoT is firstly defined	Kevin Ashton	1999
Walmart RFID	Walmart Company	2003
Arduino Hardware	Interaction Design Institute Ivrea	2003
Intel IoT Solution Groups	Intel	2011
IoT security	Multiple researchers	2016

Source: Personal creation

As a result of the accelerated development of the industry and the way of living, we are surrounded by IoT smart systems, each having it's own different implementation and purpose, aiming to create a fast and productive environment. Those systems can be found everywhere and are implemented in almost every industry in the world from agriculture, manufacture, retail or other industry fields up to the health sector, security, automobiles and even smart cities among many others. There is no doubt that all these smart devices have positively influenced the society and determined the direction and pace of how humans live their day to day lives. In 2019, Microsoft stated that more than 90% of companies will use IoT related devices by the end of 2021. On the other hand, according to Statista, the market of Internet of Things reached 100 billion dollars in market revenues in 2017, expecting to grow up to 1.6 trillion dollars by 2025 [4].

IoT is ideal for all kinds of enterprises because it enables better supply chain management with the help of GPS sensors and wireless tags, smart inventory management systems, automated inventory planning and consumer checkout, value monitoring and alerting as well as other

important attributes. Those elements combined with proper implementation and execution can create an automated environment that needs no human intervention at any task or moment.

1.2 Description of the architecture of IoT systems

In order to create a well optimized IoT system, enterprises should firstly cosider the importance of each individual piece of the puzzle and compress components together in an ordered manner. If those components are interconnected successfully, they can measure informations about the working environment precisely and eventually indicate anomalies, malfunctions and errors.

Firstly, one of the most important components of IoT devices are called sensors, little gadgets which are integrated into machines or in their immediate proximity to gather raw data that will eventually register the performance of the engine. These sensors can be either wired or wireless, depending on the specific task and location and are used to read values regarding temperature, pressure, vibration, electrical tension and current or other important indicators of the engine. In general, sensors can detect changes in the measurements and can be set to deliver alerts when the value exceeds or is below a certain maximum threshold. Even though sensors are important in predictive maintenance and monitoring, it’s challenging for companies to store and process a huge quantity of information from all of sensors simultaneously and secure the information as sensors are nowadays connected to internet, thus being vulnerable to cyber-attacks.

Secondly, centralized cloud data storage is the best option to securely save the information provided by the sensors for future processing. The traditional way of storing data in static database systems become inefficient because insights should be stored, processed and analyzed faster as the amount of data is growing exponentially. The total volume of data that is being generated by sensors, devices and other applications that continuously send massive loads of structured and unstructured data is called Big Data. This term was detailed in McKinsey Global Institute research paper as "The amount of data in our world has been exploding, and analyzing large data sets—so-called big data—will become a key basis of competition, underpinning new waves of productivity growth, innovation, and consumer surplus" [5].

Could data storage suits IoT systems nowadays because it provides scalability, ease of access and it’s cost-effective as companies are not conditioned to invest anymore is physical storage alternatives. There are a lot of cloud data storage options available, such as Amazon Web Services (Amazon S3 or Glacier), Microsoft Azure (Blob, Files), Google Could (Store and Filestore) and many others like IBM Could, Oracle Cloud, etc. Real-time database like InfluxDB time-series database and Google Firebase are also worth mentioning as they focus on efficient

ways of storing data on specific timestamps in real-time. Those types of databases are primarily used in sensor IoT system because the output is loaded in a straightforward manner into complex dashboards and graphics that act like a control panel.

Finally, data communication is vitally important when it comes to establishing connections between multiple parts of an Internet of Things system. This ability permits data to move continuously through the application and reach multiple endpoints and it's done using communication protocols. A communication protocol is defined as a system of mandatory rules that allow the informatics systems to transmit or receive messages using specific syntax and semantics rules.

The two best known IoT protocols are Bluetooth and Wi-Fi. Bluetooth runs at the frequency of 2.4 GHz and it is typically used for medium-range areas and low power consumption projects while Wi-Fi works at a frequency of 2.4 GHz up to 5 GHz and is chosen for it's amazing ability to send and receive data at a faster rate – Table 2.

Those two data transferring protocols are widely known by individuals as they are present in almost every mobile phone or personal computer in the world, allowing them to transmit and receive packets of data that will eventually form text messages, audio files, videos or any other mean of communication.

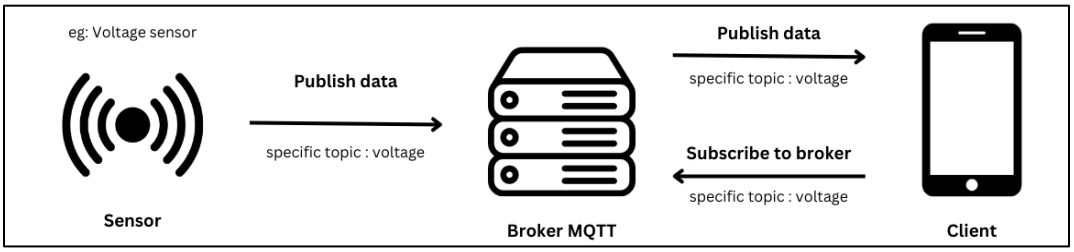
Table 2. Protocols information

Name	**Frequency used**	**Speed of transactions**	**Distance range**
Wi-Fi protocol	2.4 / 5 GHZ	60MB/S up to 600MB/S	45/15 meters
Bluetooth protocol	2.4 GHZ	150KB/S up to 1MB/S	120 metters

Source: Personal creation

Another important communication protocols are Message Queue Telemetry Transport (MQTT) and HyperText Transfer Protocol (HTTP). MQTT enables communication between nodes in both secure and unsecured networks because of its particular design, which permits it to function in extremely low bandwidth networks. MQTT has a publish/subscribe architecture, which means that certain nodes (also called brokers) publish the information, and other nodes (named clients) can read it after subscribing by connecting to the correct URL.

Picture 1. Visualization of how MQTT operates



Source: Personal creation using Canva

The World Wide Web's HTTP communication protocol was developed especially for document transfer between multiple endpoints. We are most familiar with it as one of the supporting technologies needed for web browser functionality (internet domains start with www). The HTTP client sends a request and then waits for the server's response. After obtaining the request to be sent, the HTTP client breaks the connection from the server, which then responds to the request and returns the response back to the sender.

1.3 How humans interact with machines and automated processes

Following the Industrial Revolution in England that took part between 1760 and 1840, humans began starting using machines more often for demanding tasks instead of using physical force or creating handmade objects [6]. The machine in this chapter refers to a smart system that is somehow independent in decision making and it's working autonomously to provide a desired outcome. In order to succeed, those machines may be programmed to work with or without human interaction, using machine learning, artificial intelligence, deep learning or any other complex algorithm [7].

Even though AI is evolving exponentially and is slowly becoming a normality in today's society, the majority of automated machines still depend on human supervision and interaction, thus not being completely independent. This raise another problem where human-machine interaction may be predisposed to unpredictability as humans are prone to make mistakes while machine cannot adapt to real-time situations since they work on predefined set of rules. Although this interaction between machines and humans is not perfect, it was proved that it can improve employees net performance and boost operational processes by increasing productivity as it can be seen in Amazon deposits where robots are in charge of inventory stocking and management depicted in an HBO documentary: "The Future of Work" by VICE News [8].

In the IoT context, human interaction is mostly based on manipulating graphical user interfaces (GUIs) such as mobile applications or web dashboard especially created to utilize the IoT systems. A GUI is a digital interface in which users can interact with multiple abstract digital objects like buttons, menus, animations, photos, icons and other types of layout resources. Each element should serve a predefined purpose that varies from allowing the user to navigate in the application or set their own preferences to direct interaction possibility with the rest of the system. Usually, those interfaces has a straightforward design that allow inexperienced users to connect to the devices in the network and execute various operations. Gesture and speech recognition software associated with motion and sound sensors as well as video support can improve the way people interact with machines for their every day tasks and can change human behaviors in the long run.

1.4 Internet of Things for predictive and preventing maintenance

The Internet of Things is mostly used in production environments and manufacturing industries because it permits companies to manipulate and monitor their machines and all of the equipment used in fabrication processes. Predictive maintenance can be described as a kind of advanced maintenance that can identify when a physical system is about to malfunction, allowing for monitoring and removal of unwanted machine stress before there is any noticeable deterioration of the components. This type of maintenance can be implemented using the automated structures of Internet of Things technologies [9].

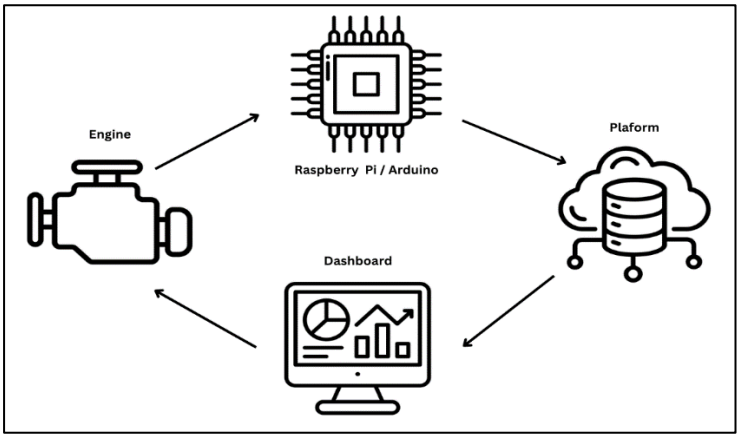
Predictive maintenance is working with data by analyzing it in order to identify certain defects and weaknesses before the machines malfunction. This type of maintenance aims to help businesses reduce production and reparation costs and as well to use the resources as good as possible by focusing on preventing equipment to malfunction or run outside the default safety parameters. Generally, for simple applications, IoT platforms use commercial or open source alternatives that are configured by default to do specific tasks like Arduino, Raspberry Pi Linux based device or ARM devices. For complex applications, software engineers can opt to install platforms like Easy-IoT, Kaa or Microsoft Azure IoT Suite that come with servers installation packages and software development kits (SDK) libraries [10].

Preventive maintenance on the other hand refers to performing consistent inspections regularly for the purpose of intercepting misfiring and anomalies. Just like predictive maintenance, it focuses on reducing overall hardware related costs and improves total production. The key difference between the two is that preventive maintenance is a task that is being scheduled by the

operators periodically while predictive maintenance is postponed until the real-time data is analyzed and it results that the machines have signs of wear or other problems.

According to the McKinsey & Company Internet of Things research, predictive maintenance can lower costs by almost 30%, decrease downtime by 45% and increase the machine lifespan by up to 20% [11]. In order to obtain this, predictive maintenance smart systems collect data from the wired or wireless sensors that are placed directly in the working environment (different types of sensors are located in various location depending of what kind of measurement is being tested) and process it using software applications and complex algorithms. The machine measurements can include working temperature and environment temperature, operating voltage and amperage, power consumption, vibration, humidity and many other factors that are usually stored into an online real-time database or online cloud storage. By doing so, sensors have the ability to send continuous feedback that is immediately transformed into valuable and crucial information for the overall state and health of the equipment. If proper actions are taken to remediate the situation, the efficiency will increase and maintenance risks and costs will be minimized.

