



# Robot CeDRI 2023: Sub-system Integration and Health Dashboard

**Andre Luis França**

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master Degree in Industrial Engineering.

Work oriented by:

Dr. André Chaves Mendes

Dr. Eduardo de Freitas Rocha Loures

Dra. Luísa Maria Garcia Jorge

Bragança

2023


# Robot CeDRI 2023: Sub-system Integration and Health Dashboard

**Andre Luis França**

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master Degree in Industrial Engineering.

Work oriented by:

Dr. André Chaves Mendes

Dr. Eduardo de Freitas Rocha Loures

Dra. Luísa Maria Garcia Jorge

Bragança

2023

# Dedication

Après un voyage à l'étranger et un an de travail acharné, j'ai fini ma thèse. Avant toute chose, je tiens à exprimer ma gratitude à mes amis d'ici et d'avant, mais surtout à ma mère, mes grands-parents et ma compagne pour leur soutien inconditionnel. Sans eux, je n'aurais jamais pu aller aussi loin, ni même m'aventurer de l'autre côté de l'Atlantique. C'est à vous tous que je dédie cette thèse avec tout mon cœur et toute ma gratitude.

# Abstract

With the constant increase in the volume of data generated and collected in several areas, data visualization has become more relevant to improve equipment management, reduce operational costs and increase process efficiency. This paper proposes developing a health monitoring system for an Autonomous Mobile Robots (AMR) equipment, which allows data acquisition and analysis for decision-making performed autonomously and by the equipment manager. Implementing the proposed system demonstrated favourable results in data acquisition, analysis, and visualization for decision-making. Using a hybrid control architecture, the data acquisition and processing showed to be effective, without significant impacts on the battery consumption or in the use of microcomputer resources embedded in the AMR. The developed dashboard demonstrated efficient data navigation and visualization, providing essential tools for decision-making by the platform administrator. This work contributes to the health monitoring of types of equipment as AMRs. It may be of interest to professionals and researchers in areas related to robotics and automation, especially those who work with equipment that uses Robot Operating System (ROS). Besides, the developed system is open-source, making it accessible and customizable in different contexts and applications.

**Keywords:** Autonomous Mobile Robots; Data Visualization; Dashboard

# Resumo

Com o aumento contínuo da quantidade de dados produzidos e coletados em diversas áreas, a visualização de dados tem se mostrado cada vez mais relevante para melhorar a manutenção de equipamentos, reduzir custos operacionais e aumentar a eficiência de processos. Este trabalho propõe o desenvolvimento de um sistema de monitoramento da saúde de um equipamento do tipo Autonomous Mobile Robots (AMR), que permita a coleta e análise de dados para tomadas de decisão realizadas tanto autonomamente quanto pelo gestor da plataforma. A implementação do sistema proposto apresentou resultados favoráveis na coleta, análise e visualização de dados para a tomada de decisões. Utilizando uma arquitetura de controle híbrida, a aquisição e processamento dos dados mostrou-se eficiente, sem impactos significativos no consumo de bateria ou uso de recursos do microcomputador embarcado no AMR. O dashboard desenvolvido mostrou-se eficiente na navegação e visualização dos dados, fornecendo ferramentas importantes para a tomada de decisão do gestor da plataforma. Este trabalho contribui para a monitorização de saúde de equipamentos como AMRs, podendo ser de interesse para profissionais e pesquisadores em áreas relacionadas à robótica e automação, em especial aqueles que trabalham com equipamentos que utilizam do Robot Operating System (ROS). Além disso, o sistema apresentado é open-source, tornando-o acessível e personalizável para uso em diferentes contextos e aplicações.

**Palavras-chave:** Autonomous Mobile Robots; Data Visualization; Dashboard

# Contents

# List of Tables

# List of Figures

# List of Algorithms

# Acronyms

**AGV** Automated Guided Vehicle.

**AHP** Analytic Hierarchy Process.

**AMCL** Adaptive Monte Carlo Localization.

**AMR** Autonomous Mobile Robots.

**API** Application Programming Interface.

**BPMN** Business Process Model and Notation.

**BSON** Binary JSON.

**CeDRI** Research Centre in Digitalization and Intelligent Robotics.

**CNRI** Corporation for National Research Initiatives.

**CPU** Central Processing Unit.

**CWI** Centrum Wiskunde & Informatica.

**DBMS** Database Management System.

**DSF** Django Software Foundation.

**GUI** Graphical User Interface.

**HTTP** HyperText Transfer Protocol.

**JS** JavaScript.

**JSON** JavaScript Object Notation.

**KPI** Key Performance Indicator.

**LiDAR** Light Detection and Ranging.

**MCDM** Multiple Criteria Decision Making.

**MIT** Massachusetts Institute of Technology.

**NoSQL** Not only SQL.

**PSF** Python Software Foundation.

**RDBMS** Relational Database Management System.

**ROS** Robot Operating System.

**RTT** Round Trip Time.

**SQL** Structured Query Language.

**TOPSIS** Technique for Order Preference by Similarity to Ideal Solution.

**UAV** Unmanned Aerial vehicle.

**UGV** Unmanned Ground Vehicle.

**URL** Uniform Resource Locator.

**UTF-8** 8-bit Unicode Transformation Format.

**WWW** World Wide Web.

# Chapter 1

# Introduction

Through the evolution of the industry, autonomous vehicles such as Autonomous Mobile Robots (AMR) and Automated Guided Vehicle (AGV) have become increasingly common equipment for the internal logistics of companies, handling the optimization of logistics processes and increased productivity [1], [2]. Meanwhile, for properly operating these types of equipment, it is fundamental to monitor their operational data effectively [1], [3].

In this context, the area of Big Data Visualization has been highlighted for the management of data and decision-making in the industry field, allowing the transformation of complex data into attractive and visually comprehensible information [4]–[6]. Within this context, data visualization becomes a powerful tool for the managers and operators of the equipment, allowing fault detection and aiding in predictive maintenance and strategic equipment decisions.

In addition to data visualization, centralized and decentralized control architectures are essential for data acquisition in industrial automation systems. In centralized systems, a central unit monitors and controls the equipment, which eases data acquisition. On the other hand, in decentralized systems, each agent is autonomous, and the information exchange happens between the agents. This way, the data capture is more complex [3], [7].

This work describes the development of a system for monitoring the health of a AMR, where the control of its systems uses the Robot Operating System (ROS). Through the acquisition, treatment, and storage of the acquired data, it supplies instruments for the system managers to perform strategic decisions related to the operation and health of this vehicle.

Therefore, this work aims to propose and implement a solution to improve the management of robotic equipment by a health dashboard, applying Big Data Visualization techniques to aid in comprehending the data generated by the equipment.

## 1.1 Objectives

This work intends to develop a system capable of communicating with the Magni robotic platform owned by the Research Centre in Digitalization and Intelligent Robotics (CeDRI). The system must acquire data that contributes to the maintainability of the equipment without affecting its autonomous capacities or significantly impacting the use of its onboard resources, such as batteries and microcomputer use. In addition, the developed system must provide platform administrators with the ability to interpret the collected data through a visual and interactive application.

The work has as specific objectives the following aspects:

- Develop tools to monitor and collect data from the robotic platform and its systems;
- Create a database that concentrates the data collected and allows for future analysis;
- Provide the platform manager's tools to assist in the decision-making process for equipment health;
- Maintain the development of tools tied to open-source services and products.

## 1.2 Document Structure

This document is divided into six chapters, which follow the objectives and structure described below:

- Chapter 1: presents an introduction to the topic of data visualization and the importance of equipment management platforms, followed by the objectives of the development of this work, and concluding with the presentation of the structure of this document;
- Chapter 2: supplies a review of Autonomous Mobile Robots, highlighting characteristics such as the core tasks expected from these devices and the control architectures. Additionally, it presents the concepts and tools belonging to Big Data Visualization. In the end, it explores the main tools used in the development of this work;
- Chapter 3: describes the choice of the control architecture, sequentially elaborating the information flow and describing the processes and sub-processes proposed to achieve the defined goals. Finally, it presents the proposal for the user interface;
- Chapter 4: describes the development of the system, beginning with the choice of tools and then exploring the implementation of the processes defined in the earlier chapter. Finally, it presents the development and composition of the user interface;
- Chapter 5: shows the results of the tests performed to determine if the implemented system conforms to the established objectives. Furthermore, it examines the impact of the execution of the processes implemented in the resources of the robotic platform. Finally, the developed project is benchmarked against a commercial product;
- Chapter 6: concludes the work, presenting considerations regarding the obtained results and proposals for future work.

# Chapter 2

# State of the art and Study of tools

This chapter introduces AMRs and the main control architectures used in these types of equipment and also bring concepts about Big Data Visualization and databases, besides briefly exploring the tools used in the development of the project, presenting some available techniques and methodologies.

## 2.1 Autonomous Mobile Robots

AMRs are vehicles similar to AGVs, where both are subcategories of the so-called Unmanned Ground Vehicle (UGV), having in common the aspect of autonomy in their movement, but with subtle differences between their capabilities. UGVs are any ground vehicle that does not require the physical presence of an operator in the equipment, and these can be teleoperated or autonomous [1], [8], as are the equipment of interest in this work.

AGVs were introduced in 1953 by Berret Electronic of Northbrook, Illinois, USA. Since then, they have been used mainly in factory logistics operations and warehousing, guiding themselves by magnetic and optical tracks. Although AGVs can move autonomously, treating routes and following the most efficient paths according to the moment, these types of equipment have limitations in their locomotion, having to follow through with previously defined tracks. [1], [9].

The AMRs differ from the AGVs mainly in the aspect of navigation, as they can move freely in the environment where it is immersed, as shown in figure 2.1, making them ideal for dynamic environments adapting and correcting their route constantly according to the need [1], [9], [10].

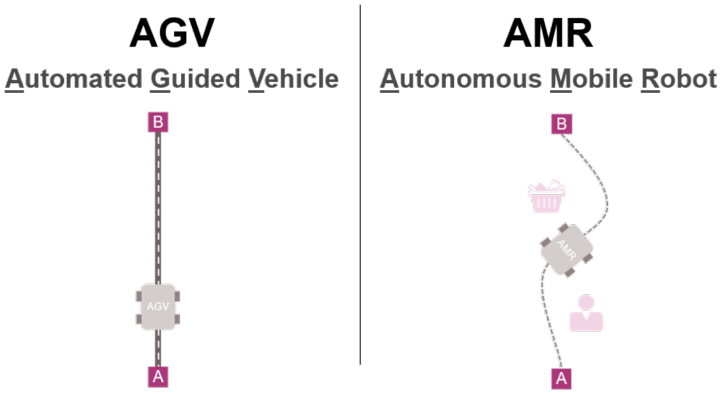



Figure 2.1: Navigation difference between AGV e AMR [11]

In a simplified way, AGVs have a behaviour similar to vehicles in a city, where they have streets and rules that they are obliged to follow. In contrast, AMRs are closer to a human in this city, where their freedom of locomotion is much greater than that of vehicles, being able to move in places where there is a well-defined pavement following it or in areas where there is no boundary of way, besides having the ability to deviate from obstacles in a dynamically and locomotion in highly dynamic environments.

### 2.1.1 Five core tasks

The AGVs and AMRs must perform some tasks and functions to operate the systems where they are inserted correctly and the vehicles themselves. According to De Ryck, Versteyhe, and Debrouwere [3], there are five core tasks that an AGV must perform, which are: Task Allocation, Localization, Path Planning, Motion Planning and Vehicle Management, as illustrated in figure 2.2.

- Task Allocation: this assigns the system elements a queue of tasks with the most efficient composition for the system. This task must have flexibility, scalability, and

responsiveness during its execution [3];

- Localization: this task is responsible for determining the position of the vehicle in the environment in which it is inserted. This task is frequently already embedded in the vehicle, and the information can then be shared with a central computer or with other vehicles, depending on the control architecture [3];

- Path Planning: this task has as its objective to trace a static route in the environment known by the AGV, not considering the dynamic elements of the environment [3];

- Motion Planning: is responsible for the dynamic planning of routes, avoiding collisions and deadlock situations or even delimiting control zones [3];

- Vehicle Management: finally, this task has the objective of management of aspects such as errors, battery, and maintenance to provide the maximum use of each vehicle in the system [3].

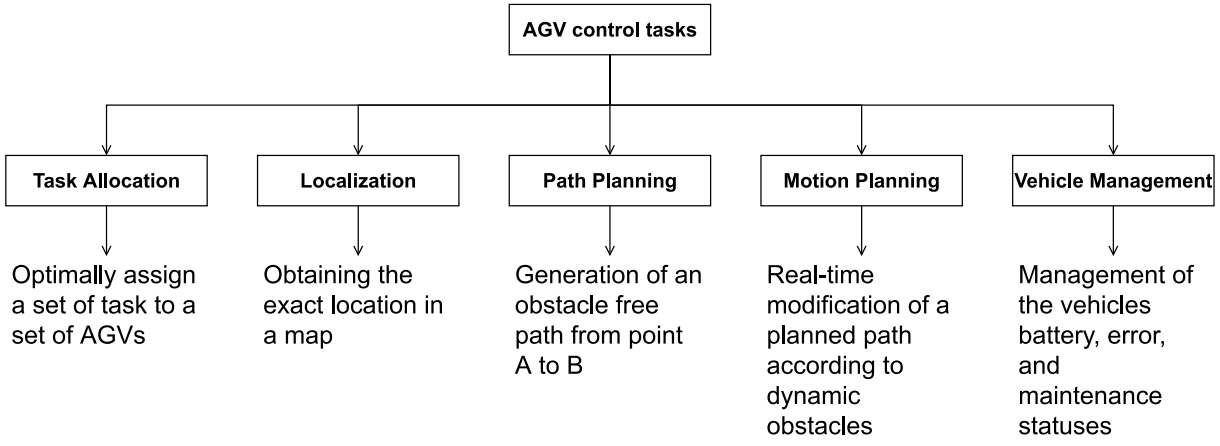
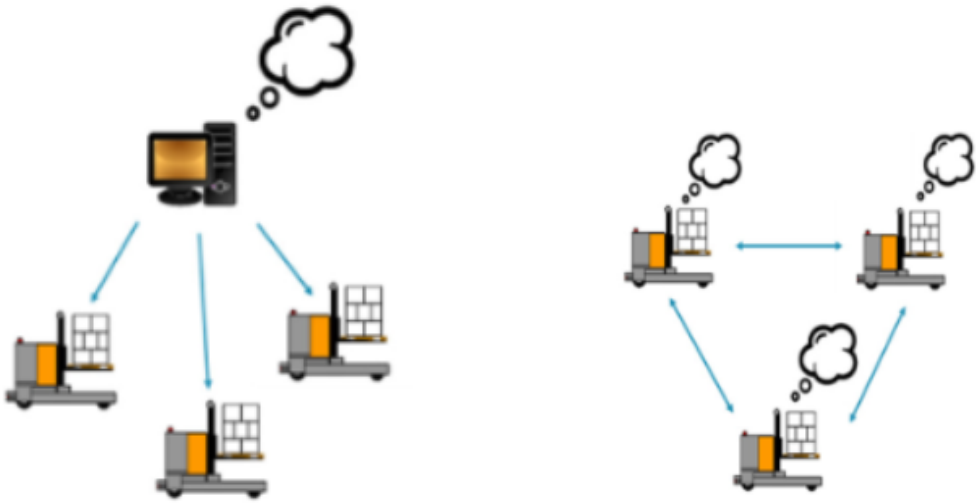



Figure 2.2: Overview of the five core AGV tasks (adapted from [3])

These tasks must be carried out by the vehicles themselves or by utilizing a centralized unit, which depends on the control architecture on which the system is based.

### 2.1.2 Control architecture

There are two main approaches to autonomous vehicle control centralized control and decentralized control [3]. In the first approach, there is a centralizing agent, which manages all the functions and decisions related to the equipment in the system, as illustrated in figure 2.3a. In the second form, the elements of the system have the capacity for self-management and communication with others. This way, it transfers a share of the function of the centralized unit to each element, as is observed in figure 2.3b.



(a) Central architecture (b) Decentralized architecture

Figure 2.3: Central and decentral architecture [3]

Through these architectures, there is a greater or lesser degree of access to these vehicles' data, parameters, and actions. These approaches also interfere with the capacity of the system to act at the maximum global or local efficiency, besides interfering in the speed of decision-making during the operation, which can be a critical issue in highly dynamic environments [3], [7], [12].

Below, the already mentioned control architectures are enumerated and described in more detail, besides introducing a third variant that seeks better use of the centralized and decentralized characteristics.

I) Centralized architecture:

The centralized control architecture is characterized by a central control unit planning and delegating orders and tasks to the system components. This control technique is already well implemented in industry and has a highly hierarchical behaviour, as illustrated by the ISA 95 automation pyramid figure 2.4 [3], [7].

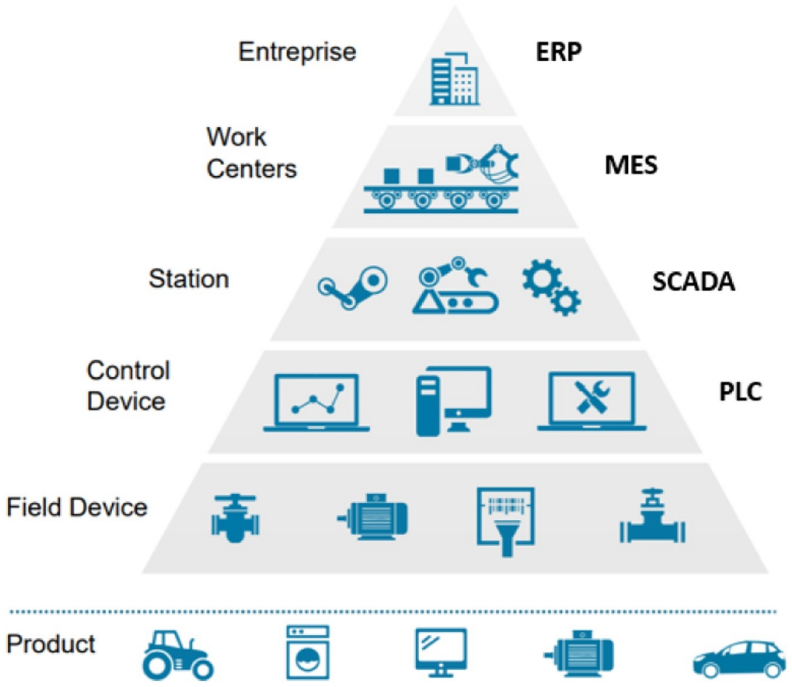



Figure 2.4: ISA 95 model [13]

In the centralized architecture, the whole context can be analyzed, enabling global system optimization and having information tracking and control over all aspects of production [7]. This control architecture requires the constant exchange of information between the hierarchical levels of the structure, feeding the central unit with the necessary information for the management of the system and sending commands to the agents [3], [7].

The latency in this communication chain and the information processing makes it difficult to manage the system efficiently in dynamic environments [3], [7]. The fact that the design is linked to the central control unit limits the modifiability and extensibility of the system [7].

II) Decentralized architecture:

The decentralized control architecture is suitable when there is a need for more adaptive responses to events in the environment, that is, systems in which several events occur during the operation and require corrections, as a counterpoint to the slow response coming from the centralized method [12].

In the decentralized architecture, the system's intelligence is not concentrated in a single orchestrator point but distributed among the elements that compose the whole structure. The communication between the agents ensures the efficiency of the entire process, allowing unexpected events in the environment to be processed and lead to a decision more dynamically when compared to the centralized method because the decision-making is taken directly by the elements of the system. Despite the benefits of this control architecture, the system has difficulty controlling the elements individually or collectively because each one is trying to achieve its goals. This behaviour does not necessarily move the system to the point of global optimization [3], [7].

For lower complexity systems, adopting decentralized architecture may not be ideal since, in this case, the centralized architecture would guarantee robustness and flexibility. However, due to the limitations of the centralized architecture, in more extensive and complex systems, the decentralized architecture stands out [3].

III) Hybrid architecture:

The hybrid architecture emerges as a mix of the two architectures already presented to minimize the disadvantages and maximize the benefits provided by each control methodology. This architecture aims to find a good balance point between centralized and decentralized control, but it also brings essential characteristics for Industry 4.0 as connectivity between machines, communication, and cooperation. Because it uses in part an already well-established architecture in the industry, it implements hybrid architectures more attractive than a radical change from a centralized to a decentralized system [3], [7].

An example of a hybrid architecture system would be a superior unit that creates a queue of tasks for the system elements. However, with decentralized control, the system's elements can change the queue's order and redistribute among the other elements, thus proposing a new queue. In the same way, the superior control unit can reject the changes proposed by the elements based on the information available from the system, thus seeking the point of greatest efficiency [7].

### 2.1.3 The mobile robot platform

The robotic platform this project is based on is a Magni Silver Robot Base, shown in figure 2.5, a platform developed and produced by Ubiquity Robotics based in California. Being a pre-built solution with navigation, power, computing, and mobility systems already available for use shortens the application's development period in which the platform will act [14].

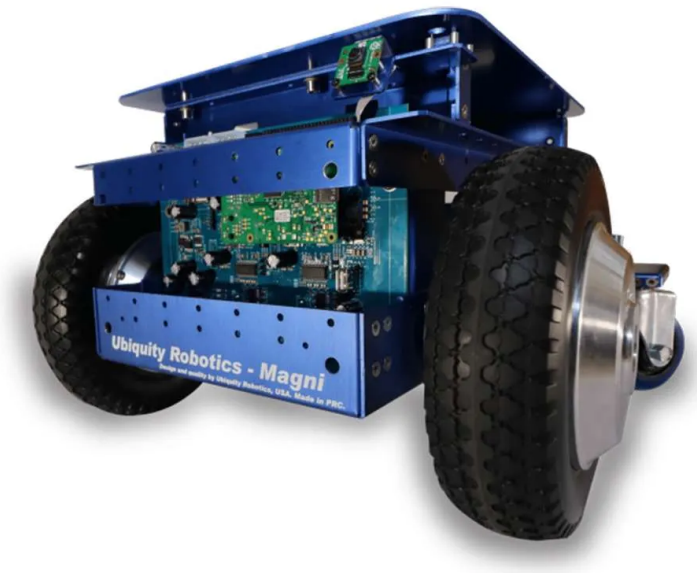



Figure 2.5: Robot Base Magni Silver [14]

The platform acquired by the CeDRI have all software systems embedded in Raspberry Pi 4 microcomputers powered by Ubuntu 20.04 and uses the ROS Noetic to run its systems, besides having features such as a load capacity of approximately 100 kg; a set of two motors equipped with an odometry system; a five-point sonar system; a Light Detection and Ranging (LiDAR) system; and a camera for reading fiducial marks [10], [14]–[16]. This platform was the target of previous projects, such as the work performed by Júnior [10], where a navigation and localization system was developed based on Adaptive Monte Carlo Localization (AMCL).

## 2.2 Big data visualization

Data visualization, especially of data with large volume, variety, and velocity of generation, the 3Vs that describe the main aspects of Big Data, has been taking prominence among researchers and industries because extracting helpful information from these data can bring competitive advantages, as well as the extraction of models and hidden patterns that assist in research [5].

