NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

RODRIGO ULISSES LOPES GOMES

Bachelor in Computer Science and Engineering

COMBINING DIFFERENT DATA SOURCES FOR INDUSTRIAL PROCESS MONITORING

EXPERIMENTING IN A CLASSIC CAR RESTORATION WORKSHOP

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
March, 2023

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

COMBINING DIFFERENT DATA SOURCES FOR INDUSTRIAL PROCESS MONITORING

EXPERIMENTING IN A CLASSIC CAR RESTORATION WORKSHOP

RODRIGO ULISSES LOPES GOMES

Bachelor in Computer Science and Engineering

Adviser: Vasco Miguel Moreira do Amaral
Associate Professor, NOVA University Lisbon

Co-adviser: Fernando Brito e Abreu
Associate Professor, ISCTE University Institute of Lisbon

Examination Committee:

Chair: Maria Armanda Simenta Rodrigues Grueau
Associate Professor, NOVA University Lisbon

Rapporteur: Maria Dulce Pedroso Domingos
Associate Professor, FCUL

Adviser: Vasco Miguel Moreira do Amaral
Associate Professor, NOVA University Lisbon

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
March, 2023

Combining different data sources for industrial process monitoring

Para os meus avós.

Acknowledgements

I want to start by thanking professor Fernando Brito e Abreu and professor Vasco Amaral for all the support and guidance offered during all the months of the thesis. They made a point of always meeting with me every week, and where new ideas and ways to solve the problems we found during the work development were always discussed.

To all the Raimundo Branco workshop workers, who always showed sympathy, a good disposition, and a willingness to help and cooperate in all my visits there, when tests needed to be made, a big thank you.

I would also like to thank my parents and brothers for always supporting me in good and bad times since the beginning of my student life. Even during my university years, despite being several hours away, they were always there for me, helping me with whatever I needed and always ready to take a call.

I would also like to thank my family for being so united and always ready to help me and to offer me moments of joy and animation that, although not directly, always help me to be in a better mood when challenges arise.

To Daniela, my girlfriend, a big thank you because she is always present, always cheerful, and always motivates me in everything I do, making me always want more than I wanted at the beginning.

I would also like to thank the one that will always be my university, FCT-NOVA, for its environment of so much union. I rarely met a colleague who was not ready to help and share some laughs. Where the conditions to study were always good, professors were always ready to help. There are always events to have fun with the rest of the community. It also gave me friends for life with whom I spent most of my university time and with whom I shared this great adventure full of challenges and good moments - Diogo Barreiros, Guilherme Ribeiro, Ruben Vaz, João Ginjinha, Henrique Ribeiro, Ronaldo Corte - to them also a big thank you.

Finally, thank you to my friends from my hometown, Vilar Formoso, for all the moments we spent together since the beginning of this student journey until today, when we always have fun when we return home.

Abstract

The historical importance, the aesthetics, the build quality, and the rarity are characteristic features that individually or collectively define a car as a classic. Keeping the authenticity of those masterpieces, i.e., maintaining them as close as possible to when they left the factory, requires expert restoration services. For this purpose, there are guidelines defined for classic car restoration.

Monitoring the classic car restoration process so that its evidence is recorded automatically is essential to reduce the need for a workshop supervisor (man-in-the-middle) to do this process control manually.

Therefore, this dissertation aims to detect the restoration processes performed on a classic car, in a restoration workshop, implementing energy consumption sensing, combining it with localization and other sensing techniques (vibration detection).

A state-of-the-art rapid systematic review was done so we could make well-informed decisions according to the system's requirements. In addition, the research made us know more about methodologies and technologies of Intrusive Load Monitoring (ILM) systems, Localization systems, and Internet of Things (IoT)/Industrial Internet of Things (IIoT) systems.

After an exhaustive comprehension of the problem, analysis requirements, and development, it was possible to reach an efficient and reliable process identification system. By combining location data, energy loads identified, and vibrations detection, using, along the way, Machine Learning (ML) algorithms, the system can infer some of the restoration processes each classic car goes through in the workshop (Painting, Curing, Sanding, Mineral Blasting), and the start and end timestamps of those processes. Also, a web application was developed to support the control of the existing sensors in the workshop.

Keywords: Process Activity Recognition, IoT, IIoT, Intrusive Load Monitoring Pattern Recognition, Machine Learning, Indoor Location, Classic Cars Restoration Charter of Turin, Industrial Internet of Things, Location Fingerprinting

Resumo

A importância histórica, a estética, a qualidade de construção e a raridade são características que individualmente ou em conjunto definem um automóvel como um clássico. Manter a autenticidade dessas obras-primas, ou seja, mantê-las tão próximas quanto possível de quando saíram da fábrica, requer serviços especializados de restauro. Para este efeito, existem directrizes definidas para o restauro de automóveis clássicos.

A monitorização do processo de restauro de carros clássicos, de modo que, as suas provas sejam registadas de forma automática, é essencial para reduzir a necessidade de ter um supervisor da oficina a realizar essa tarefa manualmente.

Portanto, esta dissertação visa detectar os processos de restauro realizados num automóvel clássico, numa oficina de restauro, implementando a deteção, com base em sensores, do consumo de energia, combinando-a com a localização e com outras técnicas de deteção sensorial (como a deteção de vibração nos carros).

Foi feita uma revisão de literatura estruturada do estado da arte para que pudéssemos tomar decisões bem informadas, de acordo com os requisitos do sistema. A investigação fez-nos conhecer melhor as metodologias e tecnologias dos sistemas de Monitorização Intrusiva de Cargas, sistemas de Localização e sistemas relativos à Internet das Coisas/Internet Industrial das Coisas.

Após uma compreensão exaustiva do problema, análise de requisitos e desenvolvimento, foi possível alcançar um sistema eficiente e fiável para identificação de processos de restauro. Combinando dados de localização, identificação de consumos elétricos e vibrações detetatas, tirando partido de algoritmos de aprendizagem automática, o sistema é capaz de inferir alguns dos processos de restauro pelo qual cada carro passa na oficina (Pintura, Secagem, Lixagem e Decapagem com Jato de Areia) e os horários de início e fim desses processos. Além disso, foi desenvolvida uma aplicação web para suportar o controlo dos sensores instalados na oficina.

Palavras-chave: Reconhecimento da Actividade do Processo, Internet das Coisas, IoT
IIoT, Monitorização Intrusiva de Cargas, Reconhecimento de Padrões
Aprendizagem Automática, Localização Interior, Restauro de Carros Clássicos
Charter of Turin, Internet Industrial das Coisas, Localização com Impressão Digital

Contents

Appendices

List of Figures

List of Tables

Acronyms

AC Alternating Current 57
ACP Automóvel Club de Portugal 1, 2
ADL Activities of Daily Living 15, 22, 23
AI Artificial Intelligence 14
ANN Artificial Neural Network 21, 27
AP Access Points 17
API Application Programming Interface 5, 34, 39, 40, 41, 44, 45, 48, 52, 54, 55, 57, 83, 84, 90, 91, 116, 117
ASE Automated Software Engineering 2
AWS Amazon Web Services 9, 11, 13, 34, 38, 39, 40, 46, 47, 87, 91

BLE Bluetooth Low Energy 3, 11, 17, 18, 24, 25, 41, 42, 58
BPMN Business Process Model and Notation 8

CPU Central Processing Unit 49

dbm decibel-milliwatts 24, 25, 26, 60, 61
DSR Design Science Research 1, 4
DT Decision Tree 60, 65, 77

FCT NOVA School of Science and Technology 3
FFNN Feed Forward Neural Network 14, 23, 77
FFT Fast Fourier Transform 52
FIVA Fédération Internationale des Véhicules Anciens 1, 7, 9

GB Gradient Boosting 60, 61, 65, 77
GMM Gaussian Mixture Model 21
GNB Gaussian Naive Bayes 60, 65, 77
GUI Graphical User Interface 40, 91

HMM Hidden Markov Models 21
HTTP Hypertext Transfer Protocol 46, 55, 57, 90

IIoT Industrial Internet of Things xi, 16
ILM Intrusive Load Monitoring xi, xxiii, 2, 3, 5, 7, 13, 14, 16, 19, 20, 22, 23, 30, 31, 32, 40, 74, 76, 77, 78, 80, 102, 103, 115
INCD Infraestrutura Nacional de Computação Distribúida 49
IoT Internet of Things xi, 2, 3, 5, 7, 9, 13, 14, 16, 17, 20, 22, 23, 24, 33, 35, 36, 37, 38, 39, 41, 46, 55, 116
IP Internet Protocol 57

KNN k-Nearest Neighbour 14, 22, 26, 27, 60, 65, 77

LM Load Monitoring 3, 7, 13, 30, 31
LR Logistic Regression 26
LSTM Long Short-Term Memory 22, 23

ML Machine Learning xi, xxiii, 7, 14, 15, 18, 21, 24, 26, 27, 40, 48, 49, 60, 61, 63, 64, 65, 66, 77, 78, 95
MLBPNN Multilayer Perceptron Neural Network with Backpropagation 21

NILM Non Intrusive Load Monitoring 7, 13, 14, 19, 20

OS Operating System 41

RF Random Forest 60, 65, 77
RFID Radio-frequency Identification 17
RMS Root Mean Square 21, 22, 57
RR Rapid Review 19
RSSI Received Signal Strength Indicator 11, 17, 18, 24, 25, 26, 52, 54, 61, 62, 63, 65, 66

SL Supervised Learning 14, 15, 16, 22, 23
SSH Secure Shell 49
SUS System Usability Scale 105, 108
SVM Support Vector Machine 14, 22, 23, 26, 27, 60, 65

TLX Task Load Index 104, 108, 109, 111

UL Unsupervised Learning 14, 16

CHAPTER 1

Introduction

This chapter introduces the context and description of the thesis, the institutional context underlying it, explaining its challenges, the problem, and the final objectives it aimed to achieve. Finally, we present our methodology, based on Design Science Research (DSR), while developing this work, listing the achieved contributions and the thesis structure.

1.1 Context and Description

The historical importance, the aesthetics, the build quality, and the rarity are characteristic features that individually or collectively define a car as a classic. Due to the many admirers, classic cars are highly valued, sentimentally and monetarily. Keeping the authenticity of those masterpieces, i.e., maintaining them as close as possible to when they left the factory, requires expert restoration services. Guidelines for the restoration of classic cars were proposed by the Fédération Internationale des Véhicules Anciens (FIVA)[1]. They may be used for the certification of classic cars by accredited certification bodies such as the Automóvel Club de Portugal (ACP)[2].

Automatically monitoring the classic car restore process [3] so that pieces of evidence are recorded is an important matter, both for managing the plant shop floor, for certification purposes, reducing the man-in-the-middle need, and for allowing classic car owners to follow the restore process remotely, reducing the need to travel to the workshop to talk with workshop supervisors about the processes done to the car, thus reducing the carbon footprint and also making the whole process more transparent.

[1] Fédération Internationale des Véhicules Anciens (FIVA), https://fiva.org, Last Access: 24/01/2023

[2] Automóvel Club de Portugal (ACP), https://www.acp.pt/classicos, Last Access: 24/01/2023

[3] The terms restoration process and restoration activity are used throughout the document with the same meaning.

1.2 Institutional Context

The research depicted in this work is the result of a partnership between the Automated Software Engineering (ASE) group from NOVA-LINCS [4], the Software Systems Engineering (SSE) group at ISTAR-IUL research centre [5] and the Raimundo Branco car body restore shop [6] where this system is expected to be implemented. There is also a collaboration with the Classic Cars Division of the largest automobile association in Portugal, ACP [7], and the Portuguese delegation of BASF [8], a big chemical multinational that produces specific paints for classic car restorations, namely the Glasurit brand [9].

1.3 Problem Statement and Final Goals

Our work aimed to create and implement an IoT monitoring system to recognize the tools and machines used in the workshop and infer restoration tasks (e.g., mineral blasting, bodywork restoration, painting) that a classic car is going through. We intended to use ILM techniques by installing smart energy meters in the workshop outlets and in the different booths' electric panels to use its data for detecting the various tools and booth activities resulting from its use by the workers in the restoration process. Furthermore, since the areas for each stage are roughly delimited within the plant (e.g., mineral blasting zone, bodywork restoration zones, paint booths, sanding zone), we also intended to create a localization system to locate each car to achieve an initial understanding of the ongoing stage. Moreover, we planned to use accelerometer sensors to associate the electric signals with the vehicles being worked on, aiming to automatically recognize the ongoing restoration activities carried out in each car.

Our work is part of a larger project which, at the moment, already has three master theses finalized in the sequence of the ongoing research [16, 22, 20]. Currently, the project's most important contribution is an online platform where the entire *Charter of Turin* has been modeled so that all restoration activities a car goes through in the workshop can be transparently recorded and sorted sequentially in this model. It is the supervisors of the workshop who do this registration. Furthermore, the platform allows them to add evidence of the activities carried out with photos and descriptions. Finally, it will enable car owners to obtain a document of their car with all this evidence of the restoration. For the moment, the use of the platform is a manual process that implies that the supervisor has to go around the workshop, checking which restoration processes are being carried out on each car, to then register on the platform. This way, he has to suspend the tasks he was doing to do that verification. As this problem exists, to increase

[4] https://nova-lincs.di.fct.unl.pt Last accessed on: 24/01/2023
[5] https://ciencia.iscte-iul.pt/centres/istar-iul/groups/sse Last accessed on: 24/01/2023
[6] https://www.facebook.com/raimundo.joaquim.branco Last accessed on: 24/01/2023
[7] https://www.acp.pt/classicos Last accessed on: 24/01/2023
[8] https://www.basf.com/pt/pt.html Last accessed on:24/01/2023
[9] https://www.glasurit.com/uk/paints-classic-cars Last accessed on: 24/01/2023

efficiency and reduce errors, it is desirable to automate the monitoring of the restoration processes performed in the workshop to eliminate the need for the man-in-the-middle.

To increase efficiency and reduce errors introduced by the intervention of the man-in-the-middle it is desirable to automate the restoration process.

So this thesis starts specifically in the follow-up of another master thesis [22] elaborated at NOVA School of Science and Technology (FCT) and concluded in early 2022, where the study and implementation of an IoT system in the same workshop was made, which consists in the detection of restoration stages carried out in each car. The detection was performed through a sensor box prototype containing humidity, temperature, and accelerometer sensors complemented with its location based on Bluetooth Low Energy (BLE) beacons. Although this work presented significant progress, it was in the prototype phase. The system lacked precision in detecting the operation of most tools, and specificity about different activities identification was low. Also, the indoor localization solution was only tested in simulated tests outside the workshop building. A control hub web application was implemented but relied on pay-per-use platforms.

The main objectives of this thesis are (i) to evolve the initial prototype into a usable product by the workshop workers on normal working days, (ii) to improve the accuracy and reliability of monitoring the restoration processes, and (iii) to reduce the need for the man-in-the-middle.

Based on the objectives of the present work, we aim to answer the following research question:

How can we implement energy consumption and location sensing by combining it with other sensing techniques (such as vibration) to detect the type of restoration activity performed on a classic car?

Therefore we can define several intermediate objectives or research steps, which together help us reach our answer:

- Study the different possibilities related to Load Monitoring (LM), especially ILM, and Localization implementations in the literature;
- Model and implement an ILM system capable of inferring electric tools usage and booths activities in the workshop;
- Model and implement a Localization system to detect the cars' location and zone in the workshop;
- Model and implement an algorithm capable of identifying the restoration process a car goes through in the workshop based on all sensors data, i.e., energy sensors, location, and accelerometers;
- Model and implement a no-cost platform to host the already existent sensors control Web Application;

- Expand the sensors control Web Application to support the control of the new sensors and add more features to make it more complete and valuable;

- Testing and validation.

1.4 Challenges

Classic car restoration is a complex process with many stages [11]. Many classic cars may be under restoration in a workshop floor plan, often each at a different restoration stage. There are restoration stages where different tools are used to do the process. These tools may have similar electrical powers, making tool differentiation more difficult. Also, in the workshop's existing booths, like painting and mineral blast booths, different activities are done in the same booth, like applying other painting products to a car or curing the paint (in the paint booth). Activities may have similar energy consumption.

Many classic cars may be under restoration, so the distance between vehicles on the plant shop floor is reduced. In addition, combining it with the several physical obstacles in the floor plan results in a challenging endeavour for the localization sensors' precision.

Various sensors monitor the processes, generating a large amount of data. Most of the times a given activity can only be identified by combining multiple inputs from heterogeneous sensors. So, several algorithms and computations over that data can be performed simultaneously, requiring high computational demands. Therefore, it is of great importance to choose where the data should be stored (i.e., whether locally, in a remote server, in the cloud, and in the latter case, on which cloud platform) and where the computations should be performed (i.e., also in a cloud platform, locally or in a remote server).

1.5 Scientific Methodology

This section will present our work using the DSR methodology we intend to follow.

DSR [32], is a methodology that defines a set of perspectives, rules, and steps to design and create artefacts to solve a well-defined problem and increase the existing knowledge in the corresponding area of research with the results obtained at the end of the process. The setting up of our DSR goes as follows:

Awareness of the problem: The restoration workshop, where this work was applied, is being automated to automatically infer and register all the stages a car has passed through during its time in the workshop. In [22], there was a first attempt to achieve that goal. Detecting several tools in the workshop plan was, however, limited. In addition, we identified various limitations in the sensor box developed and in the localization system precision. A web application to control the sensors used was also developed but was dependent on pay-per-use platforms, which is disadvantageous.

Suggestion: To solve this problem, we proposed to study different ways of identifying electronic devices and booth activities through smart plugs and electric panel meters, respectively, as well as different approaches to obtaining high precision in an indoor localization. Moreover, to combine it with the previous works sensors. We proposed to install electric sensors (smart plugs and electric panel meters) in the workshop and implement a new localization system, testing different setups, improving precision and adding reliability to the process identification. For the web application, we proposed to create and develop a dedicated backend containing a local database and an Application Programming Interface (API). We also proposed expanding the application to support the new sensors and make it more complete.

Development: The development phase started by processing the data sent by the selected sensors. Then the localization system created was used to locate the cars in the workshop, and the ILM approach was used to identify the usage of the electric tools and different activities being carried out in the booths. Using the last two pieces of information, in combination with the vibrations detected by the accelerometer sensor, already in use in the [22] work, the restoration activities carried out in each car should be inferred. We developed a local database for the web application and created an API to integrate the front end and the database. Finally, we used an available remote server to host the last two modules and the user interface, the front end.

Evaluation: To evaluate our solution, we needed to test the system in the workshop during normal working days, adding the sensors to the working routine of the workshop. Testing with cars passing throw the workshop's different restoration stages, activities, and zones. For the web application, we did some questionnaires to its users. Finally, we validated the sensor box intrusiveness by interviewing the workshop workers.

1.6 Key Contributions

Through the development of this work and the achievement of the proposed objectives, we reached the following contributions:

- Advances related to accuracy and specificity, compared to the previous work, in the restoration processes detection;
- An IoT ecosystem capable of supporting and monitoring all temporal data generated by sensors;
- A localization module to locate the cars all over the workshop;
- A load monitoring module to detect and identify different electric activities in the workshop;
- A reliable and precise system capable of detecting the restoration processes of a classic car in the restoration workshop

- A sensors control web application with no associated costs to support the control and monitoring of all sensors.

1.7 Thesis Structure

This thesis is further structured as follows:

- **Chapter 1 - Introduction:** This chapter presents the problem, the motivations for its resolution, and the objectives to reach the intended solution.
- **Chapter 2 - Background:** In this chapter, a description of the project where this work is inserted and the concepts related to this work are presented and explained, allowing a better understanding of the problem and solutions presented.
- **Chapter 3 - Related Work:** This chapter will be presented previously developed projects with similar objectives.
- **Chapter 4 - Conceptualization and Implementation:** This chapter details and presents an extended description of the architecture and implementation of our work.
- **Chapter 5 - Tests and Validation:** This chapter addresses the different tests, results, and corresponding interpretations related to validating the developed work.
- **Chapter 6 - Conclusions:** In this chapter, we present the conclusions obtained with the development of this work and proposals for future work.

CHAPTER 2

Background

This chapter starts by describing the project where this work is inserted (section 2.1). Following that, it gives an insight into essential topics for this work, such as Load Monitoring (LM) techniques comparing Intrusive Load Monitoring (ILM) with Non Intrusive Load Monitoring (NILM) (section 2.2), followed by activities inferences based on the previous topics (section 2.3). Then, we do a short introduction to Machine Learning (ML) algorithms (section 2.4). The concept of the Internet of Things (IoT) and its architectures is presented (section 2.5), and an introduction to indoor localization approaches are addressed (section 2.6). The chapter is finished with a brief conclusion on the topics described (section 2.7).

2.1 The Classic Car Restoration Workshop Digitization Project

This work is part of a project being carried out in the Raimundo Branco restoration workshop [1], to digitize the entire record of the restoration processes carried out there.

As described in the previous chapter, the historical importance, the aesthetics, the build quality, and the rarity make classic cars highly valued. So, keeping those masterpieces' authenticity requires expert knowledge and expert restoration services. Thus, to help the restoration workers and to create a certificate of excellence for the restoration process, Fédération Internationale des Véhicules Anciens (FIVA) created a guidance book named *"Charter of Turin Handbook"* [11] (we will be calling this document for simplification reasons as *Charter of Turin*). That is a book that the restoration workshops of excellence follow so the cars that come out of there have the best possible restoration and authenticity, making them more valuable.

Before the beginning of this project, the workshop supervisors needed to register manually every process done to the cars, and the conformity with the *Charter of Turin* was being made by checking the handbook. Furthermore, the communication of the state of

[1] https://www.facebook.com/raimundo.joaquim.branco Last accessed on: 04/01/2023

restoration with the car owners was made through photos and isolated messages, making the process sometimes confusing and the perception of the state of restoration unclear and not transparent.

This way, in previous researches done presented in [16] and then for a production state evolution in [20], a platform was created containing a *Charter of Turin* based Business Process Model and Notation (BPMN) process model (it can be seen in [2]). This thesis will reference this work as *Charter of Turin Monitor*. In this model, the steps defined in the *Charter of Turin* are modeled sequentially and can be marked as done when each step is completed (Figure 2.1). Furthermore, the platform allows adding as many cars as needed, so the steps concluded and the ones left can be visible. Also, the platform enables the workshop supervisors to add evidence to every restoration step finished, like photos, videos, stage start and end date, and descriptions (Figure 2.2).

Classic car owners have access to this platform, so they can follow the restoration of their cars and see evidence of the work being done, all in a more organized and transparent way. In the end, it allows the generation of a *PDF* file that summarizes the entire restoration process done to a car, thus providing the owner with a document that can serve as proof of the quality of restoration and be used for certification purposes (Figure 2.3).

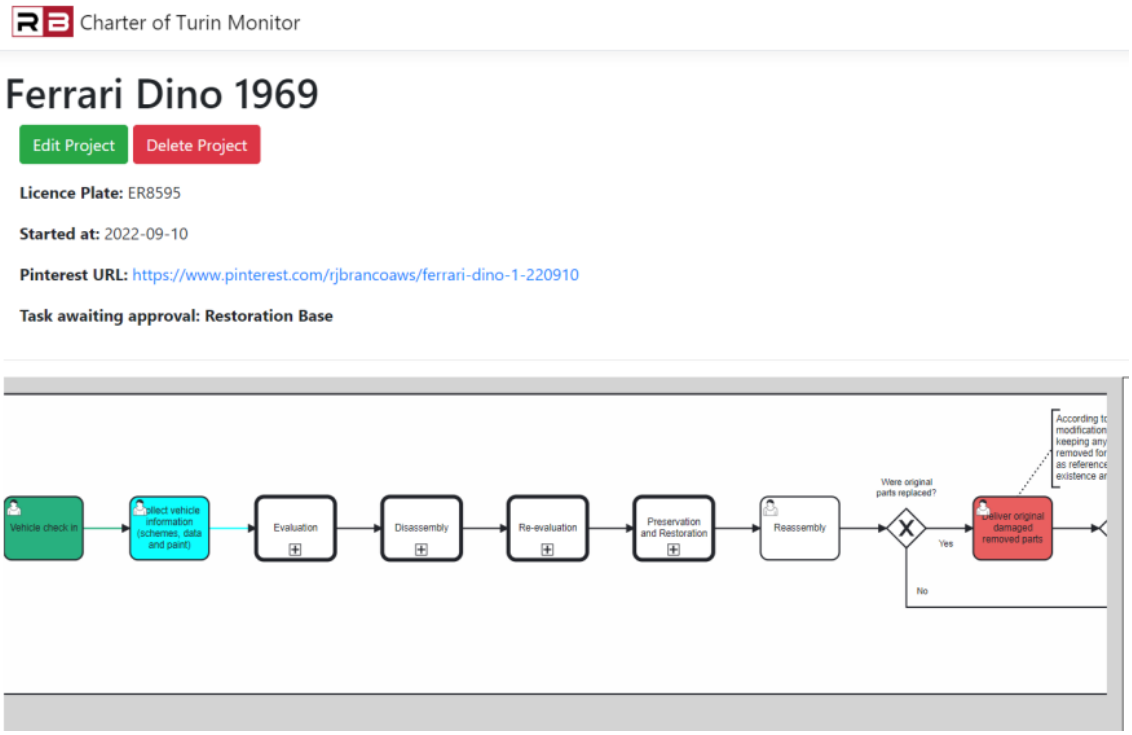

Figure 2.1: *Charter of Turin Monitor* - The workshop supervisors view where the finished restoration processes can be registered. Retrieved from [20].

The platform is already in use in Raimundo Branco and obtained good feedback from

[2] http://194.210.120.34:5000/diagram Accessed on: 03/01/2023

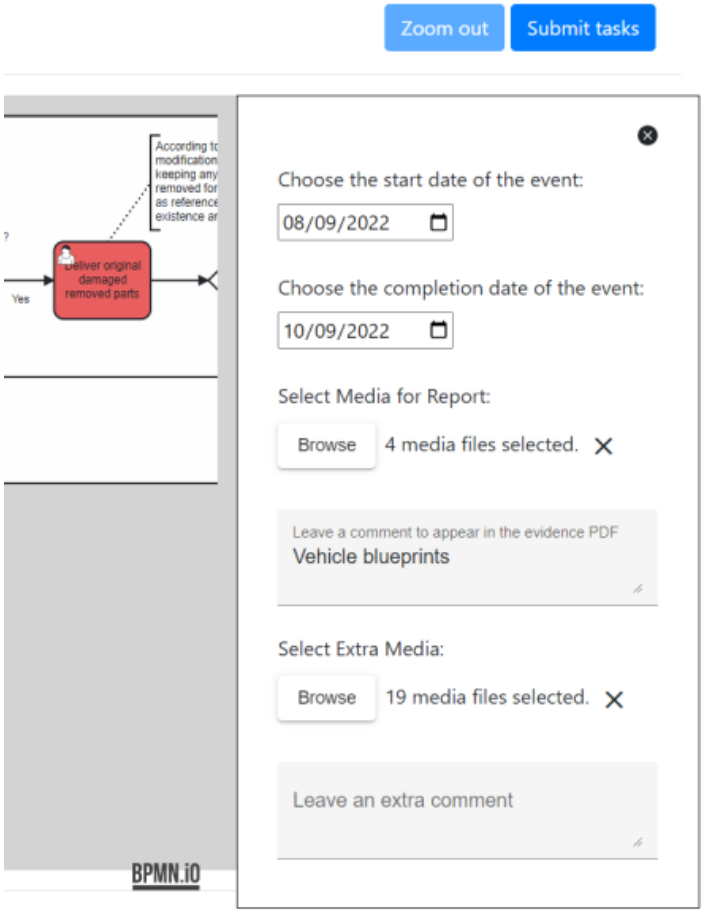

Figure 2.2: *Charter of Turin Monitor* - The workshop manager view where the evidence of the finished restoration processes is saved. Retrieved from [20].

national entities like Museu do Caramulo [3] and FIVA internationally, as a platform to be expanded to other classic cars workshops.

In this context, there is the need for the workshop supervisor responsible for updating the platform to check, by looking at the workshop space, at what stage of restoration each car is, which requires time and attention due to the high number of vehicles that may be present in the workshop and different possible activities being done. The need for a man-in-the-middle would be eliminated if this update could be done automatically.

Following this issue, a prototype was developed and tested in [22] that we will describe next, as this was the starting point for the realization of this dissertation.

That work aimed to infer the restoration processes in the Raimundo Branco workshop using temperature, humidity sensors, an accelerometer, and the car's location. An IoT system was implemented using multiple Amazon Web Services (AWS) services and sensors. A sensor box prototype constituted by humidity, temperature, and accelerometer sensors connected to a Raspberry Pi was created (Figure 2.4). The sensor box needed to be in the

[3] https://museudocaramulo.pt/ Last accessed on: 04/01/2023

3.4 Disassembly

3.4.1 Disassemble part

Desmontagem dos painéis laterais

12

Figure 2.3: *Charter of Turin Monitor* - A page of the *PDF* generated for a car. Retrieved from [20].

car body (a magnet was used). The accelerometer was used to detect vibrations in the car panels. The generated frequency was calculated and matched with the tools' frequency ranges registered previously in an AWS database. The temperature and humidity sensors were used to detect if the box was in the inside or outside part of the building and if the painting process started in the paint booths as the temperature increased in that case.

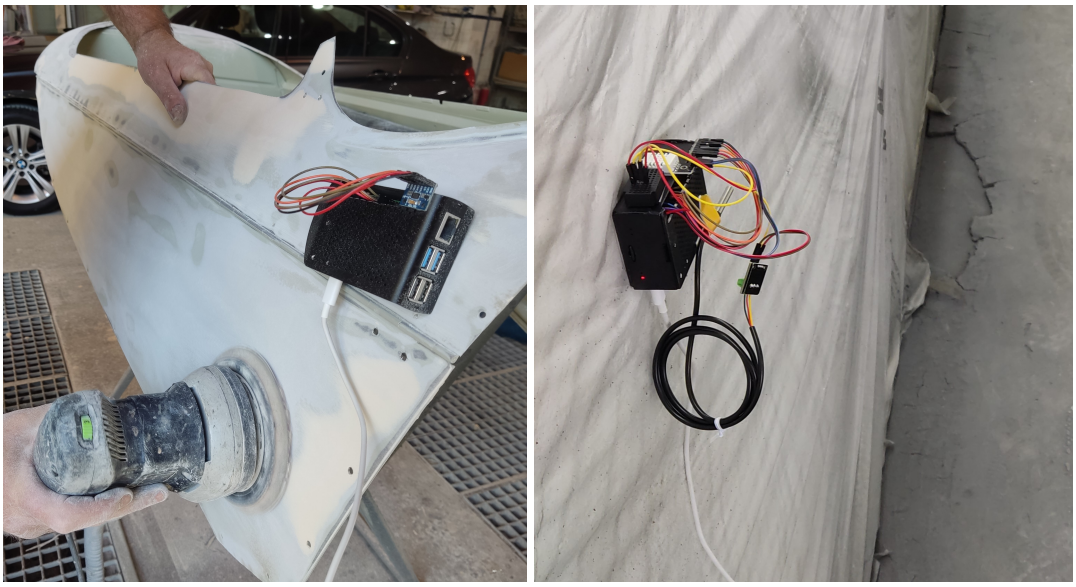

Figure 2.4: Sensor Box of the previous work. The box in the left image only has the accelerometer connected for testing purposes. Retrieved from [22].

A localization component was modeled and tested to locate the sensor box to complement this information. As there are activities in the workshop that are carried out just in some places of the building (Figure 2.5), the stage recognition can be more effectively complemented with the location data.

The Bluetooth Low Energy (BLE) Proximity technique was used. Simulation tests were done using *Estimote* BLE beacons distributed at strategic points in an old workshop building. The sensor box stored the most recent beacons Received Signal Strength Indicator (RSSI) values. After that, the nearest beacon is found based on the RSSI values, and the area/zone where the sensor box is can be inferred. The localization module developed was only tested in the previous workshop building, and isolated from the entire system, to verify if this technique could be helpful.

An algorithm was created to interpret and deal with the raw data sent by the sensor box so the tools and stages could be identified.

A web application was also created to monitor and control all the sensor boxes and their sensors, allowing the workshop supervisors to add new tools frequencies ranges and their configurations, check problematic sensors or beacons by system logs and show the output tools that were used in each car and the correspondent stages.

The work achieved significant results, but as a prototype, only simulated and controlled tests were done. The main conclusions reached with those tests conducted in

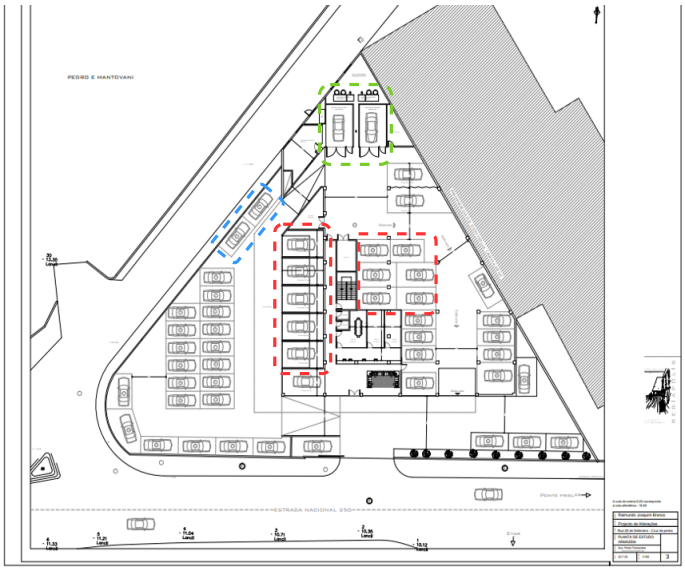

Figure 2.5: The first defined layout of the workshop (outdated). Paint booths are marked in green, the mineral blasting zone in blue, and the bodywork restoration zone in red. Retrieved from [22].

the workshop were that detecting tools with low produced frequency, like manual tools or polishers, was not proved effective and presented low accuracy values, unlike tools such as electric sanders that produce high vibrations and, therefore, good results. The localization method, as mentioned, was not tested in the actual building and only proved that good results can be achieved with the proximity technique. The web application was validated by workshop workers and received good feedback proving to be helpful.

2.1.1 Previous Work Flaws

In our work, we wanted to evolve the prototype to be able to work all day long in the workshop, outside a controlled environment, and without needing a system expert in the workshop. Furthermore, a system can detect a more extensive range of restoration activities. Therefore, it was necessary to reflect on the previous work to understand what we should keep, evolve, remove, and add so that it can work in the conditions described.

This way, some conclusions about the previous works and their flaws were reached:

- To recognize tools based on their minimum and maximum frequency is a difficult task and presents low accuracy with most of the tools, as some tools produce low vibrations, and the frequency detected varies with the proximity of the sensor to the spot where the tool is acting on the car panel. The effect of this proximity implies high care on the part of the worker in keeping the box always close to the working spot, which can easily be too intrusive.
- The electric sander was the only tool detected by the frequency that presented positive test results.

- Every time the workshop acquires a new device, the tool's minimum and maximum frequency must be measured and registered.
- The localization technique implemented was not tested in the new building. Although the localization proved relevant to the process identification, it was not designed for the new building and was not integrated with the whole system.
- The mineral blast was supposed to be detected only by location. So, the start and end times of the process would not be precise, as a car inside the booth or close to it does not mean it is being worked.
- The temperature and humidity sensors were used to detect the box inside the paint booths. So the sensor box should be inside the booth, in contact with the high temperatures reached there, and large paint powders generated, deteriorating the material over time.
- There was no differentiation between the activities carried out inside the paint booths (paint application/anti-rust application/curing). The entire process was identified as a whole, with the beginning and end temperature heating inside the cabin.
- The web application was built over pay-per-use platforms. The backend was built over AWS, and the frontend hosted in *Netlify*. This pay-per-use platform usage would be all right if the application needed a lot of memory and computation resources. However, this does not happen, which may bring unnecessary costs.

These flaws revealed that new sensing ways should be studied, some sensors and components of the previous work would be eliminated, and others adapted to build a production product. These differences and upgrades will be covered in this document.

2.2 Load Monitoring Techniques

The price of electricity, the interest in saving energy resources, and, generally, the goal to reduce the carbon footprint, in cooperation with the evolution of studies and developments on the subject of the IoT, motivates the use of technologies to monitor and control the energy costs of houses, factories, buildings or workshops helping the habitants to control their energy costs.

The LM systems emerge due to this problem. More precisely because this monitoring and control can be much more successful if it is done at the level of the electrical appliances and electronics used, allowing the habitants to understand which of these lead the way in energy cost and thus be able to take the necessary actions [9].

The LM systems are divided into two different topics, the ILM and the NILM. The characteristics, advantages, and disadvantages will be presented next.

NILM or single point sensing is the technique that presents a higher number of studies in the last years. All devices are monitored in a non-intrusive way, i.e., the system obtains the aggregated electrical energy data of all devices through a single meter, and algorithms are then applied to disaggregate this data, returning as an output the individual energy data of each device that was detected and identified [5]. This approach has a lower cost as only one or a few sensors are needed, and it is also beneficial because there is no need to install individual sensors on every device that one may want to monitor. On the other hand, even with powerful Artificial Intelligence (AI) algorithms, it is hard to classify and differentiate with high precision a vast number of different devices activated simultaneously or devices with low consumption profiles [21].

On the other hand, in ILM or distributed sensing, installing one meter per device is necessary so it can become relatively expensive. Another disadvantage is the requirement to install a new meter when a new device connected to a new outlet needs to be monitored. For this, human requisition will be needed. However, the efficiency and precision are higher. Furthermore, the computation required for device recognition is less complex, as the purpose is to recognize the device connected to the sensor. Also, with the evolution of technologies, IoT sensors are getting more affordable, becoming ILM a better alternative [9], especially in complex environments.

As this project aims to implement a system in a car restoration workshop with various processes happening simultaneously, with multiple tools working in parallel, we chose to apply ILM to make tool identification.

So, the method of ILM is divided into the following phases:

1. Energy data collection;
2. Feature extraction;
3. Classification.

Energy data collection consists of collecting a device's electrical load by sensors or smart meters. It may be current, voltage, or power data [1]. The sampling frequency of the data collected by the meters will influence the amount of data that will need to be managed and processed in the following phases [21], and in general higher sampling frequency gives better classification results.

The phase of features extraction is about acquiring relevant statistics and characteristics from the data collected by each sensor (i.e., mean power value, maximum peak, number of values above the mean) [9]. Then the selected features are computed, and a feature vector is built to serve as input to the classification algorithm. Data preprocessing is done at the beginning for feature normalization in some works.

The classification phase takes as input the feature vector and through a ML algorithm, using Supervised Learning (SL) or Unsupervised Learning (UL), i.e., k-Nearest Neighbour (KNN), Feed Forward Neural Network (FFNN), Support Vector Machine (SVM), classifies the load, associating it to a device.

In section 3.2, the methods used by state-of-the-art studies in each step will be explicitly discussed.

There are four different types of electronic devices defined in the literature [23], [9], some being more difficult to detect than others:

- Type 1 - On/Off: Represent the devices with only two energy states, On or Off.
- Type 2 - Finite State: Represent the devices with multiple power energy states, like a sanding machine with different power levels.
- Type 3 - Continuously Variable: Correspond to devices with variable energy consumption like laptops or smartphones. It is the most complex type to recognize since it provides more information and has very different load patterns.
- Type 4 - Permanently On: This could be a sub-type of type 1 devices. Relates to devices that always work like alarms or *Wifi* routers.

In our workshop context, all the electric tools used are of types 1 and 2.

2.3 Activities Inference

Due to the potential of detecting electronic devices from energy data, there are some researches and developments, [5], [9] which will be discussed further in Chapter 3 that complement this process with the inference of activities performed by the habitants of a house, Activities of Daily Living (ADL). The inference of activities can also be applied in workshops and business buildings. The fact that one can infer activities from the knowledge of the tools used is beneficial for our work since our primary goal is to detect which activity of restoring the car passed. By obtaining the tools or some of the tools used, the times at which they were activated, and combining them with the location, it should be possible to infer the corresponding stage.

2.4 Machine Learning algorithms

This section will present a brief overview of how ML algorithms work, as this is an essential part of the device recognition process.

ML algorithms are divided into two main categories: supervised and unsupervised learning. In ML algorithms, a training set to train the model is defined from the complete data set. And then, a test set different from the training one is used to get the classification results.

In SL, the possible output classification classes are known, and the training set is labelled manually with the correct output. In the learning phase, the algorithm is trained with this labelled data, finding relations between the input and the labelled output. Then

when new input data is given to the algorithm, it will predict the classification labels based on what it learned before with the training set [18].

On the other hand, in UL, there is no labelled data in the training set, and there is no knowledge about the classes that should be predicted, so given the input data, the algorithm will find relations and structures between data features and present it in the output [18].

SL has the disadvantage of the required human hand to label the training set, but due to its characteristics, in most cases presents more accuracy than UL.

2.5 IoT Systems

As mentioned before and explained with more detail in section 2.1, in [22] work was created an IoT system to support all the work structure. As we expand that system, it is essential to overview the IoT systems and how our work can be included in it.

IoT is a concept that consists in the connection of objects to the internet, "the key idea behind the IoT concept is to deploy billions or even trillions of smart objects capable of sensing the surrounding environment, transmitting and processing acquired data, and then feedback to the environment" [28]. Using IoT will result in a more efficient and sustainable industry and society. Allowing enterprises to monitor every machine thoroughly and turn it into the most efficient as possible, allowing the existence of smart cities and the potential to have feedback on everything that may be needed on the internet. Also, it could be positive for people's lives as one could control and monitor every device in the home, and it would be good, i.e., for comfort, health, and money.

Because of the involvement growing in IoT in many subjects, a sub-domain was created named Industrial Internet of Things (IIoT). This domain relates to the IoT applied to industries where, as previously mentioned, it is possible to make production more efficient and sustainable [28]. It connects all machines and collects their data, allowing statistics creation and analysis to optimize all the production lines. This subject of "smart factories" brought up another definition named Industry 4.0, related to the fourth industrial revolution and aims "to achieve a higher level of operational efficiency and productivity, as well as a higher level of automation" [17]. As our work is implemented in a car restoration workshop domain, it is inserted in the subdomain of IIoT and Industry 4.0.

Relying on IoT architectures, they need extensibility, scalability, modularity, and interoperability and are divided into layers. There are different types of proposes in the literature: a proposed architecture has four layers divided into the sensing layer, the networking layer, the service layer, and the interface layer [33]. Moreover, there is also a proposal with three layers with a "perception layer (or sensing layer), network layer, and service layer (or application layer)" [6], among others. Related to ILM systems like ours, P.Franco et al. [9] proposed an IoT based approach for ILM, that will be presented with more detail in section 3.2, where the IoT architecture was defined in five layers, from

bottom to the top: Physical Things, Perception, Communication Network, Middleware, and Application.

