Chapter

Interaction Protocols for Multi-Robot Systems in Industry 4.0

Edi Moreira M. de Araujo, Augusto Loureiro da Costa and Alejandro R.G. Ramirez

Abstract

In this chapter, the main methods of communication among multi-robot systems involved in Machine-to-Machine (M2M) applications, especially with regard the communication, reliability, stability and security among these robots, presenting various concepts through papers already published. A comparative study was carried out between two communication protocols applied in M2M technologies, the Queue Telemetry Transport (MQTT) developed by IBM along with Eurotech and the Constrained Application Protocol (CoAP). A study and survey of the characteristics of each of the protocols was carried out, as well as the method of operation of each of them and how both can be used in applications involving multiple robots. It was concluded that both protocols are considered ideal for use in in applications involving multi-robot systems. However, although the two protocols have been designed for application in environments with limited communication, the MQTT exchange protocol has advantages over CoAP, as a lower ovehead between message exchanges.

Keywords: Machine-to-Machine, protocol, multi-robot system

1. Introduction

The industry, in the last century, has undergone changes in the way it operates, generating innovation and profound social and economic changes. According to [1], is the beginning of a revolution called Industry 4.0. This industrial revolution is based on several concepts, among them, the Cyber-Physical System, Internet of Things (IoT), big data analytics, Machine-toMachine (M2M) and cloud computing. All of these concepts aim to meet the requirements of an advanced manufacturing system, promoting the integration of an entire supply chain.

The authors in [2], Industry 4.0 creates what has been called the smart factory. This factory has a modular structure in which cyber-physical systems monitor physical processes, creating a virtual copy of the physical world, making decentralized decisions using the IoT that has communication with each other and with humans in real time. These smart factories aim to solve several challenges found in large industrial systems, due not only to the increase in the complexity of processes and products, but also to the increase in the varieties of these products, which must be placed on the market due to the reduced life cycle. Thus, there is a need to make production processes more flexible, characterized according to [3] in production systems with distributed units, composed of a conglomerate of autonomous units, which operate in cooperation in an integrated manner. Such units can be industrial automation machines, manipulating robots, mobile robots or microprocessed remote units.

Distributed production systems, often composed of robot machines, are designed with the objective of providing efficiency and rationality in the use of distributed production resources, in order to favor the manufacture of products, in a dynamic and fast way. The production units must be able to respond, in an intelligent and effective manner, to unforeseen disturbances in the external environment, maintaining controlled and continuous production [3]. Considering the need to plan and control systems for these units, complete robotization of productive systems, which in turn need means or protocols of interaction and coordination between them.

Therefore, the justification for proposing this chapter is to deepen the studies on the interaction protocols for existing multi-robot systems and to design a new protocol that can be applied to concepts related to Industry 4.0, taking into account the characteristics of self- organization of robotics structure based on the concept of industrial agents.

This chapter is divided as follows. In Section 2, the Machine-to-Machine (M2M) is presented, with its levels explained. In Section 3, protocols MQTT and CoAP are presented, identifying their main characteristics and limitations. A comparison between the protocols (MQTT and CoAP) will be demonstrated in the Section 4. Section 5 is shown some studies that used MQTT protocol, along with Robot Operating System (ROS) in the context of Insdustry 4.0, in addition to presenting the conclusions of the chapter.

2. Machine-to-machine communications

According to [4] the term M2M Communications, it is the machine to machine communication, which enables the transmission of data across different devices without the need for human intervention.

This communication opens up an immense range of applications that can, among other things, register, process and manipulate the data generated and transmitted by the objects that are interconnected. For example, an application that continuously receives data from a sensor that measures the temperature of an environment and, based on the data obtained, can generate statistics that describe the sensor readings over a period of time and then send an alert via e-mail or Short Message Service (SMS) to one or more individuals if the temperature has reached very high levels, or even publish this information to another device that could use it in another way, among other things.

M2M applications have the potential to become a trend in the development of software in the coming years in view of the various sectors (such as industrial and home automation) that need an automated solution that integrates the devices that are part of their environment. Devices that are part of an M2M network have the ability to at least collect data from a given environment and transmit it to an application through a connection. Eventually, these devices will not be able to transmit this data directly to other equipment, it is necessary to use a gateway to be an intermediary for this transmission.

