



Virtualization and Monitoring of a Modular and Self-organized Multi-agent Conveyor System

Victória Clarissa de Abreu Melo

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master Degree in Industrial Engineering, in the scope of the double diploma programme with the Federal University of Technology – Paraná.

Work oriented by:

Prof. Dr. Paulo Jorge Pinto Leitão

Prof. Dr. José Fernando Lopes Barbosa

Prof. Dr. Marcos Vallim

Bragança

2019-2020



Virtualization and Monitoring of a Modular and Self-organized Multi-agent Conveyor System

Victória Clarissa de Abreu Melo

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master Degree in Industrial Engineering, in the scope of the double diploma programme with the Federal University of Technology – Paraná.

Work oriented by:

Prof. Dr. Paulo Jorge Pinto Leitão

Prof. Dr. José Fernando Lopes Barbosa

Prof. Dr. Marcos Vallim

Bragança

2019-2020

Dedication

I dedicate to everyone who helped me on this journey, especially my family and Gustavo's family.

Acknowledgement

The completion of this work marks the end of an important stage in my life. It is with great satisfaction that I thank here everyone who contributed to the development of this work.

First of all, I would like to thank my supervisors at the Polytechnic Institute of Bragança (IPB), Professor PhD Paulo Leitão and Professor PhD José Barbosa, for their mentoring, knowledge transmitted, opportunity, motivation and advice. It was an experience for which I will be forever grateful.

To my supervisor at UTFPR, Professor PhD Marcos Vallim, for the opportunity to participate in the Double Degree program.

To the organizers of the Double Degree program for supporting the research and collaboration between IPB and UTFPR and allowing the students to have this amazing experience.

To my friends and lab colleagues, more especially Jonas Queiroz, Filipe Alves, Flávia Pires and Lucas Sakurada for all the moments spent together, sharing experiences and knowledge.

Also, I want to thank my boyfriend Gustavo Funchal for all the moments that we have shared during our graduation, for all the support, affection and for making my life better.

Finally, to my family, especially to my parents who gave me all the support, encouragement and advice to achieve my goals. Nothing could be done without them.

Abstract

The digital transformation that is taking place worldwide in the context of Industry 4.0 stimulates the emergence of new concepts and technologies and is reshaping the manufacturing world. As a result, the Digital Twin concept is gaining more and more momentum. In this sense, the creation of a virtual copy of the physical system provides a connection between the real and virtual systems to collect, analyze and simulate data in the virtual model to improve the performance of the real system.

This work presents the development of virtualization of a modular and self-organized transport system, based on the use of Cyber-Physical Systems (CPS) and Multi-Agent Systems (MAS), to ensure the monitoring of its operation and allow the early detection of failures through real-time analysis of data collected from the physical system. For this purpose, Internet of Things (IoT) tools and technologies were used, such as Node-RED and the Message Queuing Telemetry Transport (MQTT) communication protocol, and the storage of collected data and cyber-security issues related to the use of IoT technologies during the data collection, suggesting the implementation of solutions to avoid possible threats, such as user authentication and data encryption.

Among the results obtained, the control panel displays system data in real-time, in addition to providing statistical information, and supports dynamic monitoring of the transport system's condition operation, generating alerts through the implementation of control rules.

Keywords: Digital Twin; cyber-physical systems; multi-agent systems; virtualization; early failure detection.

Resumo

A transformação digital que está ocorrendo em todo o mundo no contexto da Indústria 4.0 estimula o surgimento de novos conceitos e tecnologias e está remodelando o mundo da manufatura. Como resultado, o conceito Digital Twin está ganhando cada vez mais força. Nesse sentido, a criação de uma cópia virtual do sistema físico fornece uma conexão entre os sistemas real e virtual para coletar, analisar e simular dados no modelo virtual para melhorar o desempenho do sistema real.

Este trabalho apresenta o desenvolvimento da virtualização de um sistema de transporte modular e auto-organizado, com base no uso de CPS e MAS, para garantir o monitoramento de sua operação e permitir a detecção antecipada de falhas através da análise em tempo real dos dados coletados do sistema físico. Para esse fim, foram utilizadas ferramentas e tecnologias IoT, como Node-RED e o protocolo de comunicação MQTT, além do armazenamento de dados coletados e problemas de ciber-segurança relacionados ao uso de tecnologias IoT durante a coleta de dados, sugerindo a implementação de soluções para evitar possíveis ameaças, como autenticação do usuário e criptografia de dados.

Entre os resultados obtidos, o painel de controle exibe os dados do sistema em tempo real, além de fornecer informações estatísticas e suportar o monitoramento dinâmico da condição de funcionamento do sistema de transporte, gerando alertas por meio da implementação das regras de controle.

Palavras-chave: Digital Twin; sistemas ciber-físicos; sistemas multi-agentes; virtualização; detecção antecipada de falhas.

Contents

Acknowledgement	vii
Abstract	ix
Resumo	xi
Acronyms	xix
1 Introduction	1
1.1 Motivation and Framework	1
1.2 Objectives	3
1.3 Document Structure	3
2 Theoretical Background	5
2.1 Industry 4.0 and Smart Factories	5
2.2 Cyber-Physical Systems	7
2.3 Multi-Agent Systems	8
2.4 Digital twin	9
2.5 Communication Protocols	10
2.5.1 FIPA-ACL	11
2.5.2 MQTT	12
2.6 Maintenance Strategies and Techniques	13
2.7 Security in applications using IoT	15

3	Case Study	17
3.1	Description of the Case Study	17
3.2	Self-Organization Capabilities	19
3.3	Improvements Applied in the System Functionalities	20
3.3.1	Swap for conveyor with Token 1	20
3.3.2	New Devices Installed	21
4	Development of the Digital Twin	25
4.1	Virtualization and Connectivity	25
4.1.1	Motor State and Physical Sequence of the Conveyors	27
4.1.2	Sensors States and the Total Amount of Pieces Transported	32
4.1.3	Amount of Pieces Transported by Each Conveyor	34
4.1.4	Vibration on the Three Axes	35
4.1.5	Total Motor Operating Time	37
4.1.6	Battery Voltage, Power and Current	39
4.2	Real-time Monitoring	40
4.2.1	Timestamps in the transfer of parts	42
4.2.2	Vibration	44
4.2.3	Motor Time	44
4.2.4	Battery Voltage and Current	45
4.2.5	Swap	46
4.3	Security Weaknesses	46
5	Online Monitoring	51
5.1	System Virtualization in Node-RED Dashboard	51
5.2	Database Storage	53
5.3	Early Failure Detection	54
5.4	Automatic Alert by Email	58
6	Conclusion and Future Work	61

List of Tables

2.1	Parameters of the ACL message [17].	12
2.2	Levels of Quality of Service using MQTT.	13
2.3	Parameters for the client and broker connection [28].	14
2.4	Parameters to the client subscribe or publish in a broker (adapted from [28]).	14
5.1	Sample of data referring to the current value being stored in the database.	54
5.2	Mean and standard deviation values calculated for the application of Rule 1 for failure prediction.	55

List of Figures

2.1	The four industrial revolutions [5].	6
2.2	Structure of a multi-agent system [16].	8
2.3	A Digital Twin architecture [25].	11
2.4	Publish-subscribe model using MQTT (adapted from [29]).	13
3.1	Cyber-physical conveyor system.	18
3.2	Installing the display and the current and accelerometer sensors in the module.	22
3.3	Connection diagram between the display and the I2C interface of Raspberry Pi (adapted from [40]).	23
4.1	Main graphical interface of Node-RED.	26
4.2	Virtualization approach for the conveyor system.	27
4.3	Node-RED flow for the motor state and conveyor physical sequence monitoring.	28
4.4	Connecting the node MQTT input to the MQTT broker.	28
4.5	Naming of the conveyors in letters to facilitate identification.	30
4.6	LED node configuration.	31
4.7	Node-RED flow to monitor sensor states and the total amount of pieces transported.	32
4.8	Node-RED flow to monitor the number of pieces transported by each conveyor module.	34
4.9	MySQL node configuration.	36

4.10	Node-RED flow for the vibration on the three axes monitoring.	36
4.11	Node-RED flow for the total motor operating time monitoring.	38
4.12	Node-RED flow to monitor current, power and battery voltage.	39
4.13	Rules to detect outliers and trends in the conveyor operation behaviour. . .	41
4.14	Node-RED flow for the time in the transfer of parts monitoring.	43
4.15	Node-RED flow for the vibration monitoring adding the rules 1 and 2. . . .	44
4.16	Node-RED flow for the motor time monitoring adding the Rule 2.	45
4.17	Node-RED flow for the battery voltage and current monitoring adding the Rules 1 and 2.	45
4.18	Node-RED flow for the swap monitoring.	46
4.19	Security weakness in the case study.	47
4.20	Login screen to access the Node-RED admin page.	48
4.21	Login screen to access the Node-RED dashboard.	48
5.1	Dashboard for the virtual model of the conveyor system supporting the real-time monitoring.	52
5.2	Dashboard for the analysis of vibration in the three axes separately.	53
5.3	Dashboard for the monitoring of the motor operating time, battery level and power.	53
5.4	Current notification applying Rule 1 to detect failures in advance.	55
5.5	Vibration notification applying Rule 1 to detect failures in advance.	56
5.6	Notification from Rule 1 to inform a transport time of a piece higher than usual.	56
5.7	First battery notification concerning its level.	57
5.8	Second battery notification concerning its level.	57
5.9	Increasing trend detected by applying Rule 2 for current monitoring.	58
5.10	Swap notification.	58
5.11	Structure of the email received when a notification occurs.	59

Acronyms

ACL Agent Communication Language.

CA Certification Authorities.

CPS Cyber-Physical Systems.

DF Directory Facilitator.

FIPA Foundation for Intelligent Physical Agents.

GPIO General Purpose Input Output.

HTTP HyperText Transfer Protocol.

HTTPS HyperText Transfer Protocol Secure.

I2C Inter-Integrated Circuit.

IIoT Industrial Internet of Things.

IoT Internet of Things.

JADE Java Agent DEvelopment Framework.

LED Light-Emitting Diode.

MAS Multi-Agent Systems.

MQTT Message Queuing Telemetry Transport.

QoS Quality of Service.

SSL Secure Sockets Layer.

TLS Transport Layer security.

UI User Interface.

Chapter 1

Introduction

This work is framed in the context of Industry 4.0, focusing on the new concepts of “Smart Factories”, “Digital Twin”, “Cyber-Physical Systems” and “Multi-Agent Systems”. These concepts are directly related to changing consumption habits, life and behavior, both at a personal and business level.

1.1 Motivation and Framework

Computer science and the internet have become natural in business and home environments. This factor made possible a promising paradigm that is the integration of several technologies and communication solutions in order to connect objects. In this way, a large number of heterogeneous objects are connected and able to interact to offer an objective. Such objects can consist of sensors and actuators in which they are considered things capable of providing information about the environments and controlling them remotely. The objective of IoT is to provide an advanced way of communication between different systems and devices, facilitating the interaction with applications that will be used by users in the various activities of daily life.

The 4th industrial revolution, also known as Industry 4.0, is characterized by the digitization of traditional factories to allow the companies to be more competitive facing the strong pressures imposed by global markets and demanding customers, reflected in

terms of customization, quality and diversity of products, and the fast response to the demand [1]. The fundamentals of Industry 4.0 defines that through the interconnection of machines, production systems and equipment, they will give companies the ability to create smart grids along the entire value chain, and thus control and command the processes of production independently. Industry 4.0 aggregates technologies such as Big Data, Advanced Analytics, CPS, IoT, Cloud Computing and Security, which create conditions for the creation of smart factories [2]. The vision of future production contains modular and efficient manufacturing systems and characterizes scenarios in which products control their own manufacturing process [3].

Virtualization, in general, is the creation of a virtual copy of the physical system, establishing a connection, through data collection, between the real and virtual systems, which will affect the simulation of the model. In this context, virtualization, and more specifically the concept of Digital Twin, assumes crucial importance in the development of industrial CPS that are more flexible, optimized, adaptive and reconfigurable. One of the main capabilities of this digital copy is to monitor the processes of the physical world [4]. The Digital Twin should be assessed against tangible IoT benefits such as improvements in yield, quality and efficiency, as well as cost reduction and prevention of issues. The digitization allows for more efficient processes, reduction in energy consumption, minimization of waste, with highly customized products adapted to the customer. In this context, the capture, storage and analysis of data will become more and more significant.

In this sense, this work addresses the development of a Digital Twin of a self-organized transport system based on MAS composed of modules of cyber-physical components. This virtualization allows the monitoring of its operation from the data collected from the physical system in real-time and, performing the analysis of this data, it also enables the detection of failures in advance.

1.2 Objectives

The main objective of this work consists of improving the functionalities of the modular and self-organized transport system of automatic conveyors FischerTechniks using MAS technology, through the development of a digital twin that supports the monitoring of its operation and the detection of failures in advance based on the collection of data using IoT technologies and their subsequent analysis, both performed in real-time. In order to achieve this objective, secondary objectives have been established:

- Study of MAS and the deployment of agents on low-cost control platforms, such as the Raspberry PI;
- Study of the existing solution for the modular and self-organized transport system of automatic conveyors FischerTechniks using MAS;
- Implementation of improvements applied in the system functionalities namely the optimization of the self-organization capacity, including a new swap condition for the modules, and the installation of more devices to ensure that more data can be collected from system to be used in its operation monitoring;
- Definition of the Digital Twin architecture for the system under study, with the analysis of tools and technologies IoT to be used for the collection of data from the real system;
- Development of the failures monitoring system in advance using techniques for real-time analysis of the collected data.

1.3 Document Structure

This document is organized in 6 chapters, beginning with the present chapter where the proposal, contextualization and the objectives of the work were presented.

The Chapter 2, entitled “Theoretical Background”, presents a review of the fundamental contents for the development and understanding of the work.

The Chapter 3, entitled “Case Study”, describes the modular and self-organized transport system of automatic conveyors FischerTechniks using MAS and the improvements applied in the system functionalities.

The Chapter 4, entitled “Development of the Digital Twin”, presents the architecture adopted for the development of the digital twin, introducing the tools and technologies used for the connection between the real and digital system, the techniques used for real-time analysis of the collected data to detect failures in advance and some solutions for the security weaknesses found in the virtualization architecture.

The Chapter 5, entitled “Online Monitoring”, presents the developed digital system, the storage of the data collected in the database and the alarm system for early detection of failures as well as the automatic sending of alerts by email to assist in the monitoring.

Finally, the last chapter, entitled “Conclusions and Future Work”, resumes the work with the conclusions and points out future work.

Chapter 2

Theoretical Background

This chapter aims to guide the reader through a literature review on concepts related to the context of Industry 4.0. Thus, the concept of Smart Factories is presented, followed by CPS, MAS and Digital Twin. Then, it will be provided a brief explanation of some communication protocols. Finally, some maintenance strategies are presented and the concept of security in applications using IoT is also addressed.

2.1 Industry 4.0 and Smart Factories

Industry 4.0 is considered the fourth revolution in the manufacturing industry. Since the beginning of industrialization, technological changes have led to changes in the paradigm known as *industrial revolutions* [3]:

- First industrial revolution, when mechanization and steam power changed the whole concept of manufacturing;
- Second industrial revolution, when mass-production assembly lines and electrical energy took place and enabled a giant step in production efficiency;
- Third industrial revolution, when automation, computers and robots were added to production.

Now, based on an advanced digitization in the factories, the combination of Internet technologies in the field of “intelligent” machines and products brings a new fundamental paradigm shift in industrial production. Figure 2.1 illustrates the changes that have occurred in the industry.

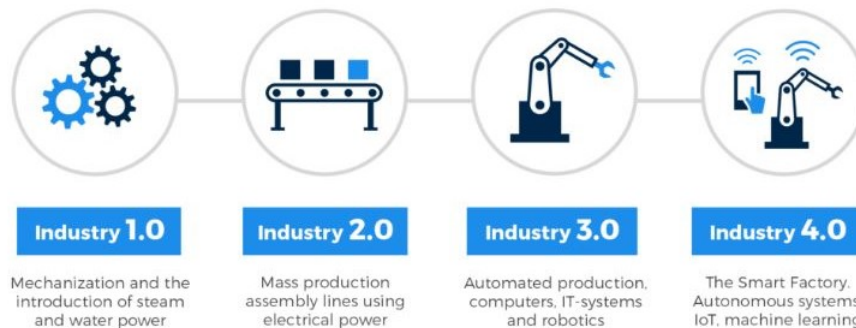


Figure 2.1: The four industrial revolutions [5].