2.6 Indoor Localization Systems

In IoT systems, the localization of a device is very helpful for its tracking and monitoring. As for the sensors, an optimal localization system would be a low-cost, accurate, and efficient one. However, many possible approaches are implemented and described in the literature. Depending on the type of localization that one may want to track, as an indoor localization or outdoor, the obstacles present in the space and the allowed error margin will influence the options available for localization systems [10].

We must track devices inside the workshop, focusing only on indoor localization systems.

For indoor localization, there are Radio-frequency Identification (RFID) options. An option consists of a radio-emitting tag with a small chip, an antenna, and a reader that captures the signal. There are approaches where the reader is linked to the object to track, many tags are disposed of through the space [27], and there are approaches where the tag is linked to the moving object, and the reader is fixed. Then the distances and position of the moving object are calculated based on the RSSI value [10]. This technique is economical and efficient.

It is also possible to locate devices with cameras using image processing algorithms. Still, the workers may not like this approach as the camera must constantly record, which may cause discomfort.

Wifi-based localization relies on *Wifi* detection and the available Access Points (AP)s. Some of these techniques also use RSSI, measuring the power present in the received signal.

Approaches are detecting the closest AP and also with triangulation techniques based on the signals received in all AP [13]. The precision will be influenced by the *Wifi* AP. The *Wifi* network should cover every area in the building for good precision.

One important approach is the BLE technology that works similarly to *Wifi* localization but uses Bluetooth transmitters named beacons. It has a low cost, low energy consumption, and is powered by batteries, making them easier to install even in less accessible places. In addition, beacons offer several possible settings, the most important being the control of the Transmit Power, increasing or decreasing the beacon coverage area, and controlling the Signal Propagation Frequency, which will increase or decrease the number of measurements generated. In general, the primary implementation with beacons consists in distributing the beacons across the building and having the object locate collecting the Bluetooth signals. Then, the location is calculated with techniques using the RSSI value.

There are various techniques related to locating devices using the BLE beacons technology with its RSSI values [2], [14]. Moreover, as these devices were used in previous

work [22], studying these techniques is of great interest.

There is the Trilateration technique where the received beacons RSSI values are converted in distances, then using at least the distance to three beacons, a trilateration algorithm is used so the receiver position can be inferred. This technique, although simple, has some flaws because the conversion from RSSI to distance is based on the average signal strength coming from the beacons. However, the signal can be influenced by many external factors, such as people's movement and the area's physics.

Another simple solution is the Proximity technique. In this solution, the device is located based on its proximity to the BLE beacons available. Beacons are placed over the area, and the device location will correspond to the closest beacon location. The closest beacon has the highest RSSI value. With this solution, the exact position of the device to locate can not be found, only the relative position.

The Fingerprinting technique is a method that encompasses two phases. First, there is a training phase in which RSSI samples are captured throughout the entire area, and the corresponding location coordinates (forming a fingerprint of the area) are used to train a ML location estimation model. Second, when a target moves around the area, given the RSSI values detected, location estimates are produced by the ML model (a pair of x, y coordinates). Minor average errors of less than 1 meter [29] can be obtained. This method, as it also uses BLE beacons, can also be negatively influenced by the area conditions. Mainly in the first phase, as it may generate a set of inconsistent fingerprints.

For our context, the fingerprinting approach is very appealing because of the low cost of the beacons, the large size of our space, and the goal of identifying the exact position of the device. As described later in this document subsection 4.4.2, this will be one of the main localization methods used, in combination with the proximity method.

2.7 Summary on Background

This chapter allowed us to understand better the project where the work is inserted and the existent topics related to the objectives of this work, familiarising us with the different concepts related to each of the areas described and the different implementation techniques used in the different areas.

The next chapter will study previously developed projects with objectives similar to ours. This way, we could make better decisions when implementing our system.

CHAPTER 3

Related Work

This chapter will introduce systems and studies with objectives in common with our work. section 3.1, has a description with the methodology and keywords used for the research. In section 3.2, studies and works related to ILM are presented, followed by studies related to fingerprinting solutions, section 3.3. The corresponding research is discussed at the end of each section.

3.1 Research Methodology

The literature review provided us foundation of knowledge on the topics and was done to develop and implement an efficient system. The Rapid Review (RR) or rapid systematic review methodology [30] was followed.

The literature review was entirely done using Scopus[1] database.

Firstly a search in the NILM area was done, but soon we concluded that it would not be a good approach for our work because of the reasons mentioned before in the Background chapter (section 2.2).

So for this literature review, two search strings were used:

- ”intrusive load monitoring” AND (”activity recognition” OR ”classification activity” OR ”activity identification” OR ”Pattern recognition” OR ”Appliance Identification” OR ”Appliance signature”)
- ”IoT” AND (”Sensor networks” OR ”sensors”) AND (”device location” OR ”localization” OR ”indoor location” OR ”indoor localization”)

The first was related to the device recognition with ILM techniques. The second was used to obtain more overall knowledge on the localization systems topic introduced in the Background chapter (section 2.6).

[1] Scopus, https://www.scopus.com/, Last Access: 11/02/2022

The snowball approach was used during the research as a complementary approach to understand the topics thoroughly.

Besides, some inclusion criteria were defined to help the filtration of documents. Only documents written in English, published after 2010, and related to the proposed objectives were included. Relative to ILM, extra priority was given to documents addressing IoT systems and documents describing ILM systems complemented with activities inference methods.

For selecting the articles, the procedure sequentially followed was the following:

1. Execute the search string;
2. Apply the inclusion criteria to the article title;
3. Apply the inclusion criteria to the article abstract;
4. Apply the inclusion criteria to the article's full text;
5. Apply snowballing to the included articles.

Step 4 was only done after the preceding steps were done for all articles resulting from the search string.

For a better understanding of topics related to our work and mentioned in the Background (chapter 2), some research with the keywords "IoT", "Machine Learning", "Supervised and Unsupervised learning" was made.

When we understood that the "Location Fingerprinting" technique was a possible optimal localization solution for our work, we also decided to conduct a more in-depth search on this topic.

With the research done, it was possible to state that the studies in NILM solutions are in further growth compared with ILM solutions. The cause should be related to the search for more "green" houses and the non-intrusive solutions' advantages as they are more affordable than the intrusive solutions.

The indoor localization system studies, and more precisely, the fingerprinting technique, have shown to be an area in expansion, presenting many highly cited articles in recent years.

3.2 Intrusive Load Monitoring Approaches

This section will present the methods used by the state-of-the-art studies and implementations about ILM. As mentioned in the section 2.2, the three ILM phases are data collection, features extraction, and classification, so special attention will be given to how each one of the related works deals with each phase. Moreover, approaches to activity recognition and ILM inserted in IoT systems will also be detailed.

Almost every solution implemented in the related works was applied to smart home environments for appliance recognition and to monitor it, reducing the energy waste in habitation by looking for the appliances with the highest energy consumption and alerting the home occupants about that. However, this approach could also be used in industries and factory environments [19] like our workshop.

In [21] was presented an Artificial Neural Network (ANN) approach using low-frequency data. For data collection, smart plugs were installed for each appliance. Then, pre-processing is done where 100 power samples of a signal different from the standby signal are captured (in Watts). The data collection is only done every 2 minutes to obtain low-frequency data. After the feature extraction is done, processing the 100 samples to acquire relevant characteristics of the power trace is done. The features extracted are as follows: Maximum power value, Minimum nonzero power value, Number of samples equal to zero, Number of samples less than or equal to 30 W, Number of samples between 30 and 400 W, Number of samples between 400 and 1000 W, Number of samples greater than 1000 W, Number of transitions greater than 1000 W, Number of transitions between 10 and 100 W and Medium value of the nonzero power samples. These features extracted serve as input to the classification algorithm, this being an ANN model, more precisely, the supervised algorithm Multilayer Perceptron Neural Network with Backpropagation (MLBPNN). The experimental tests used data from three houses and the same appliances. Results showed a positive overall accuracy of 95%. A second test was done using the ANN trained with the previous data and tested with data from a new house, but worse results emerged even after reducing the number of features. It is also mentioned that better recognition results are achieved with increasing neurons in the hidden layers of the ANN.

The work presented in [25] intended to recognize appliances and also identify the different states of each one. The data used was from the ACS-F2 database [24] containing electrical signatures of numerous appliances. The features presented were the real power (W), reactive power (var), network frequency (Hz), Root Mean Square (RMS) current (A), RMS voltage (V), and phase of voltage relative to the current. A pre-processing using *z-normalization* was used for feature normalization. An important role was given to feature selection with an entropy methodology and classification results using Gaussian Mixture Model (GMM), a ML algorithm. The classification process was done with Hidden Markov Models (HMM) algorithm, as the goal is to recognize different states, and HMM is a state-based ML algorithm. Positive results were achieved, with an accuracy of 94% for the test with appliances seen by the classifier and 74% for the test with unseen appliances. This work also built a real-time application to visualize the appliance recognition characteristics. In [26], the authors proposed a continuation of this work where a User Interaction Layer was presented. It takes as input the recognized appliance and its sequence of states and then detects and outputs the human interaction events.

The implementation described in [3] presents a built-in hardware system with smart

meters and a Raspberry Pi. The Raspberry Pi is used as the processing unit where the computations related to the three phases of ILM are done, and then, the result is sent to the server. Initially, a pre-processing normalization is done to the values. Then as the Raspberry Pi "does not come with a built-in analogue-to-digital converter, a high-precision Analog-to-Digital Converter is used" to convert the analogue voltage and current signal to digital signal. The features used were: real power (P), peak current (I_{PEAK}), RMS current(I_{RMS}), and the power factor (PF). The classifier was the SL algorithm KNN that works by "comparing the most similar piece of data to the training set and returns the class label". The tests were also done with the known and unknown set of appliances, and overall accuracy of 95% was reached with some not recognized appliances in the second case.

In [31], features related to consumption are identified and explained, i.e., Daily Power Consumption, Max Power, Power Deviation, also time-related, i.e., On-Off time, Active time, Average active time, and location features like Place of use and Sequence of usage location, for testing purposes, a public dataset [2] containing a large number of identified devices energy load with time information was used, and several SL algorithms were used and compared. Random Forest, Bagging, LogitBoost, Decision Tree, Naive Bayes, and SVM all presented accuracy values higher than 90%, but the one with the highest accuracy was Random Forest, around 96.5%. No unknown appliances data set was given to the test set. Some "different traces of different models of certain device categories" available in the database were placed separately in the test or training set. In the end, a feature ranking based on the level of importance is established based on the Random Forest classification results.

In [19] is presented a handmade low-cost prototype for collecting load data, with an Arduino and an energy sensor. This work measures electrical current (A) to keep the prototype cost low. Moreover, a real-time classification process was implemented. For classification, also different algorithms were tested, i.e., KNN, Decision Trees, Random Forests, and the neural network Long Short-Term Memory (LSTM). Random Forests was again the best algorithm. An experimental test was also conducted to obtain the best sliding window size. This window corresponds to the amount of data that should go to the classification algorithm each time.

It is explained in [9] how to do ADL recognition with the output of ILM classification, meaning with the appliances used. The ADL concept has greatly focused on remote healthcare applications. This work also presented and described an IoT system for the ILM implementation. It divides this system into five layers from bottom to top: Physical Things related to the appliances layer; Perception related to data acquisition using sensors; Communication Network which enables integration and communication between devices, e.g., Bluetooth, *Wifi* or *Zigbee*; Middleware mediating the data exchange between IoT devices and the software application, cloud computing is usually used; Application

[2] ODbL, https://opendatacommons.org/licenses/odbl/1-0/, Last Access: 11/02/2022

layer, concerning user services and interfaces. For the process implementation, the energy load data used was from UK-DALE [3] dataset. The features used were the same as described before in [21], and then three different SL algorithms were tested for classification, i.e., FFNN, LSTM and SVM. FFNN obtained the best accuracy results (90%) and was used in the activities recognition phase. When testing with unseen data, the results were worse. An algorithm that maps each ADL based on some criteria related to appliance usage like energy and switched On timestamps is presented for activity classification.

The work presented in [8] is a continuation of the previous work, where the authors defended that the feature extraction of this previous work had a problem. It limited the system implementation in practical scenarios. Also, "An appliance recognition framework with a graphical interface that enables a user to customize the training and prediction stages according to the requirements" was implemented and inserted in the IoT system Application Layer. In the experimental stage, a pre-processing technique was used to standardize features. Techniques for solving the feature extraction problem related to feature vector imbalances were presented, and methods to obtain the most important features were explained. Some attention was also paid to testing different sliding window sizes.

Some measuring meters available in the market and smart plugs manufacturers are listed in [1]. Also, a comparison between the different options available, e.g., Bluetooth, *Zigbee*, *Wifi*, is presented for the communication layer.

3.2.1 Conclusions on ILM research

The research helped us identify and understand the technologies and tools needed to solve similar problems, allowing us to draw some conclusions.

Every study talks about SL algorithms for classification proposes. This dominance may be due to the better accuracy of SL and the fact that the load datasets always have data labelled, making the process easier. Also, it is possible to label the data collected, mainly in industrial contexts. However, no consensus on the best algorithm exists, which is justified "due to the variability of the sets of appliance categories, and due to the different types of measurements, sampling frequencies and feature selections" [23]. Moreover, all the works that test its classification models with unseen data obtained worse results than with seen data, making it clear that the model should be retrained to recognize new appliances and new tools in the workshop case.

As our work relies on tool detection, only the turn-on and off events are relevant to know the tools' usage times and recognise the restoration stages. So, the tools' energy states detection is irrelevant in our context.

The relevant role of data processing and feature processing is also evident due to the importance of data normalization before starting all the ILM processes. Also significant is the role of feature classification and selection for a subsequent dimensionality reduction

[3] UK-DALE,https://jack-kelly.com/data/, Last Access: 13/02/2022

to improve the efficiency and effectiveness of a predictive model. Some details, like the sliding window size, could enhance the accuracy of the classification model. Therefore, it is necessary to test different options and use the most suitable one.

Finally, an environment with several energy sensors or meters that communicate their information to the internet through communication technologies can be considered as an IoT system, so its architecture should be well planned and decided, choosing the most efficient and adequate devices and technologies to obtain a reliable system.

3.3 Location Fingerprinting

As explained in the Background chapter (section 2.6), the fingerprinting localization technique raised great interest and was chosen together with the proximity technique to implement our localization system. So a search of related works was made to find the procedures performed in each one and the results obtained.

We also seek to understand the experimental setup, such as the density of beacons in the area, and the impact of the configurations of beacons on the accuracy, such as different frequencies of signal propagation and different transmit powers. Attention was also given to the ML algorithms used in the training phase of this technique.

In [7], the area in question has 600 m^2, and it is an office environment in everyday use. The tests were done with 19 BLE beacons placed at 1 m of height. Beacons were set to 50Hz signal frequency, and a transmit power of 0 decibel-milliwatts (dbm). The device that captures the beacon's signal is an iPhone. A walk is done through the entire office area to build the fingerprinting database. An evaluation was done on the impact of the usage of filters like mean, median, or maximum, applied to a defined window size of RSSI data received in the construction of the fingerprinting database. The results showed that using the filters always reduces the error compared to building the fingerprints using raw data (less than 3 m 95% of the time versus less than 6 m 95% of the time). Also, the influence of the window size in the result is evaluated. The results showed that using data windows of around 0.5 to 2s performs better than using 0.05s windows. The influence of the beacon signal frequency was also evaluated, and the result showed that the bigger the frequency for the same window data size, the better the results.

Increasing beacon density showed a steady improvement in average accuracy until some point (1 beacon per 60 m^2) where it tends to stabilize. The best result was achieved with 1 beacon per 37 m^2. The transmit power configuration was also tested. The average error with 0 dbm(Strong) to -25 dbm had little differences, but after a reduction of 30 dbm, the error starts to increase in larger scale (less than 2.5 m 95% of the time versus less than 4 m 95% of the time). This error increase happens because small beacon "islands" starts to form. So the best result was an error of 2.5 m with 95% confidence obtained with one beacon per 30 m^2 at frequencies between 10 and 20 Hz and a transmit power of 0 dbm.

The implementation of [4] is done in a 24 m^2 area containing furniture, using three beacons, resulting in a density of 1 beacon/8 m^2. This implementation created a grid of points separated by 1 m each, Figure 3.1, to represent the measurement fingerprint points for the training phase.

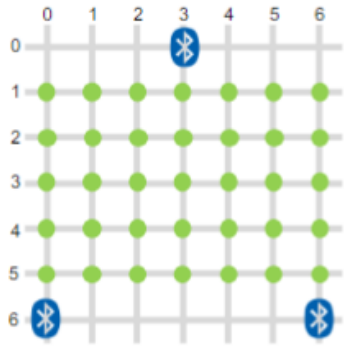

Figure 3.1: Experimental setup grid. Green dots represent the measurement/fingerprint points. Retrieved from [4].

At every fingerprint point, 40 RSSI values from each beacon were recorded. This value should be low to limit the delay of the estimations. The differences related to average error were studied when using 3 or 2 beacons, resulting in average errors of 1.18 meters and 1.27 meters, respectively.

The improvement caused by the direction of the antennas in the BLE transmitters and in the device to locate (the receiver) was also studied. So instead of just recording a fingerprint in each position, different fingerprints were recorded, varying the receiver orientation. It resulted in 8 fingerprints for each location. The results show a decrease in the average error.

A *search space restriction* method was also used to improve the fingerprinting results. This approach is used to detect human movement. It affirms that it is more likely that a measurement is matched to the same or nearby location of the previous measurement location. This method works by assigning coordinates to each fingerprint point. Then given the receiver's position, a clue of the next position can be obtained, so the fingerprints to compare can be limited. This process uses an accelerometer to detect steps and a magnetometer to detect orientation changes. Another approach is to restrict the next possible position to an adjacent one, comparing the new measure only with the fingerprints of the adjacent positions. This method had minor improvements (difference < 0.2 m) compared to the traditional approach.

In [29], the setup is in a school corridor with 2.5 x 4.5 = 11.25 m^2 (Figure 3.2) using 14 beacons at the height of 2.5 meters, making the beacon density to 1 beacon/0.8 m^2. Were used *Estimote* beacons with transmit power set to 4dbm and signal frequency set to 300 ms. The area is divided into squares of 45cm x 45cm each. Each square represents a unit in a coordinate system position over the map area. 5 squares, 4 in each corner of the map and one in the middle, were chosen for constructing the fingerprint database.

5000 samples were recorded on each of the 5 points and filtered. The filtering algorithm calculates the average RSSI values of each beacon, and if some beacon RSSI value is not received, the minimum value is inserted.

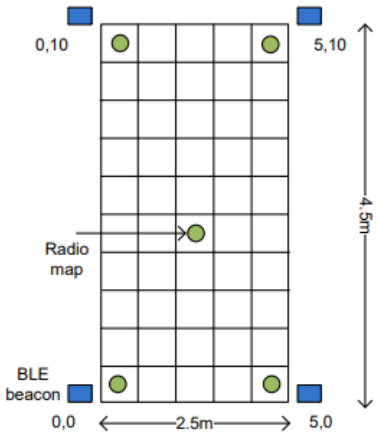

Figure 3.2: Experimental setup grid. Green dots represent the measurement fingerprint points used to train the model. Retrieved from [29].

Two different ML algorithms were tested, SVM and Logistic Regression (LR), to create the fingerprinting ML model. Furthermore, different sizes of data samples acquired at each point of the map grid were used to test the ML algorithms (from 10 to 50 samples). The result shows that the estimation error decreases as the number of samples increases. However, because the device to locate will be moving, this size can not be too large. Therefore, a minimum average estimation error of 50 cm was obtained with the SVM ML algorithm and with the maximum size of test data acquired in each point (50 samples).

In [14], a study compares the fingerprinting location results using *Wifi* transmitters and beacons, as well as the combination of both. Beacons used are from *Estimote*. Android smartphones are used to acquire data (the receivers). The receiver records RSSI data for 10 seconds in each place, and a median is then applied to create the fingerprinting database. The ML algorithm used to predict the position given the RSSI values is the KNN using Euclidean distance to find the nearest fingerprints. The experiments were conducted on one university floor, containing corridors, classrooms, and offices, with an area of 52 m x 43 m = 2236 m^2. 4 *Wifi* transmitters and 17 beacons were evenly placed in the corridors and classrooms on the ceiling. This setup makes a beacon density of 1 beacon/132 m^2. The beacon signal frequency was set to 100 ms and the transmit power to 0 dbm.

The results showed that the localization error decreased from 1m to 0.77m using both technologies (beacons and *Wifi*). However, using each one separately causes no improvements. The time acquiring RSSI values was 10 seconds, but it was also tested with lower values, from 1 to 10 seconds, so its influence could be understood. The results showed a constant improvement from a 2m error to a 0.77 m error.

Different beacon density was also tested. Two tests were done, test A with 6 beacons data selected from the initial setup (meaning 1 beacon/372 m^2) and the other, test B

using 12 beacons (meaning 1 beacon/186 m^2). So the fingerprinting error results using each beacon setup combined with the *Wifi* data was 0.99 m for the A test and 0.87 m for the B test. However, by comparing with the 17 beacon configuration (error of 0.77m), a decrease in accuracy appeared.

It is essential to highlight that using only beacon measures for the fingerprinting model, the lowest median error achieved was 1 m with the 17 beacon setup and using 10 seconds measures in each fingerprint point.

In [2], the ML algorithms: KNN, ANNs, Bayesian Rule Determination, and SVM are suggested to be used in the fingerprinting localization technique.

3.3.1 Conclusions on Location Fingerprinting

With the previous analysis of the different fingerprinting solutions, some conclusions can be made to help us build our localization solution.

As noted, the best beacon density differs for all the literature reviewed. The best results were achieved by using one beacon per 0.8 m^2 to 1 beacon per 132 m^2. Nevertheless, it is noted that the best results were obtained when more beacons were used (up to a certain number).

Relative to beacon signal frequency, higher frequency values improve accuracy as more extensive data sets are used to train the ML fingerprinting models. Moreover, the higher the transmit power, the better the results. These higher transmit power results are justified as the area covered by each beacon is more extensive, resulting in an overlap of beacon signals in each measurement point, making the data to use in the ML model more informative. Naturally, with more powerful configurations, the beacons will have a faster discharge of batteries, so this topic should be considered during implementation.

Also, there is no ML algorithm stated as the best for fingerprinting.

So when developing a system of this type, testing different algorithms and different beacon configurations is essential.

CHAPTER 4

Conceptualization And Implementation

This chapter aims to present the conceptualization and implementation of our work. It starts with the introduction of the developed system, with the context characteristics and conditions in which the system will operate (section 4.1). Afterwards, the software requirements containing functional and non-functional requirements are presented (section 4.2). Followed by the system architecture description (section 4.3), where the components and technologies used are described. And finally, the implementation solutions are presented (section 4.4).

4.1 Introduction to the Developed System

The following sections will describe the software requirements, architecture, and implementation of our system.

However, first, it is important to understand the characteristics of the problem better, that is, in our case, the different restoration activities carried out in the workshop, the characteristics and conditions of these activities, their order, and also the type of tools used in each one of them.

To this end, a study was initially carried out in conversations with the workshop workers, the workshop managers and supervisors, and with the on-site observation of the work carried out during working days in the workshop.

As was mentioned in the Background chapter, when describing the project where this work is inserted (section 2.1), this work comes from the need to automatically identify the restoration processes present in the *Charter of Turin*, carried out in the workshop, thus reducing the need for the man-in-the-middle. This charter, as also described in that section, is modeled in the platform created in [20] (it can be seen in [1]). So to better understand

[1] http://194.210.120.34:5000/diagram Accessed on: 03/01/2023

all the existing restoration processes carried out in the workshop and their sequence, a study of the model was also done. After carrying out the study, we could define which stages our system could identify and which would raise problems in identification or low accuracy.

The study carried out will be described next. As stated at the beginning of this document, we intended to use LM, more specifically ILM, as the main technique, together with the localization, to know the restoration processes being done to a car in the workshop. For that reason, it was clear from the beginning that our system could not detect activities deriving from the use of manual tools.

So, the first phase of the study consisted in clarifying which are the different activities/stages of restoration, the different types of electric tools used in each activity, and the zone of the workshop where each type of activity is carried out.

The image Figure 4.1 identifies the different restoration zones in the workshop.

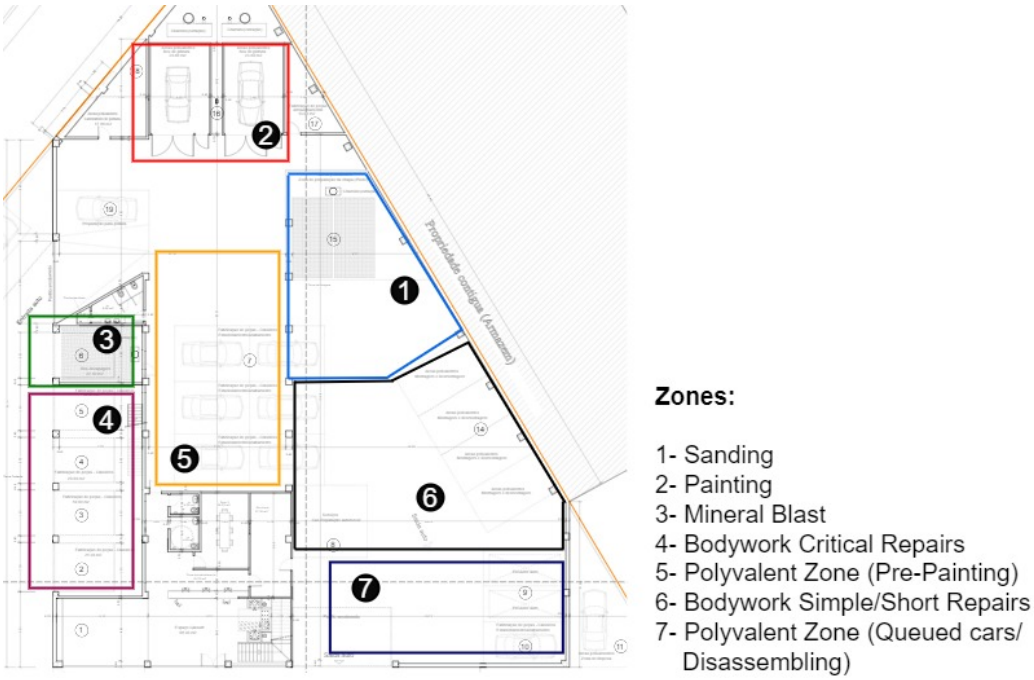

Figure 4.1: Workshop Floor Plan with the definition of the different zones, and the activities done there. Within brackets, in the caption, are presented the main restoration activities carried out there.

As can be seen in the Figure 4.1 caption, some zones are not used for a specific restoration task, this meaning that different tools can be used in those cases, and even more than one *Charter of Turin* stage can be processed there.

So the next step was to identify the tools, with particular attention to electric tools or electric machines, that are used in the existent zones, Table 4.1.

As can be noticed, there are zones where only one type of tool is used and zones where different tools are used to carry out restoration activities. Also, some tools are used in more than one zone.

Table 4.1: Matrix with relations between locations and tools used. Some non-electric tools may not appear.

Zone	Tools/Machines
Painting Booths	Dedicated Electrical Panel in each booth
Mineral Blasting Booth	Dedicated Electrical Panel
Sanding Zone	Electric Sander Sanding Paper (Manual)
Bodywork Critical Repairs	Tools for changing shape of metal sheets: English Wheel (Non-Electric) Bead Roller (Non-Electric) Metal Stretcher (Non-Electric) Angle grinder Artisanal Tools (e.g. Hammer)
Bodywork Simple Repairs	Angle grinder Spot Welder Dent Puller Electric Sander Electric Polisher Manual Tools (e.g. Hammer, Sanding Paper)
Polyvalent Zones	Electric Polisher Electric Sander Hot Air Blower Manual Tools (e.g. Sanding Paper)

Given the tools used in each zone and the *Charter of Turin* processes carried out there, we have found some system boundaries, and we could conclude the processes that would be possible to detect by using an ILM approach.

For the Painting Booths, as only one car fits in each booth, and each one has a dedicated electrical panel, with the use of an energy meter per panel and LM techniques, it will be possible to detect the different restoration activities done there, and associate it directly to the car inside.

The same works for the Mineral Blasting Booths, as there is a dedicated electrical panel for it too, and only a car fits.

In the case of the open spaces of the workshop, the solution is more complex and has more boundaries because the need emerges to match the tools detected by the ILM solution with the car's location where the tool is being used. The solution is to use the detection of vibrations caused by the work of the machines on the car panels to make this association between the car and a specific tool energy load. This process of crossing sensor data to identify processes will be explained more in-depth in the Implementation chapter (section 4.4).

Although the sensor data crossing in open spaces, briefly explained above, works correctly for some zones, given the problem conditions, there are zones where more is needed, as will be explained next.

The challenges for open spaces, some of which are briefly described in the Introduction chapter (section 1.4) of this document, are as follows:

- Just by identifying some tool being used in the workshop (by ILM), it is not possible to know in what car it is being used (tool work vibration detection in the car is needed);
- Knowing the location of the cars, and knowing by the vibrations generated that more than one car is being worked at the same time. When identifying at that moment some electric tool being used (by ILM), it is not possible to distinguish what car is being worked on. This happens because, as can be seen in Table 4.1 there are tools that are used in different zones (e.g. Angle Grinder, Electric Polisher);
- Associating the location of a car being worked on, to the closest outlet detecting energy loads, can raise accuracy problems because electrical extensions are used in the workshop.

Given this problem and the conditions in the workshop's open spaces, we decided to restrict our system to detect, in the open spaces, only the sanding zone restoration activities (sanding activity). We made that choice because:

- In the sanding zone, only the electric sander is used (in the electric tools set);
- So, if one or more electric sanders are being detected by the ILM, in the workshop, and a car inside the sanding zone vibrates, it can only mean that one of those sanders is working in that car.
- If just other types of tools are being detected by the ILM, in the workshop, even if some car in the sanding zone vibrates, it means the tool is not working in the car in that zone. (Only electric sanders can be used in that sanding zone car).

For the other activity cases, as enumerated in the challenges, there are a lot of ambiguities.

Therefore, knowing the limits and possibilities of our system, we can now present the *Charter Of Turin* restoration stages that our system should detect. We used the *Charter Of Turin Monitor* model present in the already described platform created in [20] to do it. The screenshots presented for each stage were taken from that model available in [2].

So, for the Mineral Blast, there is only one direct *Charter Of Turin Model* corresponding activity. It is the "Mineral-Blast body" activity, Figure 4.2.

In the Painting Booths, there are four different activities that the system should differentiate. "Application of anti-rust primer coat" with the following Cure, "Application of sprayable filler", the "Application of the paint primer coat" following Cure, and finally,

[2] http://194.210.120.34:5000/diagram Accessed on: 30/01/2023

the "Paint: Apply Paint" process (application of the final paint coat) and following Cure. The three first activities can be seen in the *Charter Of Turin Monitor* screenshot shown in Figure 4.3, and the fourth in Figure 4.4.

Finally, the sanding process is done in the sanding zone. A screenshot describing this process in the *Charter Of Turin Monitor* is shown in Figure 4.5.

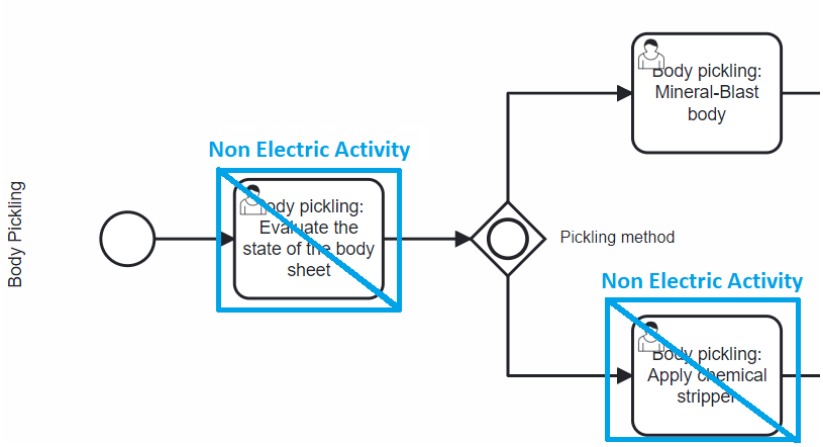

Figure 4.2: Mineral Blast activities. Print from the *Charter Of Turin* diagram model created in [20].

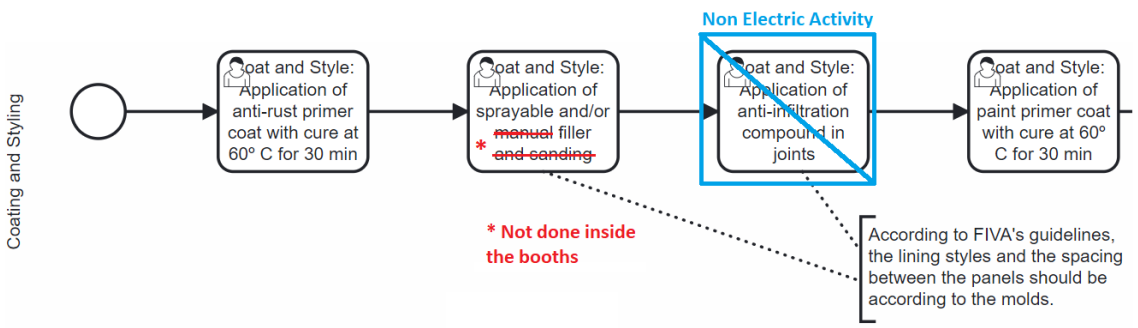

Figure 4.3: Painting Booths activities. Print from the *Charter Of Turin* diagram model created in [20]. The red marks tasks that are done outside the booth.

With this overview of the characteristics, challenges, and boundaries of our problem, we could reach the restoration activities carried out in the workshop that the developed system should detect and explain briefly the type of solutions used to create the system. Furthermore, this knowledge helped us better define our system's requirements and architecture.

Another important topic to introduce before describing the system more in-depth is the web application that will help monitor and control the IoT devices and check the restoration processes detected by the system.

This web application was already developed in the previous work [22], as already mentioned in the Background chapter (section 2.1). However, we wanted to make some

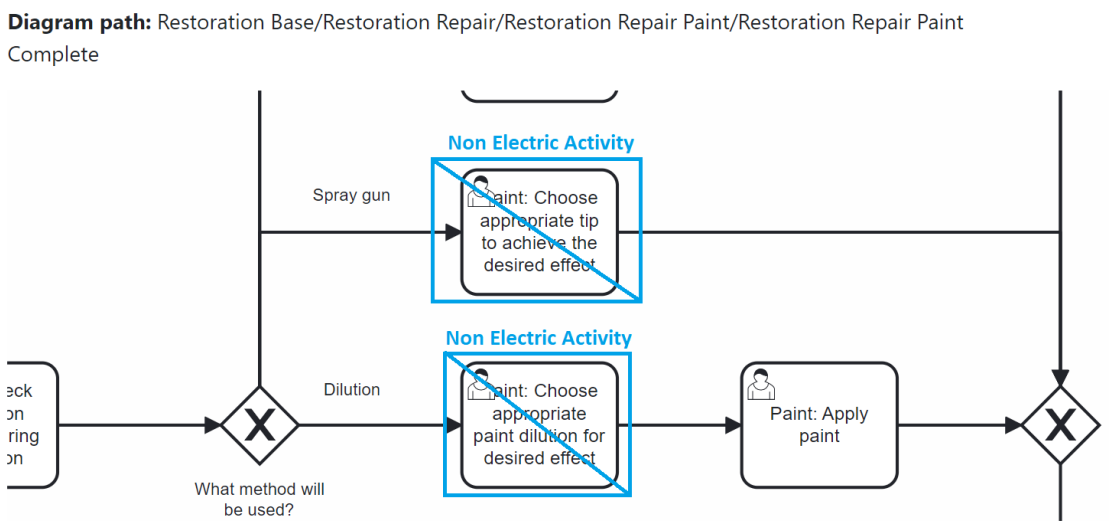

Figure 4.4: Painting Booths activities- Final paint application. Print from the *Charter Of Turin* diagram model created in [20].

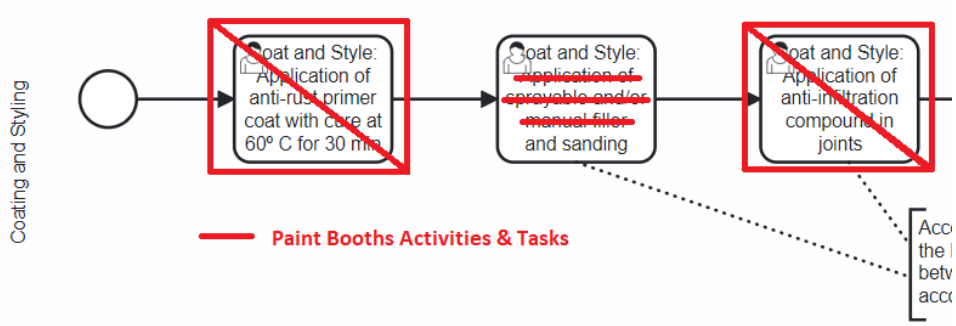

Figure 4.5: Sanding activity. Print from the *Charter Of Turin* diagram model created in [20]. The red marks activities and tasks not done in the sanding zone.

changes and extend it to offer more features. As presented in the section of the previous work flaws (subsection 2.1.1), it relied on pay-per-use software. The backend relied entirely on the AWS, and the frontend was hosted in a server provided by *Netlify* platform. So there was the need to migrate the whole application to a remote server provided to us (which will be presented later in subsubsection 4.3.3.8), create a database to support the application data, as well as an Application Programming Interface (API) to handle requests to the database.

Relative to upgrades on the application functionalities, with new types of sensors, came the need to add to the application forms of control and monitoring over these sensors. There was also the need to improve the existing monitoring solutions and develop new features.

The new computational design, changes, and upgrades related to the web application will be presented later in this document (in subsection 4.4.7).

So in the following sections, the software requirements will be specified, followed by

the architecture design, including the IoT devices and other sensors and software technologies used. Then, in the end, the implementation of the whole system is structured, from the sensor data capture to the generation of output files containing the restoration processes done to each car and the web application.

4.2 Software Requirements Engineering

Software Requirements Engineering "refers to the first phase, before any actual designing, coding, testing, or maintenance takes place" [3]. It corresponds to the specification of the features and constraints that a system to be built should have and where the system's goals are identified. This process should be done with the stakeholders so that the system aligns with their intentions.

Software Requirements should be divided into functional requirements and non-functional requirements [4]. The meaning of each one will be described in the following sections.

So in the following sections, the stakeholders that will interact with the system and the functional and non-functional requirements identified will be described.

4.2.1 Functional Requirements

The functional requirements are features or functions that the developed system must offer and implement to accomplish its objectives. The requirements can be divided between the system and users/stakeholders' requirements.

To describe the functional requirements of the developed system more clearly, we can divide it into two subsystems. First, the subsystem related to identifying the restoration processes and the web application subsystem developed to control and monitor the sensors used.

4.2.1.1 Restoration Processes Identification Functional Requirements

In the Restoration Processes Identification subsystem, no direct users interact with it, as the subsystem will receive sensor data as input and output the processes identified. So, only system requirements exist, these being:

- Locate the cars within the whole workshop space;
- Identify the beginning and end time of the Mineral Blast process done to a car;
- Identify the beginning and end time of the Sanding process done to a car;

[3] https://www.softwareengineerinsider.com/careers/software-requirements-engineering.html Accessed on: 30/01/2023

[4] https://www.geeksforgeeks.org/software-engineering-requirements-engineering-process/ Accessed on: 30/01/2023

- Identify the beginning and end time of the Curing process done to a car;
- Identify the beginning and end time of the Painting/Anti-rust/Filler application done to a car;
- Generate an output file containing the processes identified to serve as input to the *Charter of Turin Monitor* system ([20] work).

4.2.1.2 Web Application Functional Requirements

Relative to the Web Application subsystem, two stakeholders will interact with it.

- **Workshop Supervisor**: The workshop supervisors workers that will be responsible for doing the maintenance of the IoT devices/sensors;
- **Project Technician**: A technician responsible for making critical changes to the state of IoT devices, which will influence the operation and performance of the system.

Thus the system must have specific characteristics so that each stakeholder can interact with it as intended.

Relative to the **Workshop Supervisor** user requirements, the Web Application subsystem should allow this user to:

- Authenticate to the system;
- Access all the current IoT devices state;
- Access to all the current IoT devices place in the workshop;
- Access the alarms generated when problems occur with the functioning of IoT devices;
- Couple/Decouple the location Beacons from the system;
- Change the Sensor Boxes assigned car;
- Access the Sensor Boxes logs;
- Access the last restoration processes identified by each box in a Live Map of the workshop floor plan;
- Access the system output files containing the processes identified on the last day;
- Edit account password.

Relative to the **Project Technician** user requirements, the Web Application subsystem should allow this user to:

- Authenticate to the system;
- Access all the current IoT devices state;
- Access to all the current IoT devices place in the workshop;
- Access the alarms generated when problems occur with the functioning of IoT devices;
- Change the corresponding positions from all IoT devices;
- Start/Stop the Sensor Boxes from capturing data;
- Couple/Decouple the location Beacons from the system;
- Change the Sensor Boxes assigned car;
- Access the Sensor Boxes logs;
- Access the last restoration processes identified by each box in a Live Map of the workshop floor plan;
- Access the system output files containing the processes identified on the last day;
- Edit account password.

The system requirements in the web application consist of fulfilling the two stakeholders' requirements.

4.2.2 Non Functional Requirements

Non-functional requirements represent how the system should perform. They are related to the system's quality, ensuring usability and effectiveness.

So joining, in this case, the whole system, the non-functional requirements that our system should follow are:

- The restoration processes identification should be made close as possible to real-time;
- The IoT devices should be as less intrusive as possible to the workshop workers;
- The IoT devices should need minimal maintenance;
- The Web Application should have a steep learning curve;
- The Web Application application should have a short response time;
- The system must not rely on pay-per-use software;
- All the system components should be robust and faultless;

4.3 Architecture

With the system requirements defined, we can now describe the developed system's architecture, which follows these requirements. For an easier understanding, firstly, in a more abstract way, we describe the different components and interactions between them, forming a complete and functional system. Then we describe the IoT devices and technologies used to implement the mentioned components.