Thus, the M2M can be defined as a number of technologies that aims to establish communication between devices with the ability to transmit information for a particular application without the interference of a human action.

2.1 M2M architecture

According to [4], the M2M architecture (**Figure 1**) is divided into three domains, Devices, gateway and Network. The components of these domains are described as follows:

- In the domain of devices:

  M2M device: A device that runs one or more M2M applications using M2M service capabilities. The M2M device connect to network domain in the following for two manners, by Direct Connectivity (M2Mdevices connect to the network domain via the access network) and Gateway as a Network Proxy (The M2M device connects to the network domain via an M2M gateway);

- M2M Gateway: object that runs M2M applications, using M2M services and acting as a proxy between M2M devices and the network domain;

  Core network: Its main function is to ensure the functioning of the network with connectivity via IP and other means of connectivity, as well as control functions of network services, interconnection and roaming;

  Access network allows M2M devices and gateways to communicate with the core network. According to [5], examples of M2M include technologies such as IEEE 802.15.1, Bluetooth, personal area network, among others, or local networks such as power line communication with PLC and Wireless M-BUS;

- M2M area Network:

  M2M Network Area: provides the connection between devices and gateways;

  Provide M2M functions that are shared by different applications;

  M2M applications: run the service logic;

  Network Management: brings together all the functions necessary to manage the network core and the access network;

  M2M management roles: Consist of the roles needed to manage service capabilities within the network domain;

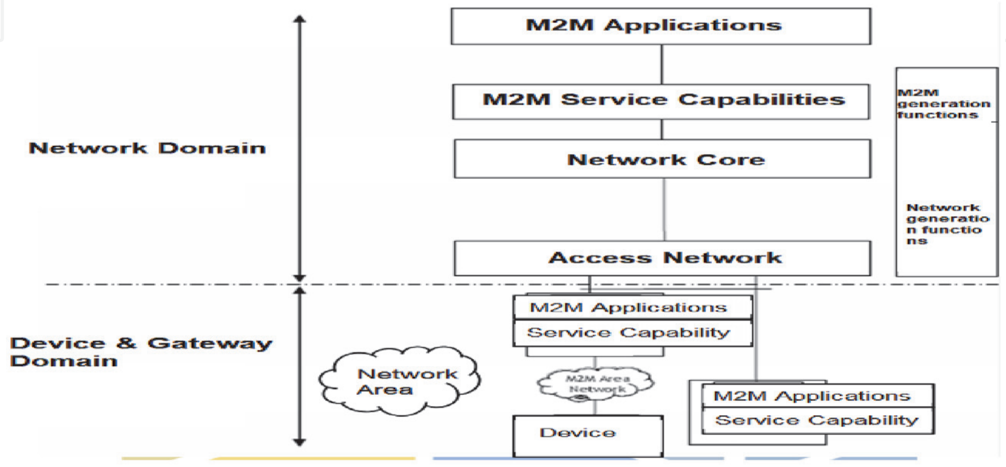


Figure 1.
M2M architecture.

In **Figure 2** a simplified version of the M2M architecture is presented, where important elements of the architecture are shown, in addition to defining the application domain.

2.2 M2M systems categories

The authors in [6] categorize M2M systems into two types, namely dynamic M2M systems and static M2M systems. The main difference between the two is its topology, where in dynamic systems, some nodes (for example, M2M devices and M2M gateways) are moving, that is, the topology is changing over time, resulting in a change in the quality of communication, and dynamic resource allocation. Examples of dynamic M2M systems include: vehicle M2M system, the medical M2M system and the robotic M2M system. In contrast, the topology for static M2M systems remains unchanged for a relatively long time, as an example the M2M power system, domestic M2M system and the industrial M2M system.

