POLITECNICO DI TORINO

Master degree course in ICT for Smart Societies

Master Degree Thesis

Development of a web application for industrial IoT machine monitoring and data visualization

Supervisors
Giulia Bruno
Franco Lombardi
Emiliano Traini

Candidate
Salvatore Giuseppe Bellino
matricola: 265465

Accademic Year 2020-2021

Summary

The Internet of Things (IoT) enables to connect heterogeneous electronic devices in an efficient, sustainable and continuous way, allowing a level of human-to-machine and machine-to-machine interaction never reached before. These devices can be exploited to collect a large amount of data, which in turn allow making decisions and driving other electronic devices for a wide variety of applications. Thanks to the technologies developed in the field of Big Data, the collected data can be stored and automatically processed in order to support decisions, improve processes, and improve the quality of services and products. With the advent of Industry 4.0, IoT technologies have been increasingly applied in the industrial sector. In this way, industrial machines can be always connected, and equipped with many devices, which enable the interaction with the outside world. In this context, new applications can be developed, such as predictive maintenance and automatic quality control.
The aim of the thesis is to firstly analyse the different approaches, protocols and technologies that can be used to exploit IoT in the industrial sector. Then, a Python web application has been developed to manage the collection, storage and visualization of data gathered from sensors positioned on an industrial machine. The real data collected by a milling machine has been used as case study.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The term Internet of Things (IoT) was used for the first time in 1999 by Kevin Ashton [1] , to describe an hardware and software system where the Internet is connected to the physical world via ubiquitous sensors. As he himself pointed out: “In the twentieth century, computers were brains without senses, they only knew what we told them. In the twenty-first century, because of the Internet of Things, computers can sense things for themselves" [1]. The concept of Internet of Things refers to the possibility of connecting electronic devices, sensors and actuators, in order to allow the exchange of data and commands between them and with the outside world, interfacing them to the Internet. The IoT enhances the capabilities and expanded the possibilities of information systems and automatic information processing. Before the advent of the IoT, software usually worked by storing and processing mostly user input. From the advent of intelligent electronic devices and the IoT, the world of automatic information has the possibility of acquiring data directly and automatically from the physical world, allowing a level of interaction between hardware and software capable of satisfying the needs of the cities of the future. This fills the gap between the physical real world and the artificial intelligence and computing capabilities provided by software and computers, opening new scenarios, challenges, applications, business models and markets.
A process therefore begun and even everyday objects have been augmented through micro-controllers, electronics to communicate, communication protocols, sensors, actuators and they have been connected to the Internet, allowing them to be automatically monitored by a software and to be reachable remotely by the user. These objects became smart objects. Depending on the application, the networks formed by these smart objects can be small

or large, characterized by different topology, protocols used, type of communication, type of power supply, computational and memory capabilities.
However, there are a large number of applications and use cases where networks formed by nodes distributed in space are needed and where energy availability is limited. To reduce the energy consumption and the cost of these electronic devices, the hardware of these nodes is typically very simple and limited to what is strictly necessary to acquire the data of interest and transmit them to a more intelligent central node with greater computational capacity, availability of memory and energy. This phenomenon is often referred to as Edge Computing [2]. The Edge Gateway, closer to the edge of the network, where the data is collected, can also filter and aggregate the incoming data from the smart objects, and then it can forward these data to Big Data storing and processing systems. The fact that the Edge Gateway can filter and aggregate the data can reduce the network bandwidth required by the IoT system and the overall latency of the acquisition and processing phase [2].
In simpler applications, the data collected by smart objects is already very informative itself and it is only stored and displayed in a way that it is understandable by the user, often through web applications or mobile applications. The data is then directly consumed by the end user, who extracts the information content and use it to solve a problem. However, there are many applications where the data collected are too large to be processed by the user or they are not very informative themselves. These Big Data are usually processed asynchronously by specific software. The simplest processing is filtering and aggregation, but machine learning, statistical analysis and artificial intelligence algorithms of considerable complexity can also be applied. These algorithms extract from the initially uninformative raw data an high information content immediately exploitable by the end user in the real world or by other connected smart devices in the cyberspace. Knowing what to measure and what type of data processing to apply is one of the biggest difficulty when developing an IoT solution.
For this reason, from an operational point of view, the approach is often systematic and sequential. Initially the data of interest and the necessary sensors are defined. The nodes are then deployed, interconnected and interfaced with the Internet, using protocols and tools that can be implemented quickly but are not very efficient from an energy consumption point of view [3]. The data are also visualized through web or mobile applications developed by applying fast prototyping techniques, but not processed. A possible and diffused approach is prototyping dashboards using Grafana [**Grafana**]

or similar tools. During this phase the prototype is proposed to potential customers also presenting some possible data processing strategies that can be implemented and how the system can support decisions and help to solve the problem of interest. Once the feedback are collected, at a later stage, if the prototype has generated sufficient market interest and the risks and costs have been assessed, the data processing modules are developed in detail, possibly training machine learning models that use the raw data, identifying hidden patterns and trends [4].
Cities are complex systems, where a large amount of data is generated and can be collected and processed to improve the citizens' quality of life, the efficiency and monitoring of processes typical of highly urbanized areas [5] . In this context, the application of the most modern data collection and processing technologies through IoT solutions is leading cities to be increasingly smart and interconnected with the cyberspace. The IoT found applications in different sectors that together constitute the concept of smart city [6] : home automation, smart electrical grids, smart agriculture, smart healthcare, automatic traffic management, smart mobility, waste management, smart factories.

1.1 Industrial control systems

Industrial control systems are characterized by both hardware and software components and their purpose is the control of industrial processes. These systems can be small, consisting of a few controllers or very large, even consisting of thousands of nodes distributed throughout the plant. The control logic can be centralized, distributed or hybrid. Industrial control systems were initially composed of programmable logic controllers (PLCs) that implemented simple control logic, but over the years they have evolved into more complex systems, such as supervisory control and data acquisition systems (SCADA) or distributed control systems (DCS).

1.1.1 Supervisory Control and Data Acquisition systems

SCADA are Industrial Control Systems (ICS) that have been used in order to monitor complex and geographically distributed equipment, machines and processes, replacing the manual intervention of operators, which is expensive and time-consuming. The SCADA central controllers are connected to the

actuators and sensors through intermediate nodes, which can be PLCs or Remote Terminal Units (RTUs). The SCADA software provide data collection, reporting, recipe management, alarming and alerting in case of relevant events or dangerous situations. SCADA systems allows to control industrial processes and machines with minimal human intervention, forwarding the commands directly to the actuators. SCADA systems can perform business logic, changing the state of machines and their operational parameters in order to provide self-healing of the individual components and reduce downtime. The human-machine interface (HMI) provided by the SCADA software let data and commands available to operators. The primary purpose of using SCADA is remote monitoring and control of field sites through a centralized control system. Instead of workers having to travel long distances to perform tasks or gather data, a SCADA system is able to automate this task. Field devices control local operations such as opening or closing of valves and breakers, collecting data from the sensor systems, and monitoring the local environment for alarm conditions.

1.1.2 Distributed control systems

Distributed control systems (DCS) consists of geographically distributed control units called controllers. In DCS, unlike traditional SCADA, the control authority is decentralized and distributed. Each controller controls a single process or machine. In a DCF there are many local controllers that communicate via a high speed local area network. Because of the distributed nature, DCSs are more fault-tolerant. DCF systems have spread to replace analog and pneumatic controllers in medium and large plants.

1.1.3 Manufacturing Execution Systems and others decision support tools

Manufacturing execution systems (MES) are information systems exploited in industrial processes and manufacturing to track the path that leads the raw materials to be transformed into semi-finished products first and finished products later [7] . MES usually make use of a large amount of complex and persistent data gathered from different sources. Because of they works with so much data, they usually are composed by numerous different user interface screens. MES can orchestrate a lot of small processes involved in the manufacturing and in facts complex business logic is typically implemented. Essentially, they are large and complex Enterprise Applications [8] . Today

they are generally implemented as web applications because they can be accessed remotely using any browser-enabled device. MES operate across multiple functional subsystems, including:

- Product definitions and catalog across their entire life cycle from raw material to finished product
- Physical and human resources allocation and scheduling
- Traceability
- Supplier and customer database
- Machine, equipment, working tools database
- Life cycle of the production order from the customer to the scheduling on the machines
- Management of warehouse movements and inventory
- Maintenance of machines and other tangible assets
- Employee database
- Downtime management for overall equipment effectiveness (OEE)
- Machine data collection and performance monitoring

Some of these function can require many information and can be very complex to be implemented. For example, some MES have the ability to translate the customer orders to production orders in the machines, considering employee and equipment availability, production quantities, parallelization and aggregation opportunities and other factors in order to optimize the production cost and maximize the generated outcome. MES can be very useful in the most regulated industries where every event, action and phase of the production process must be logged. Due to their transversal nature, MES systems can integrate functions typically covered by Enterprise Resource Planning (ERP) systems and vice versa. The line that divides them is in fact not uniquely defined. However, ERP systems are generally oriented to the management of accounting and administrative aspects [9] , while MES are developed ad-hoc to manage production processes in the plant [7].

Usually organizations use two different software, one ERP and one MES, eventually developed by different companies and deployed at different moments. The MES software is almost always installed after the ERP, as it

becomes particularly useful for medium-large companies, while the ERP software is almost always indispensable even in very small companies, for example to meet regulations that may concern the issuance of electronic invoices and other forms of IT interaction with the public administration. Because many of the data necessary for the functioning of the MES, such as the data of employees, suppliers and customers, as well as customer orders are already stored by the ERP which is manually populated by the operators, the two software are put in communication and interconnected, using any of the integration patterns used in enterprise applications [8].

For this reason, ad-hoc integration and communication modules often need to be developed, tested and maintained. This often requires the cooperation of both parties, who are not always equally available. Today, modern ERPs and MESs, natively expose documented public APIs that can be called from other information systems and applications, therefore the integration is less expensive.

These systems have role-based authentication, so that specific software features can only be used by operators who actually have the authority and know-how to do so. For example, for quality control operators, specific sections of the software are enabled, through which rejects and anomalies can be recorded in order to take corrective actions afterwards.

In some MES, through mobile devices positioned on the machines and in other strategic positions of the plant, operators can monitor the progress of the production process, register new data, take some actions. These devices, usually tablets or notebooks, are interconnected to the MES application via LAN or via Internet, if the MES is installed in the cloud. The information flow can be bidirectional, reaching a level of interaction with the industrial control system that was impossible with SCADA and DSC systems.

1.1.4 Industry 4.0 and Smart factories

With industry 4.0 begins a process of intensive digitization of production processes and factories, in their entirety made up of physical, electronic and human resources.

The fundamental principle of Industry 4.0 are connected and distributed objects in space. Many of these objects are cyber-physical systems (CPSs), capable of putting the real world in communication with the cyberspace. They are implemented through typical IoT solutions and are capable to support and orchestrate complex operations, as well as provide a large amount of

data for information systems. The operators and other members of the manufacturing organization play an important collaborative role in relation to this connected system, being in a certain sense also cyber-physical systems, acting as a link between the physical world and the software.

These IoT assets can augment the capabilities of machines that originally did not have the ability to communicate and cooperate with other systems.

By combining IoT and big data analytics, factories become smart, using new smart features, functions and services, including:

- Predictive maintenance and equipment failure forecasting
- Resource overload forecasting and avoidance
- Automatic product quality control and non-conformities detection
- Predictive demand
- Machine and equipment dynamic reconfiguration at runtime
- Augmented reality
- Smart working
- Smart logistic

Many of these features are not possible or are highly inefficient without using this new approach. For example, by collecting data related to product quality and mechanical instability of machines and tools, a MES can derive useful information on the health and maintenance status of machines. Without the use of interconnected sensors, this type of evaluation could only be carried out with manual methods by the operators. However manual surveys and evaluations are obviously discrete events over time and do not always manage to be useful and bring a significant gain of time, money and resources, as they are conducted too early, too late or too infrequently. Instead, applying the IoT these processes not only become automatic, but potentially also continuous and can provide updated information in real time to the operators and especially to other cyber-physical systems.

On the other hand, the advent of the IoT philosophy has pushed manufacturers of manufacturing equipment to produce machines that are already natively connected. These machines typically expose APIs that can be used by other computer systems, for example MES. They use different protocols, some vendor-specific, generating complications regarding interoperability and

highlighting the clear need for universally accepted standards and protocols, a point on which a lot of work has been done in recent years. These machines often provide both punctual and integral information, for example by showing the number of pieces produced or rejected since the beginning of the processing or more or less detailed information on the current processing. However, it is also possible to further increase the capabilities of these machines by mounting other IoT devices, sensors and actuators in them.

In addition to the IoT devices used for monitoring the machines and the production process, other networks of nodes can be installed in order to monitor accessory systems, for example regarding the consumption of electricity or the ventilation and air conditioning system of the plant. These nodes can collect important data in order to reduce energy consumption and therefore costs.

The MES, which until a few years ago were typically software without interactions with other electronic devices, have had to adapt to the new demands arising from the use of the IoT and the huge amount of data they can generate. The possibility of being able to communicate directly with the physical world and with machines has made it possible to develop new functions.

Now the product life cycle can not only be monitored, but potentially also orchestrated, sending instructions to machines at the right time to put orders coming from ERP systems into production, configuring the machines automatically. The data collected can be shown to the operator through the modern HMIs provided by the MES, in the form of alarms, events, states, trends, predictions. The operator is therefore informed in real time on the status of the production process and on any anomalies and supported in decisions.

MESs are therefore becoming more and more complex and useful, also acquiring functionalities that were previously implemented in SCADA systems [7]. Today the MES are often integrated with the SCADA systems already present in the plant and with other information systems such as the company ERP, centralizing the control of business and production processes.

1.2 Predictive maintenance

One of the applications made possible by the technological innovations introduced in industry 4.0 is certainly predictive maintenance. In order to adopt predictive maintenance strategies, it is necessary to periodically monitor the

status of the machines and tools through condition monitoring (CM) plans. Condition monitoring is important because it allows you to minimize the risk of downtime and failures. Modern machines, designed for industry 4.0, already include sensors and expose data. Older machines, on the other hand, must be enriched with sensors and actuators networks. The values measured by the sensors during machining can also be used to predict information about the wear of the tools, becoming in effect tool condition monitoring (TCM) systems. In machines where auxiliary tools are used intensively, these systems are very important because they allow you to replace the tool at the right time, when it is worn enough to no longer work accurately, but in any case before it breaks completely. An example are the milling machines, which mount cutting tools that wear out and become ineffective.

1.3 Objectives

The first objective of the thesis is to list the protocols and technologies that can be used in the factories of tomorrow to create large cyber-physical systems. In fact, today there are a large number of protocols and technologies, which make the choice complex and generate interoperability problems. The applicability of the different protocols to different scenarios will be evaluated considering the delay and reliability constraints present in many of the applications in the field of industrial control and monitoring. The second goal is to develop a software framework for data collection, storage and visualization for predictive maintenance applications.

Chapter 2

Communication technologies and protocols

Before the diffusion and the consolidation of wireless technologies, industrial communication systems consisted of a network of cables that physically connected electronic devices and machines. The most used technologies are Ethernet and Fieldbus. These systems are still very popular today, although they will likely be replaced by industrial wireless sensor and actuators networks (IWSAN) in the future. Backward compatibility and interoperability with these legacy technologies is often a necessity.

2.1 Industrial communication systems requirements

Industrial communication systems are used for many applications and operational needs. The International Society of Automation has proposed a classification of industrial systems into three categories and six classes based on the nature of the system, the characteristics of urgency, the maximum tolerable latency, the methodology of accessing the resource, the methodology of carrying out the operation [10]. The three categories are elencate below.
Safety / Emergency systems: these systems deal with monitoring and control in emergency conditions or in conditions that can compromise the safety of people and things, such as during a fire. When you are in an emergency situation it is important to act in the shortest time possible [11], which is why in these systems they require very low latency, in the order of milliseconds.