4.3.1 Components

The component diagram of the developed system with a color-based descriptive scheme, where we represent the newly created components, and the components taken from the previous work [22], which have changed or have been taken as a starting point for the implementations of our system implementation, is presented in Figure 4.6. It is important to note that in [22], all the sensors data, output data generated, and the web application data was saved in a cloud service provider component (corresponding to AWS). This component was changed to the *Time Series Database System* subsystem and the Database component of the *Monitor and Control Web App* subsystem.

Each component and the interactions with the rest will be described next.

The technologies used in each component and the communication channels will be described in the technologies sub-section (subsection 4.3.3), where also, the deployment diagram of our project is shown.

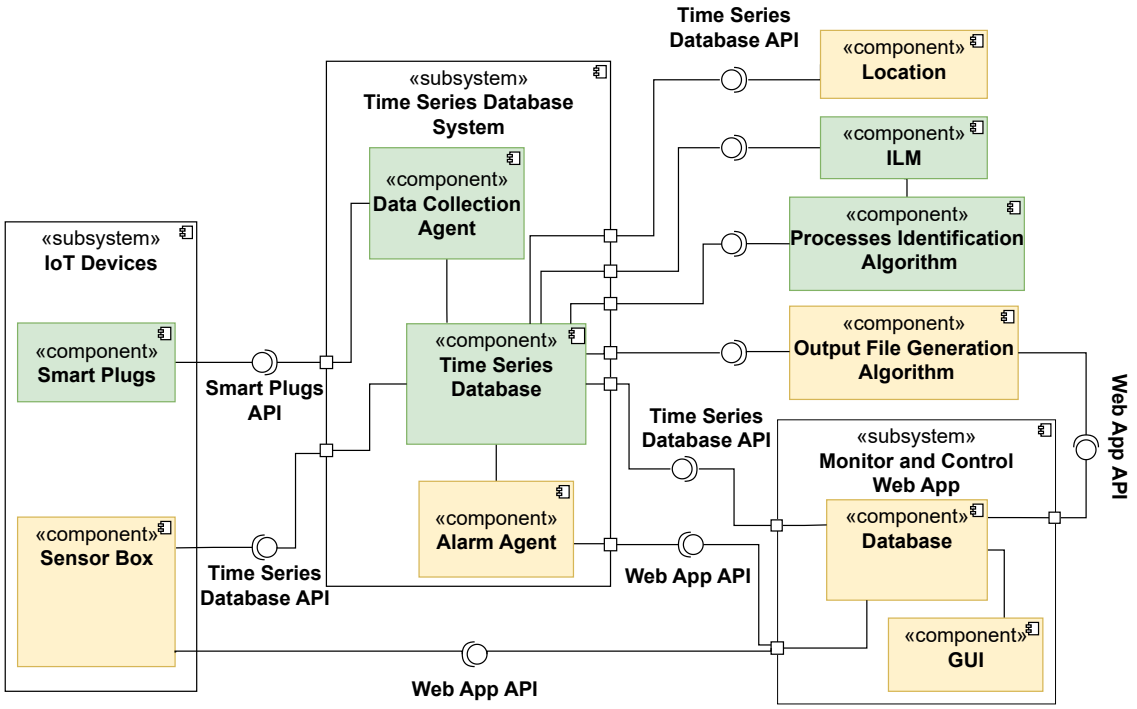

Figure 4.6: Component diagram of the developed system. The green color represents the components built from scratch. The yellow color represents the components edited.

4.3.1.1 IoT Devices Subsystem

The *IoT Devices* subsystem is responsible for the data acquisition for our system. It comprises the Smart Plugs component, responsible for measuring the energy load consumption in the workshop, and the Sensor Box component. The Sensor Box, as already mentioned, is responsible for getting location data and detecting vibrations produced in the car panels where it is attached.

There was a sensor box component also in [22]. The computations running there were adapted to the new needs and to support the communications to the new components. Related to hardware, the humidity and temperature sensors were removed from the box, keeping only the vibration sensor.

The data generated by both components are saved in the Time Series Database component, using its available API in the case of the Sensor Box component, and in the case of the Smart Plugs, is the *Time Series Database System* using its available Data Collection Agent component that uses the smart plugs interface to get the data.

The Sensor Box component also communicates with the *Monitor and Control Web App* subsystem to obtain settings and information defined there.

4.3.1.2 Time Series Database System

The Time Series Database component saves the time series data generated by our IoT devices and the output time series data generated by our computations. Besides its primary function being the ability to store time series data in different personalized tables, it also provides the possibility to make customised queries. In addition, it also provides the creation of monitoring dashboards of the data received, making it easier to observe the data variations and make it easier to study and make conclusions.

It also offers interfaces to query and use the saved data in other components.

The Data Collection Agent component is responsible for getting data from outside autonomously, only needing the address of the data source. It is used to get the data generated in the Smart Plugs component, as it offers an interface for this purpose.

The Alarms Agent component is responsible for generating alerts and alarms so the user of the Web Application can notice possible failures in the IoT Devices. The alarms are sent to the Web Application via its API. In the previous work [22], alarms were also generated using AWS services.

4.3.1.3 Web App Subsystem

The *Monitor and Control Web App* subsystem is responsible for the Web Application that will allow the system technicians and the workshop supervisors to check the status of the system, know if something has gone wrong with the sensors, control the sensors' configurations and monitoring the results/outputs given by the developed system.

The subsystem component is composed of a Database component responsible for saving the data of the actual state of the Web Application and changes that can be done there. The web application data was saved in AWS services in the previous version [22]. So in this version, the backend, where the database is included, as well as its API, will not depend on pay-per-use platforms, it will require a migration of the hosting platform, and also, it will be expanded to support new features.

Moreover, it is composed of a Graphical User Interface (GUI) component responsible for the web application's front end used by users to interact with the application. This component also existed in the previous work version and will be expanded to support new features.

The subsystem offers an API to allow other components to interact with the web application database, send data, and access its information.

4.3.1.4 Location, ILM, Processes Identification Algorithm and Output File Generation Algorithm Components

These are the components where the computations over the sensors data received are done, aiming for the restoration processes identification.

In the Location component, the computations are done relative to the location data acquired in the Sensor Box component. This component uses the data saved in the Time Series Database relative to a Sensor Box, predicts its location, and then keeps it again in the Time Series Database. The predictions are sent back to the Time Series Database, using its API, because this process repeats every time new location data is acquired. This localization component was started in the previous work, but as already mentioned, it was only developed for testing the proximity solution. Furthermore, the data used was from AWS.

The ILM component is where the energy load consumption data acquired in the workshop is interpreted. Given the latest energy data obtained as input, the component computations predict the tool or booth activity based on thresholds or a ML model. Then, equally to the Location component, the output predictions are sent back to the Time Series Database.

The Processes Identification Algorithm component is responsible for doing the computations to infer the restoration process that is being carried out. It combines the calculated location of the sensor box, the predicted electric machines, and the vibrations detected in the sensor box to infer correctly and with the least possible error the restoration process being carried out in a car. All the data used is available in the Time Series Database throw its API, and the inferences are also saved in the Time Series Database for the same reason present in the last components.

Finally, the Output File Generation Algorithm is responsible for joining all the processes detected by the previous algorithm. This algorithm gets the processes detected by the previous algorithm during the last day. With a sequence notion of the *Charter of*

Turin, it will agglomerate the processes according to their type and admit only processes in the correct order, resulting in an output file with the start and end time of all processes detected that day. It uses the data available in the Time Series Database throw its API and saves its output files in the Database component of the *Monitor and Control Web App* subsystem. The Web App API sends the files to the database. An algorithm used in [22] was used as a starting point for this algorithm's development.

It is important to note that when a component gets the input and then saves the output in the Time Series Database, it is assumed that there is one table for each case.

4.3.2 IoT Devices

Before describing the technologies used in the development of the system, first, the IoT devices used will be presented as some technology choices come from the sensors chosen.

So, in this subsection, the IoT devices used will be described, as well as the reasons and constraints behind their choice. Since, in some cases, different options exist and have been tested in order to be chosen the best one.

4.3.2.1 Sensor Box

As already explained, we need to track the location of the cars in the workshop. Also, we need to detect the use of tools in the cars, so the vibrations they produce should be detected.

These objectives were part of the previous work [22], described in section 2.1 section. So although used in different ways, as will be explained in the Implementation section (subsubsection 4.4.1.1), the same hardware was used for the Sensor Box because their validity was already proved in the prototype developed before.

Thus, for the Sensor Boxes, we maintained the *Raspberry Pi 4* powered by an *Asus ZenPower* power bank of 10050 mAh. As tested in [22], it supports 8 hours of a typical working day, making it only necessary to charge the power bank between working days (overnight). Using the power bank, the box can move with any car everywhere in the workshop. Every *Raspberry Pi* uses a *SD card* with 64GB of memory, but 16GB proved to be enough.

The *Raspberry Pi 4* allows connection to *Wifi*, allowing us to send data to the internet. Furthermore, it has the *Raspberry Pi OS*, an Operating System (OS) based on *Debian* enabling us to carry out computations on edge, decreasing the amount of data to be sent to the databases.

The *Raspberry Pi 4* also has Bluetooth 5.0, which makes it possible to use BLE Beacons (presented in subsubsection 4.3.2.2) to locate it.

The same sensor used in the previous work, the *ADXL345*, was used to detect the vibrations generated by the tools used in a car. The *ADXL345* is a three-axis accelerometer that can identify variations in movement along these axes. Thus, with the sensor attached

to a car being worked, the vibration caused by the tools in the car panel will be perceived and registered. The sensor's attachment to the cars' panels is done using a magnet.

The *ADXL345* is connected to the *Raspberry Pi 4*, resulting in our system Sensor Box, Figure 4.7. The box is attached to the car, also using a magnet.

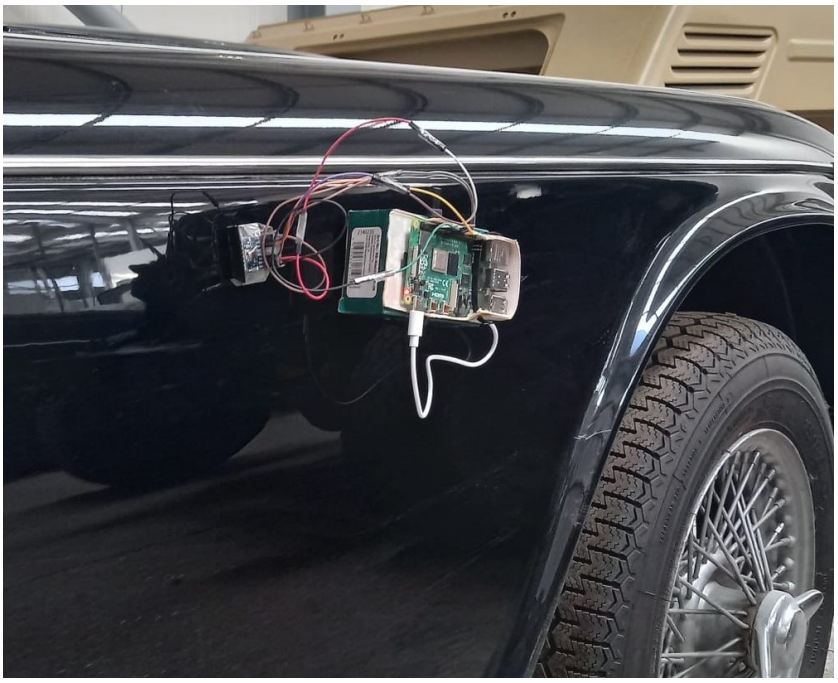

Figure 4.7: Sensor Box with the accelerometer ADXL345 attached to a car.

With all the visits to the workshop, we understood that not all cars are made of magnetic material, there are cars whose body is made of fibreglass, and nowadays, there are also cars made of carbon fibre. However, this body type is uncommon, especially in a classic car restoration workshop, so it is not a concern.

4.3.2.2 Bluetooth Low Energy Beacons

To locate the Sensor Box in the workshop, BLE Beacons by *Estimote* [5] are used. They were already used in the localization tests in the previous work [22] (as explained in section 2.1).

These are small-size devices that need 2 AAA alkaline batteries that can be easily changed by every person and have an estimated battery life of 3 years, with default settings [6]. These devices are always transmitting Bluetooth signals. And through the *Estimote* mobile application, one can configure the beacons by changing the transmitting power (with the higher transmit power, the higher the beacon range is) that can go from 1 m to 70 m and change the advertising interval (time between transmissions) that can

[5] https://estimote.com/ Accessed on: 03/01/2023

[6] https://community.estimote.com/hc/en-us/articles/202552866-What-s-the-battery-life-of-Estimote\-Beacons-Can-I-optimize-it- Accessed on: 03/01/2023

go from 100 milliseconds to 2 seconds [7]. These configurations are discussed later in the subsubsection 4.4.2.1, as they impact the result of the localization method implemented.

As the beacons are lightweight, they can be attached to the walls of the workshop using adhesive tape, as can be seen in Figure 4.8.

Figure 4.8: Beacons attached to paint and mineral blast booth in the workshop.

4.3.2.3 Smart Energy Meters

As will be described in the Implementation section (subsection 4.4.3), we needed two types of energy meters. The smart plugs type should be installed between the tools and the workshop outlets to capture the plugged tools' energy loads. And the energy load consumption meters that must be installed in the electrical panels of the painting and mineral blast booths. As already mentioned in the introduction of this chapter, the sandblasting, paint compressors, and paint drying oven in the mineral blast booth and the paint booths work directly connected to the respective electrical panels.

Relative to the smart plugs to be installed between the tools and the workshop outlets, the plug to be used must communicate with the outside via *Wifi*. There are options of plugs that communicate through *Zigbee*. However, it would be necessary to acquire a hub/controller to make the plugs accessible from the *Wifi* network and from outside the workshop [8]. Also, as the workshop has recently been rebuilt, a *Wifi* mesh network has been installed, allowing the network signal to reach the entire workshop floor area, so connecting the plugs to the network will not be a problem. So the communication via *Wifi* was chosen.

We want the sockets to capture electrical data in real-time, in Watts(W).

There are outlets with different power ranges, so the chosen outlet must be able to support all the tools used in the workshop identified at the beginning of this chapter, section 4.1. It should support at least 2500 W of energy power.

[7] https://community.estimote.com/hc/en-us/articles/201636913-What-are-Broadcasting-Power-RSSI\-and-other-characteristics-of-a-beacon-s-signal- Accessed on: 04/01/2023

[8] https://www.variantz.com/post/smart-home-connectivity-wifi-vs-zigbee Accessed on: 22/09/2022

As we want to access the data acquired by the smart plugs from the computer, so computations over this data could be done, the plug need to have an open and accessible *API*.

Considering these criteria and the quality/price relation, two smart meters were acquired and tested in the workshop with many available tools to choose the right one for our case. Both capture the electrical power in Watts(W). The Nedis also capture electrical current and voltage. Both send measures in real-time with a frequency of one measure per second. Both have an API to get the measured data and use *Wifi*, in the 2.4 GHz frequency, to send this data to the internet. The power range of the Nedis reaches 3650W, whereas the Shelly reaches 2500W.

The chosen outlet was the Shelly' one (Figure 4.9) because its API is more straightforward with easy access, its size is smaller (i.e., physically less intrusive), and its power range is enough for the tools used in the workshop. At the moment of this work's development, these meters were bought for around 15 euros.

Figure 4.9: Shelly Plug S. Used in the electrical data extraction. Retrieved from here.

The Shelly Plug S API is available on [9]. It has different endpoints that allow us to control configurations and monitor the Shelly' measurements and state. These endpoints always reply to the information in *.json*, Figure 4.10.

```
{
"power":6.01,"overpower":0.00, "is_valid":true,"timestamp":1662568051",
"counters":[6.615,0.000,0.000],"total":6
}
```

Figure 4.10: Example of the Shelly' endpoint {shelly_ip}/meter/0 response.
The "power" value represents the last measured electric power (in Watts).

For the electrical panel meters, some characteristics are also indispensable.

The meters should also communicate via *Wifi* and have an available API to obtain and use the power measured in the developed system.

After a conversation with the workshop electrician, we also realized that when the machines are started in the paint and mineral blast booths, there are amperage peaks of around 100 Amperes. So in this sense, the chosen meter should support this amperage value.

[9] https://shelly-api-docs.shelly.cloud/gen1/#shelly-plug-plugs Accessed on: 22/09/2022

In addition, the electrical panels of these booths consist of three-phase alternating current, which implies that the meter must also have the capacity to support and read the three phases.

Given these requirements, only one smart meter was found in the market. It was acquired and tested in the workshop. It was the Shelly 3 EM (Figure 4.11). At the moment of the development of this work, this type of meter was bought at a price of around 100 euros.

Besides meeting all the characteristics stated, it is easy to install. It also belongs to the Shelly brand, so it presents an interface and API [10] very simple and intuitive, as the Shelly Plug S described above. Besides that, both meters work with the same Shelly mobile application, which allows control and configurations over all the electric meters in the same mobile application.

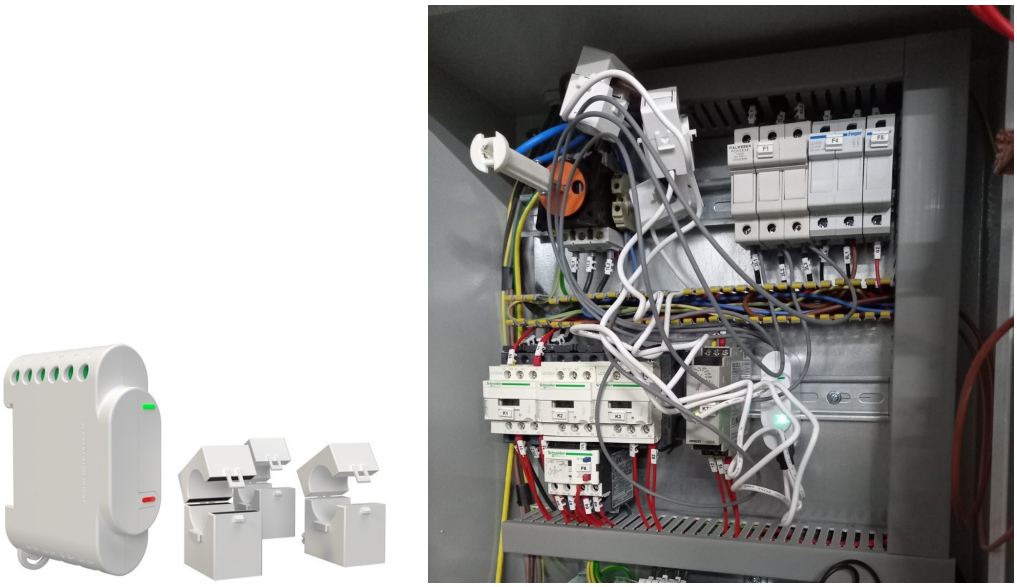

Figure 4.11: Shelly 3EM. The image on the right is a picture of the meter installed on the electrical panel of the Mineral Blast booth. Left image Retrieved from here.

Notably, acquiring these measuring devices requires some investment from a workshop, mainly concerning the electrical panel meters (Shelly 3EM), which have a relatively high cost due to being a more sophisticated device. However, it is important to note that these devices, in addition to enabling the system we are developing to work and allowing the workshop to benefit from it, will also enable the workshop to monitor its energy expenditure at various levels within the building. This will allow us to identify electrical wastes and points of higher consumption, reducing energy consumption and the electricity bill, resulting in a more eco-friendly industry.

[10] https://shelly-api-docs.shelly.cloud/gen1/#shelly-3em Accessed on: 22/09/2022

4.3.3 Technologies

With the sensors already described, this subsection will present the software technologies used to develop our work and the reasons for its choice.

Because it was the primary language used in the previous work [22], its simplicity and the large offer of existent libraries, the *Python* language was the main language used during the implementation of this work. For this reason, many of the technologies/tools used and presented next are relative to the *Python* environment.

4.3.3.1 InfluxDB

As we are working with sensor data and want to do computations over it and present results in real time, we need to have this data available as soon as it is received. As many sensors can send measures every second, saving their data in a local machine would drain the local memory resources quickly, so a cloud platform is needed. In [22], the AWS cloud platform was used. Although these major cloud services always offer a free tier for a wide range of services, which may initially seem sufficient, we may incur high additional costs if it scales. In addition, after having several components of the project running on that cloud service, there is a tendency to have the need/dependency to continue implementing on the same platform because a complete change of platform requires a lot of time and work. For these reasons, we needed to find open-source platforms.

Following these motivations, we decided to use the *InfluxDB* [11] platform to save all the sensors time series data generated, as it presents a lot of useful and relevant features for our work. *InfluxDB* is an open-source time series database for data storage and real-time analytics that collects and sends metrics and events from databases, applications, systems, and IoT sensors. Furthermore, it stores the data in buckets and allows monitoring of the data by the creation of live-data dashboards.

It offers an extensive list of client libraries ready to integrate an application. Our implementation, as described in the following sections, is done using *Python* language, so the most relevant library for our project is the Python Client Library [12]. This library allows us to create and do queries and write operations over *InfluxDB* buckets, from python, with few lines of code.

As a sensor data monitoring platform, it allows the generation of alerts related to incoming data and the consequent sending of alarms to outside the platform with the use of Hypertext Transfer Protocol (HTTP) requests, for example. In addition, it can generate alerts when no data is detected or when the data received is between some values.

Related to data ingestion, it also has many possibilities. Besides the Client Libraries write operations, the main tool available is *Telegraf* [13].

[11] https://www.influxdata.com/ Last accessed on: 07/09/2022

[12] https://docs.influxdata.com/influxdb/cloud/api-guide/client-libraries/python/ Accessed on: 05/09/2022

[13] https://www.influxdata.com/time-series-platform/telegraf/ Accessed on: 07/09/2022

Another advantage is the *InfluxDB* documentation, as it is well presented in its website [14], with short tutorials about the features available. There is also a *Youtube* channel [15] with explanatory videos, uploaded regularly.

4.3.3.2 Telegraph

Telegraf is a plugin-driven agent for collecting and reporting data from stacks, sensors, and systems. It supports many input and output plugins [16] allowing to get data from different sources and send it to different destinations autonomously.

An example of an input plugin is the *HTTP plugin* [17], used in our work, that collects metrics from one or more *HTTP* endpoints, autonomously.

An example of an output plugin is the *InfluxDB v2 plugin* [18], used to write the metrics/data collected to a specified *InfluxDB* bucket, autonomously. Every plugin has a wide variety of operational options that can be set.

4.3.3.3 SQLite

An open-source database technology needed to be chosen regarding the Database component in the *Monitor and Control Web App* subsystem. As the AWS was used in the previous work, and as explained, this is no longer desired.

As will be explained in the web application implementation section (subsection 4.4.7), the data to save in this database will not grow suddenly, and the system will not have large amounts of data, so the database that we choose should be simple and basic.

SQLite was used. It is a "lightweight disk-based database that does not require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language" [19]. There is also a *Python* library for this database.

Its software is open source, does not need any configuration, and has a size of less than 500kb, lower than other database systems. In our implementation (described in section 4.4), the *Python* language is used, therefore, to use this database, we need to install and use its *Python* library [20] that is well documented and easy to understand. It also offers enough resources for our needs, with a maximum database size of 281 terabytes, a maximum number of rows per table of 2^{64}, and a maximum blob length of 2^{31}.

[14] https://docs.influxdata.com/influxdb/cloud/Accessed on: 07/09/2022

[15] https://www.youtube.com/channel/UCnrgOD6G0y0_rcubQuICpTQ/videos Accessed on: 07/09/2022

[16] https://docs.influxdata.com/telegraf/v1.23/plugins/Accessed on: 07/09/2022

[17] https://github.com/influxdata/telegraf/blob/release-1.23/plugins/inputs/http Accessed on: 07/09/2022

[18] https://github.com/influxdata/telegraf/blob/release-1.23/plugins/outputs/influxdb_v2Accessed on: 07/09/2022

[19] https://www.sqlite.org/index.html Accessed on: 28/12/2022

[20] https://docs.python.org/3/library/sqlite3.html Accessed on: 28/12/2022

4.3.3.4 React

The frontend of the web application version developed in [22] was implemented using *React* alongside *Typescript* and *Tailwind CSS*. Because of their advantages, we decided to maintain these technologies to implement the new updates and features in the web application.

React [21] is an open-source *JavaScript* framework used for the development of web applications frontend. Enables quick construction of interactive user interfaces with small pieces of code.

Typescript [22] is a typed programming language that builds over *JavaScript* and makes it easier to write the code.

Tailwind CSS [23] is also a tool to make it easier to build web application layouts using less code. This tool is a *CSS* framework that offers many possibilities to design a user interface, e.g., controlling layouts, colors, and spacing. With this tool, there is no need to write custom *CSS* code, just utility classes with different parameters [24].

4.3.3.5 Nginx

To make available the *React* Web Application to outside the local network, we used *Nginx*.

Nginx [25] is an open-source web server that delivers static websites fast and in an easily configurable way. Moreover, it allows the developer not to worry about the incoming traffic, like the number of accesses and the possibility of simultaneous connections to the website.

4.3.3.6 Flask

To support the requests from the web application's frontend to the database, we needed to create an API. For that purpose we used *Flask* [26].

Flask is a *Python* library that allows implementing simply and quickly rest-full APIs. One can create as many endpoints as needed in just a python file. Its documentation is also well-described and easy to understand. An example of a python file with just a presentation endpoint is presented in Figure 4.12.

4.3.3.7 Pickle

The system created uses ML models, so there is the need to have a model trained and ready to give a prediction whenever this is requested. So, to save and make available the built ML models, we use the *Pickle* tool that generates *.pkl* files for each model.

[21] https://reactjs.org/ Accessed on: 28/12/2022
[22] https://www.typescriptlang.org/Accessedon:28/12/2022 Accessed on: 28/12/2022
[23] https://tailwindcss.com/ Accessed on: 28/12/2022
[24] https://blog.hubspot.com/website/what-is-tailwind-css Accessed on: 28/12/2022
[25] https://www.nginx.com/ Accessed on: 27/01/2023
[26] https://flask.palletsprojects.com/en/2.2.x/ Accessed on: 28/12/2022

```python
from flask import Flask

app = Flask(__name__)

@app.route('/hello/', methods=['GET', 'POST'])
def welcome():
    return "Hello World!"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=105)
```

Figure 4.12: Python file with simple API using Flask. Retrieved from here.

Pickle [27] is a python tool used for serializing and deserializing objects. Can be used to share, save, and load trained ML models for later use and rapid access [28].

4.3.3.8 OpenStack Server

As already mentioned, one of the requirements of this work was to use only open-source software without pay-per-use services. So, to run all the computations needed for our system and to host the backend and frontend of the web application, we take advantage of a server provided by the institute Infraestrutura Nacional de Computação Distribúida (INCD) [29].

This server is hosted in a remote machine accessible via *OpenStack*. This platform allows us to create instances with variable computational resources. For our system, the instance used has an *Ubuntu* operating system, 1 Virtual Central Processing Unit (CPU), 2GB of Ram, and 40GB of Disk memory. These resources were always sufficient to support the developed system.

To access the server from our PCs, we used Secure Shell (SSH) via the *Putty* [30] application.

4.3.4 Deployment Diagram

With the components, the sensors, and the technologies described, we can now finish the architecture description with the deployment diagram of the whole developed system, shown in Figure 4.13.

[27] https://docs.python.org/3/library/pickle.html
Accessed on: 28/12/2022

[28] https://practicaldatascience.co.uk/machine-learning/how-to-save-and-load-machine-\
learning-models-using-pickle Accessed on: 28/12/2022

[29] https://www.incd.pt/ Accessed on: 1/02/2023

[30] https://www.putty.org/ Accessed on: 1/02/2023

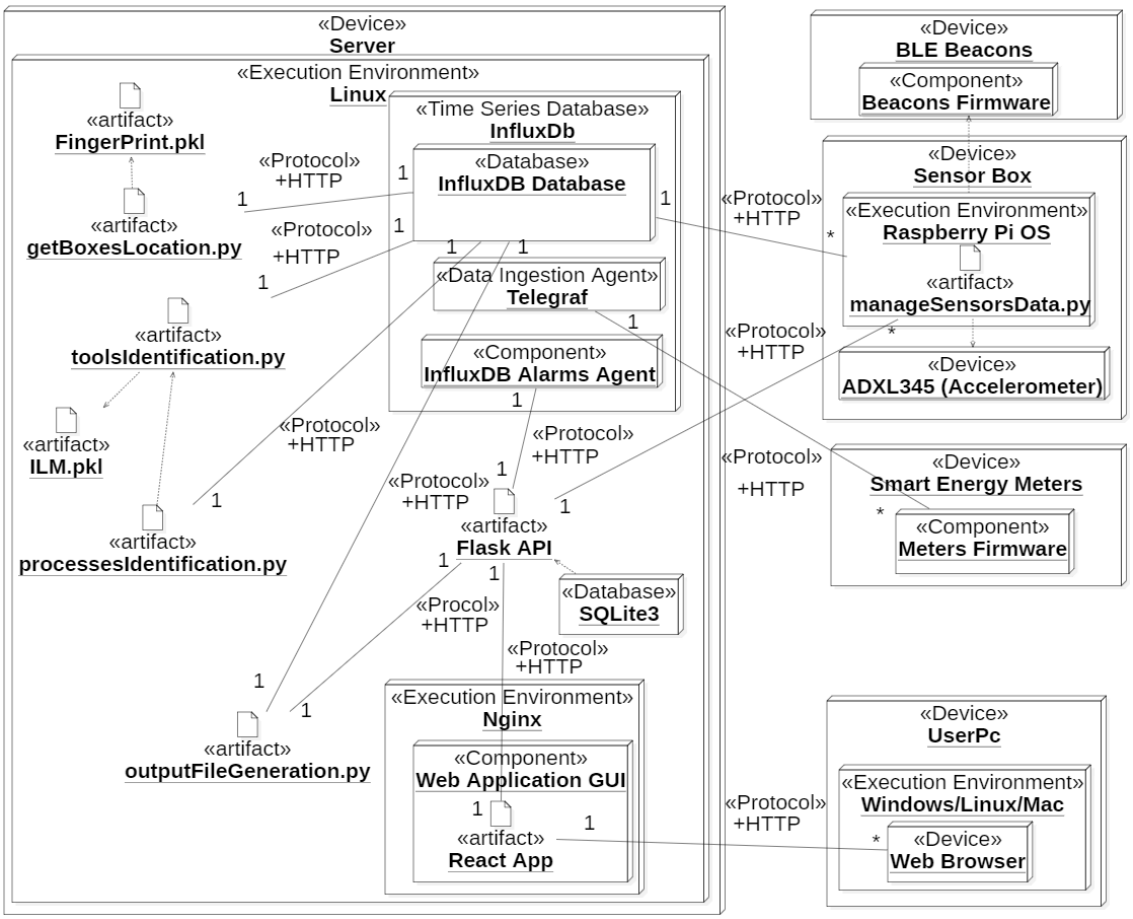

Figure 4.13: Deployment Diagram.

4.4 Implementation

With the system's architecture described, the implementation of the developed system will be detailed in the following sections.

During the development of our work, we designed and reached a diagram that represents a draft of the whole flow of the process identification, Figure 4.14. It is presented at the beginning of this section for a better and more structured understanding of the implementation.

The diagram helps us understand the information of the sensors needed first, the information necessary to take from that data, and the sequence of steps that should be taken.

Making an overall interpretation of the diagram can be noticed that, first, the location of a box needs to be obtained. Then depending on the area where the box is located, different modules may be required. For the case of painting and mineral blasting booths, the next step is to check the electrical activity and its type in the respective electrical panels. For the case of the box being in the sanding zone, there is the need to check for activity in the vibration sensor, and only then, check for electrical activity, looking for

sanders.

So in the following sections, the implementation of the localization solution will be described (subsection 4.4.2), followed by the load monitoring solution (subsection 4.4.3), the processes identification algorithm implementation (subsection 4.4.5) and finally the output file generation algorithm (subsection 4.4.6).

Before that, all the sensors data ingestion and its path to *InfluxDB* will be presented, as this was the first step needed in the development of the work (subsection 4.4.1).

In the end, the implementation of the web application will be described (subsection 4.4.7).

The differences between this work and the previous ([22]) will be stated during the descriptions of the topics.

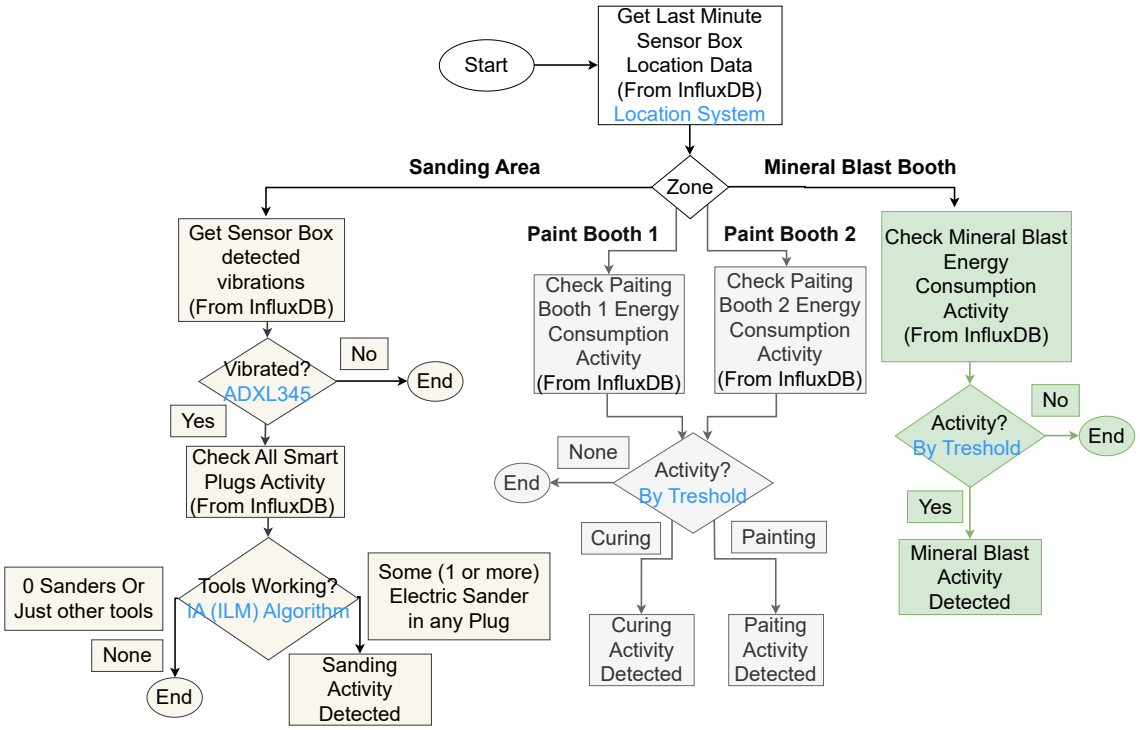

Figure 4.14: Identification of processes flow diagram - *Draft*. The blue color represents the techniques used to do the computations.

4.4.1 IoT Devices Data Acquisition and Ingestion To InfluxDB

As already described in the diagram of the components and in the deployment diagram, the sensor box data and the energy meters data are ingested separately. So the data acquisition of each one will be explained separately.

4.4.1.1 Sensor Boxes Data Acquisition and Ingestion To InfluxDB

Relative to the sensor boxes, a script is always running as a *systemd* service in the Raspberry Pi that initializes every time the sensor box boots. The script runs in a cycle. In

every cycle, it gets the last frequency obtained by the *ADXL345* accelerometer sensor and asks for the last received beacon signals received, saving the RSSI and battery value of each beacon.

To get the last frequency value, first, the last accelerometer vibrations data detected are obtained using the sensor corresponding python library [31]. Then, if vibrations are detected, the frequencies based on those are calculated. The same technique used in the previous work [22] was used for this. The Fast Fourier Transform (FFT) implementation of *Numpy* [32] is applied in each axe, calculating the respective frequency. After having the frequencies, the max magnitude between each frequency is obtained and used. Then the following steps are different from the previous work.

In the previous work, as the frequencies' maximum magnitude detected would serve to infer the tool that generated it, based on a previous register of the frequencies range of that tool, the maximum magnitude value needed to be filtered so the noise could be removed as much as possible. So techniques such as percentiles application were used. As we only want to detect if there is or is not a movement, we do not need complex noise reduction techniques. We only ignore small frequency values (below 0.025 Hz) as we understood that some frequencies are detected even with the sensor static, which is not intended.

A flow diagram of the frequency detection solution is presented in Figure 4.15. The value -1, returned when no vibrations are detected, will serve to inform the system that the accelerometer is not connected or is not working correctly, so an alarm can be generated. The web application alarms section will better detail this topic, subsubsection 4.4.7.2.

To get the last beacons signals, the ids and saved battery levels of the beacons registered in the system are requested to the web application API (this API will be described in the web application section (subsection 4.4.7). Then, only the coupled beacons (defined in the web application) are returned by the web application API, the coupled set. It is important to note that the decoupled beacons, are beacons decoupled from the system, meaning the system will not use its data. Furthermore, it is essential to note that because the beacons can not be turned off, they are always on. Then, from the last received *Bluetooth* beacons signals obtained using an *Estimote JavaScript* script, the ones with the ids included in the coupled set will be used with the corresponding RSSI value and battery level.

When new beacons with low battery levels (below 30%) are detected, a request to the web application API is made, sending the new low-level battery beacon ids. This action will generate an alarm in the web application per each beacon.

A flow diagram of the beacon data acquisition solution is presented in Figure 4.16.

With the beacons and frequency data obtained, this data is sent to an *InfluxDB* bucket with the sensor box id, the car assigned to the box, and the actual timestamp. To send

[31] https://github.com/nagimov/adxl345spi Accessed on: 06/02/2023
[32] https://numpy.org/doc/stable/reference/routines.fft.html Accessed on: 06/02/2023

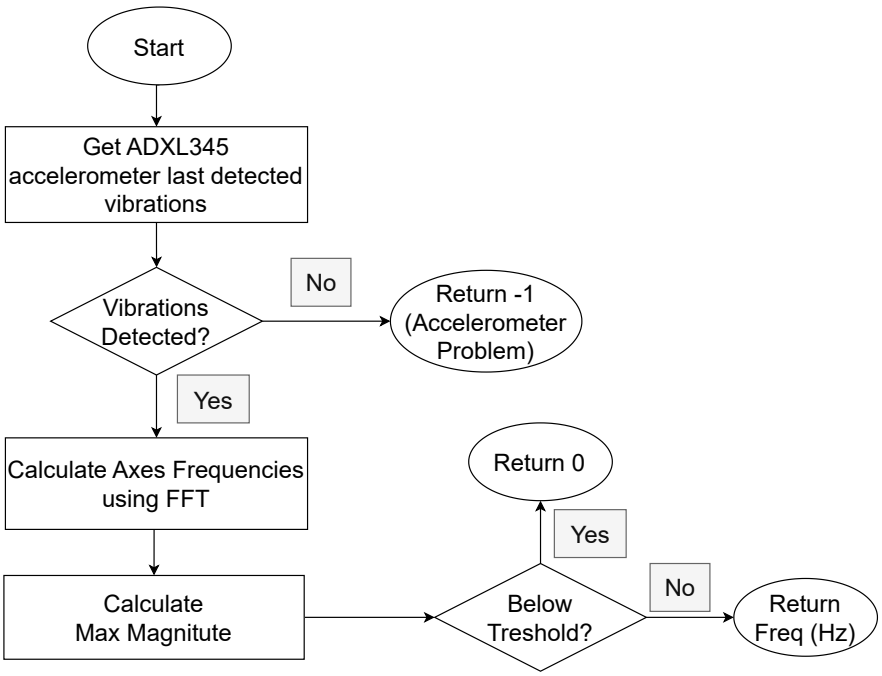

Figure 4.15: Frequencies detection flow diagram.

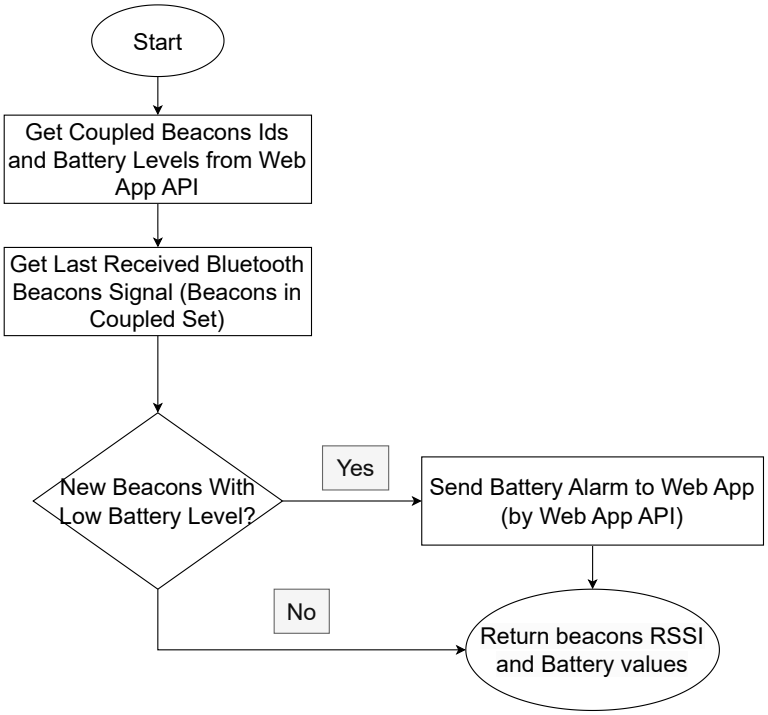

Figure 4.16: Beacons detection flow diagram.

data to *InfluxDB* from python, the *InfluxDB* Python Client Library [33] write operation is used (this library was described before, in subsubsection 4.3.3.1).

So, concerning the ingestion of the sensor box data to the SensorBoxesData *InfluxDB* bucket created, this bucket is divided into two *InfluxDB* *_measurements*.

A *_measurement* is a string "that describes the data stored in the associated fields" [34] of the bucket. It corresponds to a database table in a standard database, but to maintain the *InfluxDB* nomenclature, we will always use the term *_measurement*.

We divide it by the "beaconMeasures" and "frequencyMeasures". The "beaconMeasures" contains the beacon-related data, and the "frequencyMeasures" contains the frequency-related data. The different *_measurements* have some attributes that are the same but others that are different.

The attributes present in both are:

- Car Assigned - the car associated with the box that sends the data (the car is selected in the web application);
- Box Name - the id of the sensor box that sends the data;
- _time - (*InfluxDB* self attribute) the entry write action timestamp. This value will be defined in this case so that the frequency and beacon measures have the same timestamp and can later be associated.