The smart factory is the own factory in the Industry 4.0 sector, which is characterized by complete automation of production processes thanks to the implementation of IoT technology. The term was created from the perception of a series of technological advances that affect not only the industries but the entire market. The original IoT, in industrial settings, has evolved to give way to Industrial Internet of Things (IIoT), an internet of things adapted especially to the needs and requirements of the industrial sector.

Although it is still the first phase of the expansion of IIoT, this process has already started and will be “unstoppable”, since it allows much more efficient operation in all possible aspects. Thus, in the coming years, the smart factory will become popular and extend to all fields of industry. This will be done gradually, but without pause, at the same time that it will be adaptable to the specific needs and characteristics of each of the existing industrial sectors. In all cases, Industry 4.0 is where the automation of processes and the digitization of tasks will predominate [6].

In Industry 4.0, traditional production facilities are converted into smart factories, which in turn make smart products. The adoption of the smart factory can be a game-changing event that can transform the interaction of engineered systems just as the internet transformed the way people interact with information [7].

Exchange data and information between different devices and parties in real-time is the key element of smart factories. Such data could represent production status, energy consumption behavior, material movements, customer orders and feedback or suppliers' data. The next generation of smart factories, therefore, will have to be able to adapt, almost in real-time, to the continuously changing market demands, technology options and regulations [8].

The design of smart factories will establish that, to optimize this method, the entire process of making a product available must be integrated with digital processes that register, change and connect each step. Automations, IoT and Big Data are just a few examples of technologies included in the implementation of the Smart Factories concept. Therefore, technology is a means to automate what is already done in factories today, but in a faster, more flexible and more reliable way.

2.2 Cyber-Physical Systems

CPS is a complex engineering system that integrates physical, computation and networking, and communication processes [7], presenting a higher level of integration and coordination between physical and computational elements [9]. CPS can improve resource productivity and efficiency and enable more flexible models of work organization [10].

The CPS is based on three components, i.e. communication, control and computing, and from these components it is able to perform actions such as: detecting external events using sensors; interact and act in the environment through actuators; collect, save and evaluate data; make decisions based on the data collected and interact with other CPS [11], [12].

According to [13], the essence of Industry 4.0 is applying CPS to realize smart factories. The integration of CPS into production, logistics and services in current industrial practices would transform current factories into an Industry 4.0 factory with significant economic potential [14]. The new smart CPS will deliver innovations in sectors such as manufacturing, energy, transportation, agriculture, automation and healthcare [9]. It can

be seen that the advancement in the applications of this type of system will allow capacity, adaptability, scalability and security that will far exceed the simple systems of today.

2.3 Multi-Agent Systems

MAS [15] is a suitable approach to realize such large-scale and complex industrial CPS, due to its inherent characteristics of modularity, autonomy, decentralization and adaptation to emergence without external intervention. In particular, MAS decentralizes the control functions over distributed and intelligent software agents, running in cloud and edge computational layers, that cooperate together to achieve the system goals facing the condition change and reconfiguration needs [1]. Figure 2.2 presents the structure of MAS.

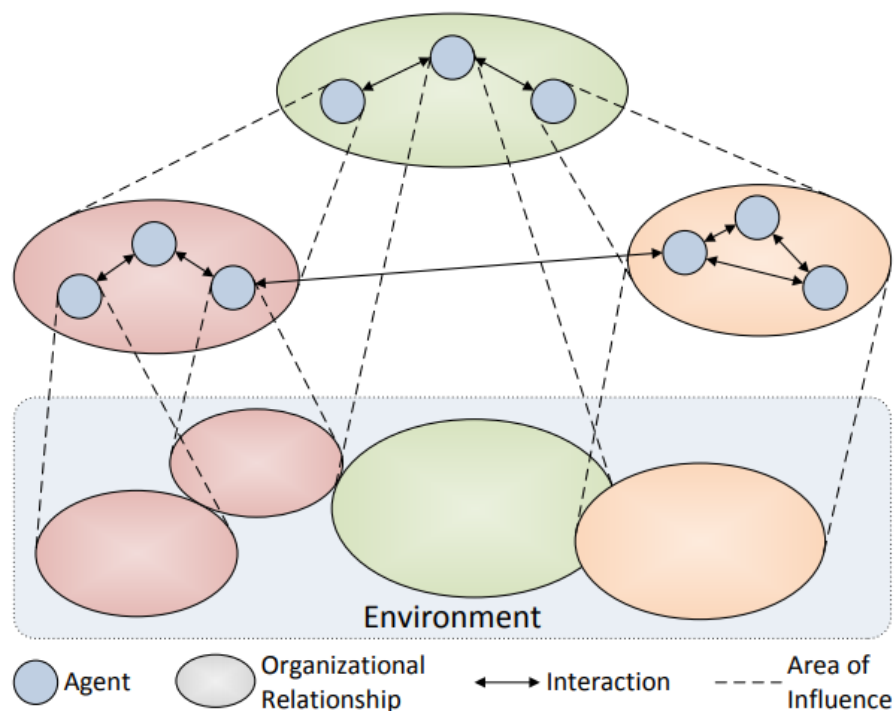


Figure 2.2: Structure of a multi-agent system [16].

The agent is an autonomous software component and endowed with an interoperable interface [17]. The agents in a MAS may decide to cooperate for mutual benefit or may compete to serve their own interests, and they interact with each other both indirectly (by

acting on the environment) or directly (via communication). Regarding the characteristics of an agent, the following stand out [12], [17], [18].

- **Autonomy:** the agent controls its behavior, i.e., its able to make decisions without human intervention.
- **Reactivity:** the agent can be equipped with sensors and actuators, so it can perceive the physical world through sensors and respond by means of actuators.
- **Proactivity:** the agent is able to take the initiative to reach its objectives or to act in response to changes in the environment.
- **Social skills:** the agent can interact with humans or other agents to achieve their goals by sharing or requesting information.
- **Learning ability:** to be autonomous and flexible, the agent must have the ability to learn, adapting to the environment and the needs of the user.

Therefore, the MAS represents and models agents as reasoning autonomously, interacting with other agents, adapting to environmental changes, and pursuing advantages in a rational manner [19]. The wide adoption of MAS in many application domains is due to the beneficial advantages offered, such as efficiency, robustness, scalability, flexibility, reusability, responsiveness and decentralized architecture [20].

2.4 Digital twin

In the context of the digital transformation, the Digital Twin emerges as one of the most promising and innovative technologies for solving key challenges in the development of a smart manufacturing environment [21], [22]. The Digital Twin concept falls under the trend of digitalized ecosystems, and Gartner [23] expects that in the next five years, hundreds of millions of objects, machines or systems will have a Digital Twin and, by 2021, most of the large industrial companies will be using Digital Twin, which will result in a 10 percent gain ineffectiveness [24], [25].

Taking into consideration the different definitions that can be found in the literature, it is possible to establish a general definition of Digital Twin as *"the digital copy of a physical object or system, that is connected and shares functional and/or operational data"*. The collected data posteriorly supports the execution of simulation and data analytics allowing the optimization and enhancement of the physical object [22], [25]. Also, similarities between machine performance and previous assets (historical information) can be measured to predict the future behavior of the machinery [14]. The digital twin allows companies to solve physical problems quickly, detecting them earlier, predicting results with a much higher degree of accuracy, designing and building better products and, ultimately, better serving their customers [26].

A conceptual architecture of a Digital Twin, illustrated in Figure 2.3 is proposed by [26] containing a sequence of six steps, which are:

- Create, step where technologies, i.e. sensors, are implemented in the physical asset to collect data from processes;
- Communication, establishing the appropriate communication protocols to send the information to the twin;
- Aggregation, responsible for gathering all the processed data into a repository.
- Analyse, where the data available in the repository is analyzed
- Insight, related to present the knowledge generated from the data analysis
- Act, step where the knowledge created in the previous steps is forwarded to the actuators in the physical asset.

2.5 Communication Protocols

This section covers some communication protocols commonly used in the interconnection of cyber and physical parts in a CPS perspective. For this purpose, the section will

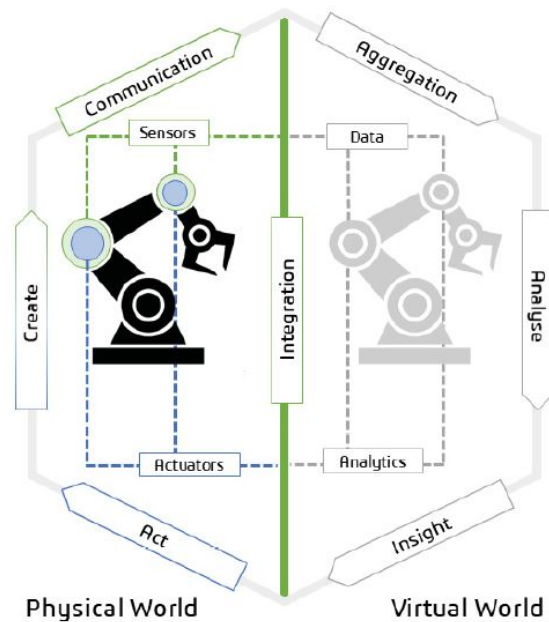


Figure 2.3: A Digital Twin architecture [25].

briefly overview Foundation for Intelligent Physical Agents (FIPA)-Agent Communication Language (ACL), which allows the communication between software agents that may be located on different computational platforms, and MQTT, commonly used in IoT applications.

2.5.1 FIPA-ACL

FIPA is an international organization dedicated to promoting the smart agent industry, openly developing specifications that support interoperability between agents and agent-based applications [27]. In this sense, all agents have to interact in a system with the same language, assigning identical meanings to the concepts under discussion, in order to be able to understand and be understood by other agents. The main way to ensure the interoperability is by defining a default framework for FIPA-ACL messages.

The FIPA-ACL defines communication in terms of a function or action, called the *Communicative Act*, performed by the act of communicating [17]. This message structure contains a set of one or more message parameters and the need for these parameters

for effective agent communication will vary according to the situation. Based on these parameters, agents are able to exchange information and make decisions when a message is received. The FIPA-ACL message parameters are shown in Table 2.1.

Table 2.1: Parameters of the ACL message [17].

Parameter	Description
performative	Type of the communicative act of the message
sender	Identity of the sender of the message
receiver	Identity of the intended recipients of the message
reply-to	Which agent to direct subsequent messages to within a conversation thread
content	Content of the message
language	Language in which the content parameter is expressed
encoding	Specific encoding of the message content
ontology	Reference to an ontology to give meaning to symbols in the message content
protocol	Interaction protocol used to structure a conversation
conversation-id	Unique identity of a conversation thread
reply-with	An expression to be used by a responding agent to identify the message
in-reply-to	Reference to an earlier action to which the message is a reply
reply-by	A time/date indicating by when a reply should be received

2.5.2 MQTT

The MQTT [28] is a messaging protocol based on publisher/subscriber mechanism. This protocol has two entities in the network, which are the broker and the clients. The broker is responsible for receiving all messages from clients and addressing them to their intended destinations, while clients comprise everything that interacts with the broker to send or receive messages. Generally, the client posts messages to a topic, which are sent to the broker. Then, the broker forwards the messages to all clients who subscribe to that topic. This exchange of messages is illustrated in Figure 2.4.

To ensure the reliability of messaging, MQTT supports 3 levels of Quality of Service (QoS) [28], [30], which indicates how consistently the messages under this topic need to be delivered to clients. This levels and their description are presented in Table 2.2.

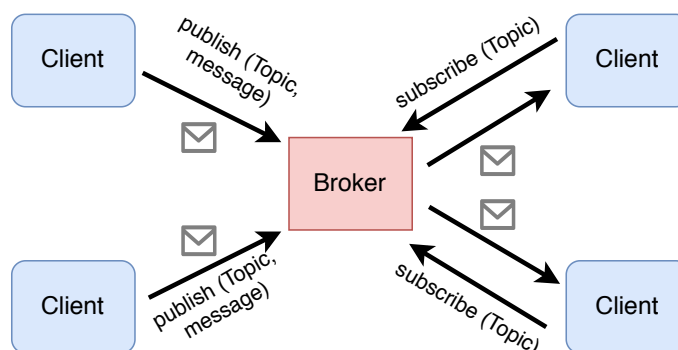


Figure 2.4: Publish-subscribe model using MQTT (adapted from [29]).

Table 2.2: Levels of Quality of Service using MQTT.

Level	Description
0	Sends message only once following the message distribution flow and there is no guarantee if the message arrived to its destination
1	Sends the message at least once, and checks the delivery status. If there is no confirmation of the message being delivered, the server can send the same message twice.
2	Sends the message exactly once. This level uses a 4-way handshake, being impossible to have a message loss. Although the message is guaranteed to be delivered, it is possible to have a long delay during this process.

For the connection between client and broker, it is necessary to define the parameters presented in Table 2.3.

Once the connection is made, the performance publish/subscribe between client and broker follows the parameters presented in the Table 2.4.

2.6 Maintenance Strategies and Techniques

In an industrial environment, maintenance activities are typically intended to reduce failures of machinery creating conditions for increasing equipment availability and consequently increasing productivity. Manufacturing data collected in real-time contains valuable information and knowledge that could be integrated within prediction systems to improve decision making and enhance productivity [31].

Table 2.3: Parameters for the client and broker connection [28].

Parameter	Description
cleanSession	This flag specifies whether the connection is persistent or not. A persistent session stores all the subscriptions and potentially missed messages (depending on QoS) in the broker.
username	The broker's authentication and authorization credentials.
password	The broker's authentication and authorization credentials.
lastWillTopic	When the connection is dropped unexpectedly, the broker will automatically publish a "last will" message to a topic.
lastWillQos	The "last will" message's QoS.
lastWillMessage	The "last will" message itself.
keepAlive	This is the time interval that the client needs to ping the broker to keep the connection alive.

Table 2.4: Parameters to the client subscribe or publish in a broker (adapted from [28]).

Parameter	Description
topicName	The topic under which the message is published.
QoS	The Quality of Service Level
payload	The actual data in the message (only set for the publisher).

Preventive maintenance and predictive maintenance are different strategies, the first being determined by the average or expected life cycle of an asset, while the predictive is identified by the conditions of the equipment. The predictive maintenance is more complex than establishing a preventive maintenance schedule based on the manufacturer's recommendations, but it can be more effective for a company to save time and money, e.g., by detecting a degradation on an equipment performance through analysis technique of vibration allowing its replacement before a total failure occurs.

According to a report by eMaint [32], businesses spend 80 percent of their time reacting to maintenance issues rather than preventing them. This translates to a huge drain of resources including employee productivity, time, and money. Smart factories overcome this problem with proactive predictive maintenance capability, that directly monitors the condition and performance of equipment during normal operation to reduce the likelihood of failures. This gives users early warnings when a machine or network's performance is going to go down, so the team can take the necessary measures to fix the same. It attempts

to keep costs low by reducing the frequency of maintenance tasks, reducing unplanned breakdowns and eliminating unnecessary preventive maintenance. Thus, no downtime or losses accrue to the business.

Therefore, the advantages of using predictive maintenance include reduced equipment interruption due to real-time monitoring, improves equipment reliability, minimizes time spent on maintenance, reduces the chances of collateral damage to the system, improves worker safety and reduces the cost of asset failures.

2.7 Security in applications using IoT

It is essential to note that one of the challenges for the success of IoT lies in information security. Given the heterogeneity of technologies, the need to protect the information collected by the devices until the application must be considered. Note that the communication of things or objects is maintained through network interfaces, which opens up the possibility for different security attacks. The attacks seek to exploit vulnerabilities and can seriously compromise privacy by containing information with sensitive data or influencing behavior or gaining control of the IoT system.

According to [33], [34], when deploying real IoT systems there are a lot of challenges related to vulnerabilities in the use of open network on devices in which it raises questions related to privacy, security, authentication and authorization. Therefore, the protection and privacy of data against unauthorized agents are extremely important to prevent attacks and spying, in order to guarantee the integrity and authenticity of the information from the protocol used on the device to the application layer.

The Digital Twin for the conveyor system is elaborated through the application of all the concepts and techniques presented in this chapter, aiming to monitor its operation and enables the earlier detection of failures based on the real-time data collection using IoT technologies. The following chapters present the system under study, the architecture developed for the digital twin and the result obtained for the real-time monitoring of this system.