Picture 2. Simple IoT scheme



Source: Personal creation using Canva

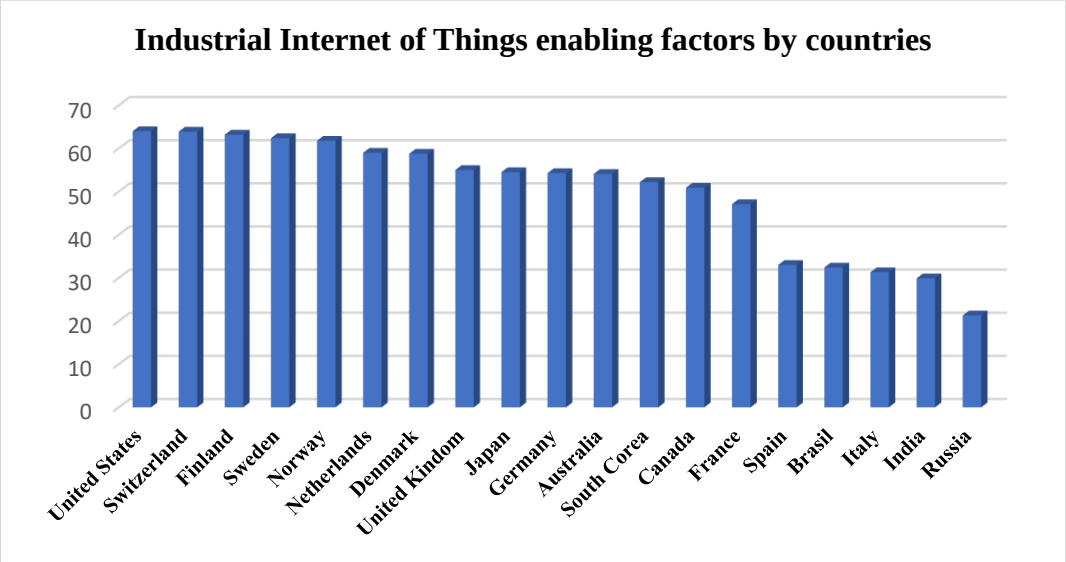
1.5 Structure of the thesis and proposal of solution

This chapter will investigate the issues related to Internet of Things in the contemporary society and will outline the most problematic aspects of incorporating programmable object interfaces (POI) into our daily activities. This purpose will be achieved by detailing the paper structure in a hierarchical manner, clearly highlighting the main goal.

1.5.1 Define an issue in the IoT market

In today's market, even though Industrial Internet solutions are the best way to increase productivity and ultimately the profit, companies face multiple problems that restrict them from using the full potential of automation through digital procedures, especially small companies that struggle to invest in new technology. According to an Accenture analytic research [12], the Internet of Things alone should boost the total cumulative GDP of the United States by over 6 trillion dollars. In the same manner, countries like Germany, United Kingdoms and China are seeking to improve their system in order to gain more profit.

Picture 3. Enabling factors of IoT

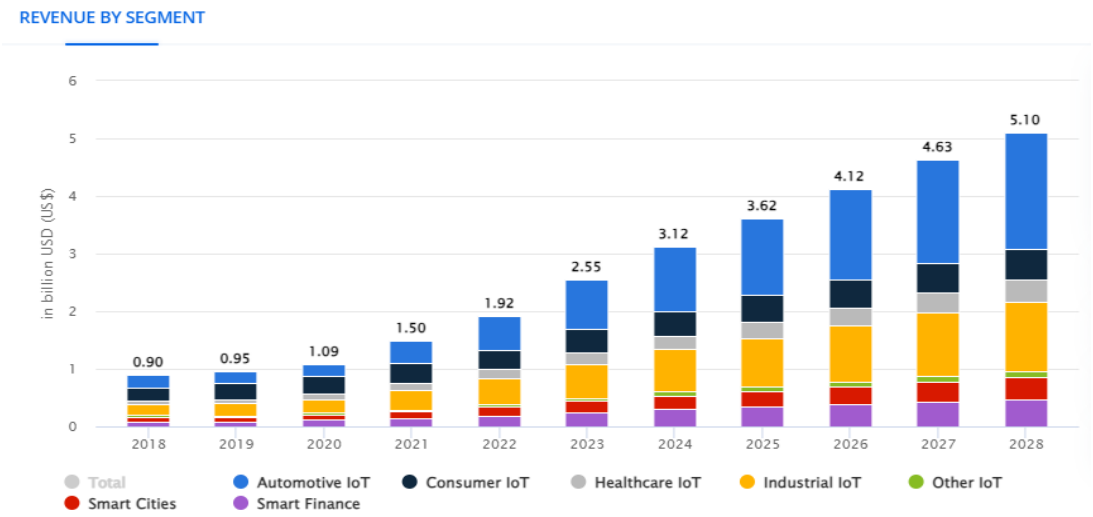


Source: Accenture research study [12]

The chart illustrates that when it comes to IoT solutions and implementation, the United States leads in the rankings, followed by a few European nations that are known for their higher level of development and digitization.

Romania does not count among the countries with a high level of informatics systems implementation, thus is has inadequate supporting conditions for a quick and effortless implementation of smart networks, making it difficult for enterprises (especially small and medium size companies that lack funds for costly investments) to cross from traditional semi-digitalized production environments to a modern approach that highlights Internet of Things qualities. This aspect is crucial for the Romanian economy as it will tremendous increase the productivity in domain like agriculture, manufacturing or the health sector, where specialized equipment is essential. The usage of smart IoT system is a must in future industrial improvements and should be taken into consideration more frequently by any kind of enterprise.

Picture 4. IoT related revenues in Romania



Source: Statista [13]

Because there aren't many options available in Romania's IoT market for someone to choose, in order to transform the already implemented projects into smart projects, the costs are high when comparing to other countries. Those factors plus the limitation in hardware and software alternatives rise the overall implementation price and prevent this segment to evolve as it should. On the open-source branch, there is almost no mobile applications that allow users to easily visualize and monitor the engines using their personal devices without paying a subscription or buying the seller products, both options being quite expensive. The best option is Grafana, which is an open-source online platform that allows individuals to create their own dashboard and link it to a real-time database for quick monitoring of their personal machines.

1.5.2 Presentation of the solution

This paper aims to project a solution to the issues outlined earlier in the Romanian industry markets and to help small organizations reduce production and reparation costs as much as possible. The goal can be achieved by providing an open-source software solution that will help users to monitor their hardware in real-time and discover weaknesses present in the structure of the industrial autonomous systems. Taking into consideration the economical insights provided by Statista [13], we can conclude that the industrial IoT market in Romania is still in its incipient state and cannot compete with other European countries yet. In order to increase the Internet of Things presence in Romania, this thesis will engage into multiple aspects of the current subject in the

following chapters and will attempt to provide an implementation by creating a prototype that is composed of two branches: software digital solution and hardware creation.

Structurally, the paper is divided into four chapters, their solely purpose being to explain how Internet of Things works by providing valuable information about the subject in question and then diving into the technologies used that are meant to help implementing the solution, as well as exemplifying it with a practical example.

The first chapter engaged the readers into discovering the history behind the Internet of Things phenomenon by providing beneficial intelligence about the domain of interest and the applicability of smart systems since the creation of the first IoT application. Humans understood the need of mechanical instruments that are created to facilitate the mass production in industrial environments by replacing the physical incapacity of human labor on certain tasks. Not only that, the machines coordinated by individuals are guaranteed to achieve a better output compared to the traditional production means, reduce production costs and time as they don't require a salary or breaks. The only thing that matters when talking to about a hardware component is its working conditions that are usually specified by the producer and the maintenance time interval for replacing some of its components, ensuing in preventing the machine to malfunction.

Internet of Things systems have evolved considerably in a short period of time, up to a level that is hard to imagine even for humans today, from smart houses that can be controlled remotely with only a smartphone or self-driving electric cars designed to take traffic decisions on spot, to human like robots capable of talking, walking or interacting with other persons.

The second chaper wants to bring insights on some of the hardware and software elements used in the creation of an IoT application for monitoring the quality of life for an industrial equipment. This part of the thesis will show some of the technologies present in the process of creation accompanied by their general presentation and usage tactics designed for a better understanding. Opting for splitting the hardware and software parts on two different branches is a must as some of the notions are quite complex, resulting in a structural way of deepening the information provided. The first branch will focus on the mechanical point by bringing out the physical elements like the electric motor, sensors, the router and microprocessor combined with the communication protocols used to share data. The software implementation is composed of multiple open-source digital solutions that aim to improve the functionalities of the static system by linking the components together and creating a flow of continuous exchanging messages inside the project, and as a result allowing the end user to visualize the measurements of the sensors and to take adequate actions if needed. To monitor the engine, the individual must remotely access a digital dashboard, in this case installed as a mobile app for Android devices. The interface of the software application should be easy to read and navigate by an inexperienced user, resulting in a

simple design choice of the UI. The measurements stored inside the real-time database will eventually populate the graphs and plots in the control panel section of the dashboard.

In the third chapter the paper will center on the informatics system and the logical structure of the application by creating diagrams using UML (Unified Modeling Language) techniques. For a project to be successful and to achieve the best final product possible, companies and developers focus to structure both the main and the secondary activities of the application using multiple kinds of diagrams that are easy to distribute, interpret or implement into a digital solution. The analysis of the system will be made using use-case, activity and class diagrams, representing the main methods of describing the model, as well as sequence and collaboration diagrams to understand the flow of the dynamic aspect.

The last part of the thesis, namely the implementation of the IoT solution, will dive into the creation of the mobile application that allows users to visualize the data received from the sensors in order to prevent a possible engine failure. The software is designed to run on Android operating systems, suitable for mobile phones or other smart devices, and includes multiple dashboards that provide valuable measurements as well as custom remote alert notifications. Each application page will be discussed is the forth chapter, starting with the login interface where the user should input his email to receive notifications and the network URL linked to the hardware system, followed the main page consisting of various graphs and dashboard elements, up to the customization of personal alerts and credentials page.

The practical project is meant to help small companies to understand and implement a relatively cheap solution into their working environments and replace the expensive alternatives provided by the Romanian IoT market in this incipient state, resulting in reducing the overall maintenance cost of industrial equipment. This kind of IoT system is mostly suitable for machines located in remote areas, where the data can be displayed directly on the phone instead of traditional on-spot visualization.

CHAPTER 2. TECHNOLOGIES USED AND ARHITECTURE OF THE APPLICATION

This chapter will bring into attention the technologies that are being used to create an IoT smart system, focusing on the ones that will form an IoT application for monitoring the engine general health. In order to do so, we will analyze the project from two perspectives: firstly, the hardware mechanism in which multiple independent gadgets create the final tangible machine and, secondly, the software programs that will allow for data communication and manipulation inside the network of the system.

2.1 Hardware branch

This branch consists of the totality of physical devices that form the hardware of the IoT system. Some hardware parts will incorporate or use software technologies that will be discussed in the second part of the chapter. The hardware is structured in three main levels: the group of measurement sensors, the communication network and the processing module.