The required reliability is obviously very high, whereas regarding to the network consumption they generally do not use a lot of traffic. The maximum tolerable delay is in the order of milliseconds.
Control system: a large portion of industrial equipment require continuous control. These systems can be classified into open or closed loop systems. Closed loop systems consist of sensors and actuators and are fully electronic and automated, while open loop systems require operator intervention. Closed loop electronic control systems have much lower delay tolerances [11] than open loop systems. Some of processes in this category are critical and a high delay or a malfunction can cause damage to the manufacturing process or to the industrial equipment. The tolerated delay can range from a few milliseconds to hundreds of milliseconds.
Monitoring: this category includes non-critical monitoring processes, for example the monitoring of process variables or metrics related to the quality and efficiency of the industrial process. Equipment health monitoring also falls into this category. The data collected by these systems can be processed a posteriori as the result of the processing never has immediate consequences [11]. Therefore, even larger delays are tolerated, in the order of minutes or hours.
It is therefore clear that in some industrial applications it is necessary to guarantee very low latency and very high reliability. Reliability can be evaluated by considering the percentage of packets not correctly received by recipients. It can therefore be measured with quantitative metrics and then optimized at the level of protocols and technologies, also applying optimal network and protocol configurations based on the installation site and field conditions. Latency and reliability can be improved by using deterministic communication protocols [12] that guarantee maximum channel access, data transmission and reception times.

Another important goal is the scalability. Networks must allow for the addition of a large number of nodes. In addition, they must reconfigure themselves at the topology and routing level automatically at runtime, providing also flexibility. This goal can clash with the goal of having a deterministic network, in fact many of the protocols at the MAC layer use time-based multiplexing schemes that impose limitations on the maximum number of nodes.
In any computer system, security is always a fundamental aspect. Especially in IoT networks, which can then expose the corporate network to the Internet, security must be guaranteed [13]. Some protocols used in sensor networks, especially wireless networks, do not implement very strong security

functions to limit protocol overhead [13]. In these cases, security must be guaranteed at the application level. Security can be improved by applying complex multiplexing schemes at the physical level.

2.2 Wired communication

The technologies that make use of physical cables have been used because they allow high speed and very low latency communications, they guaranteed good reliability and a deterministic behavior that can be modeled and predicted even in the worst scenarios [12] .

2.2.1 Industrial Ethernet

Initially, many companies independently developed various wired communication technologies, which later become standards [14]. These protocols often use serial interfaces. When Ethernet began to spread, it was considered convenient to adopt it in the industrial field to develop widespread standards that guaranteed high interoperability.
However, standard Ethernet uses a CSMA/CD approach [14]. After sensing the channel as idle, the nodes can transmit data. Hearing the channel during transmission, they can detect collision, because the transmitted signal must be equal to the eared signal. When a collision is detected, the transmission stops, and the station waits for a random period of time before it can re-transmit. These characteristics make networks probabilistic [15], in the sense that channel access, transmission and reception times cannot be guaranteed. This is a problem for industrial applications, which require very low latency times.
To solve this problem, Ethernet protocols have been developed using a MAC layer implemented in order to ensure determinism and low latency [12].

The most important and diffused industrial internet protocols are:

- PROFINET
- EtherCAT
- Powerlink
- Ethernet/IP

2.2.2 Fieldbus

Fieldbus systems adopt the bus concept. In computer science and electronics, a bus is in general a shared communication medium between that multiple nodes can use to communicate. Access to the bus is possible using a suitable bus adapter, which allows to write and read from the bus. This type of architecture has been applied to solve many problems, for example in the communication between components within a hardware system, or in the communication between processes in (micro)services-oriented software architectures [16]. In Figure 2.1 the architecture of a distributed system where a message bus is used is shown.

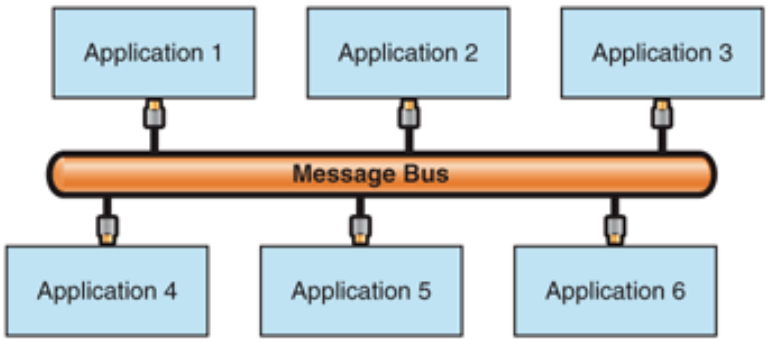

Figure 2.1: Distributed system architecture using a message bus. Image from [17].

Bus architectures can allow greater interoperability between components, greater scalability and elasticity of the network. In fact, only the adapter is needed to connect a new node to the bus. However, the bus is potentially a single point of failure and must support large networks and heavy loads.
In communication systems that use wired connections at the physical level, bus architectures can considerably reduce the costs of installation and maintenance of the cables, as well as the number of connections required compared to point-to-point connections. This has made bus systems very popular.

Some of the fieldbus-oriented protocols are [12]:

- PROFIBUS
- Foundation Fieldbus H1
- INTERBUS

However, even in this case, many companies have developed standards independently, generating interoperability problems and the need for universally accepted standards and technologies.

2.3 Wireless communication

2.3.1 Wireless industrial sensor and actuators networks challenges

The wireless channel is always shared and all communications at the physical layer must be considered broadcast communications. This implies that connections can collide with each other and makes wireless channels particularly susceptible to interference, noise, fast changes in operating conditions. Channel access and transport protocols are required to mitigate the inherent problems and limitations of the wireless channel.
Industrial environments are typically complex, consisting of many rooms, walls and electronic devices. The propagation conditions can therefore be very poor. There can be many sources of interference and noise.
Rapid variations of the operating conditions make delays unpredictable and the deterministic behavior of the network cannot be easily guaranteed. The number of nodes and the transmission powers must be optimized in order to reduce the probability of collision.
Reliability and very low latency must be guaranteed at the protocol level, as they are necessary for a multitude of industrial applications. But the physical functioning of the wireless channel makes it more difficult to develop protocols suitable for industrial conditions.
Wireless technologies can be used to create industrial wireless sensors and actuators networks (IWSANs) [18]. Using wireless technologies brings obvious advantages as regards the cost of installing and maintaining networks, since there is no need to use cables. These networks can consist of hundreds or thousands of nodes, typically battery powered. Therefore the energy consumption must be as low as possible and the batteries must guarantee the functioning of the node for years. The problem is even more evident if we consider the fact that the nodes can also be positioned in areas that are difficult to reach of the plant, for example inside the machines or in the ventilation systems of the plant. To increase the longevity of the network, energy harvesting technologies can be implemented. These solutions can exploit the mechanical conditions of the plant, for example vibrations. They can also

exploit differences in temperatures and light.

The need to increase reliability has pushed towards the use of time division multiple access (TDMA) schemes, so the need for synchronization of nodes is evident. Synchronization is typically guaranteed by transmission protocols.

Other techniques can improve reliability at the physical level, for example by using unrelated transmission frequencies and pseudo-random frequency hopping patterns. The concept of redundancy can be applied at various levels. At the frame level, error correction codes can be added to the frame. At the routing level, multiple paths must exist for each pair of nodes and possibly packets can be transmitted redundantly over multiple paths. Careful scheduling can minimize collisions and better distribute traffic by using different routes for simultaneous transmissions from different nodes.

In the upcoming paragraphs, some of the protocols that can be considered to implement IWSANs are discussed.

2.3.2 WiFi

The standards of the IEEE 802.11 family have been designed for home use and for the Internet [19]. These protocols guarantee very high transmission speeds, but they consume too much power and cannot be used for battery-powered devices. Moreover, antennas and WiFi modules are expensive.
The biggest problem is the probabilistic nature of these protocols, which cannot guarantee the determinism required by industrial control applications. The probabilistic behavior is due to the functioning of the MAC layer in this protocol family, based on the CSMA/CA approach. Using the distributed coordination function (DCF), transmissions can fail and be retried for a number of times, small delays cannot be guaranteed. Several modifications have been proposed at the MAC level [20] to improve the deterministic behavior of the protocol.

2.3.3 802.15.4

The 802.15.4 standard defines the operation of the low-rate wireless personal area network (LR-WPANs) [21]. It specifies the physical layer and MAC layer. At the physical layer it works on unlicensed industrial scientific and medical (ISM) radio bands and direct sequence spread spectrum (DSSS) is used. Depending at which frequency bandwidth it is used the chip code changes, in particular it is larger for 2.4 GHz. Because it specifies only the physical and MAC layers, protocol built on top of it must implement

routing functionalities. This has led to the development of many compatible routing protocols, each best suited to certain operating conditions and applications. This also means that a large number of types of protocols and therefore networks with different characteristics can be developed using the same hardware. Manufacturers of 802.15.4-compliant components must therefore implement only the physical and MAC layer, while the implementation of higher levels is demanded to the firmware level.
IEEE 802.15.4 was developed to support distances in the order of 10 meters. It supports relatively high data rates up to 250 kb/s. It was designed to be used in dense networks composed by constrained, simple and battery powered devices, using little or no infrastructure.

In 802.15.4, the nodes of a network can be of two types, full-function device (FFD) or a reduced-function device (RFD). One of the FFD must be the coordinator, which is a logical role. An FFD also has routing functionalities and can participate to network of any topology. RFDs can participate only to start networks, because they can can talk only FFD. Mixing FFDs and RFDs many topology are possible, for example star, P2P, cluster-based trees. The coordinator can be connected to the internet or to other sensor networks. The supported topologies are shown in Figure 2.2.

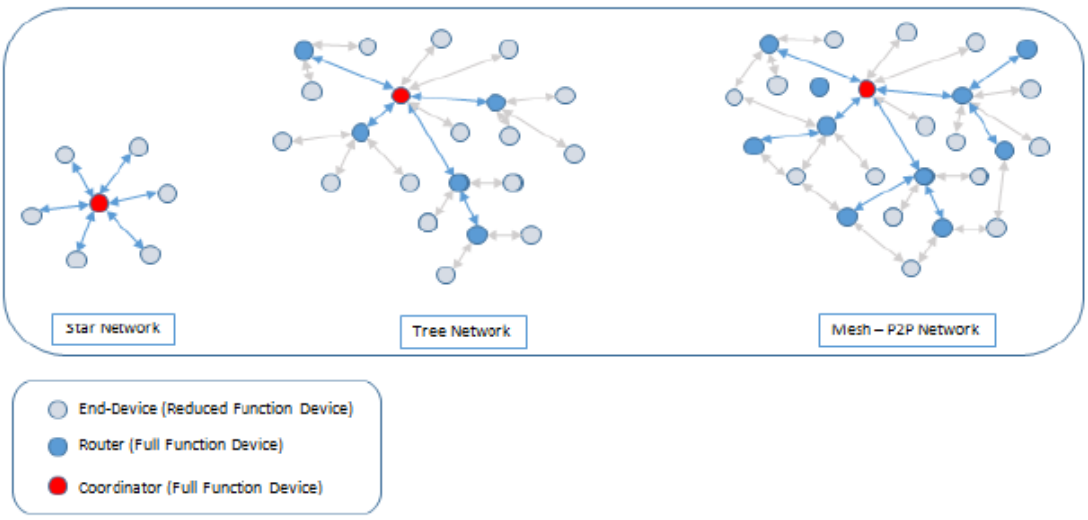

Figure 2.2: IEEE 802.15.4 topologies. Image from [22].

Regarding the MAC Layer, a coordinator can determine whether to work in beacon-enable mode or not. In beacon-enabled mode, the coordinator broadcast beacon messages in order to synchronize the nodes. These beacons

divide the time in superframes. A superframe is composed of 15 equally spaced slot. The time is slotted, beacons start in correspondence of a backoff period start. In the superframe we distinguish the Contention Access Period (CAP), the Contention Free Period (CFP) and an optional idle period in order to improve energy efficiency of the network. The superframe structure is shown in Figure 2.3.

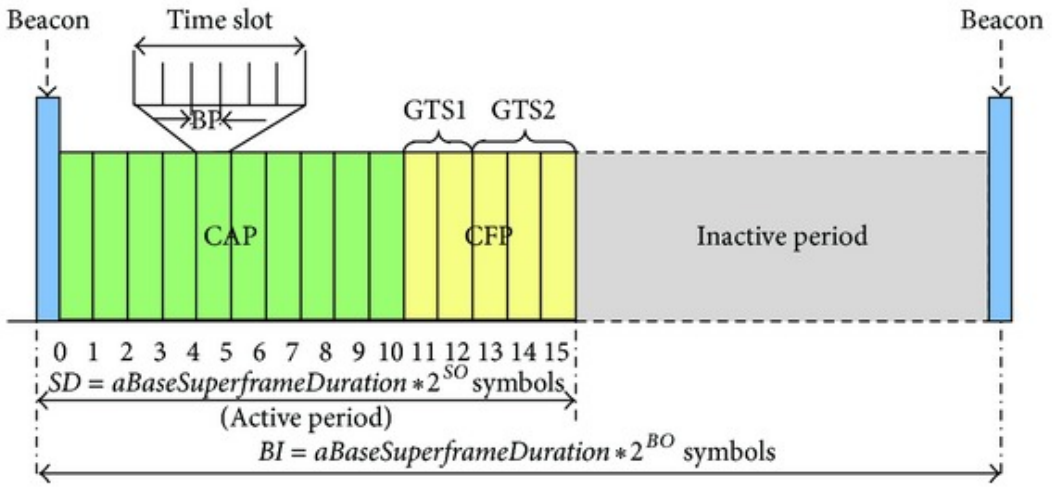

Figure 2.3: IEEE 802.15.4 superframe structure. Image from [23]

The Poll-Based Transfer is implemented in order to avoid nodes that are sleeping to lost data. The coordinator, store data for nodes that are not awake, and alerts them of this data through beacon.

In beacon-enabled mode, the Slotted CSMA/CA access scheme is used. Otherwise, the Unslotted CSMA/CA access scheme is used. The Slotted CSMA/CA employs the Binary Exponential Backoff (BEB) algorithm to reduce the probability of collisions.

During the CAP, the nodes compete for the time slots using CSMA/CA and the channel access is probabilistic. During CFP, in contrast, the channel access is deterministic and therefore the protocol supports delay-critical operations assigning Guaranteed Time Slots (GTSs). A GTS is composed of one or many time slot that are assigned by the coordinator to a specific node in a mutually-exclusive way. GTSs assignments are broadcast through beacon messages.

2.3.4 WirelessHART

WirelessHART is the first attempt to create a deterministic protocol that can be used for industrial sensor networks and process control application. It was developed by the HART Foundation and has its basis in the HART protocol, with which it is backward compatible. [24]

It is based on IEEE 802.15.4 and uses TDMA deterministic channel access scheme. At the physical layer it uses direct sequence spread spectrum and frequency hopping in the 2.4 Ghz band. The range of the nodes is about 100 meters and the maximum data rates is 250 Kbps.

WirelessHART networks are made up of nodes with different logical functions, of which the most important is the role of network manager. There is also a security manager, as well as access points, field devices, adapters. From a topological point of view, field devices are terminal nodes, therefore they are physically connected to actuators, sensors and other electronic devices. All nodes, with the exception of field devices, contribute in different ways to the formation and maintenance of network functions, the reliability of communications, routing and security.

At the MAC level, access to the channel is performed using a TDMA-based approach. The time is divided into time slots of fixed duration, sufficient to transmit packets of the maximum size and receive an acknowledgment. During transmission, the frequency is varied between 15 possible channels, allowing a theoretical maximum number of 15 parallel transmissions during the same slot.

The topology and routing are mostly maintained by the network manager, who is also responsible for assigning the transmission slots. For this reasons the network manager is typically a computationally powerful node and is not battery powered. Its computational capabilities can place limits on the maximum network size. It is potentially a single point of failure.

Graph based routing is used. The network manager builds many directed graphs that are not unique and may overlap by design in order to guarantee redundancy.

To increase reliability, also Automatic Repeat Requests (ARQs) are used. The security manager guarantees data integrity at the MAC level, implementing many functions useful for data encapsulation.

The protocol is not natively IP-compatible, therefore suitable gateways must be used to communicate with other IP-compatible networks. An examplary WirelessHART network is shown in Figure 2.4.