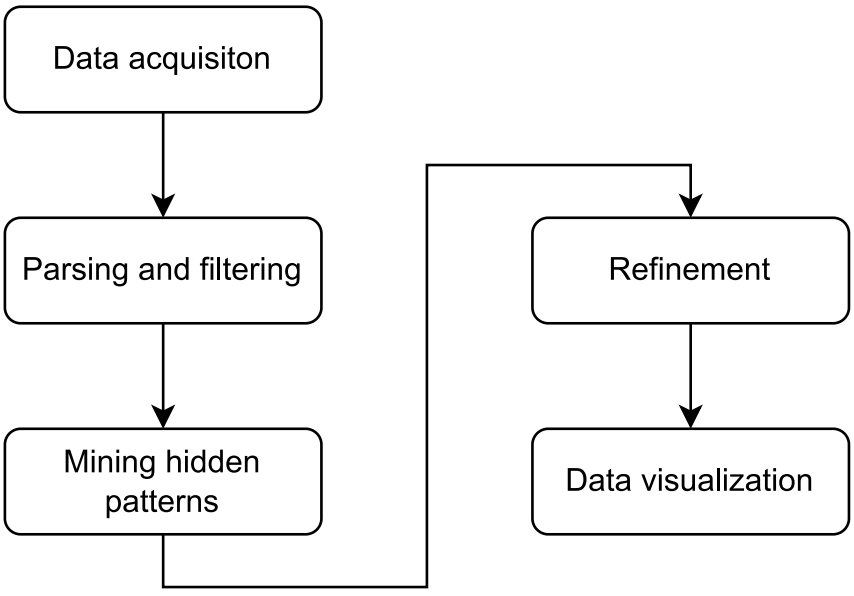



Figure 2.6: Big Data Visualization process (adapted from [5])

According to Chawla, Bamal, and Khatana [5], the process related to big data visualization goes through five steps, shown in figure 2.6, the acquisition of data, analysis and filtering, mining of hidden patterns, refinement and finally, the data visualization. Having as objective the representation of a complex data set through graphs and or images, where through these resources it is possible to extract information, detect and highlight patterns, generate insights and make decisions [5], [17].

### 2.2.1 Big data

The branch of technology related to big data has become one of the most critical trends in the information technology sector, as there is a high economic potential, especially for decision-making techniques for the economic and social sectors [4], [17]. Despite the many definitions attributed to this term, it is always focused on groups of data whose volume and or complexity are large enough that traditional analysis methods have difficulties being applied. Generally, when the term is used, it refers to a data set that traditionally cannot be processed.

As mentioned, not all data sets considered Big Data have a massive volume of information. However, they can also be endowed with a wide variety of information types, making it challenging to use traditional information analysis techniques. In the latter case, the result is that the information in question may be unstructured, incomplete, noisy, and erroneous, among other possibilities [4].

### 2.2.2 Data visualization

Data visualization is the graphical representation of a data set, using techniques and tools that make interpreting, understanding, and exploring these data more user-friendly. This more assertive communication brings advantages to those who interact with the database to understand how the data is connected, identifying patterns that would otherwise be hidden, especially in extensive data sets [4], [18].

The biggest challenge in this area today is the need to treat and extract information from a large volume and variety of data generated at high speed. The need to bring the visualization of these data in an interactive, scalable way which allows decision-making creates a need to develop new methods and computational techniques [5].

According to Mohammed, AlHabshy, and ElDahshan [4], some areas of difficulty stand out when it comes to big data visualization, as well as situations of opportunity to circumvent or mitigate these difficulties.

Difficulties:

- Perceptual Scalability: is linked to the human perception of data, as we find it difficult to efficiently extract useful data when the amount of information is too large;
- Real-time Scalability: the capability to bypass the memory and processing bottlenecks of large data sets to present information to consumers in real-time;
- Interactive Scalability: is the interactivity with the data set that helps to understand and obtain insights faster and more efficiently.

Opportunities:

- Data Reduction: reducing the size of big data by sampling and filtering to keep only data that is important and meaningful to be analyzed and exploited;
- Reducing Latency: among the possibilities are pre-computed data and parallelized data processing, which minimizes the processing demand, seeking to reduce this bottleneck in the process.

Data visualization is not only a challenge in the modern world. Several techniques and tools assist in the development and creation of visualizations. However, it is essential to know the types of data and what is expected to be obtained through the analysis to determine the most appropriate tool for the situation [4], [5], [18].

### 2.2.3 Data visualization techniques

To provide better visualization of the data that enables the user to use it better, it is necessary to consider which data will be presented and its dimension, among other characteristics. Because the representation of the data should be the simplest and easiest to understand so that the decision-maker can understand the data presented, their values, their relations, and patterns in the fastest way possible [4]. Where some of these techniques are:

I) Data tables:

In many cases, tables are not considered a type of data visualization. However, they represent a very effective technique for understanding some data because, depending on the situation, data representation in their raw form may be the most appropriate [4].

In a table, it is possible to represent numerous dimensions simultaneously. Although it is a valuable resource, it requires attention because communication occurs predominantly in a linguistic way, requiring the user to read and interpret this data [4].

II) Gauge charts and progress bar:

These visualizations are typically composed of a one-dimensional representation, where a needle or colour filling indicates the magnitude of the value represented. They are elements of easy understanding and visually satisfying due to the presence of colours and the simplicity of the visualization. Thus, they are widely used in dashboards and reports to indicate the values of Key Performance Indicator (KPI) [4], [19]. Figure D.5 presents an application of this type of visualization implemented in this work.

However, these methods must be avoided when the data that should be present has more than one dimension, or even for values whose behaviour in time is relevant, because this is only able to represent a single value, in addition, to occupying an ample space of visualization, it must be used sparingly [4].

III) Pictograms:

These representations are intended to demonstrate abstract concepts, using images, symbols and colours to represent the information [19]. The personal and cultural understanding of the symbol used can affect this visualization technique since it does not use textual language to demonstrate the information [20].

IV) Line and area charts:

Line graphs and area graphs are similar in their objectives and construction. Their construction uses continuous or discrete variables with a direct relationship, as in time series. By inserting the data in the plot plane and connecting the points using lines, it is possible to obtain a curve that indicates the behaviour and tendency of the variable analyzed. The difference between line graphs and area graphs is that in the area graphs, the area below the curve created by the variable has been filled in [4]. These plotting techniques are illustrated by the figures D.1 and D.3, which represent area charts and line charts, respectively.

Although line charts allow a practical analysis between two variables, they have limitations when several curves are analyzed simultaneously because the presence of multiple curves in the plan makes the comprehension of the behaviour of the variables more complex [4].

These limitations apply to unstratified and stratified area graphs, as the areas are occluded in the chart plotting. In the stratification strategies of the data, there is usually the stacking of areas, which may be traditional, just stacking the areas, or percentage, where the total area of the chart is constant, and the contribution of each category is their corresponding area [4].

These graphs are generally suitable for analyzing temporal variables or variables with some continuity.

V) Maps:

The maps are recommended visualizations for data that are correlated with geographic points. It is common to use maps in combination with other representations, such as bubbles and colour zones, to express the information [4]. Figure D.4 exemplifies this data visualisation technique.

This visualisation should be used with parsimony, as it takes up considerable visual space in applications. Therefore, it is an inclusion that should be adopted if the location of the data is critical to the data story. Like comparing a company's sales in various geographical locations [4].

VI) Graphs:

Graph models have origins in the study of topology and use elements called vertices and edges to describe connections. These models have several applications, ranging from interactions that occur in ecosystems to even the flow of control in computer programs [21]. An example of a graph representation is illustrated through the figure D.2.

VII) Sunburst charts:

This representation aims to describe in a radial format the data clustered hierarchically and associated quantitative values. The radial dimension of the segments is proportional to their contribution to the total value, as in a pie chart, thus being the radial version of the Icicle graph [22]. This technique of representation is demonstrated in figure D.6.

## 2.3 Databases

A database is necessary to store the data mentioned in section 2.2. Databases are an organized collection of information which nowadays is typically stored electronically in computer systems, being the most common storage format based on a series of tables composed of rows and columns, typically controlled by a Database Management System

(DBMS) [23], [24]. These databases can be classified into two main categories, which are based on the way the data is related and stored, these being:

- Relational: where data stored between database tables have a clear link between them, highly relational, i.e. data is connected through a predefined web of relationships [23], [24];
- Non-relational: in this category, the data is not required to have a direct relationship with each other or even predefined connections as in relational databases [24], [25].

### 2.3.1 Types of data

Different types of data can be stored in a database, and which kind of data, how they relate to each other and how it is organized are factors that define which classification can be assigned to the data. The data can be classified as structured, semi-structured and unstructured, the classification of these data influences the choice of the appropriate DBMS for its manipulation, and the source of these data eventually determines its characteristics [26], [27]. In simplified form, data types can be defined as follows:

- Structured: structured data have a predefined format, length and characteristics. They are general data that can be organized with a high degree of organization [27]. This data is usually managed by an Relational Database Management System (RDBMS) [26];
- Semi-structured: this type of data has no fixed schema, and its structure is implicit and often irregular, as there is no clear separation between the data and its schema, as is the case with structured data, typically managed by a Not only SQL (NoSQL) DBMS [26];
- Unstructured: unstructured data has no particular structure, typically incorporating images, documents, metadata and other data types, and it is the predominant data type in Big Data technologies [26], [27].

### 2.3.2 RDBMS

Being Structured Query Language (SQL) a programming language used in almost all relational databases to perform database manipulation [23]. The RDBMS is a DBMS that handles the databases classified as relational. This management system is suitable for applications that require complex queries with well-defined data [24].

Relational databases are the most efficient and flexible way to manage structured information [23]. However, the RDBMS has limitations such as:

- Scalability: because many of the databases do not have support for distributed processing and storage when the volume of data is high, it becomes necessary to use servers with large computational capacities [24], [25];
- Flexibility and complexity: due to the nature of structured data, the data must have its behaviour by the predefined format for its storage [24];
- Low read and write speed: when the volume of data grows, these databases significantly lose efficiency in reading and writing data [25].

### 2.3.3 NoSQL

NoSQL is a DBMS aimed at manipulating unstructured or semi-structured data, being a system used for applications involving big data and real-time data, among other applications. Khasawneh, AL-Sahlee, and Safia [24] and Han, E, Le, *et al.* [25] highlight some advantages presented by this system over RDBMS such as:

- High scalability: database expansions and modifications can be performed quickly and without interrupting database access;
- Flexibility: by not requiring a predefined scheme, it is capable of accepting different types of data, besides facilitating alterations;

- High reading and writing speed: because it presents a low latency for the queries carried out in the RDBMS especially when there is a high concurrent read and write request.

### 2.3.4 Temporal databases

The data may also present specific characteristics, as is the case of time series that give origin to temporal databases. Time is an essential aspect of the phenomenons and events in the real world because they are tied to a specific point in time, and the relationship between objects is built on time [28].

Temporal databases, by characteristic, have data whose value varies over time, so one of the dimensions of the database is time, and it is necessary to structure the database in a way that accommodates temporal logic, storing the real-time and/or collecting samples sequentially and periodically [28], [29].

Several RDBMS data models have been developed to meet the needs of time series [28]. However, as explored by Mehmood, Culmone, and Mostarda [30], some new technologies and methods put into doubt the usefulness of these established models, as NoSQL based models have more flexibility and versatility when compared to RDBMS models.

## 2.4 Tools review

During the elaborate on this work, several tools were used to make the development more dynamic, facilitating future improvements and replicating the work developed, prioritizing open-source tools or those available for free. The tools used in this work and presented in this section are Python, Django, JavaScript, React, MongoDB and the ROS.

### 2.4.1 Python and JavaScript

A programming language is a way of communicating and expressing algorithms utilizing symbols, syntax, and order of operations, inter alia, which should be arranged in

such a way as to convey to the machine the particular way in which the data should be treated [31], [32].

Among some of the many programming languages available, Python is a high-level and general-purpose programming language, created by Guido van Rossum in the late 1980s and having its first version published in the early 1990s. Its development took place at Centrum Wiskunde & Informatica (CWI). During the following years, Guido continued the development of the language at Corporation for National Research Initiatives (CNRI) and in 2001, together with his team of developers, formed the Python Software Foundation (PSF), a non-profit foundation aimed at holding the rights to the intellectual property related to Python [33]–[35]. It is a complete programming language used for many social and commercial purposes, is also suitable for scientific purposes, and can be used in several operating systems, besides being provided with good maintainability and development speed [35].

Derived from this language comes Django, a free and open-source web framework based on Python. It was created in 2003 by Adrian Holovaty and Simon Willison during their time as web developers at Lawrence Journal-World when they started developing code based on Python. After Simon's internship ended, Adrian and Jacob Kaplan-Moss continued the development of the framework. Since June 2008, the non-profit organization Django Software Foundation (DSF) that organizes and maintains Django [36], [37].

Nowadays, a popular programming language focusing on user interaction is JavaScript (JS). This language appeared in 1995, when Brendan Eich was recruited by Netscape Communications, and in only ten days, managed to create and implement a prototype with the first version of this language, Mocha. Mocha initially was supposed to execute small scripts with the web browser that would improve the sites but took proportions that could not have been predicted. Since its creation, the language has grown, gaining several new functionalities being one of the leading technologies for the World Wide Web (WWW), as it allows user interaction with the pages [38]–[40].

Today the JavaScript language is not limited to web applications but also mobile applications, desktops and servers, where it is used to run complex projects of high interactivity. There are two main reasons for its popularity: its dynamic language and permissive nature, which allows it to introduce and generate code during its execution and to follow the "no crash" philosophy where the execution errors do not generate apparent signals to the user. Today JavaScript is a brand owned by Oracle Corporation [38], [40].

In 2013, the company Facebook published an open-source library based on JavaScript to facilitate the creation of web interfaces. Since then, this library has gained attention from developers for allowing the creation of highly interactive environments [39], [41].

React, together with Django, have a development potential both for the user interface (front-end) and for the codes that run jointly with the database and the machine (back-end), which allows the creation of a complete, interactive data visualization tool. This composition enables the user to interact with the Big Data generated to support decision-making based on the analysis of the information provided [4], [42]. Several commercial solutions use these tools to create data visualization tools, such as Fusion Charts, which is a solution based on JavaScript, which uses a conjunction of language frameworks, including React and other languages, to create a more dynamic page, including Django, and the goal of this solution is to facilitate the process of dashboard development [4].

### 2.4.2 MongoDB

As explored in the 2.3 section, a database DBMS is recommended to help organize and communicate with the database. Among the many options available, MongoDB is a free and open source database software released in 2009 that handles databases classified as NoSQL, storing data as documents in a format called Binary JSON (BSON) [43]–[45].

The BSON format is similar to the JavaScript Object Notation (JSON) format. However, as verified in table 2.1, instead of encoding the data in a string 8-bit Unicode

Transformation Format (UTF-8), the data is encoded in binary form and, as such, becomes readable only for machines. The BSON format presents support for data types not supported when using the JSON format [44].

Table 2.1: Differences between JSON and BSON (adapted from [44])

| | JSON | BSON |
|---|---|---|
| Encoding | UTF-8 | Binary |
| Data support | String, boolean, number, array, object, null | String, boolean, number, array, null, date, binData |
| Readability | Human and machine | Machine |

It is a fast and flexible software with good scalability, offering support for languages with JavaScript (JS) and Python, besides having tools that allow easy transfer and copying of databases [43].

### 2.4.3 Robot Operating System

ROS emerges to assist in the deployment and development of robotic systems. Despite containing "operating system" in its name, ROS is not an operating system in the traditional sense of the term but a set of open-source frameworks that run on an operating system. The ROS aims to manage and allow the communication of the different processes that occur in the robot, besides helping in the implementation and scalability of the robotics project's complexity, acting as a bridge for various methods, as shown in figure 2.7 [46], [47].

The ROS was initiated in 2007 by PhD students Eric Beger and Keenan Wyrobek, working with Professor Kenneth Sailsbury at Stanford University [48]. Noting that the progress in the development of robotics was being delayed by the diversity and complexity of the field of robotics, they proposed the development of a basic system that would be the starting point for work in robotics. Over the years, several researchers joined the research, creating a community and a development ecosystem [47], [48].

As ROS is a very versatile platform and can be embedded in robotic manipulators, industrial robots, Unmanned Aerial vehicle (UAV) and AGV, among other various applications and robotic platforms. It is an open-source platform with a large ecosystem.

There are several areas of research, among which are task planning [49], cloud robotics [50] and security [51].

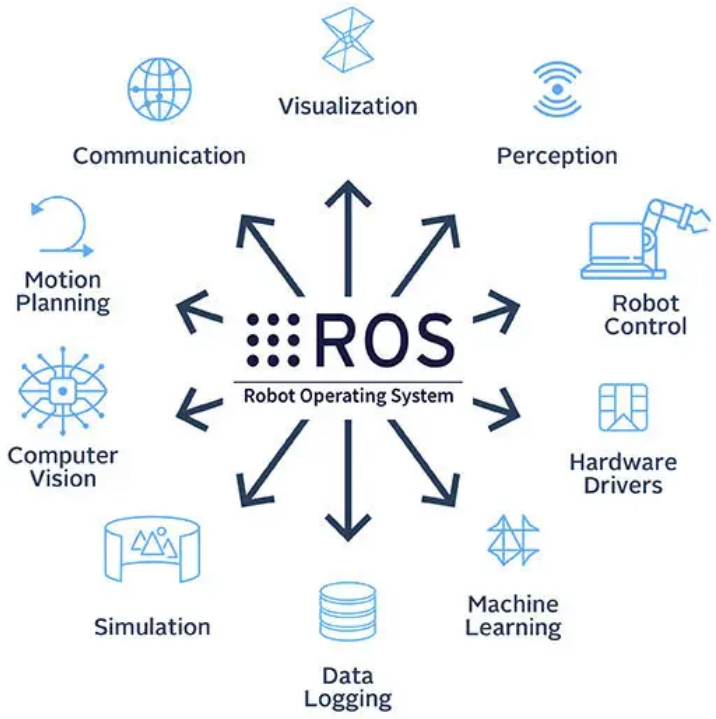



Figure 2.7: Robot Operating System [52]

# Chapter 3

# System Architecture

This chapter describes the proposed architecture for controlling and monitoring the equipment's health, exploring information flow from acquisition to decision-making. At the end of the chapter, the proposed structure for the Graphical User Interface (GUI) available to the platform manager is presented at last.

In section 2.1.2, it is possible to observe that the centralized or decentralized architectures better handle some characteristics of the environment and expected responses of the system. Being possible to generalize some aspects of these architectures as outlined in the table 3.1. Thus, it is possible to select the control architecture that better adapts to the system's needs.

Table 3.1: Comparison between centralized and decentralized architecture characteristics

| Characteristic | Centralised architecture | Decentralized architecture |
|---|---|---|
| Control of actions | Fully managed by the central unit [3], [7] | Managed by each member [3], [7] |
| Environment | Suitable for low dynamism environments [3], [7] | Suitable for highly dynamic environments [3], [7] |
| Access to information | Centralized (easy to get) [3] | Distributed (difficulty to obtain) [3], [12] |
| Complexity | Grows quickly with the number of members [3] | Reduced complexity [7] |
| Modifiability and extensibility | Limited by central unit [7] | Easily scalable and modifiable [3], [7] |

## 3.1 Proposed Architecture

For the selection of the most appropriate control architecture for this work, it is observed that the system has characteristics such as:

- Low management complexity: due to having only one AMR;
- Dynamic environment: caused by the high movement of people and changes in the environment;
- Platform autonomy: it is an objective of the system;
- It must be easily modifiable: since it is a development system in an academic environment.

Considering that the system does not fit satisfactorily in the advantages of centralized or decentralized architectures, the most appropriate choice becomes implementing a hybrid control architecture, intending to extract the benefits of centralized and decentralized control. For this purpose, three hierarchical levels are defined: Administrator, Remote Unit and AMR, as shown in figure 3.1. In this architecture, the hierarchical levels have some authority over the below levels. However, subordinate levels have the autonomy to make their own decisions and contest superior decisions if needed, as exemplified by Meissner, Ilsen, and Aurich [7].

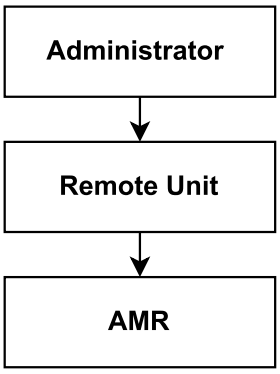



Figure 3.1: Hierarchical levels

Where the objective and function expected of each hierarchical level can be briefed as follows:

I) Administrator:

The Administrator is an operator, or operators, responsible for the system's management, maintenance and other physical or virtual aspects. Because this level comprises humans accountable for equipment management, the decisions made only by this hierarchical level should be followed compulsorily by the other levels.

This level aims to receive and analyze data from the lower levels through graphical tools that assist in decision-making to ensure the health of all systems and subsystems of the platform. Despite being the highest level of the hierarchy, it is not supposed that the Administrator acts as a controller or manager in the regular operation of the lower levels because they should have the management capacity and intelligence necessary for the autonomous operation of the system.

II) Remote Unit:

The remote unit aims to collect the information generated by the AMR and the tasks submitted by the Administrator. This level can guide actions by analyzing the current data and the data already provided by the AMR. It also has the function of generating the graphical interface used by the Administrator.

Even if this level can act as an orchestrator of the system, being able to strongly suggest the tasks that the AMR should perform, it is not the main objective of this element in the proposed architecture because the system should act mainly as a decentralized architecture system, being the AMR the decision-making agent. The remote unit must work on the data and its history, suggesting actions that help the desired characteristics in operation, for example, managing the battery consumption to maximize the availability time.

III) AMR:

Finally, the level denominated AMR represents the robotic platform. This level must operate autonomously and independently of the higher hierarchical levels. Because it

is assumed that the upper levels will only act to improve the platform's operation, not being a requirement for regular operation. This way, the robotic platform must be able to execute the five core tasks of an AGV, explored in section 2.1.1.

In other words, it is expected that for the correct performance of the proposed system, each level performs independent processes and can take decisions and communicate with the different levels, besides possessing specific functions and abilities, as exposed in the table 3.2.

Table 3.2: Functions and capacities expected from each hierarchical level

| Hierarchical level | Features and functions |
|---|---|
| Administrator | • Impose decisions at lower levels;<br>• Interpret the data;<br>• Compel actions. |
| Remote Unit | • Store and manage data from other hierarchical levels;<br>• Suggest tasks through data analysis;<br>• Generate a dashboard for the Administrator. |
| AMR | • Manage tasks;<br>• Determine their location in the environment;<br>• Capturing data from sensors and systems. |

## 3.2 Information flux

Considering the architecture proposed in the previous section and the expected functions of each hierarchical level, an information flow from data capture to decision-making is offered. This information flow is first presented in this section and, subsequently, the sub-processes. The diagram of this flow is represented in the figure 3.2, which is based on Business Process Model and Notation (BPMN) [53], to denote the proposed processes.

The process starts when the AMR is initialized and continues running until it is turned off, with the expected output of creating a task for the equipment to perform. When creating a task, the process will return to its beginning, creating a cycle that ends

only with the shutdown of the system. The diagram is divided into three lanes, each representing one of the proposed hierarchical levels.