2.1.1 Sensors

A sensor is a device that can intercept various inputs stimulus which can be represented by any unit related to physics in the surrounding environment [14]. The sensor's principal role is to transform the measurements recorded into electric signals for further processing. This application uses intelligent sensors based on SoC (system on a chip) micro-controllers from the ESP8266 family that can enable WiFi communications or run applications that allows them to have a WEB interface for better configuration. The ESP8266 hardware system of sensors permits 2.4GHz bandwidth WiFi and TCP/IP (also known as Internet Protocol Suite) connections. This project, for the current level of development, only uses two types of sensors, that being electric and thermic sensors.

The electric sensor (Shelly 3EM) authorize the measurement of the electrical tension as well as the power and amperage. In order to read the input strength, the sensors uses electric transformers that are able to measure up to 120A and indicate precisely the power factor.

The temperature sensor (Shelly 1) can measures the temperature input from three endpoints using the 18S20 thermic semi-conductors sensors and allows for measuring liquids temperature up to one meter depth for a maximum time of half an hour as it is IP67 certified.

For future development, the system is open to any other type of sensors that can connect in the network of the system either on one of the existing protocols or using an API (An application programming interface) .

2.1.2 Communication network

The chain used for communication consists of an local WiFi network in which all the devices in the system interchange information using the TCP/IP protocol. The sensors communicate using the MQTT protocol which is based on publish and subscribe actions to an MQTT broker. The broker is a software that registers all the data from the sources and redirects it to subscribers for each topic in question.

The LAN (Local Area Network) is a collection of smart devices that function in only one location, forming a closed environment, thus being limited. Other networks may be organized in WAN (Wide Are Network) or MAN (Metropolitan Are Network) which covers larger areas [15]. In our case, the Microtik Hap Lite RB941 controls the LAN and assure the functions of the grid by authentication of clients in the network, having an IP address assigning through DHCP (Dynamic Host Configuration Protocol) protocol, Network Address Translation function for connecting into the local internet and the Firewall for protecting the system not to be attacked form external sources. All the sensors are connected to the WiFi and are assigned IP address automatically from the router which will receive internet connection from an Ethernet cable using the proper slot.

2.1.3 Data manipulation processor

For gathering, processing and manipulate the data, the IoT system uses an Raspberry Pi 4 SoB (System on Board) with 4 GB RAM (Random Access Memory) available and Rasbian operating system. This kind of microprocessor is used in various ways, from building small hardware projects to implement large scale industrial applications and differs from an Arduino micro-controller as the last one is not as complex and offers less processing power. In order to deploy or execute applications on the Raspberry Pi it is necessary to use Docker containers which

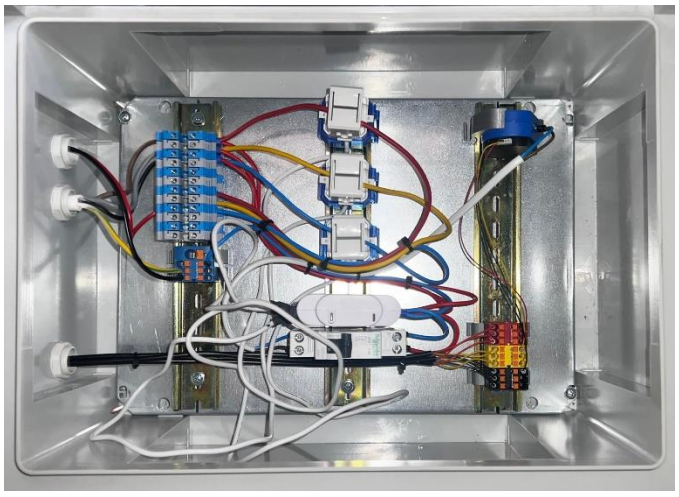
is a tool specifically created to be lightweight and to contain everything needed to run apps. For displaying the containers that are already installed or to edit the structure we can use Portainer-CE program. The Raspberry Pi 4 is equipped with an USB-C power supply port, micro HDMI ports supporting up to 4k displays, an Gigabit Ethernet slot for connecting it into the LAN, two normal USB slots and a micro SD entry.

2.1.4 Electric engine

An electric engine, also called a motor, is a physical machine that is connected to an electric source, allowing it to convert the received power into mechanical motion. The Techtop Single-Phase Capacitor Run Asynchronous Motors MY/MYT series provides a safe and reliable motor that can be use for the scope of this project as it is relatively small and easy to install, given its good performance. This mechanism presents low vibration and noise levels compared to the other options in the price range, as well as low energy consumption in relation with the total output force. These engines are perfect in case of low torque requirements and continuous working, making them suitable for personalized home appliances, water pumps, ventilation or recording meters.

The aluminium housing of the motor makes it easier to transport as it decreases the weight of the hardware and lowers the temperature of the engine while running consistently for long periods of time, reducing the possibility of overheating that can cause maintenance problems on the long run. The sensors will be placed on the case of the motor and will record the outside temperature of the engine while running.

Pictures 5 and 6. Hardware pieces used in the IoT application



Source: Personal photography of the hardware

2.2 Software branch

The software of this application consists of numerous digital development programs that are ment to implement the functionality of the project using diverse programming languages. In the current implementation, the software keep track of the physical mechanism that includes all objects related to the hardware branch that was discussed before.

2.2.1 Java programming language

According to Oracle documentation, Java is a general-purpose, class-based object-oriented programming language that is compiled by the Java Virtual Machine on byte-code commands and binary formatting [16]. It was created in 1995 by James Gosling and is continuously improving under Oracle management, having around 3 billion devices that run on Java architecture. This programming language works on numerous platforms like Windows, Mac, Raspberry Pi, being accessible to everyone as it is an open-source computer technology [17]. Following the information provided by Statista, Java ranks as the fifth most used programming language as of 2022 with 48% usage rate, making it a great tool for software development [18]. While Java is versatile and can be used in Big Data, Artificial Intelligence or Internet of Things, it has earned its reputation for creating mobile applications or mobile games by delivering complex functionalities and high compatibility across platforms as well as internal powerful libraries and frameworks.

2.2.2 Android Studio

Android Studio is an open-source software platform created by Google that allows individuals to build, debug or test complex Android applications. Android is a mobile operating system derived from Linux and was especially created for touchscreen smartphones in order to compete with Apple's Iphone running on iOS (Iphone Operating System).

Although the primary programming languages used in development are Java and Kotlin (created as an alternative to Java), some programmers can opt to use JavaScript or C++ as they are also supported. Android Studio presents interesting features like the Gradle system, device emulators that improve the design of the application as they provide real-time visualization when editing, the possibility of testing the implementation using a physical phone, XML (Extensible Markup Language) layouts that converts the design elements into readable source code as well as

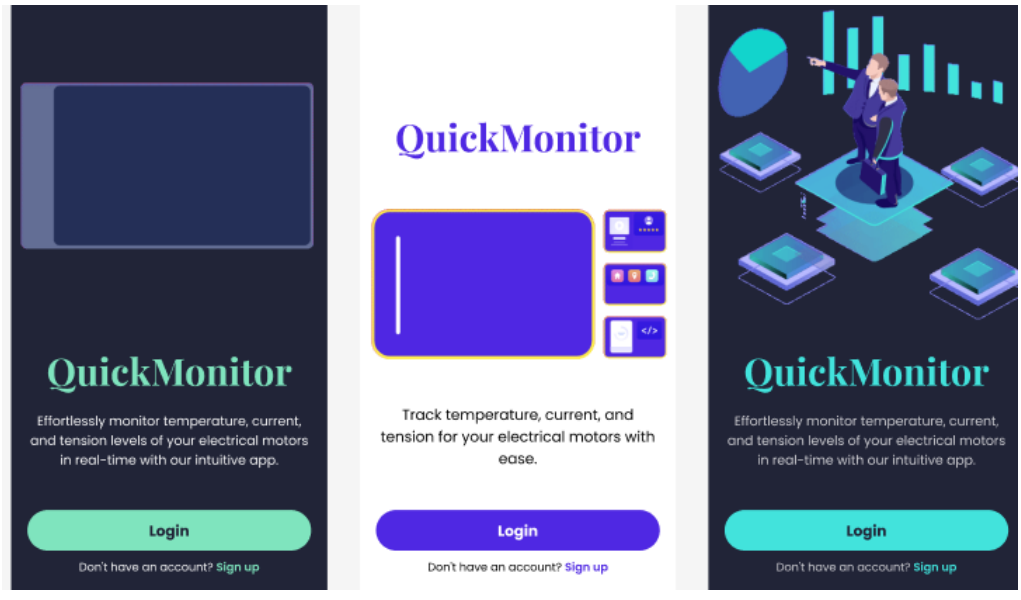
testing and monitoring internal tools [19]. Inside an application module we can find the manifest folder that includes the Android Manifest XML file which define permissions and transmits the information related to the Android project to the Android OS, the java folder that encapsulate all the source codes and the res folder which stores the design layouts and other user interface elements.

2.2.3 Figma

This design tool is built to engage people into creating and testing digital products for mobile applications, websites or any other informatics solutions [20]. Figma is a cloud-based platform that enables designers to collaborate in an easy and reliable way, unlike its rival Sketch, making it easier to agree on certain implementation or share information as all the activity happens inside a closed environment that can be accessed using a web browser. That being said, the project can receive corrections on spot, eliminating the traditional way of sharing design elements like pictures, fonts, colors or layouts throughout uncomfortable methods.

One of the most important features of Figma is Dev-Mode, which translates the layouts created by the designer into code, making it easier for the programmer to implement the solution. For example, the Dev-Mode attribute will provide all the insights about the fonts, colors or alignment of the elements in various formats like CSS or XML type.

Picture 6. Creating multiple designs for the front page



Source: Personal creation using Figma

2.2.4 Influx Database

Developed by InfluxData Inc., InfluxDB was released on 24 September 2013 and is one of the best options in the context of implementing IoT system monitoring, sensor data gathering or real-time data analytics for industrial purposes [21]. This tool is an open-source database that was optimized to work with time series data, possessing the ability to store, retrieve or process information in a fast and uncomplicated manner. Combined with and IoT system that incorporates a dashboard for displaying the data or the Grafana web platform to generate personalized graphics and dashboards, it forms a powerful tool in the information analysis domain.

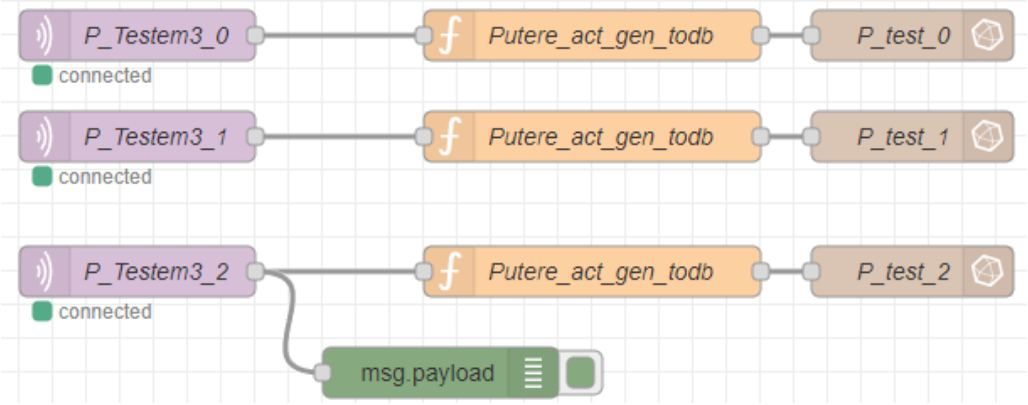
The InfluxDB gives the user the option to use InfluxQL (similar to SQL language, designed specifically for this type of database), SQL default language or Flux. It allows the data to be transmitted using HTTP or TCP protocols, being suitable for IoT implementations. For the execution of this project, InfluxDB is installed directly on the Raspberry Pi 4 microprocessor for better overall connectivity as the device takes care of the signals received from the router, the downside being that the older versions of the software do not have a GUI that can be accessed directly from the web browser using a specific port.