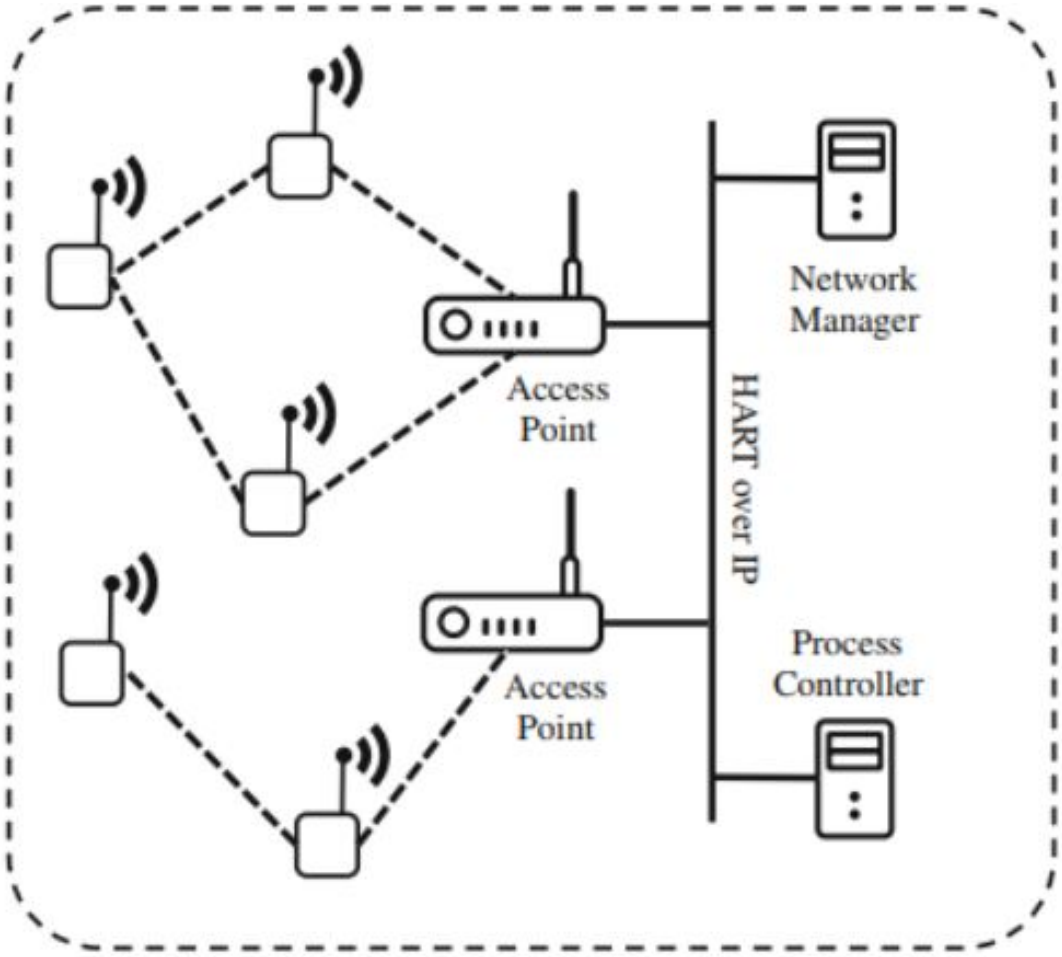

Figure 2.4: WirelessHART architecture. From [25]

2.3.5 ISA100.11a

The standard ISA 100.11a is also based on the 802.15.4 protocol and was developed for the same use cases as the WirelessHART protocol [26].

Channel access can be CSMA/CA or through deterministically assigned time slots if the communication require very low latency. During the CAP, CSMA/CA is used which, however, considers different priority classes.

During CFP, unlike WirelessHART, the size of time slots and superframes is not fixed and can also vary based on the traffic load of the single node.

The Network System Manager (NSM) provides routing functionalities building redundant routing graphs. Different graphs are used for different

traffic types and priorities, in order to improve quality of service [27]. The NSM collects statistics and in formations about the nodes and considers constraints configured by the designer of the network, like latency, throughput and data rate bounds. The generated graphs are tuned dynamically at runtime by the network system manager to consider variations on the operation conditions like battery levels and links quality.

An examplary ISA100.11a network is shown in Figure 2.5.

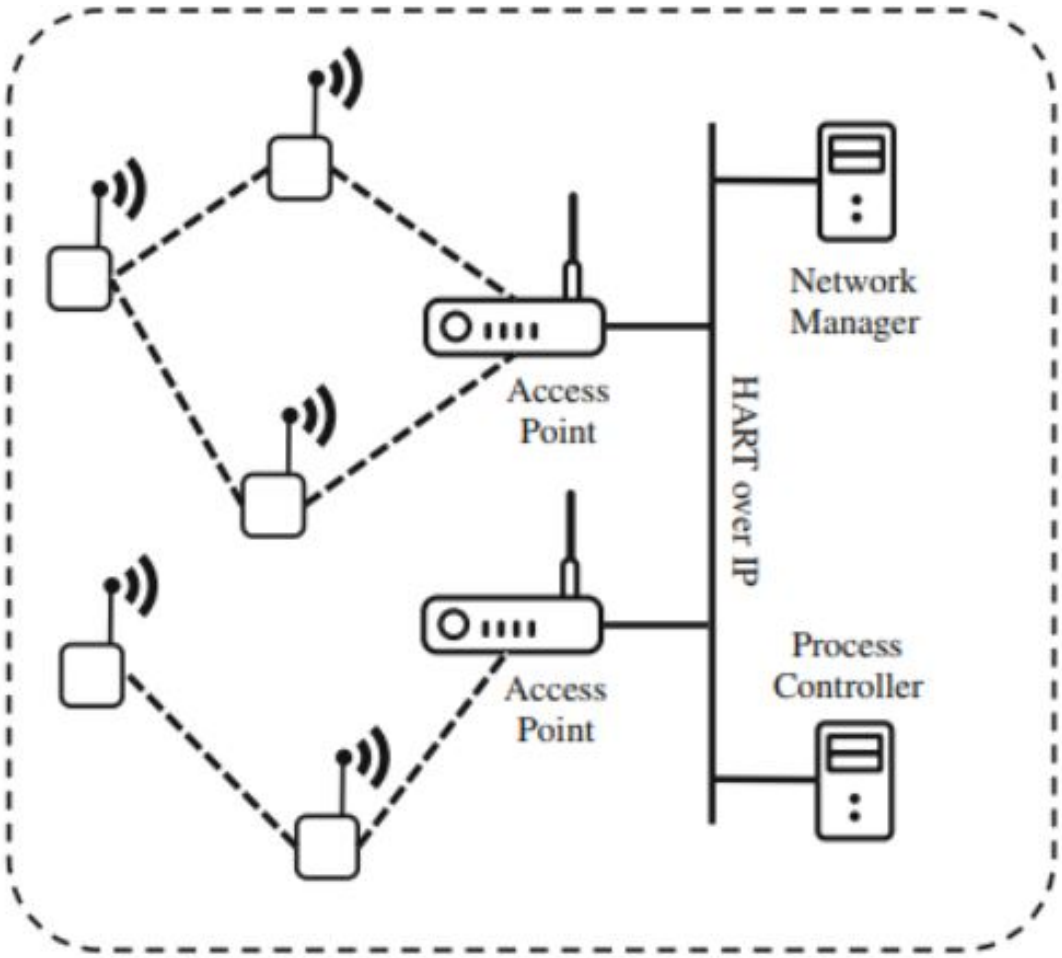

Figure 2.5: ISA100.11a architecture. From [25]

2.3.6 Bluetooth and Bluetooth Low Energy

The initial goal of the Bluetooth protocol was to develop a wireless protocol that would allow the commercialization of wireless computer peripherals. It was therefore developed for short-range networks consisting of a reduced number of battery-powered nodes. In this market segment the protocol has been very successful, considering the fact that it has become a standard included also in computers and smartphones. Therefore, a very large number of input/output devices and peripherals compatible with this standard has been built.

There are many variations and strains, including very low-energy versions like Bluetooth Low Energy (BLE) [28]. This version has a very simple state machine, which is shown in Figure 2.6.

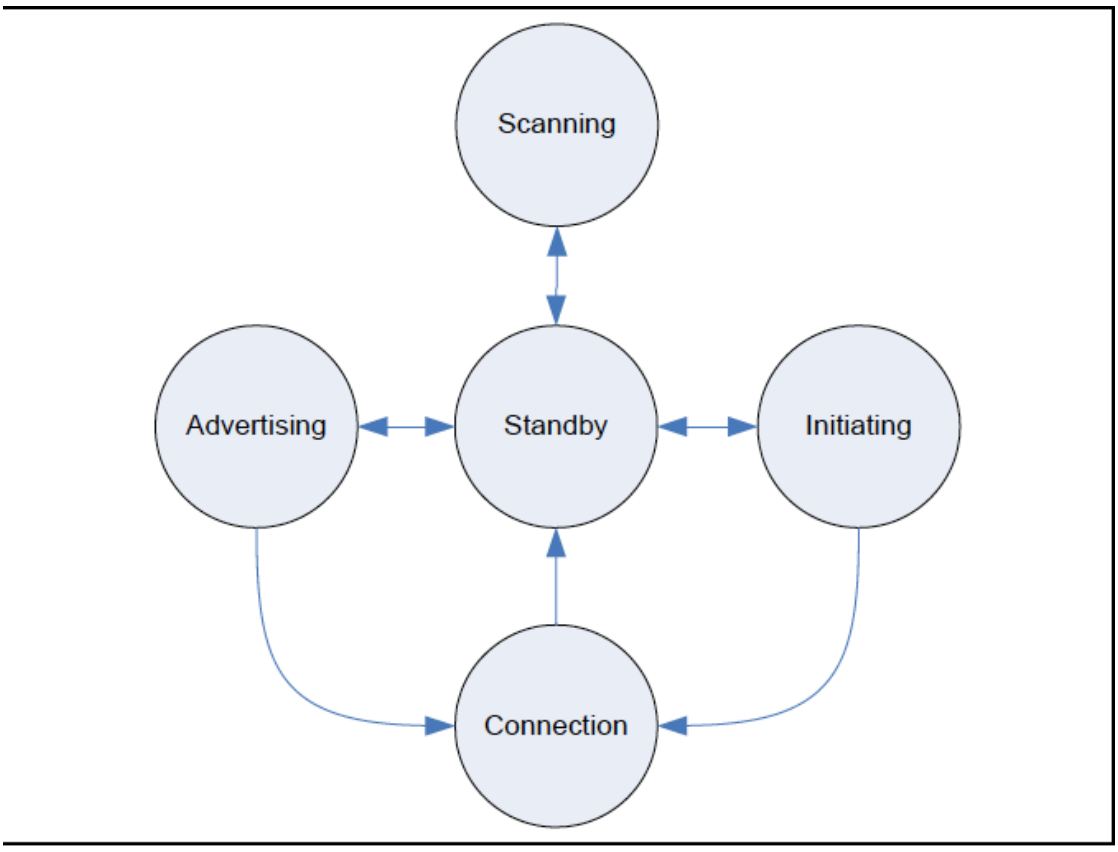

Figure 2.6: BLE device state machine. From [29].

These protocols are not based on the 802.15.4 protocol, but have been developed with different assumptions and goals in mind. Bluetooth range varies from 0.5 meters to 100 meters and it operates in unlicensed bands in

the 2.4 GHz range. Depending on the version and on the configuration, data rates are between 0.5 and 3 Mbps. It supports mesh, tree, star and mixed topology [30].

Despite their diffusion, these protocols are not usable for industrial applications as they implement highly probabilistic networks and the reliability is not high enough [31] [32].

Furthermore, these protocols are not directly compatible with IP networks.

2.3.7 Zigbee

ZigBee takes advantage of the 802.15.4 protocol at the physical and MAC layer, adding its own implementation of the higher layers [33]. It has been designed for short range, low data rate and low consumption networks. It has achieved considerable diffusion in domotic and smart home applications. Development of the protocol began in 1998, but the first version was only released in 2004. Since then many improved versions have been released, paying close attention to backwards compatibility.

Physically it uses the same frequencies as WiFi and Bluetooth, with which it can interfere.

In ZigBee networks, nodes can take on different logical roles: ZigBee coordinator, ZigBee ordinary device and ZigBee router. ZigBee routers and the coordinator are full function devices and can perform routing, while end devices can only communicate with coordinators. In this way, even complex and multi-hop topologies are supported, so the network is scalable and can cover even large distances. The coordinate also performs network formation. Logical roles can be dynamically changed to evenly distribute power consumption and network traffic.

When there are interference, collisions and consequent re-transmissions make the protocol probabilistic, therefore it is not suitable for applications oriented to industrial control, where the delay must be bounded and the overall behaviour deterministic.

2.4 Application-to-application protocols

In order to enable global identification of IoT devices, a convenient addressing mechanism was needed. Another important goal was the integration of sensor networks developed with different technologies, for example in a corporate or global network. Therefore, an IP-compatible protocol optimized for low-power devices, called 6LowPAN, was standardized [34]. The protocol is based

on the sixth version of IP, considering the exponentially increasing number of devices connected to the Internet due to the spread of the IoT. IPv6 can support a trillion cubed unique addresses. Essentially 6LowPAN is an adaptation layer between the data link and the network layer. 6LoWPAN brings the IP functionality to wireless sensor networks low-power devices so that they can become nodes in the global Internet. The protocol implements many useful features, such as header compression that allows devices to save memory, as well as packet fragmentation capabilities. The protocol also uses methods to derive the IP of the nodes using the link layer address in order to further compress the header.

Regarding routing, there are several options available. Moreover, mesh routing has been implemented, which is not available in the ordinary IP protocol [35].

One of the immediate advantages of using IP-compatible addressing was to make smart objects available to the entire ecosystem of applications, tools and protocols developed for the Web.

One of these protocols is the Hypertext Transfer Protocol (HTTP) used massively on the Web and well known by software and hardware developers already on the market [36]. However, some technical and operational characteristics of this protocol do not make it suitable for use on constrained devices, so many other application-level protocols has been developed to target constrained, low-power devices.

2.4.1 Communication patterns

2.4.2 CoAP

CoAP is an application layer protocol derived from HTTP [37]. Many features have been changed and new functions have been added. The main difference is that CoAP uses the User Datagram Protocol (UDP) at the lower level, while HTTP uses the Transmission Control Protocol (TCP). UPD is a connection-less best-effort protocol and therefore much lighter than TCP, which requires packets to be acknowledged. UPD therefore prioritizes latency over reliability. However, CoAP has implemented some features that can increase the reliability of the protocol, such as requesting confirmation at the application level. In a CoAP communication, messages of different types can be exchanged:

- Confirmable: these are messages that require the receipt of a confirmation. If the confirmation is not received, the node retries the transmission.

- Non confirmable: these are messages that do not require confirmation, therefore they are used for applications where less reliability is required.

- Acknowledgment: confirmation of receipt.

- Reset: it is sent to the transmitter when a Confirmable message has been received, but it is not possible to reply immediately. It is used to implement an asynchronous communication mechanism.

- Piggy-backed response: it is included as a part of the Acknowledgment. It the the data response.

- Separate response: asynchronous, delayed response.

The communication pattern is request-response, like HTTP. The semantics of the request are also specified by the method. The methods are in fact deliberately inspired in the name and in the semantics to HTTP methods. There are four methods:

- GET: Read-only operation, resource state representation request

- PUT: Insert or update operation (UPSERT)

- POST: Insert or update operation (UPSERT)

- DELETE: Command for delete a resource

CoAP uses the concept of a unique address resource identified by a Uniform Resource Identifier (URI). Therefore the client makes a request to the server specifying the method and the resource, the server responds possibly not a representation of the state of the resource, in accordance with the philosophy of representational state transfer (REST).
To ensure good reactivity, the nodes implement a simple distributed cache, so that if an intermediate node already has the requested data, it responds directly. The intermediate nodes that build the cache are called proxy nodes.

2.4.3 MQTT

The MQTT protocol is oriented towards publisher/subscriber communication. It has been developed specifically for communication between machines (M2M) and to be used in devices with limited computational capabilities and low consumption [38]. This is especially true from the point of view of the publisher, who is almost always a node in the sensor network, while the subscriber can also be a software application on the Cloud that is interested in the published data to store or process them. In fact, the MQTT protocol is one of the most used protocols by Cloud software and infrastructure providers as a IoT data input mechanism.
Nodes can be of two types: client and server, which is unique and it is called broker. The client nodes are the ones that generate (publishers) and consume (subscribers) the data. Communication between clients is never direct, but is always mediated by the broker, who has the task of routing the messages that are published to the nodes that have shown interest in that message. Communication between client and server is therefore two-way, so many implementations use the TCP protocol with the broker that uses ping messages to evaluate the status of connections with clients.
The protocol defines three levels of QoS:

- QoS 0: best-effort, at most once delivery;
- QoS 1: at least once delivery using acknowledgment;
- QoS 2: exactly once delivery.