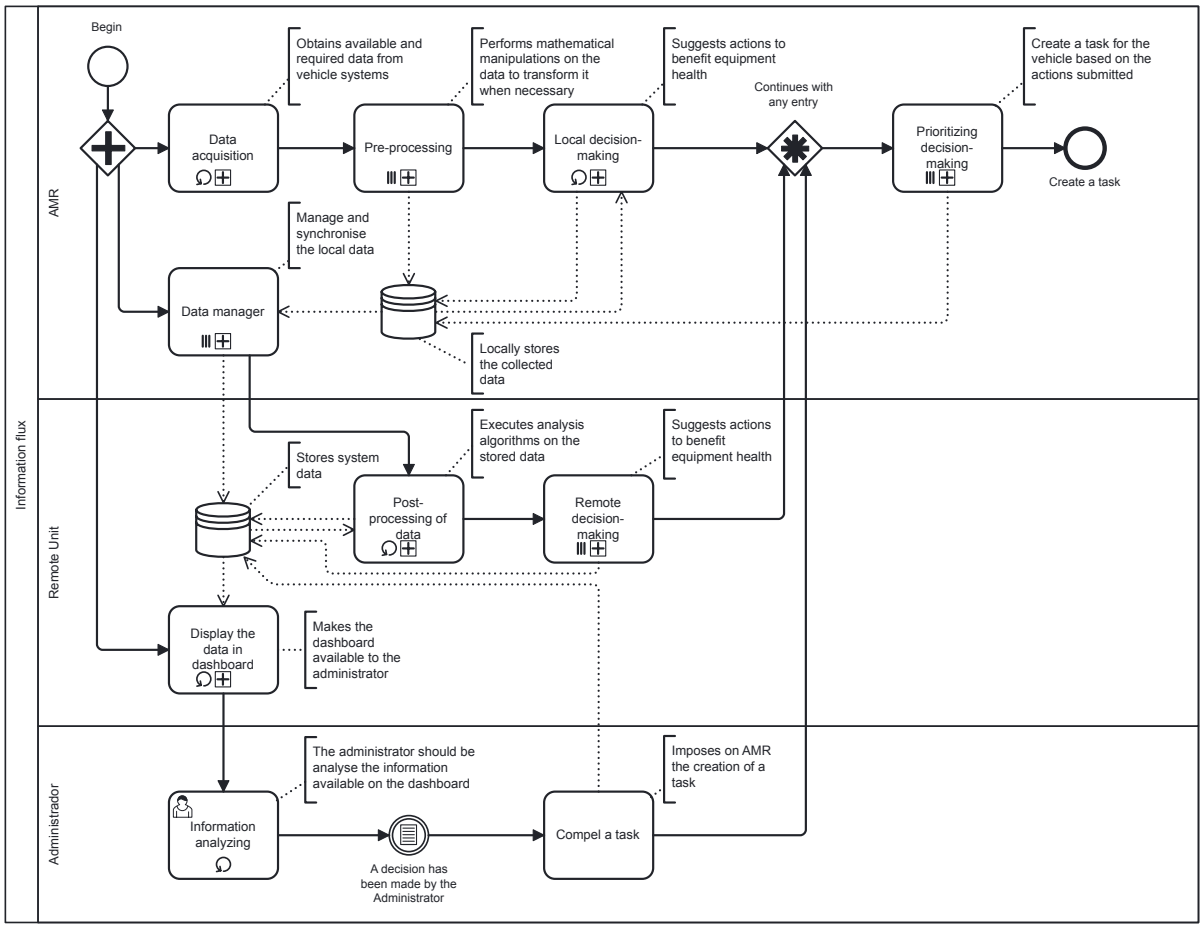



Figure 3.2: Information flux

### 3.2.1 Data acquisition and pre-processing

Data acquisition and Pre-processing are complementary sub-processes and are executed in sequence. Together they transform the information obtained through sensors and other methods into usable information for subsequent processes. It is expected to get information from the subsystems of the battery, position, and motor. These subsystems were raised as necessary for posterior analysis through an interview carried out with the support of the interview schedule in the appendix A.

The processes that involve data acquisition and pre-processing can be illustrated through the diagram in figure 3.3. In figure 3.3, through a periodic event, data acquisition is initiated, obtaining information in a parallel way from the subsystems already mentioned. Afterwards, these data are transformed, if necessary, adding more details to the process, and then these data are adapted to be stored in the local database.

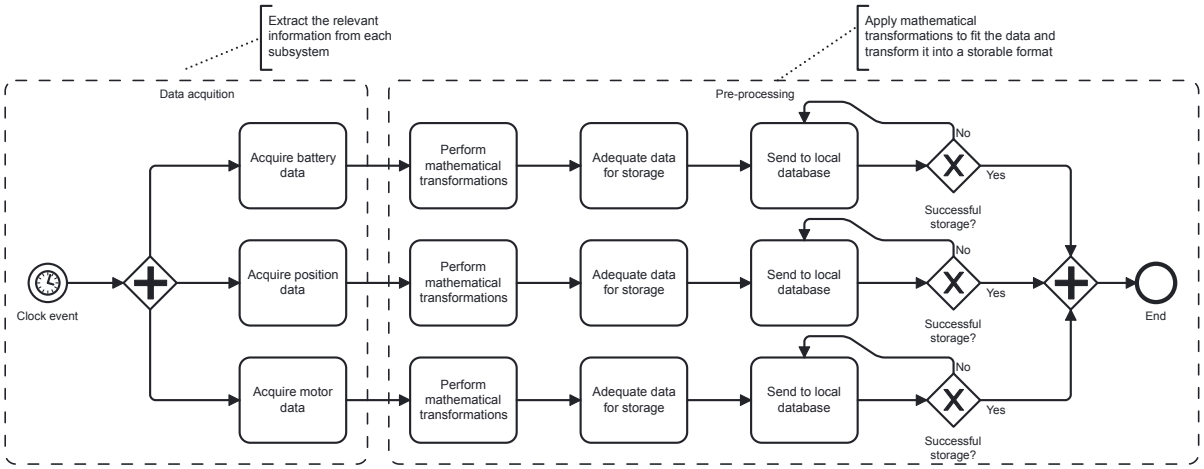



Figure 3.3: Data acquisition and pre-processing diagram

At the end of the execution of the Data acquisition and Pre-processing processes, it is expected to obtain the data indicated in the table 3.3, which will be stored in the local database and accessed through other processes.

Table 3.3: Data expected to be acquired from the AMR

| Subsystem | Expected acquired data | Description of data |
|---|---|---|
| Battery | Current | Current flowing from the battery |
| | Percent | Percentage of battery charge |
| | Power | Instantaneous power of the battery |
| | Status | If the battery is charging or discharging |
| | Temperature | Battery temperature |
| | Voltage | Voltage between battery terminals |

Table 3.3: Data expected to be acquired from the AMR (continued)

| Subsystem | Expected acquired data | Description of data |
|---|---|---|
| Motor | Current | Current flowing through the motor |
| | Power | Instantaneous power of the motor |
| | PWM | Information about the PWM to the motor |
| | Voltage | Voltage supplied to the motor |
| Position | Global position | Absolute position of the AMR |
| | Odometry | Wheel odometry of the AMR |
| | Velocity | AMR movement speed |
| | Velocity of wheels | Individual wheel speed of AMR |

### 3.2.2 Data manager

The process called Data manager is responsible for managing the local database, ensuring that the data is sent to the Remote Unit and that the designated storage space is respected.

Two parallel sub-processes are executed to perform these activities. One of these processes is intended to send the collected data. Simultaneously, the second process pretends to maintain the volume of data below a delimited value. Both processes are represented in the diagram of figure 3.4.

The data synchronization process starts whenever there is a connection to the Remote Unit, sending the most recent item yet to be synchronized to the Remote Unit's database. If the submitted item can be deleted, it is then deleted from the local database, helping to keep its volume within the designated volume.

The storage management process starts when the available space for the local database is surpassed so that it deletes all synchronized files that can be deleted. If the storage capacity is still exceeded, a random item should be selected for deletion, prioritizing deletable items. This part of the process acts outside the regular operation of the platform.

However, o the storage management process is proposed as a preventive method for creating significant continuous gaps in the data set. For the eventual subsequent data analysis, it is possible to fill the gaps using algorithms. However, as explained by Li, Wang, Fang, *et al.* [54], Wei and Tang [55], and Li, Wu, Li, *et al.* [56], the size of these gaps is a factor that directly influences the accuracy of these algorithms.

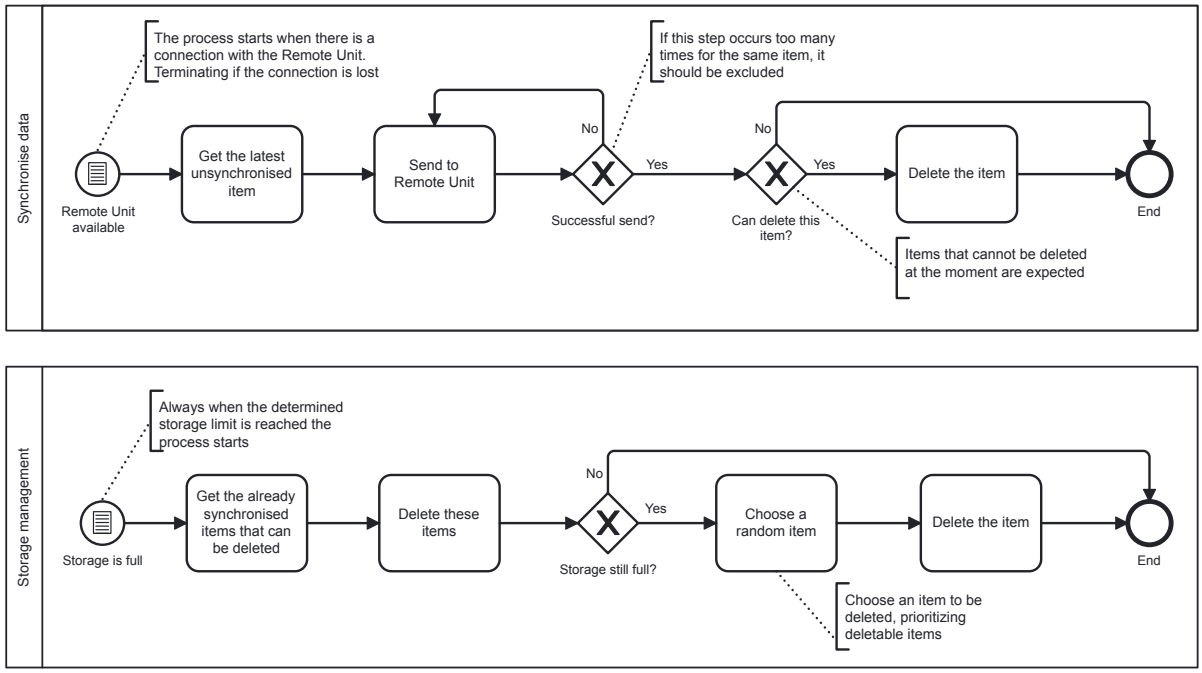



Figure 3.4: Data manager diagram

### 3.2.3 Local decision-making

The Local decision-making process ensures that the AMR operates safely, avoiding damaging or harmful situations to the equipment, such as running at over temperature. This process is illustrated in figure 3.5.

The process begins periodically, accessing the local database and, obtaining the necessary information for the analysis, then verifying if the parameters are within specified limits. If these parameters present any abnormality, an action is created to relocate the values to the safety zone. This action is sent to the action queue, where the prioritization process creates a task for the AMR to execute.

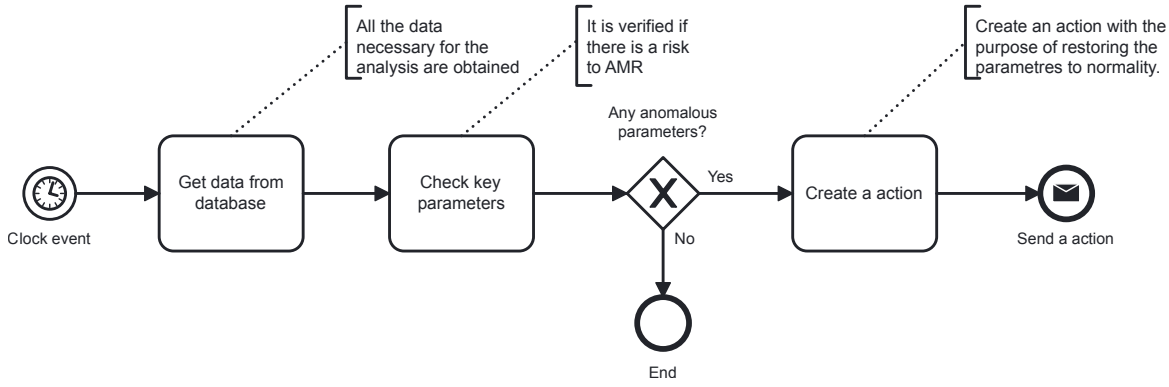



Figure 3.5: Local decision making diagram

### 3.2.4 Prioritizing decision-making

This process defines which action will be turned into a task for the AMR to perform. As shown in the figure 3.6, two sub-processes are running in parallel. One is responsible for obtaining the actions and transforming them into tasks, while the other is determining which tasks must be sent to the equipment to perform.

The sub-process of creating a new task is initiated each time an action is sent to the AMR through the decision-making processes at the different hierarchical levels, as verified in figure 3.2. This way, an action can be appended to the action queue of the AMR.

The management of the tasks queue begins each time a task is inserted into the queue, ensuring that the priority of actions is respected, allowing only equivalent tasks with the same priority in the queue. Equivalent tasks are considered to be those of the same type.

The following situation is an example of equivalent tasks: if the AMR is performing a route to the waypoint 27, but the Remote Unit determines that it needs to return to the base. The task of "go to the base" is added to the task queue, interrupting the task "go to waypoint 27" execution and starting the route to return to the base. This behaviour occurred because the tasks are equivalent, "go to" type, but the task added by the Remote Unit had a higher priority than the active task.

With these activities, the system is expected to receive and deliberate on the actions sent by the different hierarchical levels.

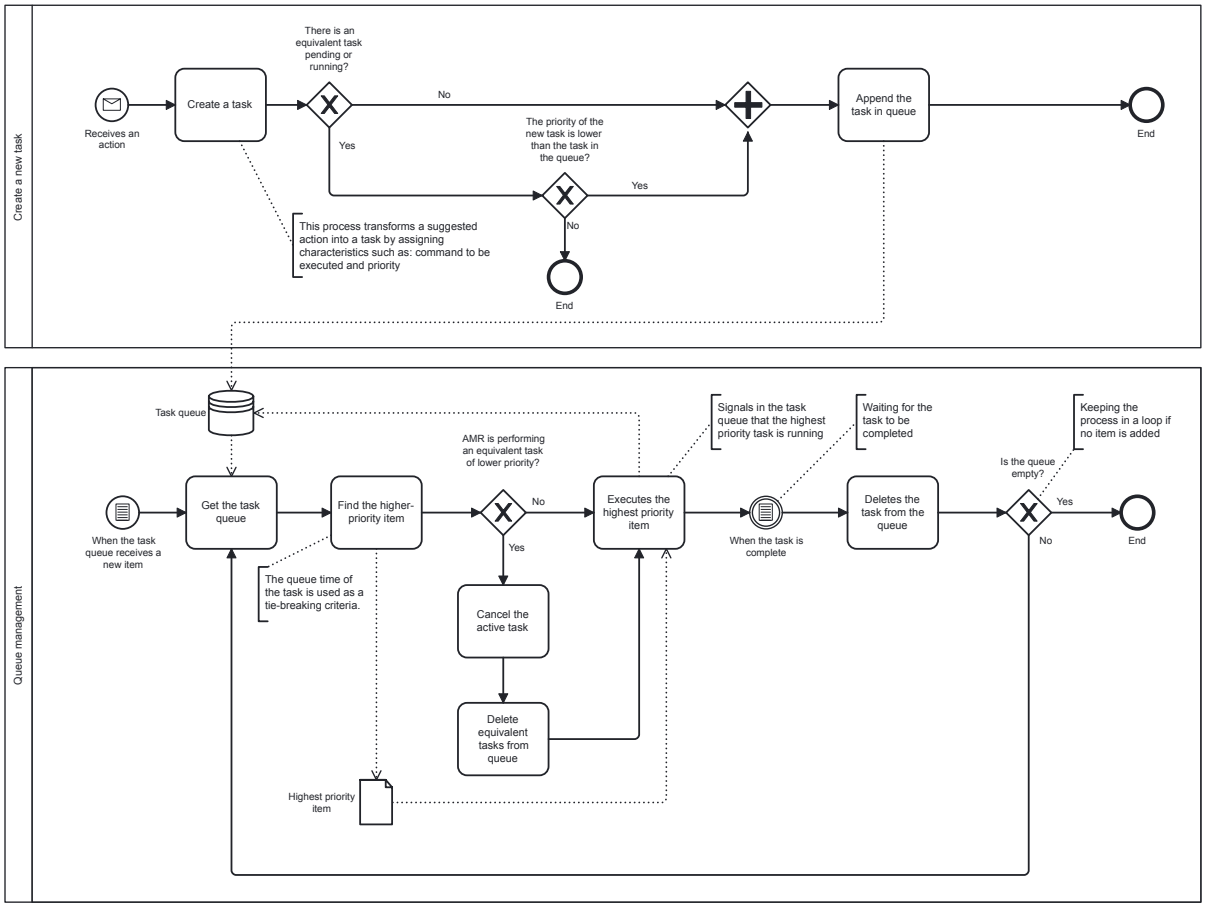



Figure 3.6: Prioritizing decision-making diagram

### 3.2.5 Post-processing of data and remote decision-making

The post-processing of data and remote decision-making are processed sequentially. At the end of the execution of these processes, the expected outcome is to obtain a database enrichment through the post-processing of this data, access to the already transformed data for the production of the visual resources through the dashboard and the creation of tasks that improve the system's performance. The tasks performed by these processes are mapped in figure 3.7.

The Post-processing process starts periodically, and two tasks are executed in parallel. One task implements algorithms for processing data and extracting information, which is a complex task with high computational costs. The other task aims to run queries on the database and provide the results for the dashboard to access.

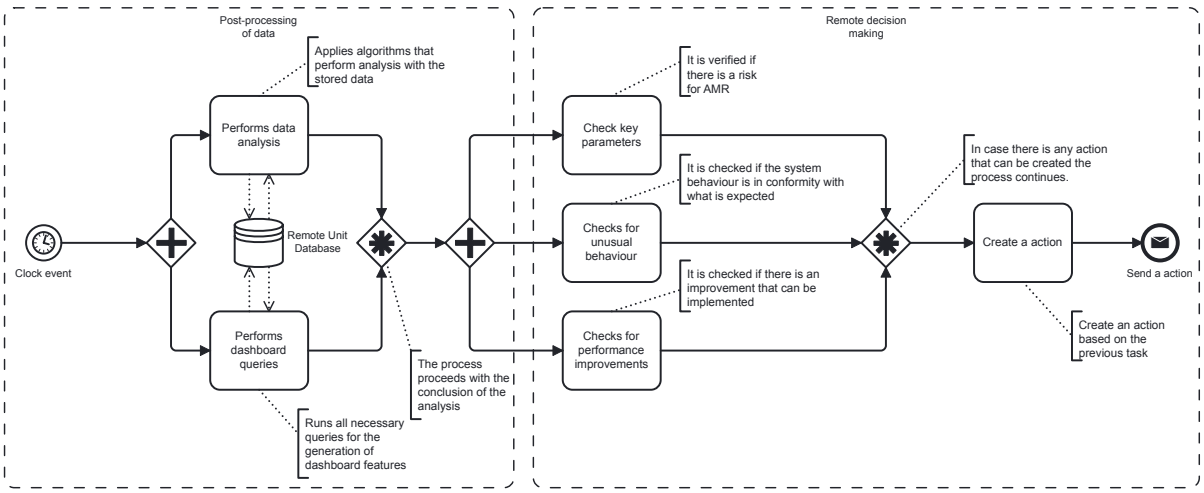



Figure 3.7: Post-processing of data and remote decision making diagram

Remote decision-making, similar to Local decision-making, seeks to ensure the safe operation of the platform from the analysis of crucial parameters. Additionally, it runs two other processes in parallel. One process checks for strange behaviours in the collected data, while the other searches for ways to improve the performance of the AMR. The expected output of these processes is an action that must be sent to the AMR.

### 3.2.6 Display information, information analyzing and compel a task

The processes proposed in this section represent the highest points in the hierarchical structure of the system. As such, they should have minimal impact on the platform's operation and require infrequent interventions. Figure 3.8 synthesizes these processes.

The Remote Unit carries out the task of presenting the information to the Administrator. This process occurs periodically, collecting processed information and updating the graphical features available in the GUI.

In sequence, the Administrator should analyze the data provided to identify any abnormal behaviour or information that could represent a risk to the health of the equipment. When it is necessary to interfere in the platform's operation, the Administrator can send a task to the AMR via the GUI, which should perform compulsorily.

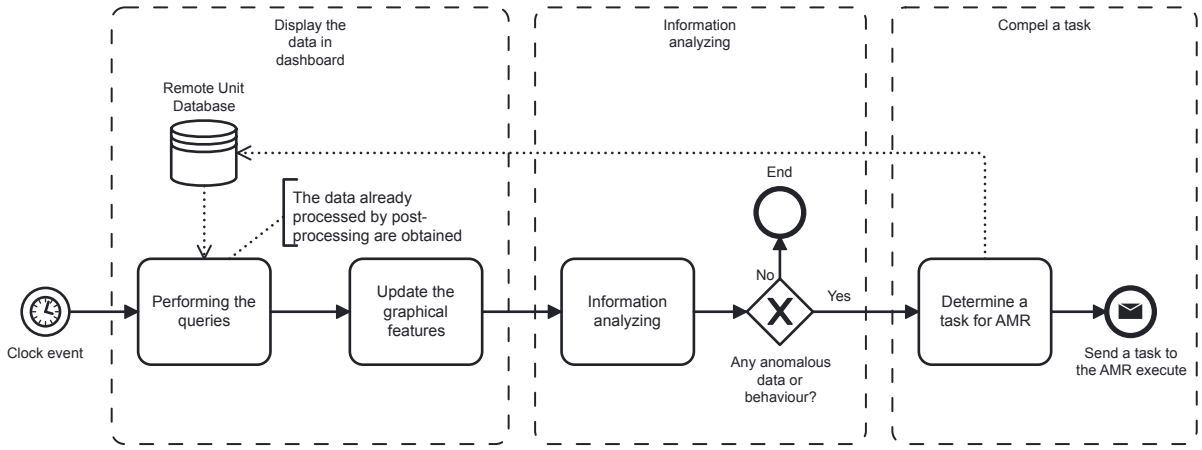



Figure 3.8: Dashboard information diagram

## 3.3 Dashboard

The GUI dashboard presents graphical and interactive demonstrations to the user, providing current and historical platform status data. The dashboard allows the Administrator to analyze and identify possible patterns in the database, providing tools to make informed decisions about the platform's health.

In other words, it is expected that through this application, the concepts verified in section 2.2.2 principally act on perceptual scalability [4], which is the extraction of useful information from large data sets. This tool is expected to be generated by the Remote Unit and made available by a web application accessible to the Administrator.

The study presented by Bach, Freeman, Abdul-Rahman, *et al.* [19] demonstrated a correlation between the elements of panel composition and their genres. To define the characteristics of dashboard composition, the method Analytic Hierarchy Process (AHP) Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) [57] is applied to identify the most suitable dashboard genres for the project's requirements. Based on the work previously mentioned, are considered six alternatives and three criteria, with thirteen subcriteria for the construction of the criteria and alternatives tree, to apply the AHP method, being this tree presented in the figure 3.9.

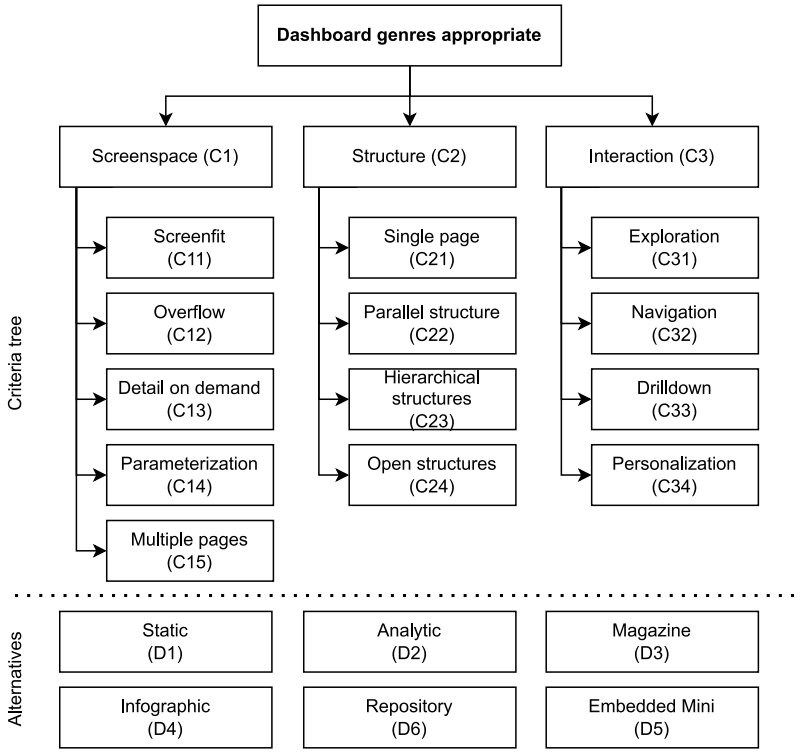



Figure 3.9: AHP criteria tree for dashboard genres

For the application of the method, the matrices presented in appendix C.2 were built, thus obtaining the closest alternative to the ideal solution, the Analytic dashboard. The Analytic dashboard genre typically uses various visualization techniques and tables to exploit an extensive and detailed data set. It is often interactive and allows for navigation between multiple pages. However, it is unusual to have page overflows, as page scrolling makes it difficult to compare data [19]. Therefore, the application is expected to present the following features available on web pages:

- Time series graphics;
- Current status of platform systems;
- Access to the system information log;
- Access to the visualization of the raw data generated;
- Capacity to send tasks to the platform.