2.2.5 Node-RED

Node-RED is a great digital tool when it comes combining hardware inputs, internet services or application programming interfaces (APIs) as it provides easily connection to the GUI of the software using a web browser that empowers the user to control the flow of the application. This program's runtime functionality is built on Node.js, an open-source, cross-platform, asynchronous programming server based in JavaScript environment [22]. Prior to the creation of Node.js, JavaScript was limited to browser execution that was possible only on Chrome and Firefox. Additionally, only front-end applications could be build using JavaScript programming language, resulting in Node.js filling the gap by boosting the weightiness of its base executor.

All the flows inside the Node-RED environment are compressed and stored using JavaScript Object Notation (JSON) in order to reduce the size of the data and to be easier to transfer between multiple digital points. It has great compatibility with the Raspberry Pi 4 device that is being used for this project and it also can be linked to the cloud for a more dynamic approach. Node-RED working interface can be easily set up, letting users to customize each node through various icons and diverse color pallets.

Picture 6. Node-RED dashboard, elements and connections



Source: Personal creation in Node-RED

2.2.6 SQLite

Defined as a fast, self-contained and lightweight open-source SQL database engine, SQLite was built using C programming language libraries and it is considered to be the most used database in the world with over 1 trillion instances actively working [23]. Owing to the fact that it does not necessitate any custom installation or configuration and the overall implementation is easy and fast forward, many developers opt for using it in creation of simple architectures. This database management system was created in 2000 by D. Richard Hipp as an relational database to facilitate administration in various domains as it doesn't require any license, server, system or configuration.

SQLite supports a wide range of queries such as: data retrieval (select, from, where, etc.), data modification (insert, update, delete), structure modifications (create table, alter table, drop table), transaction control (commit, rollback) and aggregate mathematical functions (sum, count, avg, min, max). Being a lightweight product, it provides the possibility of choosing among only four primary data types, that being integer, real, text and blob. When combining it with Android Studio, programmers usually opt to use the Room implementation, which is defined as an additional layer over the basic SQLite functionalities in order to gain better synergy between the two. Room by itself provides compile-time verification of the SQL queries, specific annotations and database migration direction paths [24].

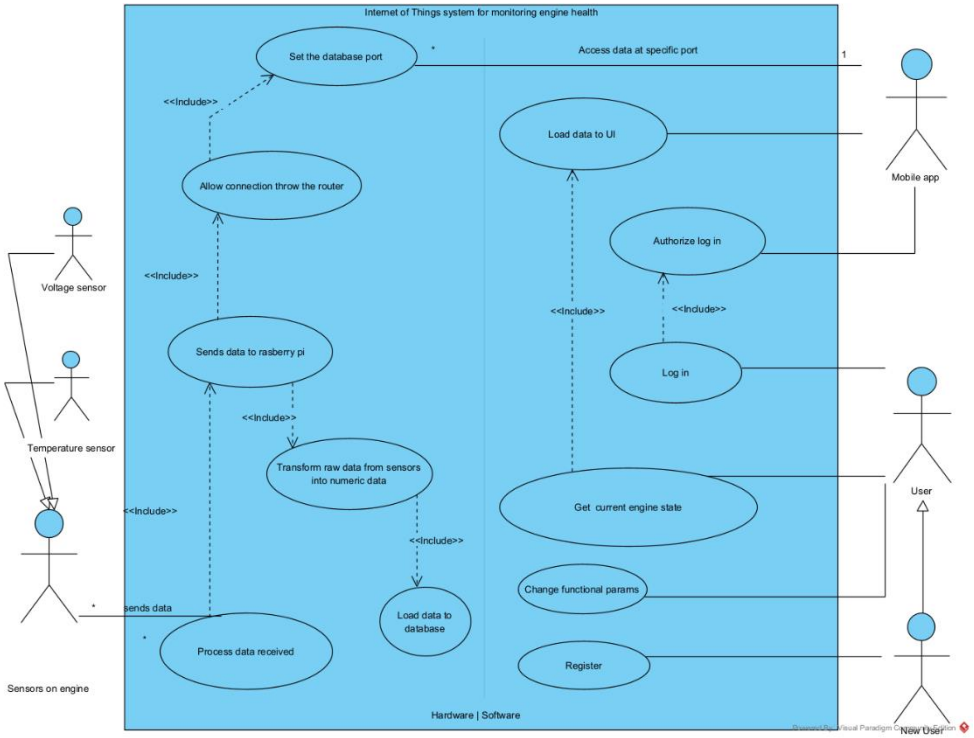
CHAPTER 3. THE INFORMATIC SYSTEM

Nowadays, in order to create a new software application designed for any domain or task, the developer should firstly create an informatics system environment that will project the functionality and the structure of the project. This is crucial because the logical structure dictates how the app will be used, how the user will interact with it, and most importantly it will allow the programmer to understand how to implement specific software artifacts and to picture the final product.

UML is a strong tool when it comes to designing and visualization of software systems as it facilitate the creation of the optimal output product by creating a full digital environment in which developers can design the structure using object-oriented elements and relational database linkage [25]. To analyze a system, UML provides different types of diagrams for various kind of activities, each having its own meaning: use case diagrams, activity diagrams, class diagrams, state machine diagrams, etc.

3.1 General Use Case Diagram

Picture 7. General use case diagram on the IoT system



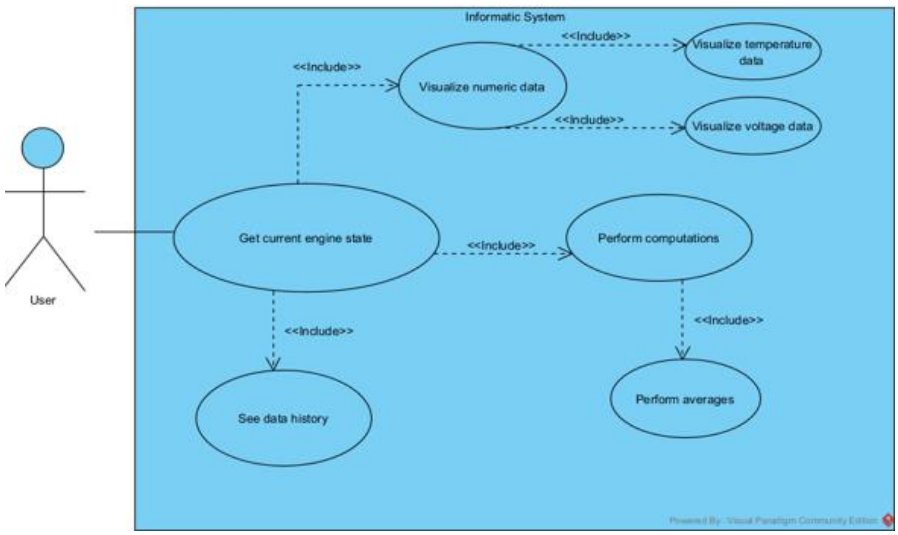
Source: Personal creation using Visual Paradigm

The purpose of the general use case diagram displayed above is to illustrate how the system works and how the actors in the environment are interconnected. This use case diagram is divided into two sections, one for hardware attributes and one for software functions. A segment line is used to separate the hardware from the software for better visualization and understanding of the IoT project. The two secondary sensors, one for temperature and one for electricity, compose the main sensor actor which is in charge of transferring information regarding the states of the engine. The mobile application it's itself an actor in the IoT application, while the human that manipulates the system is displayed as the "User" actor.

The temperature and voltage sensors are situated on the engine and their primary task is to send raw data at a specific time interval to the other IoT hardware parts for processing. The data is directed to the Raspberry Pi device, which receives the raw data and loads it into a real-time database installed on the microprocessor. The IoT system incorporates a router that permits for external devices to access the data stored in the InfluxDB database on a specific port. The mobile app connects to the database port and receives in real-time the precise data from the sensors. If the values exceed the safety limits imposed by the user, the app will display alerts by sending notification both on the phone and email.

The user has the possibility to register a new account or to log in using a valid email address and the URL of the project network, to change the working safety limits of the electrical engine and to analyze the values from sensors after the data is loaded into the UI through multiple dashboard elements like graphs and status bars.

Picture 8. Detailed secondary use case diagram of the IoT system



Source: Personal creation using Visual Paradigm

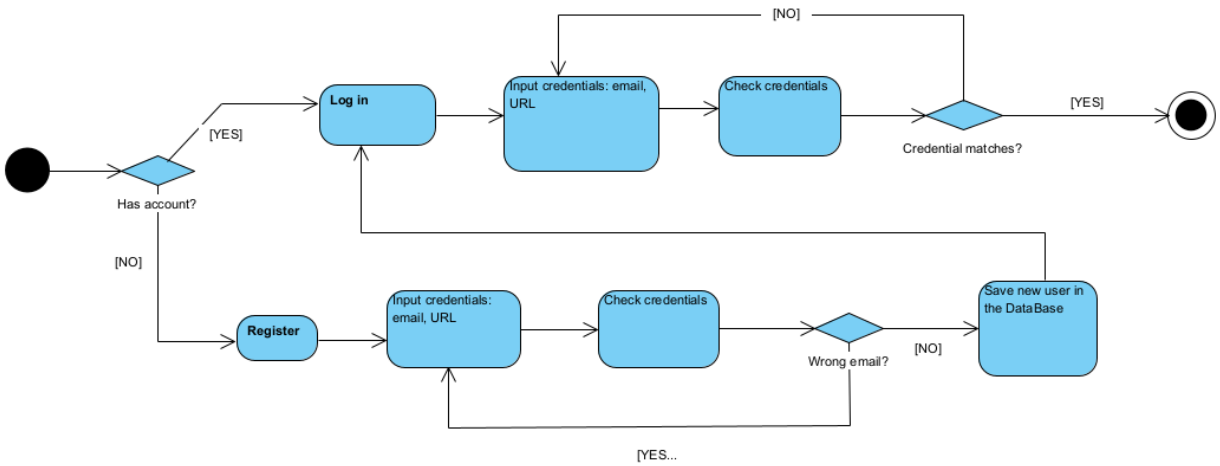
The "Get current engine state" element can be analyzed in detail using the secondary use case diagram which reveals other tasks linked to it. Many activities are essential for the flow of the application but not all of them are displayed into the general use case diagram as multiple small tasks compose one principal use case element.

At the level of "Get current engine state" use case diagram, the user can choose between multiple functionalities, namely: visualize data from sensors, access statistical computation and display data history of the engine from the database. All the information regarding the engine state are being transmitted by the sensors and presented in the mobile app. The user can see the average of the measurements in the UI interface by scrolling to the right element in the dashboard.