To support higher QoS, the overhead of the protocol increases, so it's important to make a trade-off based on the nature and usage of the individual data packet. For example, exactly-once deliver is obtained using a complex confirmation mechanism that involves publishers, brokers and subscribers. The actual reliability described by the QoS level may not be achieved if the broker implementation does not allow it in all situations and edge cases, for example if messages are stored in memory, they do not survive a broker stop or restart. More reliable implementations store messages in non-volatile memory, for example using a transactional database.
The nodes interested in a certain data flow subscribe to the corresponding topic. Subscriptions are managed and stored by the broker. The broker keeps the messages in memory until they have been transmitted to all interested parties. Depending on the QoS level, the broker keeps the messages and possibly re-transmits if it does not receive acknowledgment.

2.4.4 DSS

Data Distribution Service (DDS) is an application framework designed for low latency communication between constrained devices. The initial aim was to connect threads of different applications in a location-aware way, applying the concept of location transparency [39]. The communication pattern is topic-based publisher/subscriber, but unlike MQTT, the system is also completely distributed from a logical point of view, being brokerless. There are therefore no single points of failure. Publisher and subscriber are discovered at run-time and each device keeps its own register. The main difference is therefore architectural. This device can reduce communication latency times, as there is no broker, which is often installed in a Cloud due to relatively high computational requirements.
During the definition of the topic, the typed structure of the exchanged data is also specified. The type is specified through a typed data schema. On a practical level, developers publish typed messages, which can be instance of classes or structs based on the language used in the single application node. The framework takes care of marshaling automatically and in a language-agnostic way.
The framework implements a complex and exhaustive QoS system that touches all aspects of communication, such as data expiration, delivery methodology, persistence level over time and many other parameters. Publishers offer one level of QoS and subscribers require another, therefore communication occurs only if the two contracts are compatible, meaning that the QoS requested by the subscriber must not be more stringent than the one offered by the publisher.

Chapter 3

Proposed framework

The proposed software framework was developed for the ingestion, storage and visualization of data in applications oriented to predictive maintenance. The overall architecture of the framework is shown in Figure 3.1. The framework consists of a sensor and actuator network simulator developed with NodeRED [40] and a Python web application exposed via a Flask web server. The simulator sends the sensor readings to the web application via MQTT protocol using Mosquitto broker. The web application uses a PostgreSQL database as a persistence layer, which it accesses through the abstraction layer provided by SQLAlchemy. The components were containerized and deployed using Docker. Many of the components were chosen because they are open-source, general-purpose and widely used and documented. The individual components are discussed below.

3.1 Tools

3.1.1 Backend

Python

Python is a high-level programming language. It supports many programming paradigms and styles, including object-oriented, functional, procedural and aspect-oriented programming. Python was developed to have a general-purpose and non-specialized core, but highly extensible through the use of modules.

Python was defined to allow the development of applications that are modular, readable and free of large amounts of boilerplate code. Readability

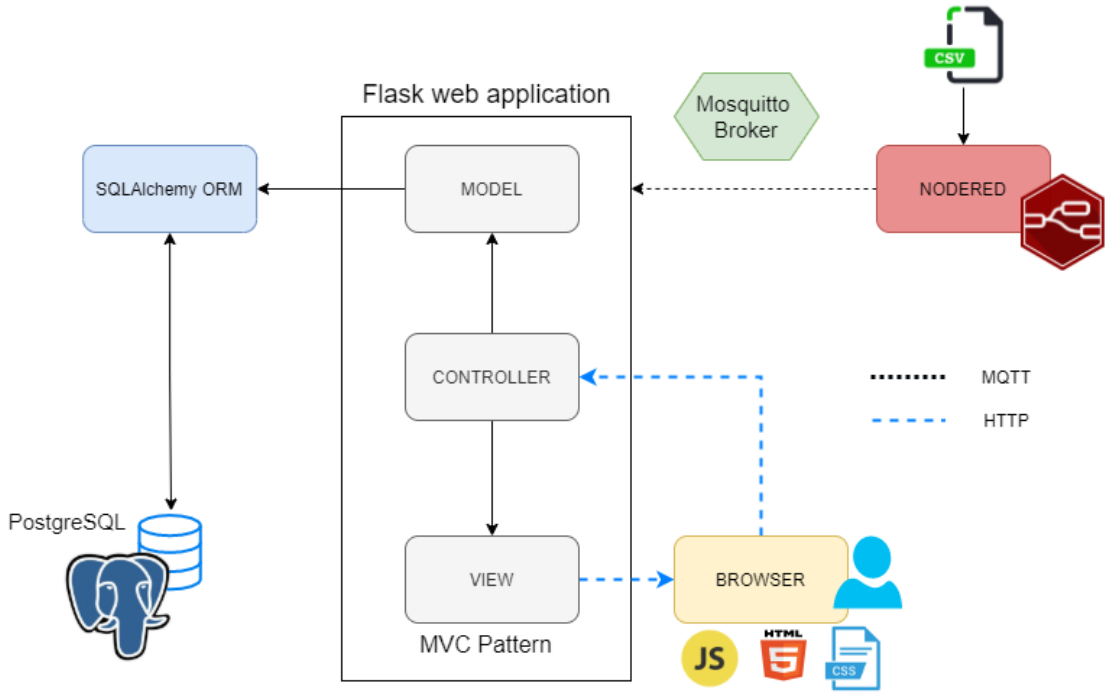

Figure 3.1: Software framework architecture

is one of the fundamental objectives, for which an indentation convention has also been implemented which must be followed by the programmer.

The Zen of Python [41] summarizes the philosophy of Python in these points:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Flat is better than nested.
- Sparse is better than dense.
- Readability counts.
- Special cases aren't special enough to break the rules.
- Although practicality beats purity.

- Errors should never pass silently.
- Unless explicitly silenced.
- In the face of ambiguity, refuse the temptation to guess.
- There should be one– and preferably only one –obvious way to do it.
- Although that way may not be obvious at first unless you're Dutch.
- Now is better than never.
- Although never is often better than *right* now.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.
- Namespaces are one honking great idea – let's do more of those!

With this philosophy and features, Python immediately became particularly attractive to scientists, who usally are not programmers. Python is indeed simple to learn. In 2018, the 2018 Kaggle Machine Learning Data Science Survey [42] resulted in nearly all data scientists using Python for their applications. This has led the community to develop a large number of libraries and modules for scientific and data analysis oriented applications. These modules allow you to collect, represent, clean, process and display even large amounts of data. Many mathematical models and operators have been implemented ready to be applied in the data processing pipeline. In addition to processing, there are many modules for viewing the results, for example to build plots and graphs of various types ready to be attached to publications or scientific documents.

Although Python's first goal is certainly not to be a very fast language, Python applications interface easily with lower-level languages such as C or Fortran. Therefore many of these modules used for scientific applications have been developed in other languages, for example Numpy is written almost completely in C. This technique makes the modules very efficient and fast.

Python is considered a general purpose programming language, it can in fact also be used for the development of web and desktop applications, for example.

Flask

Flask is a Python micro web framework. It was first developed and released on April Fool's Day in 2010 [43], as a joke. Armin Ronacher, its developer, was in fact working on a set of Python libraries, called The Pallets Projects. The set of modules was rewritten and published in a single file, riding the general interest that there was for minimalist and simple frameworks, as opposed to Django [44] , which is considered a complete and battery-powered framework. In fact, many of the functions typically needed to develop complex web applications are already included with Django, such as authentication, authorization, database management and an object-relational mapper (ORM). On the contrary, Flask is minimalist, therefore the developer adapts the design to his own needs by importing Python modules that implement the researched features. Being minimalist, it is generally a good choice for prototypes or applications that are not too complex, also being quick to learn.

The basic features implemented by Flask are:

- URL routing
- Logging
- Templating via Jinja2 [45] template engine
- XSS and other vulnerabilities protection
- Request payload serialization

The concept of minimalist framework is well explained in the official article where some architectural and functional choices of the framework are discussed [46] :

> Flask will never have a database layer. It will not have a form library or anything else in that direction. Flask itself just bridges to Werkzeug to implement a proper WSGI application and to Jinja2 to handle templating. It also binds to a few common standard library packages such as logging. Everything else is up for extensions.
>
> Why is this the case? Because people have different preferences and requirements and Flask could not meet those if it would force any of this into the core. The majority of web applications will need a template engine in some sort. However not every application needs a SQL database.
>
> The idea of Flask is to build a good foundation for all applications. Everything else is up to you or extensions.

PostgreSQL

PostgreSQL is a free and open source relational database management system (RDMS). The database is ACID-compliant. It is characterized by a high level of reliability and robustness, being the product of more than 30 years of development carried out by various software houses and the community.

It was initially developed as a successor to the Ingres database [47] , which inspired the original name, POSTGRES. In 1996 the project was officially renamed to PostgreSQL, to highlight compatibility with SQL standards.

The principal features are:

- transaction support with Atomicity, Consistency, Isolation, Durability (ACID)
- foreign keys
- SQL-compatible
- materialized views
- triggers
- stored procedures

It supports all operating systems. PostgreSQL is defined as object-oriented database, as it supports some functions typical of object-oriented languages, such as table inheritance and function overloads.

Unlike other database systems which use locks for concurrency control, PostgreSQL manages concurrency using multiversion concurrency control (MVCC). Each transaction is isolated because works with a versioned snapshot of data which can be different of the real current state of the data model. In this way, PostgreSQL does not suffer from dirty reads [48].

This has led the community to develop many extensions and plugins that allow you to support specific use cases or to improve performance for some specific types of data or queries. Some examples are:

- PostGIS provides data types, functions and operators to work geo-spatial 2D and 3D data.
- Ltree provides data types, functions and operators to work with hierchical tree-like data.
- Pgcrypto provide cryptography and encryption functions.

- many Foreign Data Wrappers (FDW) allows to connect and cooperate with other data storage systems

Furthermore, being open source, modified versions optimized for specific applications and data models have also been developed, for example:

- TimescaleDB is an open-source customization tuned to work with time-series data.
- MartenDB is .NET Transactional Document DB and Event Store based on PostgreSQL.

PostgreSQL supports unrelated, unstructured or semi-structured data models via the JSONB [49] data type, making it suitable for many applications where NoSQL databases are typically used.

SQL Alchemy

SQLAlchemy [50] is a Python object-relational mapper (ORM). It allows to separate the model at the application level from its representation in the persistence level, following the separation of concerns software design principle. The approach adopted by SQLAlchemy is complimentary-oriented, as the level of abstraction provided is not as solid as that of other ORMs. The way in which the entities are stored and the queries built can in fact be customized by the developer in a very granular way, in order to support even scenarios where performance is important and it is necessary to generate sub-optimal queries [50].

3.1.2 Frontend

The frontend of the application consists of many Hypertext Markup Language (HTML) pages. Some features and interactivity have been implemented using JavaScript. The Bootstrap web application framework was used.

Bootstrap is an open source and free framework for developing web pages. It is composed of many Cascading Style Sheets (CSS) files and some JavaScript scripting used to make some components of the framework interactive. Bootstrap was initially developed to be used internally by the Twitter development team. It was developed to avoid problems of inconsistency and etoregeity

among the developers of the team, who before Bootstrap used many external tools and libraries, making the frontend code chaotic and cooperation difficult.

Bootstrap is relatively simple for web developers to learn, because they are already familiar with HTML, CSS and JavaScript. The philosophy is to develop mobile-first and responsive components. The layout and size of the elements therefore adapt to the device in which they are displayed according to the size of the screen. The primary use of Bootstrap is to give a uniform appearance to all pages of the application by applying the global HTML element styles and colors defined by Bootstrap, which can however be customized by the developer. In addition to the basic styles, Bootstrap contains decides of web components such as dropdowns, buttons, navbars, modals and popups, breadcrumbs, ect. It also provides many CSS utility classes. Bootstrap allows you to stylize pages in such a way that their appearance does not vary depending on the browser used by the user.

Bootstrap provides the basic elements to define the page layout. The basic layout component is the Container. There are fixed-width and a fluid-width containers. The latter always fills the width of the parent element, the former are fixed-width, in the sense that they have a max-width which is constant values that depends on the size of the screen.

Containers are created by applying the "container" and "container-fluid" CSS classes to HTML elements of type div. Containers can be nested together and are then used to define the basic layout of the page. By appropriately positioning and sizing the containers, it is possible to obtain all the most common layout types such as one, two or three column layouts, optionally with the presence of footer and/or header sections.

To create components that are responsive and mobile-first, Bootstrap uses the concept of breakpoints. Five breakpoints are available, each referring to a different screen size in pixels. When a class with a breakpoint is applied to an element, it is only active for screens at least as large as specified by the breakpoint itself. The Table 3.1 shows all the available breakpoints.

Breakpoint	Starting point	Scenario
xs	0px	portrait mobile
sm	576px	landscape mobile
md	768px	portrait tablets
lg	992px	landscape tablets
xl	1200px	desktops, laptops, TVs

Table 3.1: Breakpoints in Bootstrap

The default breakpoint is xs. Larger breakpoints override smaller breakpoints and viceversa.

Another important element for developing pages with Bootstrap is the grid system. The system is mobile-first and fully responsive. It is composed of some fundamental elements, which are:

- containers
- rows
- columns

The rows must be children of the containers and the columns must be children of the rows. In Bootstrap, the width of a row is logically separated by 12 equally sized slots. By default the columns divide the 12 equally large slots, but it is possible to apply different sizes to the individual columns. For example, this code creates two columns one 4/12 wide and one 8/12 of the space made available by the container:

```
<div class="container">
    <div class="row">
     <div class="col-4">
       ...
     </div>
     <div class="col-8">
        ...
     </div>
    </div>
</div>
```

In the context of columns, the breakpoint can be defined to define the starting point from which the columns are shown as a horizontal sequence

of blocks. Below the breakpoint, the columns appear as a vertical stack of blocks. The default breakpoint is xs, so if it is not specified the columns are always shown in horizontal sequences. By specifying the breakpoint instead it is possible to have different views on different screens. The classes can also be combined with each other.

3.1.3 NodeRED

Node-RED [51] is an open source and free visual workflow editor. It allows you to create low-code complex workflows that connect different software and devices, also taking care of integration at the application level. It was initially developed specifically for the Internet of Things by IBM Emerging Technology, in fact it is included in IBM Bluemix, an IoT-oriented Platform-as-a-service package proposed by IBM.

Flows and workflows are created by linking together nodes of different types, which abstract common functionalities. It is possible to integrate custom JavaScript functions to satisfy the most complex use cases. Nodes process messages coming on their input and eventually send messages on their output nodes, continuing the flow. The system is developed in NodeJS [52] environment, so it can be installed in many devices, for example single board computers (SBCs), cloud, FPGA etc. The system takes advantage of the event-driven, asynchronous non-blocking nature of NodeJS [53] . NodeJS JavaScript environment is single-threaded, so Node-RED applications may have performance issues in CPU-intensive operations [53] , but if well designed Node-RED applications can be scaled horizontally.

Considering its drag and drop nature, it is particularly suitable in the prototyping phase and for developing proof-of-concepts, as it can also be used by people who have no experience in programming. In addition, the system supports collaborative development, because the flows can be exported to JSON and can optionally be versioned by a versioning collaborative system, such as GIT.

Node-RED can be extended by developing new types of nodes and interconnections. This has resulted in the open source community developing a large amount of nodes, making the ecosystem particularly rich.

3.1.4 Docker

Docker [54] provide operating system level virtualization for software modules. Through Docker it is possible to create self-contained packages, which

contain all the dependencies they need to be able to work in the environment that hosts them. When using virtualization through virtual machines, the use of hardware is less efficient, since in each virtual machine it is necessary to allocate the resources necessary for the operating system. Docker takes a different approach, creating a virtualization environment on top of the operating system. Nonetheless, the containers are isolated. Containers share the host operating system's kernel and hardware resources. Docker is an open source technology.

The experience provided by Docker is designed to be developer-centric and therefore to be integrated into the usual development workflow. Unlike classic virtualization, it is geared towards running applications rather than emulating hardware.

Docker is not the first technology to create containers, but compared to its predecessors it has spread particularly for the following practical reasons.

- Speed: creating, deploying and running containers is extremely fast via Docker, and can be done on any PC.
- Simplicity: Docker is simple to use and learn, especially for developers, to which the particular way is aimed.
- Portability: Docker containers are extremely portable, they can be followed in any environment where the Docker execution environment is installed. For example, applications can be moved from the development environment to the production environment very easily.
- Scalability: Because they are light and decoupled, the containers can be easily scaled. Therefore, many technologies have been developed to orchestrate containers, such as Kubernetes [55] .

The Docker environment is made up of several modules that interact with each other. the Docker environment can be seen as a client-server application [56].

The Docker Client interact with the Docker daemon, which is the server, sending commands and receiving responses. Examples of commands are those to launch, stop or build containers. Docker is shipped with a standard command line interface (CLI) executable Docker client. Client and server can be in different machines.