Based on nowadays available web applications, as illustrated in the appendix B, the general page design is proposed as shown in figure 3.10, where there is a central area where the main interactions with the user are performed, a sidebar that allows navigation between pages and tools available, along with a header bar where relevant information and quick access to the user.

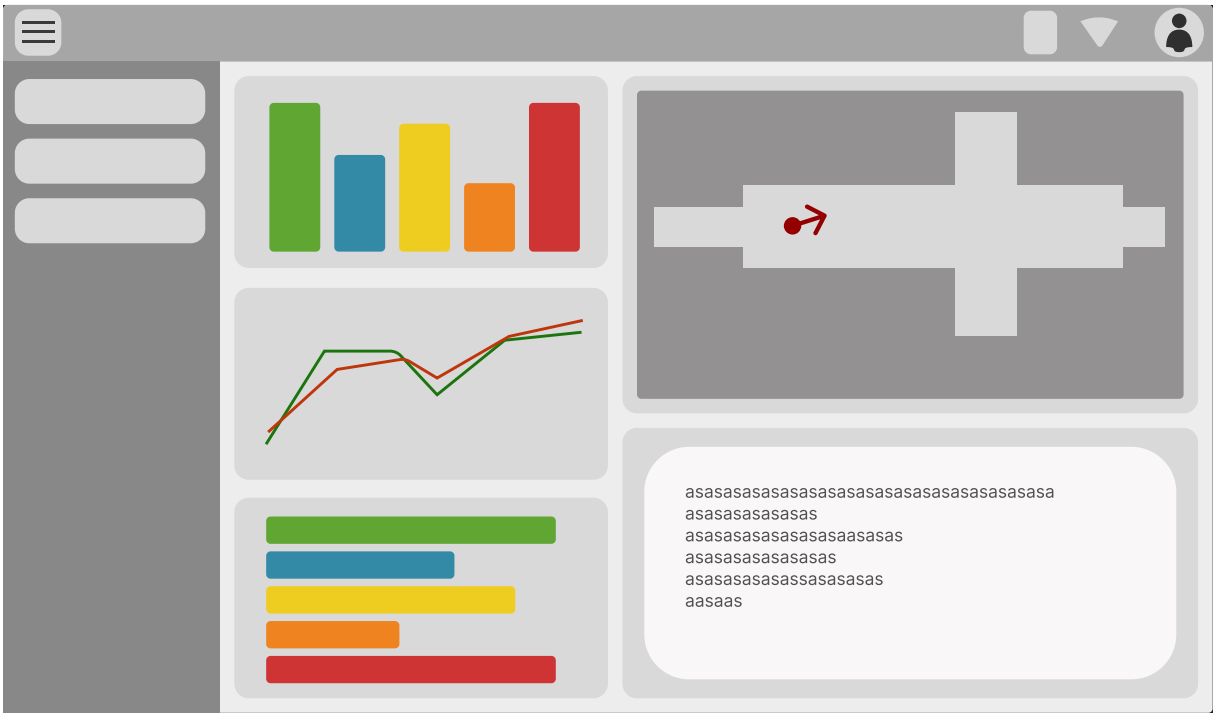



Figure 3.10: First perception of dashboard

The objective is to retain the focus on the information of interest in the central area and always visible essential details about the platform in the header. The sidebar allows more straightforward navigation between the different topics available in the application. It is similar to the well-established design for friendly interaction between the Administrator and the dashboard.

# Chapter 4

# Development

This chapter intends to present the development of the system proposed in chapter 3, describing the decisions assumed during the development of this project based on the content discussed in chapter 2.

The chapter begins with the selection of the programming languages and the choice of the DBMS, these choices form the foundation of the project's development. After selecting the programming languages and DBMS, the next step was to develop the flow of information presented in section 3.2, identifying and resolving any issues with the flow of information, and implementing necessary adjustments to ensure smooth data transfer. Finally, is described the development of the GUI proposed in section 3.3, where the Administrator can interact with data, for the management of the platform.

## 4.1 Tools used

For the elaboration of the work, it was necessary to define the tools used. This section presents the choice of the programming languages, the DBMS used, and the interest in its options. The tools defined in this section were previously explored in section 2.4.

### 4.1.1 Programming languages

The choice of programming languages is an essential point at the beginning of the project because it influences its maintainability and scalability, as verified in section 2.4. In this project, it is possible to separate it into two work fronts, one related to the internal process of the system. At the same time, the other is related to the interaction with the Administrator. This is a standard division in the world of technology, especially with web development, where the frontend is responsible for interaction, visualization and other forms of contact with the user, usually composed of HTML, CSS and JavaScript, and the backend where the activities of processing and data management occur, traditionally composed of languages like Ruby, Python, C++ among others [58].

The backend should be based on a single programming language to simplify the learning curve required for the development of this project. For the definition of the most appropriate programming language on which the backend is based, it was established the characteristics that this language must have, and these characteristics are listed below:

- Be compatible with the ROS;
- Be Suitable for backend development;
- Have a good development speed;
- Have good maintainability;
- Have an active community.

Delimiting a single programming language for all the processes to be executed by the backend, and that offers support to its community. Python has been determined as the most appropriate language for the backend, together with the Django framework, being indicated for creating tools that work with big data visualization [4].

The JavaScript (JS) programming language was adopted for frontend development, using the React development library because it is an open-source library with several component libraries and tools for rapid development. When used with the Django framework,

the React library has great development power, already selected for backend development [4].

### 4.1.2 Databases

Through the presented in the section 3.2, it is possible to synthesize a list of questions that help in the definition of the DBMS most suitable for the project. Therefore the three questions below are formulated:

1. How is this data generated?
2. What types of data will be stored?
3. How do they relate?

Through the analysis of section 3.2, it is possible to identify two main sources of information for databases: data acquisition/processing and decision-making. In the first source, the data are obtained periodically and sequentially and can be defined as temporal series [29]. They have a well-defined scheme as exposed in the table 3.3 but do not have direct relationships between them. They can be classified as semi-structured data [26]. With this information, it is already possible to assign that the most suitable DBMS is the NoSQL, which can manipulate semi-structured and unstructured data and also recommended for time series, besides being flexible to the road several types of data [24], [25].

For the selection of the DBMS software, the following characteristics were defined:

- Be open-source;
- Be classified as NoSQL;
- Have Python compatibility;
- Have compatibility with JavaScript;

- Be suitable for time series.

MongoDB has been chosen because it meets the characteristics defined above as necessary, as explored in the section 2.4.2, as well as having already been successfully implemented in time series database models for sensor data acquisition [30].

## 4.2 Implementation in the AMR

This section aims to explore the development and implementation of the information flow proposed along the section 3.2, focusing on the processes and sub-processes involving the AMR. As the platform uses the ROS Noetic [10], all the implementation of the processes presented in this section uses this tool for its execution and communication with the other processes. Therefore, in pseudocode format, it implements the diagrams proposed in chapter 3.

A limitation presented by the platform used by the CeDRI is the armhf architecture (32-bits) on which the microcomputers are based [16]. This peculiarity presented in the object of study makes it impossible to install MongoDB in the platform using traditional methods because it is only supported in 64-bits architecture [59]. Therefore, there would be a change in the information flow since the data would not be accessible in the expected way since the chosen DBMS is not supported by the system. Thus, a new information flow is designed to the AMR, shown in figure 4.1.

In this process, the data generated by the platform are not stored using a DBMS but saved in files that the Data manager process can read and send to an external database. Thus, the local database becomes a buffer for sending these files. With this solution, there is a need to adapt the local decision-making process. Since the decision-making process does not has access to the stored data, it must obtain the necessary data through pre-processing and keep the data required for analysis internally in the process.

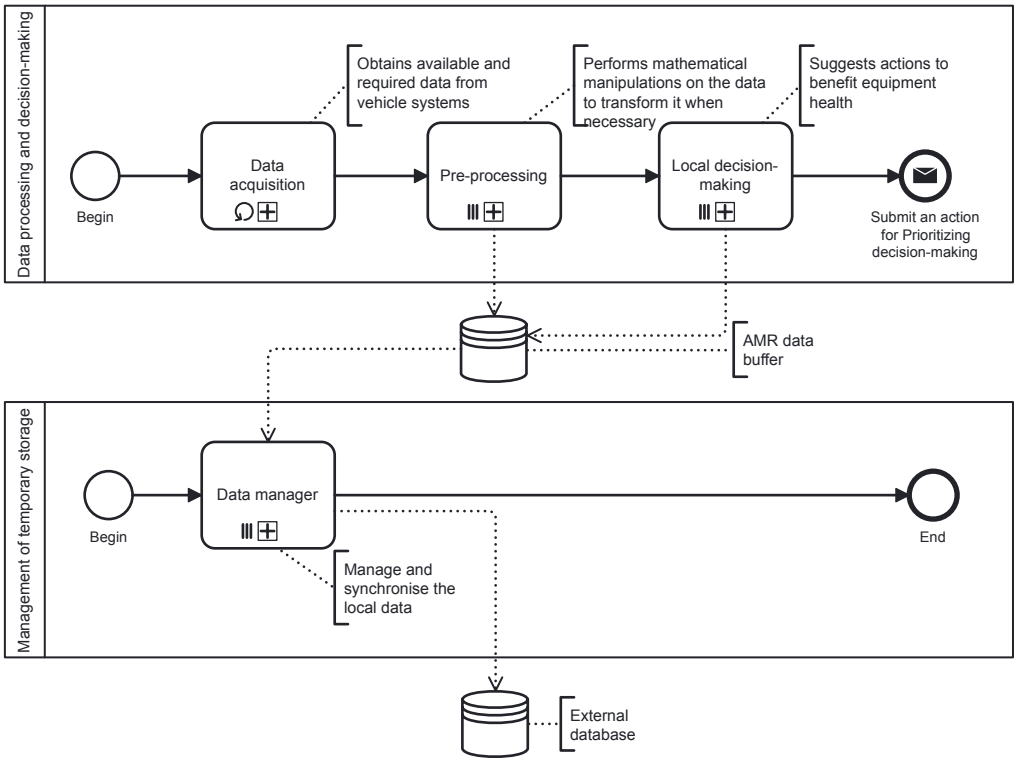



Figure 4.1: New AMR information flux

### 4.2.1 Data acquisition and pre-processing

For the implementation of the data acquisition and pre-processing processes explored in section 3.2.1, the development was divided into three parts, identification of the topics of interest, development of the script for reading and processing these topics and script for storing the information. This way, the development process can be analyzed below:

I) Identification of the topics of interest:

To obtain the minimum information required according to the table 3.3, the topics published by the AMR during its operation were analyzed. This way, it is possible to verify that for the platform on which this project is based, this information is published through the topics in the table 4.1, also showing the type of message published by this topic.

Table 4.1: Topics and types of messages of each subsystem

| Subsystem | Acquired data | ROS topic | Message type |
|---|---|---|---|
| Battery | Current | \battery_state | sensor_msgs/BatteryState |
| | Percent | \battery_state | sensor_msgs/BatteryState |
| Battery | Power[1] | - | - |
| | Status | \battery_state | sensor_msgs/BatteryState |
| | Temperature | \battery_state | sensor_msgs/BatteryState |
| | Voltage | \battery_state | sensor_msgs/BatteryState |
| Motor | Current | \motor_state | ubiquity_motor/MotorState |
| | Power[1] | - | - |
| | PWM | \motor_state | ubiquity_motor/MotorState |
| | Voltage[1] | - | - |
| Position | Global position | \amcl_pose | geometry_msgs/ PoseWithCovarianceStamped |

[1]Value not published by the topic. Therefore the processing of the data is required.

Table 4.1: Topics and types of messages of each subsystem (continued)

| Subsystem | Acquired data | ROS topic | Message type |
|---|---|---|---|
| Position | Odometry | \odom | nav_msgs/Odometry |
| | Velocity | \amcl_pose | geometry_msgs/ PoseWithCovarianceStamped |
| | Velocity of wheels | \odom | nav_msgs/Odometry |

II) Development of the storage script:

A function was developed to store the information obtained through data acquisition in the modes proposed at the beginning of this section. This function can create a document containing the data that should be stored, besides all the other necessary instructions, so the file is correctly stored in the external database. Files created by this function must also have unique names, and there must not be duplicate names between any files that have already been created. The algorithm 1 is then proposed, which meets the established requirements.

**Algorithm 1** Create a file

**Require:** *Path* as BSON
**Require:** *Data* as BSON
**Require:** *extension* ← '.cjson'
1: **function** CREATEFILE(*Path*, *Data*)
2: **if** *Path* **is not** complete **then**
**return False**
3: **end if**
4: *data* ← Encode( { 'Path': Path, 'content': Data })
5: $fileName$ ← Full time stamp + *extension*
6: $file$ ← Open a file with $fileName$
7: $file.data$ ← Write *data*
8: $file$ ← Save and close
**return True**
9: **end function**

It creates a unique file name using as a base the timestamp containing year, month, day, hour, minute, second and milliseconds, as in the following example, 2023020712

3239_918832.cjson. The file is stored in binary format data. It includes the information required to correctly send the data to the external database implemented using MongoDB, being this information, the database and collection [60].

III) Development of the sampling script:

Analyzing the information published by the topics, it is observed that there is information beyond the minimum required, so it becomes possible to take at least two approaches to obtain the minimum or complete information. These approaches influence the size of each stored document and the number of documents that can be stored in a given storage space. Both options can be described according to the algorithm 2, with the suppression of line 3 in the case of obtaining all the published data, and this algorithm must be implemented for each topic to which it is desired to get information.

**Algorithm 2** Get topic information

**Require:** $Path$ as BSON
**Require:** $Topic$ as string
**Require:** $MessageType$ as a specific message type
1: **while** *ROS.core* **is** *running* **do**
2: $raw \leftarrow$ TopicMessage($Topic$, $MessageType$)
3: $data \leftarrow$ GetInformationRequired($raw$)
4: $data \leftarrow$ ProcessesInformation($data$)
5: $dataBSON \leftarrow$ EncodeBSON($data$)
6: createFile($Path$, $dataBSON$)
7: **end while**

In addition to the topics listed in table 4.1, four nodes have been developed to obtain more information about the operation and status of the platform, thus providing the Administrator and the Remote Unit with a larger scope of data to assist in decision-making. These topics were named as \getSignal, \getROSnodes, \getHTOP and \robot_diagnostics. Their average message size, obtain and generation rate is exposed in table 4.2. Where the \robot_diagnostics and \getROSnodes nodes only store information when there is any change in the acquired information.

To evaluate which capture method (complete or partial) should be implemented for data acquisition, samples of each topic raised in table 4.1 were collected to identify the

average message size. In figure 4.2, it is possible to observe the difference in the stored size between the two methods. On the horizontal axis are present the subsystems, as presented in table 4.1, in addition to the diagnosis subsystem, where are present the items of table 4.2. On the vertical axis is the data stored size in bytes. It was considered a capture rate of 1 Hz for the Battery, Motor and Position subsystems.

Table 4.2: Platform diagnostic topics

| Node | Average size (B) | Sample rate (Hz) |
|---|---|---|
| \robot_diagnostics | 387 | By changes |
| \getHTOP | 51650 | 0,0167 |
| \getROSnodes | 33280 | By changes |
| \getSignal | 63 | 1 |

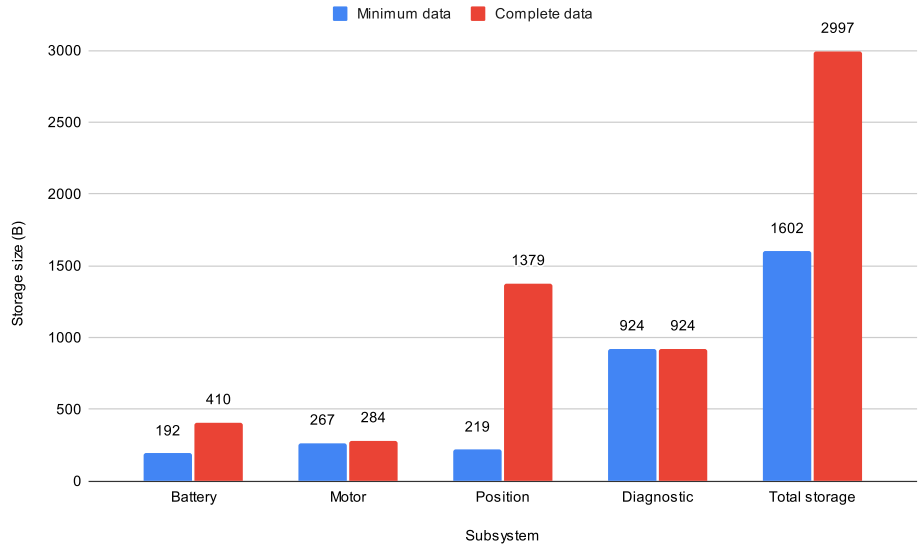



Figure 4.2: Comparison of stored size obtaining complete or minimum data

In this way, it is observed that when selecting the desired information, the space needed to store the data is approximately half that required to keep all the information provided by the topics.

It is assumed that there is 1 GiB (1073741824 B) of dedicated storage available for the buffer. The time required to fill the storage is about 186 h to get the minimum data or 100

h to get the complete data. Since it is assumed that the AMR does not remain unavailable to send information to the external database for long periods, the period of 100 h does not impact the operation of the equipment. Therefore, the option to obtain the complete data is considered the most suitable, allowing future analysis of the collected information that was not deemed necessary in the preliminary analysis identified in section 3.2.1.

With this decision, it is possible to modify the algorithm to generalize it. This way, it was decided to create a class in which subscribers are created for all topics in a list. By inserting certain information in this list, the script can interpret it, create the subscriber in ROS for each item, and store the information published by these topics in files containing the necessary data for the correct forwarding of the information to the remote database.

This list was based on the Python language, where the data is stored in a list of dictionaries that has the exposed structure through the table 4.3, containing the name of the node that you want to listen, the type of message published by this node, the capture rate, the callback function and the necessary data for storage. The data capture script is generalized to any robotic system based on ROS.

Table 4.3: Dictionary structure for Data Acquisition and Pre-processing algorithms

| Dictionary structure of the sample list | | |
|---|---|---|
| Key | Data type | Description |
| node | string | Topic name to request information |
| msg | ROS message | Type of message that is published |
| rate | float | Sample rate for this message |
| callback | function | Function that executes data alterations |
| dataPath | dictionary | Dictionary with information concerning the storage of data |
| Dictionary structure of dataPath | | |
| Key | Data type | Description |
| dataBase | string | Database name |
| collection | string | Collection name in the database |

### 4.2.2 Data manager

Due to the limitations provided by the microcontroller's architecture, it is necessary to modify the process proposed by figure 3.4. This way, the redefined process is represented through the diagram in figure 4.3. As the diagram exposed in figure 4.1, there is no need to maintain the documents stored for its access to the process of Local decision-making.

The development of this process is divided into two stages. The first is dedicated to creating the script responsible for sending the information to the external database, and the second is the development of the script that performs the management of storage space.

I) Synchronize data sub-process

Based on the model presented in figure 4.3, the algorithm 3 is elaborated, which aims to synchronize the stored data. This process consists of waiting for documents to be sent, creating a sending queue, which prioritizes the most recent documents, and sending them whenever there is a connection with the Remote Unit.

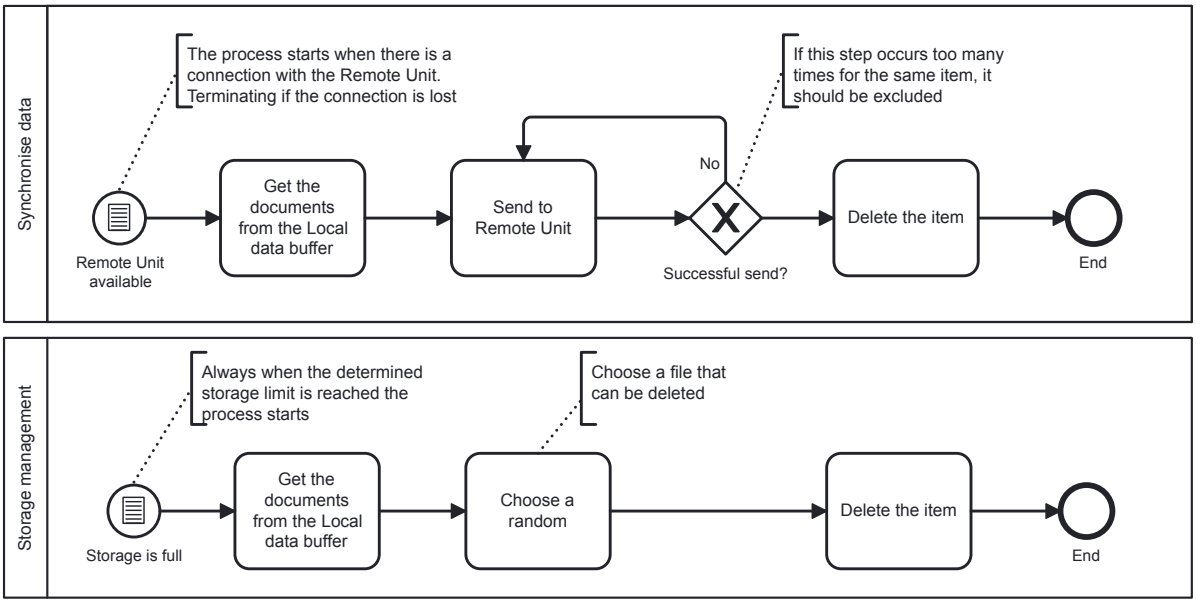



Figure 4.3: New Data manager flux

**Algorithm 3** Synchronize data sub-process

**Require:** Documents in the buffer were created using algorithm 1

```
1: while ROS.core is running and ExternalDB is available do
2:     DocumentsList ← All documents in buffer
3:     DocumentsList ← sort(DocumentsList by Creation date)
4:     for document in DocumentsList do
5:         data ← document.content
6:         path ← document.Path
7:         response ← SendToExternalDB(data, path)
8:         if response is (Success or Duplicate or FormatError) then
9:             Delete(document)
10:        else if response is SendingError then
11:            break
12:        end if
13:        next
14:    end for
15: end while
```

II) Storage management sub-process

For the development of the script that executes the storage management sub-process, the algorithm 4 is applied. That observes the space occupied by the buffer, comparing it with the limit space defined by the System Administrator. In case of extrapolation of this space, a random document is selected and deleted. In this way, it is sought to cause gaps with the shortest length, as discussed in section 3.2.2.

**Algorithm 4** Storage management sub-process

**Require:** *StorageLimit* as integer

```
1: while ROS.core is running do
2:     ActualSize ← Actual buffer size
3:     if ActualSize ≥ StorageLimit then
4:         DocumentsList ← All documents in buffer
5:         document ← RandomDocument(DocumentsList)
6:         Delete(document)
7:     end if
8: end while
```

This is a script which is not supposed to interact with the buffer during the regular operation of the equipment, intending to act in situations where the communication failure

between AMR and the Remote Unit exceeds the available storage time verified in section 4.2.1.