3.2 Activity Diagram

The dynamic aspect of the system can be easily described using activity diagrams as it is the perfect tool to illustrate the non-stationary elements of the project. The activity diagram is a behavioral type of diagram, same as the use case and state machine diagram, and describes all the operations between the starting point of the workflow up to the end point of the activity. This coordination of the actions of the system is done using UML elements specifically created for this reason, usually displayed as various shapes and arrows.

Picture 9. Log in / Register activity diagram

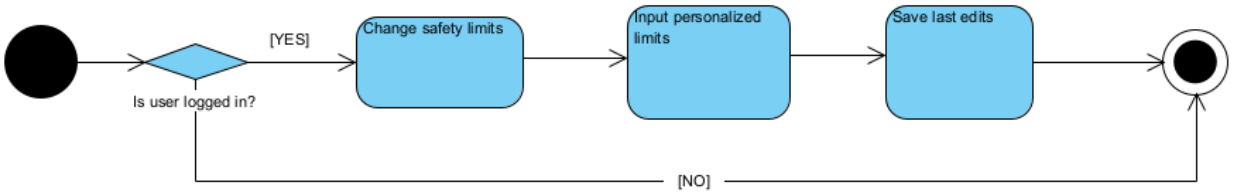


Source: Personal creation using Visual Paradigm

This activity diagram illustrates the process of logging in and registering in the monitoring application. When accessing the form, the user will be able to choose between login or register by

pressing one of the buttons. In the login case, the users will need to input their email and the network URL and if the data matches the information in the database, then the user will be logged in and will have full access to the information provided within the focused network. As for the register case, the user will need firstly to register its email and URL and then the system will save those credentials into the database, redirecting him automatically to log in page, where the input fields will be already completed with the user credentials by utilizing an auto-complete function. The diamond shaped element represents an if-statement with two possible outcomes, the positive one that meets the condition and negative option that does not satisfy the condition.

Picture 10. Change safety limits activity diagram



Source: Personal creation using Visual Paradigm

The user has the possibility of changing the safety limits of the engine if and only if he is logged in before the action so it has full control over the functionality of the product. The modifications are saved and the values are stored until the user decides to edit them again. Those limits prevent the engine to malfunction by alerting the operator in application or online using a Gmail server client.

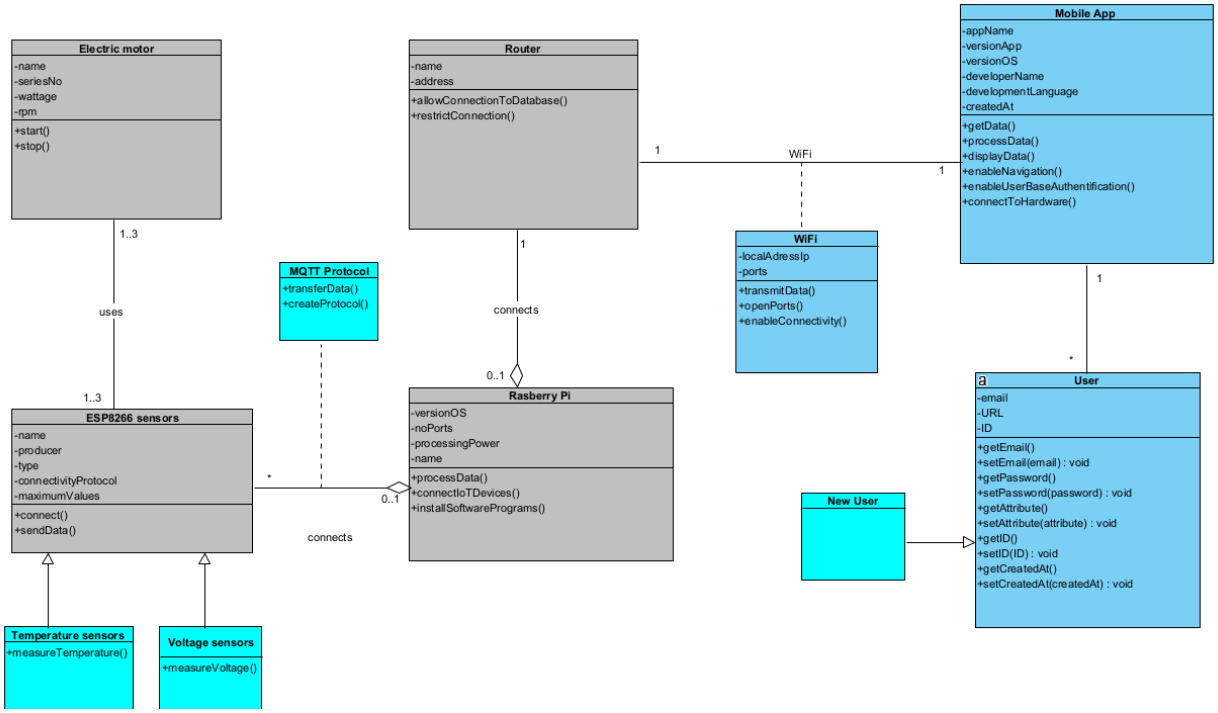
3.3 Class Diagram

According to Visual Paradigm documentation, a class diagram is a defined as an non-volatile, static structure that empowers the developer to create structural classes for the systems and to define the relationship between them [26]. The class diagram is a great tool in relation with object-oriented programming because of the logic used in implementation, which is similar in organization and programming purposes, resulting the fact that the developer can work on the solution faster and easier by looking at the graphical representation of the plots.

This kind of diagram is sliced into three components, namely the name, the variables and the functions. All of those traits are found in object-oriented programming and form the basis of understanding the informatics environment. The name of the class is used for identification, the

attributes represent the personal characteristics of each class and the methods or functions are actions that can be taken by any object of that class.

Picture 10. IoT class diagram



Source: Personal creation using Visual Paradigm

The IoT system in question is divided into multiple classes for each primary component like the engine, the microprocessor or the user. Each class has its own attributes, usually marked with a minus sign in front of the name, thus creating a private field. For public fields that can be accessed anywhere inside the project, one should use the plus sign. The methods can be found at the bottom of the class diagram and are marked with two round brackets.

Between the classes we can found arrows and lines, their solely purpose being to interconnect the classes and form a tight system. The normal line is called association and can present numbers at its end points, also known as multiplicity operators. The empty arrow marks a relationship based on inheritance (for example two types of sensors are merged into a single class as they have the same functionality), while an empty diamond represents the aggregation link (the child class can exist without the parent class, as is can be seen in the router and sensors approach, where none would have a functionality without the microprocessor).

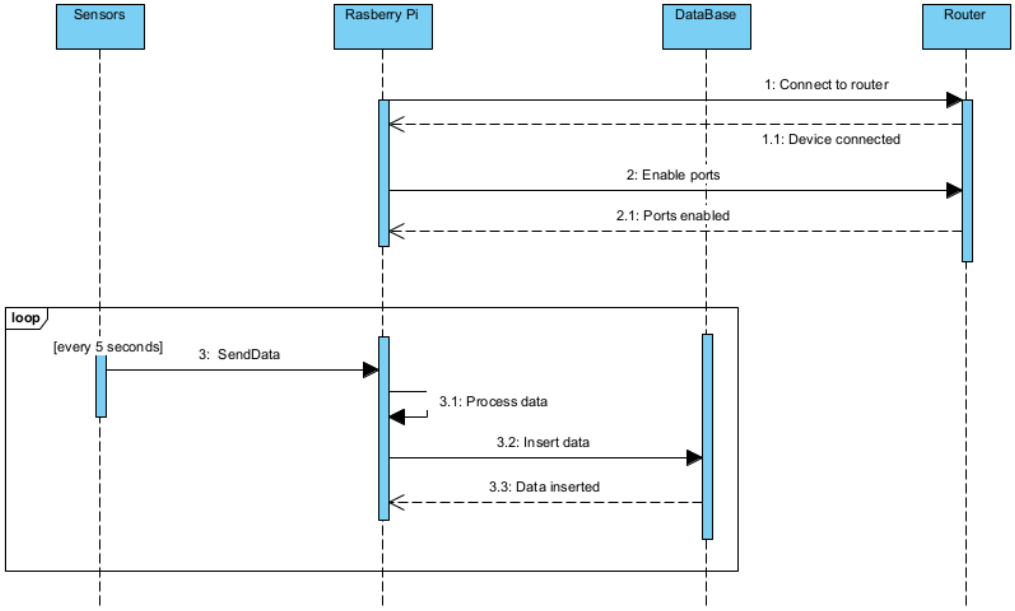
The dashed lines represent anchors that add a middleware element between two main components. The MQTT protocol allows the sensors to transmit data fast and secure to the "brain"

of the hardware, the Raspberry Pi 4, while Wi-Fi enables the router to push measurements into the mobile app efficiently.

3.4 Sequence Diagram

A sequence diagram purpose is to illustrate how interactions are made between the objects of an informatics system, taking into account the moment when the action took place in time. In order to display this, a sequence diagram uses long vertical lines, also called lifelines, as well as arrows with action related messages to link the lines.

Picture 11. Sequence diagram of the IoT system



Source: Personal creation using Visual Paradigm

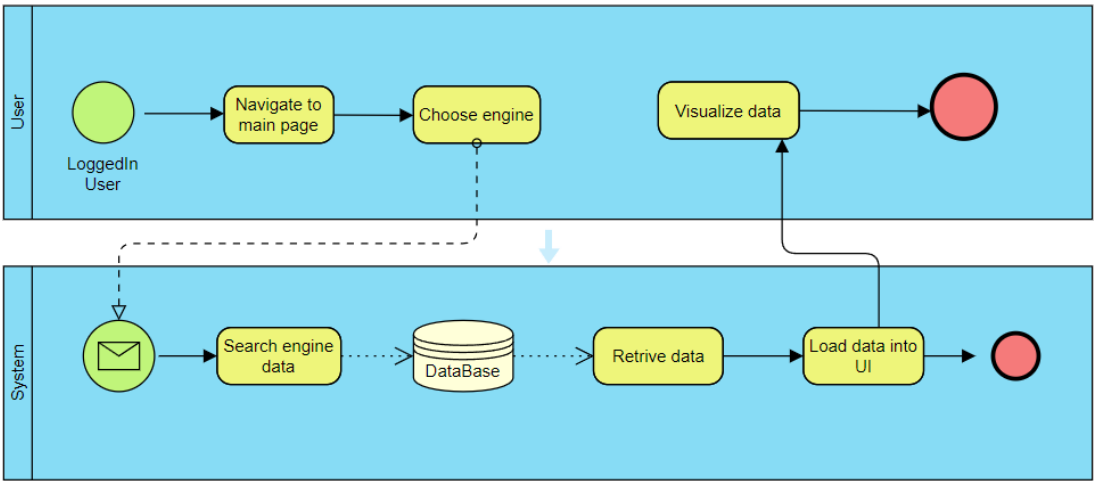
In the figure presented above, there are four components included in the IoT system that are linked together and exchange messages in a timely effective way. The sensors transmit information continuously every five seconds, creating a loop in relation with the Raspberry Pi 4 device. The data is directly loaded to the database after the microprocessor have established the connection to the Wi-Fi router and the ports are enabled for the systems installed on it.