Then the exclusive "beaconMeasures" attributes are:

- Rssi - Contains RSSI values of each beacon, sent by the sensor box;
- Battery - Contains the battery levels of each beacon, sent by the sensor box;
- BeaconId - Contains the ids of each beacon sent by the sensor box.

Then the exclusive "frequencyMeasures" attributes are:

- MaxFreq- Contains the frequencies sent by the sensor box.

A diagram with the *InfluxDB* SensorBoxesData bucket is shown in Figure 4.17.

Although the data flow from the sensor boxes is explained, in this script that runs infinitely in a cycle in each box, other actions are also performed in each cycle. They should be explained as they are essential for a complete system understanding.

So in each cycle, before looking for sensor data, the box script asks the web application API for the car assigned to itself and if it should get/record sensor data (its state). When it should not get sensors data, the box sends to the SensorBoxesData bucket in *InfluxDB* to the "frequencyMeasures" *_measurements*, an entry with the frequency value (MaxFreq)

[33] https://docs.influxdata.com/influxdb/cloud/api-guide/client-libraries/python/ Accessed on: 05/09/2022

[34] https://docs.influxdata.com/influxdb/v1.8/concepts/glossary Accessed on: 06/02/2023

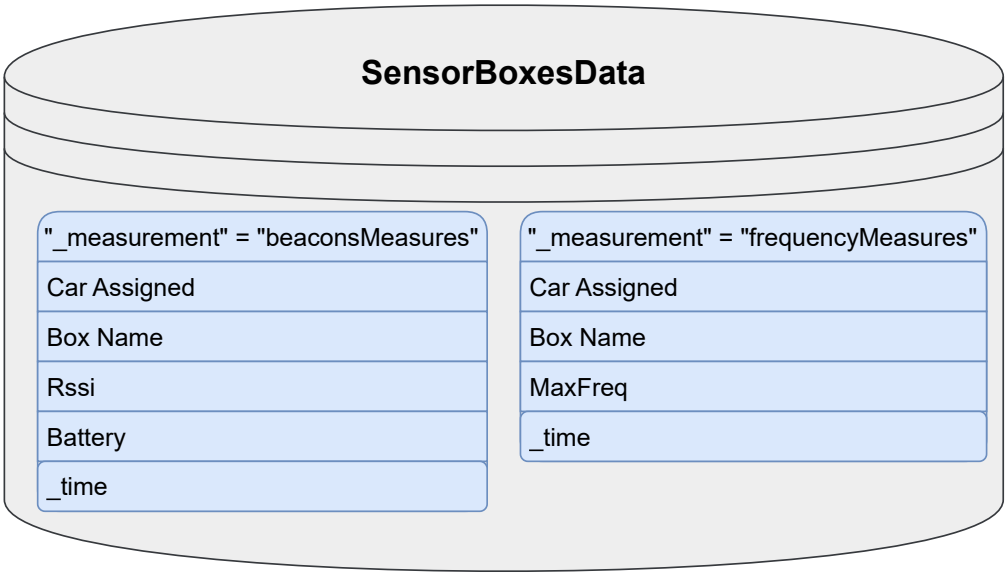

Figure 4.17: SensorBoxesData *InfluxDB* bucket.

equal to -2. This entry signals that the sensor box is alive, this is important for alarm control purposes (explained in the web application alarms section subsubsection 4.4.7.2).

If it should get sensor data, the steps to get the beacons values and the frequency value explained above are done.

So the general flow of the sensor box script is shown in Figure 4.18. The *Get Last Beacons Received* and *Get Max Frequency* actions are the ones specified previously (in Figure 4.15 and Figure 4.16 respectively) and are represented in abbreviated form in this diagram.

4.4.1.2 Energy Meters Data Ingestion To InfluxDB

Regarding smart energy meters, the data acquisition solution is less complex than the data acquisition in the sensor box. As mentioned before, in the IoT Devices section (subsubsection 4.3.2.3), the meters chosen (*Shelly Plug S* and *Shelly 3 EM*) have an available API, moreover, as explained in the *InfluxDB* technology description (subsubsection 4.3.3.1), it has the *Telegraf* data ingestion agent which provides us with automatic data ingestion by using plugins (also described in subsubsection 4.3.3.2).

We created a *Telegraph* configuration file for each meter. We configured it to make HTTP requests, using the *HTTP plugin*, every second to the meter API, and to save the data received in an *InfluxDB* bucket.

As the *Shelly's* offers many endpoints, we looked for the one that provided more electric information. In the *Shelly Plug S* API [35], the best endpoint is the *{shelly_ip}/meter/0*.

In the *Shelly 3 EM* API [36], the best endpoint is the *{shelly_ip}/emeter/0*.

[35] https://shelly-api-docs.shelly.cloud/gen1/#shelly-plug-plugs Accessed on: 22/09/2022
[36] https://shelly-api-docs.shelly.cloud/gen1/#shelly-3em Accessed on: 22/09/2022

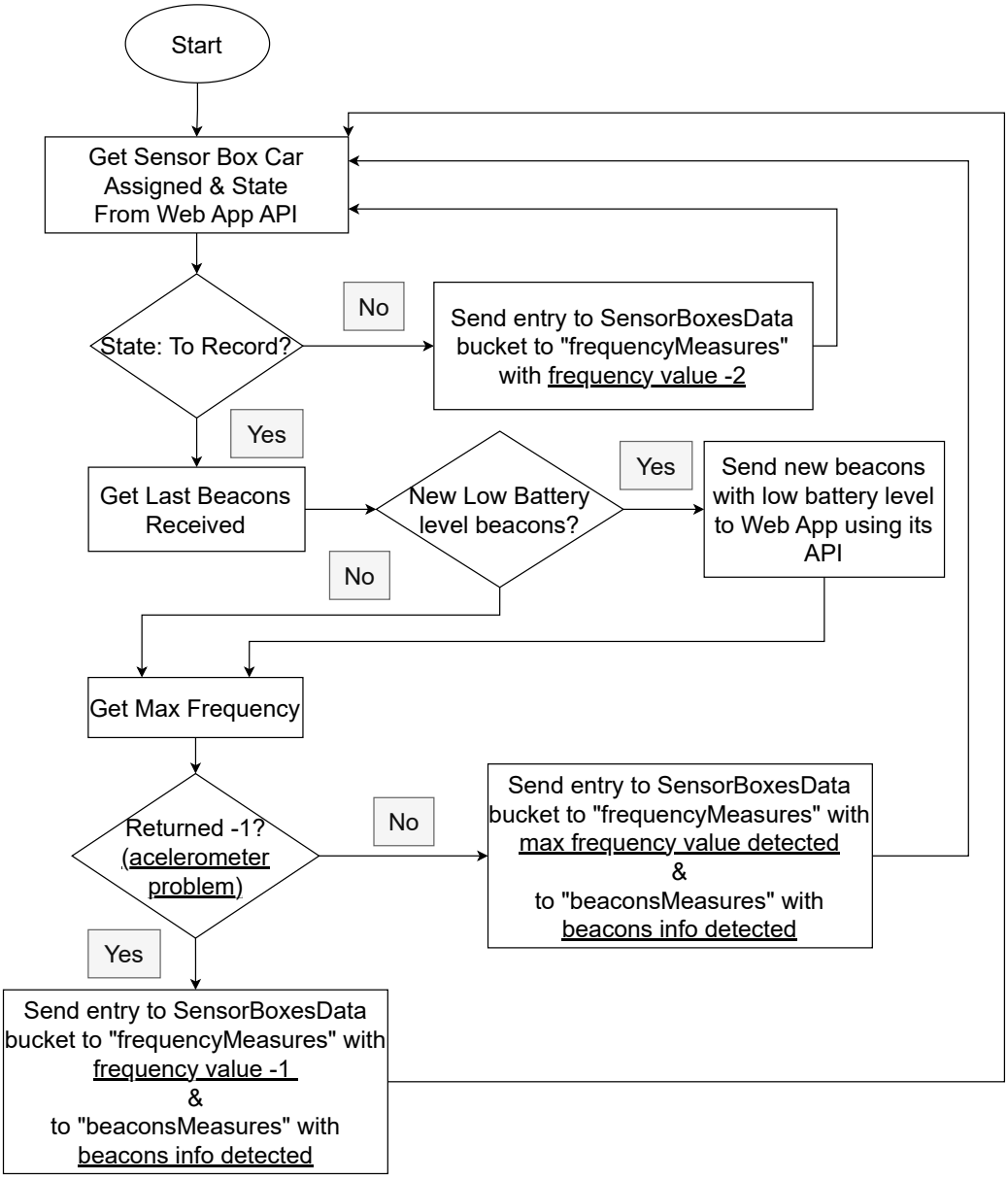

Figure 4.18: Sensor Boxes Data Acquisition - algorithm flow.

Before running each meter configuration file, it was necessary to port forward each meter Internet Protocol (IP) in the workshop router. This action was necessary, as the *InlfuxDB* is running in the remote server, and without port forwarding, it would not be possible to access each meter's endpoints.

As already described in the smart energy meters section (subsubsection 4.3.2.3), the API response is in *.json* format. Furthermore, the *Telegraf* configuration file with HTTP plugin saves the *json* fields automatically, in a selected *InlfuxDB* bucket in a given *_measurement* generating attributes on it equal to those fields.

So we created a bucket named Smart Plugs, and each energy meter *Telegraf* configuration file saves its data in a *_measurement* with the meter name. So all the *Shelly Plug S* meters have the same table/*_measurement* attributes, and the same happens with the *Shelly 3 EM*.

The meters API's selected endpoint return fields that are not important to our case so we will present only the important fields returned and saved in the bucket.

Relative to the *Shelly Plug S* meters *_measurement*, it has the attributes:

- Power - Alternating Current (AC) power detected in (W)atts;
- Overpower - Value in Watts, on which an overpower condition is detected (power over 2500W);
- Total - Total energy consumed by plugged devices in Watt-minute;
- _time - (*InfluxDB* self attribute) the entry write action timestamp.

The *Shelly 3 EM* meters *_measurement* has more information:

- Power - AC power detected in Watts;
- Current - Electric Current in (A)mpere;
- Voltage - RMS voltages, in (V)olts;
- Total - Total energy consumed by devices in Watt-hour (Wh);
- Total_Returned - Total energy returned by devices in Watt-hour (Wh);
- _time - (*InfluxDB* self attribute) the entry write action timestamp.

A diagram with the *InfluxDB* Smart Plugs bucket is shown in Figure 4.19.

With the knowledge of how the data is treated when acquired and then saved in the buckets as well as its structure, we can now start describing the computations needed to detect the restoration processes. So the next step is to understand how the sensor box localization works.

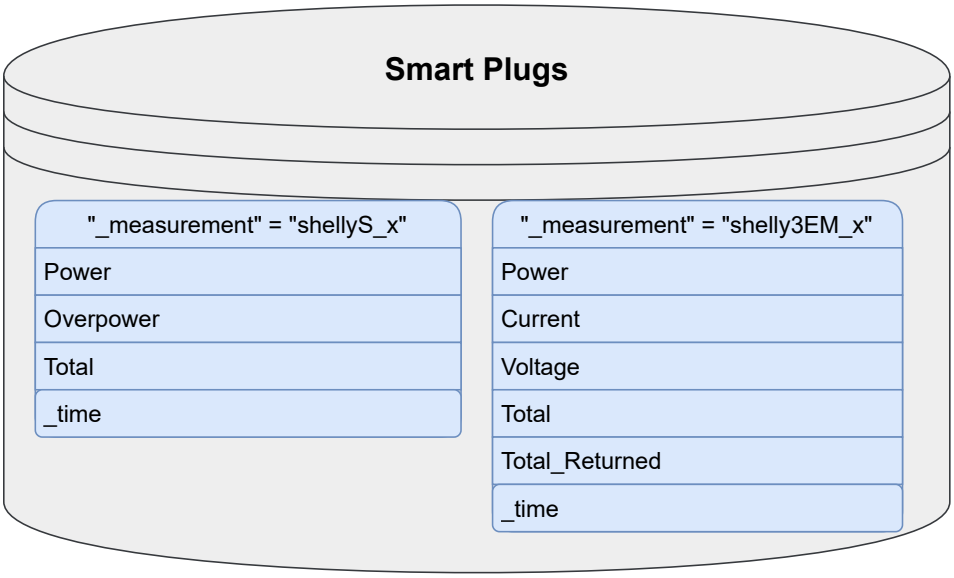

Figure 4.19: Smart Plugs *InfluxDB* bucket.

4.4.2 Localization

The box's localization is the first step, always done in the processes identification, as was briefly presented in the section 4.4, and in the draft diagram in Figure 4.14.

As was already stated in this document, a solution with BLE *Estimote* beacons was built. Because of the large size of the workshop containing a large open space area, the fingerprinting technique was used. However, a proximity technique was also used because of the importance of zone identification in the booths and in the sanding area. Both techniques were introduced in the Background chapter (section 2.6). So we built a mixed-localization solution. The proximity solution will be described first, followed by the fingerprinting solution, and finally, the localization algorithm that takes the beacons values as input and outputs the box's location.

Before explaining each solution, the final beacons placement in the workshop is presented in the plan shop floor figure, Figure 4.20.

Naturally, the placement of the beacons in the figure is not the initial version. It has changed during the tests done and attempts to achieve the best accuracy when differentiating the specific zones. Some of those iterations over the beacons placement and its consequent accuracy will be explained in this section.

As can be seen in the figure, we separate the beacons used for proximity and fingerprinting. We did that because early on, we realized that fine-tuning the proximity solution required changes in the settings of the beacons used for this purpose, as well as changes in their positions (this iterations will be addressed next in subsubsection 4.4.2.2). In order not to perform the rebuild of the entire fingerprinting each time these changes were necessary, we separated the beacons to be included in each solution.

So, to start all the beacons placement, we had 16 beacons available. In the Related

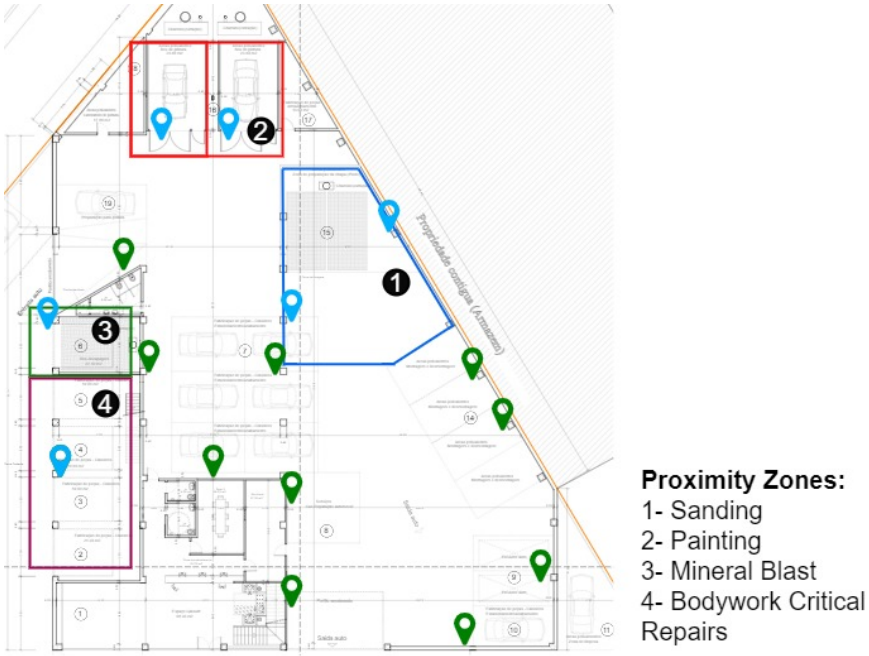

Figure 4.20: Workshop Floor Plan with the definition of the different proximity technique zones and the final beacons placement. Blue marks correspond to the beacons used for proximity.

Work chapter, in the conclusions taken from the fingerprinting location technique solutions (subsection 3.3.1), we find that the best results were achieved using beacons densities from 1 beacon per 0.8 m^2 to 1 beacon per 132 m^2, however, knowing that with more density, the better the accuracy results would be.

We understood that five beacons for the sanding zone, painting, and mineral blast booths would be necessary (justification presented next in subsubsection 4.4.2.2) to the proximity solution.

The workshop fingerprint area is around 485 m^2, so by using 10 beacons to the fingerprinting solution, the beacons density of the fingerprinting is one beacon per 49 m^2, value in the middle of the density range found in the literature. So we considered that value acceptable, even assuming that the accuracy could be better with more beacons. So, we distribute the 10 beacons equally over all the workshop spaces designated for fingerprinting.

In the Related Work chapter, in the conclusions taken from the fingerprinting location technique solutions (subsection 3.3.1), we also understood that different beacons configurations result in different accuracy. So, in the first phase of the work, we did some tests to reach the best configuration. These tests will be presented next before the description of the solutions implemented.

4.4.2.1 Beacons Configuration Tests

The beacon advertising period (the frequency of the signal transmissions) and the transmit power of the beacons (with the higher transmit power, the higher the beacon range is) are the main configurations available on the beacons, already presented in subsubsection 4.3.2.2.

Relative to the beacon advertising period, the signal transmission frequency can be defined between 1 signal per 100 ms and one signal per 10 seconds. As we want to maintain the localization as live data, we want to receive beacons signals every second, so we define every beacon to send one signal every 500 ms.

We understand that this frequency is relatively high and could consume the beacon battery relatively faster. For this reason, we keep track of the beacon's battery level using that signal frequency configuration for five months. In 5 months, the battery levels have dropped 10 values on average, on a scale of 0 to 100. Considering a minimum battery level of 30% means a predicted battery life of almost three years, which is acceptable.

Relative to transmit power, *Estimote* beacons configuration offers an extensive range of different transmit power configurations, from -30 dbm to 4 dbm, with an approximated corresponding distance signal range of 1.5 m and 70 m [37] (factory announced). The operating environment, obstacles, beacon placement, and sensitivity of the receiver device can influence this distance.

We knew in advance that, for fingerprinting solutions, based on related works (subsection 3.3.1), the higher the beacons transmit power, the higher tend to be the precision of the solution. So, the best results should be achieved when testing with all beacons set to the maximum value of 4 dbm. However, to perceive this impact in the context of our workshop, we tested with the 4 dbm, and then we repeated the test with beacons in -10 dbm configuration. This way, we could evaluate the system when every measurement point receives signals from every beacon and compare it to when measurement points only receive signals from the closest beacons.

A first fingerprinting setup was built in the workshop to test the different transmit power configurations, shown in Figure 4.21. Because of the average car length, we intended to separate each measuring point by around 3,5 m. In Figure 4.21, the measurement points, green icons, and the beacons positions in blue are shown.

We followed every step of the fingerprinting technique. We tested the location inferences generation based on fingerprinting-related works [2],[14],[29], with the KNN, Decision Tree (DT), Random Forest (RF), Gradient Boosting (GB), SVM and Gaussian Naive Bayes (GNB) ML algorithms. We started testing with the 4 dbm transmit power, followed by the -10 dbm configuration. Then, we compared the best accuracy result achieved with each configuration.

The difference in accuracy values was quite notable. For the -10 dbm setting, we got a

[37] https://support.kontakt.io/hc/en-gb/articles/4413258518930-Beacon-transmission-power-range-\and-RSSI Last accessed on: 17/09/2022

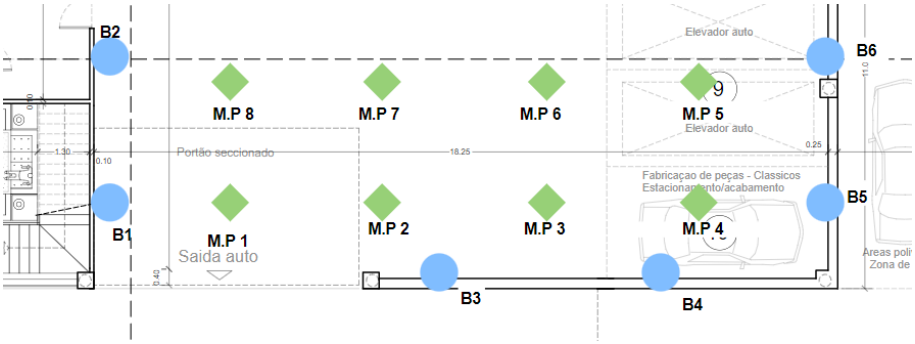

Figure 4.21: Workshop wide space zone test set up, with fingerprinting measurement points (green) and beacons placement (blue).

maximum accuracy of 53.3% (with the GB), while with the 4 dbm setting, the maximum accuracy was 76.5% (with the GB as well), which means that 4 dbm is optimal transmit power configuration.

Regarding the last statement, it is possible to understand its justification by evaluating the data. Given the 4 dbm configuration, in all measurement points, the sensor box always received RSSI values from all beacons, even with human movement and cars placed in the area. However, with the -10 dbm configuration, there were measurement points where RSSI values were not received, in the sensor box, from some beacons, justified by the reduced transmit power. Having these misses in the ML feature training data, we fill it in with the highest possible/distant RSSI value (100). However, as the data recorded in the same position sometimes receives data from some beacon and sometimes not, inconsistency causes misunderstandings in the ML model.

The battery life test, explained before, was also done with the 4 dbm transmit power configuration to understand the impact of this maximum intensity level on the battery of the beacons.

Given the results (around three years), meaning that the battery life of the beacons is acceptable in these configurations, we decided to set the default to transmit power of the beacons used to 4 dbm in our system.

As said, this is the default value used. However, in constructing the proximity solution, it was necessary to vary the transmission power of some beacons. This solution is presented below.

4.4.2.2 Proximity Build

As the processes that we specified for the system to detect occur in the sanding zone, painting, and mineral blast booths, identifying the box location in these zones must have an accuracy close to 100%. Because of that, the proximity technique was chosen for these areas.

Furthermore, due to the increased need for precision in identifying these areas, we have also decided always to prioritize the proximity solution regarding the number of

beacons to be used. If the number of beacons is restricted, we prefer higher precision in the proximity solution than in the fingerprinting solution.

For example, in the first beacon set-up, only one beacon was placed in the sanding zone. However, when locating cars in the lower part of the zone, the localization system mistook the closest beacon. So, in that case, it was necessary to take a beacon from the fingerprinting and change it to the sanding zone to work in that solution.

It can be verified, in Figure 4.20, that the bodywork critical repairs zone, which is in contact with the outside, is also defined as a zone where the proximity solution is used and has a proximity beacon. It happens because, initially, this was a beacon belonging to the fingerprinting set, and this zone would be divided into four fingerprinting measurement points. However, with only that beacon, the accuracy in differentiating the four points was reduced. However, we chose to prioritize the localization precision in the interior space of the workshop, not removing beacons from that area to be placed in the bodywork critical repairs zone. We made that choice as the inside area is where electric tools are used most and where cars spend most of their time. So, the proximity solution was also used to locate the car inside that zone.

So how the proximity solution works in the workshop? As mentioned, the sensor box should be permanently attached to his assigned car. Moreover, as mentioned before, the sensor box constantly collects signals from the beacons (positions shown in Figure 4.20).

So, concerning the sanding area, when the car enters the area, the beacon with the higher RSSI value - the closest beacon - should be one of the two beacons of that zone. When the car leaves that zone, the closest beacon must be another. The same succession work for the bodywork critical repairs zone.

For the painting and mineral blast booths, the beacons are placed close to the doors. Then, when the assigned car enters the booth, the sensor box should be attached close to the booth beacon (Figure 4.22). This is possible because the walls of the booths are made of metal, and the box has magnets. So when the sensor box is placed there, the closest beacon should be the beacon corresponding to the booth where it is placed. When the car assigned leaves the booth, the box should be attached to it again, and the closest beacon should be another one.

To get to the final version of the positions of the proximity beacons (showed in Figure 4.20), several tests were made in the workshop, moving the sensor box from the center of the proximity zones to their extremities, leaving them afterwards to identify to what extent the beacon identified as the closest was the correct one. When this did not happen, a change was made in the transmit power configuration of the nearby beacons, or changes were made to their positions, and the test was repeated.

The computations and data flow relative to the calculation of the closest beacon is presented in subsubsection 4.4.2.4. It will be presented after the fingerprinting solution is described because both are handled on the same script.

Figure 4.22: Sensor Box attached to a painting booth. And close to the corresponding beacon.

4.4.2.3 Fingerprinting Build

As mentioned before, the priority consisted in locating the box in the booths and sanding zone, as that is where the restoration processes to identify are carried out. However, it is also interesting to locate the car throughout the workshop space so that we have information on its route while in the workshop. Also, the fact that we can locate the car in all the workshop space can facilitate the expansion of the system in the future, with the identification of other restoration processes.

So, as the precision can be lower than in the previous solution, we decided to define the fingerprinting measurement points with around 3.5 meters between each other in the places where cars are mainly positioned. These points are shown in the workshop floor plan Figure 4.23. The measurement points where the RSSI values were taken are represented by the crosses, and the green circles represent the area that should be associated with that fingerprint measurement point.

To record the fingerprinting data, the sensor box was positioned in each measurement point, static, at the height of around 1.5 meters or car height, Figure 4.24, and then five files, containing 1 minute of received beacons RSSI signals each, were saved. Then for each file, a features entry was generated, containing the average RSSI value of each detected beacon and, as the ground truth, the name of the corresponding measurement point, Table 4.2. The five entries per measurement point in the features file are used in the ML model.

The features file was used to build the ML model that will predict the fingerprint point

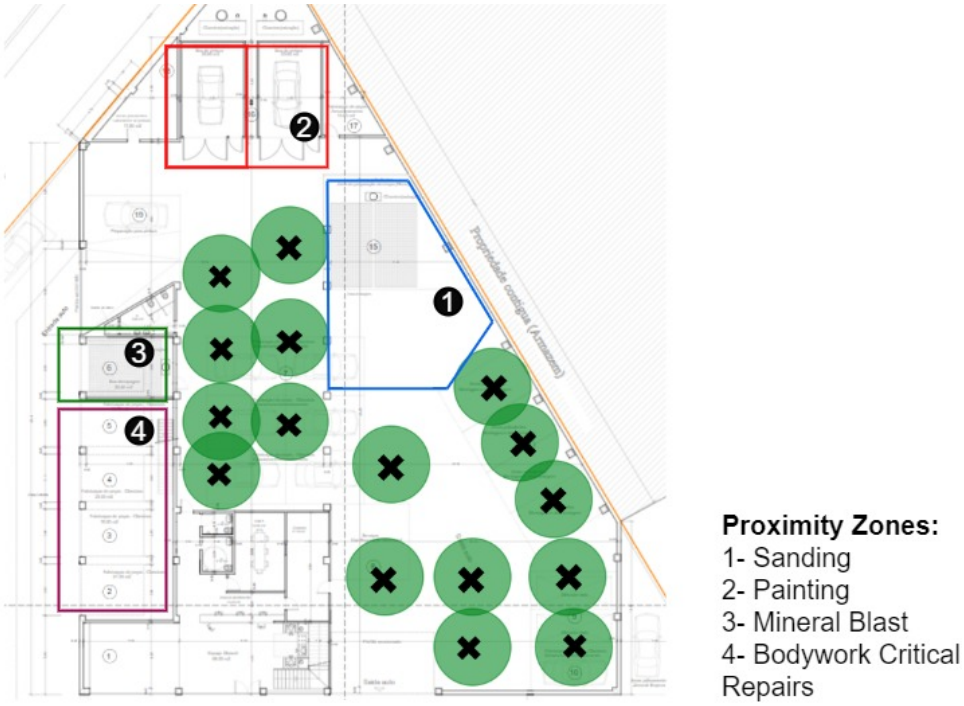

Figure 4.23: Workshop Floor Plan with the different fingerprinting measurement points positions.

Table 4.2: Example of a features file row of the data acquired in a fingerprinting measurement point to serve as input to the ML model.

Beacon id1(*RSSI*)	Beacon id2(*RSSI*)	Beacon id3(*RSSI*)	Meas. Point Id
90.5	80.6	30.5	M1

Figure 4.24: Photos of fingerprinting measurements.

where the box is located. We built the training set containing three feature entries from each point randomly chosen from the five entries, and for the test set, the two remaining entries were appended.

Then we trained and tested the model with the different ML algorithms used in the literature (presented in the Related Work chapter, section 3.3), i.e., the *Random Forest Classifier*, *K-Nearest Neighbors*, *Decision Tree*, *Gradient Boosting*, *Support Vector Machine*, *Gaussian Naive Bayes*. The one that achieved the best accuracy was chosen and used in our system architecture. As shown in Table 4.3, two algorithms resulted in better accuracy. However, in the RF, the predicted position errors were closer to the correct position, so the chosen one was the *Random Forest*.

Table 4.3: Accuracy of the tested ML algorithms, for fingerprinting.

Algorithm	Accuracy
KNN	84.4%
DT	54.1%
RF	84.4%
GB	60.5%
SVM	75%
GNB	59.3%

After being trained, the fingerprint ML model is serialized in a *Pickle* file to be loaded every time a position needs to be predicted. It is used in the localization algorithm, which will be explained in the next section (subsubsection 4.4.2.4).

Due to the large number of beacons used in an ample space like our workshop, the fingerprinting solution requires more work than the proximity solution regarding maintenance or updates. It requires more work because the training of the machine-learning model depends on the measurements obtained from all the beacons at each measurement point. If it is necessary to increase the number of beacons, change the position of the beacons or replace a beacon with a newer one, the signal strengths of the beacons received at each position in the workshop will be different, and this implies repeating the measurement of the signals at all measurement points, and forming a new features file in order to have a model trained for the new set-up.

Another disadvantage found in the recording of beacons data for the fingerprinting was the difference between RSSI values received when changing the sensor box position angle. This difference will increase the complexity of the training of the ML model because each measurement point will have different RSSI values data depending on the box angle. As this was noticed early on, during the measurement phase, the box was placed at different angles in each recorded data file to minimize the influence of the box angle on the location prediction as much as possible. This influence minimization is essential because asking workers to always place the box at a certain angle would be excessively intrusive.

Also, on a typical working day, the workers' movement in the workshop or the position

of the cars in the workshop may differ from the day when the measurements were taken to build the fingerprinting solution, further complicating the precision of this solution.

4.4.2.4 Localization Algorithm

Knowing how they were built and how the localization solutions used work, we can now describe the script that runs the algorithm responsible for obtaining the beacons data, performing the computations for each solution, and storing the inferred location. The algorithm flow is shown in Figure 4.25.

The script responsible for the localization inferences is always running as a *systemd* service in our remote server that initializes every time the server boots.

The script runs in a cycle. Every cycle gets the last 30 seconds of RSSI values and the correspondent beacon ids from the *InfluxDB* SensorBoxesData bucket. To do that, it uses the *InfluxDB* Python Client Library [38] read operation. If no data is returned, the script "sleeps" for 10s and asks the *InfluxDB* for new beacon data again.

When beacons RSSI data is returned, it calculates the closest beacon by iterating over the received beacons RSSI, looking for the one with the higher value. Then, if the closest beacon does not belong to any zone, "sleeps" for 10s and asks the *InfluxDB* for new beacon data again.

If the closest beacon belongs to a proximity zone, the paint booth 1, paint booth 2, mineral blasting booth, or sanding zone, an entry will be created and sent for a bucket in *InfluxDB* with the name of the Zone and the name of the Location, which in these proximity solution zones is the same, because of its small space.

If the closest beacon belongs to the set of beacons related to the fingerprinting zone, a request is made to the ML model (the pickle file), passing the RSSI values of the received beacons. And then, as above, an entry will be created and sent to a bucket in *InfluxDB* with the name of the Zone, which we called for this zone, Bodywork Zone, and the Location, which will be the ML model predicted point (a fingerprinting point).

If the closest beacon belongs to the critical repairs zone, we consider it as in the Bodywork Zone, so an entry will be created and sent to a bucket in *InfluxDB* with the Zone name - Bodywork Zone - and for the Location, we use a defined position number, compatible with the numbering of the positions used in the fingerprinting solution.

After sending the new Zone and Location inferred, the algorithm "sleeps" for 10 seconds and asks the *InfluxDB* for new beacon data again.

The bucket created is called Predictions, and these entries are sent to the *_measurement* "Location Predictions". Not only the Zone and Location are sent to *InfluxDB*, but information about the sensor box and the car assigned is also sent. The attributes of this *_measurement* will be described next. To send the entries to *InfluxDB*, the Python Client Library write operation is used.

[38] https://docs.influxdata.com/influxdb/cloud/api-guide/client-libraries/python/
Accessed on: 05/09/2022

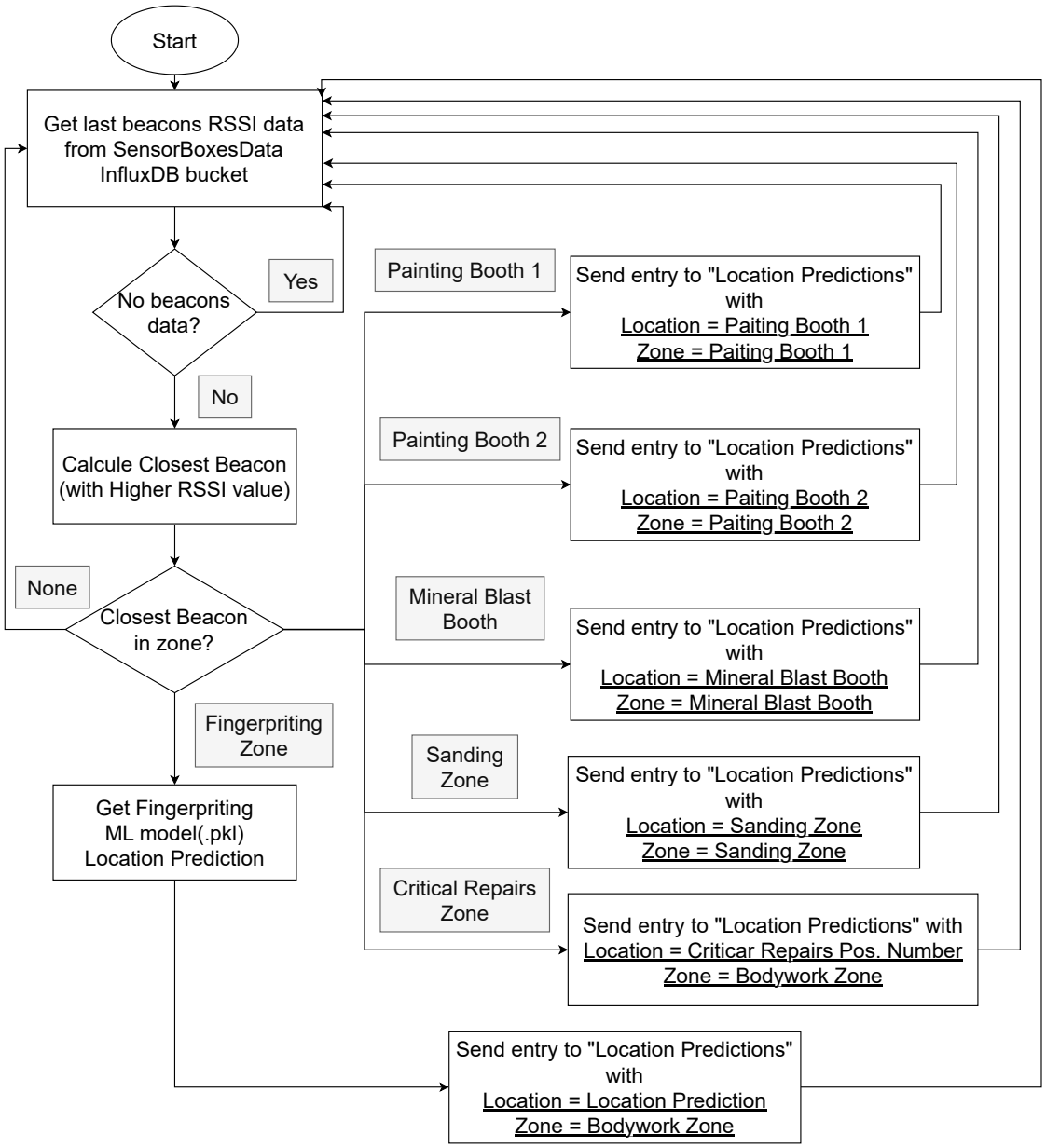

Figure 4.25: Localization - algorithm flow.

The Predictions bucket receives the location predictions and other predictions (i.e., the processes identified by the system, explained later in this document). In this case, we will only present the "Location Predictions" *_measurement* because that is where the localization script sends the data. The localization script sends, per entry, always the following attributes:

- Box Name - the id of the sensor box which recorded the beacons data;
- Car Assigned - the car associated with the sensor box;
- Zone - Prediction of the zone where the assigned car is located;
- Location - Prediction of the location where the assigned car is located;
- _time - (*InfluxDB* self attribute) the entry write action timestamp.

A diagram of the "Location Predictions" *_measurement* attributes is presented in Figure 4.26.

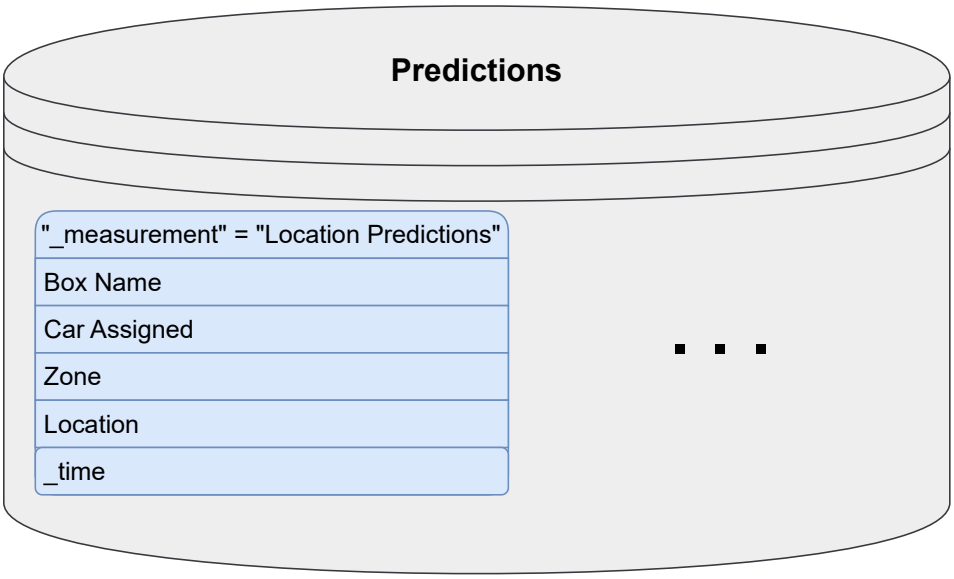

Figure 4.26: Predictions *InfluxDB* bucket. Not all *_measurements* shown.

With the localization description concluded, the next step consists of explaining the electrical load monitoring solution because it is the module requested after localization in most cases (as can be seen in the draft diagram in Figure 4.14).

4.4.3 Load Monitoring Build

As already mentioned, energy meters, including the ones to be installed in the electric panels and the smart plugs to be plugged into the outlets, were installed in the workshop. The first type of meters was installed in the electric panels of the two painting booths and the mineral blasting booth. The others were installed in the outlets around the sanding zone. The meters positions map in the workshop is shown in Figure 4.27.

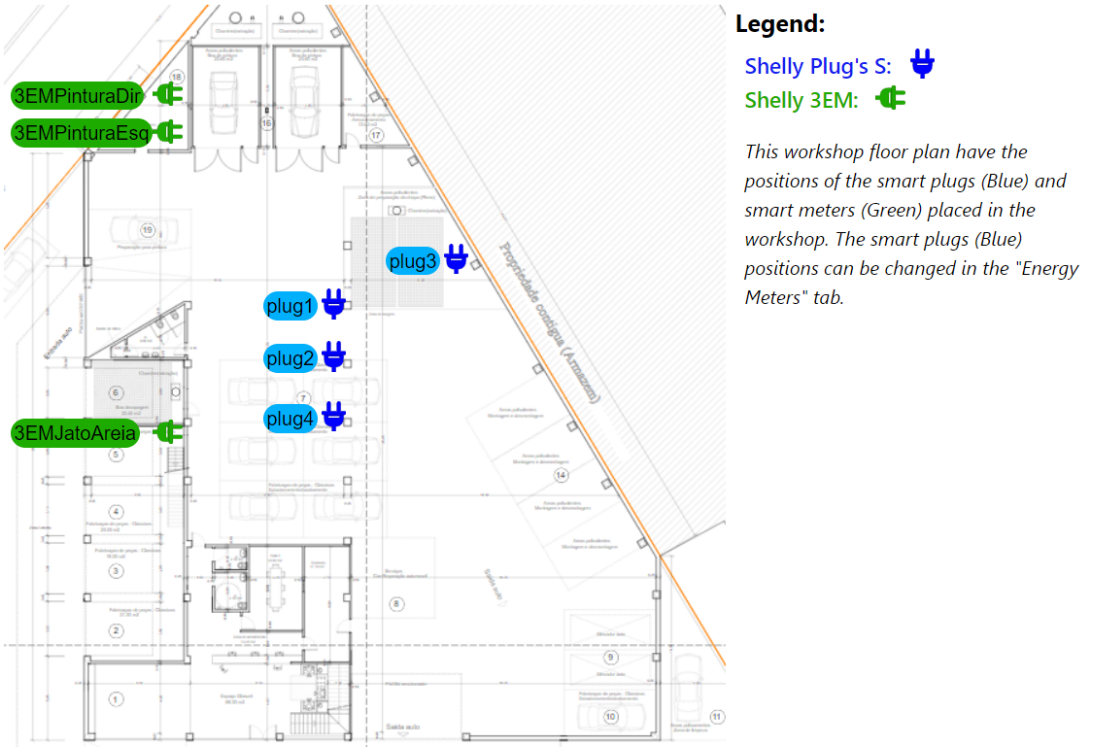

Figure 4.27: Workshop floor plan with the different energy meters position. Screenshot of the web application developed.

As mentioned before, in the introduction of this chapter (section 4.1), the electric conditions in the booths and in the sanding zone are different. In the first case, every booth has a self-electric panel, so we must analyze its consumption to understand what is done inside the corresponding booth. In the other case, different tools can be connected to the smart plugs, so there is a need to differentiate the electric sanders' usage.

So, in the following subsections, we will describe the consumption loads detected in each case and the methods used to differentiate the activities done there. We will separate the description between the painting booths, the mineral blast booth, and the sanding zone.