3. M2M communication protocols

There are several communication protocols responsible for managing the transfer of data between computers on the internet, among them we can mention some examples such as HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol) and SFTP (SSH File Transfer Protocol). When the communication needs to be made between two or more devices (or several applications) connected in a network, the need arises to have a protocol that manages this communication, that is, the exchange of data between the devices in an efficient way considering the characteristics and restrictions imposed by the environment. According to [4], in this scenario, two protocols arise that can be used in restricted environments: Messaging Queue Telemetry Transport (MQTT) and the Constrained Application Protocol (CoAP).

The following will present the MQTT and CoAP protocols, identifying their main characteristics and limitations, in addition to highlighting the best scenarios where each can be applied in the context of Industry 4.0.

3.1 MQTT: message queue telemetry transport

Created by IBM in 1999, MQTT is an open source protocol designed to be simple, lightweight and easy to implement. It is a messaging protocol based on the

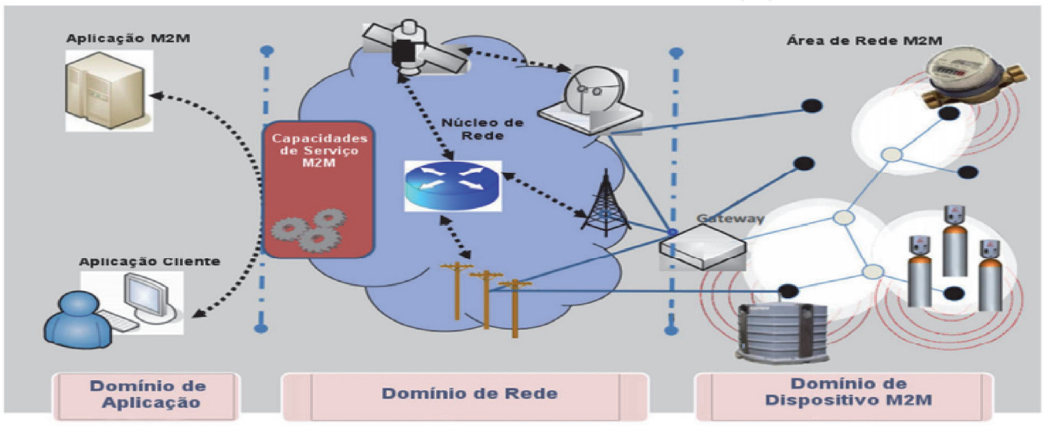


Figure 2.
Simplified version of high-level architecture.

publish/subscribe architecture (**Figure 3**), which has a small transport overhead (fixed byte header of 2 bytes), making MQTT an interesting solution for unreliable networks with limited resources, such as bandwidth and high latency [4]. This protocol is based on a broker (**Figure 4**) using the message pattern *publish/ subscribe*, while the server broker acts as a intermediary for messages sent from a device that publishes to subscribing customers, providing a distribution of one-to-many messages decoupled from the use case of the application.

For a client to send a message, it needs to publish it in a topic (called a MQTT broker) (**Figure 4**). If another client wants to receive the content of this message, he will have to subscribe to this same topic. A client can publish or subscribe to multiple topics at the same time, and there may be situations where the publication or subscription on a topic is disputed between different clients, thus having a system that is asynchronous [7].

The PDU (Protocol Data Unit) of the MQTT protocol is encapsulated by the TCP (Transmission Control Protocol) protocol, that is, the MQTT header and data are sent in the TCP data area [8]. In this way, the MQTT protocol messages have a fixed header (**Figure 5**) composed of two bytes, where the first byte contains the field that identifies the type of message, such as also the markers (DUP, QoS level and RETAIN). There is a version of MQTT, called MQTT-SN (MQTT Sensor Network), where PDU is encapsulated by the UDP protocol, which, in turn, is encapsulated by the IP or the 6LowPAN protocol. One of the main differences between two standards, in addition to the network layer they focus on, is the simplification of