Chapter 3

Case Study

This chapter presents the modular and self-organized conveyor transfer system using MAS from which the digital twin to monitor its operation will be carried out. In addition, will be presented the improvements applied to this system to guarantee greater flexibility and capacity for self-organization, as well as allowing efficient monitoring of its functioning from collecting a greater amount of data that allows the detection of failures in advance.

3.1 Description of the Case Study

The conveyor system comprises a sequence of cyber-physical components. The physical part consists of modular Fischertechnik conveyors, where each one comprises a belt operated by a 24V DC motor and two photoelectric sensors used to detect parts in the input and output positions of the conveyor belt, while the logical part is performed through a Raspberry Pi, in which agents are implemented, what is illustrated in the Figure 3.1.

The Raspberry Pi control board is powered by a standard USB cable with a 5V power supply and the General Purpose Input Output (GPIO) ports support 3.3V power. In this way, an interface board, commonly named as a shield, was developed to supply the power from the battery (12V) to the physical part (motors and sensors) and the Raspberry Pi board, ensuring the physical connection between the physical and the logical part of the transport system.

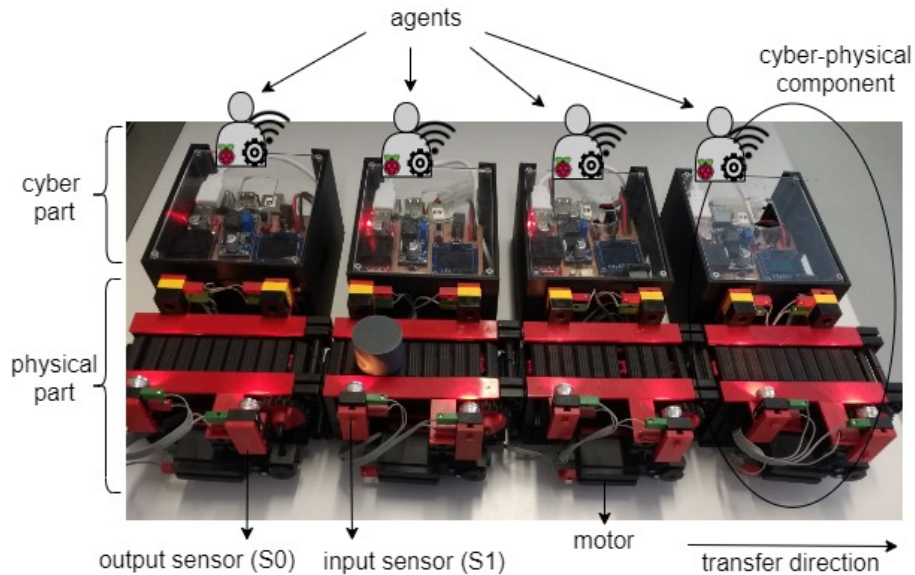


Figure 3.1: Cyber-physical conveyor system.

This system aims to transport a part from a starting point to an ending point through a self-organized set of modular conveyors. The transfer among the conveyors should respect some precedences, e.g., considering the first two conveyor modules in the Figure 3.1, the second conveyor starts its motor when the part arrives to sensor S0, and the first only stops its motor when the part arrives to the sensor S1. The same performance is considered for the other conveyors in the sequence following the transfer direction of the part. These interdependencies between conveyor modules make it necessary to know the sequence of the modules for the application of the logical control, increasing the complexity of the system. The logical control of the modular conveyor system can be implemented using several approaches, as discussed in [1]. An alternative design approach is the use of MAS with self-organization capabilities deployed in a CPS for a modular conveyor system, allowing to achieve scalability and the dynamic system reconfiguration, enabling to remove, add or swap conveyor modules on-the-fly, i.e. without the need to restart, stop or reprogram the system.

The agent-based system for this conveyor system was developed using the Java Agent Development Framework (JADE) [35] and deployed in Raspberry Pi single-board computers, being the communication among them performed through a WiFi network and

exchanged messages formatted according to the FIPA-ACL standard [1].

3.2 Self-Organization Capabilities

Agents need to interact with each other to determine their positions in the sequence and also to regulate the behavior of each individual conveyor system. For the transmission of parts, as described in [1], the sequence of the conveyors modules is not predefined and the agents transmits tokens among them for the transfer operation, similarly to the token passage mechanism in a 4x100m relay run. In this way, when an agent is added to the system, its necessary to determine its position. When a part reaches the input sensor, the conveyor agent will assume that it is the first of the system sequence, assigning itself the position 1 to the *position ID* parameter, and starts the transfer of the part by turning on the conveyor belt. Later, when the conveyed part reaches the output sensor, the agent will propagate a *tokenTransOut* message with its *position ID* to the other conveyor agents, i.e. it will pass the token. The conveyor agent which *position ID* is equal to the token + 1 will start its motor to receive the part. In case that a given conveyor does not know its position, i.e. does not have a *position ID*, it will also start its motor in order to detect if it is the next conveyor in the system. This procedure is repeated until the part reaches the last conveyor.

In addition, when a conveyor agent is added, it registers its skills in the Directory Facilitator (DF) service and sends a message to the other agents informing that it is ready to work. This information makes the agents present in the system update the number of conveyors placed in the system, stored in the *conveyorCount* variable and send this number in response to the agent that was added so that he can update this value locally. In case of a removal of a conveyor module, the conveyor agent sends a message saying that it is leaving the sequence. The conveyor agents receiving this message will decrease the *conveyorCount* variable that indicates the number of agents in the system and those placed after the removed conveyor will downward their locations by one position.

The mechanism to identify the order change of the conveyor modules requires a slightly

different approach since the conveyor counting is kept and the new positions must be discovered. In normal operation, when a part is at the output sensor, a message is broadcasted to all the conveyor agents and is processed by the succeeding conveyor agent and discarded by the rest. When the succeeding conveyor agent detects that the part has not reached by its input sensor (by means of a timeout), it broadcasts the *thereIsASwap* message that warns for a possible order change (and containing the indication of the current position). All the conveyor agents that have a token higher than the received position, will start their motors (naturally guaranteeing that they are in a valid situation, e.g., without having a part to convey). Afterwards, when a conveyor agent receives the piece at its input sensor, it will update its current system position and broadcasts a *swapTokenFound* message with its previous and new positions (updated with the position received during the *thereIsASwap* message). The other conveyor agents that receive this message only update their new positions. More details about the functioning of the system and the protocols of messages exchanged can be found at [1] and [36].

3.3 Improvements Applied in the System Functionalities

This section covers the improvements in the system functionalities, such as the inclusion of a new swap condition between the modules and the installation of new devices allowing to collect more data from the system to be used by the digital twin.

3.3.1 Swap for conveyor with Token 1

The swap mechanism described previously works perfectly for all possible conveyor agent position changes, except for the case where a conveyor agent is changed to the beginning of the sequence to assume the token equal to 1. Thus, in order to improve the performance of the system under study, an optimization of the self-organization capacity was proposed with the inclusion of a new swap condition between the modules. An additional rule was

created which consists of detecting if a conveyor agent with a token greater than 1 receives the indication that a part is placed in its input sensor, without being previously notified by a *TokenTransOut* message. The conveyor agent that identifies the exchange sends a *thereIsASwap* message to the other agents and, those agents that receive this message, will update the token value to 0 and start the motors to re-determine the correct sequence.

3.3.2 New Devices Installed

As the main objective of this work is to develop the Digital Twin for this conveyor system aiming to monitor its operation and enable the earlier detection of failures based on the real-time data collection using IoT technologies, it was observed the need to collect more data from the system to be used as a parameter for monitoring, analyzing the condition of its operation and to be applied in some methods to enable the detection of failures. For that reason, a display and two new sensors were installed and they are better described below.

Display OLED

To the *shield* that connects the battery to the physical and logical parts of the system, an OLED display was added to allow be shown on the small screen in real-time the information if the motor is on or off. This display is very small (0.96”) and has a good resolution. In addition, it does not need a backlight, which configures a very low energy consumption. Due to the characteristics of its technology, the black background color of the display is very defined, giving a great contrast to the displayed image or text [37].

Accelerometer and Current Sensors

Vibration is defined as the movement or mechanical oscillation related to the equilibrium position of a machine or component and it is usually measured by an accelerometer, which is a sensor that measures the dynamic acceleration of a physical device in the form of tension. For monitoring the vibration of the conveyor module, the accelerometer

ADXL345 was used. The ADXL345 is a small, thin, ultralow power, 3-axis accelerometer. It measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (3.9mg/LSB) enables measurement of inclination changes less than 1.0° [38].

Also, the INA219 Current Sensor Breakout [39] was used to allow current and battery voltage monitoring simultaneously.

These two new sensors have been placed only on conveyor A. Figure 3.2 shows how they were installed on the module and also allows viewing the message "motor off" on the display. All of these new devices have been connected to the Raspberry Pi Inter-Integrated Circuit (I2C) bus, being necessary to use them that the interface I2C on Raspberry Pi is enabled. Figure 3.3 shows how the display was connected to the Raspberry Pi. The connections for the other sensors follow the same wiring diagram.

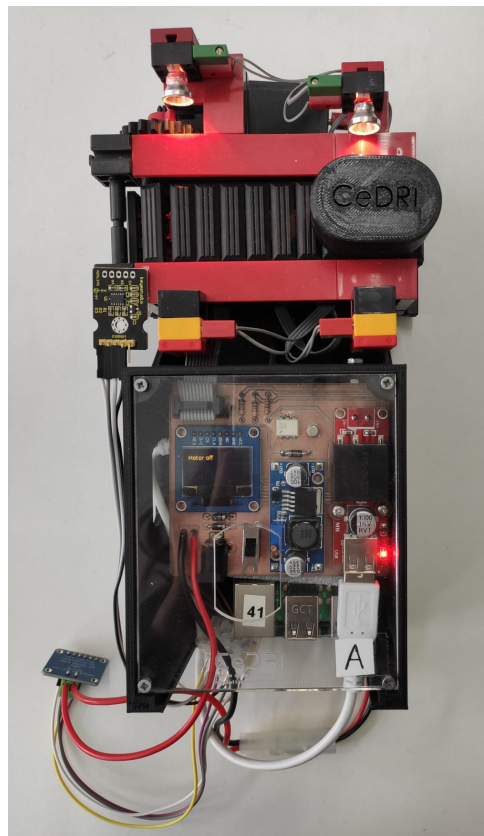


Figure 3.2: Installing the display and the current and accelerometer sensors in the module.

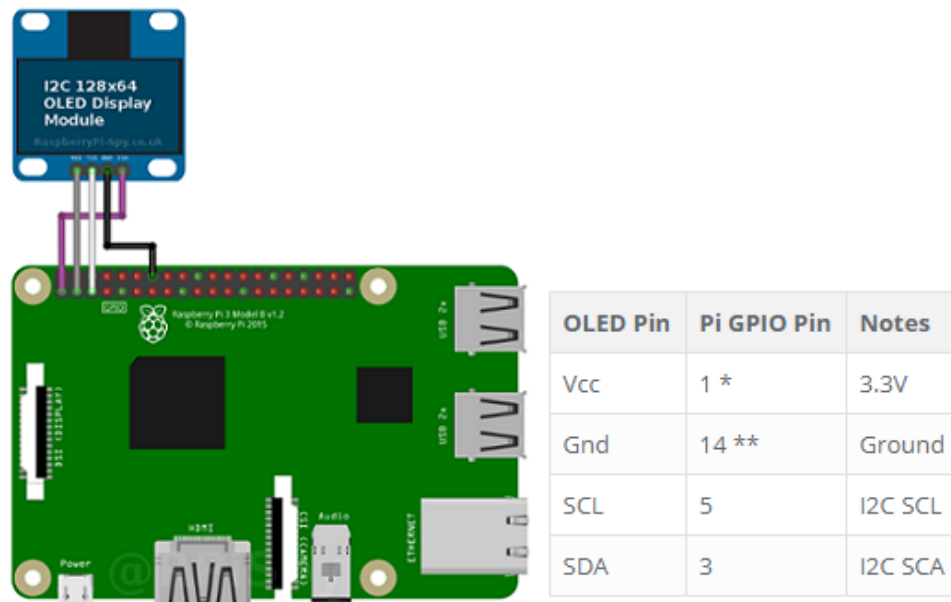


Figure 3.3: Connection diagram between the display and the I2C interface of Raspberry Pi (adapted from [40]).

Chapter 4

Development of the Digital Twin

This chapter presents the solutions adopted for the development of the work, addressing the connectivity between the physical and digital systems and the organization of the data collected to enable virtualization. This chapter also addresses the implementation of control rules for generating alerts, enabling the early failure prediction, and the measures taken to ensure security for the use of the collected data.

4.1 Virtualization and Connectivity

The virtualization of the described conveyor system uses a Node-RED dashboard application, where the physical system is virtually modeled and represented to allow the dynamic visualization and monitoring of its condition state. Node-RED is a visual tool that was used to connect IoT devices, allowing the view of all the messages posted via MQTT through their nodes. The Node-RED dashboard allows building a control panel capable of displaying graphs, presenting data at defined time intervals or in real-time, as well as allowing the manipulation of that data in a friendly interface. Node-RED already provides a comprehensive set of nodes for the construction of basic panels suitable for IoT systems, providing graphics, gauges, controllers, texts, in addition to enabling the making of nodes and custom functions from the JavaScript language [41].

The main graphical interface of Node-RED can be seen in Figure 4.1, where the

blocks are the processing nodes and the set of these nodes form a flow where data is being handled.

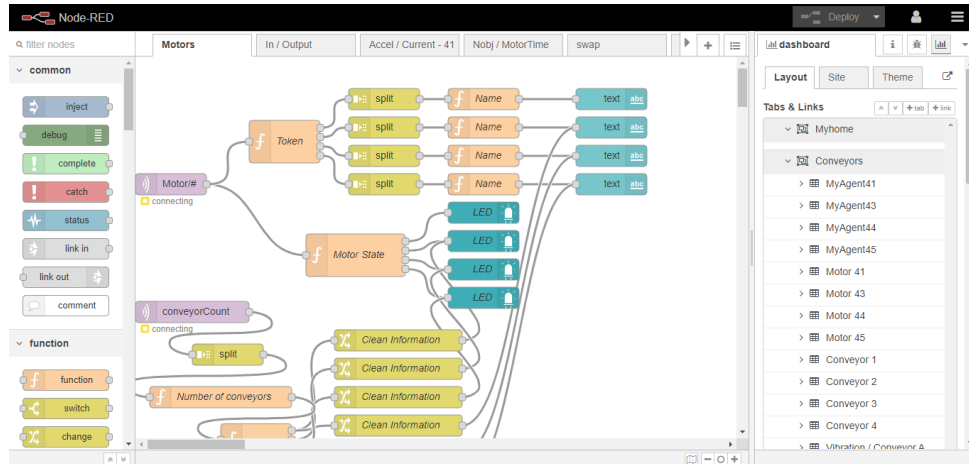


Figure 4.1: Main graphical interface of Node-RED.

The data regarding the conveyor system operation, namely the motor and sensors states, the timestamps in the transfer of parts, the physical sequence of the conveyors, the total amount of pieces transported, the total motor operating time for each conveyor module, the vibration on the three axes, the battery voltage and the current in the motor, is collected by using the MQTT protocol, which stands out for its ease use in IoT solutions and follows the publish-subscribe schema.

As illustrated in Figure 4.2, each conveyor module sends the required data by posting it into specific topics in the MQTT broker, which will forward the received data to the clients that had subscribed that topics, in this case, the Node-RED dashboard application. To ensure that the data collected from the system is posted on its proper topics with their respective contents, the library Eclipse Paho-MQTT [42] was used in the Java programming code used for the implementation of the agents. The Eclipse Mosquitto [43], an open-source broker that implements the MQTT protocol, was used as a broker, running on a different Raspberry Pi connected to a local WiFi network, and also communicates with Node RED via a program node responsible for this communication. Also, all data collected to be used for the virtual system are being stored in a MySQL database.