In this case, contrary to what was done in the localization solution, the identification based on energy consumption is directly requested in the Processes Identification Algorithm explained later in this document (subsection 4.4.5).

4.4.3.1 Mineral Blast Consumption Solution

The first step was understanding the energy consumption or signature of the mineral blasting booth when the process started there.

As said before, the data from the electric meters is available in real-time on *InfluxDB*. From what has been analyzed, for more than one month, in the energy consumption of the booth, it turns out that the pattern is always the same. Two examples are presented

in Figure 4.28. As can be seen, there are only two different main levels of energy consumption. Moreover, by checking in the workshop, only the booth light is on when the consumption level is around 500W, meaning the worker is positioning the car or car parts inside the booth. The higher level of consumption is always correlated with the switching on of the sandblast. This level was never below 1000W and usually keeps close to 1700W.

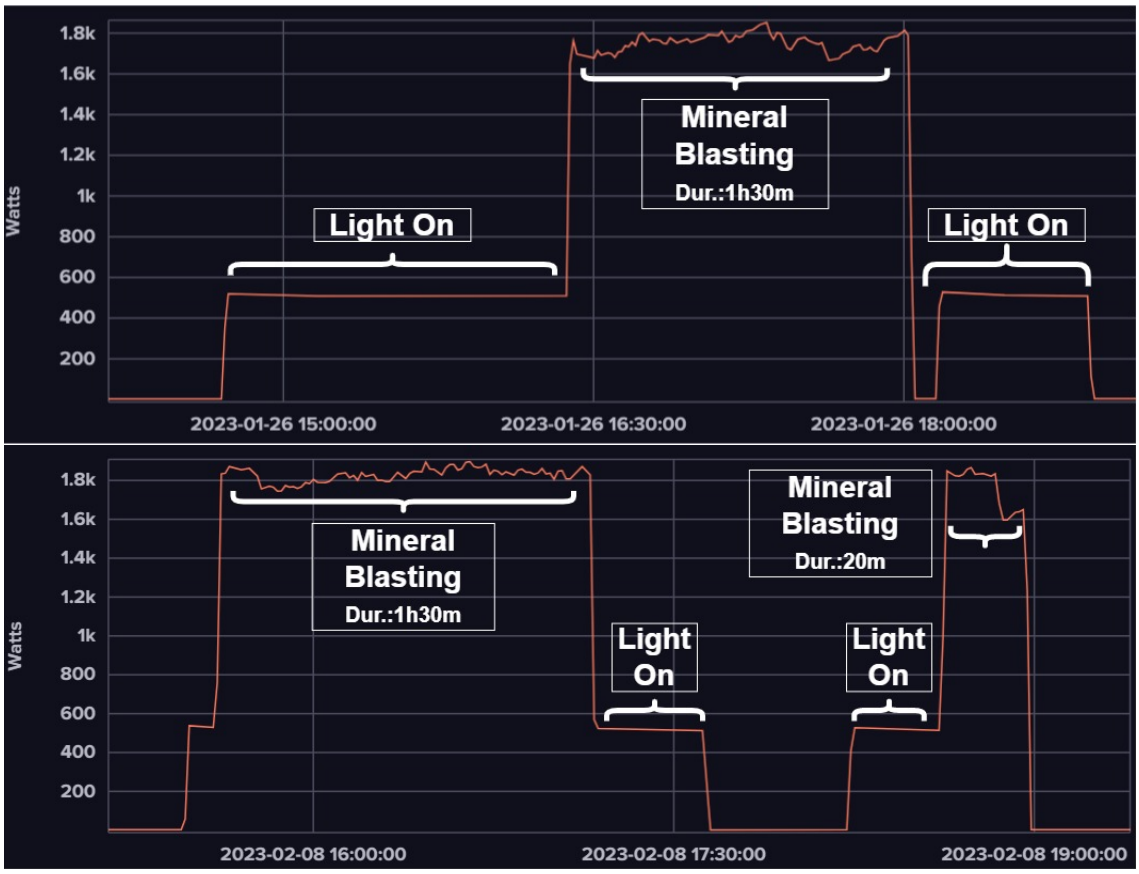

Figure 4.28: Mineral Blast booth electric consumption examples. Screenshots from *InfluxDB* dashboards.

So as already described in the introduction of this chapter, when identifying the different *Charter of Turin Monitor* restoration processes to be inferred by our system, in the mineral blast booth, only the mineral blast activity should be detected (Figure 4.2). So we ignore the light-switching step by seeking to identify only the blasting activity.

The algorithm steps to identify the mineral blast activity are simple and are represented in the diagram in Figure 4.29. First, a query with a time interval, passed in argument to the algorithm, requests the *InfluxDB* Smart Plugs bucket, requesting the *Power* data of the *_measurement* shelly3EM_MineralBlast, corresponding to the booth meter. The query uses the *InfluxDB* Python Client Library. Then the *Power* data returned is iterated, and every time its value is above 1000W, a counter (counterBlast) is incremented. Ultimately, it is verified if the counter value (number of values above 1000W) exceeds 2/3 of the total returned measures. Although values outside the levels previously mentioned have never been verified, we perform this step to ignore the cases where outliers may

appear, as a good practice rule. If it is higher True is returned, and False in the other case. Once again, this algorithm will be used by the Processes Identification Algorithm directly.

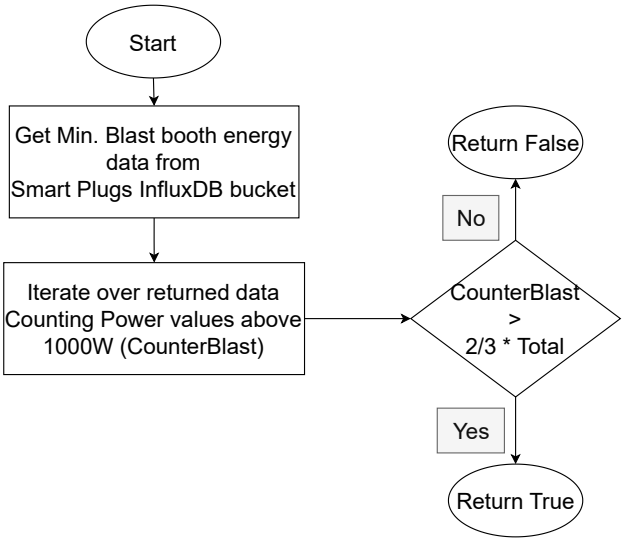

Figure 4.29: Check Mineral Blast Booth Activity - algorithm flow.

4.4.3.2 Painting Booths Consumption Solution

Regarding the painting booths, the first step is also to understand the energy signature and possible energy levels of the booths usage.

So by asking the workshop workers, we understood that the paint booths have two possible main states. The painting state is used when the painting products such as anti-rust, filler, paint primer, or final paint are applied. That is the state that consumes more energy as the booth maintains a high temperature, the compressor, which will allow the spray to be used with the painting product selected, is active, and the ventilation responsible for cleaning the impurities generated when the spray is used is also activated. The other state is selected to perform the curing process, used to dry the painting products applied to the car. The curing state consumes less energy as the booth only needs to maintain a high temperature (60ºC) for some time (usually 30 minutes).

The workers set these states from a switch unique to each booth (Figure 4.30).

So to understand better the energy consumption of each booth concerning the activities carried out there, a study was done on the energy data available in the *InfluxDB* Smart Plugs bucket. We analyzed the records for over a month and compared them with reports of the start and end times of the activities in the workshop. The comparison showed that as well as in the mineral blast booth, there are recurrent patterns.

One example of the left and right booth energy consumption with the corresponding state and activities described are presented in Figure 4.31.

Figure 4.30: Photos of painting booth state selector.

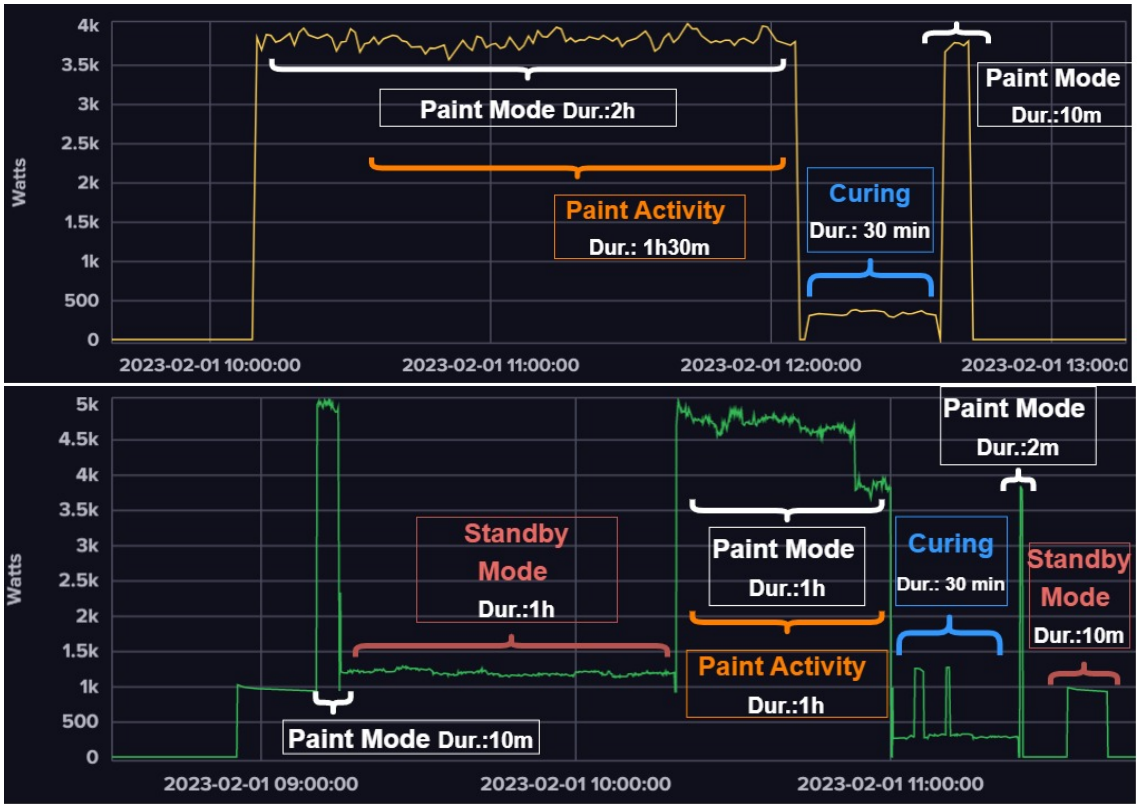

Figure 4.31: Up - Left painting booth energy load. Bottom - Right painting booth energy load. Screenshots from *InfluxDB* dashboards.

The first thing to note is the huge difference in energy consumption between the state for painting and the state used for curing, as well as the stability in the level of consumption in these two states. Although this difference in both booths is relatively equal, it can be seen from the graphs that the right booth consumes more energy in the painting state. This occurs due to its larger size, requiring more energy to maintain the same conditions.

Still, in both booths, in all the registers checked, it can be seen that the state for painting has a consumption consistently above 3000 W and that the state for curing has a consumption between 200 W and 350 W.

In the right booth, another state was revealed, with fixed consumption of around 1000 W. It was understood that it was a standby state, where no restoration activity was done, so this state was ignored. It usually is used to maintain the booth prepared for the next painting process, do some manual action inside, or change the car. So, the painting and curing energy state values are the ones used in the painting booths detection algorithm.

Another important pattern is that the cure always comes after the painting's state.

Although the differentiation of states is simple, as seen in the graphs, sometimes the booth is placed in the painting state, but the worker is not performing the painting activity. For example, in the top chart, although the booth entered the painting state shortly after 10 am, the worker only started the activity at 10:30 am. Moreover, after the curing activity, it goes to paint mode again. Also, in the bottom graph, the painting state is visible around 9:15 am, but the painting activity only started around 10:20 am. Talking to the workers in the workshop, we realized that the booths are often switched on, and left in painting mode, before or during the placing of the car inside, so that when it is time to paint, the cabin is in the necessary conditions for it to start. We were also told that for the same reason, the cabin is sometimes left switched on while the car in the cabin is changed.

Naturally, this problem will result in lower accuracy in detecting the beginning of the painting activity because by looking at the consumption, there is no difference when the activity starts. This lower accuracy is a tolerable problem. However, the fact that sometimes, after curing, the cabin goes again to the painting state will cause the algorithm to identify that painting activity again, which in reality never happens. In this way, there is the need to verify whether the curing has already been performed in each car. If it has already been performed, it will be ignored if the painting activity is detected afterward. However, this part is not treated in this algorithm (only on the Output File Generation algorithm, subsection 4.4.6).

Looking at the different *Charter of Turin Monitor* restoration processes to be inferred by our system in the painting booth (Figure 4.3, Figure 4.4), we understand that based only on the energy consumption of the booths it is not possible to differentiate the four different product applications carried out in the booth, as all the different applications use the same described painting state. An approach was thought out where we check the average time that the different application activities take. However, we were told in

the workshop that this time was variable as it depended on many variables, such as the number of car parts to be worked on or their size, for example.

Therefore, based on consumption, we can only differentiate between the activity of applying painting products, which we call the Painting activity, and the Curing activity. In the Future Work section, we will describe some ideas, out of the electrical scope, for solving this specific identification problem (section 6.2). Thus we realized that the proposed requirement of the system (presented in subsection 4.2.1) to identify and differentiate the application of different paint products would not be met. So we assumed the system capacity only to identify and differentiate the beginning and end time of the Painting activity (using any product) and Curing, as explained.

The algorithm to differentiate the painting booths activities is presented in Figure 4.32. It starts with a query to the *InfluxDB* Smart Plugs bucket, requesting the *Power* data of the *_measurement* corresponding to the booth energy meter passed in argument to the algorithm. This query is done for a time interval also passed in argument. The query uses the *InfluxDB* Python Client Library. Then the *Power* data returned is iterated, and every time its value is above 3000 W, a counter (*counterPaint*) is incremented. On the other hand, if the value is between 200 W and 350 W, another counter is incremented (*counterCure*). These values and ranges come from the consumption study described above. In the end, it is verified if the *counterPaint* value is greater than 2/3 of the total of returned measures, returning, if True, the string "Painting" (corresponding to the Painting activity). If that is not the case, it is verified if the *counterCure* value is greater than 2/3 of the total of returned measures. If True, the string "Curing" (corresponding to the Curing activity) is returned. If no condition is met, the algorithm returns an empty string. Once again, the 2/3 condition disregards outliers in the middle of an activity.

4.4.3.3 Sanding Zone Consumption Solution

Relative to the sanding zone load monitoring, the goal was to identify only the electric sanders' work.

So a simple solution identified consisted of installing a smart plug in every electric sander's hoover. This would work because every sander is connected by a tube to a hoover that accompanies the sander throughout the workshop, Figure 4.33. The sander is always connected to the hoover. So the setup would consist of installing the smart plug in the hoover's outlet where the sander is connected.

A smart plug for each hoover would be a simple solution. However, with a view to future work, looking to identify more than the sanding tool, we decided to implement an ILM solution. So, a more complete tool identification system was developed, capable of being easily expanded in future works.

For this tool identification system, we sought to differentiate the tools that are mostly connected to the sockets in the sanding area, with particular attention to the distinction of the sander. Naturally, the most commonly used tool is the sander with its hoover, but

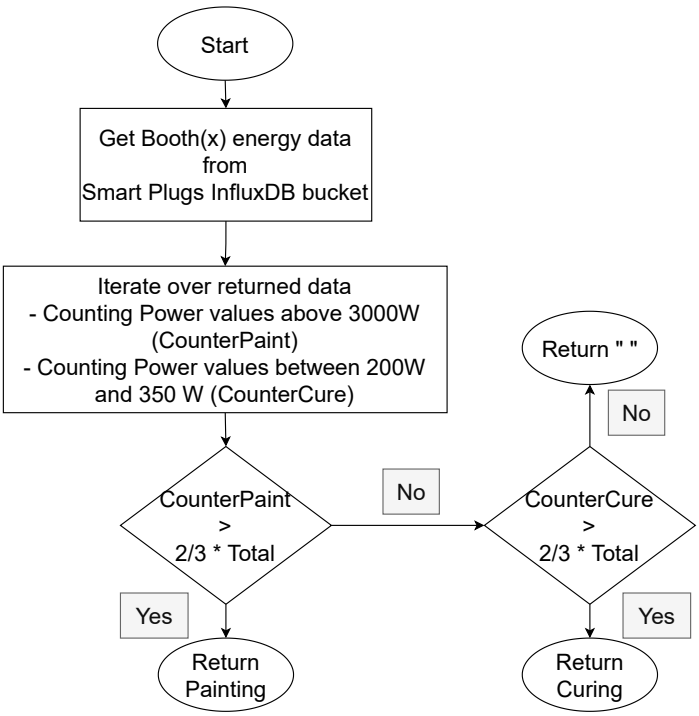

Figure 4.32: Check Painting Booths Activity - algorithm flow.

Figure 4.33: Electric sander and its hoover.

also hot air blowers and angle grinders are sometimes connected there (to be used in the adjacent areas).

So to build the ILM, as described in the Related Works chapter (section 2.2), three main steps are needed: Energy Data Collection, Feature Extraction, and Classification. So we will describe next these three steps applied to our problem.

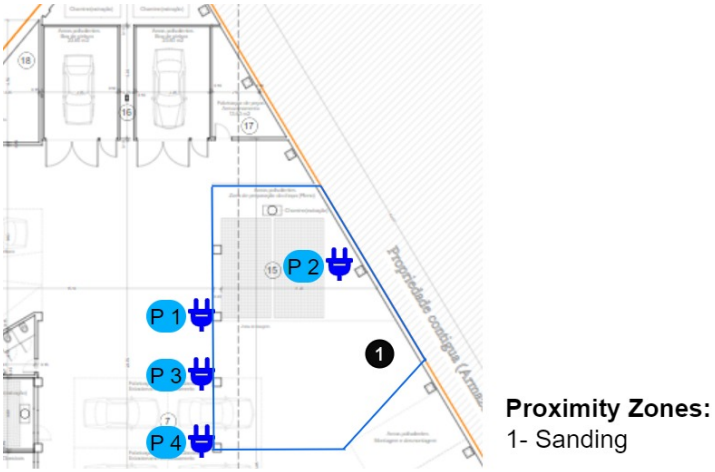

Figure 4.34: Smart Plugs (Shelly Plug S) installed positions around sanding zone.

So the energy data collection is done by the smart plugs Shelly Plug S. As mentioned, we installed them in all the outlets around the sanding zone, Figure 4.34.

Then, the feature extraction step is needed because the tools involved belong to type 2, tools with different power states (different types in section 2.2). So the developed features extraction is based on the techniques described in [9] and [21]. Given a time window of power data, it calculates all features regarding power levels and variations. So nine features were chosen based on the previous works [9], [21]:

1. Maximum power value;
2. Minimum nonzero power value;
3. Mean power for nonzero values;
4. Number of samples with power less than or equal to 30 W;
5. Number of samples with power between 30 and 400 W;
6. Number of samples with power between 400 and 1000 W;
7. Number of samples with power greater than 1000 W;
8. Number of power transitions between 10 and 100 W;
9. Number of power transitions greater than 1000 W.

These features seek to capture all relevant information from the data received. Features 1 to 3 capture more general and simple information. Features 4 to 7 will detect the time duration a tool uses more or less electric power. Features 8 and 9 aim to detect the transition sizes of power a tool tends to do.

Each time window of energy data will generate an entry that serves as input to the ML model.

Regarding the time window, when we check the activation duration of the tools in question when they are used, it is around one minute long. Therefore, we decided to use a 60-second time window. We did not test different time windows. However, in case of an increase in the number of tools to be identified by the ILM solution, increasing its complexity, it would be a good strategy to perform this step.

Regarding classification, to build the ML model, we built a features file containing ten feature entries from the energy consumption of each of the three different tools during a typical working day in the workshop. Then we built the training set by randomly choosing seven feature entries from each tool, and the three remaining entries were used for the test set.

Then we trained and tested the model with the more accurate ML algorithms used in the literature (presented in the Related Work chapter, section 3.2), i.e., the RF, KNN, DT, GNB, GB and the FFNN. These six different supervised ML algorithms were implemented, tested, and compared.

In subsection 3.2.1, the importance of data normalization and features classification and selection is also noted. So these actions were done using some *Sklearn* [39] libraries. However, there were no improvements in the results.

As seen in the Table 4.4, 100% accuracy was achieved with three different algorithms. This percentage of correctness is justified by the considerable differences between the three tools' energy signatures.

Being equal to choosing any of the algorithms that had the result of 100% accuracy, we decided to use KNN in the system architecture.

After being trained, the tool identification ML model is serialized in a *Pickle* file to be loaded every time a tool needs to be predicted.

Table 4.4: Accuracy of the tested ML algorithms, for ILM.

Algorithm	Accuracy
KNN	100%
DT	100%
RF	97.2%
FFNN	92.3%
GB	100%
GNB	84.6%

[39] https://scikit-learn.org/stable/ Accessed on: 28/03/2023

With the ILM ML model building description, we can now present the algorithm used to get the power data, extract features and verify if an electric sander was used, Figure 4.35.

The algorithm starts by querying the *InfluxDB* Smart Plugs bucket, requesting the *Power* data of all the smart plugs connected in the workshop. This query is done for a time interval passed in the argument to the algorithm. Again, the query uses the *InfluxDB* Python Client Library. Then as the *Power* values are returned grouped by the corresponding smart plug, a dictionary "valuesByPlug" is built, iterating the data returned. This dictionary will contain as key the smart plug's names and, as values, the *Power* values of each plug. Then, by iterating over the "valuesByPlug" dictionary, the features resulting from the *Power* data of each plug are calculated for each entry. So, another dictionary is built, named "featuresByPlug", containing the corresponding features for each smart plug name. Finally, an iteration of this dictionary is done. For each plug, the corresponding features calculated are passed as input to the ILM ML model, available in the *Pickle* file, so that it can infer the tools used in each plug. If any of the identified tools is an electric sander, the algorithm ends by returning True. If no inferred tool is a sander, it returns False.

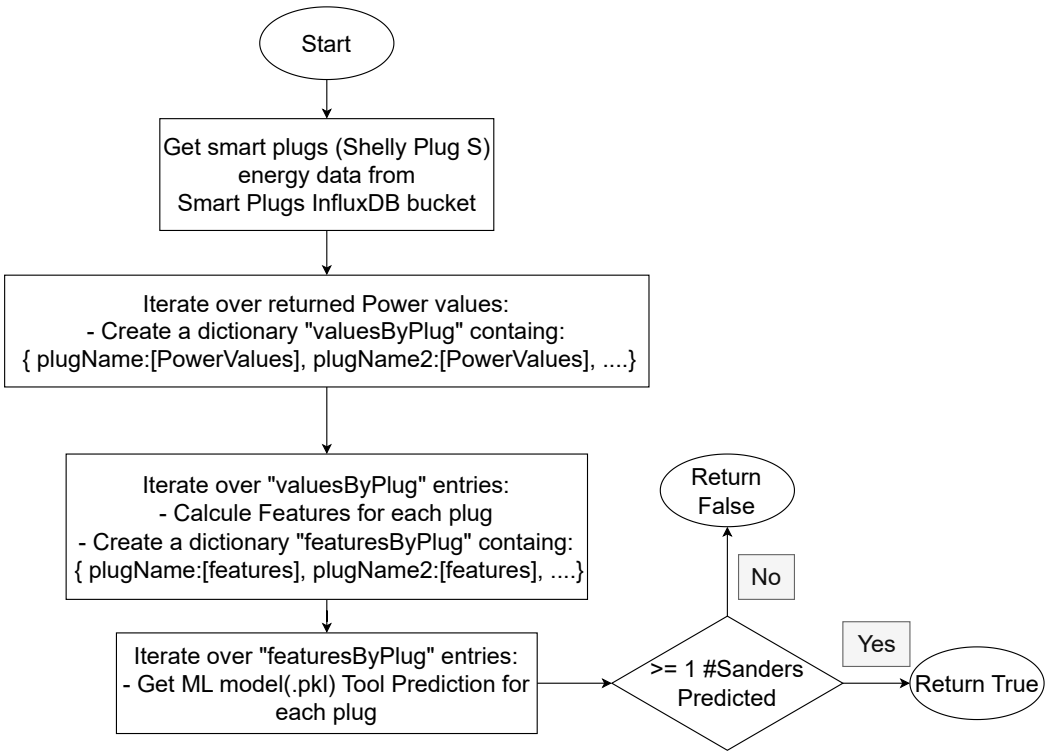

Figure 4.35: Check Electric Sanders Activity - algorithm flow.

4.4.4 Vibration Detection

Another step required in the processes identification algorithm, regarding the sanding zone, is the sensor box vibration detection, as seen in the diagram of Figure 4.14.

This detection is needed because the system needs to know in what car an electric sander is being used. For the vibration detection to work, the worker using the tool must attach the box to the car panel. The attachment must be made to a panel on the same side of the car where the panel being worked is.

So the vibration detection algorithm, Figure 4.36, starts by querying the *InfluxDB* SensorBoxesData bucket, requesting the frequency measures of the sensor box name passed in argument to the algorithm for a time interval also passed in argument. Using the *InfluxDB* Python Client Library to do the query.

Then the frequency values are iterated, and every time a frequency value is higher than 0, a counter (*counterVibrations*) is incremented. Ultimately, the algorithm returns True if the counter value exceeds half of the total frequency values. If it does not happen, then the algorithm returns False. This verification is done only to return true when most of the returned frequencies in that time interval are above 0. This verification disregards outliers or slight movements in the sensor box that does not represent vibration produced by tools.

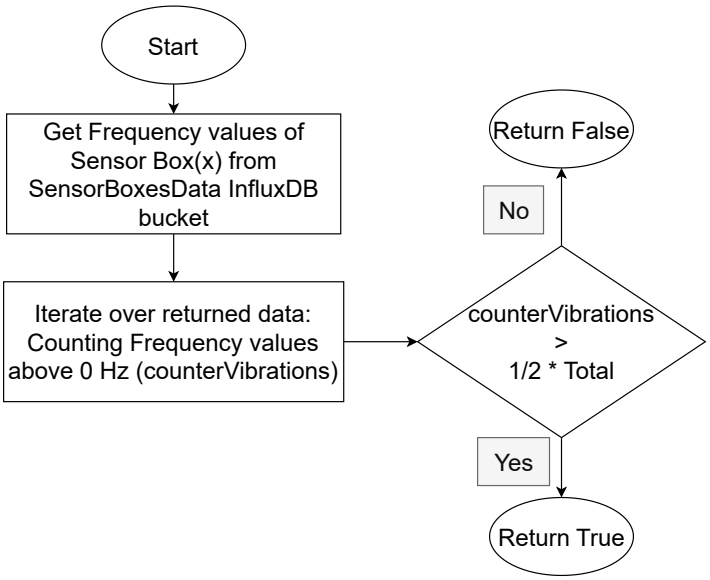

Figure 4.36: Check Box Vibration Activity - algorithm flow.

4.4.5 Processes Identification Algorithm

The Processes Identification Algorithm is where all the algorithms explained above are required in order to be able to infer which restoration processes each car, with an attached box, goes through in the workshop.

The processes identified will be available in the *InfluxDB* Predictions bucket already presented, in a new *_measurement* called "Processes Identified". The characteristics of this *_measurement* will be explained later in this subsection.

As the work requirement was to stay close to real-time predictions, we decided to use last-minute data to do the calculations. Because with a range of one minute, we can

perform calculations of consistency to reduce the possibility of error in the algorithm. For example, infer the box's location, using the location predicted more times during that minute, thus reducing the influence of outliers or noisy wrong predictions.

This minute time is also compatible with the ILM solution time window (60 seconds), resulting in a correct and consistent integration when this solution is required.

Even so, as the activities performed in the workshop last, in most cases, at least 30 minutes, this time range can become more extensive. However, the process identification would get increasingly far from being real-time identification.

So a python script containing the algorithm is always running as a *systemd* service in our remote server. The script runs in a cycle. Every cycle uses data from the last minute, does the calculations, saves the processes identified by each sensor box, "sleeps" for 40 seconds, and starts again. We choose 40 seconds as an intermediate time because, being in the time range of 60 seconds, we should not wait near it not to lose data, and on the contrary, the more we reduce the sleep time, the more repetitive data we will get in each cycle.

Now the algorithm will be described. It is also represented in the Figure 4.37 flow diagram. The blue color squares represents the algorithms explained previously.

The algorithm starts by querying the *InfluxDB* Predictions bucket, requesting the last-minute "Location Predictions" *_measurement* entries. It organizes the returned data in a dictionary containing as key the sensor box names. This dictionary will have the following structure {boxName1:{zones:[z1,z2,z3], locations:[l1,l2,l3], carAssigned:"car1"}, boxName2:{zones:[z2,z1,z3], locations:[l6,l5,l4], carAssigned:"car2"}, ...}.

Then if the dictionary is not empty, it iterates over it, and for each box data/entry, it starts by calculating the confidence zone. The confidence zone will be the zone that appears more than 2/3 of the total number of zones predicted at the last minute. We use this calculation because, as the localization solution works with beacons, the signal strength may have slight fluctuations, or some person or object could have passed between the box and the beacons. These factors could generate some wrong variations in the predictions, so we are reducing the influence of these noises by doing this calculation.

The following steps depend on the confidence zone calculated.

In case the confidence zone is the mineral blast booth, a request to the Check Mineral Blast Activity algorithm (described in subsubsection 4.4.3.1 and represented in Figure 4.29) is done passing the last minute time interval as an argument. If it returns True, it means the Mineral Blasting process was identified. Then, a new entry with the mineral blasting activity is sent to the Predictions *InfluxDB* bucket, in the *_measurement* "Processes Identified". If it returns False, the algorithm starts the calculations for the following boxes' data, or if there are no more, it "sleeps" and starts again.

In case the confidence zone is one of the painting booths, a request to the Check Painting Booths Activity algorithm (described in subsubsection 4.4.3.2 and represented in Figure 4.32) is done passing the last minute time interval and the booth name (booth 1 or 2). Then if the activity returned is Painting or Curing, a new entry containing the

activity is sent to "Processes Identified" in *InfluxDB*. If it returns None, the algorithm starts the calculations for the following boxes' data, or if there are no more, it "sleeps" and starts again.

In case the confidence zone is the Sanding Zone, a request to the Check Box Vibrations algorithm (described in subsection 4.4.4 and represented in Figure 4.36) is done passing the last minute time interval and the actual data sensor box name, to verify if some car panel is vibrating/being worked on, in that zone. If it returns True, a request is made passing the same time interval to the Check Electric Sanders Activity algorithm (described in subsubsection 4.4.3.3 and represented in Figure 4.35) to verify if some electric sander caused the vibrations. If it returns True, the Sanding activity is detected, so a new entry with that activity is sent to *InfluxDB* "Processes Identified" *_measurement*. If some of the last two requested algorithms return False, the algorithm starts the calculations for the following boxes' data, or if there are no more, it sleeps and starts again.

Finally, if the confidence zone is one of the other zones that we call the "Bodywork Zone" (where the fingerprint solution is mainly requested), our system can not identify any process there. Thus we are only interested in getting the correct location of the box. So, to mitigate the fingerprinting errors and possible consequent variations, a calculation of the location point prediction that appears more in the last minute is done (majority location). This way, the system will be immune to small fluctuations of the location points inferred by the location solution. Then a new entry with the activity called Bodywork Activity is sent to *InfluxDB* "Processes Identified" *_measurement*.

After a new entry is sent to *InfluxDB*, the algorithm starts the calculations for the following boxes' data, or if there are no more, it "sleeps" and starts again.

It is important to note that, as already mentioned, the called majority location point is only calculated in the named Bodywork Zones because the fingerprint solution is mainly requested there. In the other zones, the location is the same as the zone (as described in subsection 4.4.2).

As mentioned, the processes identified entries are sent to the Predictions *InfluxDB* bucket to a new *_measurement* called "Processes Identified". So every time a process is identified, a new entry is created and sent to *InfluxDB* through its Python Client Library.

The "Processes Identified" *_measurement* contains the following attributes:

- Box Name - the id of the sensor box corresponding to the process identified;
- Car Assigned - the car associated with the sensor box;
- Activity - the restoration activity/process identified;
- Zone - the zone where the activity/process was carried out;
- Location - the location position where the activity/process was carried out;
- t_Initial - As in every iteration of the algorithm, the last minute data is used, we decided to save the start timestamp of the interval.

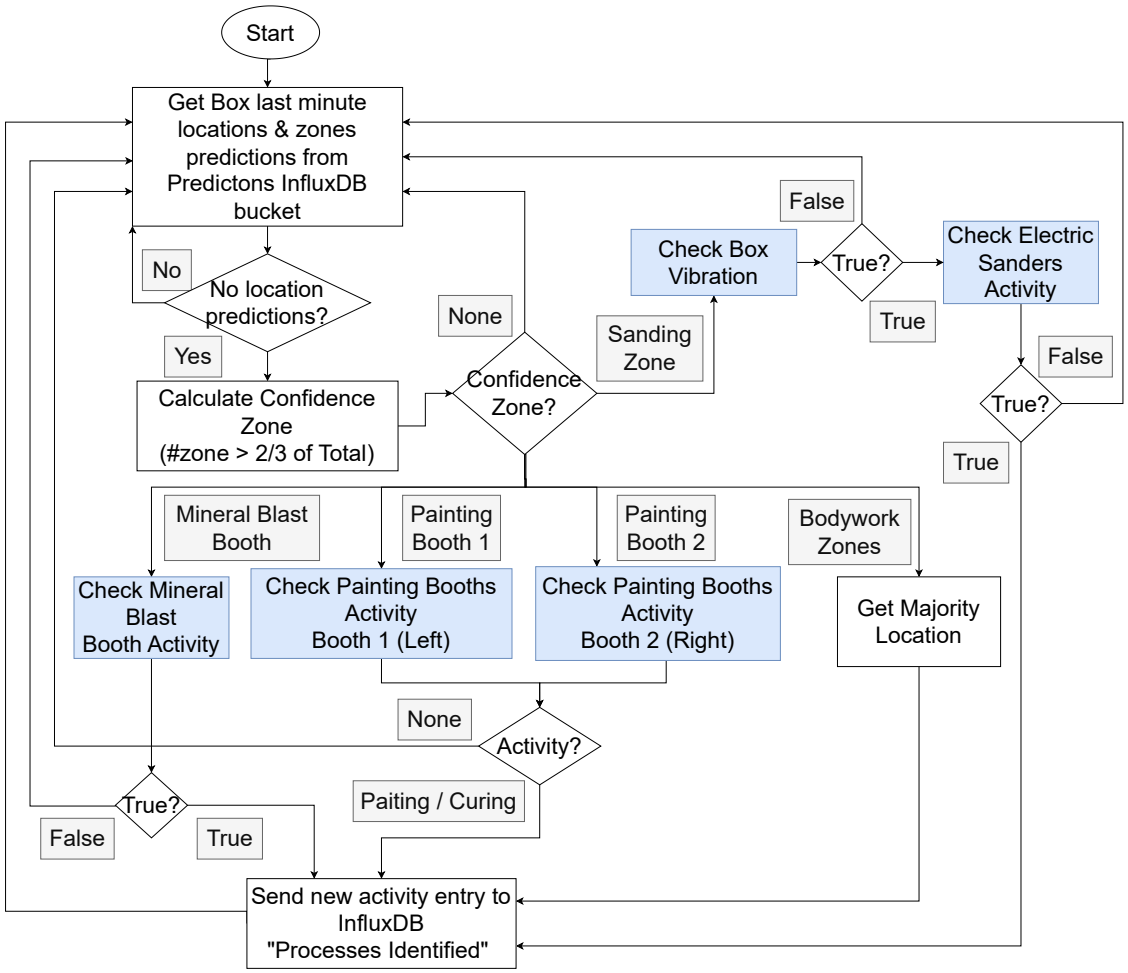

Figure 4.37: Processes Identification Algorithm - algorithm flow. Runs for each sensor box data retrieved.

- _time - (*InfluxDB* self attribute) the entry write action timestamp. In this case, the algorithm sets this value as the end timestamp of the one-minute interval.

A complete Predictions bucket diagram, containing the "Processes Identified" *_measurement* attributes, is presented in Figure 4.38.

In conclusion, the algorithm can send five different types of entries to *InfluxDB*. The entry with Activity being Mineral Blast, where the Zone and Location will be Mineral Blast Booth. An entry with the Activity being Painting, where the Zone and Location will be Painting Booth 1 or 2. The entry with the Activity being Curing, where the Zone and Location will be Painting Booth 1 or 2. The entry with the Activity being Sanding, where the Zone and Location will be Sanding Zone. Finally, an entry with the Activity being Bodywork Activity where the Zone will be Bodywork Zone and the Location will be the point of the workshop identified by the localization solution.

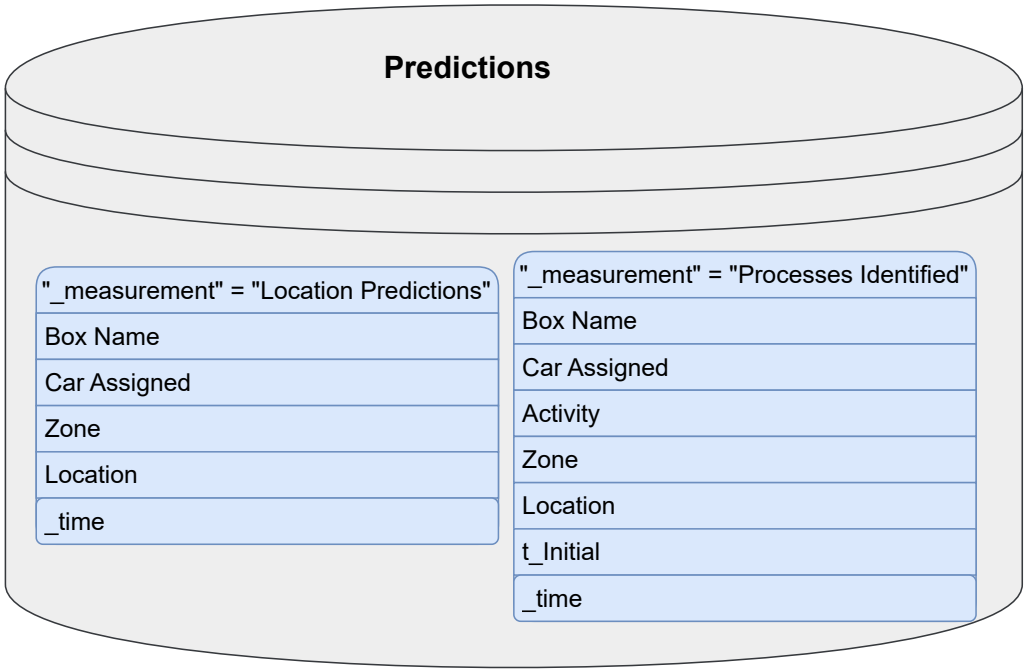

Figure 4.38: Predictions *InfluxDB* bucket. With the "Processes Identified" *_measurement* added.

4.4.6 Output File Generation Algorithm

The previously mentioned Processes Identification Algorithm is an algorithm that continuously detects new processes, generating process inferences to *InfluxDB* every minute. As was mentioned, to avoid losing the concept of live data, the algorithm only evaluates data relating to the last minute. However, as described in the painting booth consumption solution (subsubsection 4.4.3.2), this algorithm alone can raise some errors, as it does not perceive previously identified processes (in the case of the paint booth, after a curing activity, painting is never done again without the car going through another activity first).

In summary, the Output File Generation algorithm will obtain all the processes identified by the Processes Identification Algorithm during one day of work. Then, it will work on this data with a notion of process sequence present in the *Charter of Turin*.

Thus, what is planned for this algorithm is the full integration of the processes detected by our system with the processes registered in the *Charter of Turin Monitor* developed in [20] work. There should be a constant sending of the processes detected by our system to the model. This way, we could have a sequence notion of the set of possible restoration processes to be performed next on a particular car, restricting our system only to detect those same processes and reducing the error and ambiguity in identifying some processes.

However, in the *Charter of Turin Monitor* [20] work, only one endpoint was created in the API of that system with this goal in mind. This endpoint returns the list of cars inserted in the system, with the corresponding list of processes already performed on

the car. There is no endpoint to update the *Charter of Turin Monitor* with a new detected process. Due to the deadline of our work and the fact that the system developed in [20] was out of our scope, this missing endpoint was not developed (reason to be added to future work, section 6.2).

So concerning the *Charter of Turin Monitor*, we will only be able to know which activities have already been registered in the model for each car. However, it should be noted that restoration activities performed on cars are not registered in the model as soon as they are completed. They can sometimes only be registered weeks later by the workshop supervisor. Therefore the need for our system, as already stated.

This way, as we cannot update and keep track of the activities already performed in a given car, detected by our system, directly in the *Charter of Turin Monitor*, we decided to create a local dictionary to be used in this algorithm. In the dictionary, we will keep, for each car, the activities already detected during the execution of the algorithm and thus limit which activities can be detected next.

So, given our system, we gave each Processes Identified Algorithm possible activity a number with the supposed order. Then, we need to check that the Mineral Blast activity, when detected, needs to be the first one, and a Curing activity always follows up the painting activity (never the other way round).

So the algorithm will be explained next. It runs in background, in our remote server, as a *systemd* service. It runs once a day, collecting all the activities identified, during that day, by the Processes Identified Algorithm (explained in the previous section, subsection 4.4.5) and will generate an output *.json* file for each existent sensor box, being available for download in the system web application.

It starts by queering the *InfluxDB* Predictions bucket, requesting the last-day processes/activities identified by the system. Then it groups the data by sensor box name and iterates it. In the iteration over each sensor box activities identified, the first thing that is checked is if the car associated with the identified activity is part of the projects already created in the *Charter of Turin Monitor*, through that system's API. If it is, it is verified if the identified activity has been concluded before or if that phase of restoration has already been done. If so, the activity is ignored, and the algorithm moves on to the next. If the car is not registered in the system or the activity is in the regular restoration order, the algorithm goes on to the next check.

The next check consists in verifying through the local dictionary of the already done activities of each car (during the day) if the detected activity is according to the expected regular order. If so, the algorithm proceeds to the last check.

The last verification relates to how activities are written in the output *.json* file. An activity detected, with its associated car, sensor box, location, zone, initial timestamp, and end timestamp, is only written to the file if the last activity written has a different car assigned or is of a different type. In the opposite case, these values being the same, only the end timestamp of the last activity written is changed to the end timestamp of the current activity. This writing method was adapted from an algorithm in the previous

work [22].