The Docker daemon listens for incoming requests sent by the Docker client application. It is installed on the host machine, where the virtualization is needed.

In essence, containers are runnable instances of Docker images. Docker images are templates to build containers. They are described by the corresponding Docker file, where every dependency of the application is listed. In the Docker file many other things can be configured, for example regarding network functions or any volumes to allocate. In practice, the instructions for building the image are encoded in the Docker file. Docker files are text files, meant to be written by the developer, as an integral part of the software development pipeline.

Modern applications often consist of many collaborating services. For example, applications often need a database, which can be provided via a Docker container. For this reason, Docker Compose was developed, a tool for configuring and running applications made up of multiple containers [57]. The application and required dependencies are described via a YAML file. The configuration file eventually specifies the correct boot order and other necessary configurations to allow individual containers to cooperate.

3.2 Data model

The data model was developed focusing on predictive maintenance applications. Since the proposed data model is relatively simple, it is described below in terms of database tables. These correspond to classes in the application. As the model is simple, the classes mirror their tables. One-many relationships are mapped through Python lists. In general this is not true, since for complex applications there is a tendency to separate the software model from the one that describes its physical persistence, in the perspective that the persistence model and the Domain model [58] can be changed and developed independently. The complete model of the tables is shown in figure 3.6.The main entities are described in the following paragraphs:

3.2.1 Machine

The **Machine** entity is used to represent the machines to be monitored in the manufacturing process. In general, in manufacturing processes some machines may need to be configured for the work session they are about to start. These configurations have been abstracted with the concept of **Working Parameter**. The operator then specifies for each machine which are the necessary settings and what type they are (float, string).

3.2.2 Tool

The **Tool** entity is used to represent the tools that are mounted in the machines to perform the working activity. The tools are subject to wear. Tool wear is described by the **Wear Parameter** abstraction. The operator defines the various parameters that can numerically or textually describe the level of wear of the tool.

3.2.3 Sensor

The sensors mounted in the machines to monitor the parameters of interest are abstracted from the **Sensor** entity, to which the relative **Sensor Readings** are linked. In the context of predictive maintenance, these sensors often measure continuous numerical quantities related to ongoing mechanical or thermal stresses and variations.

3.2.4 Working

It represents a process performed by a **Machine**. The processing is characterized by an instant of beginning and end. The **Tool** that was mounted on the **Machine** and the values of the **Working Parameters** needed by the machine are specified. These values are stored in the working_configuration table. At the end of the machining it is possible to specify which was the value of the tool's **Wear Parameters**. The wear values are stored in the wear_configuration table.

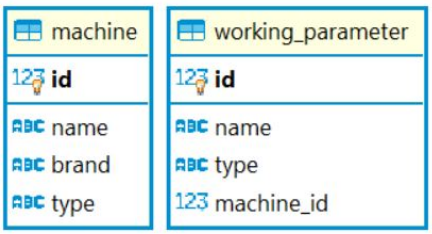

Figure 3.2: Machine model

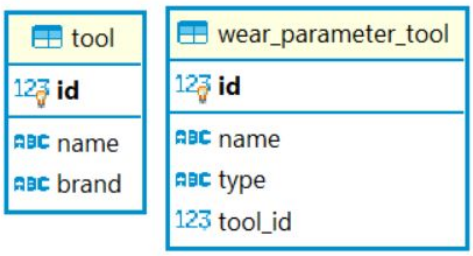

Figure 3.3: Tool model

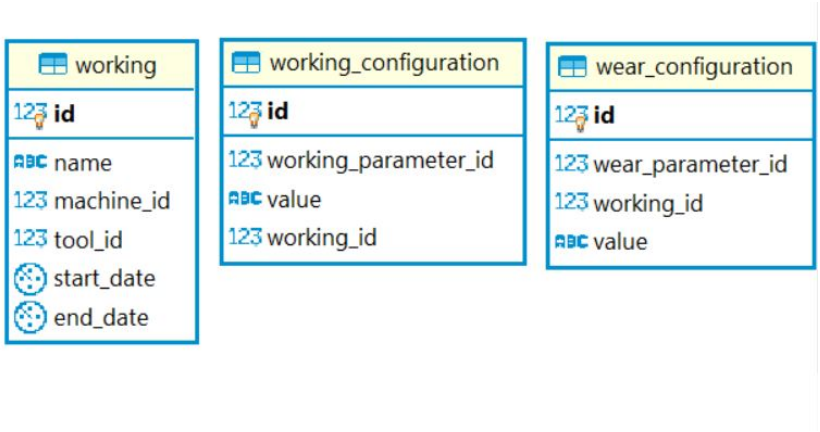

Figure 3.4: Working model

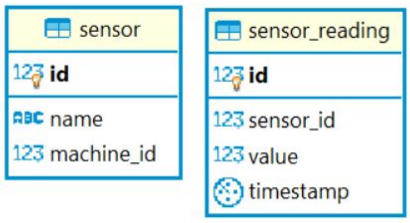

Figure 3.5: Sensor model

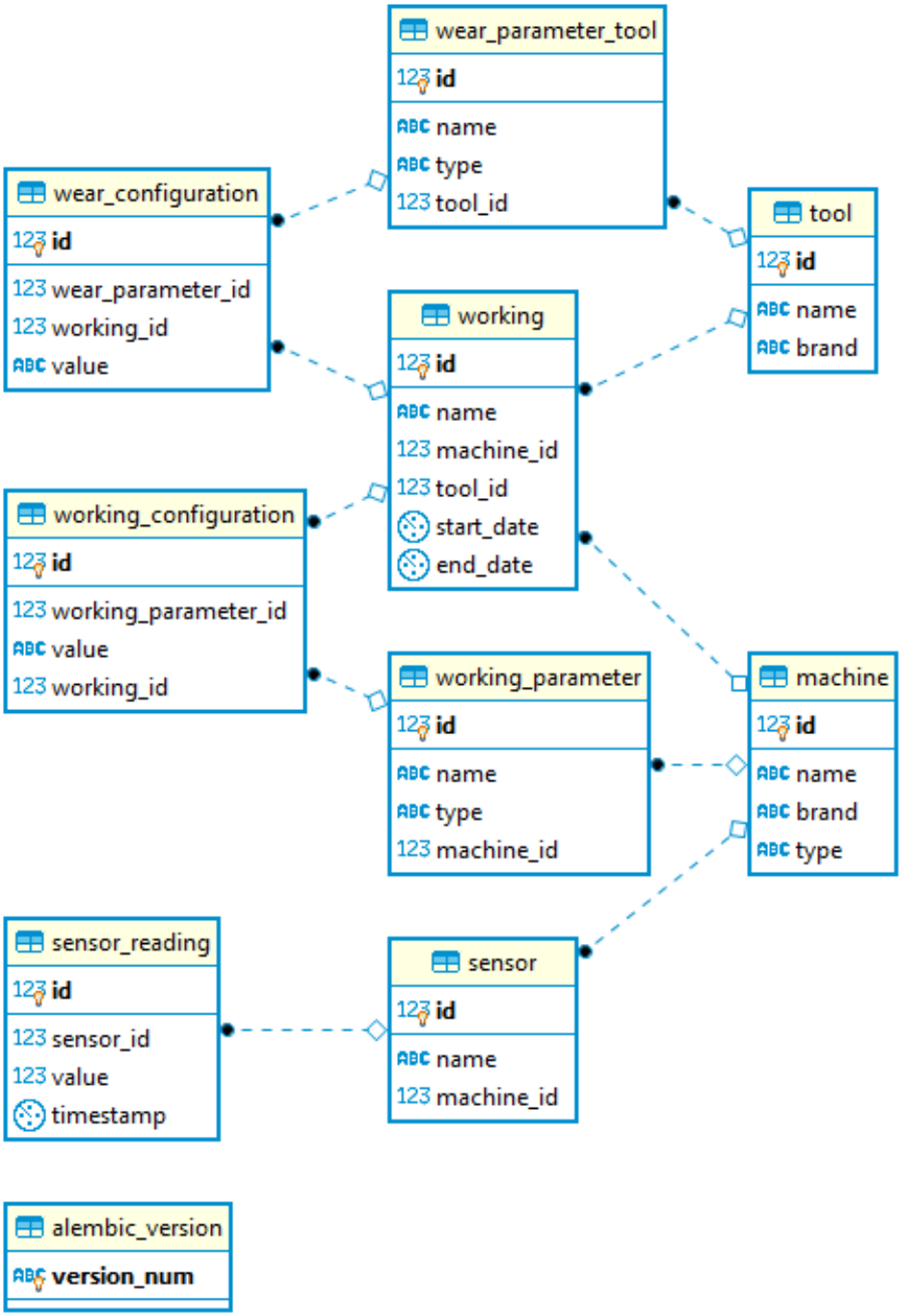

Figure 3.6: Data model

3.3 Machine acquisition simulator

In order to validate the MQTT data acquisition system of the web application, a data acquisition simulator was developed using Node-RED. The Node-RED application sends messages to the MQTT topic to which the web application listens.

The structure of the topic is the following:

device/+/sensor/+

For example, to send data to the vibration sensor positioned in the Milling machine, it is necessary to publish to the following topic:

device/Milling machine/sensor/Vibration

The subscriber application recognizes the corresponding sensor and populates the sensor_readings table, where the data acquired by the sensors during processing are stored. The acquired data are then shown by the web application on the processing detail page.

Several flows were used to publish fake sensor readings. The proposed flows can be combined to generate more complex flows, possibly publishing the births of several sensors at a time. By also introducing actuator nodes, many scenarios can be simulated, such as control loops.

3.3.1 Single-sensor random value publisher

The flow shown in Figure 3.7 has the effect of publishing a random reading of a sensor on the corresponding topic. From left to right, it is composed as shown below.

- Inject node: it is used to trigger the flow by clicking the node's button. The node can be configured to trigger the flow only once or at regular time intervals. It can possibly generate a message that will be passed into the input branch of the next node, but in this case it has not been used.
- Function node: The function node is used to define a JavaScript custom function that manipulates the message the node receives in the input branch. In this case the function node was used to define the message to be published.

```
max = 100
min  = 5
msg = {
    payload: {
        timestamp: new Date().getTime() / 1000,
        sensor_value:
        Math.floor(Math.random() * (max - min + 1) + min)
    }
}
return msg;
```

- MQTT out node: This node receives the message generated by the function node and publishes it in the corresponding topic. The configuration of the node is shown in Figure 3.8. Through the configuration panel it is possible to configure the connection to the broker to be used. In this case, an instance of Mosquitto was used.

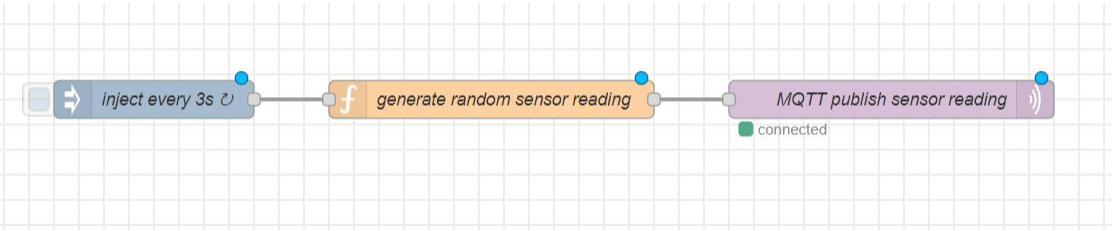

Figure 3.7: Single-sensor random value publisher

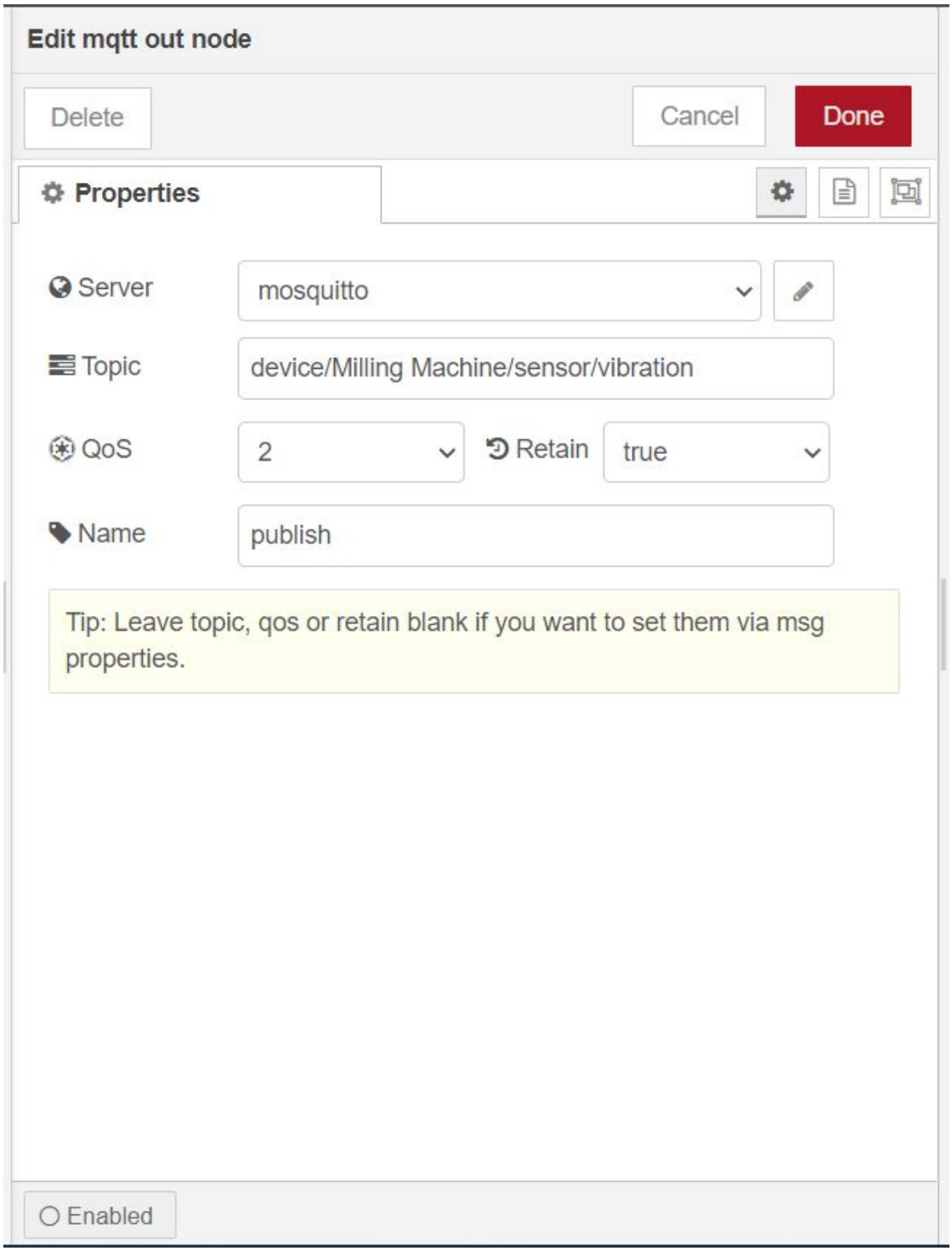

Figure 3.8: MQTT out node configuration

3.3.2 Single-sensor value publisher using a file as source

The flow shown in Figure 3.9 has the effect of publishing a sensor reading obtaining the corresponding value reading the content of a comma-separated-value (CSV) file. From left to right, it is composed as follows:

- Inject node: same as before. In this case, the single trigger mode was used.
- File node: The file node is used to read the file. It has been configured to read the file one line at a time, assuming that in each line there is a sensor value corresponding to a different instant of time.
- CSV node: This node receives the message CSV and converts it to a JSON message. The columns of the CSV are mapped in JSON properties with the same name.
- MQTT out node: as before.

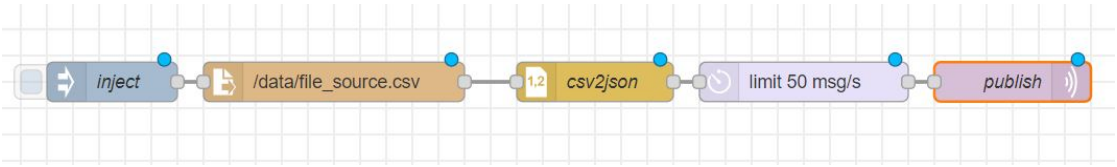

Figure 3.9: Single-sensor value publisher using a file as source

3.4 User Experience and User Interface

In order to support the different use cases, the application consists of many pages. The application was developed to be simple to use for an operator who has no experience with other information systems.