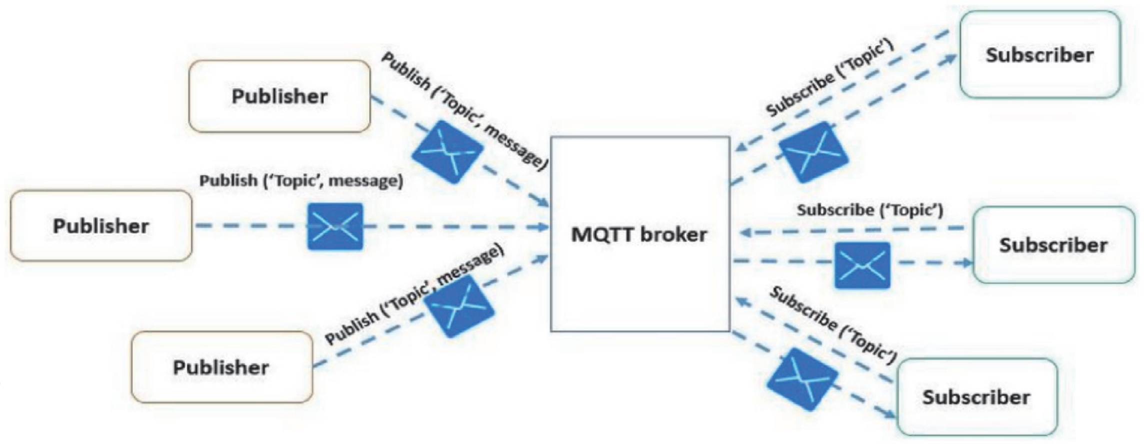


Figure 3.
Broker Publisher/subscriber messaging template.

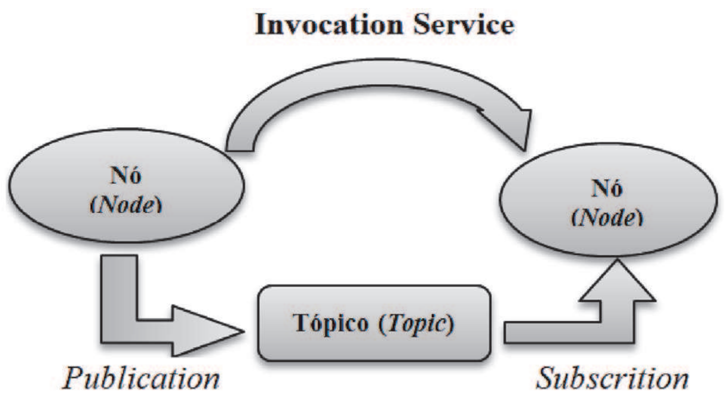
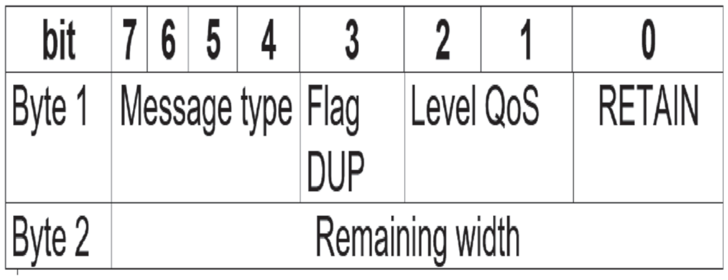
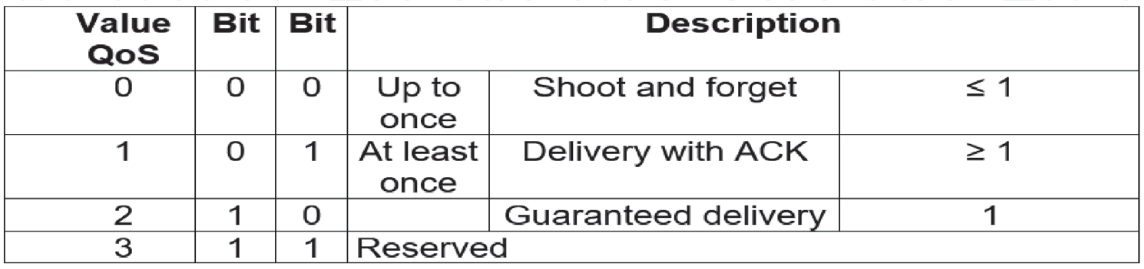


Figure 4.
Publisher/subscriber messaging template.



bit	7	6	5	4	3	2	1	0
Byte 1	Message type				Flag DUP	Level QoS		RETAIN
Byte 2	Remaining width							

Figure 5.
Fixed header of an MQTT message.