This last verification is done because the output file should only contain each identified activity's start and end timestamp, concatenating all the records of each activity generated by the Processes Identification Algorithm. A screenshot of an output file example, generated on 09-02-2023 day, is shown in Figure 4.39, where a painting followed by a curing process was identified. For example, the Curing activity written in the file lasted 10 minutes, but it is an agglomeration of various curing activities detected by the Processes Identification Algorithm.

The output files should be sent to the *Charter of Turin Monitor* platform, integrating our work with *Charter of Turin Monitor* work [20], so the processes identified by our system could be marked as done in the *Charter of Turin Monitor* platform, eliminating the need for such registration to be done by the workshop supervisor. However, as explained at the beginning of this section, this integration is part of future work.

The output files will be available also in the web application.


```
[
{       "BoxName": "SensorBox_07",
         "Car": "Lancia Flaminia 1960 - GOI XVE",
        "Location": "P32",
        "Zone": "Bodywork Zone",
        "Activity": "Bodywork Activity",
        "StartTimestamp": "2023-02-09T09:21:48Z",
        "EndTimestamp": "2023-02-09T09:25:07Z"},
{
        "BoxName": "SensorBox_07",
        "Car": "Lancia Flaminia 1960 - GOI XVE",
        "Location": "Painting Booth 1",
        "Zone": "Painting Booth 1",
        "Activity": "Painting",
        "StartTimestamp": "2023-02-09T09:36:52Z",
        "EndTimestamp": "2023-02-09T10:11:50Z"},
{
        "BoxName": "SensorBox_07",
        "Car": "Lancia Flaminia 1960 - GOI XVE",
        "Location": "Painting Booth 1",
        "Zone": "Painting Booth 1",
        "Activity": "Curing",
        "StartTimestamp": "2023-02-09T10:11:52Z",
        "EndTimestamp": "2023-02-09T10:20:38Z"},
{
        "BoxName": "SensorBox_07",
        "Car": "Lancia Flaminia 1960 - GOI XVE",
        "Location": "P7",
        "Zone": "Bodywork Zone",
        "Activity": "Bodywork Activity",
        "StartTimestamp": "2023-02-09T10:22:01Z",
        "EndTimestamp": "2023-02-09T16:45:09Z"}
]
```


Figure 4.39: Output file generated example.

4.4.7 Web Application

As was already introduced in the introduction of this chapter, in section 4.1, the web application developed in the previous work [22] needed to suffer some changes and be

extended to offer more features and to support the new sensor devices. So we will present next, a comparison between the old version and the new version of the web application. Followed by a description of some specific features created and their meaning, the alarms generation and then the backend built to support those features and frontend tools will also be presented.

4.4.7.1 Old Version vs. New Version

The comparison between the two versions is presented in Table 4.5. All the new features meet the web application requirements, described in subsubsection 4.2.1.2.

Table 4.5: Main differences between web application features of the old and new version.

Old Version	New Version
Backend running in AWS	Backend running in Private Remote Server
Frontend running in *Netlify*	Frontend running in Private Remote Server
Authentication with AWS Cognito	Authentication built manually
Access sensor boxes state (Connected/Not Connected) —— Start/Stop sensor box from recording data Change sensor box assigned car Change sensor box responsible Change sensor box sensors configurations Access inferences of the restoration processes identification system Access sensor box alarms	Access sensor boxes state (Connected/Not Connected) Access the state of the sensor box vibration sensor (Connected/Not Connected) Start/Stop sensor box from recording Data Change sensor box assigned car —— —— Access the daily register of restoration processes identified in a file Access sensor box alarms
Access and change beacons status (Coupled/Decoupled) Access beacons zone Access beacons battery alarms	Access and Change beacons status (Coupled/Decoupled) Access and Change beacons exact position with a dynamic map Access beacons battery alarms
Access, add, and change the configuration of tools frequency signatures	Not Applied
Not Applied	Access energy meters status (Connected/Not Connected) Access and Change meters exact position with a dynamic map Access meters connection alarms
——	Access live map with last processes identified
Access sensor box log files	Access sensor box log files

All the web application pages are presented with screenshots in the appendix (Appendix C).

Interpreting the table now, the first thing to note is the transfer of the backend and

frontend to our remote server. This transfer was done because of the requirement to cease using pay-per-use platforms, as explained at the beginning of the chapter. The development related to this migration will be explained below, in subsubsection 4.4.7.3 and in subsubsection 4.4.7.4.

Due to the same reasons, user authentication is no longer done through the AWS *Cognito* service but through the developed backend, which stores usernames and passwords in our database and generates tokens to be used as a cookie in the web browser. Access security was not a priority at this phase, so the access characteristics are basic and straightforward, as the password and tokens are not encrypted. Because of that, this security topic is discussed in the Future Work section (section 6.2).

Regarding the Sensor Boxes, the main features are the same, but some differences exist. The first one is the new live information related to the connection state of the vibration sensor to the box. Helping the user to know if the box is prepared to be used. A screenshot of the sensor box details page containing this feature is shown in Figure 4.40.

Another difference lies in some available box configurations, i.e., in the old version, the time between measurements in the sensor box and the vibration sensor sampling frequency could be changed. However, we understand that these changes would change the system's accuracy, and testing should be done before setting these configurations. This way, some of these configurations that could change the final result of the system were removed from the application. Also, defining the person responsible for the box was considered unnecessary, as this information is now irrelevant.

The last difference is related to the processes' identified output files. In the old version, a new output file was generated when the number of inferred activities reached a specific number. In the new version, a new file is automatically generated at the end of the day through a scheduled request to the backend.

Related to beacons, the most significant evolution had to do with creating a dynamic map with the various possible positions for the beacons and the possibility to define and change, for each beacon, the corresponding position within a set of available positions, Figure 4.41. Besides that, a dynamic map page was added, allowing the user to see which beacons have low battery and additional beacon information when the mouse is placed over the relative pin, Figure 4.42.

Relative to the tools' frequency features in the previous version, they have all been removed, as our system does not use them.

Because the energy meters were added to our system, an expansion of the web application to support it needed to be done, so features equal to the ones available for the beacons were also added for the energy meters. One can access the list of energy meters and see their status (if they are connected to the *Wifi* or not). A dynamic map with the available positions and the possibility of setting the meters for those positions is also offered. Moreover, alarms generated when some meter loses connection can be accessed too.

The last new feature added was the live map with the last activities/processes identified by each sensor box by the Processes Identification Algorithm. A screenshot of the live map in the web application is shown in Figure 4.43. The map will update every 5 seconds and show the position of each sensor box, and when mouse over it, the car associated, the restoration activity identified, and the detection timestamp.

A video walkthrough of the web application where its features are shown is available at here.

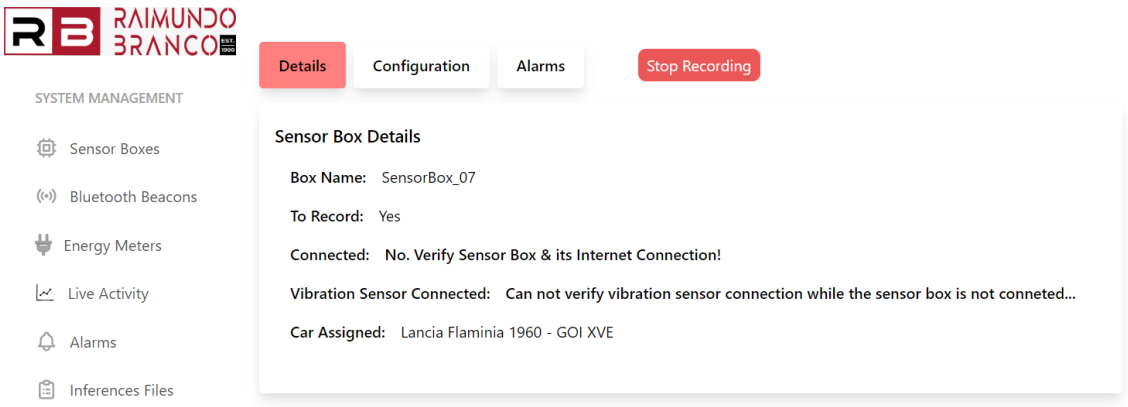

Figure 4.40: Sensor box details page, screenshot from WebApp.

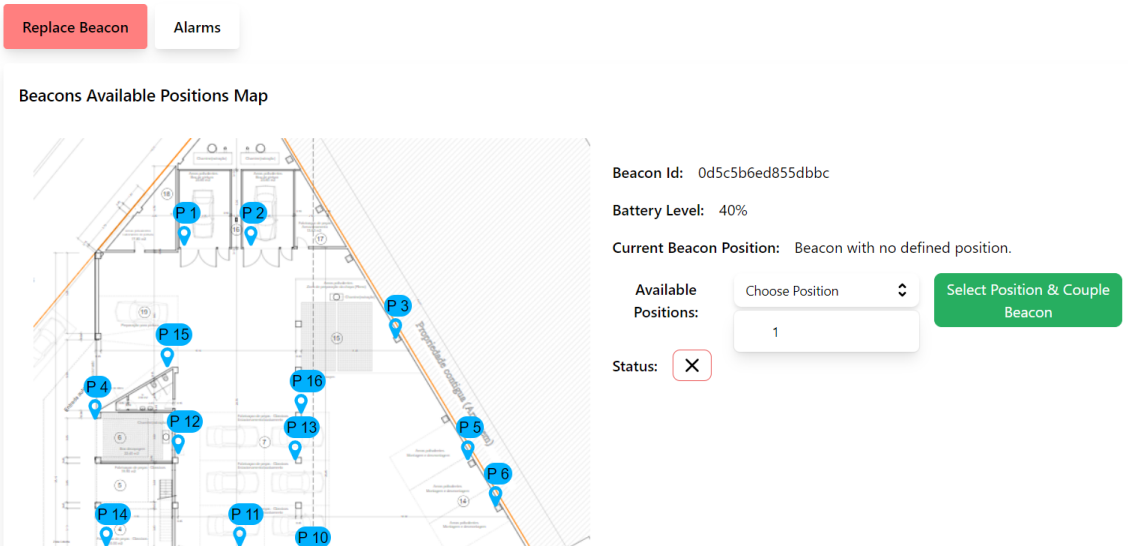

Figure 4.41: Beacons dynamic replacement page, screenshot from WebApp.

4.4.7.2 Alarms

As described above, the web application users can see the alarms generated to better monitor the system, as this is the application's main goal. So next, the different alarm types and its goal will be enumerated:

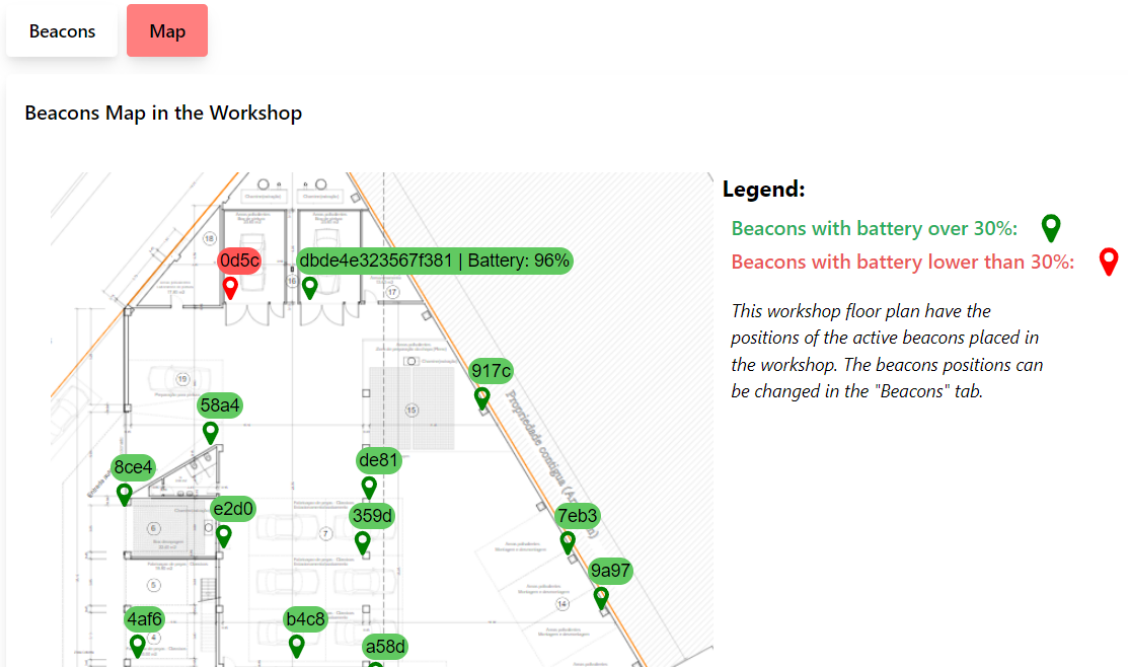

Figure 4.42: Beacons dynamic map page, screenshot from WebApp.

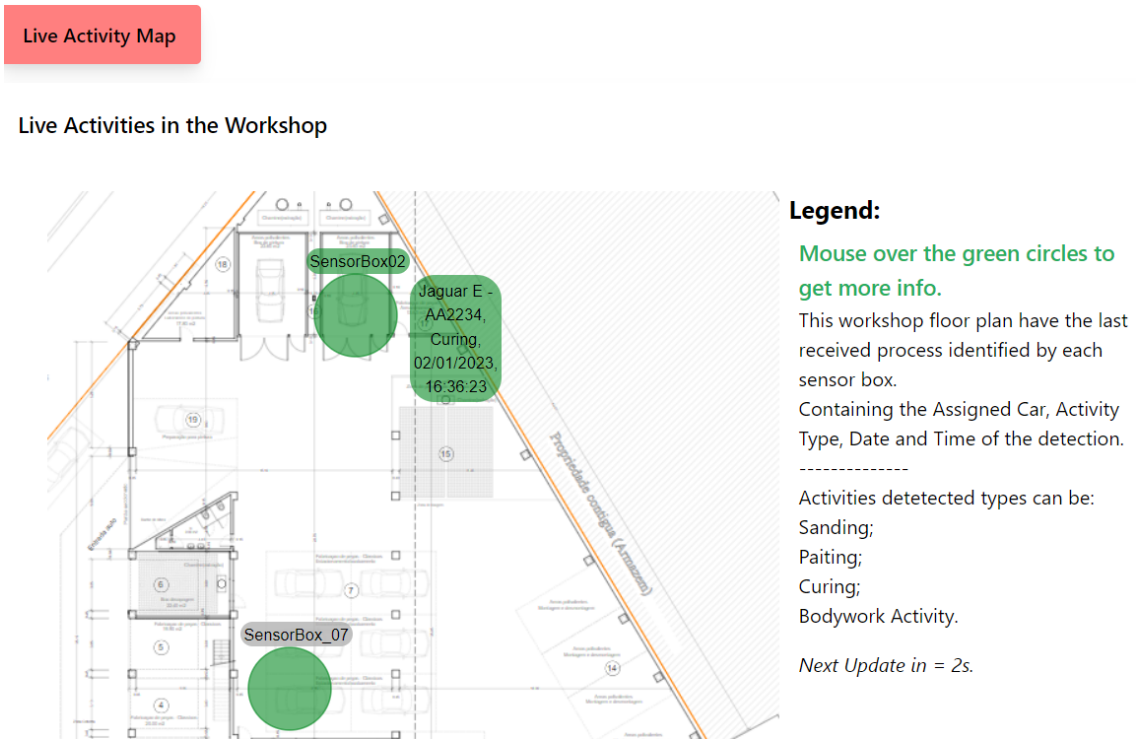

Figure 4.43: Activities live map page, screenshot from WebApp.

- Sensor Box X not detected - To alert for some internet connection problem or battery disconnection in the sensor box X;

- Vibration Sensor of Sensor Box X not detected - To alert for some cable connection problem between the box and the sensor;

- Energy Meter X not connected - To alert for some energy meter that was unplugged or lost internet connection;

- Beacon X with low battery - To alert for some beacon with the battery level below the minimum (30%).

Different approaches were used to implement the alarms.

For the energy meters and sensor box alarms, the *InfluxDB Alerts* service was used.

This service, as explained in the technologies section (subsubsection 4.3.3.1), allows the creation of alerts to detect when no data arrives during some range of time (called deadman alert) and to detect when the data received is between or equal to some values (called threshold alert). Moreover, when some of these alerts are generated, the service can send a HTTP request to some endpoint. In our case, we send an HTTP request to our web application developed API.

So, for the "Sensor Box X not detected" alarm, a deadman alert was created, detecting when no frequency value is received during 2 minutes in the SensorBoxesData bucket in the "FrequencyMeasures" *_measurement*. An alert was created for each sensor box available.

For the "Vibration Sensor of Sensor Box X not detected" alarm, a threshold alert was created. This alert will detect and generate an alarm every time the frequency value is equal to -1 (in SensorBoxesData bucket in the "FrequencyMeasures" *_measurement*). As explained before, in subsection 4.4.1, in the data ingestion script running in the sensor box, when the vibration sensor (ADXL345) is not detected, it sends the frequency with a -1 value so that this alarm could work. Also, an alert needed to be created for each sensor box available.

For the "Energy Meter X not connected" alarm, another deadman alert is created detecting when no power data is received during 2 minutes in each of the meters *_measurements* in the Smart Plugs bucket. Also, an alert needed to be created for each energy meter.

Finally, the "Beacon X with low battery" is the only one not implemented using *InfluxDB Alerts* service because using it would require creating an alert for every beacon of the 16 available. So, we did it in the data ingestion script running in the sensor box (subsection 4.4.1). So, every time beacon information is received in the sensor box, when its battery value is below the minimum value, a request to the web application API is made, alerting for the low battery level of the corresponding beacon.

4.4.7.3 Backend

As the backend of the old version was all developed on AWS, and as already mentioned, there was the need to leave pay-per-use platforms, all of it was rebuilt.

To build the web application backend, as was already mentioned in the technologies section (subsection 4.3.3), the *Flask Python* library was used to build the API, and the *SQLite* was used to build our database.

Relative to the *Flask* API, we created all the endpoints needed by the web application features. The endpoints created are shown with a description of their purpose in the appendix, Appendix B

As the remote server was available, we decided to use it to host the API. So, the *Flask* file script is always running as a *systemd* service in the remote server. However, we needed the endpoints to be accessible from outside the server network, so it was necessary to do port forwarding from one port and deliver the API on that port.

For the API to access the database, the *SQLite* database file is also on the remote server's file system, so the access is direct.

The database table structure and corresponding attributes are also in the appendix, Appendix A.

4.4.7.4 Frontend

Opposite to what happened with the development of the backend, the frontend of the web application, containing the GUI, built using *React*, developed in the previous work [22], was used and expanded to support the new features. However, as already mentioned, it was hosted in *Netlify* pay-per-use platform, so we also decided to use our remote server to host the frontend.

To do it, a *Nginx* web server (technology presented in subsubsection 4.3.3.5) was used to deliver the web application outside the remote server. The *Nginx* containing the web application *build* is always running, also as a *systemd* service in the remote server. For the same reason as with the backend, a port forwarding was also needed, so the *Nginx* could deliver the *React* application on that port and be accessible from outside.

4.5 Summary

At this point, all the system developed was described. We started by enumerating the system's requirements, followed by a description of the architecture, and finished with a description of the work's implementation.

The next step is the description of the validation tests and the interpretation of the results obtained.

CHAPTER 5

Tests and Validation

This chapter aims to present the validation of the system proposed in this work. It starts with validating the localization solution (section 5.2). Then the system's capacity to identify the restoration processes is validated (section 5.3). Afterwards, the web application module is validated (section 5.4). Finally, the sensor box intrusiveness validation is described in (section 5.5).

The tests used, the results, and their interpretation is presented in every module validation.

5.1 Validation Introduction

In this chapter, we will verify if the system can fulfil all the objectives for which it was built, thus validating it.

Thus, we decided to perform its validation in parts. We will start by presenting the validation of the sensor box localization solution implemented, section 5.2. Afterwards, we will present the validation of the system's ability to identify the proposed restoration processes, section 5.3. Next, the validation of the new version of the web application will be described, section 5.4. Finally, we will present the workers' opinion regarding the sensor box, validating its portability and intrusiveness in a working day, section 5.5.

Before we begin describing the validation results and their interpretation, it is essential to note that in the course of producing this thesis, a paper [12] was submitted to the ICITA 2022 conference [1]. This paper was written in an intermediate phase of the development of this work, and it encompasses the first tests and setups used to identify the restoration processes. It also had an introductory overview of the implementations performed in the meantime, in the final phases of this work. The submitted paper was accepted under a peer-review evaluation. The conference was held between the 20th and the 22nd of October 2022, and a presentation of the paper was done.

[1] https://www.icita.world/ Accessed on: 09/03/2023

Also, our work was presented to be validated by two major stakeholders in the world of classic cars in Portugal, the ACP Clássicos [2] and Museu do Caramulo [3] who showed an interest in the project from the beginning, and the feedback was very positive.

Both the acceptance of the paper at the conference and the positive feedback from these entities reinforce the importance of this work's development and the relevance of the proposed objectives.

5.2 Location Identification Validation

The best possible case to validate the Processes Identification Algorithm localization step would be to put the box on a car that arrived at the workshop and carry out the restoration processes with the box attached. Then we would have the real path of a car, in the workshop space, from the beginning to the end of the restoration.

However, due to the time available to do this work, the fact that the cars are in the workshop for months, the regular queue in a workshop, and the need sometimes to wait for parts from other companies, this kind of test can not be performed.

So we needed to do the tests by simulating paths. The purpose of the paths is to test the differentiation of the different proximity zones with their peripheries, as well as the accuracy of the fingerprinting.

We decided, on each path, to always enter one of the proximity zones and exit to the periphery, as this is normal to happen during the work in the workshop. This differentiation is necessary for the operation of the Processes Identification Algorithm to be correct, without confusion in the localization of cars.

The chosen paths, mainly the parts concerning the points where the fingerprinting solution is used, may not be done the same by the cars in a real situation. However, as already mentioned, the intention is to understand the accuracy of the fingerprinting, and this is independent of the point where the car was previously.

So, to choose the paths, we take the fingerprinting measurement points map and the proximity zones and numerate the possible points where a car could be, considering the positioning of cars observed during the visits to the workshop. The positions numbers are shown in Figure 5.1

Four paths were defined and represented in Figure 5.2. The first one consists in the sequence 2, 6, 12, 20, 16, which simulates a car that comes from the different polyvalent zones (pos. 2, 12), the bodywork simple repairs zone (pos. 6), enters the left paint booth, or paint booth 1 (pos. 20), and leaves it for its adjacent position, covered by the fingerprinting point, pos. 16.

The second one consists in the sequence 4, 9, 10, 19, 15, which simulates a car that also passes the different polyvalent zones (pos. 4, 10), the bodywork simple repairs zone (pos.

[2] https://www.acp.pt/classicos Accessed on: 09/03/2023
[3] https://museudocaramulo.pt/ Accessed on: 09/03/2023

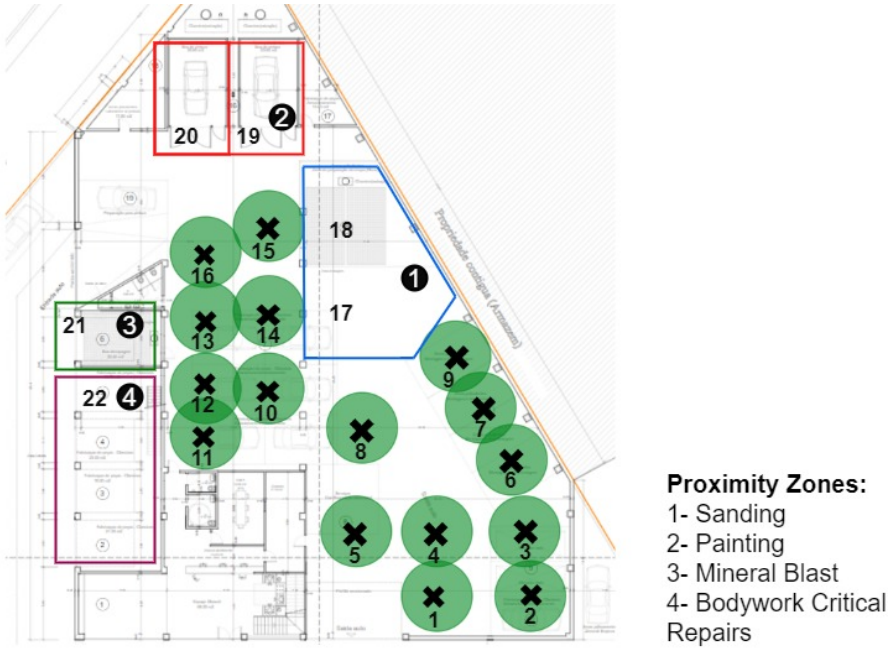

Figure 5.1: Workshop Floor Plan with the different position points. With proximity zones and fingerprinting points identification.

9), enters the right paint booth or paint booth 2 (pos. 19), and leaves it for its adjacent position, covered by the fingerprinting point of pos. 15.

The third one consists in the sequence 8, 17, 14, 21, 22, which simulates a car that passes the bodywork simple repairs zone (pos. 8), enters the bottom part of the sanding zone (pos. 17), leaves it for one of its adjacent fingerprinting position (pos. 17), then goes to the mineral blasting booth (pos. 21) and leaves it to its adjacent position in the bodywork critical repairs zone (pos. 22).

The fourth and final path consists in the sequence 18, 15, 7, 11, simulates a car that was in the upper part of the sanding zone (pos. 16), leaves it to one of its adjacent fingerprinting positions (pos. 15), goes to the bodywork simple repairs zone (pos. 7) and finishes in one of the polyvalent zones positions (pos. 11).

As mentioned in this paper, depending on the restoration activity, the cars stay at least half an hour in the same place. They can even stay days or weeks in the same place, so the simulation should be done by keeping the box static for some time in the same position. Also, as described before in the Processes Identification Algorithm (subsection 4.4.5), the algorithm developed uses that last minute most predicted location to reduce the error, so for that reason, we maintained the box static for at least four minutes in every path point, so we could get enough outputs from the algorithm, to build a trustworthy result set.

In the table Table 5.1, each path point's percentage of correct location inferences, generated by the Processes Identification Algorithm, will be presented. Also, as already described in the fingerprinting implementation section (subsubsection 4.4.2.3), by the accuracy table of the ML algorithms tested (Table 4.3) for the fingerprinting model, the

Figure 5.2: The four location validation paths.

accuracy was not 100%, so, as errors are expected in the points belonging to the fingerprinting solution, in those cases, the percentage of wrong predictions corresponding to adjacent points is presented. Finally, the last column will show the number of wrong predictions that fall on the proximity zones.

5.2.1 Location Validation Results Interpretation

Analyzing the table (Table 5.1), we can make several conclusions about the quality and validity of the localization solution developed.

The first characteristic to note has to do with the results present in the third column. This column consists of the percentage of correct predictions in the total predictions set. So, in this case, the higher the percentage, the better the results. It can be noted that concerning the points where the fingerprinting solution is requested, the overall result is negative because the hit percentage is 24.2%, which stays below 50%. On the contrary, the results are excellent in the points where the proximity solution is required, with 100% of correct answers in all points.

This result means that our fingerprinting solution should not be used for precise

Table 5.1: Results of location validation tests. Bold locations are the proximity solution locations.

Path	Location Point	$\frac{\text{Correct Hits}}{\text{Total}}$	$\frac{\text{Adjacent Hits}}{\text{Errors}}$	$\frac{\text{Proximity Zones Hits}}{\text{Errors}}$
Path 1	P2	$\frac{4}{15}$ = 26,7%	$\frac{7}{11}$ = 63.6%	0%
	P6	$\frac{3}{15}$ = 20%	$\frac{4}{12}$ = 33.3%	0%
	P12	$\frac{2}{11}$ = 18.2%	$\frac{8}{9}$ = 88.9%	0%
	Paint 1 (P20)	$\frac{11}{11}$ **= 100%**	——	——
	P16	$\frac{10}{11}$ = 90.9%	$\frac{1}{1}$ = 100%	0%
Path 2	P4	$\frac{3}{16}$ = 18.8%	$\frac{12}{13}$ = 92.3%	0%
	P9	$\frac{0}{16}$ = 0%	$\frac{16}{16}$ = 100%	0%
	P10	$\frac{1}{10}$ = 10%	$\frac{9}{9}$ = 100%	0%
	Paint 2 (P19)	$\frac{10}{10}$ **= 100%**	——	——
	P15	$\frac{0}{7}$ = 0%	$\frac{5}{7}$ = 71.4%	$\frac{5}{7}$ = 71.4%
Path 3	P8	$\frac{0}{9}$ = 0%	$\frac{9}{9}$ = 100%	0%
	Sanding (P17)	$\frac{9}{9}$ **= 100%**	——	——
	P14	$\frac{1}{10}$ = 10%	$\frac{9}{9}$ = 100%	0%
	Min.Blast (P21)	$\frac{10}{10}$ **= 100%**	——	——
	P22	$\frac{11}{11}$ **= 100%**	——	——
Path 4	**Sanding (P18)**	$\frac{10}{10}$ **= 100%**	——	——
	P15	$\frac{0}{13}$ = 0%	$\frac{11}{13}$ = 84.6%	$\frac{3}{13}$ = 23.1%
	P7	$\frac{0}{7}$ = 0%	$\frac{7}{7}$ = 100%	0%
	P11	$\frac{13}{13}$ = 100%	——	——
Total	——	**Fing.:** $\frac{37}{153}$ **= 24.2%** **Prox.: 100%**	$\frac{98}{115}$ **= 85.2%**	$\frac{8}{115}$ **= 7%**

location identification, as expected. On the contrary, when the proximity solution is requested, it works with total accuracy, so we can state that the box's location within the proximity zones is correctly identified.

The fourth column shows the percentages of location prediction hits relative to points adjacent to the box's correct location. This information is vital to understand if, whenever the system incorrectly predicts a position, it is close to the position that would be correct or if it is far away. In the former case, it may mean just low accuracy of the developed solution. In the latter case, the implemented solution may not be the best for our objective since the system predicts points unrelated to the correct location.

The results shown in that column are positive. Around 85% of the time, the system predictions fell into positions adjacent to the one that would be correct. Meaning that, although most of the time, the system does not identify the exact position where the box is (in the fingerprinting points), the tendency is that the predicted point is next to the correct one.

Due to the non-existence of exact accuracy in the fingerprinting points, there is a need to check when the box is at the points on the periphery of the proximity zones and whether or not the prediction tends to fall in those proximity zones.

Naturally, this is an important topic because, in a real situation with multiple boxes, it will mean that only cars within the zone will be associated with that zone, which is correct, or that cars on the periphery may also be associated with that zone, not desirable. The last case, when applied to the booths, for example, means the system would understand that two cars can be inside, assuming both are there, which is not desirable because it is not possible in reality.

In every path, every time the box enters a proximity zone, it leaves it after to a point within a zone in the periphery. The points in the periphery were P16 in path 1, P9 and P15 in path 2, P8 and P14 in path 3, and P15 in path 4.

So in the fifth column of the results table, we calculated the percentage of wrong location prediction hits that fell wrongly inside proximity zones. Again, as can be seen, the results are excellent. Moreover, the only location point that raised errors was the P15. That happens because P15 is a point between the paint booth 2 zone, the sanding zone, and the start of the fingerprinting zone, where there are different beacons close to that position, being hard to distinguish them.

Even so, calculating the same percentage, only for P15, as it appears in two paths, we realize that it is $\frac{8}{20} = 40\%$. A value below 50%. Also, analyzing those wrong predictions, they always fell inside the sanding zone. As explained before, this is a fault that we can support, as vibration detection is used afterwards to confirm the sanding activity.

5.2.2 Conclusions on Location Validation

With the validation tests and results described, we can conclude that, for the proximity zones, the proximity solution works correctly, with total precision, and the tendency is for

there to be no failures. Regarding the fingerprinting solution, the lack of exact precision in identifying the box's position is remarkable. However, the tendency is for the system prediction to be an adjacent point, close to the one that would be correct. Also, even with this lack of precision, a point close to a proximity zone, only 7% of the time, is associated with that type of zone. When it happens, it tends to be associated with the sanding zone, which we can tolerate.

Overall, the results are positive. Because, knowing that the priority of the localization module is to identify the car in the proximity zones, as it is there that the different restoration processes are identified, the system successfully meets this requirement. Regarding open spaces (fingerprinting solution), the system, despite not having total accuracy, tends to locate the cars near their real location without interfering with the proximity zones, which is also positive.

5.3 Activities Detection Validation

The most important part of validating the developed system involves identifying the different restoration processes present in this system's functional requirements: the sanding process, mineral blasting, painting, and curing.

For this purpose, the Output File Generation Algorithm that encompasses the Processes Identification Algorithm should be tested and validated.

So, to validate this part of the system, a sensor box was left in the workshop during some regular working days. On certain days the box was left in the paint booths, others in the mineral blasting booth, and lastly, it was also used during car sanding. It was explained to the workers how they should interact with the box, and they were asked to record the start and end times of the restoration activities they performed there. Sometimes these records were also made by us during visits to the workshop.

The second part compares those records with the records of the processes identified by our system, available in the daily output files generated by the Output File Generation Algorithm.

So in the following sections, the start and end timestamps of the identified restoration processes will be compared with the reality timestamps to verify their validity. This comparison will be shown in tables and separated by mineral blasting, painting booths, and sanding.

All the reality timestamps, except for sanding, are not presented in the second scale because the workers make these records some minutes before or after the execution of the work, so there is no such precision.

In the sanding tables, the reality timestamps were based on the load of the sander used, captured in the smart plug where it was connected. So, in that case, the seconds are also shown.

5.3.1 Painting Booth Activities Detection Validation

In the painting booths, we intend to identify the beginning and end times of the painting activities and the curing. So in Table 5.2, we present four records for each booth, containing the start and end time recorded by the workers and the times inferred by our system for the painting activities and curing.

Table 5.2: Reality vs. Results of the Output file in Paint Booths.

Paint Booth (Record num.)	Reality: Start/End Painting Activ.	**System: Start/End Painting Activ.**	Reality: Start/End Curing Activ.	**System: Start/End Curing Activ.**
Booth 1 (R1) 27/1/23	Start: 11:30 End: 11:50	Start: 11:00:00 End: 12:04:00	Start: 12:05 End: 12:35	Start: 12:05:00 End: 12:35:00
Booth 1 (R2) 1/2/23	Start: 10:30 End: 12:00	Start: 10:10:23 End: 12:06:00	Start: 12:06 End: 12:36	Start: 12:08:05 End: 12:36:01
Booth 1 (R3) 7/3/23	Start: 12:00 End: 12:30	Start: 11:34:05 End: 12:34:12	Start: 12:31 End: 13:00	Start: 12:34:14 End: 12:58:08
Booth 1 (R4) 7e8/3/23	Start: 15:30 End: 16:50 Start: 9:00 End: 9:40	Start: 14:19:44 End: 16:50:04 Start: 08:59:47 End: 09:36:46	Start: — End: — Start: 09:40 End: 10:00	Start: — End: — Start: 09:38:08 End: 09:55:46
Booth 2 (R5) 19/1/23	Start: 16:50 End: 19:30	Start: 15:07:41 End: 19:43:22	Start: 19:47 End: 20:15	Start: 19:44:35 End: 20:13:25
Booth 2 (R6) 6/3/23	Start: 14:20 End: 15:00	Start: 13:54:48 End: 15:05:57	Start: 15:07 End: 15:14	Start: 15:05:58 End: 15:12:44
Booth 2 (R7) 7/3/23	Start: 10:40 End: 12:30	Start: 10:05:03 End: 12:32:32	Start: 12:30 End: 13:00	Start: 12:33:13 End: 13:02:15
Booth 2 (R8) 7/3/23	Start: 16:20 End: 17:00	Start: 15:22:21 End: 16:57:49	Start: 17:00 End: 17:10	Start: 16:58:29 End: 17:11:46

With the study on implementing the load identification in the painting booths activities (subsubsection 4.4.3.2), we understood the problem related to the booth staying in a painting state when the painting is not yet being done and the incapacity to identify, based on energy consumption, this difference. Thus, this inaccuracy was expected in the validation tests.

Therefore, we will interpret the registration table shown in Table 5.2, comparing the registrations made by the workers with the results of the system developed.

Starting by focusing on the painting process, it can be noted that the end times recorded by the workers are very similar to the results of the system. Except for R1 and R5, the difference is always less than 6 minutes. This low 6-minute difference can be justified by the delay or uncertainty of the worker when writing the record times so that it can be ignored. On the contrary, regarding cases R1 and R5, the difference has some proportion. Although it can still be justified in the same way as in the previous case, most likely, in this case, the worker has stopped painting, but the cabin remained in the same state, being in this way an error of precision of our system.

Regarding the start timestamps of the painting activity, the difference is, except for one case (R4, part 2), always more than 20 minutes, reaching 1h40m in the R5 record. In these significant difference cases, the system resulting start timestamp is always before the start of the painting activity. This problem is due to the problem of setting the booth in the state of painting from before the painting activity started, and as expected, the system can not identify that start. Therefore, this precision problem must be addressed (some solutions are discussed in the future work section (section 6.2)).

Concerning the curing activity results, the differences are always below 5 minutes. These differences are justified by the delay or uncertainty of the worker when writing the record. Therefore this can be ignored, and we can assume that the system works correctly in identifying the beginning and end timestamp of the curing activity.

Thus, we assume the system needs more accuracy in identifying the beginning of the painting activities. However, the results show that a painting activity, despite its inferred start timestamp being, most of the time, earlier than the reality, the activity is always identified, allowing us to guarantee that whenever a painting activity is performed in a car, it will always be registered in the system.

5.3.2 Mineral Blasting Activity Detection Validation

Table 5.3: Reality vs. Results of the Output file in Mineral Blast Booth.

Date (Record num.)	Reality: Start/End Min. Blast	**System: Start/End Min. Blast:**
7/2/23 (R1)	Start: 15:20 End: 17:50	Start: 15:35:44 End: 17:48:25
8/2/23 (R2)	Start: 15:28 End: 17:10	Start: 15:35:45 End: 17:09:05
22/2/23 (R3)	Start: 09:00 End: 12:30	Start: 09:10:01 End: 12:31:21
24/2/23 (R4)	Start: 09:00 End: 10:00	Start: 09:06:02 End: 10:04:07

As mentioned in the load identification implementation of the mineral blast (subsubsection 4.4.3.1), identifying this restoration activity's start and end times is straightforward.

This simplicity can be proven by the records present in table Table 5.3. Interpreting the results, we verify that relevant differences between the worker's register and the system exist only at the activity start time. There is a noticeable difference ranging from 5 to 15 minutes from the time of the employee's registration and the activity start time detected by the system. The justification for this deviation has to do with the fact that the worker's time registration is done before the preparation of the car or its parts for the mineral blasting activity. However, the system only detects when the mineral blasting machine is switched on, which is the crucial part.

So, related to mineral blasting detection, the system can detect its start and end time with precision, meeting the objectives of the work.

5.3.3 Sanding Activity Detection Validation

In the sanding activity results table, shown in Table 5.4, besides the comparison of the start and end timestamps of the activity records, as was done in the other processes previously presented, the identification capability of the ILM solution, used during the process identification, in identify the sander activations is shown, for each record, in the last column of the table.

In every record, as a sanding activity was carried out, only the sander tool was used. However, the worker can activate it at low or high-intensity levels. Therefore, it is essential to understand the system's ability to identify this tool regardless of the power level used.

The results will be interpreted next.

Table 5.4: Reality vs. Results of the Output file on Sanding.

Date (Record num.)	Reality: Start/End	**System (Output file): Start/End**	Accuracy on sander tool identification (ILM): $\frac{\text{Num. Sander Activations Detected}}{\text{Num. Sander Activations}}$
21/2/23 (R1)	Start: 09:22:23 End: 10:32:24	Start: 09:22:28 End: 10:32:06	$\frac{5}{5} = 100\%$
6/3/23 (R2)	Start: 10:55:04 End: 11:46:35	Start: 11:19:33 End: 11:42:25	$\frac{2}{4} = 50\%$
7/3/23 (R3)	Start: 13:45:39 End: 14:02:21	Start: 13:46:17 End: 14:02:46	$\frac{2}{2} = 100\%$
7/3/23 (R4)	Start: 14:05:22 End: 14:49:24	Start: 14:14:26 End: 14:50:11	$\frac{2}{5} = 40\%$
7/3/23 (R5)	Start: 09:58:48 End: 10:12:19	Start: 09:58:58 End: 10:12:02	$\frac{4}{6} = 67\%$

In this case, as described early, the reality records timestamps are based on the energy consumption of the sander that is being used in each case, so the difference between the reality and system timestamps should be minimal. However, as can be noted, differences exist in some cases (Start of R2, Start of R4). This error happens due to the ILM solution

not identifying the sander in some of its activations during the sanding process carried out in the car. This assertion is corroborated by the data in the last column of the results table, where the hit percentage of ILM regarding these activations is checked.

As noted, in R2 and R4, 2 of the 4 and 2 of the 5 sander activations, respectively, the ILM solution did not identify the sander but another tool. Moreover, in both cases, the first sander activation was identified as another tool, resulting in later detection by the system.

Also, in the R5 record, it can be verified, in the last column of the table, that the ILM solution accuracy is not total. However, in this case, it does not interfere with the task's identified start and end timestamp. It does not interfere because only intermediate activations of the sanding tool were misidentified.

By exploring more in-depth the ILM training set and comparing it with the loads generated by the sander activations related to the R2, R4, and R5 records, we understood that when the sander is used at low-intensity levels (the workers change the level according to the necessary finesse), its load is very similar to the angle grinder load. So, the ILM solution infers, in this case, the angle grinder.

However, checking the sanding activities that we controlled, where the 5 records in the table are included, the sander's medium/high-intensity levels are constantly used when this activity is done. As shown in the results table, this results in a correct identification of the sanding process in a car, although with some imprecision.

Nevertheless, even if it is unlikely, we face a validity threat if a certain sanding job implies a low level of intensity in the sander during the whole process, meaning that the system would not identify that sanding work.

5.3.4 Conclusions on Activities Detection Validation

Summarising the validation results regarding the paint booths, we notice that the system identified all painting and curing activities. Then, from the point of view of detecting the start and end timestamp of the processes, the system accurately identifies the end timestamp of the painting activities and the start and end timestamp of the curing activity. Regarding the start of the painting activities, there is some imprecision, as the system usually returns the start time earlier than the real-time.

Regarding the mineral blasting carried out in the mineral blasting booth, the process was always detected, and the system correctly returned both the start and end timestamps of the activity execution.