This mechanism is the most important aspect of the communication inside the IoT system for engine health monitoring since it creates the basis of the functionality of the application by combining multiple hardware pieces to the main database storage, from which all the measurements can be extracted and loaded into the mobile application dashboard.

3.5 Collaboration Diagram

A collaboration diagram is formed by linking multiple actors and classes together in order to analyze how they communicate and engage in certain actions. By working together, two or more object of a class can create a new functionality for the system in question, almost the same as the sequence diagram. This type of diagram is mostly used in simple design architecture, since it allows the creator to use multiple symbols and elements that would decrease the overall visibility, making it hard to be understood [27].

Picture 12. Collaboration diagram describing the link between User and System



Source: Personal creation using Visual Paradigm Online

After the user logs in using the personal email and the network URL, he needs to choose an engine using the dashboard in the main page in order to see its measurements. The software system will search for the data specific to that exact engine and will output it in the main dashboard consisting of multiple graphs and plots, as well as exact calculations based on the information retrieved. When the data finish loading, the user will be able to interact with the dashboard.

All the UML diagrams are unique and play an important aspect in modeling the structure and the design of a system since they allow for better visualization of the behavior of the elements, establish connections between multiple classes and actors and permits them to exchange information, making the analysis and testing of the functionalities easier for the developer.

CHAPTER 4. IMPLEMENTATION OF SOLUTION AND FUNCTIONALITIES

The fourth chapter of the paper will center its focal point on the practicality of the application by covering all the functionalities and the implementation of the main elements as well as explain how users can interact with it. The users are not required to know anything in advance about the hardware system except for the URL that links the software to the hardware and the limits of the electrical motor that was previously chosen to do a certain specified task, resulting in an overall easy and pleasant digital experience.

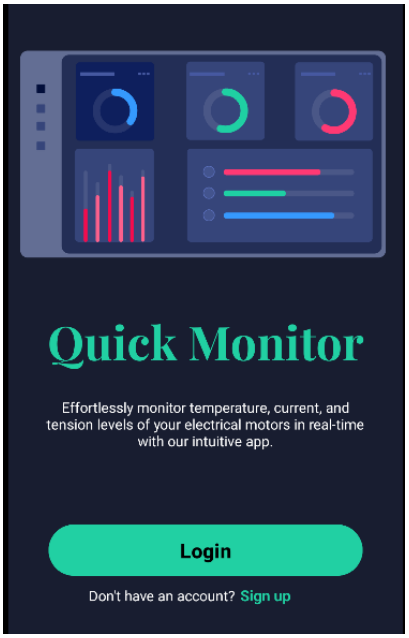
The whole mobile application is designed to run on Android OS and was created using Android Studio running Java programming language. The interfaces and buttons were designed using Figma, a great tool for managing layers of design, shapes, colors or dynamic attributes. The background color, the color of the buttons or the text fonts used were all carefully chosen from the beginning because the end product must be simple, modern, easy to read or to navigate into and all of the important aspects like alerts or engine safety parameters should be clearly defined – Appendix 1. In the following pages the paper will reveal the design and the code used into the creation of the app and the justification of choosing specific elements.

4.1 Entry Page

The first page of the software solution greet the user with a dynamic illustration which resembles with an industrial type of dashboard for monitoring multiple kinds of measurements with circular progress bars, graphs or line charts. This design element was created using LottieFiles, an open-source animation website that allows programmers to insert various design elements into their code by downloading certain online libraries, thus being very common in mobile Android projects. The login and sign up buttons have the same color but differentiate themselves as the first one is the main button of the page, in consequence being larger and more prominent so all the attention is cast towards it, while the second one is only in bold font type and is included in the text below, hence resulting into a secondary button alternative. Both will send the user to different pages – Appendix 2, one developed to directly log in the user into the main page of the application only if they input a valid email and URL link that has been already registered into the system while the other will create a new entry into the database with the new

customer's credentials. After registering, the individual will be redirected to the login page, where an auto-complete function will input the data that just has been registered in the process of creation of the new account, achieving an easy and fast forward process for even the most inexperience person. In order to check the validity of the inputs the app uses a verification function that checks if the format of the email or the URL are correct, and only if the test is passed the user is allowed to continue. – Appendix 3

Picture 13. Login page Quick Monitor App



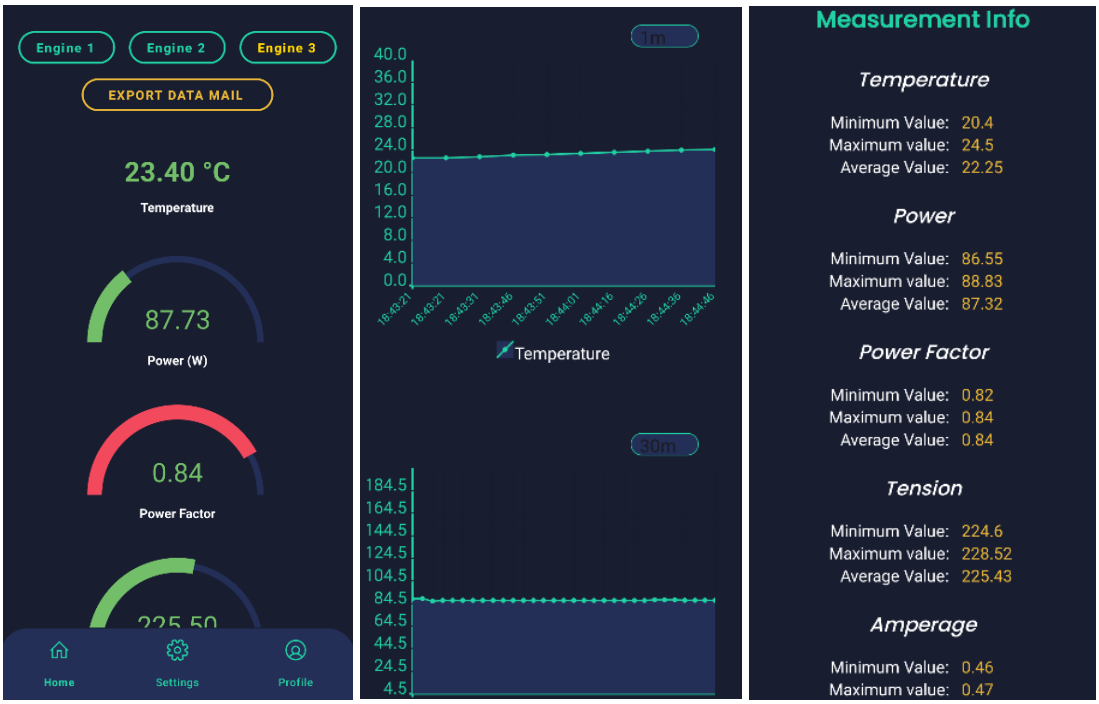
Source: Personal creation using Android Studio

4.2 Home Page

The dashboard page is the core of the application, loading the measurement data received from the physical smart sensors through InfluxDB timestamp database and multiple communication protocols to the digital graphs and circular progress bars present in the home layout. It is the most important page of the application and serves the purpose of a control panel for monitoring industrial equipment, designed in a way that is simple and easy to understand for anybody. In the top corner of the screen users will find up to three buttons, each corresponding to an electrical motor, that will dynamically load into the dashboard information about the engine in question and will erase the previous populated graphs and replace them with new ones once the user decides to change the targeted motor. When lunching the home page, the application will load by default the data from the first known working engine and will display it on the screen without any need of interaction. If pressed, the button will turn from light green to yellow, keeping track

of which engine is in focus at that current time. Right below those buttons users will find a new clickable element that allows for exporting the history of measurements to the email that is currently logged in – Appendix 4

Pictures 14, 15 and 16. Main dashboard



Source: Personal creation in Android Studio

The measurement for temperature is written in numeric format in order to be easily seen as it is the most important attribute of a running engine, followed by the power calculated in watts, the factor of power, the electrical tension expressed in volts and the amperage. Each of those kinds of measurements are crucial in the field of preventive and predictive maintenance and need to be studied in order to prevent the malfunction of the system, or even worse, a major problem that shuts down indefinitely the entire equipment. Circular progress bars are used in modern system dashboards for their capability of grouping the interval of data inside an arch like structure, enabling users to better visualize the information as the progress increases when it is closer to the upper limit of the interval. A progress bar is a element of design that reveals the progress of a certain task and is very much used in Android programming because it is highly customizable and it can be found into the real-time default editor of the platform [29]. Because the interval of each measurement may vary, it's mandatory to show the numerical value under the graphical representation. Another important aspect of the design of those progress bars is the color that can be changed to show the state of the hardware for that specific measurement. As it can be seen in the picture above, the green color represents the fact that the electrical motor is working in normal

conditions, while the red color depicts a situation where the electrical engine might have an urgent problem that could cause damage. For example, in the example provided in the picture, for the best output provided the power factor should be as close to one as possible, one being the best running condition that can be achieved. Even though the engine is not running at one hundred percent power rate, it will only affect the output force and not the equipment itself, while the temperature and electrical measurements may break some of the mechanical parts if they exceed the normal threshold because of working under high stress, usually for longer periods of time.

Graphs are the best tool to use for displaying concomitantly live and history data, creating a structured interval for all the values of a measurement. In the current implementation, the graphs are customized for each type of evaluation, having on the horizontal x-axis the limit interval and on the vertical y-axis the timestamp of each input. The drop-down menu enables users to choose the time interval they want to analyze (from small time segments like one, two or three minutes up to twenty four hours) and to better understand the running patterns of the equipment during the working program. The dynamic character of the graphs can be seen when a new value is registered into the database and the graph redraws itself, this time by pushing the oldest value in the interval out and inserting the new one accompanied by its timestamp.

Picture 17. Part of the method for plotting the graph

```
plot.clear();
XYSeries series1 = new SimpleXYSeries(Arrays.asList(seriesNumbers), SimpleXYSeries.ArrayFormat.Y_VALS_ONLY, title);
int endIndex = series1.size()-1;
int startIndex = Math.max(0,endIndex-interval);
LineAndPointFormatter series1Format = new LineAndPointFormatter(
        getApplicationContext().getColor(R.color.green_avocado),
        getApplicationContext().getColor(R.color.green_avocado),
        getApplicationContext().getColor(R.color.blue_ediText),
        plf: null);
plot.addSeries(series1, series1Format);
plot.setRangeBoundaries(lowerBoundry, upperBoundry, BoundaryMode.FIXED);
plot.setRangeStep(StepMode.INCREMENT_BY_VAL, increment_range_by);
plot.setDomainBoundaries(startIndex,endIndex,BoundaryMode.FIXED);
plot.setDomainStep(StepMode.INCREMENT_BY_VAL,increment_domain_by);
```

Source: Personal creation in Android Studio

Even though graphs exists in Android programming widgets, plotting this exact graph is made possible using the XYSeries library that enables developers to post customized charts, setting the upper or lower boundaries, colors, dynamically changing the step of the measurement and real-time modifications. In this example, every time a graph is rendered, the previous one is cleared, the series that will eventually populate the graph is defined, and the right colors are set for the graph lines. After these steps, the ranges of the vertical and horizontal lines are set within the values received in the parameters of the method, resulting in a functional, well designed illustration for visualizing current information and history data.