3.4.1 Machine registry

The machine registry allows the operator to computerize the machine park used in the manufacturing process. It is possible to add other machines to the register or view the details of the machines present by clicking on the corresponding row. In Figure 3.10 the machine registry page is shown. When a new machine is registered, it is possible to specify the sensors that

are mounted on the machine and the working parameters. In Figure 3.11 the page to register a new machine is shown. By clicking on a row on the machine list page, the details of the selected machine and the details of any machining in progress are displayed through the machine details page, shown in Figure 3.12.

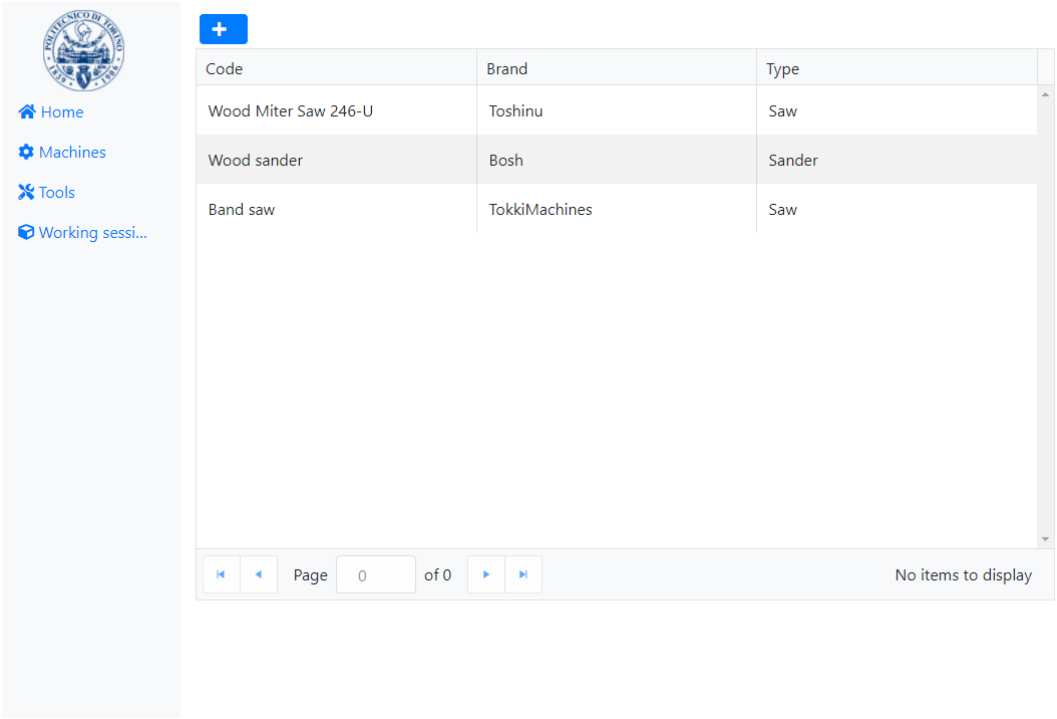

Figure 3.10: Machine registry page

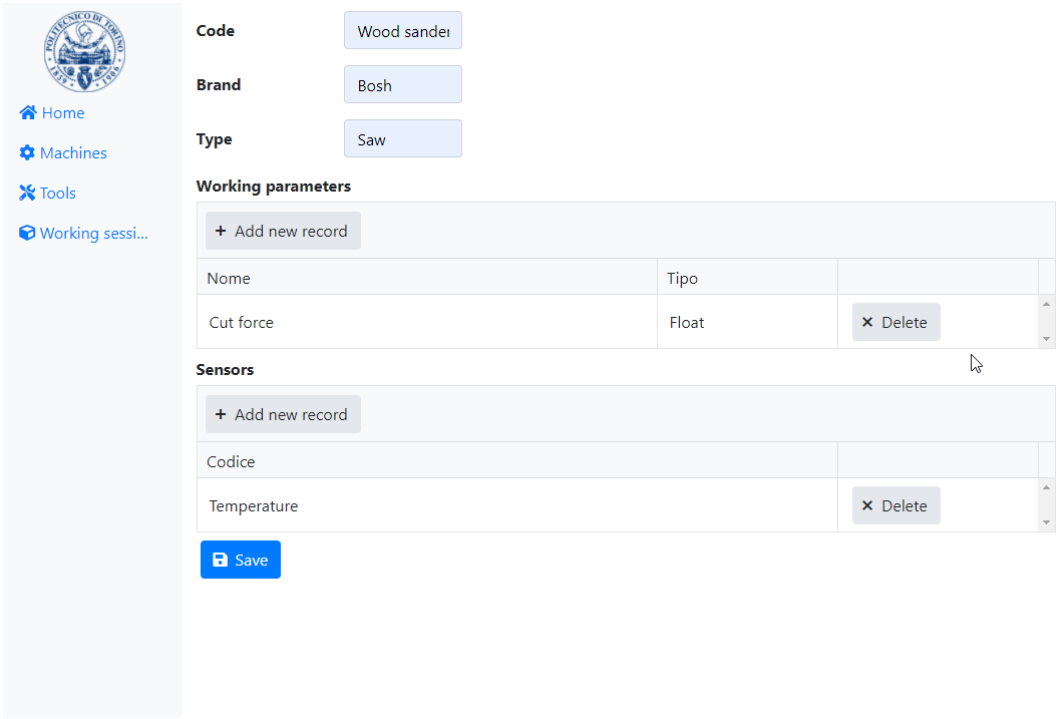

Figure 3.11: Machine registration page

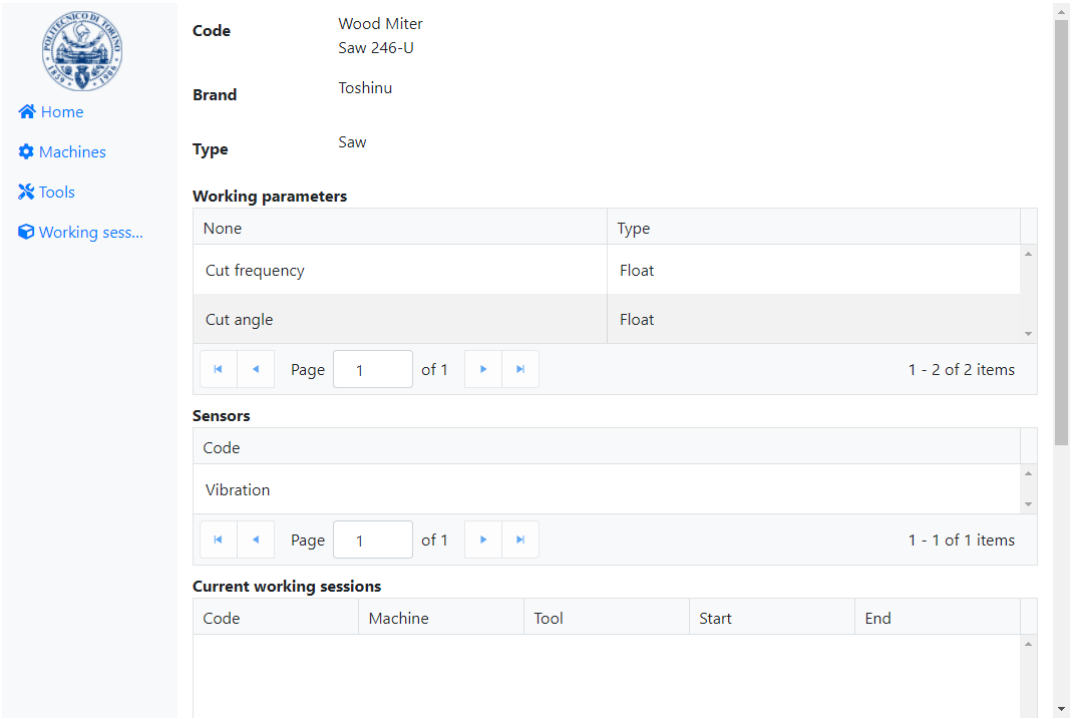

Figure 3.12: Machine details page

3.4.2 Tool registry

Using the tool registry, it is possible to register the tools that will be mounted in the machines during the machining processes. On the tool registry page, shown in Figure 3.13, the tools are shown in a grid. It is possible to add new tools or view the details of the tools already registered. In Figure 3.14 the page for register a new tool is shown. When adding a tool, it is possible to specify the parameters that describe its wear over time. The details of a tool are shown in the tool detail page, shown in Figure 3.15. This page also shows the evolution of the tool's wear, described by its wear parameters. The actual value of the wear parameters are specified by the operator at the end of each work session.

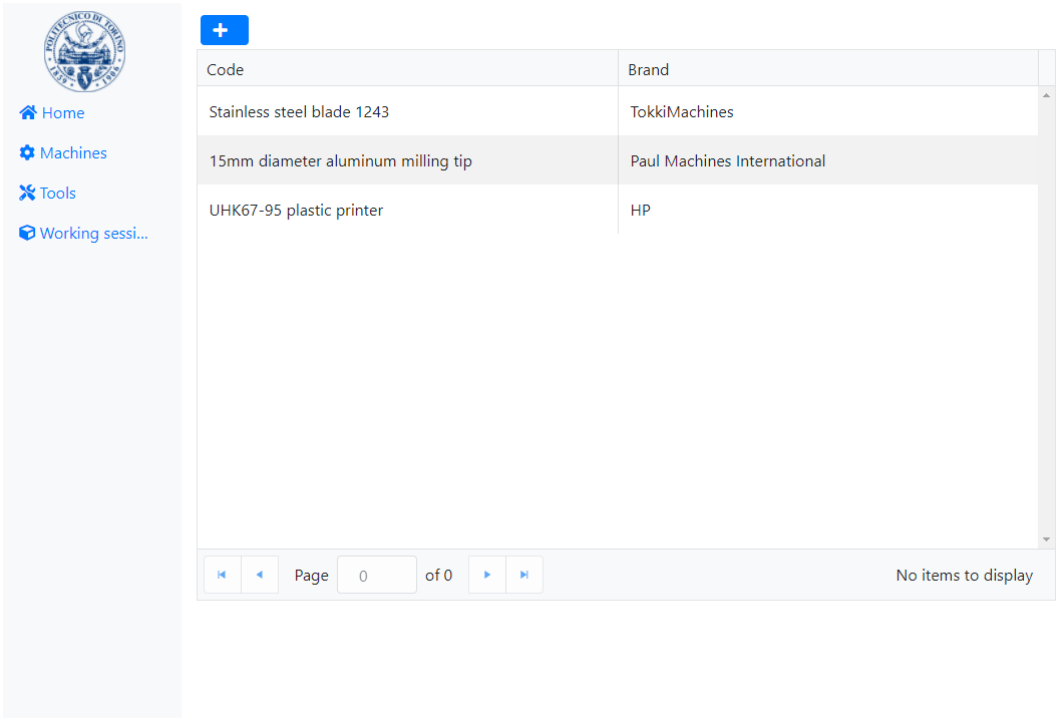

Figure 3.13: Tool registry page

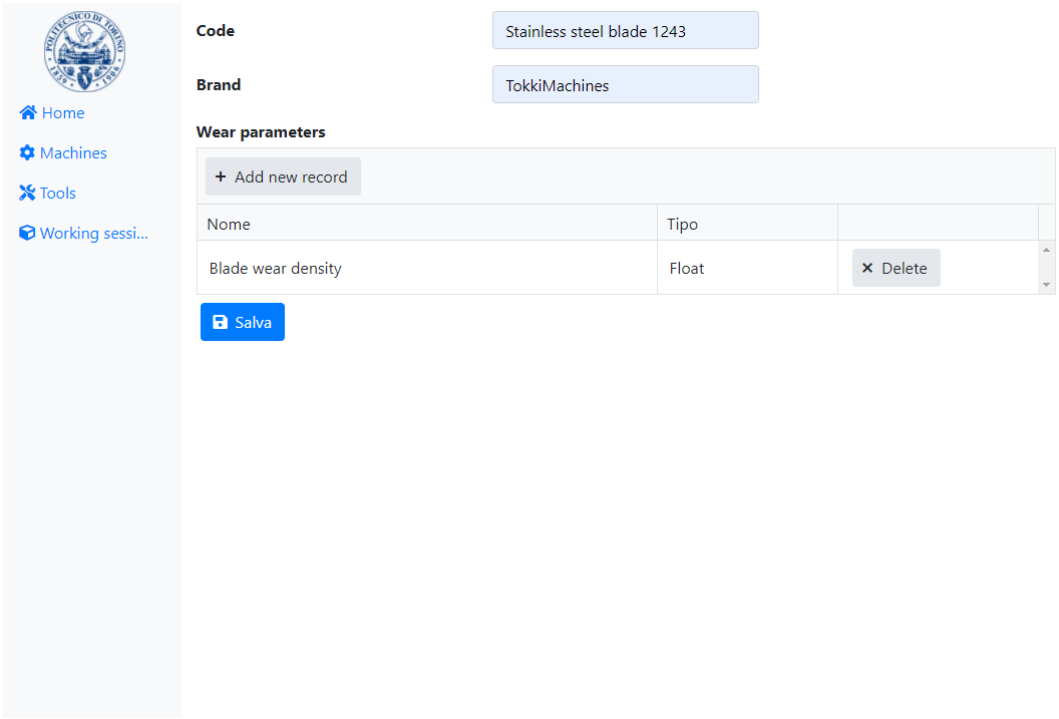

Figure 3.14: Tool registration page

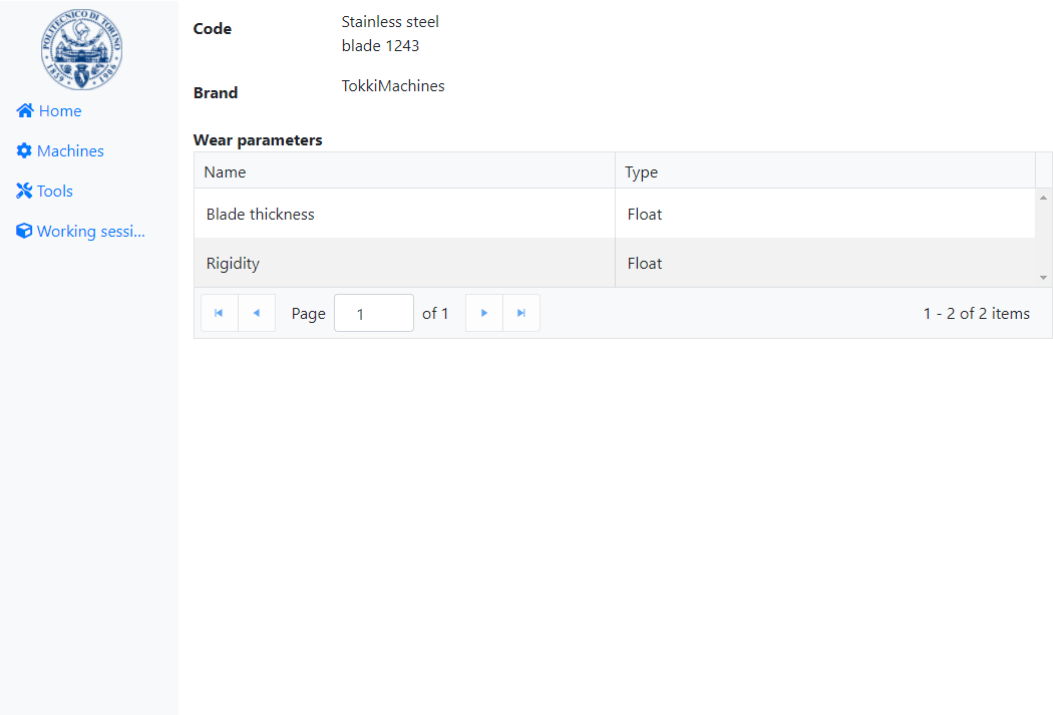

Figure 3.15: Tool details page

3.4.3 Work sessions history

Through this section it is possible to view the history of the work sessions carried out by the machines. The page is shown in Figure 3.16. As with other entities, it is present a page to register a new work session, shown in Figure 3.17. It is necessary to specify the machine that performed the work session, the tool that was mounted in the machine and the value of the working parameters required by the machine. The work session details page shows important details about the work session, for example the value assumed by the sensors during work session. The operator can also specify the value of the tool wear parameters at the end of the machining, in order to trace the evolution of the tool wear.