Value QoS	Bit	Bit	Description		
0	0	0	Up to once	Shoot and forget	≤ 1
1	0	1	At least once	Delivery with ACK	≥ 1
2	1	0		Guaranteed delivery	1
3	1	1	Reserved		

Figure 6.
QoS levels.

messages exchanged between broker and clients, using predefined topic identifiers and short names of topics in addition to a short message format [7].

According to [8], the PDU of the MQTT protocol is encapsulated by the TCP protocol, that is, the MQTT header and data are sent in the TCP data area (**Figure 5**). In this way the MQTT protocol messages have a fixed header (**Figure 5**) composed of two bytes, where the first byte contains the field that identifies the type of the message, as well as the markers (DUP, QoS level and RETAIN). There is a version of MQTT, called MQTT-SN (MQTT Sensor Network), where your PDU is encapsulated by the UDP protocol, which, in turn, is encapsulated by the IP or the 6LowPAN protocol. One of the main differences between two standards, in addition to the network layer they focus on, is the simplification of messages exchanged between broker and clients, using predefined topic identifiers and short names of topics in addition to a short message format [5].

As can be seen in **Figure 5**, "byte 1" is responsible for four fields:

- DUP (Duplicate delivery): Marker that occupies bit 4 and is activated when the server tries to resend messages of type PUBLISH, PUBREL, SUBSCRIBE or UNSUBSCRIBE (**Figure 5**).

- QoS: This marker represents the reliability of message delivery, indicating the level of guarantee of delivery of a PUBLISH message. You can have up to three levels of guarantee (**Figure 6**). Level 0 is used by those who publish/send the message at most once and does not check whether the message has reached its destination. This lower level is called "fire-and-forget" and the message can be lost depending on network conditions. In Level 1, called the recognized delivery, who publishes/sends a message at least once and check the delivery status using a status check message. However, if this verification message loses, the server broker can possibly send the same message twice, since there was no confirmation that the message has been delivered. Finally we have the QoS2, called guaranteed delivery due to its complicated process, there may be delays end-to-end larger, but no lost messages at this level. The higher the level of QoS, the greater is the packet exchange. If the loss of messages a problem, a lower level of QoS can be used, resulting in less consumption of available bandwidth and less end-to-end delay, which represents limited networks wired or wireless. To further reduce the use of the band, the UDP can be used instead of TCP, but with reduced guaranteed message delivery [9].

3.2 CoAP: constrained application protocol

The Constrained Application Protocol (CoAP), created by the Internet Engineering Task Force (IETF) working group called restricted RESTful environments (CoRE), has been adapted from HTTP, being optimized for devices with limited processing power and capacity, generally applied to intelligent objects in IoT environments [9]. CoAP acts on the UDP transport layer, specifying a minimum set of restrictions such as POST, GET, PUT and DELETE, with some support for resource storage and discovery of embedded resources.

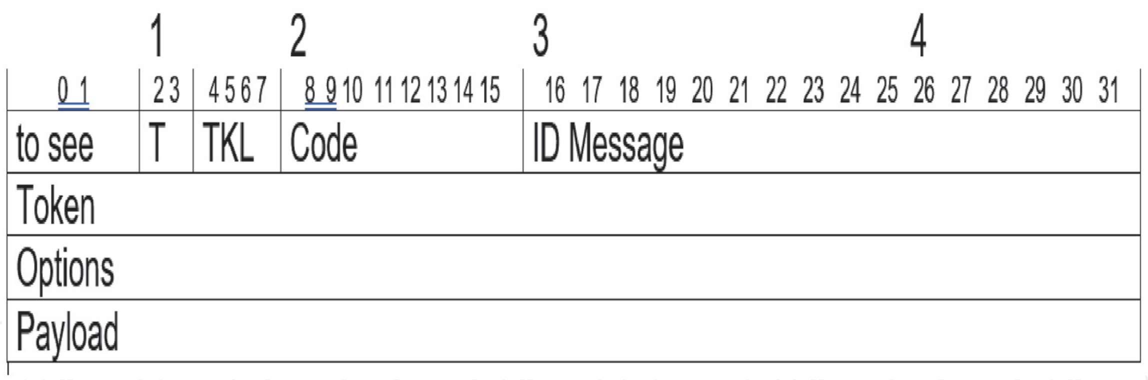
According to [10], CoAP is a transfer protocol aimed at nodes and restricted networks, being designed for M2M applications, such as home automation. CoAP has four types of messages: Confirmable, Non-confirmable, Acknowledgmente and Reset.