Another important part of the dashboard is the real-time data calculator that stores the minimum, maximum and average value recorded for each type of data. This may not seem to be much, but in reality it is very important to keep track of those basic information as it can expose malfunction trends over periods of time. For example, if the average temperature keeps increasing every week by 0.2 degrees Celsius, the normal on-spot measurement won't be able find the problem on time as the temperature may vary depending on the moment of the day, the environment temperature, the air supply and other external factors. By analyzing average values for the same interval of time over the years, users can discover changes in the behavior of the equipment, resulting in a better management of the hardware and life expectancy.

All the instances of the informational dashboard are placed inside a Scroll View, which is an Android element specialized in allowing the user to scroll through multiple widgets of the same activity that spread more than the visible screen. It enchants the overall user experience, creating an universal way of navigating the platform because many smartphones nowadays have different display sizes or ratios, depending on the model and producer.

For the dashboard to be functional and to receive all the information from the working environment, the data loaded into the application is firstly received using e fetcher method. This function is characterized by its ability to execute a task independently, at a given interval of time. In order to achieve this, a scheduled executor was used to run a predefined piece of code every ten seconds, connecting to the database after accessing the input information of the client which were previously stored into a shared preferences object. The database messages are converted into a JSON array that will finally be processed element by element, resulting a new string array of explicit measurements. After the outcome is secured, the connection to the database is closed and will open again after another ten seconds, while the data set is forwarded to other activity for further manipulation – Appendix 5.

Notifications are essential for a monitoring system as they alert the user when the device is not in focus that something may go wrong. When the sensor data is processed and some out of range anomaly is detected, the system will automatically update the user by sending text alerts through normal app notification and personalized emails with the measurements recorded so adequate actions can be taken. The thresholds are set to universal default values that can be personalized by the person in charge of the smart system, depending on the task of the motor. Current Android application require an SDK version above Tiramisu in order to send notification in an up to date format, the downside being the need to receive a permission to display the message, usually received as a pop-up when firstly lunching the application. – Appendix 6

Picture 18 and 19. Application notifications

Source: Personal creation

The navigation bar is situated at the bottom of the page and it solely purpose is to facilitate the navigation inside the mobile platform. The footer is split into three compartments, each for a page of the application: the home dashboard, the settings interface and the profile activity. The home page was discussed before and it allows users to visualize measurements, while the settings interface empowers the client to change the limits of the sensors so alerts can trigger on point. The profile page displays the user email and URL that were previously inserted into the login page and enables the client to change the credentials. When one of the buttons is clicked, the current activity is replaced by the new targeted one in the top of the activities' stack, while the old one remains in stand-by. The navigation element is created separately in a new XML document and imported into all three main interaction activities. While the user is browse the Scroll View layout of the interface, the footer bar remains into the same position at the bottom of the screen, employing a static centering using a Relative Layout type.

Picture 20. Bottom navigation implementation in Home page

```
bottomNavigation.setOnItemSelectedListener(new NavigationBarView.OnItemSelectedListener() {
    1 usage  Alexandru Mihai Ionel
    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        if(item.getItemId() == R.id.navHome)
        {

        }
        else if(item.getItemId() == R.id.navSettings)
        {

            Intent toSettingsIntent = new Intent(getApplicationContext(), SettingsActivity.class);
            toSettingsIntent.putExtra( name: "keyHome",receivedUserLogged);
            toSettingsIntent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_SINGLE_TOP);
            startActivity(toSettingsIntent);

        }
        else
        {
            Intent toProfileIntent = new Intent(getApplicationContext(), ProfileActivity.class);
            toProfileIntent.putExtra( name: "keyHome",receivedUserLogged);
            toProfileIntent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_SINGLE_TOP);
            startActivity(toProfileIntent);
        }
        return true;
    }
});
```

Source: Personal creation in Android Studio

If the user that is logged in wants to navigate to the Settings or Profile page, the system will create an Intent with all the account credentials and will send it further to the desired destination. Using the "startActivity" function combined with the new created Intent as a parameter, the application will lunch the targeted activity and will transmit the Intent data. The flags related to the intents are important in Android programming, as the first one, namely "CLEAR_TOP", is used to clear the stack of activities situated above the target and checks if the activity to be lunched is already present in the stack. If the activity cannot be found, then a new instance of it will be inserted into the stack. Moreover, the "SINGLE_TOP" flag searches the instance of the activity at the top level, and if it already exists in that particular spot, it won't be duplicated.

4.3 Settings page

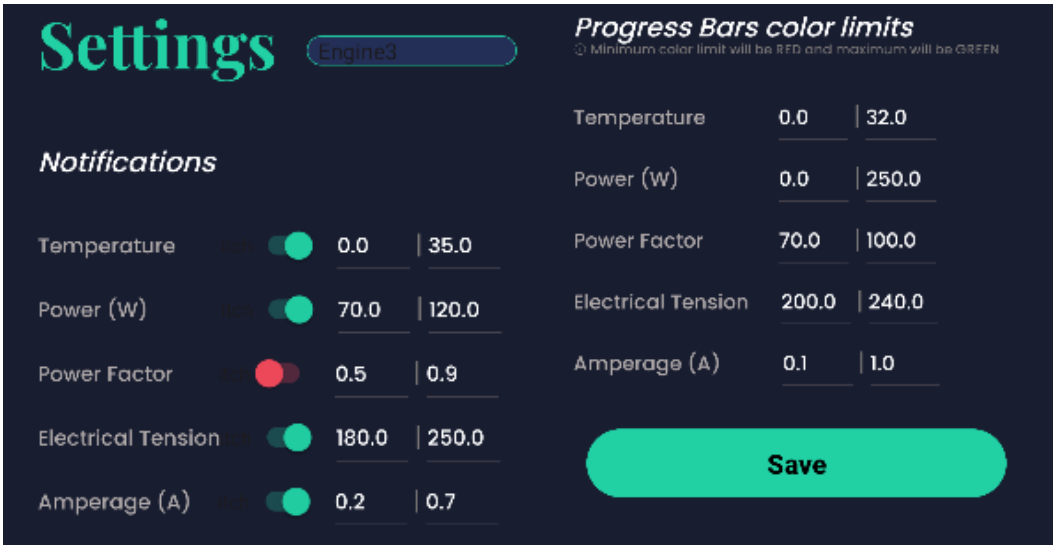
In order to customize the alert messages that are being sent directly from the digital system or to edit the interface appearance, users can opt to modify certain elements inside the activity provided at the Settings section. The pictures below depicts the Settings Activity which is composed of the Notifications area and Progress Bar color limits part, each having its own functionality, permitting the person in charge of the maintenance program to modify certain aspects of the application.

The first step that must be taken is to choose for which engine to modify the default values, resulting in three different setups that can be used for all the electric motors. The default choice is the third engine, which is incorporated into the current implementation and is linked to the physical hardware part. After this action, the user can modify the maximum and minimum thresholds for each measurement and mark it as active. If the measurement alert is marked in green, it will send the information regarding the state of the engine via email and app alerts, while if it is marked as red, thus being in disable state, it will not alert the customer at all. As mentioned at the level of sending alerts, the message will send to the end client the measurements of the engine in question that exceeded certain imposed values.

The second step consists is the option of overwriting the default coloring of the progress bars in the second part of the Activity by changing the upper and lower limits of the interval of colors. This in an important aspect as it will allow the user to better visualize and understand the state of the engine on spot just by looking at the color of the circular progress bar. If the data is situated inside the maximum and minimum limit of the interval, than the engine is working

properly. Further on, the person in charge can analyze the numerical values and see if the engine is running in normal conditions and respecting the producer advice. If the electric motor is working in normal conditions, the color of the interface elements will be marked as green, while if the engine is functioning outside the imposed interval, the color will change to red.– Appendix7

Pictures 21 and 22. Settings page



Source: Personal creation in Android Studio

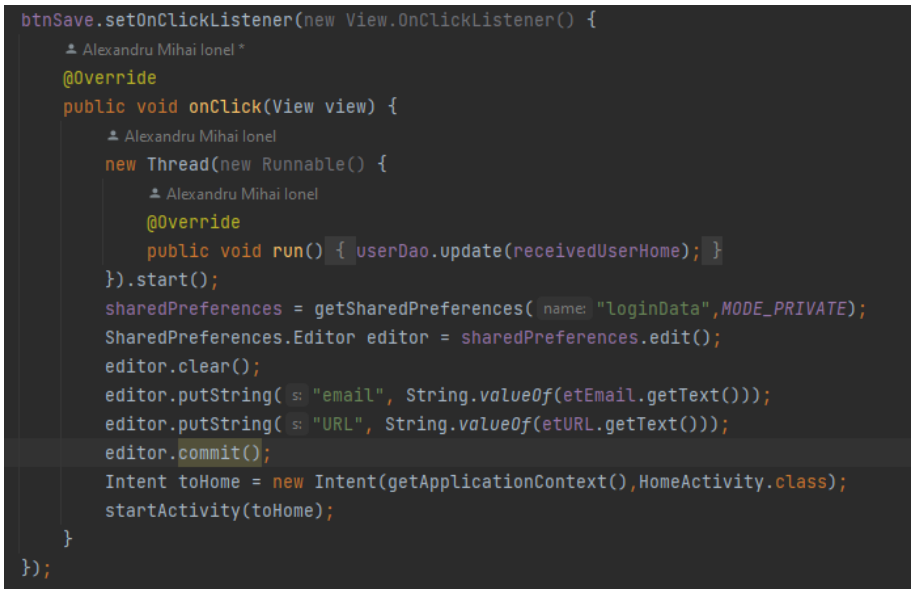
The Save button will register the modifications made in this activity and will apply them directly when the dashboard is loaded. In order to create a functional dashboard, the changes must take effect immediately and the value registered should be stored and remain the same after the application restart or shutdown. The system does not require a lot of personalization and offers the option of modification only for the relevant aspects that are oriented to the functionality of the application, making it simple and straightforward for any type of client, experienced or not.

4.4 Profile page

When an user registers into the application, it basically creates a new entry in the local database, resulting in a new created profile. This profile incorporates its unique email address and the URL of the system network that is meant to be analyzed. When opening the Profile Activity, the credentials are already matched by the system with the ones currently in use, which were registered at Login phase. Here, the client can opt to change one of the credentials or both of them by pressing the pencil icon present in the field box and retype the new information. This type of

interface is mandatory in a modern application as it facilitate modifications of personal data, resulting in a better experience and customization. Once the edit is done, the user need to save the changes made into the database by pressing the Save button.