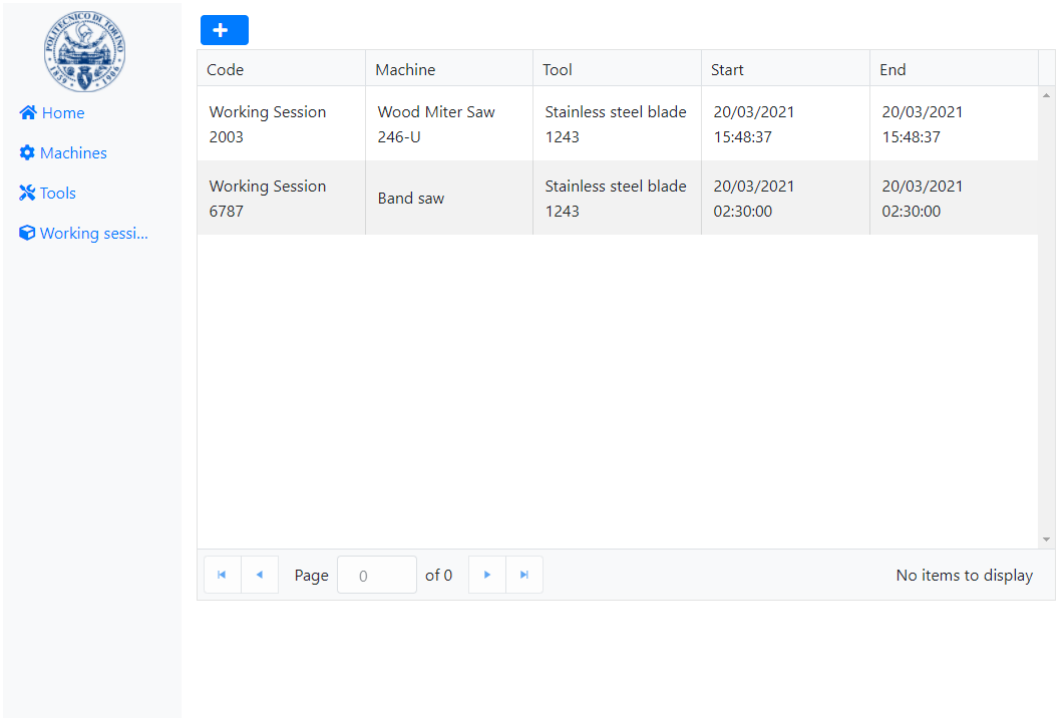

Figure 3.16: Work sessions history page

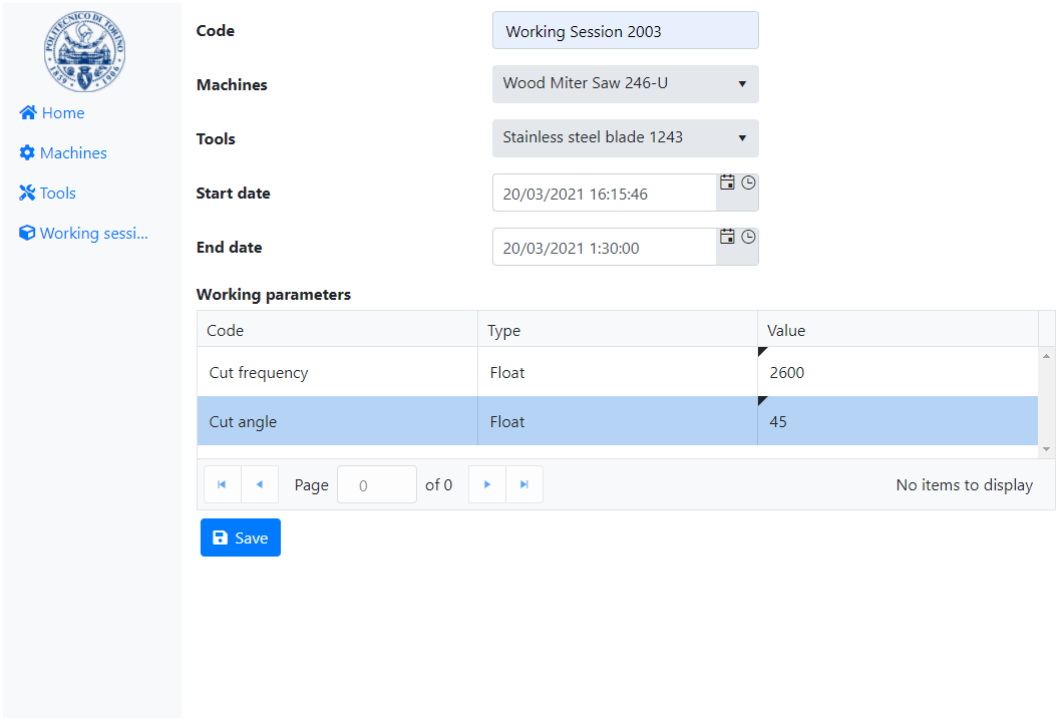

Figure 3.17: Work session registration page

Chapter 4

Use case: data collection from a milling machine for predictive maintenance applications

In order to validate the applicability of the proposed framework, the application database was populated using real data collected in the context of milling operations. In this context, the study and validation of machine monitoring techniques for predictive maintenance purposes is very important, as tools are used that wear out as an integral part of the machining process.

4.1 Introduction

Milling machines operate by removing material from the surface of the object they are working on. The milling process can generally be adapted and calibrated for each specific machining through some operational parameters, typically configurable in the milling machine. Some of the parameters used are shown in Table 4.1

Another important aspect to consider is the quality of the tool used for milling. In particular, the material can greatly affect the life of the tool.

As the number of jobs increases, the wear mechanisms affecting the tool become ever more important until a critical level is reached, and the tool is no more capable to ensure the required quality level. The wear of cutting

tools is of different types, but the most important is certainly the flank wear, as this type of wear is the one that generally progresses faster and limits the life time of the tool. Flank wear occurs at the tool flanks, which is the part of the tool in contact with the object being cutted. Flank wear is usually measured considering the maximum observable width in millimeters of the wear band VB.

Name	Description
Spindle speed	Rotational frequency of the spindle of the machine, it is measured in revolutions per minute (RPM).
Cutting speed	Rotational speed of the tool, usually measured in m/min.
Feed rate	Velocity at which the cutter is fed, usually measured in mm/min
Depth of cut	How deep the tool is cutting into the material, usually measured in mm

Table 4.1: Milling machine working parameters

4.2 Dataset

The NASA Milling Dataset [59] was used. This dataset stores data collected during milling operations under different machine configurations, operating conditions and tools used. During the milling, data from six sensors positioned in different positions were collected. Vibration, acoustic and current sensors were used. At the end of some work sessions flank wear (VB) was measured.

The data is organized in a 1x167 Matlab struct array composed by the fields listed in Table 4.2.

Sixteen different cases were considered, each with different parameters and operating conditions. Some cases have been repeated for multiple processes. The cases are enumerated in Table 4.3.

Name	Description
case	Case number, between 1 and 16
run	Counter for experimental runs in each case
VB	Flank wear, measured after runs. This field is optional
time	Duration of the experiment (restarts for each case)
DOC	Depth of cut (does not vary for each case)
feed	Feed rate (does not vary for each case)
material	Cutting tool material (does not vary for each case)
smcAC	AC spindle motor current values
smcDC	DC spindle motor current values
vib_table	Table vibration sensor values
vib_table	Spindle vibration sensor values
AE_table	Acoustic emission at table
AE_spindle	Acoustic emission at spindle

Table 4.2: Dataset fields

Case	Depth of cut	Feed rate	Tool material
1	1.5	0.5	1 - cast iron
2	0.75	0.5	1 - cast iron
3	0.75	0.25	1 - cast iron
4	1.5	0.25	1 - cast iron
5	1.5	0.5	2 - steel
6	1.5	0.25	2 - steel
7	0.75	0.25	2 - steel
8	0.75	0.5	2 - steel
9	1.5	0.5	1 - cast iron
10	1.5	0.25	1 - cast iron
11	0.75	0.25	1 - cast iron
12	0.75	0.25	1 - cast iron
13	0.75	0.25	2 - steel
14	0.75	0.5	2 - steel
15	1.5	0.25	2 - steel
16	1.5	0.5	2 - steel

Table 4.3: Considered cases

4.3 Data ingestion

The data was entered into the web application's date base using a single-purpose Python script.

The script has populated the **machine** registry with a reference milling machine. This machine has been registered with two mandatory **working parameters**, which are precisely those considered in the dataset: depth of cut and feed.

The tools used in the machining have been registered. For each case a single **tool** was generated, considering that in the experiments the same tool was used in the various runs of the same case. All tools were registered with a **wear Parameter** to describe flank wear.

A different **work session** was therefore generated for each run. The work session has always been created considering the reference milling machine and the correct instance of the tool, which is therefore the same for a fixed case number. The **working parameters**, required by the reference milling machine, have been configured as described in the various runs of the dataset.

4.4 Results

The data model developed in the application was found to be suitable for representing real-world data referred to the context of milling and monitoring for predictive maintenance purposes.

Figure 4.1 shows the work sessions history page with the data coming from the dataset. Going into the details of the single work session, it is possible to view the values of the working parameters and the time series of the sensor values during the work session, as shown in Figure 4.2. The page also shows the value of the wear parameters at the end of the work session, which in this case is not a user input, but has been imported from the dataset.

Many tools have been generated, for each case of the dataset, visible in the grid on the tools list page. Going to the details page of the single cutting tool, it is possible to view the evolution of the wear parameters, so it is shown how the VB has varied in the various runs, with the case fixed, considering that a tool has been generated for each case. An example detail page is shown in Figure 4.3, where the increasing wear trend can also be easily noticed.