The data flows created for the collection, processing and visualisation of the system's

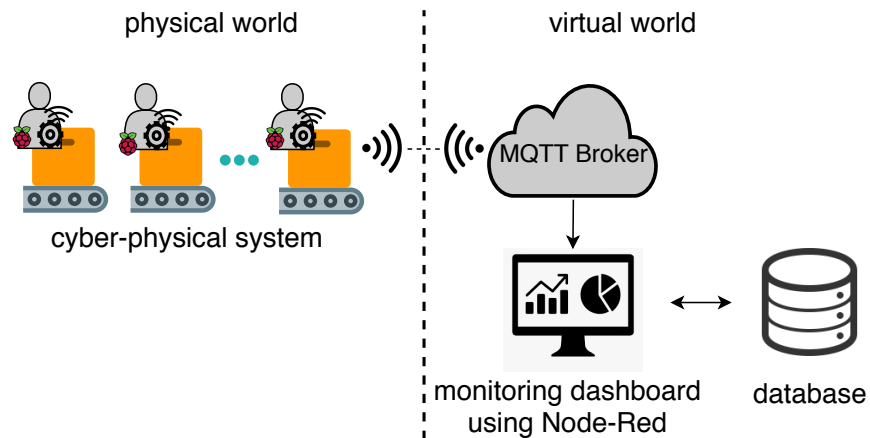


Figure 4.2: Virtualization approach for the conveyor system.

data to allow real-time monitoring of the physical system's functioning will be presented in the sequence. The detailed description of each node used can be found in [44].

4.1.1 Motor State and Physical Sequence of the Conveyors

Figure 4.3 presents a diagram of the data flow created to allow monitoring the motor state and the name of the conveyor that is publishing the information. In general, it is possible to notice in this diagram that the input type nodes (purple nodes) are responsible for the connection between the message published by MQTT and the Node-RED tool, and the nodes that end the flows (blue nodes), are responsible for posting messages on the dashboard.

In order to be able to identify in real-time the motor state and the respective conveyor that is publishing the message, a topic and a content was defined to allow the connection between the publisher and the subscriber. Thus, in the Java programming of the agent, the topic was defined as "Motor/ name of the agent that is publishing the message" and the content with the messages "Motor off" or "Motor on", followed by the Token information, which indicates its position regarding the order in which the parts pass through the system. Thus, the MQTT input node was used in Node-RED, ensuring that the published message from the agent to the broker can be received according to its

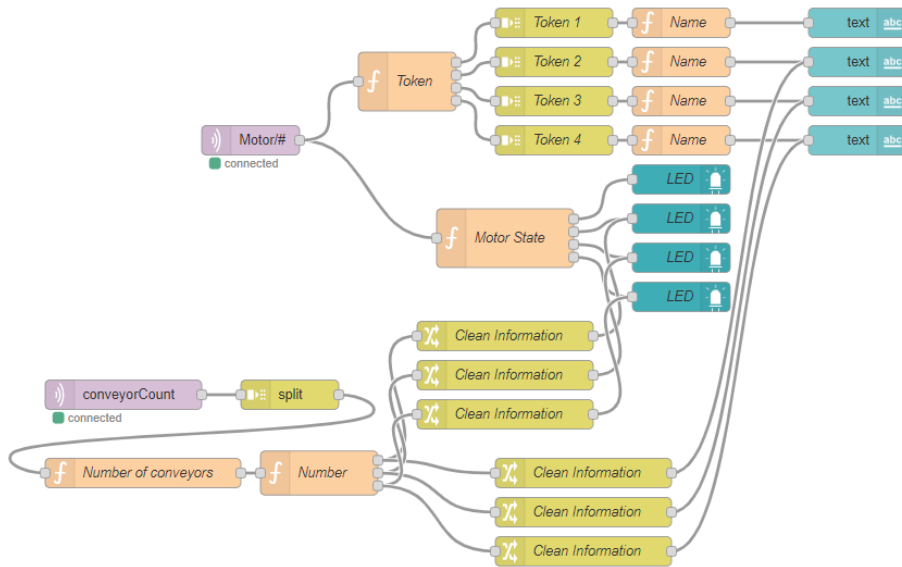


Figure 4.3: Node-RED flow for the motor state and conveyor physical sequence monitoring.

specified topic. For this node, the topic was defined as “Motor/#” to make possible to receive all messages from the topic Motor, regardless of the name of the agent posting the message. In addition to this configuration for this node, the connection between that node and the Mosquitto broker was defined with the IP of the Raspberry Pi where the broker was installed (10.20.38.28), the port 1883, and the QoS was set to 0. Figure 4.4 presents this configuration in Node-RED.

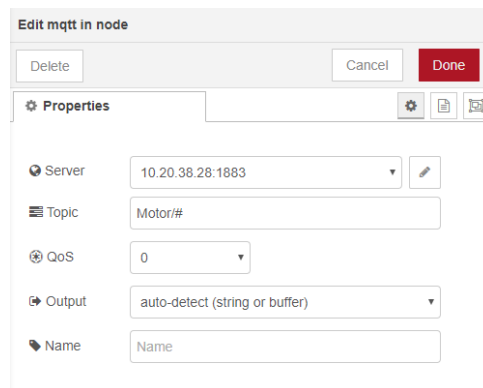


Figure 4.4: Connecting the node MQTT input to the MQTT broker.

For the name of the conveyor that is publishing the information, the upper data flow

is used, in which there is a function node “Token”, being responsible for checking in the content of the received message the number of the Token sent by the agent. According to this number, the messages related to the topic can follow different flows through the four possible outputs for that node, e.g., if the content includes “Token: 1”, the published topic will be transmitted by the first output of the function node; if the content includes “Token: 2”, the published topic will be transmitted by the second output of the function node; if “Token: 3”, the topic goes through the third output of the node and, if “Token: 4”, the last output of the node is used, being possible for the digital copy to reproduce the same physical sequence as the conveyor modules.

The Split nodes (Token 1, Token 2, Token 3, Token 4) are responsible for separating the message sent in the topic in two parts, returning two messages as output: the first “Motor” and the second being the name of the agent posting the message. Once again, a function node “Name” is placed to cancel the messages received related to the word “Motor” and transform the names of the agents, which until then still refers the name of the Raspberry IP in which they are implemented, in letters of the alphabet (A, B, C, D). These letters are published through the text node in the User Interface (UI), allowing the user to identify the conveyor module that is sending the message. The name of the conveyor modules can be seen in Figure 4.5.

The data flow for the identification of the motor state begins with the function node “Motor State”, in which the content of the published message is analyzed first according to Token, determining in which function output the message will be transmitted, in the same way as previously explained from Token 1 to 4, and then it is checked whether that same content includes the message “Motor on” or “Motor off”. If the received content includes “Motor on”, this message is changed to the Boolean type “true” and, if it includes “Motor off”, this message is changed to the Boolean type “false”. The “true” message causes the Light-Emitting Diode (LED) node to light a green LED on the UI, indicating that the motor is working. The message “false”, on the other hand, causes the LED to turn red, indicating that the motor of that conveyor module is off. Figure 4.6 shows this configuration for the LED node used. In this way, there is a LED for monitoring the

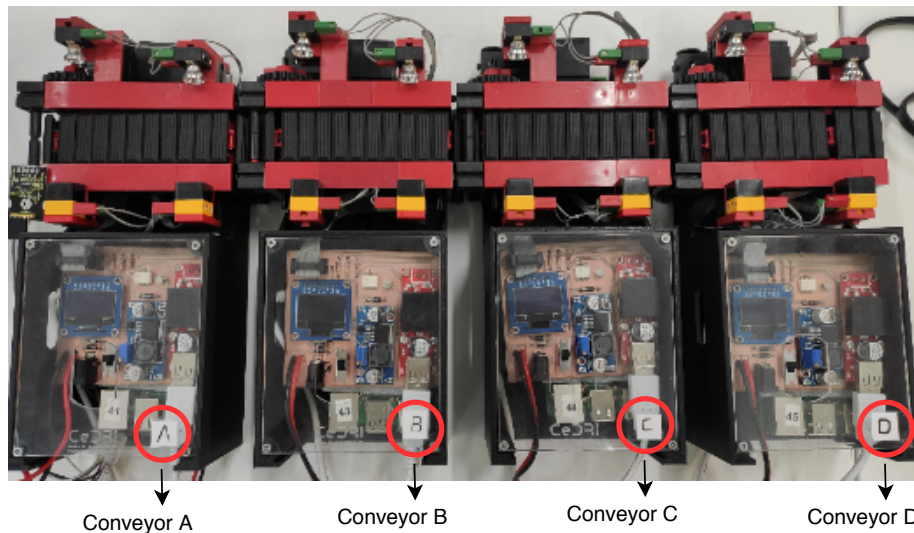


Figure 4.5: Naming of the conveyors in letters to facilitate identification.

motor operation of each conveyor module, being activated according to the physical order of the conveyor.

Observing the functioning of the virtualized system, it was noticed that after the information is published on the dashboard, it is maintained until new information is published. Thus, in the case of a removal of a conveyor module, the last previously published information was not deleted from the dashboard since there was no new information to replace it, giving the impression that the conveyor was still present in the sequence. Thus, to be able to delete information when a carpet is removed from the sequence, a new variable from the agent’s Java programming was used to publish the number of conveyors in use, which is the “conveyorCount” variable, which is updated during system operation whenever a conveyor is added or removed from the sequence.

Therefore, the second MQTT input node was defined with the topic “conveyorCount” and the content published in that topic is “ConveyorCount: Number of conveyors identified”, and this number can vary from 1 to 4 according to the number of conveyors. The Split node in the sequence of the flow is used to separate the message sent in the content in two parts when the “:” is found, resulting in two messages, “ConveyorCount”

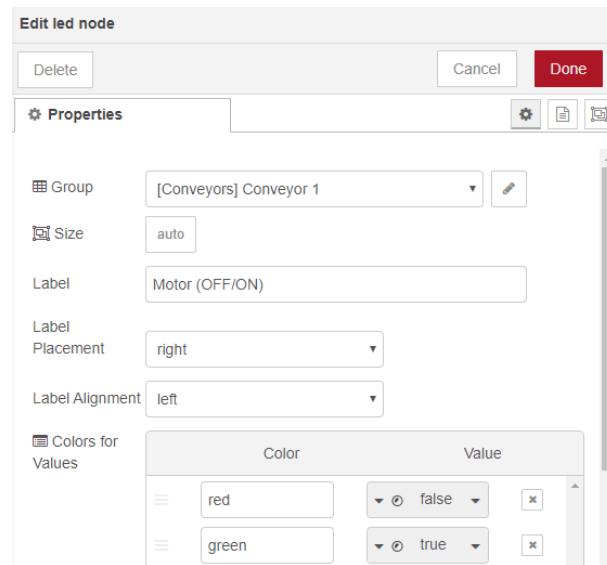


Figure 4.6: LED node configuration.

and “Number of identified carpets”, being the function node “Number of conveyors” responsible for discarding the first message and transmitting only the identified number. The next function node “Number” identifies the quantity received in the message and, if `conveyorCount = 1`, it means that only one conveyor module is in use and, thus, the three outputs of the function node transmit an empty message, responsible for eliminating from the dashboard with letters and motorLEDs of the three other modules that are not in operation. If `conveyorCount = 2`, two modules would be in operation, so the first output does not transmit a message and the two other outputs transmit an empty message ([]) to eliminate information from the other two modules that are not in operation. If `conveyorCount = 3`, only the last output transmits an empty message since only one module would not be operating. Finally, for `conveyorCount = 4` it is not necessary to delete the remaining information since the four modules will be publishing on the dashboard.

As explained earlier, the LED nodes receive messages of the Boolean type and the text nodes receive messages of the string type, so it is not possible to use the same type of message to eliminate the information from these two types of nodes at the same time. Thus, the first three “Clean Information” nodes are responsible to change the empty message received so that the LED nodes identify that the content received is empty, and

the other three nodes do the same for the text nodes.

4.1.2 Sensors States and the Total Amount of Pieces Transported

The diagram in Figure 4.7 presents the data flow to monitoring the state of the photoelectric sensors and the total amount of pieces transported.

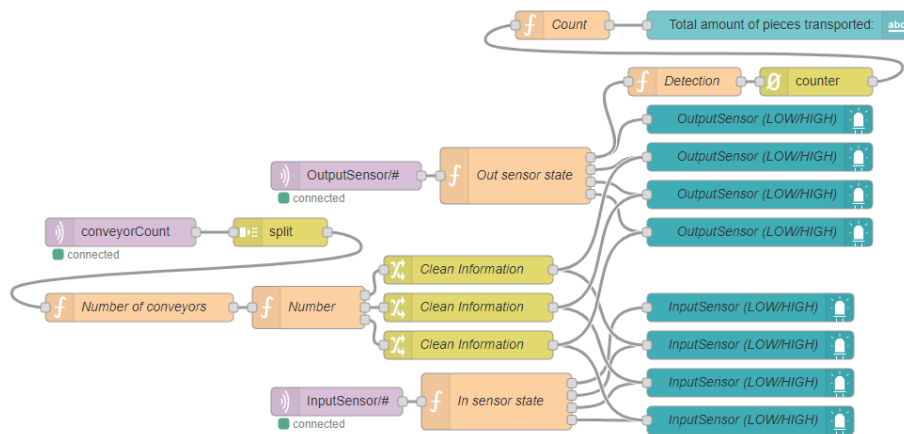


Figure 4.7: Node-RED flow to monitor sensor states and the total amount of pieces transported.

For the identification of the input sensor state of each module, the publisher sends the topic “InputSensor/ name of the agent that is publishing the message” and the content “InputSensor: low” or “InputSensor: high”, followed by the Token information of the conveyor to inform its position in relation to the sequence being used. In this way, the data flow created begins with the MQTT input node whose subscribed topic is defined by “InputSensor/#” to allow receiving all messages from the InputSensor topic, for all agents that are publishing the message. The other configurations for this node follow the model shown in Figure 4.4. Then, the function node “In sensor state” is used to check the content of the published message, first analyzing the informed Token, determining in which output of that node the message will be transmitted, and the content with Token 1 forwarded to the first output and so on until Token 4 is forwarded to fourth output. Followed by this analysis, it is verified whether this same content includes the message

“InputSensor: high” or “InputSensor: low”. If the received content includes “InputSensor: high”, this message is changed to the Boolean type “true” and if it includes “InputSensor: low”, this message is changed to the Boolean type “false”. The “true” message causes the “InputSensor (LOW / HIGH)” LED node to light a green LED on the UI, indicating that a part is passing through the sensor, which causes it to be activated. The “false” message causes the LED to turn red, indicating that there is no part passing through the sensor. A LED was configured on the dashboard to monitor the input sensor of each conveyor module.

The data flow for the output sensor was created in a similar way to the one explained above for the input sensor, with the difference that the topic published by the agent is defined as “OutputSensor/ name of the agent that is publishing the message” and the content is “OutputSensor: low” or “OutputSensor: high”, followed by the Token information of the conveyor. The MQTT input node is then subscribed to the topic “OutputSensor/#”. The function node “Out sensor state” has the same function as the “In sensor state” node explained, checking if the content of the received message includes the message “OutputSensor: high” or “OutputSensor: low”, changing these messages to the Boolean type “true” or “false”, respectively, responsible for activating the “OutputSensor (LOW/HIGH)” LEDs on the dashboard.

Also to this flow was added a block of nodes responsible for monitoring the total amount of pieces transported. This number of pieces is identified whenever a piece is detected in the module’s output sensor with Token 1, which is the first in the sequence. The fact that this counting is performed on the first conveyor ensures that the parts are computed regardless of the quantity of conveyor being used in the sequence since the first, necessarily, will always exist to allow some transport. Thus, the “Detection” function node is responsible for identifying when a “true” message is sent to the LED, making the “counter” node to start counting the number of times the sensor was activated, which indicates the number of pieces that went through it. The next function node “Count” is responsible for identifying the number sent by the counter and transmitting that number to the text node “Total amount of pieces transported”, which will display this value on

the dashboard.

The block of nodes that receive the variable conveyorCount is used again to erase unwanted information if any conveyor is removed from the sequence and presents identical configurations to those presented previously.

4.1.3 Amount of Pieces Transported by Each Conveyor

The count of the total amount of pieces transported presented in the previous subsection is done without taking into account the number of conveyors running in the sequence and from which the transport is being carried out. Differently, this topic addresses a new count performed by the agent itself when it identifies activation of the output sensor, indicating that the conveyor on which it is implemented transported a part. For this, in the agent’s programming, a “quantity” variable was created, which is increased according to the condition that an activation of the input sensor and the module’s output sensor has been identified, to ensure that the count is made for a piece that passed through the entrance and reached the exit of the conveyor. Thus, for MQTT communication, the topic “Nobjects/ name of the agent that is publishing the message” and the content “Number of pieces: Value of the quantity variable” was defined. The diagram in Figure 4.8 presents the data flow to monitoring the number of pieces transported by each conveyor.