- Confirmable (CON): Are those messages that need confirmation at the destination;
- Non-confirmable (NON): They are those messages that do not require acknowledgment of receipt, being very useful for applications that receive constant readings in a short period of time, where the loss of one or the other message does not affect the process;
- Acknowledgment: These are the messages that confirm receipt of a messages, Confirmable;
- Reset: Its function is to indicate that an NOC or NON message was received, but for lack of some context the same could not be properly processed. It can occur in case a device has restarted and the message sent was not properly interrupted.

The COAP uses the request/response model, where devices act as a client or as a server, supporting service discovery and include Web services such as the Uniform Resources Identifiers (URIs) [9].

The following are the main features of the COAP:

- The Coap message exchanges are transported over UDP, and encoding the same are made in binary format with a 4 byte header (**Figure 7**), followed by a variable width token (0 to 8 bytes);
- It has a binary header UDP-based transport, causing thus less delay and reduced battery consumption during transmission;
- Asynchronous message exchange, allowing smart objects to send information only when the application changes;
- HTTP mapping that allows proxies to provide access to CoAP resources.



	1		2	3 4
0 1	2 3	4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
to see	T	TKL	Code	ID Message
Token				
Options				
Payload				

Figure 7.
CoAP message format.

4. Comparison of the protocols

The following will demonstrate a comparison between the protocols presented (MQTT and CoAP), as implementation, data transport, communication standards, reliability and QoS and data security will be analyzed.

4.1 Implementation

Regarding the implementation, the MQTT protocol has a simpler specification in relation to CoAP, thus facilitating customer development. As already mentioned in the 3.2 section, CoAP clients act as HTTP clients but in binary mode, which is simpler than HTTP, but even more complex than MQTT [9].

4.2 Data transport

The MQTT employs a connection oriented communication given by TCP and the CoAP uses UDP. The TCP protocol uses more data to exchange messages between the client and the server in relation to UDP, thus having a higher. Both the MQTT, like CoAP are designed for limited networks like 6LoWPAN (IPv6 over low-power personal area networks). According to [9], if TCP or UDP are not needed, an alternative is to choose the MQTT-SN over 6LoWPAN (IPv6 over low power personal area wireless networks) 4 or even ZigBee, avoiding the complexity of the complete TCP/IP stack. The CoAP It is also designed for limited networks such as 6LoWPAN, in order to maintain short message overload, thus limiting the need for fragmentation that causes significant reduction in the probability of packet delivery.

Regarding the message format, both MQTT and CoAP are suitable to be used in limited bands. Both have a binary message format, different from protocols like AMQP (Advanced Message Queuing Protocol), which uses uses XML formatted messages, in this case requiring the use of more interpreters complex, increasing the hardware requirement.

4.3 Security

One of the main problems to be solved when M2M protocols, is the issue of security [11]. The CoAP protocol is based on DTLS (Datagram Transport Layer Security), so it transfers security handling to the transport layer. Four security modes are allowed:

- NoSec: no DTLS security mechanism is applied;

- PreSharedKey: used with devices that are already pre-programmed with the necessary key switches, where each key has a list of nodes that can communicate;
- RawPublicKey: the device has a pair of asymmetric keys without using a certificate, which is validated by an out-of-band mechanism;
- Certificate: the protocol makes use of the DTLS with an X.509 certificate, the device also has a list of known roots.

According to [12], MQTT security as well as CoAP security (**Table 1**) is performed by Transport Layer Security (TLS). In [13] a safe application model for MQTT is proposed, namely SMQTT. This model is based on a lightweight attribute that provides encryption by broadcast, on elliptical curcas. According to the authors, SMQTT was resistant to attacks from known plaintext, known ciphertex and man-in-the-middle.