### 4.2.3 Local decision-making

As a result of the changes made in the information flow at the level of the AMR as shown in figure 4.1, it becomes necessary to adjust the proposed process in section 3.2.3.

Therefore, the process is applied as shown in figure 4.4, with information from the previous process, pre-processing. In this case, it is used in the callback function presented in table 4.3 to create a publisher that publishes this information, to be captured by a subscriber in the local decision-making process.

The necessary data is acquired to perform the decision-making and action-creation process. The transmission of the action to the Prioritizing decision-making process occurs via a publisher that sends the information to the subsequent process.

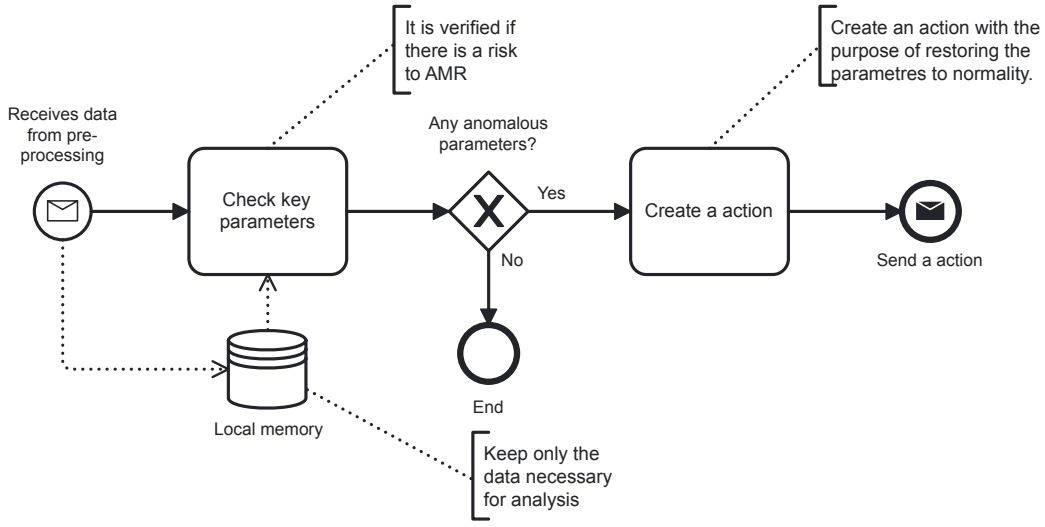



Figure 4.4: New Local decision-making flux

### 4.2.4 Prioritizing decision-making

The development of the Prioritization decision-making process is divided into two parts. Each part is dedicated to developing each sub-process presented in the figure 3.6.

This development is described below:

I) Create a new task:

For creating a task, the action submitted must include, as a dictionary, the information in the table 4.4. Thus, it is possible to determine the priority and source besides the required information to create a task.

Table 4.4: Dictionary structure to create a new task

| Key | Data type | Description |
|---|---|---|
| priority | float | The execution priority of this action [0 to 100]. |
| source | integer | Indicates the source of the action [0 to 2] |
| topic | string | The topic where the task is to be created |
| msg | string | type of message that is published |
| status | integer | Action status, being sent as 0 |
| command | string | Command that should be sent |

To add a new task to the task queue, the algorithm 5 is executed, performing the verifications presented in figure 3.6. It is then checked for the presence of the minimum information required and if similar actions are pending or at execution by the AMR.

**Algorithm 5** Create a task

**Require:** $TaskQueue$ as list
**Require:** $IncomingMessage$ as dictionary
**Require:** $CanAppend \leftarrow$ **True**
1: **function** $AppendToTaskQueue(TaskQueue, IncomingMessage)$
2: $CanAppend \leftarrow CanAppend$ **and** ($IncomingMessage$ **contains** minimum information)
3: $Temp \leftarrow TaskQueue.filter(Item.topic$ **is** $IncomingMessage.topic)$
4: $CanAppend \leftarrow CanAppend$ **and** ($Temp.command$ **not contains** $IncomingMessage.command)$
5: **if** $CanAppend$ **then**
6: $TaskQueue$.Append($IncomingMessage$)
7: **end if**
8: **end function**

The requests to create a task may be sent internally or externally to the AMR. The internal procedure is based on a subscriber available to receive the requests arising from local decision-making.

For actions submitted by the Remote Unit or by the Administrator, use the database of the Remote Unit as a communication channel, also being used as a history for the actions submitted. Therefore, the algorithm 6 is executed periodically to check for new actions submitted by superior hierarchical levels, besides including actions requested through the AMR in the database to maintain the history and the status of the actions.

**Algorithm 6** Get task queue from Remote Unit Database

**Require:** $TaskQueue$ as list
**Require:** $AppendToTaskQueue()$ from algorithm 5
1: $RemoteQueue \leftarrow$ GetFromRemoteUnit($Actions$ **not** finished)
2: **for** $Item$ **in** $RemoteQueue$ **do**
3: $Local \leftarrow TaskQueue.find(Item)$
4: **if** $Local$ **is** empty **then**
5: $AppendToTaskQueue(TaskQueue, Item)$
6: **end if**
7: **if** $Item.status \neq Local.status$ **and** $Local.status$ **is** finished **then**
8: $Item.status \leftarrow Local.status$
9: **else**
10: $Local.status \leftarrow Item.status$
11: **end if**
12: **end for**
13: **for** $Item$ **in** $TaskQueue$ **do**
14: UpdateOrAddInRemoteUnit($Item$)
15: **end for**

II) Queue management

As proposed in section 3.2.4, the task queue management aims to send tasks from the queue to other subsystems of the AMR. For this purpose, the algorithm 7 is performed, which obtains the tasks queue produced by the algorithm 5. It sorts the queue according to the priority of task execution, in addition to handling the finished and running tasks and submitting the queue's updates to the remote unit's database.

**Algorithm 7** Queue management

**Require:** $TaskQueue$ as list
1: **while** $TaskQueue$ **is not empty do**
2: $TaskQueue \leftarrow TaskQueue.sort(priority)$
3: **for** $Task$ **in** $TaskQueue$ **do**
4: $Temp \leftarrow TaskQueue.filter(Item.topic$ **is** $Task.topic$ **and** $Item.priority < Task.priority)$
5: $Temp.status \leftarrow$ cancelled
6: $TaskQueue.update(Temp)$
7: **if** $Task$ **is** finished **then**
8: $Task.status \leftarrow$ finished
9: **else if** $Task$ **is** pendant **then**
10: SendCommand($Task$)
11: **end if**
12: UpdateRemoteUnit($Task$)
13: **if** $Task.status$ **is not** pendant **then**
14: $TaskQueue.remove(Task)$
15: **end if**
16: **end for**
17: **end while**

## 4.3 Implementation of the Remote Unit

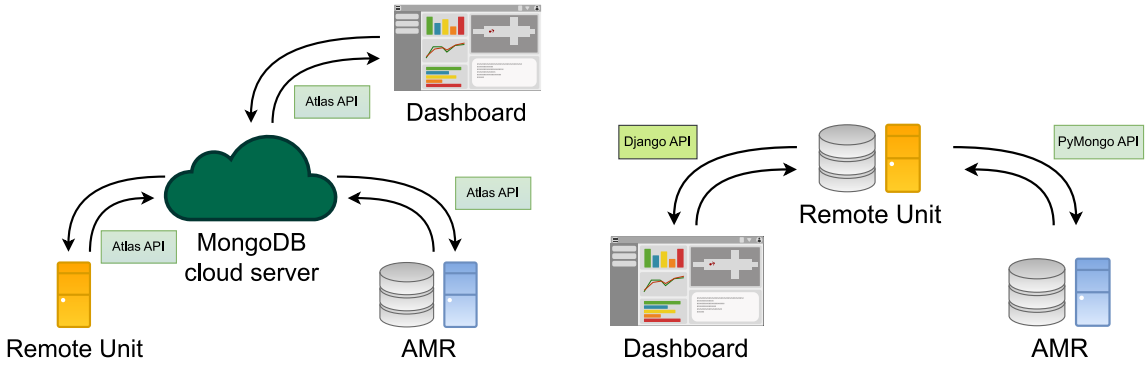
This section aims to explore the development of the processes attributed to the Remote Unit through figure 3.2, focusing on developing the mechanisms and routes through which the information should flow. This section also includes how information is sent from the Administrator to the other hierarchical levels. This section is divided into three parts: management and data storage, data processing, and communication with the dashboard.

The Remote Unit is hosted in a cloud server made available for the project, powered by Ubuntu 20.04, the same operating system implemented in the AMR.

### 4.3.1 Data storage and management system

By opting for this DBMS in the section 4.1.2, there are at least two possibilities for implementing the Remote Unit Database. The first is to base the remote database on the cloud storage service, which would act as an intermediary service through which the

information must pass, as illustrated in figure 4.5a, where the necessary queries would be performed through the existing services in the Atlas platform [61].




(a) Using MongoDB Atlas (b) Using MongoDB Community

Figure 4.5: Remote Unit Database connection diagram

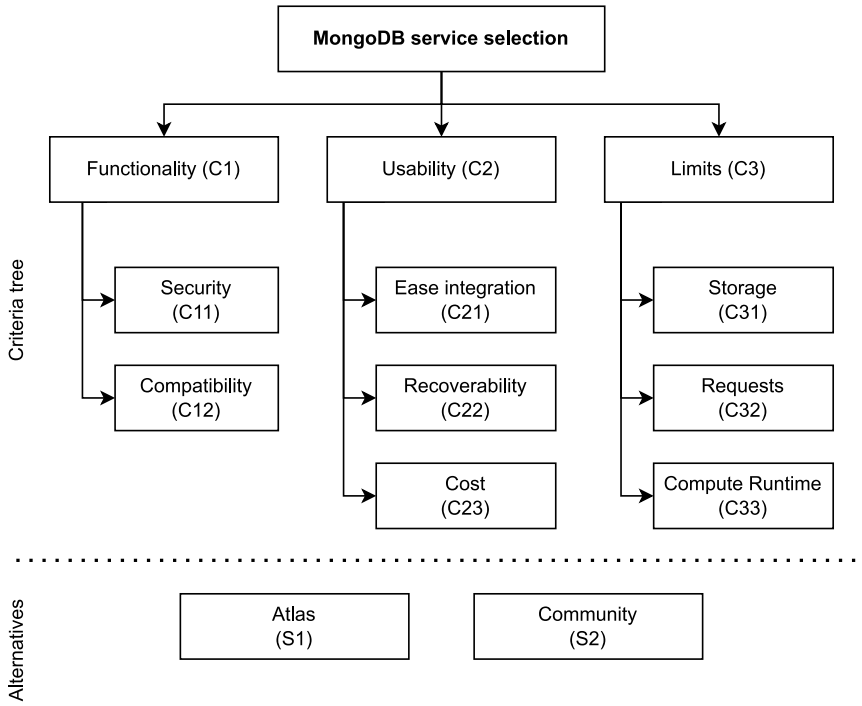



Figure 4.6: AHP criteria tree for Remote Unit Database

In another way, the storage service is executed internally through MongoDB Community software, as illustrated in figure 4.5b, and the query services are intermediated through Application Programming Interface (API)s in Django [62] and PyMongo [63],

[64]. The Remote Unit provides the dashboard, but in figure 4.5, it is being handled as an application.

Based on the criteria presented in the study conducted by Hanine, Boutkhoum, Tikniouine, *et al.* [65], the criteria tree shown in figure 4.6 is drawn up, which contains two software alternatives, three criteria and eight subcriteria. Through the application of the Multiple Criteria Decision Making (MCDM) TOPSIS [57], it is expected to obtain the software solution that best fits the needs presented by this work.

The matrices of the weights and other tables used to apply the method are available in the appendix C.1. According to the applied method, it is possible to conclude that the most suitable software is MongoDB Community.

### 4.3.2 Post-processing of data and remote decision-making

Post-processing and remote decision-making are complementary processes, as illustrated in figure 3.7, being performed in the Remote Unit by using scripts that perform queries in the database of the Remote Unit. In this way, these processes obtain the required information for the proposed verifications.

The dashboard query process is equivalent to the GET method presented in section 4.3.3. Therefore, it will not be considered in this section.

The Remote decision-Making subprocess follows the algorithm 8, where the data contained in the database is obtained through PyMongo [63], using pipelines to perform the necessary queries for the data analysis. Subsequently, data are shared with functions able to analyze the behaviour, check the parameters and suggest actions to improve the platform's functioning in some aspects.

After each loop execution, if there are actions, these are attached to the database, which should be handled through the scripts presented in section 4.2.4 to create tasks for AMR.

**Algorithm 8** Post-processing of data and remote decision-making

**Require:** *Client* as PyMongo Client
**Require:** *Pipelines* as a list of MongoDB aggregation pipelines
**Require:** *Results* as list
**Require:** *Actions* as list
**Require:** $CheckParameters()$ as functions that return actions
**Require:** $CheckBehaviours()$ as functions that return actions
**Require:** $CheckImprovements()$ as functions that return actions
**while** *Client.available* **and** AMR **is** running **do**
    $Results.clean$
    **for** *Pipeline* **in** *Pipelines* **do**
        $Results.append(Client.aggregation(Pipelines))$
    **end for**
    $Actions.append(CheckParameters(Results)$
    $BehavioursAction \leftarrow CheckBehaviours(Results)$
    $ImprovementsAction \leftarrow CheckImprovements(Results)$
    $Actions.append(ParametersAction, BehavioursAction, ImprovementsAction)$
    $Client.insertMany(Actions)$
**end while**

### 4.3.3 Dashboard communications

The Dashboard application and the database are communicated through a Django API, where it receives requests in HyperText Transfer Protocol (HTTP) and performs the necessary queries to the Remote Unit Database. It provides a Uniform Resource Locator (URL) with the capability of GET and POST methods, where the data needed for the query is sent together with the request body. Both methods are treated in more detail below:

I) GET:

This method performs database queries using the aggregation operations of MongoDB [66], which utilize pipelines that manipulate the documents in one or more database collections. For the correct performance of this operation, the request's URL must contain a query string with the parameters presented in the table 4.5.

When the aggregation has been performed against the MongoDB server, the Django server responds to the request with the list of documents resulting from the aggregation

in the format JSON. Since there are divergences between the types of variables supported by the MongoDB BSON and the response of the request, as exposed in Table 2.1, it is necessary to apply transformations in the values not supported to adequate them.

Table 4.5: Query string parameters of the GET method

| Name | Data type | Description |
|---|---|---|
| database | string | Database name |
| collection | string | Collection name |
| pipeline | string | Aggregation pipeline [66] |

II) POST:

This method performs a function called find_one_update_one [67] in the database. When a specific document is searched and modified with the informed values, a new document containing the specified values is created if the document is not found. In the request's body, sending an object JSON with the information in the table 4.6 is necessary. The response of this method informs only of the success or failure of the request.

Table 4.6: JSON expected in the body of the POST method

| Name | Data type | Description |
|---|---|---|
| database | string | Database name |
| collection | string | Collection name |
| filter | string | Query selectors [68] |
| update | string | Aggregation pipeline [66] |

## 4.4 Dashboard development

Following the 3.3 section, the dashboard genre best suited to the project is the Analytic dashboards, a versatile category for viewing detailed data sets.

The application Dashboard is based on the React library as defined in section 4.1, following the initial concepts determined through the response obtained with the interview schedule present in appendix A and explored in section 3.3. Thus, a web application is

accessible to the Administrator, where through this application, the analysis and interaction of the data provided by the platform allow the Administrator to interact with the equipment.

To support the development of the application, the template Mantis Free – React Admin Dashboard Template [69] is selected, which is a template for a web application that is available for use under the MIT Licence[2], which allows the use and modification for all purposes [71].

Two component libraries are also utilized that use the same type of licence with similar terms [72], [73]. These libraries are Ant [74], and MUI [75].

The development of the application is divided into three parts explored throughout this section: layout components, dashboard content and interactivity with the Administrator. Through these topics, it is expected to define the dashboard composition as proposed by Bach, Freeman, Abdul-Rahman, *et al.* [19] through the definition of the aspects not previously described as layout, structure and interaction, besides the visual representations available to the user.

### 4.4.1 Components of layout

The pages of the application, as delineated in section 3.3, possess three main elements of layout, the header, sidebar and central area. In addition to these three elements, the structure of the application and page layouts should be defined. These layout elements are explored below:

I) Header:

The header page, illustrated through the figure 4.7, is an element of the application layout and is intended to provide information in a streamlined format concerning the platform's current status, besides having two resources for interaction with the Administrator. As an element present on all pages, it seeks a minimalist format to not protrude

[2]It is a type of licence created by the Massachusetts Institute of Technology (MIT) specific for software [70].

over the primary part of the application but bring elements that are easily recognized and interpreted by the user. The items listed in the figure 4.7, are described below:


Figure 4.7: Application header

1) Sidebar button:
   By interacting, it hides or shows the menu sidebar;
2) Connection indicator with the AMR:
   It is an indicator icon whose colour and shade vary according to the connection speed of the AMR with the Remote Unit. It is discussed in more detail throughout the 4.4.2 section;
3) AMR battery charge percentage indicator:
   An indicator that, like the previous item, varies its colour and shade to indicate to the Administrator the current charge percentage of the AMR. Explored in more detail in the 4.4.2 section;
4) Button for sending "Go to" command:
   This interactive button can send a task to the AMR via the POST method explored in section 4.3.3. Following this, the internal platform processes will handle the task submitted.

II) Dashboard structure:

The structure used in the application uses the open type where there are different pages in the navigation of data, and there is no direct way to increase the detail of information, as occurs in the hierarchical structure [19]. With the routes indicated in the figure 4.8.

These routes allow the user to navigate freely through the different levels of detail of the data available in the application. Therefore, there are two types of pages, the

initial page, which presents information in visual format from different data sources, and the database page, where the information is presented in table format and contains the complete data referring to a specific data source.

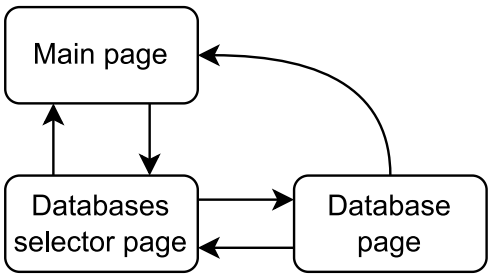



Figure 4.8: Pages structure

III) Sidebar:

This element is intended to allow navigation between pages and features of the Dashboard via the buttons. The sidebar is illustrated in figure 4.9, where the sidebar is shown or hidden through the interaction with element 1 of figure 4.7.

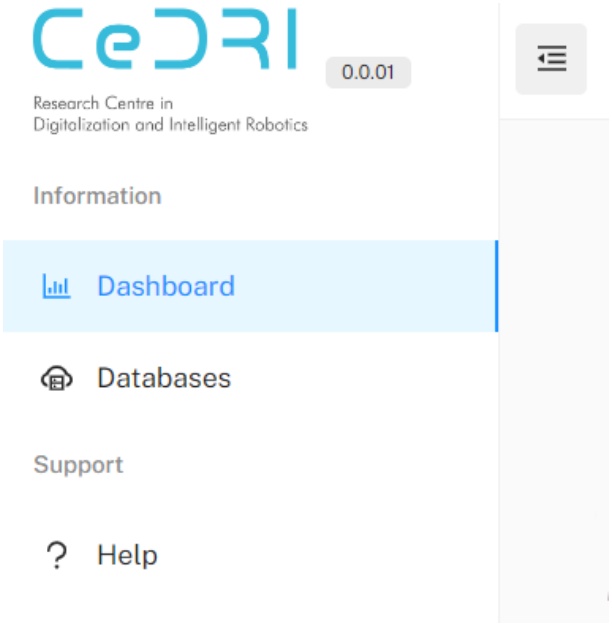
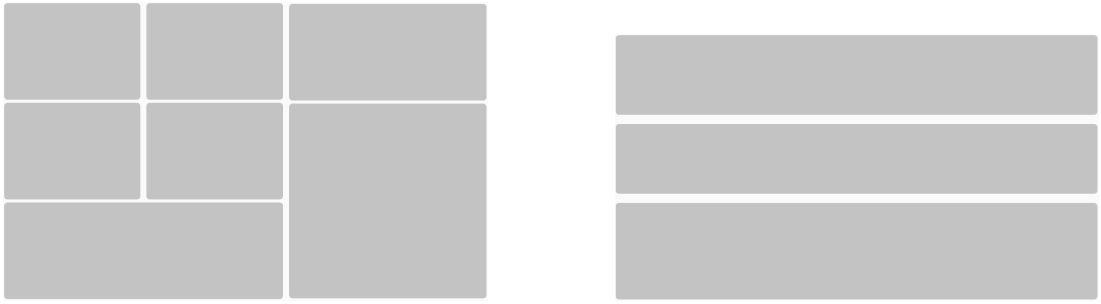



Figure 4.9: Application sidebar

IV) Page Layout:

Regarding the main page of the application, the open format is chosen, as illustrated in figure 4.10a, where no semantic organization or association among the frames is required, which are organized in a grid that holds different formats and data sources [19].

The Database Page provides information from only one data source, and its objective is to facilitate the correlation between the raw data and the processed data. The layout in the table is chosen, as disposed of in the figure 4.10b, that arranges the visualization frames in lines, assisting in the recovery and correlation of the information [19].



(a) Main page layout (b) Data base page layout

Figure 4.10: Main page and data bases pages layout

### 4.4.2 Dashboard content

The dashboard content is composed of elements that abstract the data present in the dataset through visual representations [19]. This section aims to define the characteristics of the graphical representations used in the Dashboard to help the Administrator interpret the data. Thus, the development is divided into two segments intended for the types of representation adopted in this project: informative and standard. The resources and objectives of the segments are explored below:

I) Informative representations:

The informative representations communicate the status of the platform systems to the Administrator using images. These representations are included in the header of

the application, as described in section 4.4.1, which uses simple icons and variations of filling colour to communicate the information to the user, as illustrated through the figure 4.11a, referring to the pictogram of connection quality of the platform with the Remote Unit, while in the figure 4.11b there is shown the pictogram that indicates to the Administrator the percentage of battery. Both pictograms transmit a value through visual representation, in the form of filling and colour scale, and can therefore be classified as filled pictograms [19].

(a) Connection quality pictogram (b) Battery percentage pictogram

Figure 4.11: Battery and connection pictograms

In addition, another feature that uses these visual representations is the diagnosis of the subsystems, represented by the figure 4.12, where it is possible to identify the use of the status icon, the relative icon to each subsystem and the corresponding name. The objective is to transmit the information assertively and quickly, using data transmitted by each subsystem to express the presence of errors and alarms.

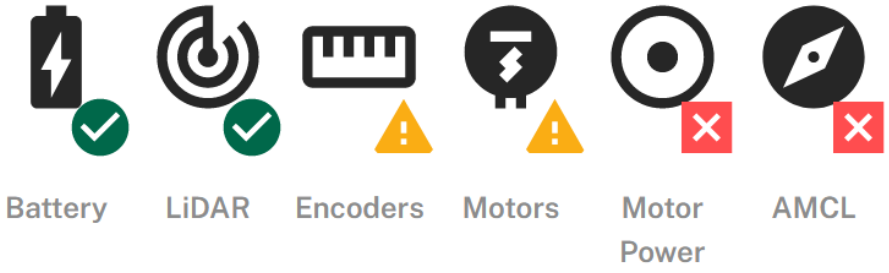



Figure 4.12: Pictograms of subsystem status

II) Standard representations:

The standard representations compose all the remaining data visualizations available in the application. Using the visualization techniques explored in section 2.2.3 to provide

the Administrator with tools for data abstraction, pattern detection and other needs for better platform management.

To meet the Administrator's needs, the visualizations presented in the table 4.7 have been developed, where it is exposed the subsystem, the analyzed variable, the type of chart and the indication of the reference figure available in the appendix D.