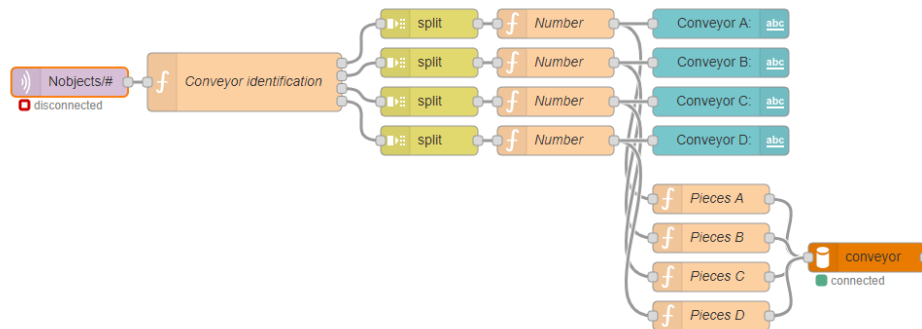


Figure 4.8: Node-RED flow to monitor the number of pieces transported by each conveyor module.

As can be seen with Figure 4.8, the MQTT input node has been configured with the

topic “Nobjects/#” to receive all quantities being published by all agents. The “Conveyor identification” node in the sequence is responsible for analyzing the name of the agent published by the topic to determine in which output the message will be transmitted, considering the first output being for Conveyor A messages, the second for Conveyor B, the third for C and the last for D. The Split nodes are then responsible for separating the content into two messages, "Number of pieces" and "Value of the quantity variable". The function node “Number” cancels the first message and transmits only the identified number, which will be shown on the dashboard through the text nodes Conveyor A, Conveyor B, Conveyor C and Conveyor D.

In addition, the quantity of parts transported is transmitted to the “Pieces” function nodes for each conveyor, this node being responsible for defining the command that inserts the value received in the message into the desired table and column, e.g., "INSERT INTO piecesA (number) VALUES (" +msg.payload+");", allowing the data to be stored in the “Pieces” tables for each conveyor in the database. Finishing the flow, the MySQL database node is used to allow the connection between the data received from the system and the database. The configuration of this node is shown in Figure 4.9. This node is already configured with the Host and Port, and besides them, it is necessary to define User, Password and the database with which the communication will be made. MySQL has a user named “root” with an empty password and, for this project, the database “conveyor” was created, according to the configuration shown in the image.

4.1.4 Vibration on the Three Axes

Data from the accelerometer were collected to monitoring the equipment condition and its behavior when it is working. For this purpose, a function that was run in parallel to the normal operation of the agent was added to the agent’s programming, being responsible for measuring the acceleration in the three axes during the period that the motor remains on and publishing these values through the MQTT protocol. Thus, for the publication of messages, the topic “Accelerometer/ name of the agent that is publishing the message”

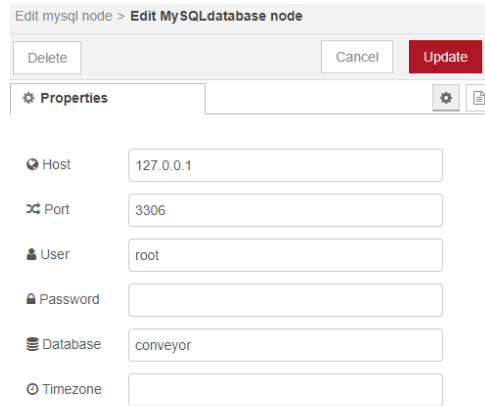


Figure 4.9: MySQL node configuration.

and the contents “Acceleration/ X-Axis: Value”, “Acceleration/ Y-Axis: Value” and “Acceleration/ Z-Axis: Value ” so that the values obtained are sent in separate messages, with the identification of which axes they refer to. Figure 4.10 presents a diagram of the data flow created to allow monitoring of the vibration on the three axes.

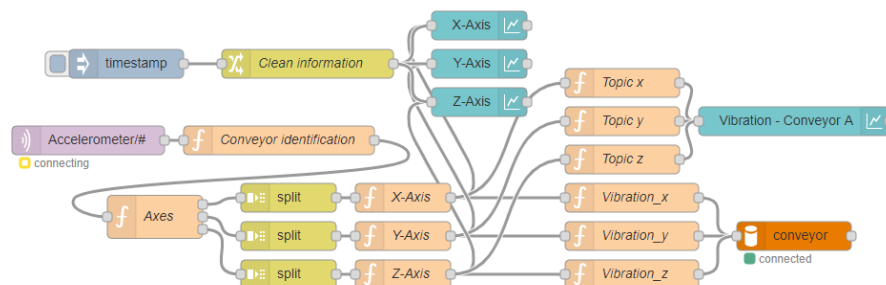


Figure 4.10: Node-RED flow for the vibration on the three axes monitoring.

Although only one conveyor has been equipped with the accelerometer sensor, the programming in Node-RED was developed in order to be easily adapted if other sensors are placed in the other modules. The MQTT input node has been configured with the topic “Accelerometer/#” to receive all messages being published on that topic. The “Conveyor identification” node has the same function as the nodes of this type explained in the previous subsections, however, as the topic is published in this case by only one agent, this node has only one output, referring to Conveyor A. The “Axes” node in the sequence separates the transmission of messages according to the three contents received,

the first content (X-Axis) being transmitted by the first exit, the second (Y-Axis) on the second exit and the third (Z-Axis) at the third exit. Split nodes are responsible for separating the content received in two messages, the first with the text and the second with the obtained acceleration value. The function nodes “X-Axis”, “Y-Axis” and “Z-Axis” cancel the first messages and transmit only the identified numbers.

The three output nodes for the dashboard at the top of the flow show the values identified for each axis in different graphs in the UI to enable the user to better visualize each curve. The "timestamp" and "clean information" nodes are used to clean the data in these graphs without having to restart the program.

To allow a summary demonstration of the values for the vibration in the three axes, the graphic node “Vibration - Conveyor A” was configured, which will be displayed in the main interface, along with the main system data. The “Topic x, y and z” nodes define a different topic for each curve related to an axis, allowing to differentiate the information plotted in the graph. Finally, for the values to be stored in the database, the function nodes “vibration_x”, “vibration_y” and “vibration_z” are connected to the MySQL database “conveyor” node and commands similar to the one described in the previous subsection is used for the data transmission.

4.1.5 Total Motor Operating Time

In order to check the total motor operating time, a time count was performed between starting and stopping the motor. Monitoring this time is important because, if a value was higher than the normal expected for the transport of a part, this may indicate the occurrence of a problem in the transport process, i.e. something is blocking the conveyor belt or even the battery voltage is dropping affecting the system operation. The topic defined for the publication of this message was “MotorTime/ name of the agent that is publishing the message” and the content “Motor running time: Value”. The diagram created to monitor this time is shown in Figure 4.11.

The MQTT input node, as in the other diagrams, was configured with the topic

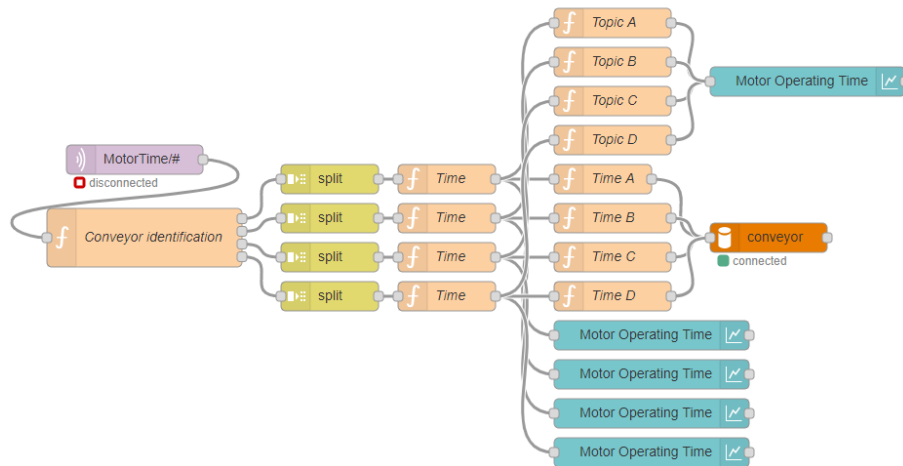


Figure 4.11: Node-RED flow for the total motor operating time monitoring.

“MotorTime/#”, being subscribed to all topics regarding motor operating time for all agents publishing the message. The “Conveyor identification” node, similar to that shown in other diagrams, is responsible for analyzing the name of the agent published in the topic to determine which output the message will be transmitted on, e.g., first output to Conveyor A, second to Conveyor B, the third for C and the last for D. The Split nodes are then responsible for separating the content transmitted in two messages, "Motor running time" and "Value", this value is the identified time of the motor running. The function node “Time” cancels the first message and transmits only the identified number. The “Topic A, B, C and D” nodes define a different topic for each curve regarding the running time of each motor, allowing to differentiate the information plotted on the “Motor operating time” graph that is displayed on the main interface of the dashboard, being responsible for showing the four curves with the time values for the four conveyor modules.

For storing data in the database, the function nodes “Time A, B, C and D” are connected to the MySQL database “conveyor” node and commands similar to the one described in the previous subsections are used to allocate the values in the table and corresponding column, e.g., “Time A” inserts the data in the “MotorTimeA” table in the “Time” column, and the others follow the same pattern.

The four output nodes for the dashboard at the bottom of the flow, named “Motor

operating time”, shown in different graphs in the UI the identified time values for each conveyor module, allowing the user to better visualize each curve.

4.1.6 Battery Voltage, Power and Current

Battery monitoring is important to ensure that the drop in its level does not affect the operation of the system, e.g., decreasing the transport speed of the carpets, causing failures in sending data between agents or between the physical and virtual system, or even causing the entire system to shut down. The current monitoring allows analyzing its consumption for normal operation of the system and, a consumption that exceeds the expected value, may indicate a problem occurring with the conveyor or the part being transported. Power is also a monitored variable, obtained by multiplying voltage and current. These data were collected from the system using the current sensor, adding to the agent’s programming a function executed in parallel to the normal operation of the agent, being responsible for measuring these three variables during the period when the engine remains on and publishing these values through of the MQTT protocol. Thus, for the publication of messages, the topic “inaSensor/ name of the agent that is publishing the message” and the contents “Voltage: Value”, “Power: Value” and “Current: Value” were defined. For the monitoring of the variables in question, the data flow presented in the diagram of Figure 4.12 was elaborated.

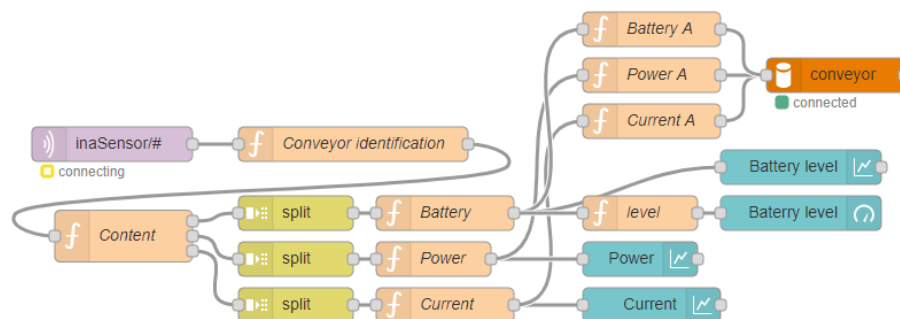


Figure 4.12: Node-RED flow to monitor current, power and battery voltage.

The sensor for measuring these variables was also added only to one conveyor, however, the programming in Node-RED was developed in order to be easily adapted if other

sensors are placed in the other modules. Thus, the MQTT input node was configured with the topic “inaSensor/#” to receive all messages being published on that topic. The “Conveyor identification” node has the same function as the nodes of this type explained in the previous subsections, however, as the topic is published in this case by only one agent, this node has only one output, referring to Conveyor A. The “content” node separates the transmission of messages according to the three contents received, the first content (Voltage) being transmitted by the first output, the second (Power) on the second output and the third (Current) on the third output. Split nodes are responsible for separating the content received into two messages, the first with the text and the second with the value of the variables obtained. The “Battery”, “Power” and “Current” function nodes cancel the first messages and transmit only the identified numbers.

In sequence, the function nodes “Battery A”, “Power A” and “Current A” are connected to the MySQL database “conveyor” node and commands similar to those described in the previous subsections are used to allocate the values in the corresponding table and column, e.g., "Battery A" inserts the data in the "BatteryA" table in the "Voltage" column, and the others follow the same pattern.

The function node “level” is used only to transmit the voltage value received to the “Battery level” dashboard output node which generates a gauge type graph on the main interface of the dashboard. The chart node “current” also displays the current value on the main interface of the dashboard, and the other two chart nodes “Battery level” and “Power” present these values on another interface on the dashboard.

4.2 Real-time Monitoring

The collected data is analyzed in real-time to monitor the health condition of the conveyor system and its correct behavior, and also to allow the detection of abnormal situations, preferably in advance. In this system under study, abnormal situations can be considered as a sudden disconnection of the modules, delay in the transport time of the part or even an elevation of the current and vibration values. These situations can be caused

by a decrease in the battery level, incorrect positioning of the part on the conveyor belt, alteration of the material of the transported part, removal of the transported part before it reaches the exit of the conveyor, blocking of the conveyor belt or changing the order of the conveyor modules.

For this purpose, an alert system was developed to warn risk situations related to the functioning of the conveyor system. Some of these alerts are generated by applying process control methods, such as Nelson rules [45]. These rules are based on the mean value (\bar{X}) and the standard deviation (σ) of the samples, applied to a control chart on which the magnitude of some variable is plotted against time, making it possible to determine if a measured variable is out of control or presents a trend that shows the variable is near to be out of control, as illustrated in Figure 4.13.

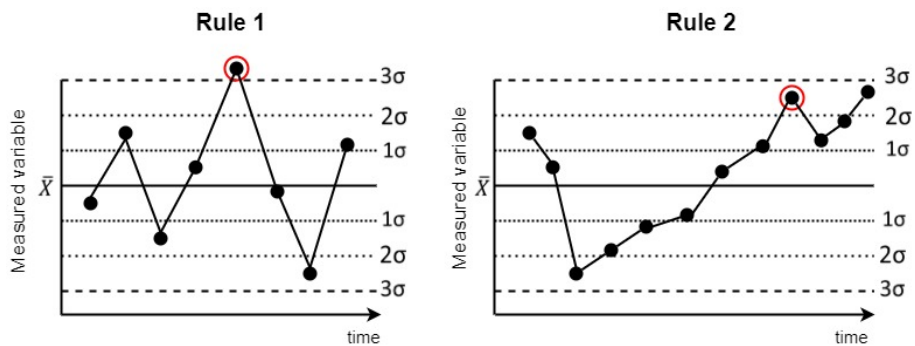


Figure 4.13: Rules to detect outliers and trends in the conveyor operation behaviour.

Figure 4.13 presents two rules used to identify or detect any evidence that the process average and its dispersion are not operating at stable levels, indicating that the conveyor is in an out of control condition.

Rule 1 is used to detect an outlier in the operation of the conveyor, that happens when a value of the variable being analyzed is greater than 3σ . However, as it is desired to detect an abnormal situation in advance, the alerts generated from this rule were triggered when the values of the variables exceed 2σ . In this way, this rule was applied for the variables of current, battery, vibration in the three axes and for the time in the transfer of parts in a conveyor, according to their mean and standard deviation values considering the normal system operation.

Rule 2 allows identifying a continuous growth trend in values of the measured variable, i.e. six successive increasing values in a row, allowing to detect in advance possible failures. For the alerts generated from this rule, the variables of current, vibration and motor operating time were used. Whenever a value of one of these variables is added to the database, an analysis is performed with the last six values recorded, checking whether there is an increasing or decreasing trend.

When alerts are generated, a pop-up menu appears on the dashboard, indicating which measure is near to be out-of-control and also, emails are sent to the responsible for monitoring conveyor system, notifying the type of problem that has occurred. To make this possible, some nodes were added to the data flow diagrams of these variables previously presented, i.e. the function nodes, which allow defining the values for rules 1 and 2, the notification nodes and email nodes. The changes in the diagrams are shown in the following figures, except for Figure 4.14 diagram, which is presented for the first time since this collected data was used exclusively with the application of rule 1.