Table 2 provides a summary of the security modes of the MQTT and CoAP protocols. In the AAA and Integrity field, it refers to Authorization and

Atack Type	Description
Protocol Parsing and Processing of URIs	It is possible to exploit vulnerabilities in the parsing process (process that analyzes an input sequence), to, for example, generate a denial of service attack by inserting text which will result in parser very extensive.
Proxyinge Caching	The proxy is, in itself, a man-in-the-middle, breaking all the security of IPsec and DTLS. Threats are amplified when proxies allow to cache data.
Amplification Risk	Responses in CoAP are generally larger than requests, which can facilitate amplification attacks
IP Spoofing Attacks	Since there is no handshake for UDP, the final node that has network access can perform spoofing to send messages from ACK instead of CON, preventing from retransmission; spoo pretend the entire payload; spoofing of multicast requests; etc
Cross-Protocol Atacks	They involve using CoAP to send attacks to other protocols, to pass through the firewall, for example
Restriction with Nodes	Whether energetic, memory or processing, make it difficult that devices have good entropy. Therefore, it is assumed, that the processes that need entropy, such as calculating keys, do it externally

Source: [11].

Table 1.
Threats to the CoAP – RFC 7252 protocol.

Protocol	Security Modes	AAA e Integrity	Confidentiality
COAP	No Sec PreSharedKey Raw Public Key Certificate	List of Trusted Roots Uses DTLS	AES-CCM
MQTT	Uses DTLS	Field for name and password uses DTLS	Uses DTLS

Source: [11].

Table 2.
Summary of the security modes of the MQTT and CoAP protocols.

Accountability (Authentication, Authorization and Accountability). In the confidentiality field, the encryptions used by the protocol are described.

4.4 Communication standards

The IoT supports some communication standards that can be defined as:

- In the Telemetry standard, information is sent from devices to the cloud, informing possible changes in states;
- In the query pattern, devices send requests to the cloud to collect information;
- In the Commands pattern, Systems send commands to devices so that they can perform specific activities;
- In the Notification standard, Systems send information to devices in order to inform possible changes in the state of the physical world;

As can be seen in **Figure 8**, the pattern Telemetry becomes suitable for the MQTT protocol, because it has a public/subscribed model, which is equivalent to the telemetry standard. CoAP is not suitable for the Telemetry standard because the connection needs from the system (client) to the server, which faces addressing problems such as mobile roaming or NAT [9]. The CoAP protocol has a better performance for the query communication pattern in relation to the MQTT protocol, since it is based on the request/response model (**Figure 9**). The MQTT has a

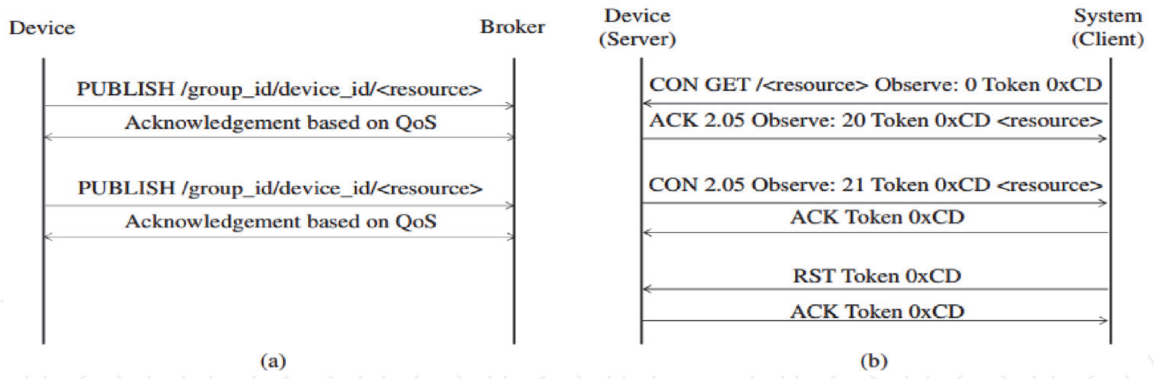

Device
Broker
PUBLISH /group_id/device_id/<resource>
Acknowledgement based on QoS
PUBLISH /group_id/device_id/<resource>
Acknowledgement based on QoS
(a)
Device (Server)
System (Client)
CON GET /<resource> Observe: 0 Token 0xCD
ACK 2.05 Observe: 20 Token 0xCD <resource>
CON 2.05 Observe: 21 Token 0xCD <resource>
ACK Token 0xCD
RST Token 0xCD
ACK Token 0xCD
(b)