The sanding process was always detected too, but in this case, there were errors in some cases in the start and end timestamps of the activity execution. Depending on the use or not of high-intensity levels in the sanders.

We can therefore conclude that, in general, the objectives have been achieved as the system identifies all the different types of restoration processes for which it has been built.

Moreover, it also accurately identifies, in almost all situations, the start and end time of the process's execution.

5.4 Web Application Validation

To validate the web application, we should test all components of it, i.e., the backend functions and front end. However, due to the large extension of the application, we decided to cluster everything, creating test scenarios that will guide the users, who will test the application, through the different features of the application, thus validating the whole web application system plus the usability.

Those test scenarios are needed because our system is a first production version, so it has yet to be used during every working day in the workshop. Because of that, we have no experienced users with the application. So, to perform the validation of our web application, we needed to create some test scenarios/guidelines, which the people chosen to perform the tests would follow so every application requirement could be tested and validated.

Every test scenario will meet the most frequent uses of the application during a typical working day in the workshop and will test all the features present in the functional and non-functional requirements of the web application (defined in subsubsection 4.2.1.2).

As the application has two stakeholders, the project technicians and the workshop supervisors, to carry out the tests, two groups of people were collected, containing people with technical knowledge as close as possible to the stakeholders.

For the workshop supervisors, the two existing workshop supervisors in the Raimundo Branco workshop participated. The supervisors mentioned will be the ones who will use the application when it is fully functional in the workshop.

For the project technician group, the user's necessary characteristics are to be used to web applications like ours and to have an easy understanding and use of computer systems. Therefore, 8 students from the computer science course from *FCT-NOVA* were chosen to test the application as project technicians.

After following the test scenarios and performing the respective tasks, each user will be asked about the tests performed through questionnaires.

Two types of questionnaires were used. After every task accomplishment, the user will answer a questionnaire with 6 questions about the execution of that task. This questions will evaluate the Task Load Index (TLX) [4]. This index will evaluate each task, subjectively for each user, by its mental, physical, temporal, performance, effort, and frustration demand. Every question will have a 21-point slider, from 1 representing the answer "Very Low" to 21 representing "Very High". At the end of each task questionnaire, an open question was presented so the user could give suggestions related to the task.

[4] https://humansystems.arc.nasa.gov/groups/TLX/index.php Accessed on: 027/02/2023

This type of questionnaire by test scenario will help us validate every specific part and feature of the application system and its capability to accomplish its requirements.

After the execution of all test scenarios, the user will answer a final questionnaire containing usability and user-friendliness questions where the System Usability Scale (SUS) is used.

The SUS is a tool for measuring a system's usability, consisting of a set of 10 questions, which are answered by one of 5 options. From "Strongly Disagree" to "Strongly Agree" [5]. The 10 SUS question are the following:

- I think that I would like to use this system frequently.
- I found the system unnecessarily complex.
- I thought the system was easy to use.
- I think that I would need the support of a technical person to be able to use this system.
- I found the various functions in this system were well integrated.
- I thought there was too much inconsistency in this system.
- I would imagine that most people would learn to use this system very quickly.
- I found the system very cumbersome to use.
- I felt very confident using the system.
- I needed to learn a lot of things before I could get going with this system.

After the SUS questions, the workshop supervisors group users are presented with 6 more questions related to the web application features relevance. Only the supervisors answer these 6 questions, as they are the only ones who know the project context and can understand if the web application is relevant to the project.

The first 5 of the 6 have the same type of SUS possible answers, where the users can give their opinion on the importance of the application's main features. Questions 2 and 4 refer to the main new features implemented in this new version so that we can evaluate the users' opinions about these new updates. Finally, the sixth question is an open question where the user can suggest improvements for the web application. The 6 questions are the following:

1. I find it helpful to have a sensors monitor hub where I can see all sensors' states and positions.

[5] https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html Accessed on: 28/02/2023

2. I find it important to have a dynamic map of the sensor's exact positions in the workshop (beacons and smart plugs).
3. I consider it important to have access to sensors alarms.
4. I find it relevant to have access to a live map of the last activities identified in the workshop.
5. I think the continuous monitoring of the evolution of the vehicle restoration process through the system is important.
6. Do you have any suggestions for new features to add?

Before presenting the results of the questionnaires, the 8 test scenarios created are still to be presented. As there are two stakeholders, there will be two different scenarios for each part of the application, one for each stakeholder. Each scenario will require different parts of the application so that there is an easy understanding and a more straightforward execution by the user that it will test. The first scenario involves the control of the sensor boxes, the second with the control of the beacons, and the third with the smart energy meters. Finally, the fourth concerns the live map and the latest activities detected in the workshop.

5.4.1 Workshop Supervisors Scenarios

João, the workshop supervisor, is responsible for the sensor control in the workshop this week. To do this, he will use the system web application.

First Scenario: Because it was the end of the working day, João, put the box called "SensorBox_07" to charge. The next day, at the very beginning of the working day, as the box will be placed in a new car, the "Jaguar E-Type 1960 – GXA 234", João, checks if this car is already on the project's car list, if not, he adds it to the system and associates it to the box. Next, he connects the box to the battery, checks if it is well connected as well as its vibration sensor, and if everything is ok, he puts the box in the corresponding car. Finally, he checks if any alarms were generated, on the current day, on the box.

Second Scenario: João checks if there is any beacon on the beacons map whose battery is already below the minimum level. If the battery of a beacon is low, it must be replaced. So, John checks where the beacon is in the workshop space, goes to the location and changes the battery of the beacon. When he returns to the computer, he checks whether the new beacon already has the updated full battery level. Finally, he checks whether there are any new beacon alarms on the current day.

Third Scenario: João checks the system to see if there is any energy meter which is not connected. If he finds one not connected, he looks up the position of the meter on the map of these, moves to its position and connects the meter to the corresponding socket. He goes back to his computer and checks if it is already identified as connected. Finally, he checks for new alarms concerning other meters on the current day.

Fourth Scenario: João is going to have a visit from partners at the workshop, so he wants to show the operation of the restoration processes identification system, so he accesses the live map of the last activities detected by the system. And checks which was the last activity detected by the ”SensorBox_07”.

5.4.2 Project Technicians Scenarios

Afonso is a computer engineer who is responsible for the restoration processes identification system. So when he goes to do updates with the sensors in the workshop, he needs to access the web application.

First Scenario: Afonso, is going to put a box, already registered in the system, called ”SensorBox_03” back into operation in the workshop. To do that he accesses the application, and associates the car ”Jaguar E-Type 1960 - X43 AHJ” to the box, checking first if this car is already in the project's car list, if not, he adds it to the system and associates it to the box. Then he checks if the box and its vibration sensor are well connected, and finally, he puts the box to record data. At the end of the day, Afonso checks if the box has any alarms that day.

To confirm that everything went normally with the sensor box ”SensorBox_07”, he also goes to the logs of that box and downloads the last log file available.

Second Scenario: Afonso, accesses the application and decides to do a check-up on the system's list of alarms. Checks the most recent alarm of a beacon that has a battery level below the minimum. Then he goes to the beacon list to find the corresponding beacon and decouples it. Afonso then goes looking for a beacon with a high battery level and sets it on position 1, where the beacon with no battery was before. Finally, with Afonso present in the workshop, he checks on the beacon map, which is the new position, and he will place the new beacon in this position. Before moving on to another task, checks if the beacon ”917cd714e1103994” has any alarms of the current day.

Third Scenario: Afonso has a new smart meter, called ”plug6”, already registered in the system but disconnected and not associated with any position. As Afonso wants to put the socket in operation, he sets the position of the new socket to ”P4”, then connects the socket in that place, checking the map to verify where the position is. Back at the computer, he checks that it is already identified as connected. Before moving on to another task, he checks if the socket ”plug3” has alarms generated in the current day.

Fourth Scenario: Afonso is going to have a presentation of the system to partners, so he wants to show how the restoration processes identification system works, so he accesses the live map of the last activities detected by the system and checks which was the last activity detected by the ”SensorBox_07”. Besides, he downloads the most recent output inferences file of that box.

5.4.3 Web Application Validation Results

Regarding the SUS questionnaires, as each of the 10 questions has 5 possible answers, these are evaluated from 0 to 10, with intervals of 2.5 [15]. The order of the best and worst scores will depend on how the question is asked. Adding up all the questions, the maximum score of the questionnaire is 100. The SUS system has a scale table, where the 100 score means an A+ grade (Table 5.5). In [15], a score higher than 80 is considered a positive result meaning a positive user experience.

Table 5.5: SUS Scale.

Grade	SUS	Percentile Range
A+	84.1 - 100	96-100
A	80.8 - 84.0	90-95
A-	78.9 - 80.7	95-89
B+	77.2- 78.8	80-84
B	74.1- 77.1	70-79
B-	72.6 - 74.0	65-69
C+	71.1 - 72.5	60-64
C	65.0 - 71.0	41-59
C-	62.7 - 64.9	35-40
D	51.7 - 62.6	15-34
F	0 - 51.6	0-14

So in the Figure 5.3, we will show the calculated average and standard deviation of the results for each stakeholder separately and then the same measures joining both.

Relative to the TLX questionnaires, although the four scenarios of the two stakeholders address the same parts of the web application, they are different, so we decided only to calculate here the mean and standard deviation values separately. Thus, in the Figure 5.3 table, the mean and standard deviation values of the users' answers related to each topic tested by the TLX questionnaire are presented.

To better understand the ease of use of the application regarding each part, we also decided to create the graph present in Figure 5.4, where we can compare each scenario, using the corresponding average results of each topic verified in the TLX questionnaires. As mentioned before, scenario 1 has to do with the control of the sensor box, scenario 2 with the control of the beacons, and scenario 3 with the smart energy meters. Finally, scenario 4 has to do with the live map and the latest activities detected in the workshop.

Relative to the 6 questions about the relevance of the web application features, the SUS approach was used by scoring each answer from 0 to 10. The mean and standard deviation of the answers results are also shown in Figure 5.3.

We then start by analyzing the results of the table shown in Figure 5.3.

Relative to SUS, the results are positive. The total average value is 80.8 with a low standard deviation of 7.4, above the minimum accepted proposed in [15] (80 score), which means an A score for the web application usability.

Quest.	Metric	Workshop Supervisor		Project Technician		Total	
		μ	σ	μ	σ	μ	σ
SUS	SUS-Score	100	0	60,8	5,6	80,8	7,4
Nasa-TLX	Mental Demand	1,5	0,8	4,9	3,9		
	Physical Demand	1,8	1	---	---		
	Temporal Demand	1,5	0,5	3,8	2,7		
	Performance	20,5	1,1	19	2,6		
	Effort	1,6	1,1	4,7	4,7		
	Frustration	1,3	0,5	1,6	1,3		
Web App. Relevance	(0-100) Score	100	0				

Figure 5.3: SUS and Nasa-TLX and Web Application Relevance Results. With Average and Standard Deviation.

Looking at the results of the different user groups, the difference between the two is noticeable. This difference is justified because the scenarios of the supervisors are more straightforward and shorter than those of the technicians. In addition, the first group was already familiar with the purpose of the web application.

Relative to the TLX, the same difference between the two user groups appeared, and the same justification is applied. In this case, the interpretation of the results needs to be done just by user type.

So, for the supervisors, the results are, in every metric, close to optimal values. In this test, every optimal metric value is 0, except for the performance metric, whose optimal value is 21. The most significant difference to an optimal value is 1.8 in the physical demand metric (relative to the action of charging the box, the change of the beacons battery, and the plugging of the energy meters). Moreover, the standard deviation in every metric is positive, as it is always below 1.1.

In the technicians' group results, although worse, they are also positive, as the most significant difference to an optimal value is 4.9. However, in this case, the worst metrics were mental demand and effort because the tasks performed require more steps and require some understanding of the system to be performed.

Exploring further the TLX results by interpreting the graph in Figure 5.4, we can note that scenarios 1 and 2 got the worst results in every metric. These worst results can be justified due to the more extensive complexity of these two scenarios. Because the tasks requested there require several steps, not being as direct as the tasks of the following scenarios.

The questionnaire presented to the workshop supervisors about the relevance of the

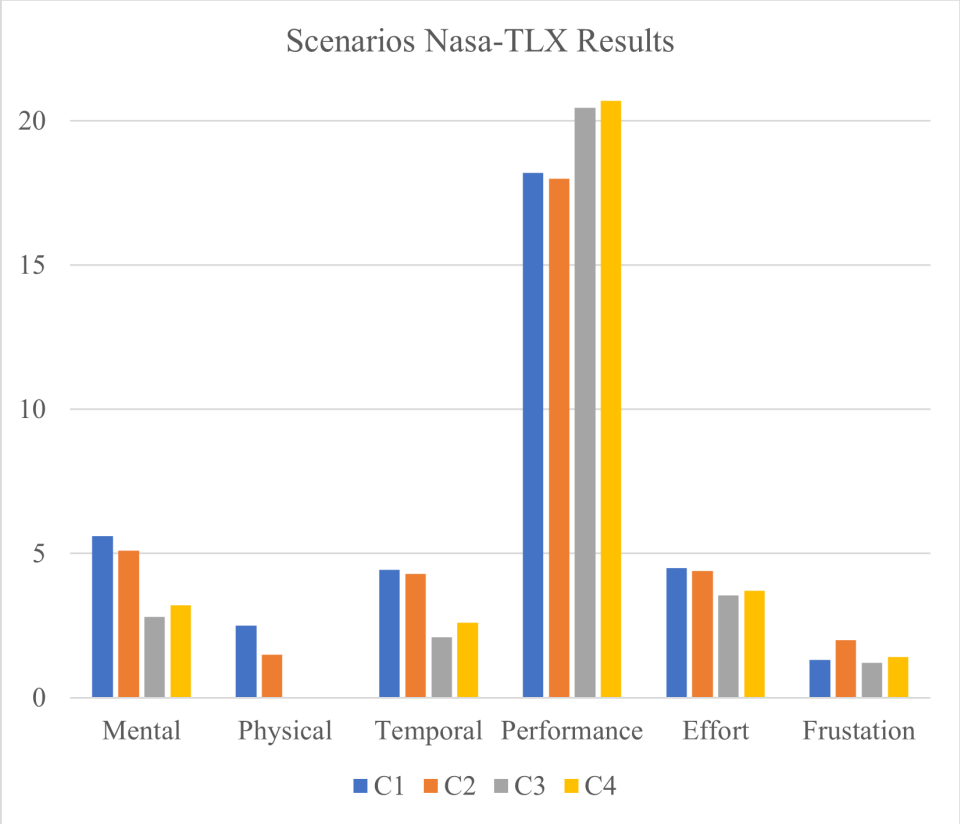

Figure 5.4: Nasa-TLX results per scenario.

different features of the web application returned optimal results. Validating the importance of the goals of the web application, as well as the new main features added (dynamic maps with dynamic positions of sensors and live detected activities map), and the importance of the system developed for the workshop.

As explained in the presentation of the questionnaires, there were questions where suggestions to improve usability or suggestions for new features were asked. Only usability suggestions were made, and they are as follows:

1. Filters and customized sorting (e.g., by date, device type) of the system alarms list;
2. Search by id in the beacons list page;
3. In the beacons list page, have the minimum battery level information displayed or a color pattern, like is done on the beacons map page;
4. Be able to change the position of a beacon without having to decouple the beacon in that position (warning the user that it will be decoupled automatically if the change is accepted);
5. Be able to access the inferences files, also, in the details of the respective sensor box.

The suggestions given resulted from the difficulties users encountered when carrying out the test scenarios.

When searching for the most recent alarm of a given beacon, in scenario 2 of technicians, many users took a while to understand the ordering logic of the alarms list page. Therefore, they took some time to find the desired beacon alarm, justifying suggestion 1. Also related to that scenario, when decoupling the low-battery beacon, some users suggested the possibility of searching by beacon id so they could quickly find the beacon (suggestion 2). Moreover, another suggestion, given by supervisors and technicians users, related to quickly finding the low battery level beacons was to have the minimum battery level information shown on the beacon list page (suggestion 3).

The last beacon context suggestion (suggestion 4) was related to the need for a position to be free so that a beacon can be set there. For example, suppose the user wants to place the beacon X in position 1. In that case, it must first uncouple the beacon associated with that position so that the position is free, and only then can the beacon X be placed in it. This procedure caused some users some discomfort.

Suggestion 5 has to do with test scenario 4, from the technicians, where some users started going to the details page of the corresponding box when searching for the inference files. However, the inference files are available on the inference page.

For the reasons explained, all suggestions are legitimate and acceptable. Whereas, we understand that suggestions 1,2,3,4 are those that caused the most difficulty for users and that they should be treated and put into practice as a priority. Also, these suggestions are related to test scenario 2, matching the lower results of this scenario in the TLX test graph (Figure 5.4).

5.4.4 Conclusions on Web Application Validation

With the tests and results obtained, we conclude that the application achieved the proposed objectives because the results are close to optimal, existing only minor imperfections.

The suggestions raised by new users met the difficulties encountered, demonstrating the ease in perceiving the intent of the application. In addition, the application proved easy to learn, simple to use, and compact, meeting the frontend requirements.

The users also did not notice any inconsistencies and affirmed an excellent system integration, thus validating the developed backend.

5.5 Sensor Intrusiveness Validation

To validate the usability of the sensor box during a working day, we built a questionnaire to be carried out through an interview with the workers familiar with the sensor box who used the box while performing some activities, for example, keeping it attached to the car during some movement in the workshop, changing it to the doors of the booth when the car entered them, or attaching it on the car panels to be sanded, when they performed these activities.

The questionnaire consists of a set of open and closed questions. The questions will be presented below. In addition, the interviews were transcribed, so the workers' responses, in summary, will be presented later in this document.

It was thought to make an audio recording during the interviews. However, in an initial conversation with some workers about the ease of use of the box, we noticed a certain tendency towards the answer "I do what I am told to do" due to the fear that their opinion, being recorded, could be heard and taken as without right, even with the explanation that the work in question is independent of the workshop and that the opinion would be anonymous and essential for the evolution of the work. So by not recording the interviews, the conversations would be more honest, and the workers would be more comfortable.

Before each interview, the worker was given a briefing to reinforce how to use the sensor box and the components that form the box. Finally, the purpose of the questionnaire was explained, and the interviewees were informed that their answers would be anonymous.

So the questionnaire that guided the interviews consisted of the following questions:

1. In what stage of restoration did you interact with or use the sensor box?
2. Do you think that using the sensor box has interfered or might interfere or become a nuisance to your work in any way?
3. If the previous question is answered with yes, the following question is asked: How did the box interfere with your work?
4. Do you think it might be inconvenient to use the box every working day?

As mentioned, we decided to talk to all workers who had already interacted with the box to answer the questionnaire. In this way, we interviewed 6 workers. 4 of them are workers who sand the cars, so they were familiarized with the need to attach the box, at least on the car side where it is being sanded. The other 2 are painters, responsible for preparing the cars for painting and painting them in the booths, familiarized with the need to attach the box to the booth door when the car enters it, and put it back on the car when the car leaves.

The highlights of each worker's answer will be presented next:

1. **Sanding Worker 1** - "There's no problem on my side. It's necessary to explain and ask all the workers to use the box. In terms of use, it would mean always attaching the box on the car panel closest to the panel being worked on, which is not a nuisance at all."
2. **Sanding Worker 2** - "I have no problem with using the box. It's simple."
3. **Sanding Worker 3** - "I work with it without a problem, if I can sand a whole side of the car without worrying about the box, there is no problem."

4. **Sanding Worker 4** - "It doesn't seem to make any difference using it, you just have to remember that you have to move the box, that's the difficult part."

5. **Painter 1** - "The process of putting the box on the door and putting it back in the car is simple. I had no problem doing it. What can be problematic is getting the guys used to it. There are forgetful people, and if they leave a car in the booth, curing, with the box in the door, they will do other things, later they will remove the car from the booth and they may not remember that the box was there."

6. **Painter 2** - "The problem is remembering. The box is small and portable, so moving it from one place to another doesn't bother you. The problem is that a person goes from one car to another and could forget about the box."

5.5.1 Sensor Intrusiveness Results Interpretation and Conclusions

By interpreting the interviews, we can safely say the results are positive.

Our main concern was using the box when sanding, which implies a greater interaction with the box. However, all the sanding workers said this interaction would not be a nuisance. Moreover, the way it is constructed, being attached with magnets, does not imply any discomfort for the workers.

The biggest problem the workers raised has to do with the possibility of forgetting the box. This problem is relevant for the system because if the vehicle changes position and the box does not accompany it, we will lose the possibility of identifying the processes.

Thus, before the sensor boxes are put into operation in the workshop, a briefing should be made to the workers drawing their attention to the importance of the boxes always accompanying the associated car. Furthermore, in the first phase, there should be supervision to help them get used to working with the box.

CHAPTER 6

Conclusions

This chapter presents the conclusions reached with the development of this work and future work proposals.

6.1 Conclusions

This work aimed to create an algorithm capable of identifying the restoration processes that a car goes through in the workshop, crossing the data from an ILM system used to detect the use of tools in the workshop space and the different activities performed in the paint and mineral blasting booths, a localization system used to locate the cars inside the workshop space and a vibration detection solution. To control the sensors installed in the workshop was decided to migrate the existing web application to a no-cost platform and expand it to support the new sensors and functionalities to make it more valuable.

We started by performing a rapid systematic review on the topics mentioned so that we could carry out well-supported implementations.

Thus, energy meters were installed in the workshop outlets and in each cabin's electrical panels to implement the ILM solution.

Beacons were placed throughout the workshop space so that with the sensor box working as a receiver of the Bluetooth signals, it was possible to implement the localization system. The localization system is based on proximity and fingerprinting techniques.

An accelerometer was used in the sensor box to detect vibrations produced by the tools in the car panels and thus make the association between the car being worked on and the tool.

We realized early on that we would only be able to identify the activity of painting, curing, mineral blasting, and sanding.

By the results shown in the validation chapter, we could verify that the implemented localization system can locate a car inside the workshop with great precision, mainly when the proximity technique is requested.

We can also verify that the system can assertively identify all the processes mentioned above. Moreover, it can also identify most activities' start and end times. Also, in these results, it was possible to verify the capacity of the developed ILM system to differentiate the electric sander, the tool necessary for the identification of the sanding activity.

Regarding the web application, it was migrated to a remote server, a database in *SQLite* and an API in *Flask* were developed, and new features were added regarding the control of the new sensors and the improvement of functionality. The application proved simple and valuable for the users who tested it.

Therefore, the whole system significantly improved the previous work with these achievements. The localization solution works correctly in the new workshop building. The system can differentiate more restoration activities, it start and end times with precision. The web application is now hosted in free platforms. And the system is prepared to be integrated with the work in the workshop, eliminating the need for the man-in-the-middle and making the workshop more digitized.

With the development of this work, we realized how IoT systems work and the whole process of ingestion and processing of temporal data generated by them. We acquired knowledge in data processing while developing some algorithms related to that subject, knowledge in machine learning and web application implementations, and both backend and frontend development.

We hope this work could be a follow-up of future projects or businesses, aiming to improve the implemented solution or to use this solution in larger environments.

6.2 Future Work

Regarding future work, several developments can be made in the continuity of this work, as well as new ideas emerged regarding the automation and monitoring of the workshop.

In terms of the design of the sensor box, we used a handmade assembly sufficient to start implementing the box in the day-to-day work of the workshop. However, creating more professional components to support the box will be a priority. In previous work [22], a 3d model was presented. However, it has yet to be produced. This sensor box upgrade is a priority for inserting several boxes in the workshop.

Regarding developments, the web application has many possibilities. Mainly because the web application is not self-sustained, supporting only simple configurations. Many changes that may need to be done to the workshop sensors setup require changes in the system settings that can not be done from the application.

The setup changes include adding or removing new sensors, such as sensor boxes, beacons, and energy meters, and creating new positions for beacons and energy meters. These are the main features that are not controlled in the application.

Regarding the addition of the sensors, besides the insertion in the database of their data, it would be necessary to automatically integrate the new sensors with *InfluxDB*, which has some complexity.

Regarding the new positions, as far as the beacons are concerned, the fingerprinting solution would have to be rebuilt again when a new position is added, implying a new data acquisition at the workshop and new training of the machine learning model, resulting in a complex process to be done from the application alone.

Also related to new sensors, a new sensor added from the web application requires the generation of alarms. As explained in the web application alarms implementation (subsubsection 4.4.7.2), most of the alarms are generated by *InfluxDB*. So it would require an automatic integration with *InfluxDB* to create an alarm for the newly added sensor.

A study was already done to understand possible solutions to solve this issue. Related to the sensors adding and removal, and its alarms, the one that seemed the best, is the use of the *InfluxDB* available API [1]. The API offers endpoints for controlling every *InfluxDB* feature, like adding new buckets, new alarms, and new data ingestion configurations. So, when the user adds some sensor in the web application, it triggers a series of requests both in the system backend and in the *InfluxDB* API so that the whole system remains consistent.

Concerning new positions and the consequent fingerprinting problem, the application may have a system of steps with instructions to be followed by the user to reconstruct the fingerprinting model, allowing the user to upload the newly recorded measurements, generate a features file and retrain the machine learning model, all within the application. However, regarding this problem, specific technology or implementation solutions have yet to be investigated.

Still, in the application context, the access security implemented was not highly developed (as explained in subsubsection 4.4.7.1). The solution implemented is based on tokens stored as cookies, which are not encrypted, passing in plain text to the browser. The same happens with the passwords of the existing user accounts, which are stored as plain text in the database. So a more secure solution, with password and token encryption, should be implemented.

As explained in the system output file generation (subsection 4.4.6), the integration with the work in [20] is not complete. However, it is necessary so that the *Charter of Turin Monitor* platform can always be up to date with the new detected processes. It would also present greater certainty in detecting the restoration processes, as the sequence of the processes done on each car could be better perceived by our system. So this integration should be done by creating endpoints and computations to support it in [20] work environment.

During the development of this work, some ideas regarding the possible detection of more types of processes also emerged to remove the significant ambiguity of restoration activities in most of the workshop sites, where different activities are carried out in the same places. An idea consists of acquiring a large amount of data, obtained during weeks of work in the workshop, with a subsequent analysis of these data trying to find patterns

[1] https://docs.influxdata.com/influxdb/v2.6/api/ Accessed on: 13/03/2023

of activity/location sequence in order to be able to predict with some certainty the most likely activity to be carried out next. This process could be helped with machine learning algorithms.

Another idea is to convert the *Charter of Turin Monitor* to a graph so that the system can perceive the activity that should be performed next in a given car, searching only for that activity, thus reducing the existing ambiguity.

The inability of the system to identify the application of the different paint products in the booths has been identified. Therefore, the solution that is the most suitable and more uncomplicated to solve this problem is the previous definition through the *Charter of Turin Monitor*, by the workshop supervisors, of which products will be applied. This way, we would know which product is associated with the first painting activity, the second, and so on. However, this previous definition may require extra supervisors' dedication, going against removing the need for the man-in-the-middle. So, this solution must be studied and discussed with the different stakeholders.

With the energy load monitoring solutions implemented, especially in the painting booths (subsubsection 4.4.3.2), it was possible to identify unnecessary consumption. We realized that sometimes the booths are On too long without anything being done inside. Thus, future work would be to conduct a study of all electrical consumption in the workshop, trying to find unnecessary consumption and identify electrical wastes and points of higher consumption that could be reduced, to reduce unnecessary costs and decrease the ecological footprint. In addition to identification, the system could control the origins of this consumption, for example, by automatically turning off a painting booth when it is not required.

Bibliography

[1] I. Abubakar et al. "Application of load monitoring in appliances' energy management – A review". In: *Renewable and Sustainable Energy Reviews* 67 (2017). cited By 93, pp. 235–245. DOI: 10.1016/j.rser.2016.09.064 (cit. on pp. 14, 23).

[2] D. Čabarkapa, I. Grujić, and P. Pavlović. "Comparative analysis of the Bluetooth Low-Energy indoor positioning systems". In: *Proceedings of the 12th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services (TELSIKS 2015)*. IEEE, 2015, pp. 76–79. DOI: 10.1109/TELSKS.2015.7357741 (cit. on pp. 17, 27, 60).

[3] J. Cunado and N. Linsangan. "A Supervised Learning Approach to Appliance Classification Based on Power Consumption Traces Analysis". In: *Proceedings of the IOP Conference Series: Materials Science and Engineering*. Vol. 517. 1. cited By 4. IOPscience, 2019. DOI: 10.1088/1757-899X/517/1/012011 (cit. on p. 21).

[4] T. De Schepper, A. Vanhulle, and S. Latré. "Dynamic BLE-based fingerprinting for location-aware smart homes". In: *Proceedings of the Symposium on Communications and Vehicular Technology (SCVT)*. IEEE, 2017, pp. 1–6. DOI: 10.1109/SCVT.2017.8240316 (cit. on p. 25).

[5] M. Devlin and B. Hayes. "Non-Intrusive Load Monitoring and Classification of Activities of Daily Living Using Residential Smart Meter Data". In: *IEEE Transactions on Consumer Electronics* 65.3 (2019). cited By 44, pp. 339–348. DOI: 10.1109/TCE.2019.2918922 (cit. on pp. 14, 15).

[6] M. Domingo. "An overview of the Internet of Things for people with disabilities". In: *Journal of Network and Computer Applications* 35.2 (2012). cited By 400, pp. 584–596. DOI: 10.1016/j.jnca.2011.10.015 (cit. on p. 16).

[7] R. Faragher and R. Harle. "Location Fingerprinting With Bluetooth Low Energy Beacons". In: *IEEE Journal on Selected Areas in Communications* 33.11 (2015). cited By 557, pp. 2418–2428. DOI: 10.1109/JSAC.2015.2430281 (cit. on p. 24).

[8] P. Franco et al. "A framework for iot based appliance recognition in smart homes". In: *IEEE Access* 9 (2021). cited By 0, pp. 133940–133960. DOI: 10.1109/ACCESS.2021.3116148 (cit. on p. 23).

[9] P. Franco et al. "IoT Based Approach for Load Monitoring and Activity Recognition in Smart Homes". In: *IEEE Access* 9 (2021). cited By 4, pp. 45325–45339. DOI: 10.1109/ACCESS.2021.3067029 (cit. on pp. 13–16, 22, 76).

[10] S. Ghorpade, M. Zennaro, and B. Chaudhari. "Survey of localization for internet of things nodes: Approaches, challenges and open issues". In: *Future Internet* 13.8 (2021). cited By 8. DOI: 10.3390/fi13080210 (cit. on p. 17).

[11] K. Gibbins. *Charter of Turin Handbook*. Tech. rep. Version 2. Accessed on 2022-02-04. Fédération Internationale des Véhicules Anciens (FIVA), Nov. 2018. URL: https://fiva.org/download/turin-charter-handbook-updated-2019-english-version/ (cit. on pp. 4, 7).

[12] R. Gomes, F. Brito e Abreu, and V. Amaral. "Combining Different Data Sources for IIoT-Based Process Monitoring". In: *Proceedings of International Conference on Information Technology and Applications*. Springer Singapore, May 2023. Chap. 10 (cit. on p. 93).

[13] F. Khelifi et al. "A Survey of Localization Systems in Internet of Things". In: *Mobile Networks and Applications* 24.3 (2019). cited By 47, pp. 761–785. DOI: 10.1007/s11036-018-1090-3 (cit. on p. 17).

[14] P. Kriz, F. Maly, and T. Kozel. "Improving Indoor Localization Using Bluetooth Low Energy Beacons". In: *Mobile Information Systems* 2016 (2016). cited By 152. DOI: 10.1155/2016/2083094 (cit. on pp. 17, 26, 60).

[15] J. Lewis and J. Sauro. "The Factor Structure of the System Usability Scale". In: vol. 5619. Springer, July 2009, pp. 94–103. ISBN: 978-3-642-02805-2. DOI: 10.1007/978-3-642-02806-9_12 (cit. on p. 108).

[16] D. Lívio. "Process-Based Monitoring In Industrial Context: The Case of Classic Cars Restoration". MA thesis. Monte da Caparica, Portugal: Nova School of Science and Technology, Jan. 2022. URL: http://hdl.handle.net/10362/141079 (cit. on pp. 2, 8).

[17] Y. Lu. "Industry 4.0: A survey on technologies, applications and open research issues". In: *Journal of Industrial Information Integration* 6 (2017). cited By 1235, pp. 1–10. DOI: 10.1016/j.jii.2017.04.005 (cit. on p. 16).

[18] M. Mahdavinejad et al. "Machine learning for internet of things data analysis: a survey". In: *Digital Communications and Networks* 4.3 (2018). cited By 379, pp. 161–175. DOI: 10.1016/j.dcan.2017.10.002 (cit. on p. 16).

[19] R.-C. Mihailescu, D. Hurtig, and C. Olsson. "End-to-end anytime solution for appliance recognition based on high-resolution current sensing with few-shot learning". In: *Internet of Things (Netherlands)* 11 (2020). cited By 4. DOI: 10.1016/j.iot.2020.100263 (cit. on pp. 21, 22).

[20] P. Moura. "Assessing The Impact Of Process Awareness In Industry 4.0". MA thesis. Monte da Caparica, Portugal: Nova School of Science and Technology, Nov. 2022. URL: http://hdl.handle.net/10362/151100 (cit. on pp. 2, 8–10, 29, 32–34, 36, 83–85, 117, 126).

[21] F. Paradiso et al. "ANN-based appliance recognition from low-frequency energy monitoring data". In: *Proceedings of the 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2013*. cited By 20. IEEE, 2013. DOI: 10.1109/WoWMoM.2013.6583496 (cit. on pp. 14, 21, 23, 76).

[22] D. Pereira. "An automated system for monitoring and control classic cars' restorations: An IoT-based approach". MA thesis. Monte da Caparica, Portugal: Nova School of Science and Technology, Jan. 2022. URL: http://hdl.handle.net/10362/138798 (cit. on pp. 2–5, 9, 11, 12, 16, 18, 33, 38–42, 46, 48, 51, 52, 85, 91, 116).

[23] A. Ridi, C. Gisler, and J. Hennebert. "A survey on intrusive load monitoring for appliance recognition". In: *Proceedings of the 22nd International Conference on Pattern Recognition*. cited By 81. IEEE, 2014, pp. 3702–3707. DOI: 10.1109/ICPR.2014.636 (cit. on pp. 15, 23).

[24] A. Ridi, C. Gisler, and J. Hennebert. "ACS-F2 - A new database of appliance consumption signatures". In: *Proceedings of the 6th International Conference on Soft Computing and Pattern Recognition (SoCPaR 2014)*. cited By 31. IEEE, 2014, pp. 145–150. DOI: 10.1109/SOCPAR.2014.7007996 (cit. on p. 21).

[25] A. Ridi, C. Gisler, and J. Hennebert. "Appliance and state recognition using Hidden Markov Models". In: *Proceedings of the 2014 International Conference on Data Science and Advanced Analytics (DSAA 2014)*. cited By 14. IEEE, 2014, pp. 270–276. DOI: 10.1109/DSAA.2014.7058084 (cit. on p. 21).

[26] A. Ridi, C. Gisler, and J. Hennebert. "User interaction event detection in the context of appliance monitoring". In: *Proceedings of the International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops 2015)*. cited By 5. IEEE, 2015, pp. 323–328. DOI: 10.1109/PERCOMW.2015.7134056 (cit. on p. 21).

[27] S. Saab and Z. Nakad. "A standalone RFID indoor positioning system using passive tags". In: *IEEE Transactions on Industrial Electronics* 58.5 (2011). cited By 307, pp. 1961–1970. DOI: 10.1109/TIE.2010.2055774 (cit. on p. 17).

[28] E. Sisinni et al. "Industrial internet of things: Challenges, opportunities, and directions". In: *IEEE Transactions on Industrial Informatics* 14.11 (2018). cited By 626, pp. 4724–4734. DOI: 10.1109/TII.2018.2852491 (cit. on p. 16).

[29] P. Sthapit, H.-S. Gang, and J.-Y. Pyun. "Bluetooth Based Indoor Positioning Using Machine Learning Algorithms". In: *Proceedings of the International Conference on Consumer Electronics - Asia (ICCE-Asia)*. cited By 24. IEEE, 2018, pp. 206–212. DOI: 10.1109/ICCE-ASIA.2018.8552138 (cit. on pp. 18, 25, 26, 60).

[30] A. Tricco et al. "A scoping review of rapid review methods". In: *BMC medicine* 13 (Sept. 2015), p. 224. DOI: 10.1186/s12916-015-0465-6 (cit. on p. 19).

[31] A. Tundis, A. Faizan, and M. Mühlhäuser. "A feature-based model for the identification of electrical devices in smart environments". In: *Sensors (Switzerland)* 19.11 (2019). cited By 11. DOI: 10.3390/s19112611 (cit. on p. 22).

[32] V. K. Vaishnavi and W. L. Kuechler. "Design Science Research in Information Systems". In: *Ais* (2004). Accessed: 2022/02/07, pp. 1–45. DOI: 10.1007/978-1-4419-5653-8. URL: http://www.desrist.org/design-research-in-information-systems/ (cit. on p. 4).

[33] L. Xu, W. He, and S. Li. "Internet of things in industries: A survey". In: *IEEE Transactions on Industrial Informatics* 10.4 (2014). cited By 2834, pp. 2233–2243. DOI: 10.1109/TII.2014.2300753 (cit. on p. 16).

APPENDIX A

Web Application Database Tables

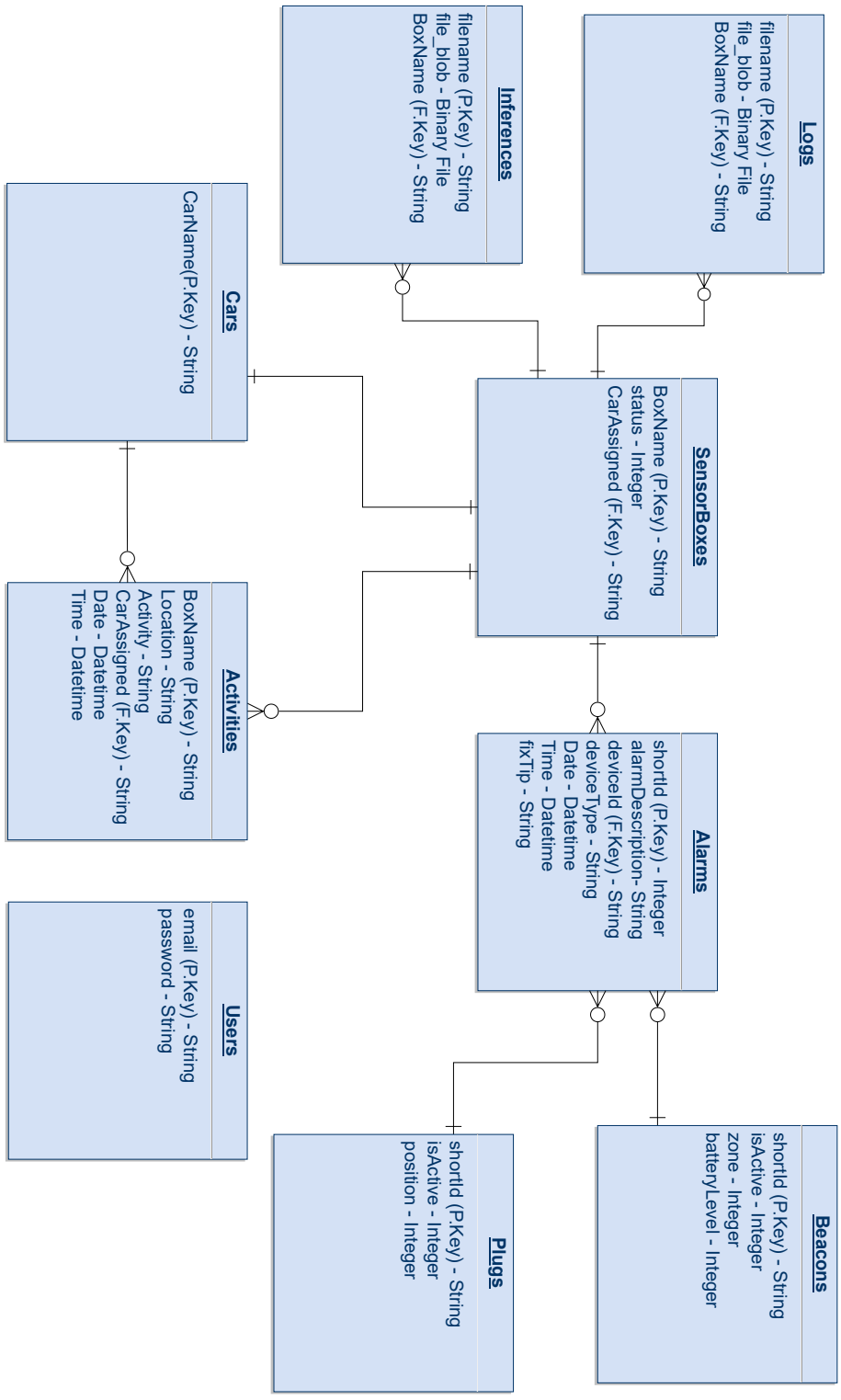

Figure A.1: Web Application Database Tables.

APPENDIX B

Web Application API

In every endpoint when using {var}, means the var is passed as json body through the request. When using param{var} means the var is passed as request query parameter.

In every request, the token is passed as an 'Authorization' header, and is always verified before any other endpoint action.

Energy Meters:

- **Get** /plugs/ - Return all registered energy meters data;
- **Put** /plugs/replacement - Change plug {PlugId} to position {NewPosition};
- **Get** /plugs/positions - Return all positions with no connected energy meter.

Beacons:

- **Get** /beacons - Return all registered beacons data;
- **Put** /beacons/replacement - Change beacon {beaconId} to position {newPosition} and couple beacon;
- **Get** /beacons/positions - Return all positions with no coupled beacon.
- **Put** /beacons/state - Change state of beacon {beaconId} to {desiredState} (decoupled or coupled).

Sensor Boxes:

- **Get** /sensorboxes - Return all registered sensor boxes data;
- **Get** /sensorboxes/reachable - Return if each registered sensor box is connected to the internet and if the vibration sensor is well connected to the corresponding box;
- **Get** /sensorboxes/car - Return sensor box param{boxId} assigned car;

- **Get** /sensorboxes/data - Return status (if to record data) and car assigned of sensor box param{boxId};
- **Put** /sensorboxes - Update sensor box assigned car to {newCarAssigned};
- **Put** /sensorboxes/state - Update sensor box {sensorBoxId} status (if to record data) to {desiredState}.

Cars:

- **Get** /cars - Return all registered cars data, from the *Carter of Turim Model* work ([20]);
- **Get** /cars/database - Return all registered cars data, saved in the database;
- **Post** /cars - Add the new car {"newCarAssigned"} in the database.