4.2.1 Timestamps in the transfer of parts

Figure 4.14 presents the data flow created for the analysis of the Timestamps in the transfer of parts. Unlike the “Motor operating time”, which is sent to the digital copy only when the motor stops running, this new data was collected through a function that was run in parallel to the normal operation of the agent was added to the agent’s programming, being activated whenever the motor starts and it remains to send the time relative to that operation until the motor stops. Thus, for the publication of messages, the topic “piecestime/ name of the agent that is publishing the message” and the content “ConveyorCount: number of conveyors / Time to transport the piece: time” was created.

The MQTT input node has been configured with the topic “piecestime /#” to receive all messages being published on that topic. The “Conveyor identification” node has the same function as the nodes of this type explained in the previous subsections, separating in each output of this function the agent who posted the message. Since the number of

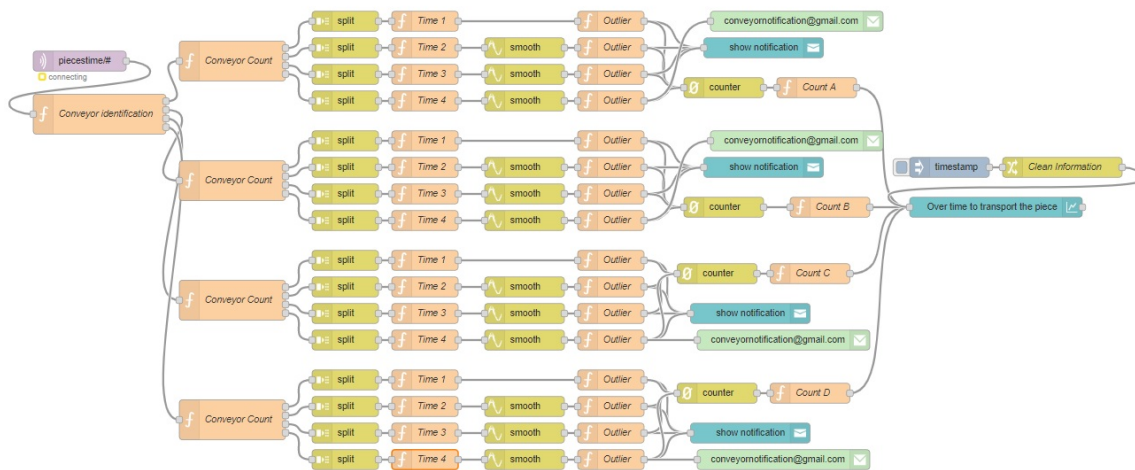


Figure 4.14: Node-RED flow for the time in the transfer of parts monitoring.

conveyors present in a sequence interferes with the transport time of the piece, e.g., only one conveyor transports faster than using 2 conveyors and so on, the “conveyorCount” nodes were used to check the message content for the number of registered conveyor agents. Split nodes separate the message content between “ConveyorCount: number of conveyors” and “Time to transport the piece: time”, and the Time 1, Time 2, Time 3, Time 4 nodes identify the time value in seconds sent by each agent according to the number of modules in the sequence. The smooth nodes average between 2 values received and this average is analyzed by the Outlier node, in which if the received value exceeds 2σ in relation to the average defined for this variable, a topic “Transport time of a piece” and a message “Problem with the piece on the conveyor A” is defined. This topic and message are shown in the alert generated on the dashboard and an email is sent recording the alert. In addition, the occurrence of the outlier activates a counter, “counter” node, and this count is transmitted to an “Over time to transport the piece” graph which will be displayed in the main interface. The “Count A”, “Count B”, “Count C” and “Count D” nodes are used to define which conveyor the count belongs to, separating the information that will be presented in the graph.

4.2.2 Vibration

Figure 4.15 shows the nodes added to the diagram shown in Figure 4.10 for the application of Rules 1 and 2.

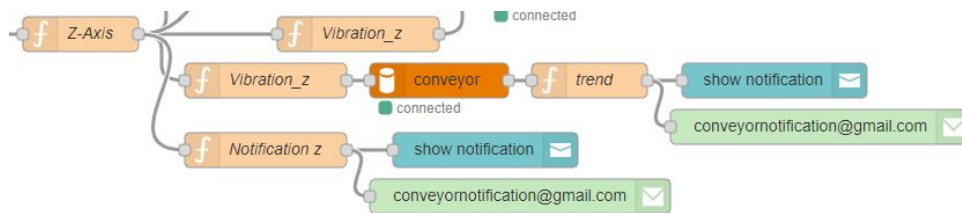


Figure 4.15: Node-RED flow for the vibration monitoring adding the rules 1 and 2.

The “Vibration_z” node searches the database for the last 7 values entered using the command “SELECT z_axis FROM vibrationA_z ORDER BY ID DESC LIMIT 7;”. The trend node checks whether these values retrieved from the database show an increasing or decreasing pattern and, if one of these patterns is detected, a notification is generated with the topic "Trend Asc" or "Trend Desc" and a message "Vibration Z_axis" indicating for which variable the trend was detected.

The Notification z node is responsible for defining the rule for outlier detection. Thus, if the received value exceeds 2σ in relation to the average defined for this variable, a topic “Acceleration Z-Axis” and a message “The acceleration in Z-Axis is higher than usual” is defined.

4.2.3 Motor Time

Figure 4.16 shows the nodes added to the diagram shown in Figure 4.11 for the application of Rule 2.

The nodes added to identify a trend perform functions similar to those explained previously concerning this rule. However, the command for checking the database is “SELECT time FROM motortimeA ORDER BY ID DESC LIMIT 7;” and the message generated for notification is “Motor operating time” indicating for which variable the trend was detected.

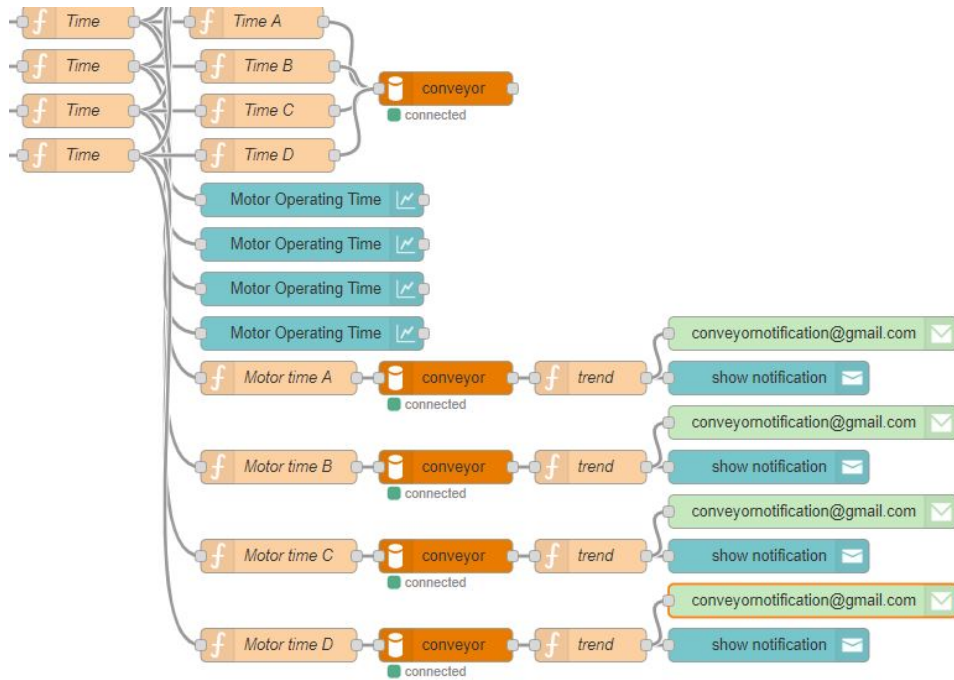


Figure 4.16: Node-RED flow for the motor time monitoring adding the Rule 2.

4.2.4 Battery Voltage and Current

Figure 4.17 shows the nodes added to the diagram shown in Figure 4.12 for the application of Rule 2.

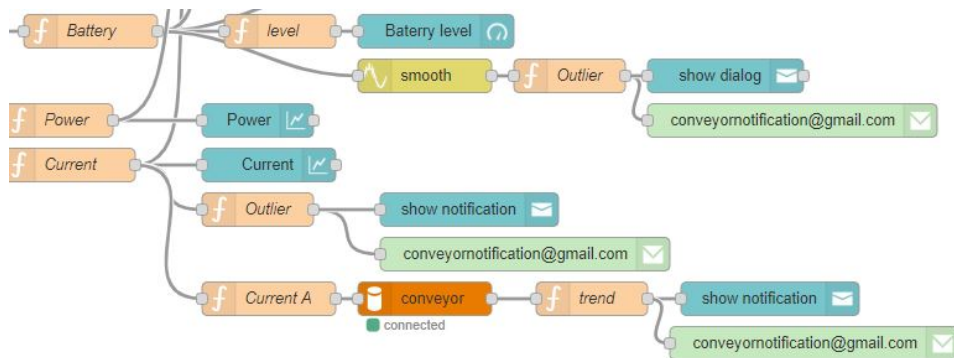


Figure 4.17: Node-RED flow for the battery voltage and current monitoring adding the Rules 1 and 2.

The addition of the rules for monitoring the current and battery voltage follow the same principles explained above.

4.2.5 Swap

Also, when the virtual system identifies the occurrence of a swap in the physical conveyor system, an alert is displayed in the dashboard, and an update on the conveyor sequence displayed in the dashboard is performed. The diagram created to monitor the swap is shown in Figure 4.18.



Figure 4.18: Node-RED flow for the swap monitoring.

Whenever the agents detect a swap in the sequence of modules, the `swapAlert` variable receives a value of “true”. A Swap detection can justify a change in the digital order of the modules and also an increase in the operating time of the motors since when a change occurs, the agents need time to be able to identify it. Thus, for the publication of messages by MQTT, the topic “theresASwap/ name of the agent that is publishing the message” and the content “swapAlert: true” were defined. In Figure 4.18, the MQTT input node was configured with the topic “theresASwap/#”. The function node “Swap notification” analyzes whether the received content message contains the word “true”, returning a payload message “Swap detected. Date” and a topic “Swap” to be displayed in the notification. The notification node defines the place on the dashboard where the notification will be displayed, the color and the display time. The email node sends the message and the topic set to the registered email.

4.3 Security Weaknesses

In the IoT context, due to the inherent heterogeneity of the internet-connected objects and the ability to monitor and control physical objects, security and privacy are a challenge [34], [46]. The use of connected systems and traffic through computer networks allows information to be vulnerable and subject to threats from external elements, which can

compromise an application [47].

The system under study was analyzed in order to identify possible weak spots that are vulnerable to threats. The identified vulnerabilities in the case study are represented in Figure 4.19.

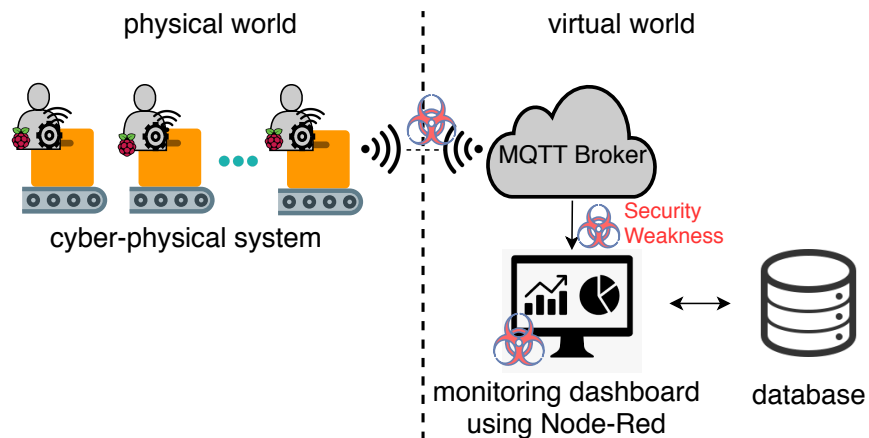


Figure 4.19: Security weakness in the case study.

The default configuration of the MQTT protocol allows the easy observation of the messages being exchanged with the use of a software tool, known as a sniffer, to monitor the network traffic. Sniffers can be found for free over the internet, e.g., Wireshark, and can be used by anyone, facilitating the monitoring and information eavesdropping. The access to the IP address of the broker and client is also allowed using a sniffer. Also, the Node-RED platform is not secured, and if knowing the IP host where the Node-RED application is running, it is possible to access the editor and deploy changes. In spite of the editor allows the user to know which pages are being modified, it is unable to act against it. For that reason, Node-RED applications present a weak spot with a huge vulnerability from outside attacks.

Knowing this security weakness, some measures were applied to eliminate the identified security vulnerabilities. User authentication for the Node-RED was configured, creating credentials (user id and password) to the admin page and the Node-RED dashboard, consequently securing the access to the editor using a login screen to connect to these

pages, as shown in Figures 4.20 and 4.21, respectively. However, even with the user authentication, a sniffer program can easily obtain both user's and password's information, as it works over HyperText Transfer Protocol (HTTP). The Secure Sockets Layer (SSL) and Transport Layer security (TLS) protocols were used to encrypt the communications by acquiring keys and certificates, making the Node-RED server to work over HyperText Transfer Protocol Secure (HTTPS).

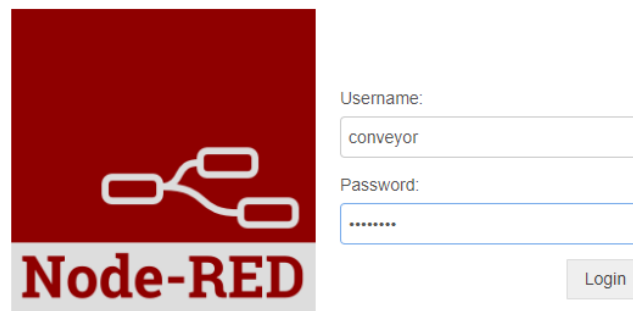
The image shows the Node-RED login interface. On the left is the Node-RED logo, which consists of a red square with a white flow diagram and the text "Node-RED" below it. To the right of the logo is a login form with two input fields: "Username:" containing the text "conveyor" and "Password:" containing a series of dots. A "Login" button is positioned to the right of the password field.

Figure 4.20: Login screen to access the Node-RED admin page.

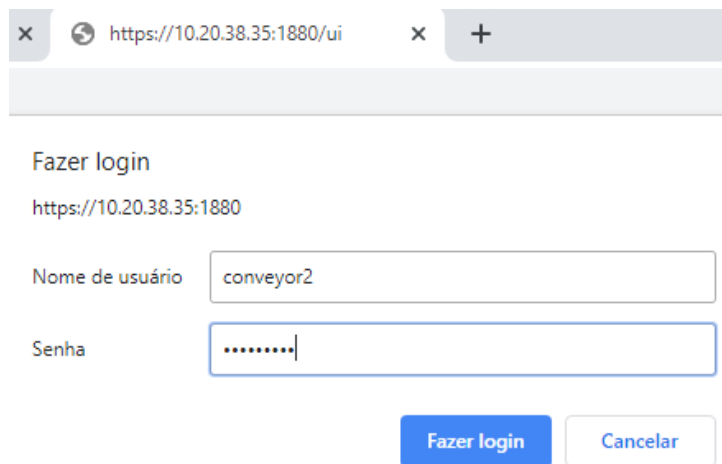
The image shows a browser window with the URL "https://10.20.38.35:1880/ui". The page content is in Portuguese and titled "Fazer login". Below the title is the URL "https://10.20.38.35:1880". There are two input fields: "Nome de usuário" containing "conveyor2" and "Senha" containing a series of dots. At the bottom are two buttons: "Fazer login" (blue) and "Cancelar" (white).

Figure 4.21: Login screen to access the Node-RED dashboard.

For the MQTT broker, these problems can be solved by requiring the user authentication to the broker and securing communications through TLS. The first one can be done just modifying the configuration file of the broker to deny anonymous connections to the broker and signing up allowed clients. TLS provides the identification and encryption

to secure communications. It is common to use external Certification Authorities (CA) that provides useful information to trustingly identify the server (broker when talking of MQTT) before the communication begins. It is also possible to use a cheaper solution: creating keys and certificates for the broker and clients self-signed with a local CA using the open-source tool OpenSSL [48].

Chapter 5

Online Monitoring

This chapter aims to present the Digital Twin developed that is performing in a Node-RED dashboard the dynamic and real-time monitoring of the health condition of a self-organized conveyor transfer system. In addition, it presents the storage of all data collected from the physical system, the alerts developed to detect failures in advance and the automatic sending of these alerts by email to the person responsible for monitoring this system.