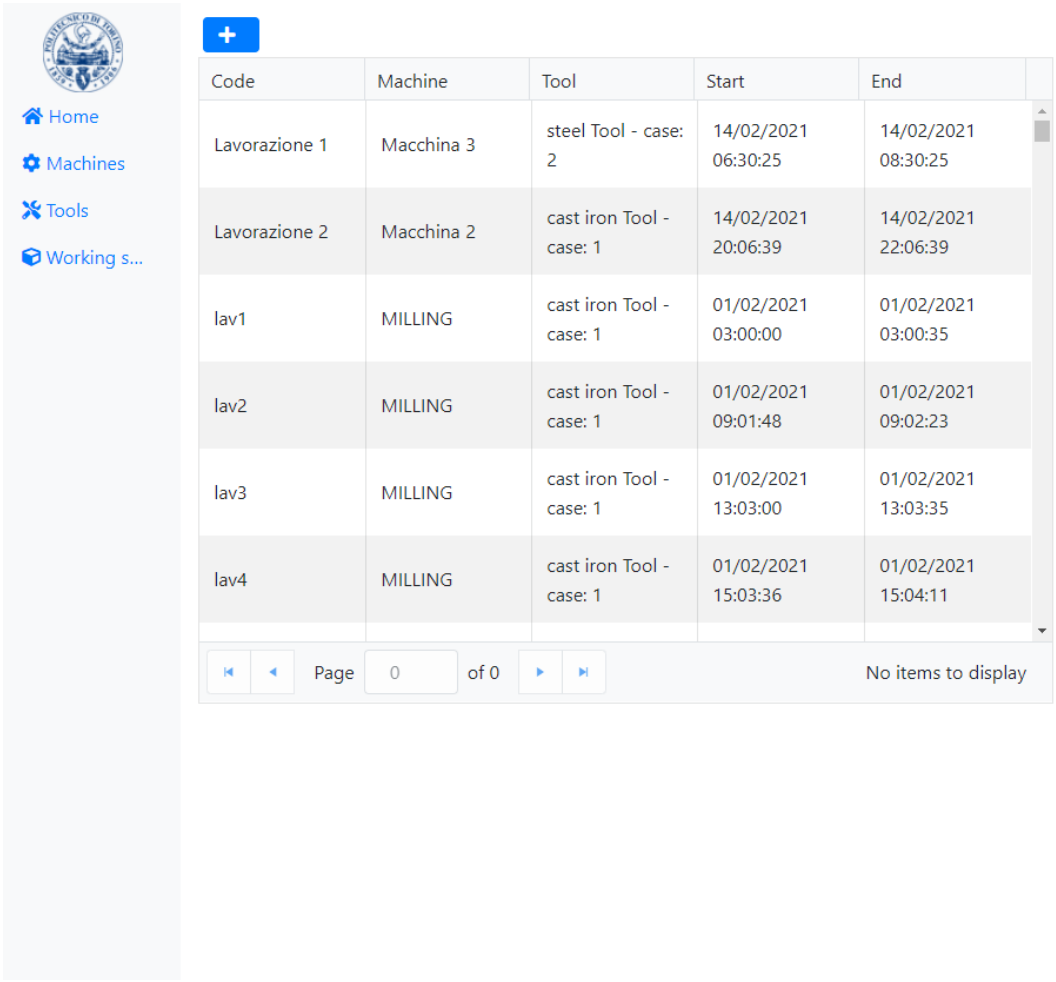

Figure 4.1: Work sessions history page - NASA Milling dataset

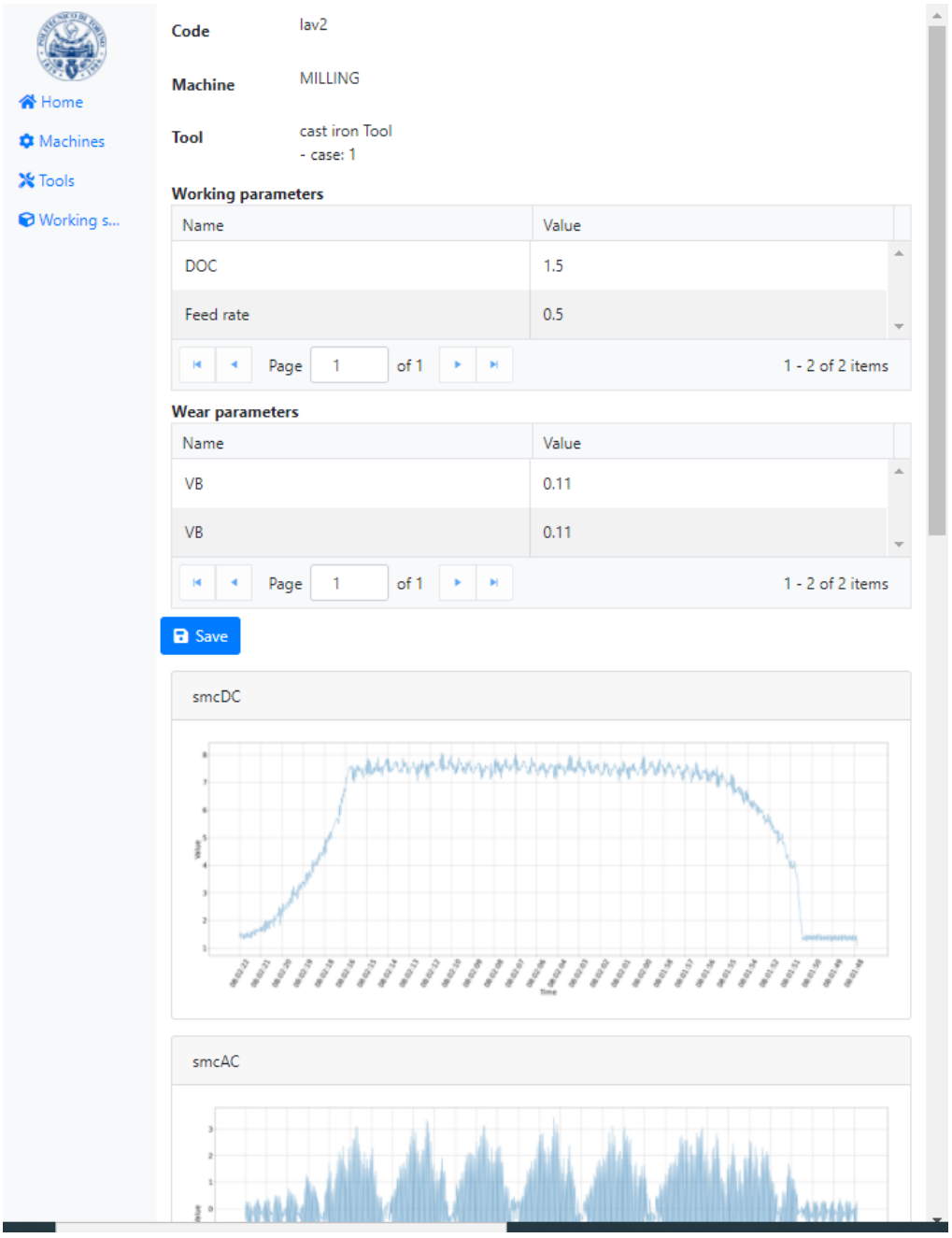

Figure 4.2: Work sessions details page - NASA Milling dataset

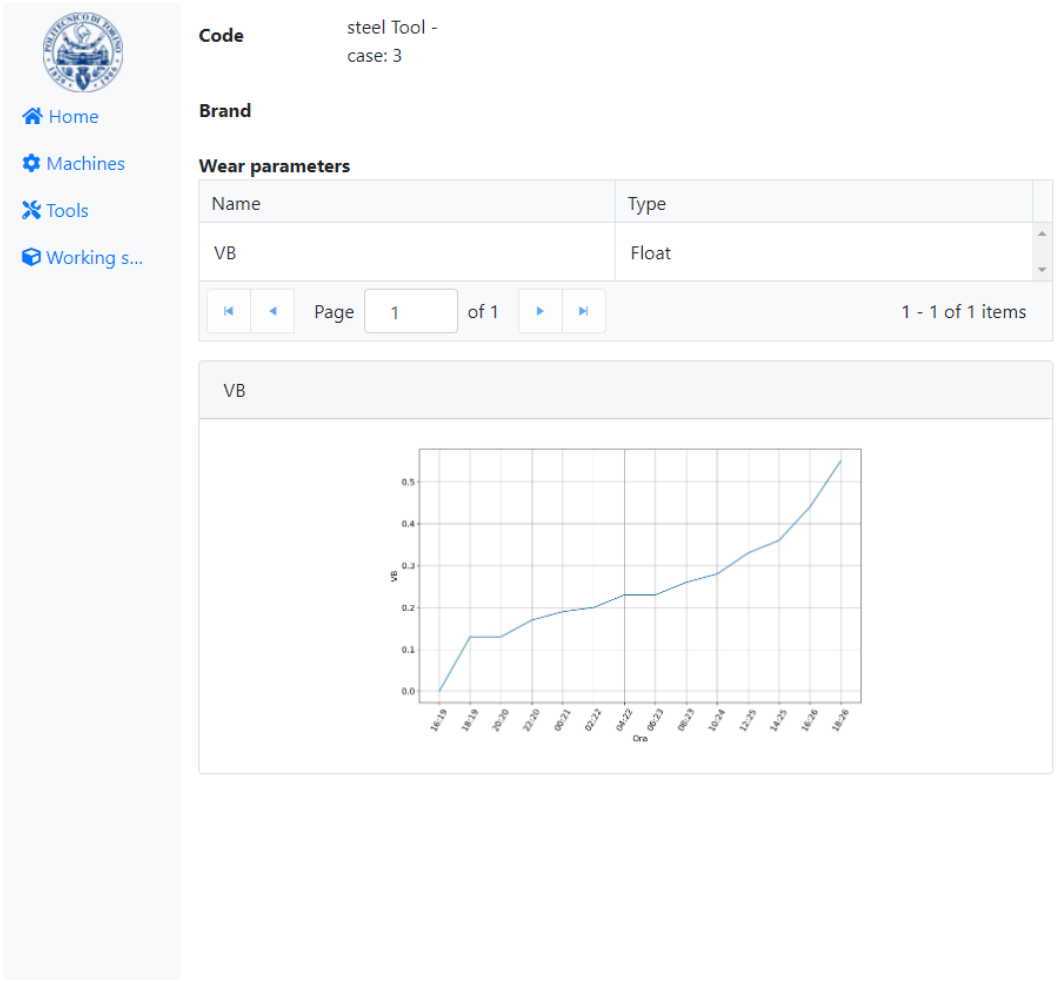

Figure 4.3: Tool details page - NASA Milling dataset

Chapter 5

Conclusions and future Works

The development of IoT technologies has made possible the interconnection of hardware and software devices, generating complex networks of cyber-physical systems that cooperate to make processes more efficient or to collect a large amount of data that can be used to solve problems that were previously difficult to solve. In this thesis the protocols that can be used to develop networks of cyber-physical systems in the industrial and manufacturing environment have been explored and discussed, with particular attention to wireless protocols that have a deterministic behavior and can therefore guarantee high levels of reliability and low latency, even in industrial environments where propagation conditions can be unstable or bad. Some possible applications of sensor and actuator networks were then discussed and a software framework for data collection for predictive maintenance scenarios was developed. The framework was developed with open-source, free and widely used technologies. The applicability of the proposed framework was confirmed considering the data collected during milling work sessions performed under different operating conditions and using different cutting tools. The proposed framework can be improved and extended in functionality by implementing in future works a module for predicting the wear of the machines and tools, to be combined with the data collection and visualization module already developed, providing a complete predictive maintenance platform.

It is also necessary to investigate which sensors are necessary and where they can be positioned to obtain information related to the wear of the machines and tools, considering the different types of machines and processes

commonly used in manufacturing processes. Different predictive models, statistical analysis and machine learning algorithms need to be tested to get the best predictive performance in each case.

Bibliography

[1] G. Hohpe. *Enterprise integration patters: TOC*. 2015. URL: https://www.enterpriseintegrationpatterns.com/patterns/messaging/toc.html (visited on 03/13/2021).

[2] W. Shi et al. «Edge Computing: Vision and Challenges». In: *IEEE Internet of Things Journal* 3.5 (2016), pp. 637–646. DOI: 10.1109/JIOT.2016.2579198.

[3] A. van den Bossche, R. Dalcé, and T. Val. «OpenWiNo: An open hardware and software framework for fast-prototyping in the IoT». In: *2016 23rd International Conference on Telecommunications (ICT)*. 2016, pp. 1–6. DOI: 10.1109/ICT.2016.7500490.

[4] F. Piccialli et al. «Decision Making in IoT Environment through Unsupervised Learning». In: *IEEE Intelligent Systems* 35.1 (2020), pp. 27–35. DOI: 10.1109/MIS.2019.2944783.

[5] Zaheer Allam and Zaynah A. Dhunny. «On big data, artificial intelligence and smart cities». In: *Cities* 89 (2019), pp. 80–91. ISSN: 0264-2751. DOI: https://doi.org/10.1016/j.cities.2019.01.032. URL: https://www.sciencedirect.com/science/article/pii/S0264275118315968.

[6] Tai-hoon Kim, Carlos Ramos, and Sabah Mohammed. «Smart City and IoT». In: *Future Generation Computer Systems* 76 (2017), pp. 159–162. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2017.03.034. URL: https://www.sciencedirect.com/science/article/pii/S0167739X17305253.

[7] Małgorzata Skrzeszewska and Justyna Patalas-Maliszewska. «Assessing the Effectiveness of Using the MES in Manufacturing Enterprises in the Context of Industry 4.0». In: *Distributed Computing and Artificial Intelligence, 16th International Conference, Special Sessions*. Ed.

by Enrique Herrera-Viedma et al. Cham: Springer International Publishing, 2020, pp. 49–56. ISBN: 978-3-030-23946-6.

[8] M. Fowler. *EnterpriseApplication*. 2015. URL: https://martinfowler.com/bliki/EnterpriseApplication.html (visited on 03/13/2021).

[9] Despina Galani, Efthymios Gravas, and Antonios Stavropoulos. «The Impact of ERP Systems on Accounting Processes». In: *International Journal of Economics and Management Engineering* 4.6 (2010), pp. 774–779. ISSN: eISSN: 1307-6892. URL: https://publications.waset.org/vol/42.

[10] Pouria Zand et al. «Wireless Industrial Monitoring and Control Networks: The Journey So Far and the Road Ahead». In: *Journal of Sensor and Actuator Networks* 1.2 (2012), pp. 123–152. ISSN: 2224-2708. DOI: 10.3390/jsan1020123. URL: https://www.mdpi.com/2224-2708/1/2/123.

[11] M. Raza et al. «A Critical Analysis of Research Potential, Challenges, and Future Directives in Industrial Wireless Sensor Networks». In: *IEEE Communications Surveys Tutorials* 20.1 (2018), pp. 39–95. DOI: 10.1109/COMST.2017.2759725.

[12] Texas Instruments. *An inside look at industrial Ethernet communication protocols*. 2021. URL: https://www.ti.com/lit/wp/spry254b/spry254b.pdf (visited on 03/13/2021).

[13] Rolf H. Weber. «Internet of Things – New security and privacy challenges». In: *Computer Law Security Review* 26.1 (2010), pp. 23–30. ISSN: 0267-3649. DOI: https://doi.org/10.1016/j.clsr.2009.11.008. URL: https://www.sciencedirect.com/science/article/pii/S0267364909001939.

[14] Grafana. *Grafana website*. 2021. URL: https://grafana.com/grafana/dashboards (visited on 03/13/2021).

[15] EETimes. *Industrial Ethernet–The basics*. 2021. URL: https://www.eetimes.com/industrial-ethernet-the-basics/# (visited on 03/13/2021).

[16] Microsoft. *Implementing event-based communication between microservices (integration events)*. 2021. URL: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/multi-container-microservice-net-applications/integration-event-based-microservice-communications (visited on 03/13/2021).

[17] N. Brown. *Message Broker or Bus – what's the difference?* 2021. URL: https://neiljbrown.com/2017/05/13/message-broker-or-bus-whats-the-difference/ (visited on 03/13/2021).

[18] Saleem Raza, Muhammad Faheem, and Mesut Günes. «Industrial wireless sensor and actuator networks in industry 4.0: Exploring requirements, protocols, and challenges-A MAC survey». In: *International Journal of Communication Systems* 32 (Aug. 2019), e4074. DOI: 10.1002/dac.4074.

[19] Brian P Crow et al. «IEEE 802.11 wireless local area networks». In: *IEEE Communications magazine* 35.9 (1997), pp. 116–126.

[20] Adnan Aijaz. «High-Performance Industrial Wireless: Achieving Reliable and Deterministic Connectivity over IEEE 802.11 WLANs». In: *IEEE Open Journal of the Industrial Electronics Society* PP (Mar. 2020), pp. 1–1. DOI: 10.1109/OJIES.2020.2983259.

[21] Ed Callaway et al. «Home networking with IEEE 802.15. 4: a developing standard for low-rate wireless personal area networks». In: *IEEE Communications magazine* 40.8 (2002), pp. 70–77.

[22] A. Ince et al. «Data Reliability and Latency Test for ZigBee-based Smart Home Energy Management Systems». In: June 2014.

[23] Feng Xia et al. «Service differentiated and adaptive CSMA/CA over IEEE 802.15.4 for cyber-physical systems». In: *TheScientificWorldJournal* 2013 (Oct. 2013), p. 947808. DOI: 10.1155/2013/947808.

[24] Jianping Song et al. «WirelessHART: Applying wireless technology in real-time industrial process control». In: *2008 IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE. 2008, pp. 377–386.

[25] Sabina Jeschke et al. «Industrial internet of things and cyber manufacturing systems». In: *Industrial internet of things*. Springer, 2017, pp. 3–19.

[26] Stig Petersen and Simon Carlsen. «WirelessHART versus ISA100. 11a: The format war hits the factory floor». In: *IEEE Industrial Electronics Magazine* 5.4 (2011), pp. 23–34.

[27] Remous-Aris Koutsiamanis et al. «From best effort to deterministic packet delivery for wireless industrial IoT networks». In: *IEEE Transactions on Industrial Informatics* 14.10 (2018), pp. 4468–4480.

[28] Carles Gomez, Joaquim Oller, and Josep Paradells. «Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology». In: *Sensors* 12.9 (2012), pp. 11734–11753.

[29] Virginia Commonwealth University. *BLE specification*. 2021. URL: https://music.lab.vcu.edu/boss/ble-specification/ (visited on 03/13/2021).

[30] Chatschik Bisdikian. «An overview of the Bluetooth wireless technology». In: *IEEE Communications magazine* 39.12 (2001), pp. 86–94.

[31] V. Díez et al. «Reliability Evaluation of Bluetooth Low Energy for Industry 4.0». In: *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2019, pp. 1148–1154. DOI: 10.1109/ETFA.2019.8869211.

[32] Raul Rondon, Mikael Gidlund, and Krister Landernäs. «Evaluating Bluetooth Low Energy Suitability for Time-Critical Industrial IoT Applications». In: *International Journal of Wireless Information Networks* 24 (Sept. 2017), pp. 1–13. DOI: 10.1007/s10776-017-0357-0.

[33] Sinem Coleri Ergen. «ZigBee/IEEE 802.15. 4 Summary». In: *UC Berkeley, September* 10.17 (2004), p. 11.

[34] Geoff Mulligan. «The 6LoWPAN Architecture». In: *Proceedings of the 4th Workshop on Embedded Networked Sensors*. EmNets '07. Cork, Ireland: Association for Computing Machinery, 2007, pp. 78–82. ISBN: 9781595936943. DOI: 10.1145/1278972.1278992. URL: https://doi.org/10.1145/1278972.1278992.

[35] Gee Keng Ee et al. «A review of 6LoWPAN routing protocols». In: *Proceedings of the Asia-Pacific Advanced Network* 30 (2010), pp. 71–81.

[36] Tim Berners-Lee, Roy Fielding, and Henrik Frystyk. *Hypertext transfer protocol–HTTP/1.0*. 1996.

[37] Zach Shelby, Klaus Hartke, and Carsten Bormann. «The constrained application protocol (CoAP)». In: (2014).

[38] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. «MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks». In: *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*. IEEE. 2008, pp. 791–798.

[39] Gerardo Pardo-Castellote. «Omg data-distribution service: Architectural overview». In: *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.* IEEE. 2003, pp. 200–206.

[40] *Node-RED*. https://nodered.org/. (Accessed on 03/21/2021).

[41] Python. *PEP 20 - The Zen of Python*. 2021. URL: https://www.python.org/dev/peps/pep-0020/ (visited on 03/13/2021).

[42] Zaggle. *2018 Kaggle Machine Learning Data Science Survey*. 2018. URL: https://www.kaggle.com/kaggle/kaggle-survey-2018 (visited on 03/13/2021).

[43] Armin Ronacher. *April 1st Post Mortem*. 2021. URL: https://lucumr.pocoo.org/2010/4/3/april-1st-post-mortem/ (visited on 03/13/2021).

[44] Django. *Django Project*. 2021. URL: https://www.djangoproject.com/ (visited on 03/13/2021).

[45] Armin Ronacher. *Jinja Project*. 2021. URL: https://jinja.palletsprojects.com/ (visited on 03/13/2021).

[46] Armin Ronacher. *Flask design principles*. 2021. URL: https://flask.palletsprojects.com/en/1.1.x/design/#design (visited on 03/13/2021).

[47] L. A. Stonebraker M.; Rowe. *The design of POSTGRES*. 1986. URL: http://db.cs.berkeley.edu/papers/ERL-M85-95.pdf (visited on 03/13/2021).

[48] PostgreSQL. *PostgreSQL isolation levels*. 2021. URL: https://www.postgresql.org/docs/9.5/transaction-iso.html (visited on 03/13/2021).

[49] PostgreSQL. *PostgreSQL JSON types*. 2021. URL: https://www.postgresql.org/docs/9.4/datatype-json.html (visited on 03/13/2021).

[50] SqlAlchemy. *SQLAlchemy Website*. 2021. URL: https://www.sqlalchemy.org/ (visited on 03/13/2021).

[51] Node-RED. *Node-RED web page*. 2021. URL: https://nodered.org/ (visited on 03/13/2021).

[52] NodeJS. *NodeJS web page*. 2021. URL: https://nodejs.org/ (visited on 03/13/2021).

[53] NodeJS. *Overview of Blocking vs Non-Blocking*. 2021. URL: https://nodejs.org/en/docs/guides/blocking-vs-non-blocking/ (visited on 03/13/2021).

[54] Docker. *Docker*. 2021. URL: https://www.docker.com/ (visited on 03/13/2021).

[55] Kubernetes. *Kubernetes*. 2021. URL: https://kubernetes.io/ (visited on 03/13/2021).

[56] Docker. *Docker overview*. 2021. URL: https://docs.docker.com/get-started/overview/ (visited on 03/13/2021).

[57] Docker. *Docker Compose overview*. 2021. URL: https://docs.docker.com/compose/ (visited on 03/13/2021).

[58] Wikipedia. *Domain model*. 2021. URL: https://en.wikipedia.org/wiki/Domain_model (visited on 03/13/2021).

[59] Vinayak Tyagiv. *NASA Milling Dataset*. 2021. URL: https://www.kaggle.com/vinayak123tyagi/milling-data-set-prognostic-data (visited on 03/13/2021).