Inferences Files:

- **Get** /inferences - Return all available inference files filename;
- **Get** /inferences/blob - Return inference file param{filename} blob (binary File);
- **Post** /inferences - Add the new inference file param{filename}, with blob {"file_data"} and box paramboxId.

Sensor Boxes Log Files:

- **Get** /logs - Return all available logs files filename;
- **Get** /logs/blob - Return log file param{filename} blob (binary File);
- **Post** /logs - Add the new log file {filename}, with blob {"file_data"} and box {boxId}.

Alarms:

- **Get** /alarms - Return all registered alarms data.
- **Get** /alarms/device - Return all alarms data of device param{deviceId};
- **Post** /alarms/beaconsbattery - Add new low battery beacons alarms {lowBatteryBeaconsDictionary};
- **Post** /alarms/boxconnection - Add new bad connected sensor box alarm of sensor box param{boxId};
- **Post** /alarms/vibration - Add new bad connected vibration sensor of sensor box param {boxId};

- **Post** /alarms/shellys - Add new energy meter disconnection alarm of energy meter param{plugId}. Used also to change energy meter param{plugId} connection status.

Activities:

- **Get** /activities - Return all registered activities data.

Authentication:

- **Post** /login - Return access token if password pass matches email {email} user account;
- **Put** /login/newpassword - Updates password of user account with email {email} to {password}.

APPENDIX C

Web Application Pages

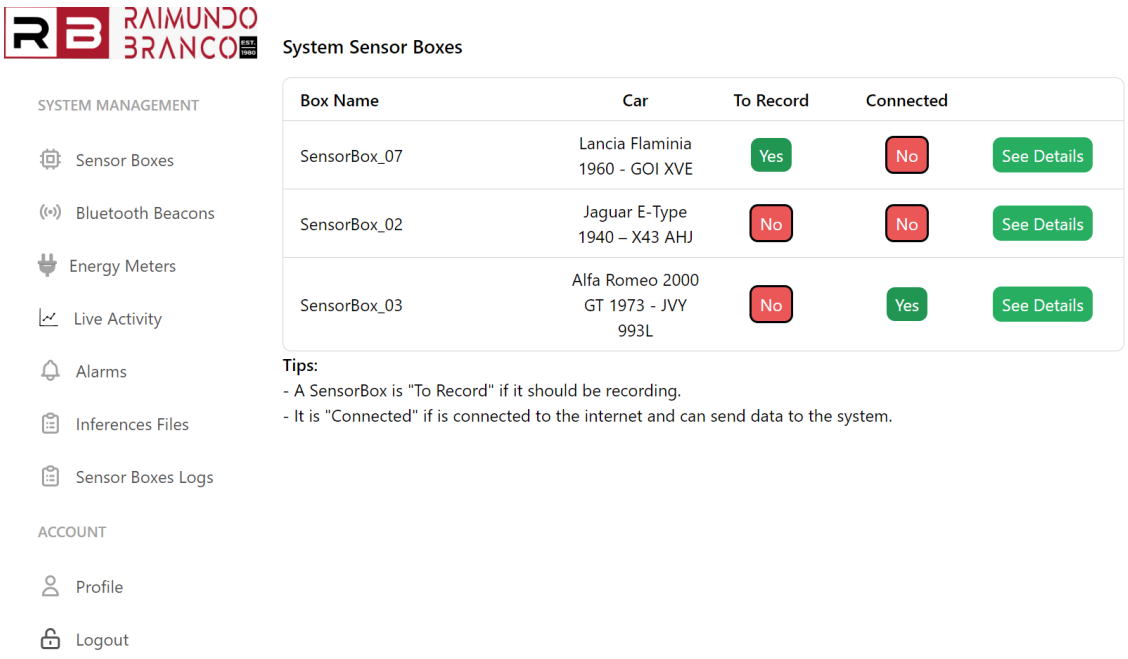

Figure C.1: WebApp Sensor Boxes List Page.

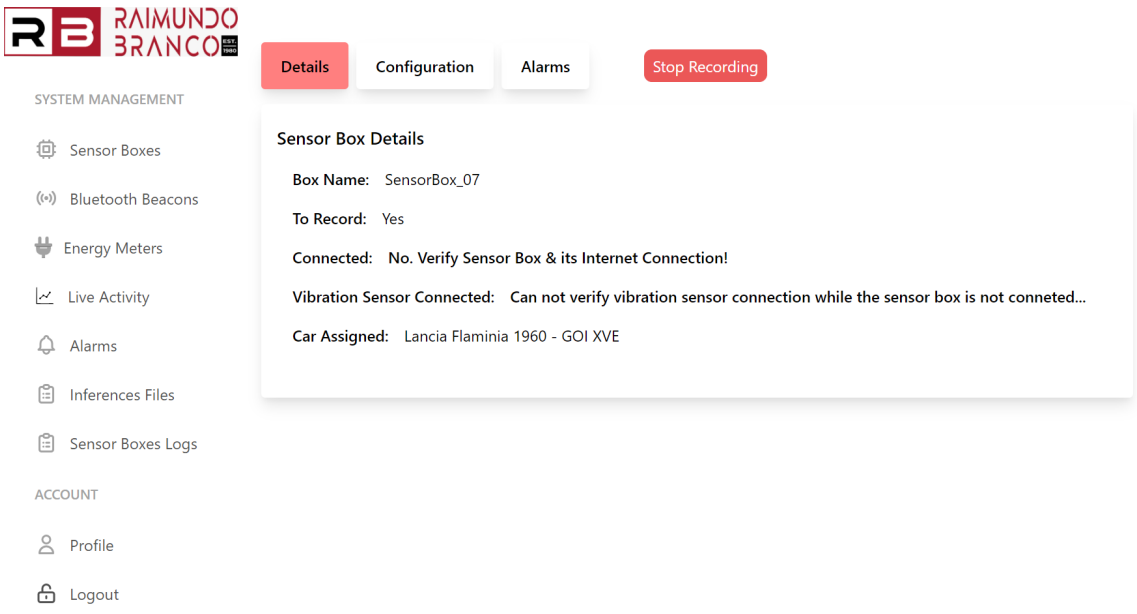

Figure C.2: WebApp Sensor Box Details Page.

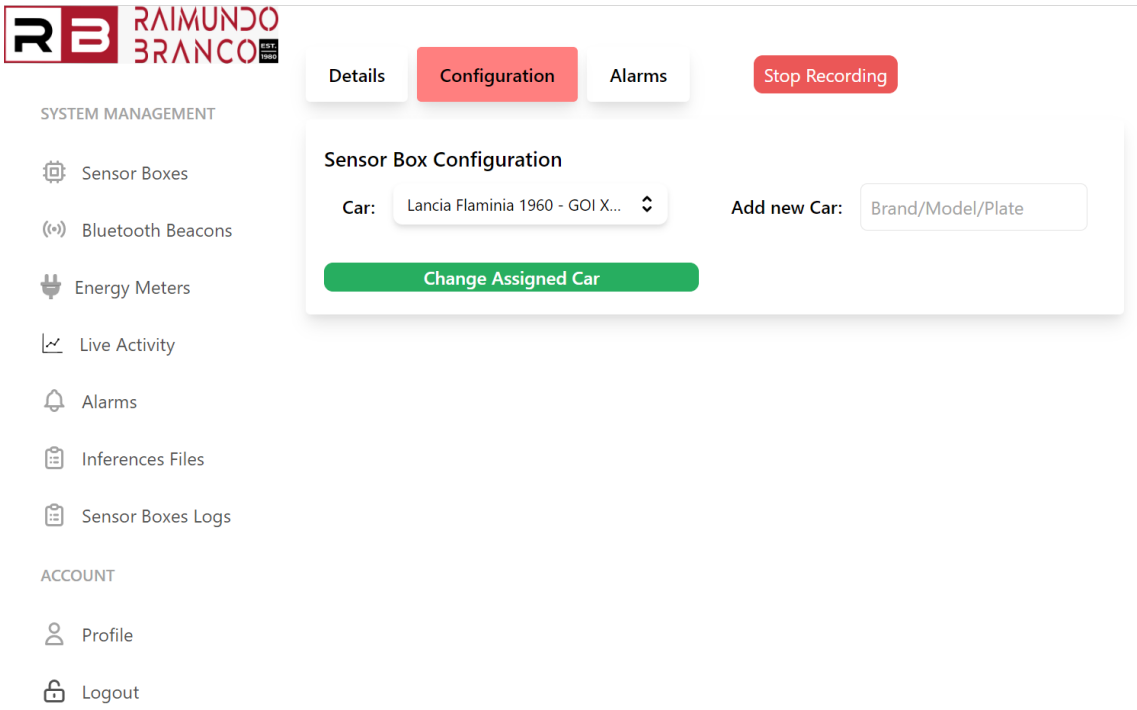

Figure C.3: WebApp Sensor Box Configuration Page.

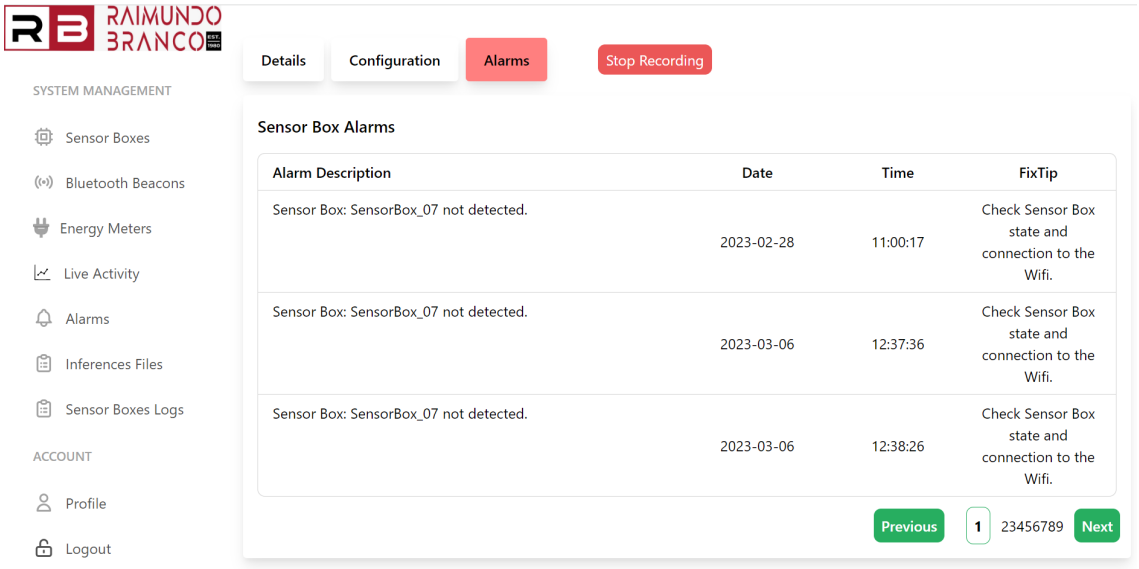

Figure C.4: WebApp Sensor Box Alarms Page.

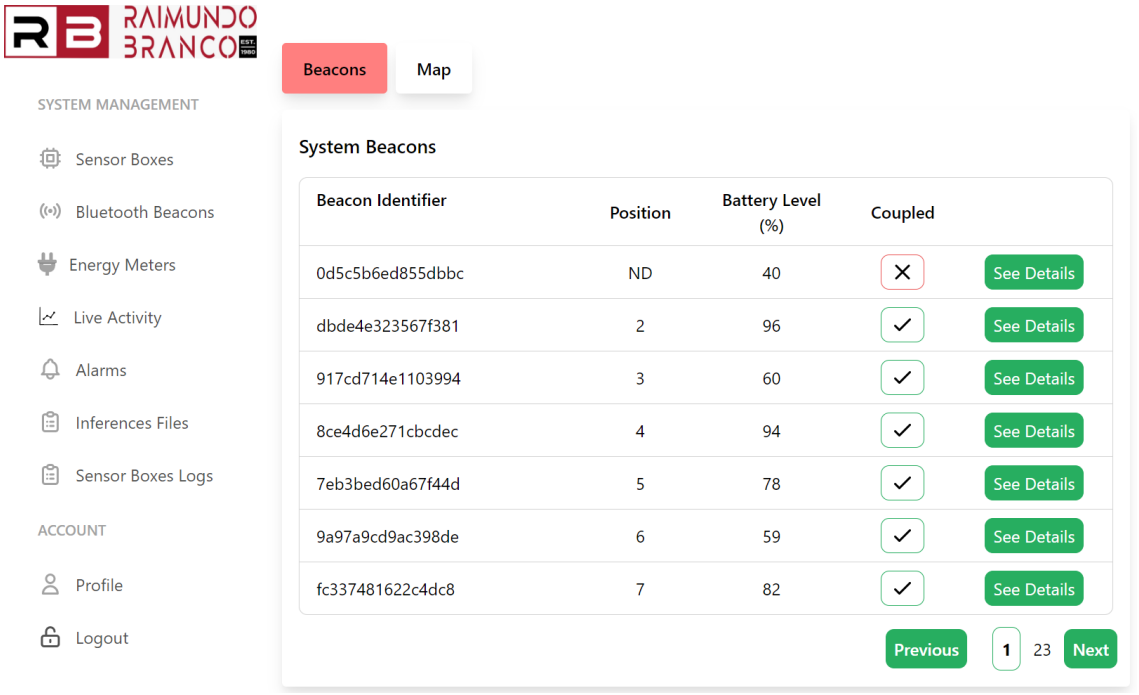

Figure C.5: WebApp Beacons List Page.

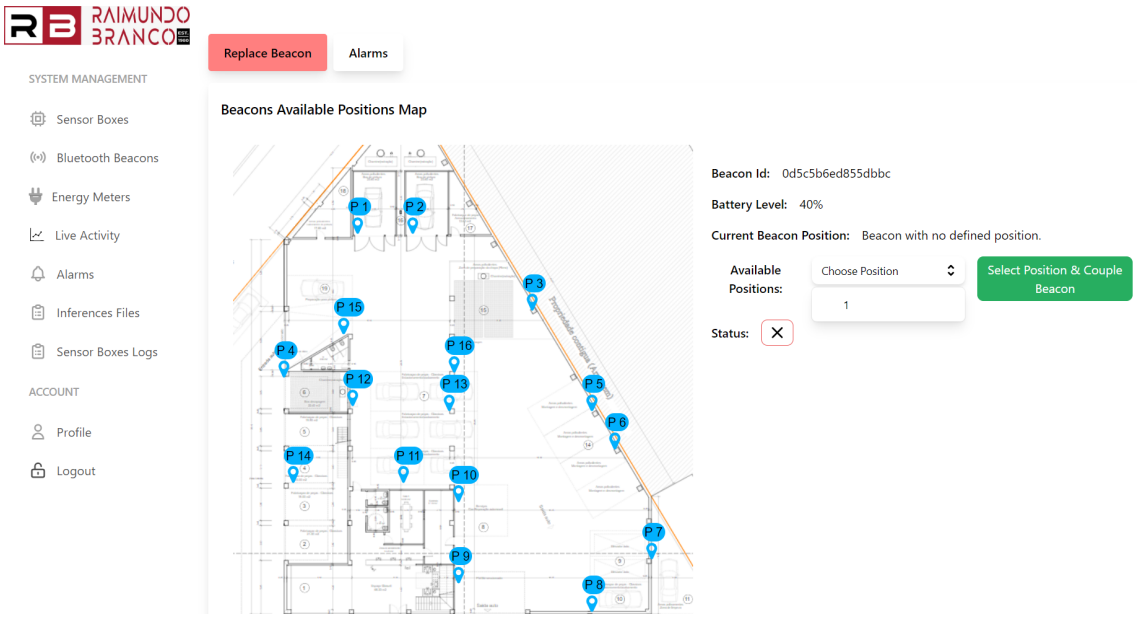

Figure C.6: WebApp Beacon Replacement Page.

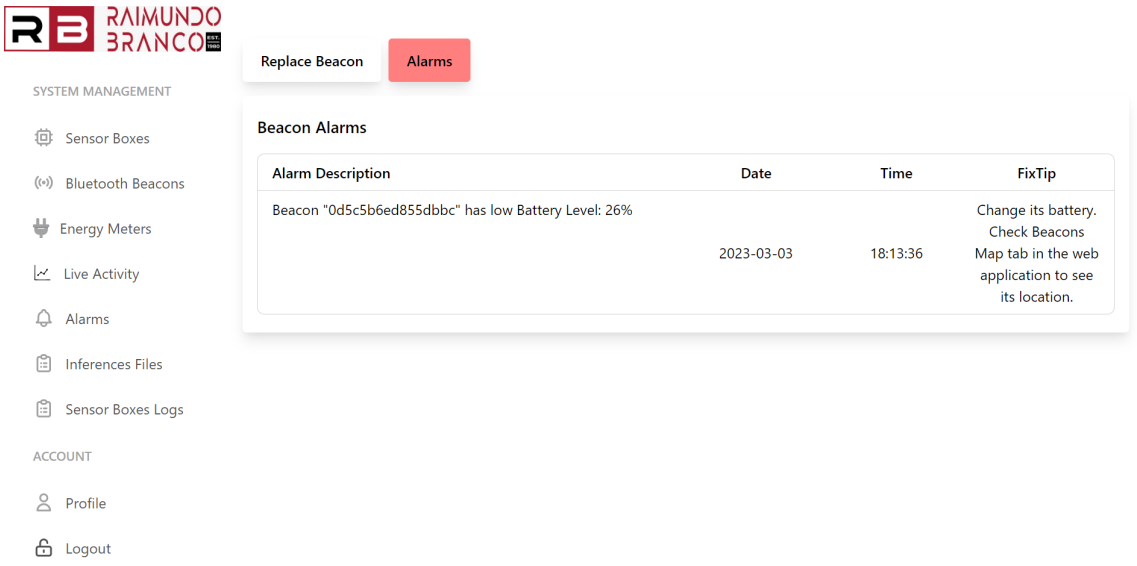

Figure C.7: WebApp Beacon Alarms Page.

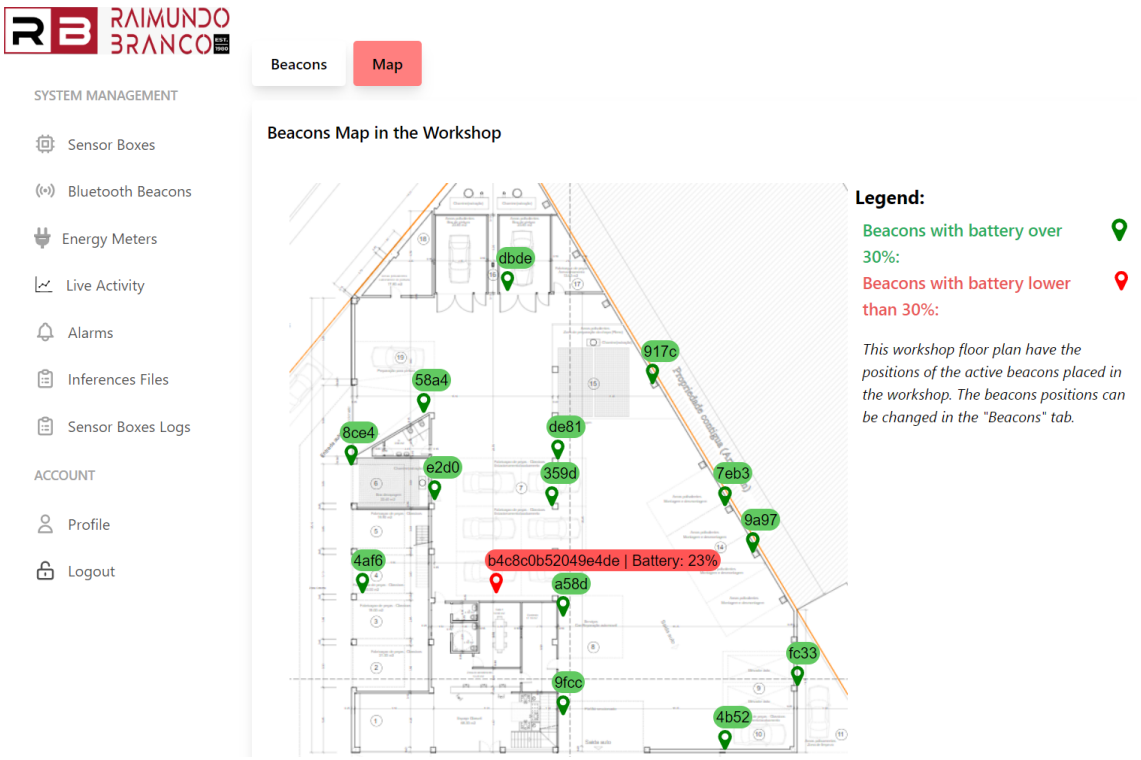

Figure C.8: WebApp Beacon Map Page.

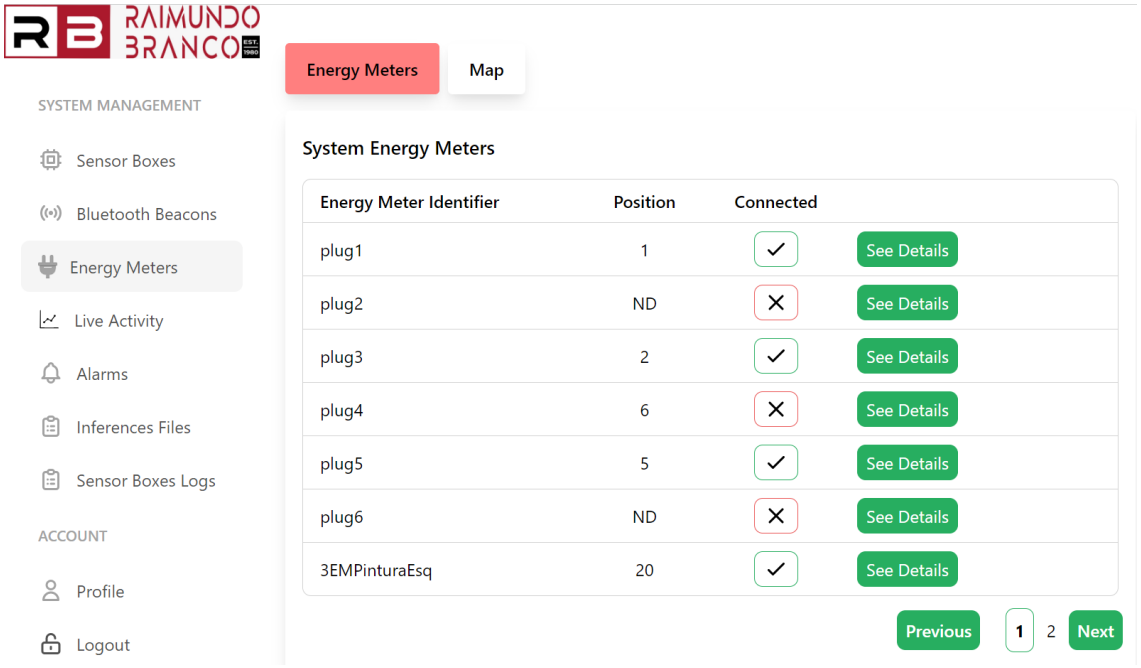

Figure C.9: WebApp Energy Meters List Page.

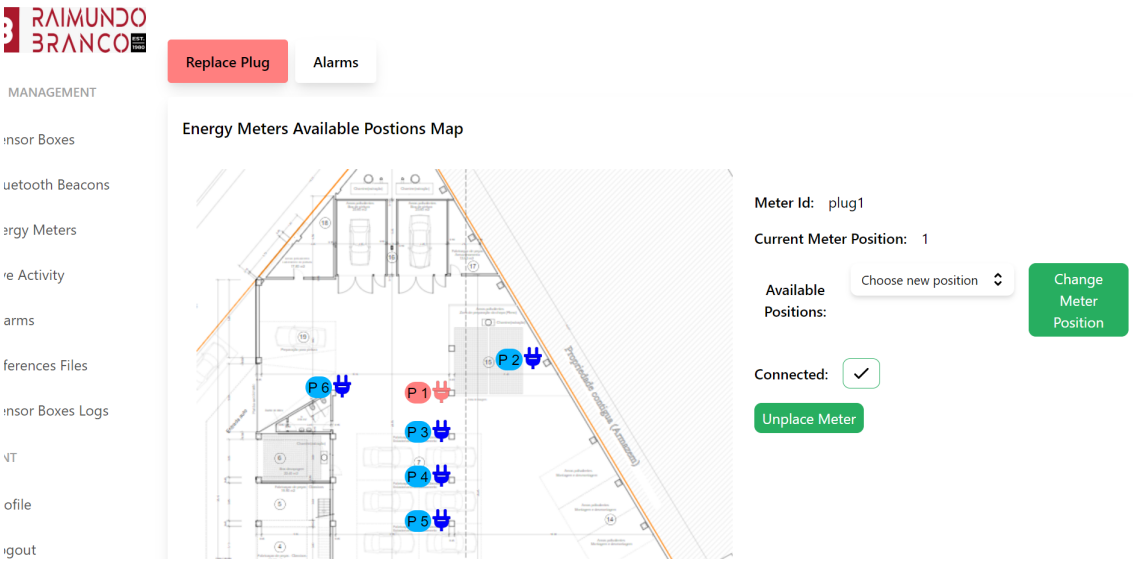

Figure C.10: WebApp Energy Meter Replacement Page.

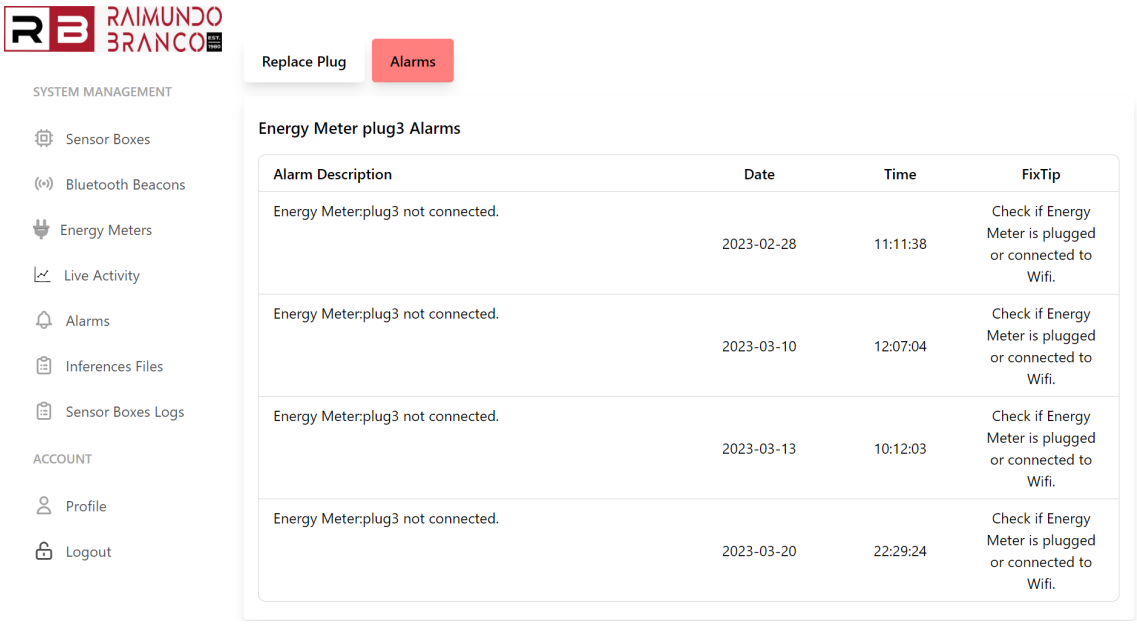

Figure C.11: WebApp Energy Meter Alarms Page.

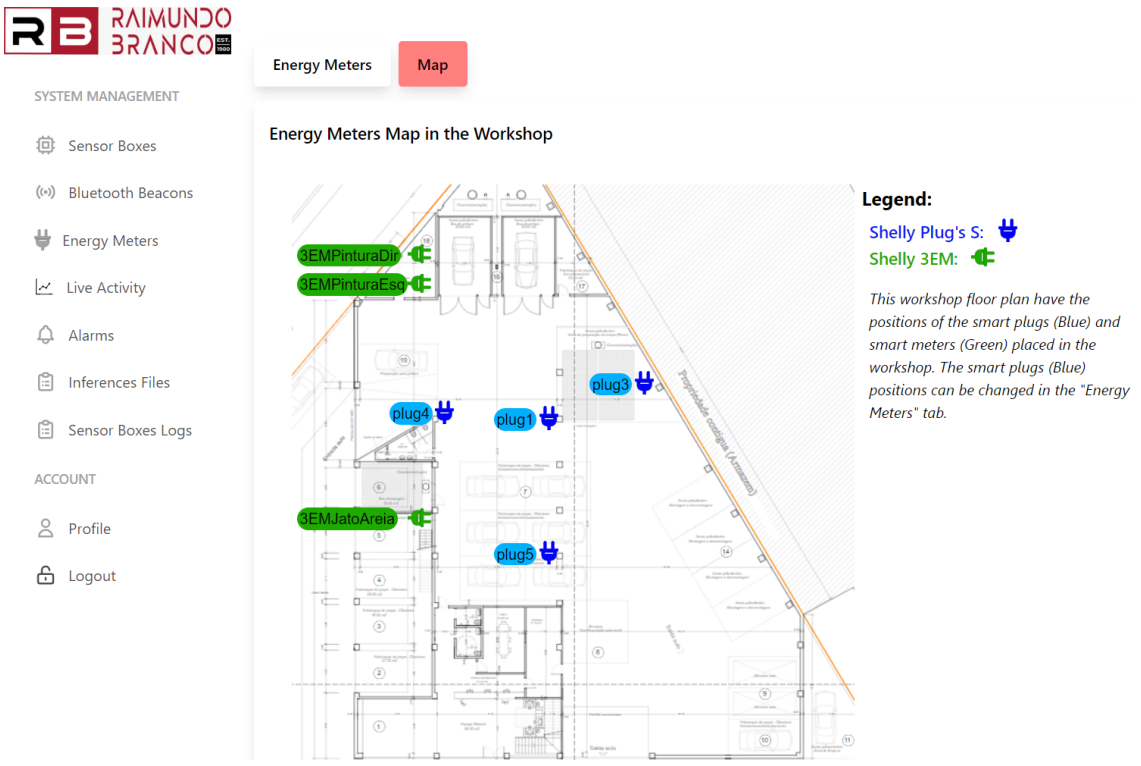

Figure C.12: WebApp Energy Meters Map Page.

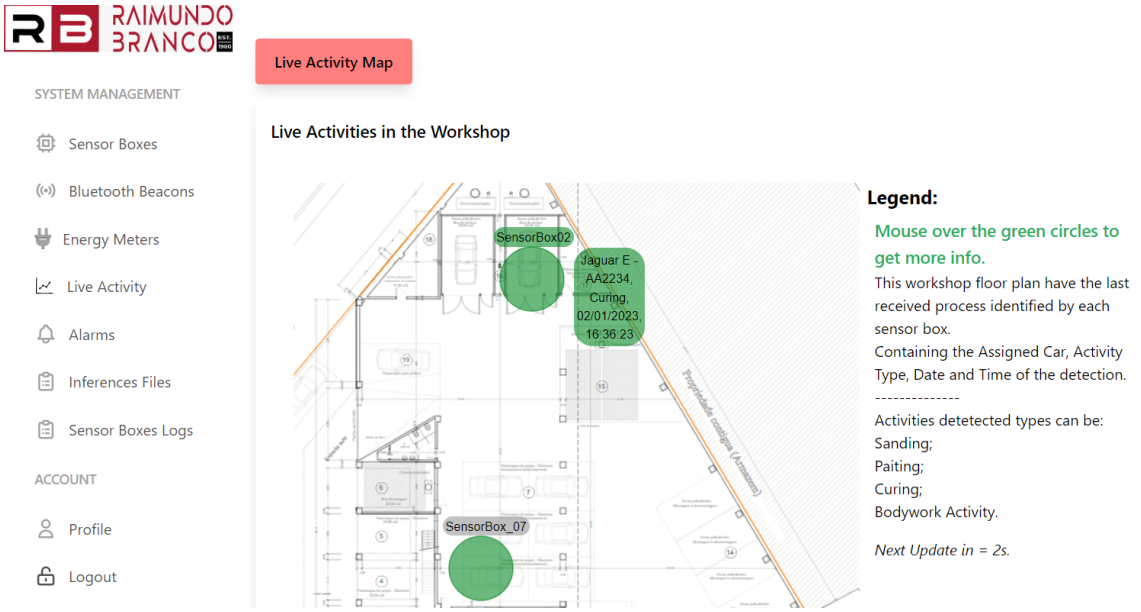

Figure C.13: WebApp Activities Live Map Page.

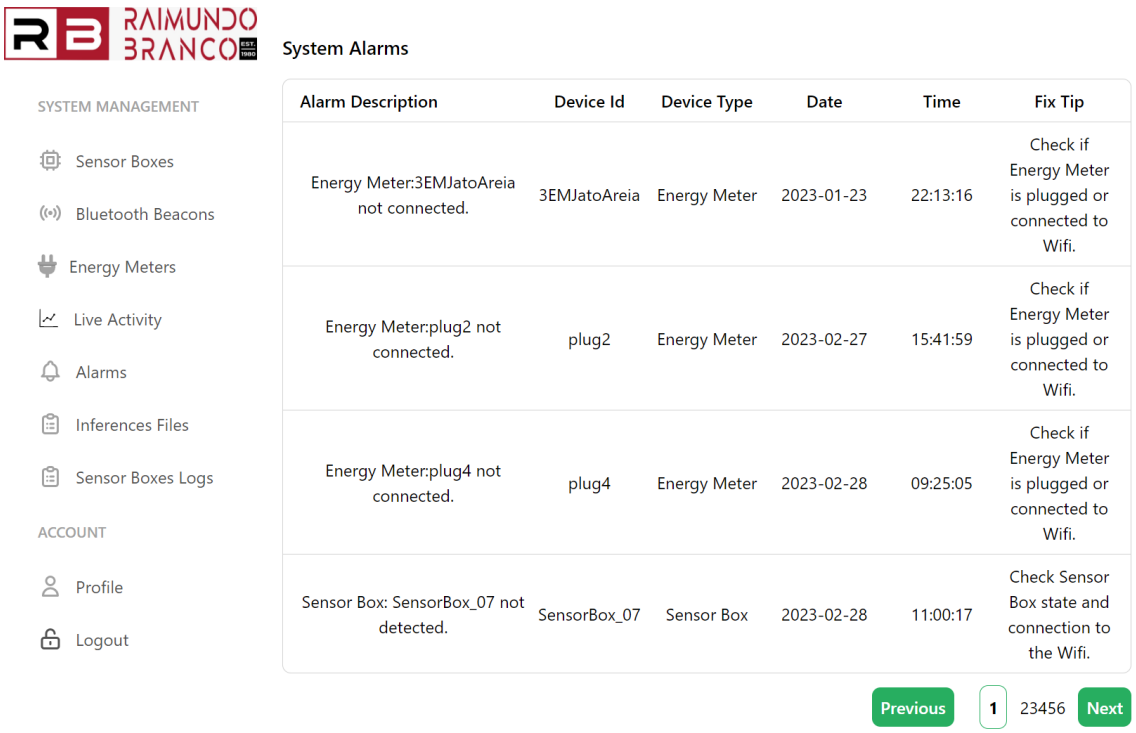

Figure C.14: WebApp All Alarms List Page.

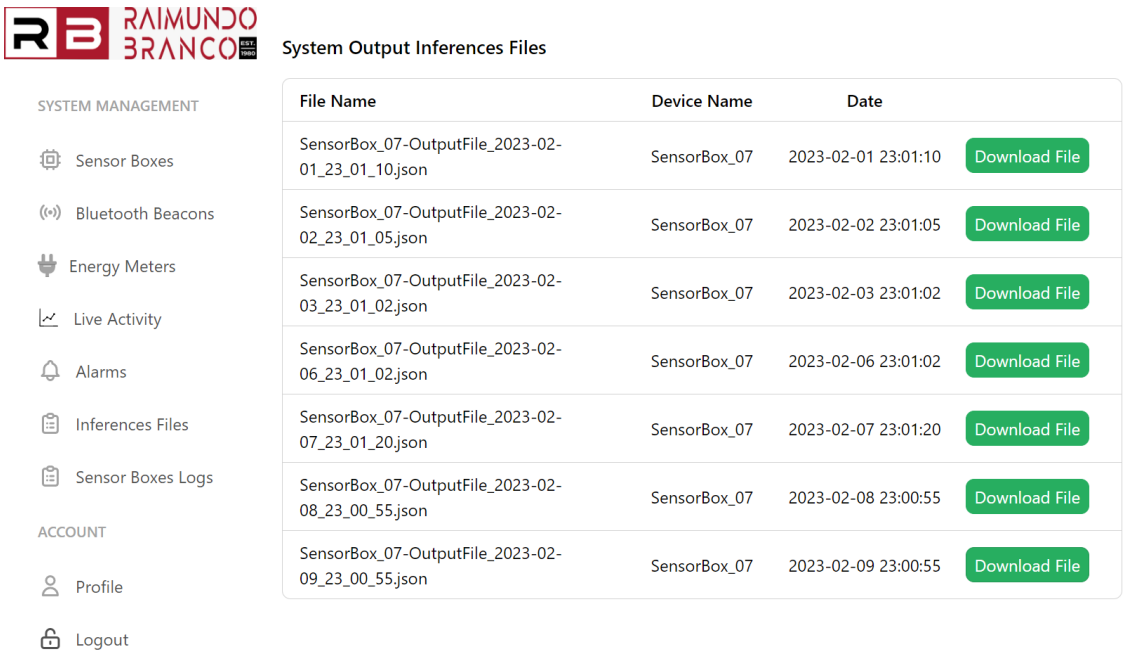

Figure C.15: WebApp Inference Files List Page

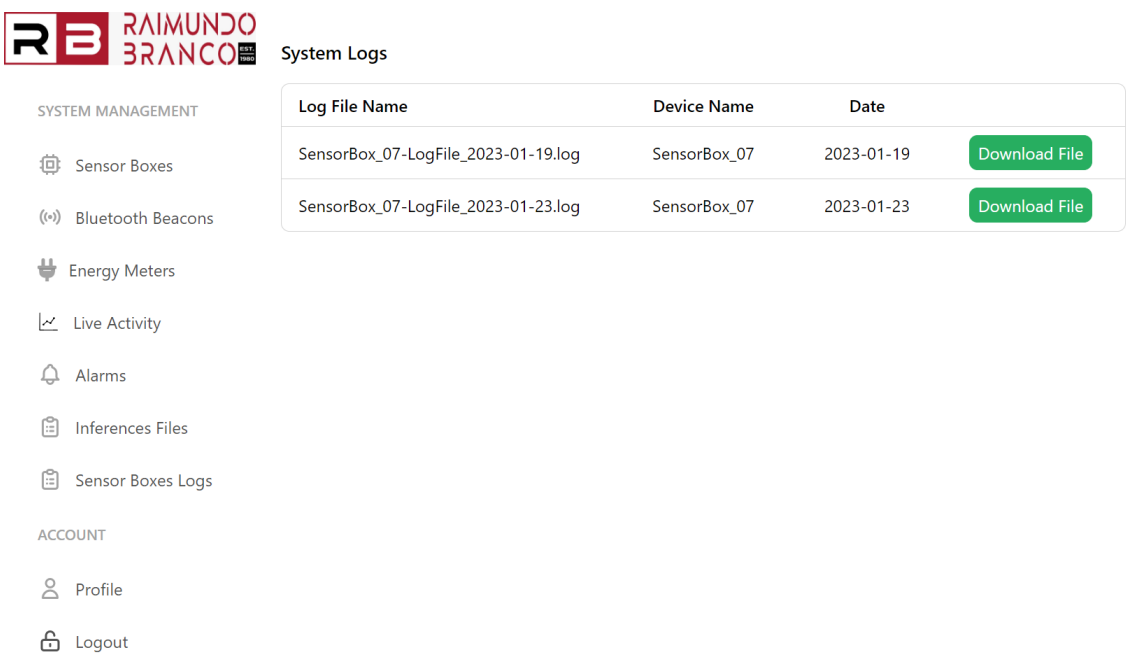

Figure C.16: WebApp Sensor Boxes Log Files List Page

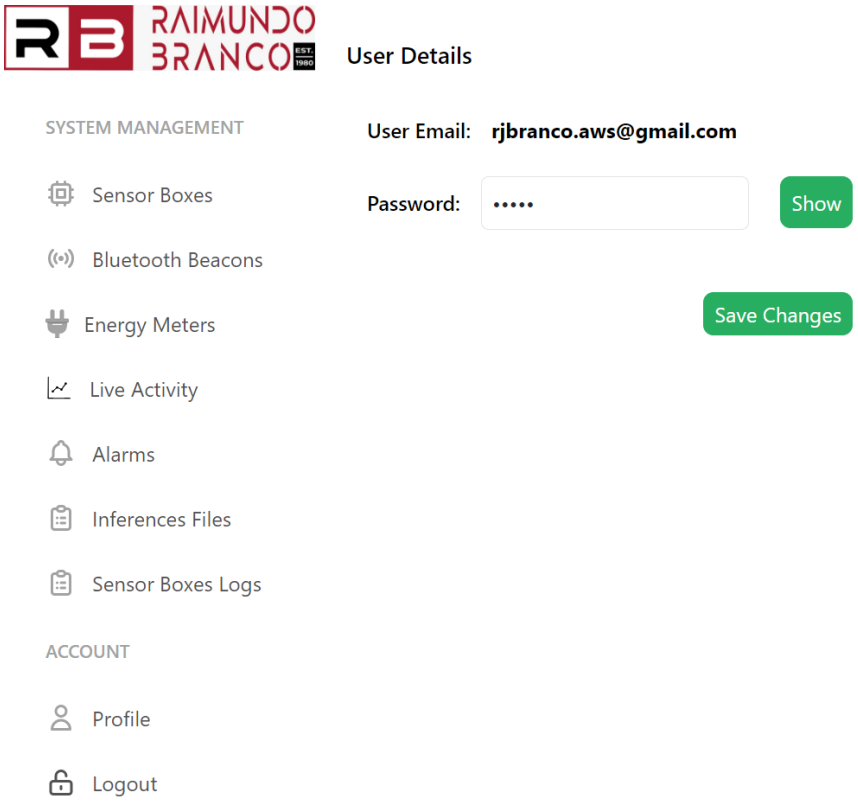

Figure C.17: WebApp Password Change Page