Table 4.7: List of standard dashboard content

| Subsystem | Variables | Graph type | Figure |
|---|---|---|---|
| Battery | Percent | Progress bar [4], [19] | D.5a |
| | Current | Progress bar [4], [19] | D.5a |
| | Voltage | Progress bar [4], [19] | D.5a |
| | Temperature | Progress bar [4], [19] | D.5a |
| | Percent; date | Area [4] | D.1a |
| | Voltage; date | Area [4] | D.1b |
| Computer | Memory; date | Area [4] | D.1c |
| | CPU; date | Line [4] | D.3d |
| | Memory; process | Sunburst [22] | D.6a |
| | CPU; process | Sunburst [22] | D.6b |
| Connection | RTT; date | Area [4] | D.1d |
| | RTT | Map [4] | D.4c |
| | Connection | Map [4] | D.4a |
| Motors | Current | Progress bar [4], [19] | D.5b |
| | PWM | Progress bar [4], [19] | D.5b |
| | Rotation Rate | Progress bar [4], [19] | D.5b |
| | Current; date | Line [4] | D.3a |
| | PWM; date | Line [4] | D.3b |
| | Rotation Rate; date | Line [4] | D.3c |

Table 4.7: List of standard dashboard content (continued)

| Subsystem | Variables | Chart type | Figure |
|---|---|---|---|
| Nodes | Node; topic | Graph [21] | D.2 |
| Position | X; Y | Map [4] | D.4d |
| | X; Y | Map [4] | D.4b |
| | X; Y | Map [4] | D.4e |
| | Yaw; date | Area [4] | D.1e |

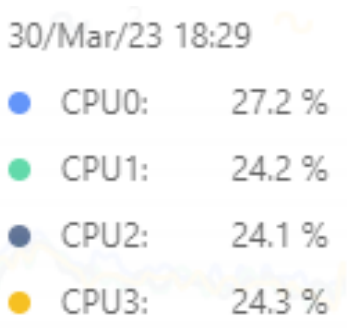
### 4.4.3 Interactivity with the Administrator

The interactivity with the Administrator occurs through the GUI provided by the Remote Unit. Where occurs the transference of information from the Administrator to the AMR with the intermediation of the Remote Unit as verified in the algorithm 6, besides the ability to interact with the data provided by the dashboard application, as proposed in the section 3.3.

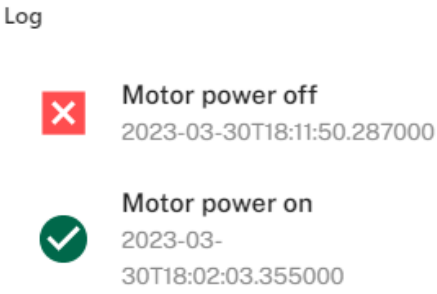
Furthermore, the application includes data exploration and customization mechanisms to help users interpret information, such are mechanisms that support the user in obtaining relevant information [19]. The data exploration and customization features are discussed below:

I) Exploration:

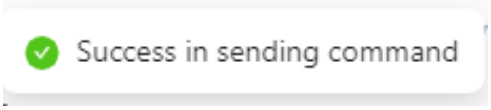
The pop-ups are the exploration mechanisms present in this application, which are simple resources that increase the user's interactivity with the data. The purpose of these pop-ups is to provide more details about the application feature, as illustrated by figure 4.13, showing different examples of pop-ups available in the application.




(a) Pop-up of CPU usage chart




(b) Pop-up of subsystems status visualization



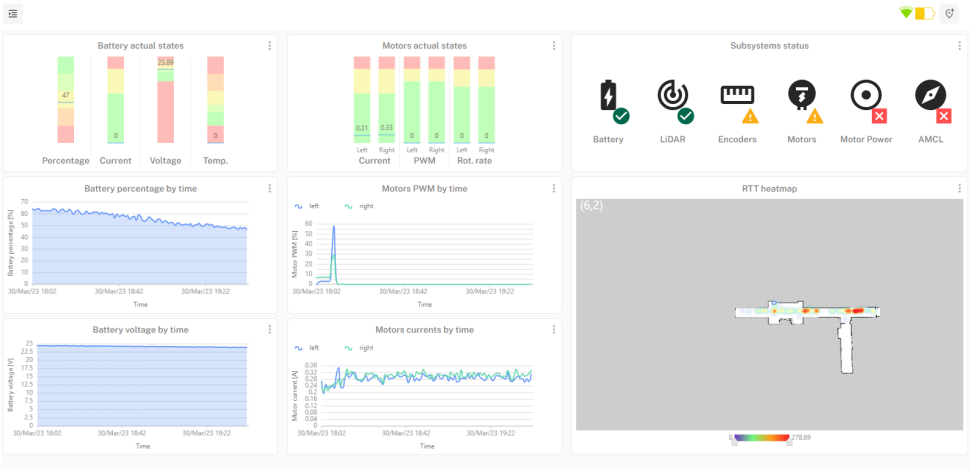

(c) Pop-up of command sending

Figure 4.13: Example of application pop-ups

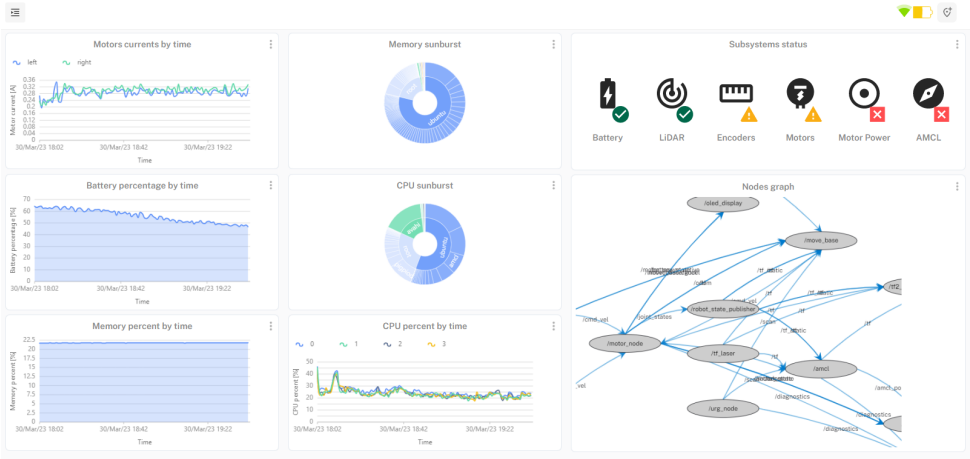
II) Customization:

Besides the data exploration pop-ups, the application also allows for the customization of the main page, and it is an interactive mechanism which enables the Administrator to change the visual features present on the main page of the application. Therefore, this allows the Administrator to adapt the available tools to the current needs and objectives [19]. This feature is illustrated in figure 4.14, which shows the change in the content available on the application page.

This customization mechanism occurs by storing the Administrator preferences in the Remote Unit's database. Using the POST method, as described in section 4.3.3, the information contained in the database is modified. The metadata related to the visual feature, the position of this feature in the main page grid and the visual size are stored. These data are queried by the application when the page is loaded to assemble the visualization that the user defines.




(a) Example 1




(b) Example 2

Figure 4.14: Example of the customization of the main page

# Chapter 5

# Results Analysis

This chapter presents the analysis of the tests performed to verify if the development described in chapter 4 satisfies the requirements described in chapter 3 as well as testing the impact of the system on platform performance, battery consumption, and computer resource usage during operation.

In addition to these tests, the system developed is compared with a commercial solution for managing and monitoring fleets of UGVs. Checking if the solution implemented in this work satisfactorily meets commercially-used feature requirements.

The chapter is divided into three sections. The first evaluates the information flow, the second analyses the use of embedded resources in the platform, and the third compares the developed project and the commercial solution.

## 5.1 Information flow assessment

This section is intended to present the tests and their results conducted to ensure that the process blocks implemented align with their proposed objectives and requirements.

This section is divided into two parts, focused on analyzing aspects related to data capture and manipulation and information transmission. Therefore, it encompasses the processes proposed in figure 3.2.

### 5.1.1 Data acquisition and processing

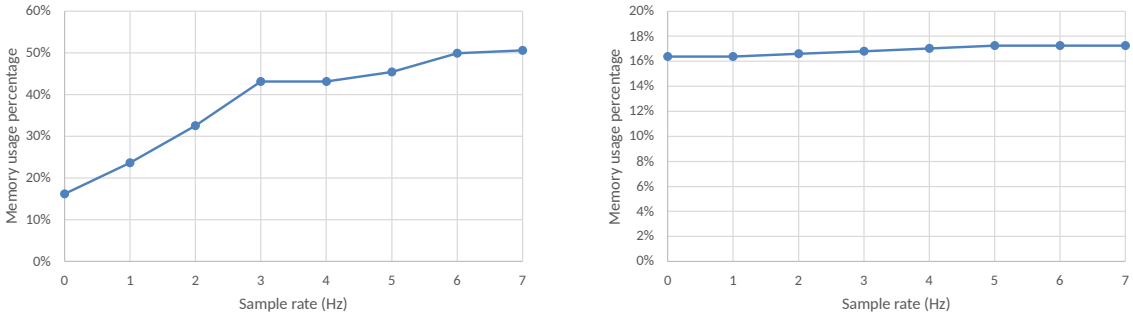
The acquisition and processing of data are associated with the processes of data acquisition, pre-processing and data management, which are performed on board the AMR, as explained in 3.2.1 and 3.2.2.

When executing the scripts of data capture and manipulation in the nodes defined in the table 4.2, using acquisition rates from 1 Hz to 3 Hz, it is observed that the processes occur in a way consistent with the proposal and without significant effects on the platform operation.

However, when sample rates were between 3 Hz and 6 Hz, it was observed that while the vehicle is in motion, there occur time delays in the reaction to obstacles by the Motion Planning task. Therefore, the AMR gets closer to the obstacles with the increase of the capture rate.

When the sample rate is higher than 7 Hz, it is observed that the Motion Planning task is prejudiced so that the equipment cannot recognize obstacles correctly. This behaviour is potentially harmful to the health of the equipment. The same situation is observed when acquiring the information provided by the LiDAR, independently of the acquisition rate.

When analyzing the behaviour of using the Central Processing Unit (CPU) and memory of the onboard microcomputer in the AMR, comparing these data with the sample rate. The curves demonstrated in the figures 5.1a and 5.1b are obtained.




(a) CPU usage test results by sample rate (b) Memory usage test results by sample rate

Figure 5.1: Sample rate impact test in the use of the microcomputer

It can be observed that there are no significant changes in the use of volatile memory. Although, for the use of CPU, it shows significant growth, exceeding 50% of the consumption of this resource for sample rates higher than 5 Hz.

### 5.1.2 Transmission of information

The transmission of information in the project occurs in two different modes as illustrated in the figure, 4.5b, AMR and Remote Unit or Remote Unit and Dashboard. The effectiveness of the communication and the corresponding scripts that perform the sending and receiving of information is tested in this section. For this purpose, the analysis conducted below is divided into the modes mentioned earlier.

I) AMR and Remote Unit:

The exchange of information between these components occurs via the PyMongo API, as illustrated in figure 4.5b, that establishes a connection between the AMR and the MongoDB server that is executing in the Remote Unit [64].

The scripts that are developed from the algorithms 3 and 6 allow the exchange of information between the AMR and the Remote Unit. The data transmitted by the AMR is exclusively information about aspects and status of the platform's systems and is directly stored for use by the subsequent processes. The data transmitted to the AMR consists of commands generated through the decision-making processes, which come from either the Remote Unit or the Administrator.

The communication between the AMR and the Remote Unit is considered efficient, as no loss or error is observed. However, analyzing the data collected by the system developed in this project makes it possible to identify attenuation zones in the wireless signal.

Where through the heat map presented in figure 5.2, the distribution of the average RTT obtained from the connection established between the AMR and the Remote Unit can be observed. The red zones indicate signal attenuation zones of the platform's wireless

connection.

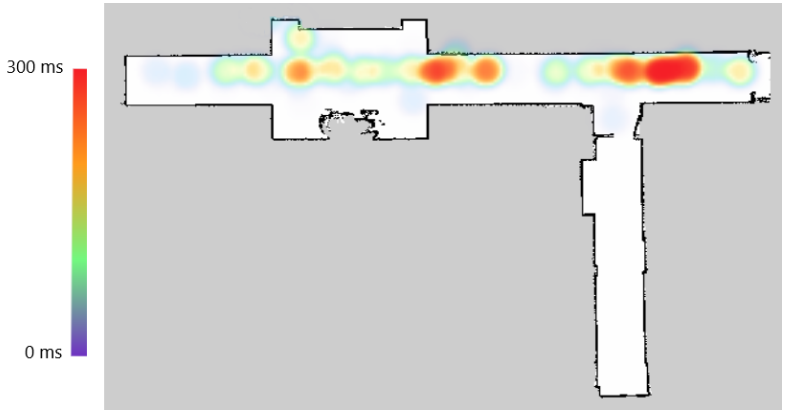



Figure 5.2: Heatmap of the average RTT between AMR and Remote Unity

These zones can cause partial or complete loss of communication with the AMR, and a potential development in future work is the development of path planning algorithms that can avoid or mitigate the platform crossing through these zones.

II) Dashboard and Remote Unit:

The dashboard and the remote unit are communicated via a Django API. This communication occurs via the HTTP protocol and uses the GET and POST methods, as described in the 4.3.3 section. The correct generation of resources on the dashboard and the ability to force the execution of actions on the robotic platform is observed. No failures in communication or information generation were observed. The average response time for requests sent to the server is approximately 90 ms.

## 5.2 Use of embedded resources

Through the execution of the routines developed in chapter 4, with a sample rate of 1 Hz. The objective of this section is to verify the impact caused by the use of the system in its totality in the operation of the platform, evaluating the use of memory, the use of the CPU and the battery cycle. Therefore, these data are collected every five minutes while the platform performs the one-hour looping route.

Obtaining the graphs represented in the figures 5.3, 5.4a and 5.4b, where the horizontal axis is the time, and the vertical axis is the observed magnitude, and the results are grouped in two groups containing the average of the results for the control samples and the samples with the execution of the system.

I) Battery test result:

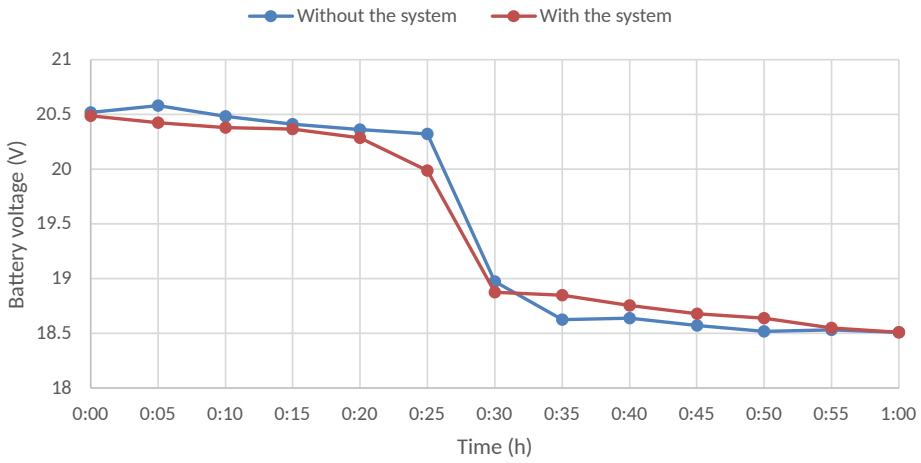



Figure 5.3: Battery test results by time

No significant voltage and battery capacity decrease can be observed in these tests compared to the control tests. This way, it is possible to infer that the execution of the system does not cause significant losses in the autonomy of the platform.

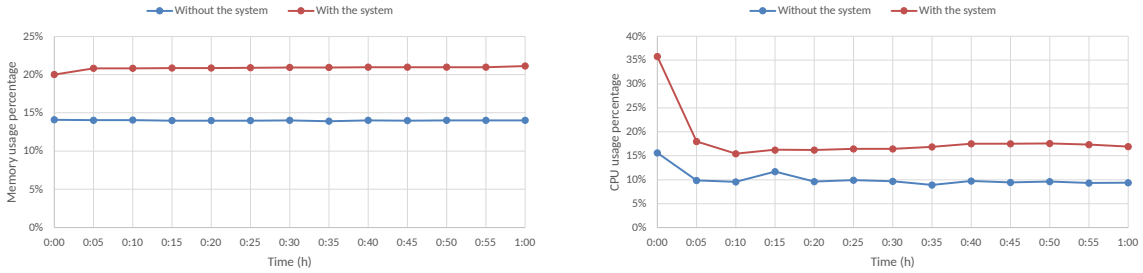
Through these tests, it is possible to identify a discharge curve not following the expected battery discharge curve [76], which may indicate damage in one or more battery pack cells embedded in the platform.

II) Microcomputer test result:

Regarding the use of the microcomputer's volatile memory, an average increase of approximately 6.5 percentage points is observed compared to the control tests. Although the total usage was close to 20%, it does not represent a significant usage value.

For the usage of the CPU, the average use value of the four cores present in the microcomputer is obtained. In this way, it is possible to observe an average increase of

9 percentage points compared to the control values. However, the average value of the use of the CPU is maintained below 20% of the total in the main part of the samples collected, meaning that it does not represent a significant increase in the use of CPU.




(a) Memory usage test results by time

(b) CPU usage test results by time

Figure 5.4: Results of the impact test in the use of the microcomputer

## 5.3 Comparison with the commercial solution

This project proposes a free solution for projects based on the ROS, which enables data capture, storage, command transmission, and decision-making capabilities in the system. Similar solutions that either include or partially include the features proposed in this project can be found in commercial products [77]–[79].

For comparative purposes of the developed solution along this work, the commercial solution developed by the company Freedom Robotics is selected. This platform allows managing fleets of UGVs that use the ROS [77], [80].

For comparison purposes, the developed solution was compared with a commercial solution developed by Freedom Robotics [77], [80]. Table 5.1 presents a feature comparison between the two solutions.

Table 5.1: Feature comparison between the solution developed and the commercial product

| Feature | Commercial solution | Proposed solution |
|---|---|---|
| Alerts | Yes | Yes |
| Customizable layout and graphs | Yes | Yes |
| Data store | No | Yes |
| Decision-making capability | No | Yes |
| Maps | Yes | Yes |
| Multiple UGVs | Yes | No |
| Remote manual operation | Yes | No |
| Remote SSH | Yes | No |
| Security features | Yes | No |
| Sending commands | Yes | Yes |

The table suggests areas for improvement in future work and that the solution proposed in this work satisfies, in part, the product that is commercially offered. It is important to note that the commercial solution also includes information security characteristics that should be studied in further work.

Additionally, the commercial solution does not focus on storing information for long periods or storing information acquired. At the same time, the equipment is disconnected, allowing different approaches to information management compared to the strategies adopted in the development of this work.

# Chapter 6

# Conclusion and Future Work

This work presented a solution for implementing a platform management and monitoring system based on ROS. The solution attended to the necessities and particularities offered by the Magni robotic platform, which belongs to the CeDRI.

The control architecture proposed in this project combines centralized and decentralized architectures. Through the processes presented in this work, it is possible to obtain data that support autonomous or non-autonomous decision-making besides supplying the system manager with graphical tools for analyzing the acquired data. This approach does not interfere with the autonomous capabilities already operational in the robotic platform.

To analyze the equipment's behaviour and the impacts caused by the implemented processes over these tasks. On some occasions, the equipment was exposed to stress conditions, particularly in Motion Planning and Path Planning tasks, using dynamic obstacles such as people. In some of these stressful situations, it was observed that the platform exhibited instabilities, suggesting potential for system improvements. However, the implemented processes and their features have been validated in diversified conditions in the area mapped and known by the equipment.

Furthermore, the application, developed to provide tools for analyzing the data transmitted by the AMR, has several characteristics and functionalities shared with a commercial product. The shared features indicate that this project segment is following industrially used products. However, the approach for acquiring and handling the data

presents significant differences, which can be explored in future work.

In summary, the tools developed in this work presented a satisfactory result and agreed with the established objectives. However, there is potential for improvements and development in future works concerning the platform that was the object of study and for the expansion and implementation of these tools in other robotic platforms.

## 6.1 Future Works

As a suggestion for the continuity of this work, it is proposed to increase its resources, complexity and available tools, besides verifying the effect of different approaches of those adopted in the development of this work. Considered the following suggestions:

- Implement systems and mechanisms for the security of information. To protect the data and privacy of the communications conducted by the system [81], [82];
- Expand the management capacity of the system to include multiple robotic platforms and users, enabling the system to handle more complex management situations [77];
- Integrate Artificial Intelligence for the decision-making process performed by the Remote Unit and the robotic platform, using more complex data sets to support autonomous decision-making [83];
- Improve task prioritization by applying multi-criteria decision methods to determine the optimal choice among available alternatives [2];
- Integrate additional sensors to acquire more information about the operation and status of the platform, enabling better decision-making;
- Implement distributed computing systems, which act on demand, to reduce the use of microcomputers onboard the robotic platform [84];

- Identify and compare alternative data acquisition and management methods to minimize the impact on resources embedded in the robotic platform [80].

# Bibliography


[1] E. A. Oyekanlu, A. C. Smith, W. P. Thomas, *et al.*, "A review of recent advances in automated guided vehicle technologies: Integration challenges and research areas for 5g-based smart manufacturing applications," *IEEE Access*, vol. 8, pp. 202 312–202 353, 2020, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3035729.

[2] D. Agarwal and P. S. Bharti, "A case study on AGV's alternatives selection problem," *International Journal of Information Technology*, vol. 14, no. 2, pp. 1011–1023, Mar. 1, 2022, ISSN: 2511-2112. DOI: 10.1007/s41870-018-0223-z. [Online]. Available: https://doi.org/10.1007/s41870-018-0223-z (visited on 04/14/2023).

[3] M. De Ryck, M. Versteyhe, and F. Debrouwere, "Automated guided vehicle systems, state-of-the-art control algorithms and techniques," *Journal of Manufacturing Systems*, vol. 54, pp. 152–173, Jan. 1, 2020, ISSN: 0278-6125. DOI: 10.1016/j.jmsy.2019.12.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0278612519301177 (visited on 12/25/2022).

[4] L. T. Mohammed, A. A. AlHabshy, and K. A. ElDahshan, "Big data visualization: A survey," in *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, Jun. 2022, pp. 1–12. DOI: 10.1109/HORA55278.2022.9799819.

[5] G. Chawla, S. Bamal, and R. Khatana, "Big data analytics for data visualization: Review of techniques," *International Journal of Computer Applications*, vol. 182, Oct. 1, 2018. DOI: 10.5120/ijca2018917977.

[6] Ishika and N. Mittal, "Big data analysis for data VisualizationA review," in *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Sep. 2021, pp. 1–6. DOI: 10.1109/ICRITO51393.2021.9596423.

[7] H. Meissner, R. Ilsen, and J. C. Aurich, "Analysis of control architectures in the context of industry 4.0," *Procedia CIRP*, 10th CIRP Conference on Intelligent Computation in Manufacturing Engineering - CIRP ICME '16. [Edited by: Roberto Teti, Manager Editor: Doriana M. D'Addona], vol. 62, pp. 165–169, Jan. 1, 2017, ISSN: 2212-8271. DOI: 10.1016/j.procir.2016.06.113. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212827117300641 (visited on 12/26/2022).

[8] D. W. Gage, "UGV history 101: A brief history of unmanned ground vehicle (UGV) development efforts," *UGV history 101: A brief history of Unmanned Ground Vehicle (UGV) development efforts*, 3rd ser., vol. 13, Jan. 1, 1995, ISSN: 17486815. DOI: 10.1016/j.bjps.2007.12.046. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1748681508002933 (visited on 02/04/2023).

[9] M. B. Alatise and G. P. Hancke, "A review on challenges of autonomous mobile robot and sensor fusion methods," *IEEE Access*, vol. 8, pp. 39 830–39 846, 2020, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2975643.