Picture 23. Save user after edit in Profile Activity



Source: Personal creation in Android Studio

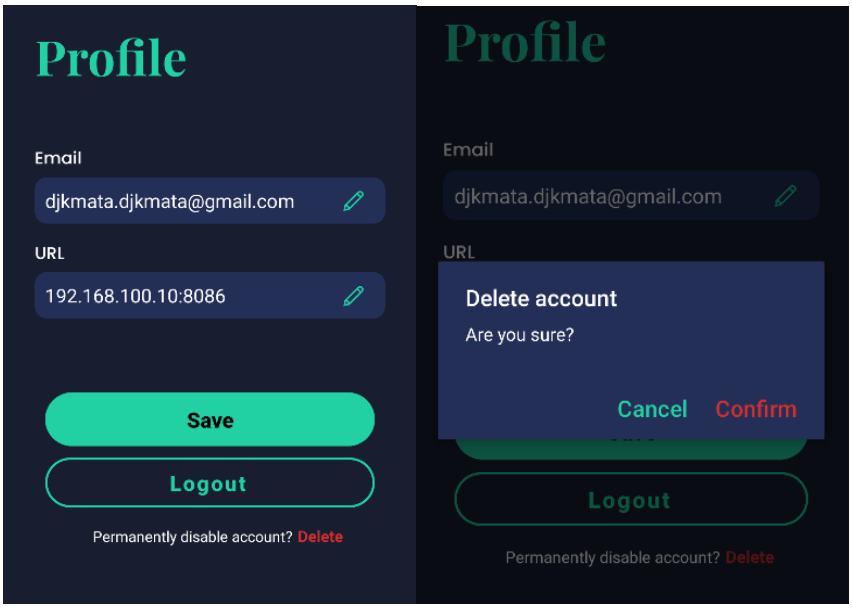
The functionality of the Save button can be seen in the piece of code provided at Picture 23. The user is updated in the database with the information written in the EditText elements in the moment of writing the new credentials. Opting to perform this action on a new thread is a widely spread tactic to avoid blocking the main thread of the application and keep the activity responding. The Shared Preferences will access the part of memory allocated for storing the Login data and will replace it with the new values provided by the user. After the change is complete and the credentials are stored, the Home Activity will be launched automatically.

The Logout button enables the individual to stop the usage of the current account, redirecting him back to the Login page while waiting for new inputs. A special red button is used for the deletion of the account, launching a safety pop up when pressed that will ask again the user if he wants to perform the deletion task as the action is irreversible. If the option for deleting the user is chosen, then the database will be cleared of all information regarding the account and the individual will be directed to the first page of the application. If the user wants to access the functionalities of the smart system again, he will firstly need to re-register the account.

Modern application use this type of coloring, highlighting the buttons that are usually meant to be pressed by users. In the example exposed below, the Save button is designed to be filled with green as it is the optimal and the primary button of the page, while the Logout is a secondary button and it is not filled in order not to draw attention towards it. The third button,

namely the Delete button, is colored in red in order to signal an action that may have an impact on the account if not used correctly. – Appendix 8

Pictures 24 and 25. Profile page



Source: Personal creation in Android Studio

4.5 Important methods within the application implementation

This sub-chapter will be focused on discussing other important procedures and methods used in the creation of the project. Every function present in the implementation files of the application is essential for obtaining a stable product that can be used for monitoring the hardware. In order to understand the particularities of the functions related to data processing and manipulation or designing the interfaces, this part of the paper will include some of the methods used in the current development and will try to explain their purposes.

The real-time Influx Database provides the measurement data together with the date of the their recording, the only problem being the timestamps in UTC format, while Romania is situated in the UTC+3 timezone. In order to treat this issue, the application uses a time parser method that will format all the timestamps into UTC+3 format. This action is made possible by parsing each element of the initial array of values provided from the storage software into the correct date representation through the SimpleDateFormat class provided by default in Java programming for Android Studio. The method will set the new timezone to GMT+3, which is the equivalent of UTC+3 suitable for Romania timezone, and will return the new corrected array. – Appendix 9

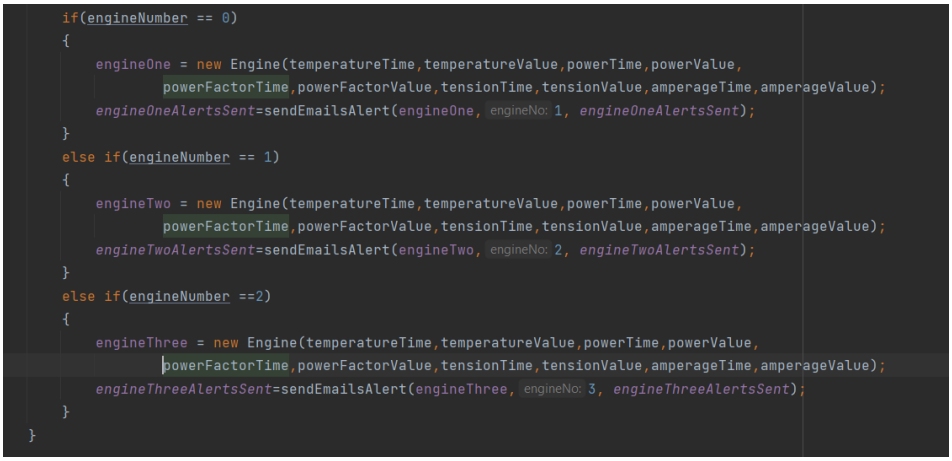
Another important interaction with the system is made when the user wants to display information about another engine, resulting in manipulating the whole dashboard to show the specified data belonging to another machine incorporated in the hardware solution. If one of the buttons is clicked, the graphical elements of the dashboard at that particular time will load again the data of the new engine in focus, meaning that the graphs and the progress bars will modify their values to match the current electrical motor measurements. In order to do so, the history of the data for the engine in question will be received, followed by the action of populating with new values the graphs and other UI elements. – Appendix 10

After the values are loaded, the user can interact with the graphs and change the interval of values that are displayed on certain periods of time. If the user decides to change the spinner default value, the application will display the graph again with the history of the results that are stored into the real-time database. – Appendix 11

Live data is injected into the application using a fetcher method that receives the data from the database and creates three engine objects in the current configuration. Each engine has its own different measurements that were previously recorded by the sensors situated in the working environment. The data is received in this function in a simple format, all information being stored inside a string and are separated by comma. The information is split into multiple values that are stored into a string array of variables representing the measurements made on the motors. After all the inputs are received, the system saves the values into other variables in order to structure the code better and creates engine objects using the constructor with all the attributes based on the variables stated before, followed by a checker method for sending alerts if the values previously saved exceed the normal imposed limits. Only three engines objects are created as the application is designed to accept just three elements of that type in the same time.

Picture 26 and 27. Fetcher method for creation of engine objects

```
//method for populating the engines with their values and send emails if they exceed the maximum threshold
1 usage   Alexandru Mihai Ionel *
public void fetchLiveDataForEachEngine(ArrayList<ArrayList<String>> groupedEngines)
{
    for( int engineNumber=0; engineNumber < groupedEngines.size(); engineNumber++)
    {
        ArrayList<String> displayEngine = groupedEngines.get(engineNumber);
        //get all the information for engine as a String
        String allData = String.valueOf(displayEngine);
        //get rid of the [ in the begining and ] at the end in order to get unaltered values
        String editedData = allData.substring(1,allData.length()-1);
        //divide the string in order to get the exact values for each measurement
        String divideData[] = editedData.split( regex: ",\\s*") ;// in ordet to split by , and space
        // Create 5 attributes for each measurement of an engine
        String temperatureTime = divideData[1];     //T
        Double temperatureValue = Double.parseDouble(divideData[2]) ;    //T
        String powerTime = divideData[4];           //P
        Double powerValue =  Double.parseDouble(divideData[5]);          //P
        String powerFactorTime = divideData[7];     //PF
        Double powerFactorValue = Double.parseDouble(divideData[8]);    //PF
        String tensionTime = divideData[10];        //V
        Double tensionValue = Double.parseDouble(divideData[11]);       //V
        String amperageTime = divideData[13];       //I
        Double amperageValue =  Double.parseDouble(divideData[14]);      //I
```



Source: Personal creation in Android Studio

The last function discussed in this chapter is the one in charge with sending emails to the users when one or more measurements exceed the value imposed in the Settings Activity. The state of the measurement is obtained from the allocated memory by the Shared Preferences class, and if the resource fails to arrive, a default value will be set depending of the type of the registration. A string variable will save the message to be sent to the user depending on which threshold limit was surpassed and will contain a concise text regarding the state of the engine at that time. After the alert text is saved and the process is started, a flag variable of type integer will be marked and will allow the system to deliver the email to the address that is logged and start a pop-up notification transmission directly in the mobile phone. – Appendix 12

CONCLUSIONS

This paper aimed to increase the awareness of businesses in the domain of Internet of Things by creating a structured thesis on how to implement a smart system solution in today's economy. The project provided a simple and straightforward example of an IoT system specialized in monitoring multiple electrical engines and displaying the measurements registered by the sensors situated in the proximity of the hardware throughout a modern mobile application dashboard.

Firstly, the hardware played a crucial role in obtaining the valuable information that could predict an incoming failure, resulting in reducing the overall repair costs and boosting the enterprise profit in the long run. The sensors were mounted directly on the electrical engine and in its vicinity and their solely purpose is to register and send data regarding the state of the main component. The information is then redirected to the component in charge with processing within the hardware system, the Raspberry Pi microprocessor, on which the clients of Node-Red and InfluxDB are already set up. The Node-Red software program handles the links between the end-points in charge with data administration, while the Influx Database saves the values registered before, concatenates the timestamp of each recording and offers an easy way to access it at a certain static and default URL.

Secondly, the software solution allows the user to easily navigate and visualize the data in order to take adequate actions if needed. The mobile application consists of multiple pages created in Android Studio that are meant to facilitate the monitoring task of the product, increasing the productivity by implying preventive and predictive maintenance tactics. The client firstly needs to register into the application using an email address and the URL of the database, after which he has full access over the dashboard. The dashboard is situated in the Home Activity and is made up of multiple elements like progress bars and graphs that are created to display in real time the measurements registered in the working environment. The Settings page permits the modification of the minimum and maximum limits at which the engine is working in normal parameters with the option of automatically sending email alerts to the address of the user that is logged in. The Profile page enables the client to change the credentials, resulting in better coverage as multiple systems can be connected with their specific URL.

Furthermore, the current implementation of the solution requires multiple improvements that would increase the functionality and applicability of the product. The most important improvement that can be done is to create a universal prototype that would work efficiently with

any kind of engine connected with an IoT solution that is specialized in monitoring and predictive maintenance and would not necessitate imposed specifications. More sensors need to be installed as the system allows a maximum of three engines to be observed in the same time, while companies usually tend to use a larger number of physical components. Another step that would elevate the system complexity would be the incorporation of a smart switch, capable of killing the power to the engines and terminate the process just with a click of a button that is present in the mobile application, resulting in a product that can be used in remote and inaccessible areas where the human intervention is not possible in time.

In conclusion, the IoT system presented provides a strong tool that can be used in preventive and predictive maintenance for any kind of machine. It not only boost the efficiency and productivity of the physical mechanism, but increase the profitability of companies in the long run and encourages the usage of modern tools and mobile solutions, that could lead to future innovation and digital evolution. The predictive maintenance brings a hidden advantage, as the improvements in engine management can bring huge environmental changes by reducing the pollution caused by emissions and old fuel consumption processes. With future improvements, the IoT systems can become a efficient utensil ment to improve the quality of life and production of goods, aiming to achieve in a relatively short period of time a futuristic sustainable world.