5.1 System Virtualization in Node-RED Dashboard

The UI in Node-RED is organized in tabs and groups. Tabs are different pages in the UI, like several tabs in a browser. Inside each tab, some groups divide them into different sections, allowing better organizing the widgets provided by the dashboard nodes. Figure 5.1 represents the virtual model of the conveyor transfer system using the Node-RED platform being displayed on the main interface called “Conveyor”, where the collected data, as well as the aggregated statistical information, is visualized in different groups.

On the top left of the dashboard there are four displays showing the current status of the conveyor modules, with LEDs indicating if the motor is operating (green color) or not (red color), and if the input and output sensors are activated (green color) or not (red color). The letters are indicating the sequence of the conveyor modules in the physical conveyor system, being dynamically updated when a change in the sequence of conveyor

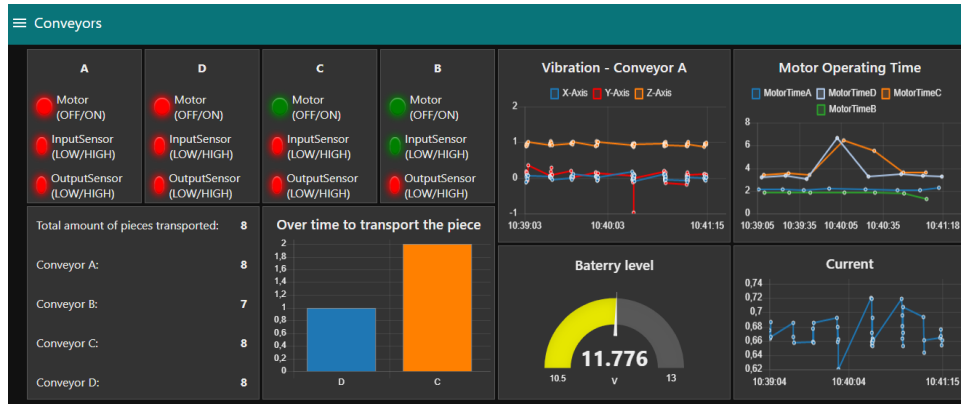


Figure 5.1: Dashboard for the virtual model of the conveyor system supporting the real-time monitoring.

modules occurs.

The data from the accelerometer was collected to monitor the conveyor system condition and its behavior when it is working, and the values acquired for the X, Y and Z axes are published in the vibration chart. In the same manner, the motor operating time chart shows the time in seconds that the motor of each conveyor module works to transport a part.

The bottom left presents the total amount of transported parts, regardless of how many conveyors are working, and the total amount of transported parts for each conveyor. If the time to transport the part in a conveyor is higher than the normal, it will be indicated in the bar chart, which shows how many parts had problems in each conveyor during a period of work. The battery voltage is presented in a graph with defined maximum and minimum values, making easier to perceive when it should be changed so that the system's operation is not affected. The last chart monitors the current values, i.e., how much current is being consumed during the system operation.

The Conveyor tab presented was created to allow the user to find the main information about the functioning of the system clearly and objectively. This presentation with a condensed of information in the main interface makes some graphs present many curves, generating difficulty for the user to perceive in detail the numbers marked, e.g., to verify the occurrence of an increase in time or vibration but not to find the value referring to

this peak due to the number of values being shown. Thus, secondary tabs were created to display separately each curve for a variable and, if necessary, the user can have more detailed information. These tabs are shown in Figures 5.2 and 5.3.

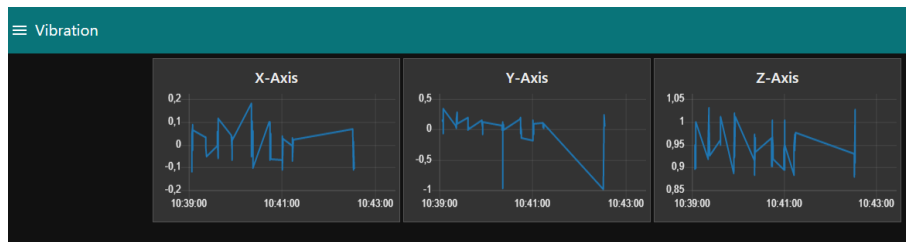


Figure 5.2: Dashboard for the analysis of vibration in the three axes separately.

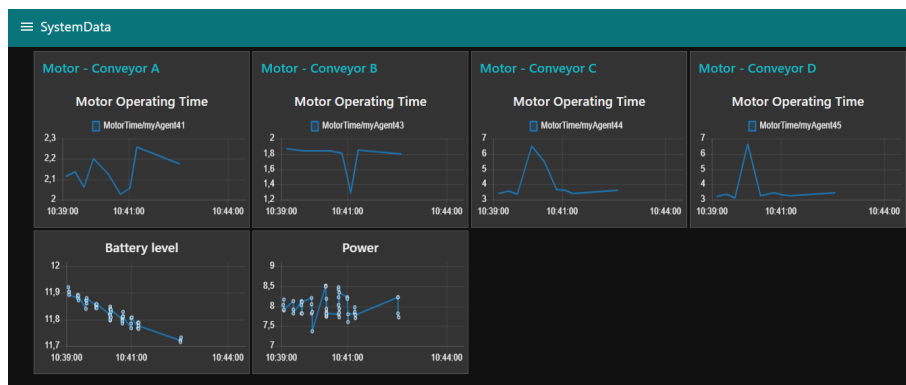


Figure 5.3: Dashboard for the monitoring of the motor operating time, battery level and power.

The Vibration tab in Figure 5.2 presents three charts referring to the vibration obtained for each axis. The SystemData tab of the Figure 5.3 presents charts for the total operating time of the four motors, in addition to presenting for the first time the chart with the power values and presenting a battery voltage in a chart allowing the user can follow the curve of its decay.

5.2 Database Storage

As presented during the description of the data flow diagrams described in item 4.2, the data collected from the system is stored in the MySQL database called “conveyor”. For

this, 14 tables were created, one for each variable being collected, and in these tables, columns were created in order to organize the data being received. Table 5.1 presents a sample of stored data for the current to exemplify how the tables in the database were built.

Table 5.1: Sample of data referring to the current value being stored in the database.

CurrentA		
ID	Current	Date
1	0.671973	2019-11-13 11:19:55
2	0.645996	2019-11-13 11:19:55
3	0.690332	2019-11-13 11:20:03
4	0.653906	2019-11-13 11:20:03

A primary key was defined for each table, the “ID”, a parameter that serves as a reference index and is responsible for not allowing values to be repeated in the same table, is incremented with each new insertion of values in the database. This column is extremely important for Rule 2 applied for failure detection, being used to return the last data received concerning the variable under analysis to verify the existence of a Trend. A second column, in the case of Table 5.1 called “Current” and in the other tables named according to the type of data being received, is used to store the values received from the system. The last column receives the local date and time when the data is received in the database.

The 14 generated tables were, namely, VibrationA_x, VibrationA_y, VibrationA_z, BatteryA, CurrentA, MotorTimeA, MotorTimeB, MotorTimeC, MotorTimeD, PiecesA, PiecesB, PiecesC, PiecesD, PowerA. These tables follow the same pattern as that exemplified through Table 5.1.

5.3 Early Failure Detection

In order to allow the real-time monitoring of the health condition of the self-organized conveyor transfer system and guaranteeing early detection of possible failures through the analysis of data collected from the physical system, two rules were proposed for the

analysis of its functioning. For that, considering the variables in which Rule 1 was applied, i.e. current, battery, vibration in the three axes and the time in the transfer of parts, the average values and standard deviations were calculated considering the normal system operation and these values are presented in Table 5.2.

Table 5.2: Mean and standard deviation values calculated for the application of Rule 1 for failure prediction.

	Current (A)	Battery (V)	Vibration_x (g)	Vibration_y (g)	Vibration_z (g)	Piece time (s)
\bar{x}	0,6527	12,1302	0,0180	0,0796	0,9331	2,9756
σ	0,0279	0,5691	0,0524	0,0818	0,0562	0,4792

The values presented in the table were used to define the alerts, which are triggered when a variable presents values higher than 2σ in relation to the average. In these cases, the outlier alert will be detected if the current value exceeds by 8,5% the average value, 12,1% for the acceleration on the z-axis and 32,2% for the time to transport the piece. The notifications generated on the dashboard for each of these variables are shown in Figures 5.4, 5.5 and 5.6.

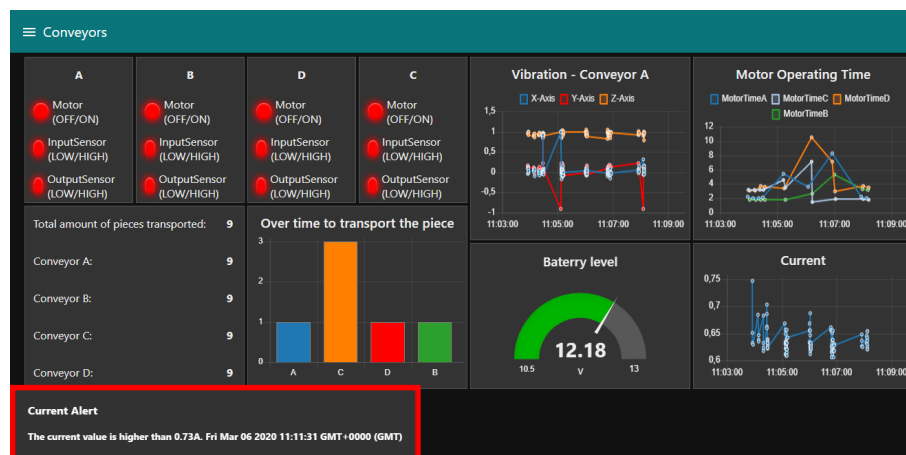


Figure 5.4: Current notification applying Rule 1 to detect failures in advance.

In the case of vibration on the x and y axes, it is observed that the standard deviation is higher than the average value and alarms would be triggered if the values of these variables exceeded by 580,6% and 205,6% the average value, respectively, requiring a very

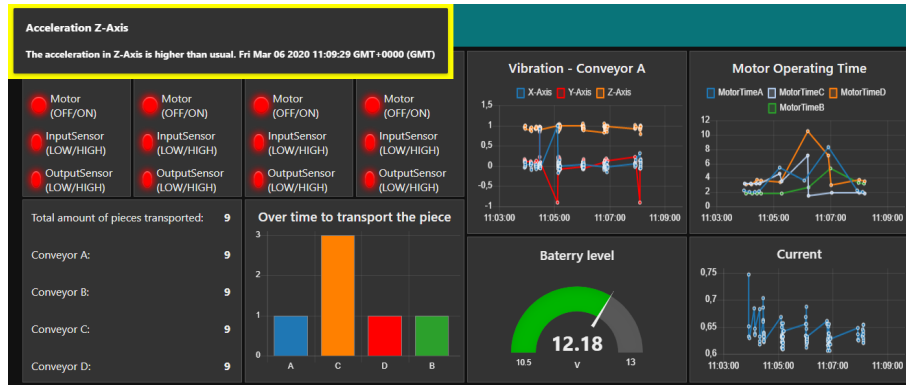


Figure 5.5: Vibration notification applying Rule 1 to detect failures in advance.

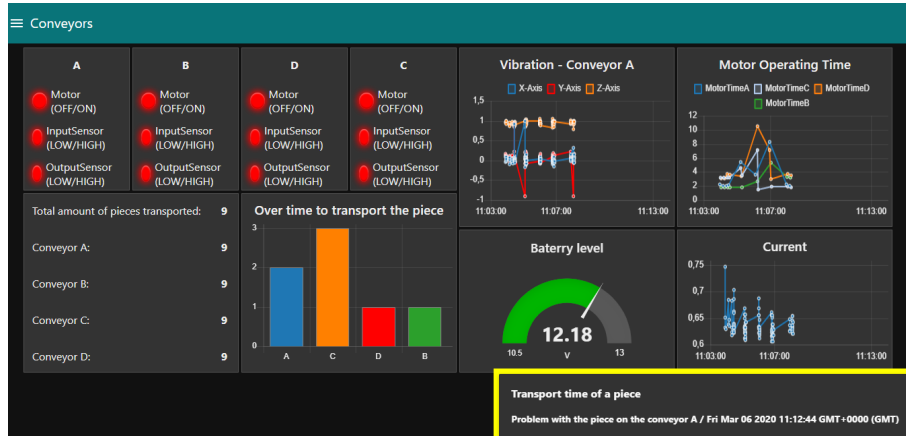


Figure 5.6: Notification from Rule 1 to inform a transport time of a piece higher than usual.

significant increase or decrease in their values in relation to the average to generate the notification. These two high values compared to the value of 12,1% obtained for the z-axis allow to conclude that the analysis of vibration in this last axis is more relevant than the other two since a smaller variation in relation to the average could already cause an outlier. Thus, the vibration notification was created only for the z-axis.

However, for the battery voltage, it was observed that when reaching the value of 10.5V (3σ), the voltage supplied was no longer sufficient to maintain the system functioning. Thus, two alerts were created concerning its level, the first being “*Low Battery*” when it reaches 11.5V (1σ), indicating that its level is getting low and will need to be replaced as soon as possible, and the second “*Battery is too low*” is generated if the system continues

to operate on the same battery when it reaches 11V (2σ), indicating that the battery should be replaced immediately to ensure system integrity. Besides, as battery monitoring is extremely important to avoid sudden system shutdown, notifications generated on the dashboard do not disappear after a while like the others, requiring the user to select "ok" to remove it to ensure this notification to be viewed. Figures 5.7 and 5.8 show the two types of notifications generated.

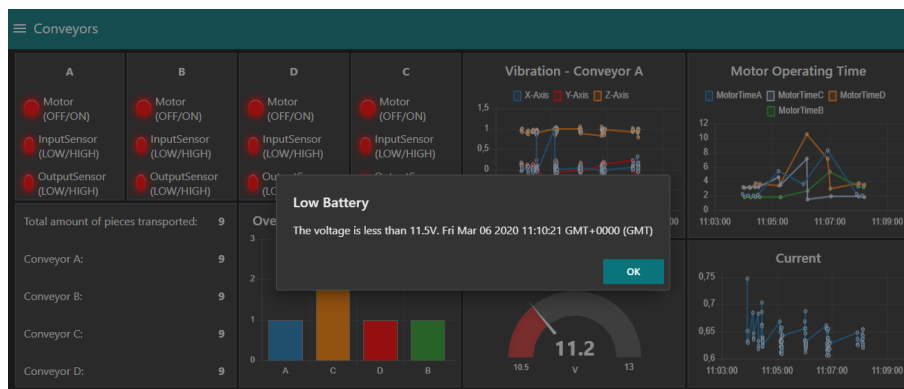


Figure 5.7: First battery notification concerning its level.

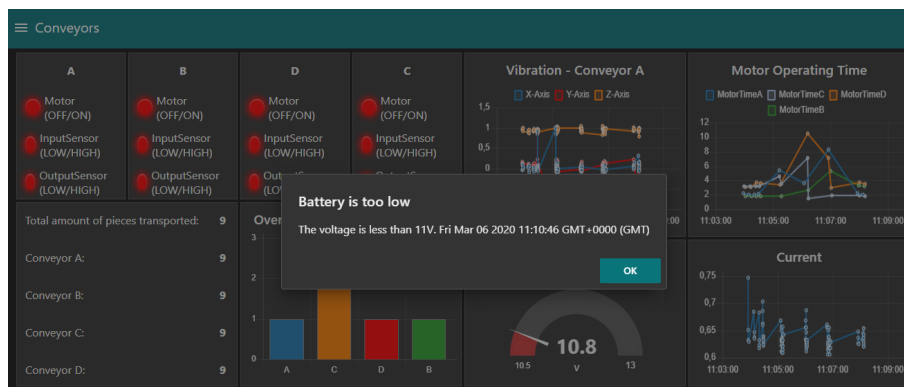


Figure 5.8: Second battery notification concerning its level.

The notifications generated from the detection of a trend concerning the values of the variables under analysis, i.e. current, vibration and motor operating time, as proposed by the application of Rule 2, follow the same model as the notifications presented generated from Rule 1 and are shown in the same places on the dashboard. However, these notifications present the topic “Increasing trend” or “Decreasing trend” according to the type

of trend identified, and the message indicates which variable this trend refers to. Figure 5.9 illustrates a notification for Rule 2 being applied to the current variable, over which an increasing trend has been identified.