[10] A. d. O. Júnior, "Combining particle filter and fiducial markers in a SLAM-based approach to indoor localization of mobile robots," Masters' theses, Polytechnic Institute of Bragança (IPB) and Federal Technological University of Paraná (UTFPR), Bragança, 2021, 108 pp.

[11] I. T. AG, *Mobile robots (agv, amr)*, https://www.infineon.com/cms/en/applications/industrial/robotics/mobile-robots/, 2022. (visited on 12/11/2022).

[12] D. Trentesaux, "Distributed control of production systems," *Engineering Applications of Artificial Intelligence*, Distributed Control of Production Systems, vol. 22, no. 7, pp. 971–978, Oct. 1, 2009, ISSN: 0952-1976. DOI: 10.1016/j.engappai.2009.05.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197609000797 (visited on 12/26/2022).

[13] K. Schweichhart, "Reference architectural model industrie 4.0 (rami 4.0)," *An Introduction. Available online: https://www. plattform-i40. de I*, vol. 40, 2016.

[14] U. Robotics, *Robot base magni silver*, 2022. [Online]. Available: https://www.ubiquityrobotics.com/product/magni-silver/ (visited on 12/11/2022).

[15] U. Robotics, *Ubiquity robotics GitHub*, 2022. [Online]. Available: https://github.com/UbiquityRobotics (visited on 12/11/2022).

[16] U. Robotics. "Magni documentation." (2023), [Online]. Available: https://learn.ubiquityrobotics.com/ (visited on 02/16/2023).

[17] J. Wang, Y. Yang, T. Wang, R. S. Sherratt, and J. Zhang, "Big data service architecture: A survey," *Journal of Internet Technology; Vol 21, No 2 (2020)*, Mar. 1, 2020. [Online]. Available: https://jit.ndhu.edu.tw/article/view/2261/2274.

[18] N. Bikakis, "Big data visualization tools," *CoRR*, vol. abs/1801.08336, 2018. arXiv: 1801.08336. [Online]. Available: http://arxiv.org/abs/1801.08336.

[19] B. Bach, E. Freeman, A. Abdul-Rahman, *et al.*, "Dashboard design patterns," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 342–352, Jan. 2023, Conference Name: IEEE Transactions on Visualization and Computer Graphics, ISSN: 1941-0506. DOI: 10.1109/TVCG.2022.3209448.

[20] T. H. Clawson, J. Leafman, G. M. Nehrenz, and S. Kimmer, "Using pictograms for communication," *Military Medicine*, vol. 177, no. 3, pp. 291–295, Mar. 2012, ISSN: 0026-4075, 1930-613X. DOI: 10.7205/MILMED-D-11-00279. [Online]. Available: https://academic.oup.com/milmed/article/177/3/291-295/4283647 (visited on 04/04/2023).

[21] J. L. Gross and J. Yellen, *Graph Theory and Its Applications, Second Edition*. CRC Press, Sep. 22, 2005, 799 pp., ISBN: 978-1-58488-505-4.

[22] L. Woodburn, Y. Yang, and K. Marriott, "Interactive visualisation of hierarchical quantitative data: An evaluation," in *2019 IEEE Visualization Conference (VIS)*, Oct. 2019, pp. 96–100. DOI: 10.1109/VISUAL.2019.8933545.

[23] Oracle. "What is a database?" (2022), [Online]. Available: https://www.oracle.com/database/what-is-database/ (visited on 12/30/2022).

[24] T. N. Khasawneh, M. H. AL-Sahlee, and A. A. Safia, "SQL, NewSQL, and NOSQL databases: A comparative survey," in *2020 11th International Conference on Information and Communication Systems (ICICS)*, ISSN: 2573-3346, Apr. 2020, pp. 013–021. DOI: 10.1109/ICICS49469.2020.239513.

[25] J. Han, H. E, G. Le, and J. Du, "Survey on NoSQL database," in *2011 6th International Conference on Pervasive Computing and Applications*, Oct. 2011, pp. 363–366. DOI: 10.1109/ICPCA.2011.6106531.

[26] K. Sambrekar, V. S. Rajpurohit, and J. Joshi, "A proposed technique for conversion of unstructured agro-data to semi-structured or structured data," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, Aug. 2018, pp. 1–5. DOI: 10.1109/ICCUBEA.2018.8697432.

[27] A. Eberendu, "Unstructured data: An overview of the data of big data," *International Journal of Computer Trends and Technology*, vol. 38, pp. 46–50, Aug. 25, 2016. DOI: 10.14445/22312803/IJCTT-V38P109.

[28] G. Ozsoyoglu and R. Snodgrass, "Temporal and real-time databases: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 7, no. 4, pp. 513–532, Aug. 1995, Conference Name: IEEE Transactions on Knowledge and Data Engineering, ISSN: 1558-2191. DOI: 10.1109/69.404027.

[29] C. Jensen and R. Snodgrass, "Temporal data management," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, pp. 36–44, Jan. 1999, Conference Name: IEEE Transactions on Knowledge and Data Engineering, ISSN: 1558-2191. DOI: 10.1109/69.755613.

[30] N. Q. Mehmood, R. Culmone, and L. Mostarda, "Modeling temporal aspects of sensor data for MongoDB NoSQL database," *Journal of Big Data*, vol. 4, no. 1, p. 8, Mar. 31, 2017, ISSN: 2196-1115. DOI: 10.1186/s40537-017-0068-5. [Online]. Available: https://doi.org/10.1186/s40537-017-0068-5 (visited on 12/30/2022).

[31] K. E. Iverson, "A programming language," in *Proceedings of the May 1-3, 1962, Spring Joint Computer Conference*, ser. AIEE-IRE '62 (Spring), event-place: San Francisco, California, New York, NY, USA: Association for Computing Machinery, 1962, pp. 345–351, ISBN: 978-1-4503-7875-8. DOI: 10.1145/1460833.1460872. [Online]. Available: https://doi.org/10.1145/1460833.1460872.

[32] B. C. Pierce, *Types and Programming Languages*. MIT Press, Jan. 4, 2002, 656 pp., Google-Books-ID: ti6zoAC9Ph8C, ISBN: 978-0-262-16209-8.

[33] P. Virtanen, R. Gommers, T. E. Oliphant, *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, Mar. 1, 2020, ISSN: 1548-7105. DOI: 10.1038/s41592-019-0686-2. [Online]. Available: https://doi.org/10.1038/s41592-019-0686-2.

[34] PSF, *History and license*, Oct. 18, 2022. [Online]. Available: https://docs.python.org/3/license.html (visited on 12/11/2022).

[35] A. Kumar and S. Panda, "A survey: How python pitches in IT-world," in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, Feb. 2019, pp. 248–251. DOI: 10.1109/COMITCon.2019.8862251.

[36] J. Kaplan-Moss, *Announcing the django software foundation*, Jun. 17, 2008. [Online]. Available: https://www.djangoproject.com/weblog/2008/jun/17/foundation/ (visited on 12/11/2022).

[37] S. Willison, *What is the history of the django web framework? why has it been described as "developed in a newsroom"?* 2016. [Online]. Available: https://www.quora.com/What-is-the-history-of-the-Django-web-framework-Why-has-it-been-described-as-developed-in-a-newsroom/answer/Simon-Willison (visited on 12/11/2022).

[38] C. Severance, "JavaScript: Designing a language in 10 days," *Computer*, vol. 45, no. 2, pp. 7–8, Feb. 2012, ISSN: 1558-0814. DOI: 10.1109/MC.2012.57.

[39] E. Wohlgethan, "Supportingweb development decisions by comparing three major javascript frameworks: Angular, react and vue. js," 2018, Publisher: Hochschule für Angewandte Wissenschaften Hamburg.

[40] E. Andreasen, L. Gong, A. Møller, *et al.*, "A survey of dynamic analysis and test generation for JavaScript," *ACM Comput. Surv.*, vol. 50, no. 5, Sep. 2017, Place: New York, NY, USA Publisher: Association for Computing Machinery, ISSN: 0360-0300. DOI: 10.1145/3106739. [Online]. Available: https://doi.org/10.1145/3106739.

[41] F. O. S. React. "React – a JavaScript library for building user interfaces." (2023), [Online]. Available: https://reactjs.org/ (visited on 02/08/2023).

[42] P. D. Dutonde, S. S. Mamidwar, M. S. Korvate, S. Bafna, and D. D. Shirbhate, "Website developmemt technologies: A review," Jan. 2022.

[43] A. Boicea, F. Radulescu, and L. I. Agapin, "MongoDB vs oracle – database comparison," in *2012 Third International Conference on Emerging Intelligent Data and Web Technologies*, Sep. 2012, pp. 330–335. DOI: 10.1109/EIDWT.2012.32.

[44] MongoDB. "MongoDB: The developer data platform | MongoDB." (2022), [Online]. Available: https://www.mongodb.com/ (visited on 12/31/2022).

[45] A. Amazon. "What is MongoDB? – non-relational database," Amazon Web Services, Inc. (2022), [Online]. Available: https://aws.amazon.com/documentdb/what-is-mongodb/ (visited on 12/31/2022).

[46] M. Quigley, B. Gerkey, K. Conley, *et al.*, "ROS: An open-source robot operating system," *ROS: an open-source Robot Operating System*, vol. 3, p. 5, May 2009.

[47] K. Wyrobek, "The origin story of ROS, the linux of robotics," *IEEE Spectrum*, Oct. 31, 2017. [Online]. Available: https://spectrum.ieee.org/the-origin-story-of-ros-the-linux-of-robotics (visited on 12/13/2022).

[48] E. Ackerman and E. Guizzo, "Wizards of ROS: Willow garage and the making of the robot operating system," *IEEE Spectrum*, Nov. 7, 2017. [Online]. Available: https://spectrum.ieee.org/wizards-of-ros-willow-garage-and-the-making-of-the-robot-operating-system (visited on 12/13/2022).

[49] M. Cashmore, M. Fox, D. Long, *et al.*, "ROSPlan: Planning in the robot operating system," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 25, no. 1, pp. 333–341, Apr. 2015. DOI: 10.1609/icaps.v25i1.13699. [Online]. Available: https://ojs.aaai.org/index.php/ICAPS/article/view/13699.

[50] A. Koubaa, "ROS as a service: Web services for robot operating system," *Journal of Software Engineering for Robotics*, vol. 6, no. 1, pp. 1–14, 2015, ISSN: 2035-3928.

[51] B. Dieber, B. Breiling, S. Taurer, S. Kacianka, S. Rass, and P. Schartner, "Security for the robot operating system," *Robotics and Autonomous Systems*, vol. 98, pp. 192–203, 2017, ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2017.09.

017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889017302762.

[52] Automate, *Robot operating system*, https://www.directindustry.com/pt/prod/automationware/product-192516-2394360, 2022. (visited on 12/11/2022).

[53] S. A. White, "Introduction to BPMN," *Ibm Cooperation*, vol. 2, 2004.

[54] H.-m. Li, P. Wang, L.-y. Fang, and J.-w. Liu, "An algorithm based on time series similarity measurement for missing data filling," in *2012 24th Chinese Control and Decision Conference (CCDC)*, ISSN: 1948-9447, May 2012, pp. 3933–3935. DOI: 10.1109/CCDC.2012.6244628.

[55] W. Wei and Y. Tang, "A generic neural network approach for filling missing data in data mining," in *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance (Cat. No.03CH37483)*, ISSN: 1062-922X, vol. 1, Oct. 2003, 862–867 vol.1. DOI: 10.1109/ICSMC.2003.1243923.

[56] Z.-X. Li, S.-H. Wu, C. Li, and Y. Zhang, "Research on methods of filling missing data for multivariate time series," in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, Mar. 2017, pp. 382–385. DOI: 10.1109/ICBDA.2017.8078845.

[57] E. Roszkowska, "Multi-criteria decision making models by applying the TOPSIS method to crisp and interval data," *Multi-criteria decision making models by applying the TOPSIS method to crisp and interval data*, pp. 200–230, 2011. [Online]. Available: https://mcdm.ue.katowice.pl/files/mcdm11.pdf (visited on 03/16/2023).

[58] H. M. Abdullah and A. M. Zeki, "Frontend and backend web technologies in social networking sites: Facebook as an example," in *2014 3rd International Conference on Advanced Computer Science Applications and Technologies*, Dec. 2014, pp. 85–89. DOI: 10.1109/ACSAT.2014.22.

[59] MongoDB. "32-bit limitations | MongoDB blog," MongoDB. (Jul. 8, 2009), [Online]. Available: `https://www.mongodb.com/blog/post/32-bit-limitations` (visited on 02/16/2023).

[60] MongoDB. "Databases and collections — MongoDB manual." (2023), [Online]. Available: `https://www.mongodb.com/docs/manual/core/databases-and-collections/` (visited on 02/17/2023).

[61] MongoDB. "Atlas m0 (free cluster), m2, and m5 limitations — MongoDB atlas." (2023), [Online]. Available: `https://www.mongodb.com/docs/atlas/reference/free-shared-limitations/` (visited on 01/16/2023).

[62] J. Vainikka, "Full-stack web development using django REST framework and react," Bachelor of Engineering, Metropolia University of Applied Sciences, Helsínquia, Finlândia, May 16, 2018, 46 pp.

[63] MongoDB. "API documentation — PyMongo 4.3.3 documentation." (2023), [Online]. Available: `https://pymongo.readthedocs.io/en/stable/api/index.html` (visited on 03/15/2023).

[64] MongoDB. "PyMongo — MongoDB drivers." (2023), [Online]. Available: `https://www.mongodb.com/docs/drivers/pymongo/` (visited on 03/15/2023).

[65] M. Hanine, O. Boutkhoum, A. Tikniouine, and T. Agouti, "Application of an integrated multi-criteria decision making AHP-TOPSIS methodology for ETL software selection," *SpringerPlus*, vol. 5, no. 1, p. 263, Mar. 2, 2016, ISSN: 2193-1801. DOI: `10.1186/s40064-016-1888-z`. [Online]. Available: `https://doi.org/10.1186/s40064-016-1888-z` (visited on 03/16/2023).

[66] MongoDB. "Aggregation operations — MongoDB manual," Aggregation Operations. (2023), [Online]. Available: `https://www.mongodb.com/docs/manual/aggregation/` (visited on 03/23/2023).

[67] MongoDB. "Db.collection.findOneAndUpdate() — MongoDB manual." (2023), [Online]. Available: https://www.mongodb.com/docs/manual/reference/method/db.collection.findOneAndUpdate/ (visited on 03/21/2023).

[68] MongoDB. "Query and projection operators — MongoDB manual." (2023), [Online]. Available: https://www.mongodb.com/docs/manual/reference/operator/query/#std-label-query-selectors (visited on 02/21/2023).

[69] M. UI. "Mantis free - react admin dashboard template," MUI Store. (2023), [Online]. Available: https://mui.com/store/items/mantis-free-react-admin-dashboard-template/ (visited on 01/02/2023).

[70] J. H. Saltzer, "The origin of the "MIT license"," *IEEE Annals of the History of Computing*, vol. 42, no. 4, pp. 94–98, Oct. 2020, Conference Name: IEEE Annals of the History of Computing, ISSN: 1934-1547. DOI: 10.1109/MAHC.2020.3020234.

[71] Mantis. "Mantis-free-react-admin-template/LICENSE at main · codedthemes/mantis-free-react-admin-template," mantis-free-react-admin-template/LICENSE. (Feb. 28, 2022), [Online]. Available: https://github.com/codedthemes/mantis-free-react-admin-template/blob/main/LICENSE (visited on 03/20/2023).

[72] A. Design, *Ant design*, original-date: 2015-04-24T15:37:24Z, Nov. 28, 2018. [Online]. Available: https://github.com/ant-design/ant-design/blob/822dec20a44d099e599747899d113
LICENSE (visited on 03/20/2023).

[73] M. UI. "Pricing - MUI." (2023), [Online]. Available: https://mui.com/pricing/ (visited on 03/20/2023).

[74] A. Design. "Ant design - the world's second most popular react UI framework." (2023), [Online]. Available: https://ant.design/ (visited on 01/02/2023).

[75] M. UI. "MUI: The react component library you always wanted." (2023), [Online]. Available: https://mui.com/ (visited on 01/02/2023).

[76] Xunzel. "SOLARX™-8," Xunzel. (2023), [Online]. Available: https://www.xunzel.com/inicio/solarx-8/ (visited on 04/10/2023).

[77] F. Robotics. "Welcome!" Freedom Robotics. (Sep. 2022), [Online]. Available: https://docs.freedomrobotics.ai/docs/alerts-generate-a-smart-alert (visited on 04/11/2023).

[78] R. Robotics. "PA-AMR | rapyuta robotics." (2023), [Online]. Available: https://www.rapyuta-robotics.com/solutions-pa-amr/ (visited on 05/21/2023).

[79] W. Robotics. "WAKU sense - tool for robot operators in logistics & production." (2023), [Online]. Available: https://www.waku-sense.com/en/?utm_source=google_ads&utm_medium=cpc&utm_campaign=analytics&gclid=CjwKCAjwitShBhA6EiwAq3RqA2_DkhyU4ARP7i0kE6zfyCMu4F6ufaC3n-umv0YvegczCJaggLCpiBoCdDwQAvD_BwE (visited on 04/21/2023).

[80] F. Robotics. "API reference," Freedom Robotics. (Dec. 2023), [Online]. Available: https://docs.freedomrobotics.ai/reference/rest-api (visited on 02/22/2023).

[81] G. S. Funchal, "Development of security mechanisms for a multi-agent cyber-physical conveyor system using machine learning," Accepted: 2020-12-21T10:14:25Z, masterThesis, 2020. [Online]. Available: https://bibliotecadigital.ipb.pt/handle/10198/22980?mode=full (visited on 04/14/2023).

[82] J. Prinsloo, S. Sinha, and B. von Solms, "A review of industry 4.0 manufacturing process security risks," *Applied Sciences*, vol. 9, no. 23, p. 5105, Jan. 2019, Number: 23 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2076-3417. DOI: 10.3390/app9235105. [Online]. Available: https://www.mdpi.com/2076-3417/9/23/5105 (visited on 04/14/2023).

[83] Z. M. Çınar, A. Abdussalam Nuhu, Q. Zeeshan, O. Korhan, M. Asmael, and B. Safaei, "Machine learning in predictive maintenance towards sustainable smart manufacturing in industry 4.0," *Sustainability*, vol. 12, no. 19, p. 8211, Jan. 2020, Number: 19 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2071-1050. DOI: 10.3390/su12198211. [Online]. Available: https://www.mdpi.com/2071-1050/12/19/8211 (visited on 04/14/2023).

[84] G. Lee, W. Saad, and M. Bennis, "Online optimization for UAV-assisted distributed fog computing in smart factories of industry 4.0," in *2018 IEEE Global Communications Conference (GLOBECOM)*, ISSN: 2576-6813, Dec. 2018, pp. 1–6. DOI: 10.1109/GLOCOM.2018.8647441.

[85] A. França. "Gmail screenshot for layout example." (Mar. 21, 2023), [Online]. Available: https://mail.google.com/ (visited on 03/21/2023).

[86] A. França. "Outlook screenshot for layout example." (Mar. 21, 2023), [Online]. Available: https://outlook.office365.com/ (visited on 03/21/2023).

[87] A. França. "YouTube screenshot for layout example." (Mar. 21, 2023), [Online]. Available: https://www.youtube.com/ (visited on 03/21/2023).

# Appendix A

# Interview schedule for the dashboard concepts

The document presented in this appendix is an interview schedule, which aims to elicit the current requirements and expectations for the system proposed in this thesis. The application of the script was oral to encourage communication.

# Initial dashboard concepts

This form aims to collect information about the visualization and feature priorities that should be developed for the dashboard.

*Obrigatório

1. What information/indicators do you believe should be presented to the system administrator? *

2. The administrator must have what action capabilities with the dashboard (e.g., remotely cancel the current action of the robot) *

3. Could you draw what you believe the layout of resources should look like?

   Arquivos enviados:

# Appendix B

# Examples of application layouts

This appendix exposes some layouts of established web applications used to inspire the application structure developed for this work. These being of email applications in figures B.1 and B.2, and of entertainment in figure B.3.

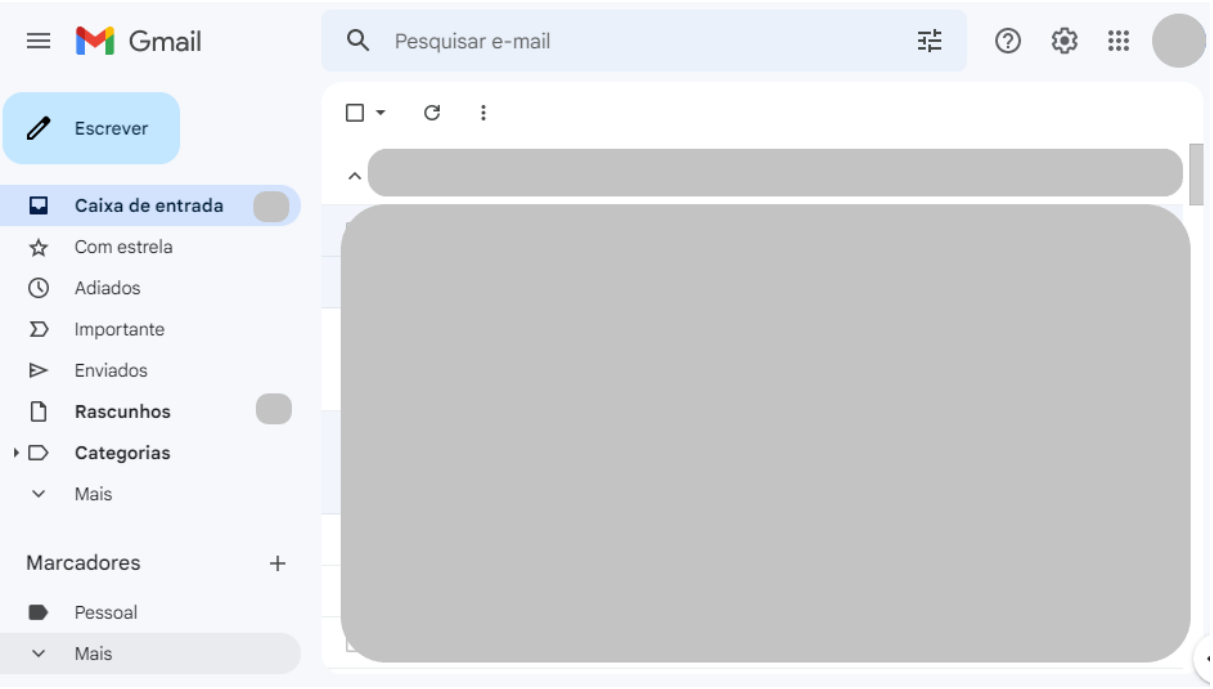



Figure B.1: Screenshot of Gmail for layout example [85]