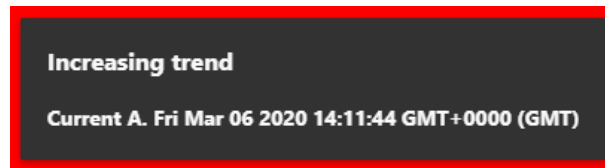


Figure 5.9: Increasing trend detected by applying Rule 2 for current monitoring.

In addition to these notifications generated as a way of detecting failures in advance based on the application of the process control methods analyzed, a notification was also generated when a change in the order of the conveyor modules is detected. This notification is shown in Figure 5.10.

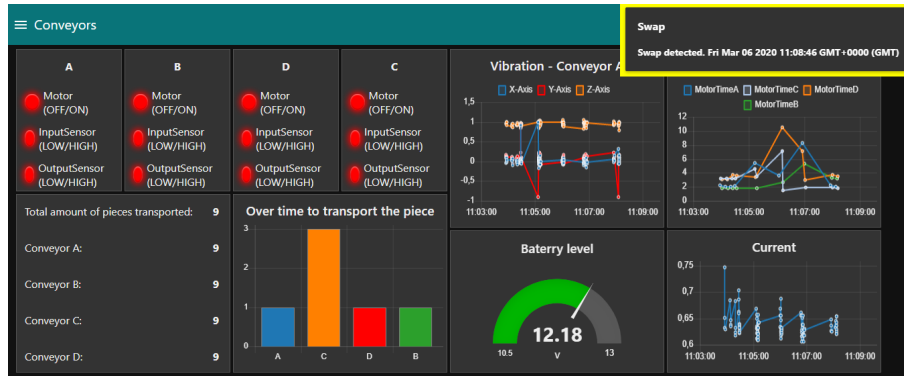


Figure 5.10: Swap notification.

5.4 Automatic Alert by Email

As presented in section 4.2, in addition to generating notifications via a pop-up on the dashboard when an alert situation is detected, emails are also sent to the responsible for supervising the conveyor system informing the occurrence of a notification. The email sent by this application follows the structure of the email illustrated in Figure 5.11.

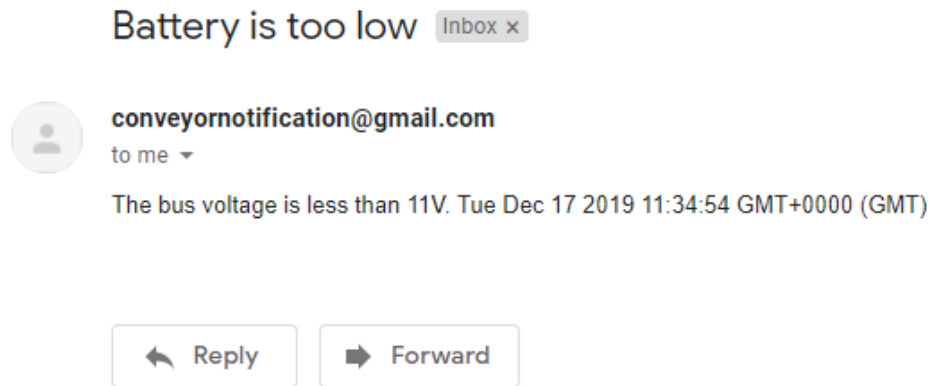


Figure 5.11: Structure of the email received when a notification occurs.

The subject of the email is built using the topic of the message sent by the email node in Node-RED, being in this case “Battery is too low”, and the body of the email is constructed in the payload of the message, i.e “The voltage is less than 11V. Tue Dec 17 2019 11:34:54 GMT+0000 (GMT)”. The other notifications send emails with a structure similar to the one presented according to the variable being monitored.

Receiving notifications also by email is advantageous because, if the user for some reason does not see the notification generated on the dashboard, it will be registered in the inbox, preventing the system’s alert information from being lost.

Chapter 6

Conclusion and Future Work

The modernization of the industry related to the concept of industry 4.0 makes it more productive, rich, efficient, flexible, and globally competitive. The interconnection of machines, production systems and equipment to the internet allows dynamism and the use of data in an intelligent way, stimulating the creation of differentiated services. Digital Twin is becoming popular in this context, allowing the creation of digital copies of the physical systems that are connected and share the functional and operational data, allowing to exercise of the virtual model to support the monitoring, diagnosis, prediction and optimization of the system operation to anticipate possible risks or condition changes.

The present work developed the virtualization of a self-organized conveyor transfer system aiming for the real-time monitoring of its health condition, using a Node-Red dashboard application, being the connectivity performed by using the MQTT protocol. This work also considers cyber-security issues related to the use of IoT technologies during the data collection, suggesting the implementation of simple solutions to prevent potential security threats, such as user authentication and the use of keys and certificates to encrypt communications based on the TLS/SSL protocol.

In terms of connectivity, the MQTT protocol, used to enable the data transmission from a physical twin to digital twin has good availability of open-source libraries, which makes it easy to implement, and has flexibility due its publish and subscribe model that makes it possible to support diverse application scenarios for IoT devices and services. In

terms of real-time monitoring, the collected data can be used smartly, to supervise the condition operation of the system and to detect anomalies and performance degradation in advance, supporting the diagnosis, prediction and optimization of the system operation to anticipate possible risks. Node-RED, despite being a tool that does not allow the 3D visualization of the system, was success used to show in a dynamic way the system's behavior, to display the real-time system data, as well as statistical information, to store data in a database and allow data analysis for early failure prediction by generating alerts through the implementation of control rules, ensuring the manufacturing interest in optimizing their process.

This work demonstrated the implementation of the Digital Twin concept through the interaction of the real production processes and the digital counterpart model, confirming its potentialities and benefits, ensuring remote monitoring, predictive maintenance, higher efficiency and security.

Future work will be devoted to add current, voltage and accelerometer sensors to other modules to ensure monitoring of the entire system, to apply some algorithms using big data and machine learning in the analysis of the real-time data collected to improve the diagnosis of a risk condition and predictive failure detection and to make other improvements related to the system's functionality, such as allowing the passage of more than one piece simultaneously and developing rules so that the conveyors can also work in clusters.

Bibliography

- [1] P. Leitão, J. Barbosa, G. Funchal, and V. Melo, “Self-organized Cyber-Physical Conveyor System using Multi-agent Systems,” *Accepted to be published in the International Journal of Artificial Intelligence*, 2020.
- [2] *Indústria 4.0 a Quarta Revolução industrial*, (visited on 11/02/2020). [Online]. Available: https://www.compete2020.gov.pt/noticias/detalhe/Industria_4ponto0.
- [3] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, “Industry 4.0,” *Business Information Systems Engineering*, vol. 6, pp. 239–242, Aug. 2014. DOI: 10.1007/s12599-014-0334-4.
- [4] M. Hermann, T. Pentek, and B. Otto, *Design principles for industrie 4.0 scenarios: A literature review*, Jan. 2015. DOI: 10.13140/RG.2.2.29269.22248.
- [5] *Industry 4.0 and how smart sensors make the difference*, (visited on 05/03/2020). [Online]. Available: <https://www.spectralengines.com/articles/industry-4-0-and-how-smart-sensors-make-the-difference>.
- [6] *Smart factory: a fábrica da Indústria 4.0*, (visited on 05/03/2020). [Online]. Available: <https://blog.infaimon.com/pt/smart-factory-fabrica-industria-4-0/>.
- [7] J. Lee, “Smart factory systems,” *Informatik-Spektrum*, vol. 38, no. 3, pp. 230–235, May 2015. DOI: 10.1007/s00287-015-0891-z.

- [8] F. Shrouf, J. Ordieres, and G. Miragliotta, “Smart factories in industry 4.0: A review of the concept and of energy management approached in production based on the internet of things paradigm,” in *2014 IEEE International Conference on Industrial Engineering and Engineering Management*, Dec. 2014, pp. 697–701. DOI: 10.1109/IEEM.2014.7058728.
- [9] L. D. Xu, E. L. Xu, and L. Li, “Industry 4.0: State of the art and future trends,” *International Journal of Production Research*, vol. 56, no. 8, pp. 2941–2962, 2018, ISSN: 1366588X. DOI: 10.1080/00207543.2018.1444806.
- [10] Germany Trade & Invest (GTAI), “INDUSTRIE 4.0 - Smart Manufacturing for the Future,” 2014.
- [11] C. K. Keerthi, M. A. Jabbar, and B. Seetharamulu, “Cyber physical systems (cps): Security issues, challenges and solutions,” in *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCCIC)*, Dec. 2017, pp. 1–4. DOI: 10.1109/ICCCIC.2017.8524312.
- [12] L. Sakurada, “Development of Industrial Agents in a Smart Parking System for Bicycles,” Master’s thesis, Polytechnic Institute of Bragança, Portugal, 2019.
- [13] A. Kusiak, “Smart manufacturing,” *International Journal of Production Research*, vol. 56, no. 1-2, pp. 508–517, 2018. DOI: 10.1080/00207543.2017.1351644.
- [14] J. Lee, B. Bagheri, and H. A. Kao, “A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems,” *Manufacturing Letters*, vol. 3, pp. 18–23, 2015, ISSN: 22138463. DOI: 10.1016/j.mfglet.2014.12.001.
- [15] P. Leitão, “Agent-based Distributed Manufacturing Control: A State-of-the-art Survey,” *Engineering Applications of Artificial Intelligence*, vol. 22, no. 7, pp. 979–991, Oct. 2009. DOI: 10.1016/j.engappai.2008.09.005.
- [16] W. Lepuschitz, “Self-reconfigurable manufacturing control based on ontology-driven automation agents,” PhD thesis, Apr. 2018.

- [17] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*. Wiley, 2007, ISBN: 978-0-470-05747-6. [Online]. Available: <https://www.amazon.com/Developing-Multi-Agent-Systems-Fabio-Bellifemine/dp/0470057475?SubscriptionId=AKIAIOBINVZYXZQZ2U3A&tag=chimbori05-20&linkCode=xm2&camp=2025&creative=165953&creativeASIN=0470057475>.
- [18] P. Leitão and S. Karnouskos, *Industrial Agents: Emerging Applications of Software Agents in Industry*. Mar. 2015, ISBN: 9780128003411.
- [19] J.-H. Lee and C.-O. Kim, “Multi-agent systems applications in manufacturing systems and supply chain management: A review paper,” *International Journal of Production Research*, vol. 46, no. 1, pp. 233–265, 2008.
- [20] B. Parasumanna Gokulan and D. Srinivasan, “An introduction to multi-agent systems,” in Jul. 2010, vol. 310, pp. 1–27. DOI: 10.1007/978-3-642-14435-6_1.
- [21] F. Tao, S. Member, H. Zhang, A. Liu, and A. Y. C. Nee, “Digital Twin in Industry: State-of-the-Art,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, 2019. DOI: 10.1109/TII.2018.2873186.
- [22] F. Pires, V. Melo, J. Almeida, and P. Leitão, “Digital Twin Experiments Focusing Virtualisation, Connectivity and Real-time Monitoring,” in *Accepted to be published in the proceedings of the IEEE International Conference on Industrial Cyber-Physical Systems (ICPS 2020)*, 2020.
- [23] *5 trends emerge in the gartner hype cycle for emerging technologies, 2018*, (visited on 05/03/2020). [Online]. Available: <https://www.gartner.com/smarterwithgartner/5-trends-emerge-in-gartner-hype-cycle-for-emerging-technologies-2018/>.
- [24] *Prepare for the Impact of Digital Twins*, (visited on 05/03/2020). [Online]. Available: <https://www.gartner.com/smarterwithgartner/prepare-for-the-impact-of-digital-twins/>.

- [25] F. Pires, A. Cachada, J. Barbosa, A. P. Moreira, and P. Leitão, “Digital Twin in Industry 4.0 : Technologies , Applications and Challenges,” in *Proceedings of the IEEE International Conference on Industrial Informatics (INDIN’19)*, 2019, pp. 721–726, ISBN: 9781728129273.
- [26] A. Parrott and L. Warshaw, “Industry 4.0 and the digital twin,” *Deloitte University Press*, pp. 1–17, 2017.
- [27] *FIPA, FIPA ACL Message Structure Specification*, (visited on 20/02/2020). [Online]. Available: http://www.fipa.org/specs/fipa00061/SC00061G.html#_Toc26669700.
- [28] *IBM, Getting to know MQTT*, (visited on 20/02/2020). [Online]. Available: <https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/>.
- [29] G. Cherradi, A. Bouziri, and A. Boulmakoul, “Smart data collection based on iot protocols,” Dec. 2016.
- [30] S. Lee, H. Kim, D.-K. Hong, and H. Ju, “Correlation analysis of mqtt loss and delay according to qos level,” Jan. 2013, pp. 714–717, ISBN: 978-1-4673-5740-1. DOI: 10.1109/IC0IN.2013.6496715.
- [31] P. Bastos, I. Lopes, and L. Pires, “Application of data mining in a maintenance system for failure prediction,” in *Safety, Reliability and Risk Analysis: Beyond the Horizon - Proceedings of the European Safety and Reliability Conference, ESREL 2013*, 2014, pp. 933–940, ISBN: 9781138001237. DOI: 10.1201/b15938-138.
- [32] *Predictive Maintenance*, (visited on 01/02/2020). [Online]. Available: <https://www.accelix.com/community/predictive-maintenance/predictive-maintenance-explained/>.
- [33] M. A. Razzaque, M. Milojevic, A. Palade, and S. Clarke, “Middleware for internet of things: A survey,” *IEEE Internet of Things Journal*, vol. 3, pp. 1–1, Jan. 2015. DOI: 10.1109/JIOT.2015.2498900.

- [34] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of things: A survey on enabling technologies, protocols, and applications,” *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, Fourthquarter 2015, ISSN: 2373-745X. DOI: 10.1109/COMST.2015.2444095.
- [35] F. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*. Wiley, 2007.
- [36] J. Teixeira, “Desenvolvimento de um Sistema de Transporte Modular com Auto-Organização de Componentes Ciber-Físicos,” Master’s thesis, Instituto Politécnico de Bragança, 2016. [Online]. Available: <http://hdl.handle.net/10198/14079>.
- [37] *OLED Display*, (visited on 20/02/2020). [Online]. Available: <https://learn.adafruit.com/adafruit-oled-displays-for-raspberry-pi>.
- [38] *ADXL345 Digital Accelerometer*, (visited on 20/02/2020). [Online]. Available: <https://www.sparkfun.com/datasheets/Sensors/Accelerometer/ADXL345.pdf>.
- [39] *Adafruit INA219 Current Sensor Breakout*, (visited on 20/02/2020). [Online]. Available: <https://cdn-learn.adafruit.com/downloads/pdf/adafruit-ina219-current-sensor-breakout.pdf>.
- [40] *Using an I2C OLED Display Module with the Raspberry Pi*, (visited on 11/02/2020). [Online]. Available: <https://www.raspberrypi-spy.co.uk/2018/04/i2c-oled-display-module-with-raspberry-pi/>.
- [41] *Node RED Programming Guide*, (visited on 19/02/2020). [Online]. Available: <http://noderedguide.com/>.
- [42] *Eclipse Paho Java Client*, (visited on 20/02/2020). [Online]. Available: <https://www.eclipse.org/paho/clients/java/>.
- [43] *Mosquitto, Eclipse Mosquitto An open source MQTT broker*, (visited on 20/02/2020). [Online]. Available: <https://mosquitto.org/>.
- [44] *Node-RED User Guide*, (visited on 19/02/2020). [Online]. Available: <https://nodered.org/docs/user-guide/>.

- [45] L. S. Nelson, “The Shewhart Control Chart-Tests for Special Causes,” *Journal of Quality Technology*, vol. 16, no. 4, pp. 237–239, 1984. DOI: 10.1080/00224065.1984.11978921. eprint: <https://doi.org/10.1080/00224065.1984.11978921>.
- [46] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, “Future internet: The internet of things architecture, possible applications and key challenges,” in *2012 10th International Conference on Frontiers of Information Technology*, Dec. 2012, pp. 257–260. DOI: 10.1109/FIT.2012.53.
- [47] A. Whitmore, A. Agarwal, and L. Xu, “The internet of things—a survey of topics and trends,” *Information Systems Frontiers*, vol. 17, pp. 261–274, Apr. 2014. DOI: 10.1007/s10796-014-9489-2.
- [48] J. D. Morenas, C. Miller, G. S. Funchal, V. Melo, M. Vallim, and P. Leitao, “Security Experiences in IoT based applications for Building and Factory Automation,” in *Proceedings of the IEEE International Conference on Industrial Technology (ICIT 2020)*, 2020.