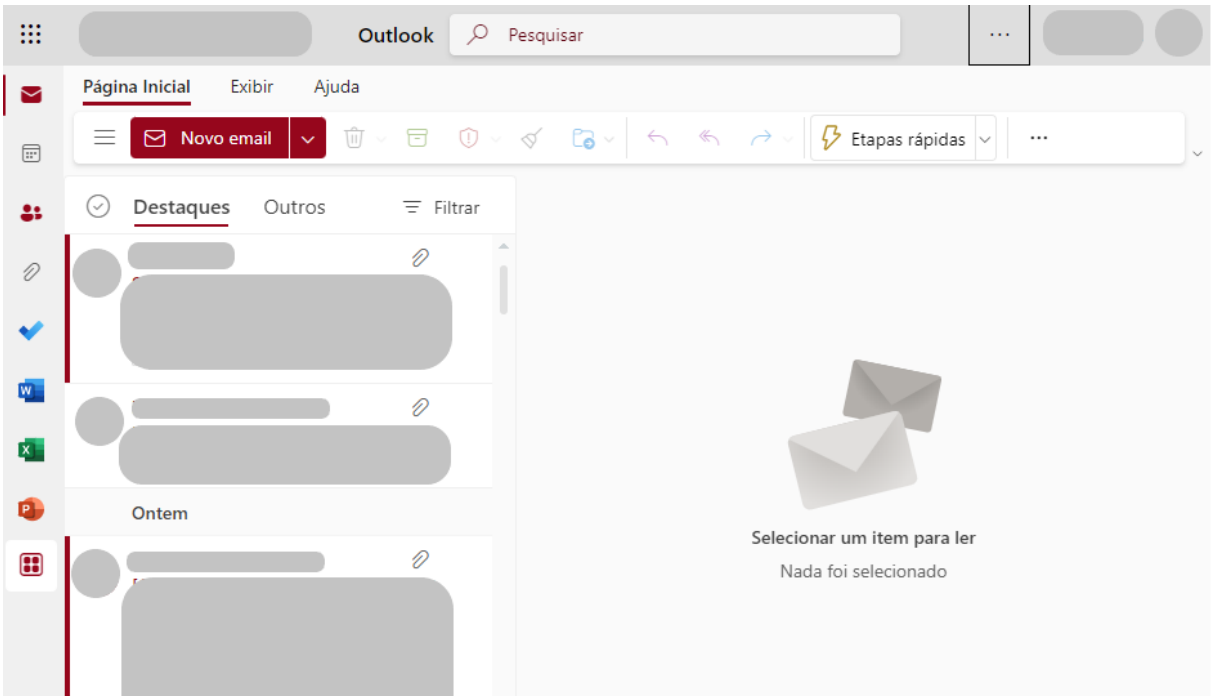



Figure B.2: Screenshot of Outlook for layout example [86]

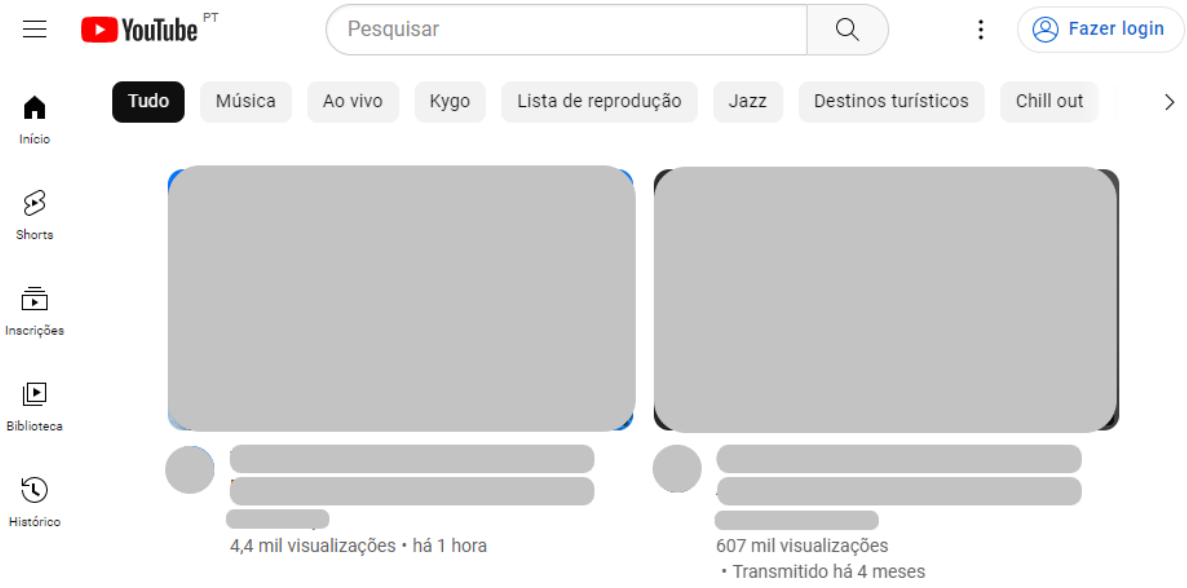



Figure B.3: Screenshot of Youtube for layout example [87]

# Appendix C

# AHP-TOPSIS analyses

This appendix aims to demonstrate the matrices used for the application Multiple Criteria Decision Making (MCDM) Analytic Hierarchy Process (AHP) Technique for Order Preference by Similarity to Ideal Solution (TOPSIS).

## C.1 Remote Unit Database

This section demonstrates the matrices of weights and results obtained for selecting the Database Management System (DBMS) software of the Remote Unit. The criteria, sub-criteria and alternatives are listed below:

**Criteria:**

- **C1** Functionality
  - **C11** Compatibility
  - **C12** Compatibility
- **C2** Usability
  - **C21** Ease integration
  - **C22** Recoverability
  - **C23** Cost
- **C3** Limits
  - **C31** Storage
  - **C32** Requests
  - **C33** Compute Runtime

**Alternatives:**

- **S1** Atlas
- **S2** Community

The tables C.1, C.2, C.3 and C.4 denote the weights attributed to the criteria. The table C.5 indicates the normalized weights already applied to the presented alternatives. The C.6 table shows the classification and evaluation of the available alternatives.

Table C.1: Criteria comparison matrix for Remote Unit Database

| | C1 | C2 | C3 | Weights |
|---|---|---|---|---|
| C1 | 1 | 1/2 | 2 | 0.266 |
| C2 | 2 | 1 | 1/5 | 0.242 |
| C3 | 1/2 | 5 | 1 | 0.492 |

Table C.2: Sub-criteria C1 comparison matrix for Remote Unit Database

| | C11 | C12 | Weights |
|---|---|---|---|
| C11 | 1 | 2 | 0.667 |
| C12 | 1/2 | 1 | 0.333 |

Table C.3: Sub-criteria C2 comparison matrix for Remote Unit Database

| | C21 | C22 | C23 | Weights |
|---|---|---|---|---|
| C21 | 1 | 2 | 1/4 | 0.217 |
| C22 | 1/2 | 1 | 1/5 | 0.114 |
| C23 | 4 | 5 | 1 | 0.669 |

Table C.4: Sub-criteria C3 comparison matrix for Remote Unit Database

| | C31 | C32 | C33 | Weights |
|---|---|---|---|---|
| C31 | 1 | 4 | 4 | 0.539 |
| C32 | 1/4 | 1 | 5 | 0.374 |
| C33 | 1/4 | 1/5 | 1 | 0.087 |

Table C.5: Normalised weights for Remote Unit Database

| | C11 | C12 | C21 | C22 | C23 | C31 | C32 | C33 |
|---|---|---|---|---|---|---|---|---|
| | + | + | + | + | - | + | + | + |
| S1 | 1.062 | 0.44 | 0.265 | 0.168 | 1.296 | 0.53 | 0.368 | 0.086 |
| S2 | 0.708 | 0.528 | 0.265 | 0.112 | 0.324 | 1.855 | 1.288 | 0.301 |
| V+ | 1.062 | 0.528 | 0.265 | 0.168 | 0.324 | 1.855 | 1.288 | 0.301 |
| V- | 0.708 | 0.44 | 0.265 | 0.112 | 1.296 | 0.53 | 0.368 | 0.086 |

Table C.6: Evaluation and ranking of alternatives for Remote Unit Database

| | Si+ | Si- | Pi | Rank |
|---|---|---|---|---|
| S1 | 1.898 | 0.358402 | 0.158868 | 2 |
| S2 | 0.358 | 1.897572 | 0.841132 | 1 |

## C.2 Dashboard genre

This section demonstrates the matrices of weights and results obtained for selecting the dashboard genre. The criteria, sub-criteria and alternatives are listed below:

**Criteria:**

- **C1** Screenspace
  - **C11** Screenfit
  - **C12** Overflow
  - **C13** Detail on demand
  - **C14** Parameterization
  - **C15** Multiple pages
- **C2** Structure
  - **C21** Single page
  - **C22** Parallel structure
  - **C23** Hierarchical structures
  - **C24** Open structures
- **C3** Interaction
  - **C31** Exploration
  - **C32** Navigation
  - **C33** Drilldown
  - **C34** Personalization

**Alternatives:**

- **S1** Static
- **S2** Analytic
- **S3** Magazine
- **S4** Infographic
- **S5** Repository
- **S6** Embedded Mini

The tables C.7, C.8, C.9 and C.10 denote the weights attributed to the criteria. The table C.11 indicates the normalized weights already applied to the presented alternatives.

The C.12 table shows the classification and evaluation of the available alternatives.

Table C.7: Criteria comparison matrix for dashboard genre

| | C1 | C2 | C3 | Weights |
|---|---|---|---|---|
| C1 | 1 | 2 | 1/2 | 0.333 |
| C2 | 1/2 | 1 | 1/2 | 0.190 |
| C3 | 2 | 2 | 1 | 0.476 |

Table C.8: Sub-criteria C1 comparison matrix for dashboard genre

| | C11 | C12 | C13 | C14 | C15 | Weights |
|---|---|---|---|---|---|---|
| C11 | 1 | 4 | 0.5 | 1 | 1 | 0.223 |
| C12 | 1/4 | 1 | 4 | 2 | 1 | 0.245 |
| C13 | 2 | 1/4 | 1 | 3 | 2 | 0.245 |
| C14 | 1 | 1/2 | 1/3 | 1 | 1/3 | 0.094 |
| C15 | 1 | 1 | 1/2 | 3 | 1 | 0.193 |

Table C.9: Sub-criteria C2 comparison matrix for dashboard genre

| | C21 | C22 | C23 | C24 | Weights |
|---|---|---|---|---|---|
| C21 | 1 | 2 | 2 | 1/3 | 0.269 |
| C22 | 1/2 | 1 | 2 | 1/2 | 0.202 |
| C23 | 1/2 | 1/2 | 1 | 2 | 0.202 |
| C24 | 3 | 2 | 1/2 | 1 | 0.328 |

Table C.10: Sub-criteria C3 comparison matrix for dashboard genre

| | C31 | C32 | C33 | C34 | Weights |
|---|---|---|---|---|---|
| C31 | 1 | 2 | 3 | 3 | 0.412 |
| C32 | 1/2 | 1 | 3 | 1 | 0.252 |
| C33 | 1/3 | 1/3 | 1 | 3 | 0.214 |
| C34 | 1/3 | 1 | 1/3 | 1 | 0.122 |

Table C.11: Normalised weights for dashboard genre

| | C11 | C12 | C13 | C14 | C15 | C21 | C22 |
|---|---|---|---|---|---|---|---|
| | + | + | + | + | + | + | + |
| D1 | 0.594 | 0.082 | 0.082 | 0.031 | 0.064 | 0.41 | 0.038 |
| D2 | 0.371 | 0.082 | 0.245 | 0.157 | 0.322 | 0.154 | 0.115 |
| D3 | 0.074 | 0.653 | 0.245 | 0.094 | 0.193 | 0.41 | 0.038 |
| D4 | 0.074 | 0.653 | 0.082 | 0.031 | 0.064 | 0.41 | 0.038 |
| D5 | 0.149 | 0.653 | 0.082 | 0.157 | 0.515 | 0.154 | 0.115 |
| D6 | 0.149 | 0.163 | 0.163 | 0.157 | 0.064 | 0.051 | 0.115 |
| V+ | 0.594 | 0.653 | 0.245 | 0.157 | 0.515 | 0.41 | 0.115 |
| V- | 0.074 | 0.082 | 0.082 | 0.031 | 0.064 | 0.051 | 0.038 |

| | C23 | C24 | C31 | C32 | C33 | C34 |
|---|---|---|---|---|---|---|
| | + | + | + | + | + | + |
| D1 | 0.038 | 0.062 | 0.196 | 0.12 | 0.102 | 0.058 |
| D2 | 0.115 | 0.187 | 1.57 | 0.96 | 0.814 | 0.465 |
| D3 | 0.038 | 0.062 | 0.589 | 0.36 | 0.305 | 0.174 |
| D4 | 0.038 | 0.062 | 0.196 | 0.12 | 0.102 | 0.058 |
| D5 | 0.115 | 0.187 | 1.57 | 0.24 | 0.814 | 0.116 |
| D6 | 0.115 | 0.312 | 0.589 | 0.96 | 0.305 | 0.174 |
| V+ | 0.115 | 0.312 | 1.57 | 0.96 | 0.814 | 0.465 |
| V- | 0.038 | 0.062 | 0.196 | 0.12 | 0.102 | 0.058 |

Table C.12: Evaluation and ranking of alternatives for dashboard genre

| | Si+ | Si- | Pi | Rank |
|---|---|---|---|---|
| D1 | 1.978 | 0.631 | 0.242 | 6 |
| D2 | 0.704 | 1.871 | 0.727 | 1 |
| D3 | 1.455 | 0.877 | 0.376 | 4 |
| D4 | 1.964 | 0.675 | 0.256 | 5 |
| D5 | 0.973 | 1.733 | 0.640 | 2 |
| D6 | 1.444 | 1.011 | 0.412 | 3 |

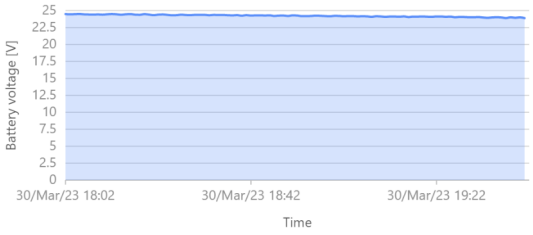
# Appendix D

# Dashboard content

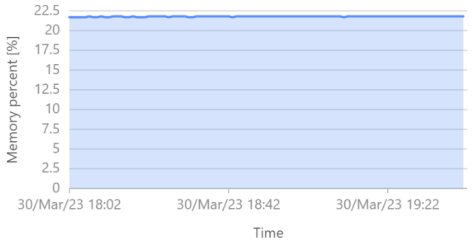
This appendix demonstrates examples of the visualisations available in the dashboard application developed.

# D.1 Area charts

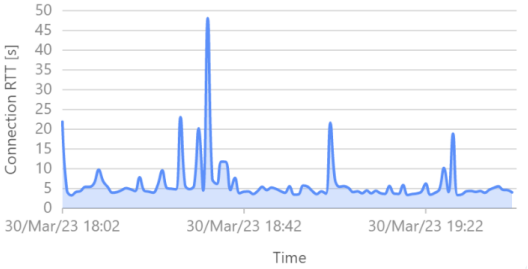

(a) Battery percentage area chart



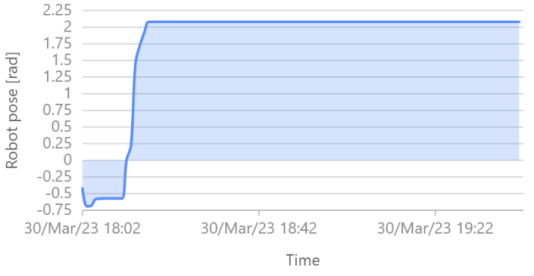

(b) Battery voltage area chart




(c) Computer memory usage area chart




(d) Connection RTT area chart




(e) Robot yaw area chart

Figure D.1: Area charts

## D.2 Graph charts

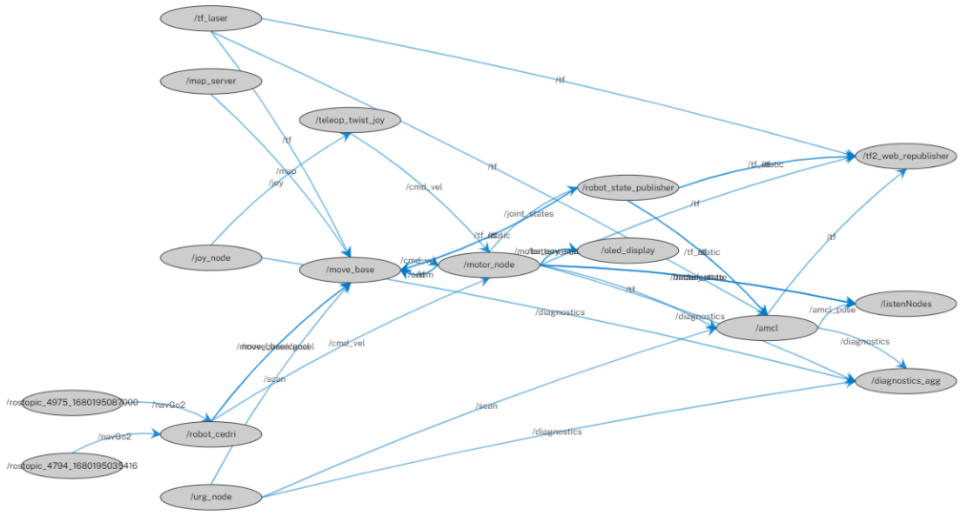
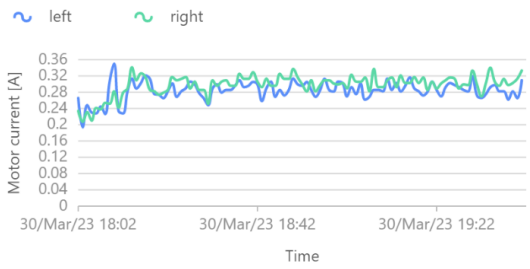


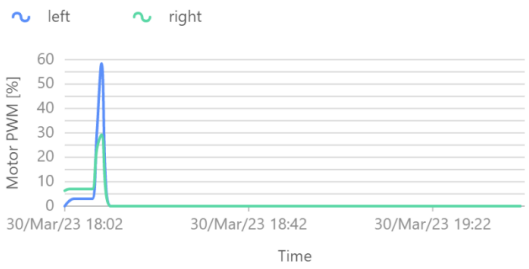

Figure D.2: Nodes graph

## D.3 Line charts

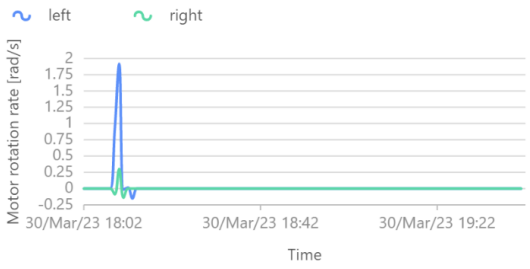
In the figures D.3a, D.3b and D.3c, the label left and right indents denote the motor in the platform the chart references.In the figure D.3d the legend denotes the CPU core.



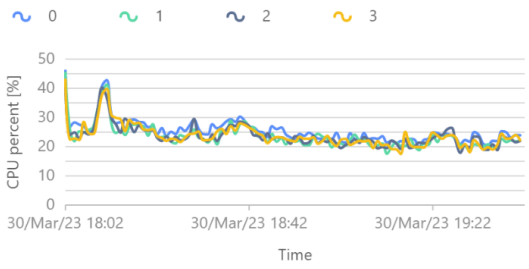

(a) Motor current line chart




(b) Motor PWM line chart



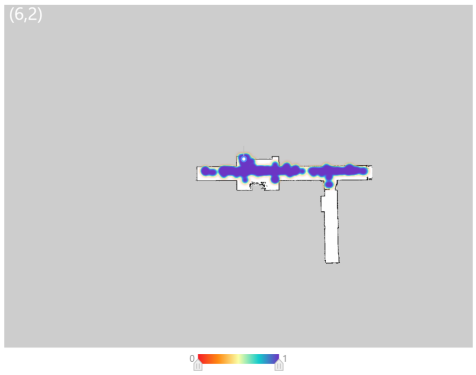

(c) Motor rotation rate line chart




(d) Computer CPU usage line chart

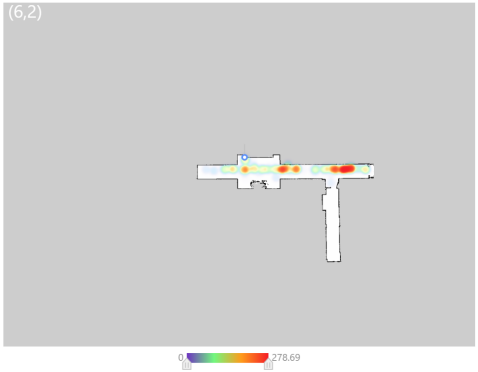
Figure D.3: Line charts

## D.4 Maps charts

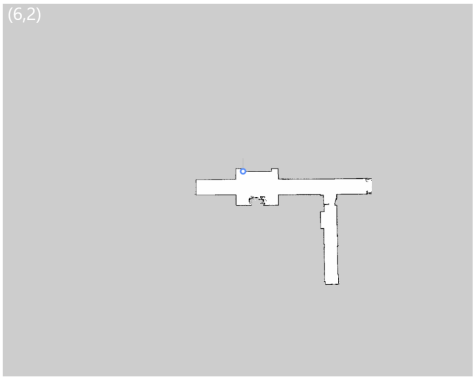



(a) Connection heatmap

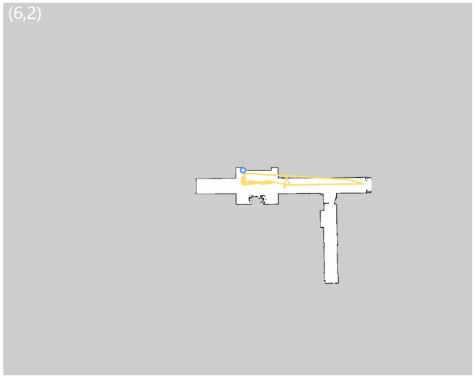

(b) Position heatmap




(c) RTT heatmap




(d) Position map

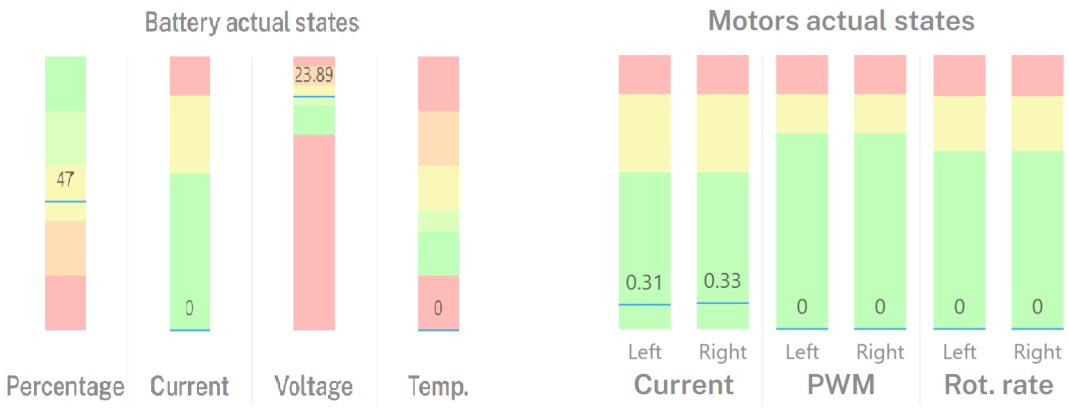



(e) Breadcrumbs map

Figure D.4: Maps charts

## D.5 Progress bar charts

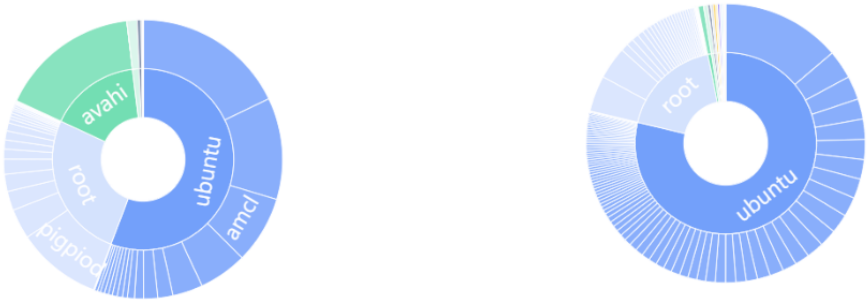
In the picture D.5b, the label left and right indents denote the motor in the platform the chart references.




(a) Battery progress bar charts (b) Motor progress bar charts

Figure D.5: Progress bar charts

## D.6 Sunburst charts




(a) Computer CPU usage sunburst chart (b) Computer memory usage sunburst chart

Figure D.6: Sunburst charts