Figure 8.
Telemetry communication pattern example for (a) MQTT, (b) CoAP.

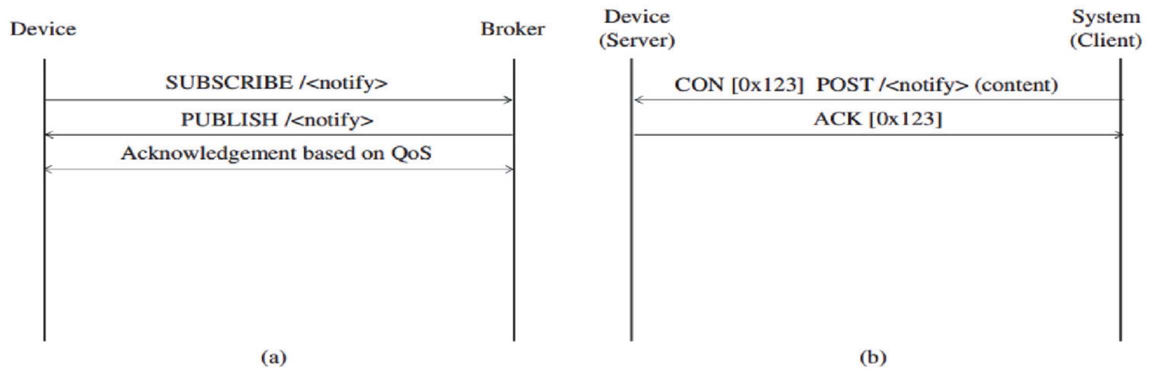


Figure 9.
Example of communication pattern notification for (a) MQTT, (b) CoAP.

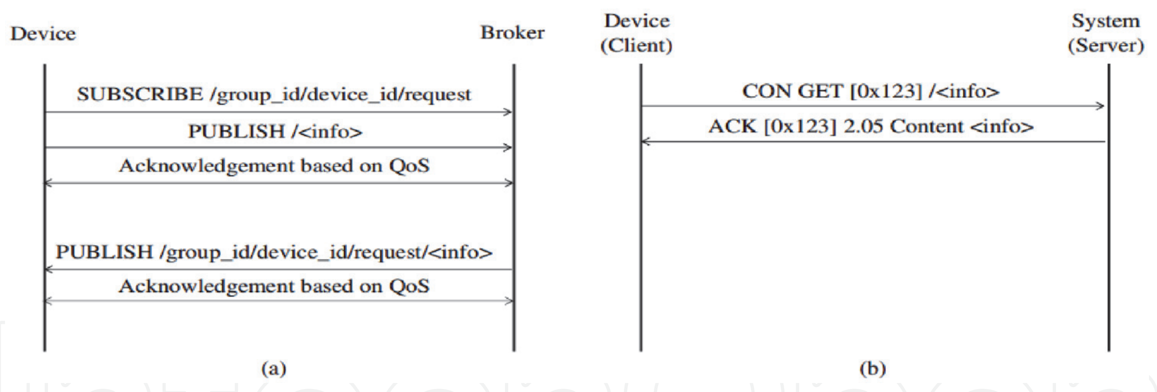

Device
Broker
SUBSCRIBE /group_id/device_id/request
PUBLISH /<info>
Acknowledgement based on QoS
PUBLISH /group_id/device_id/request/<info>
Acknowledgement based on QoS
(a)
Device (Client)
System (Server)
CON GET [0x123] /<info>
ACK [0x123] 2.05 Content <info>
(b)

Figure 10.
Communication pattern example query for (a) MQTT, (b) CoAP.

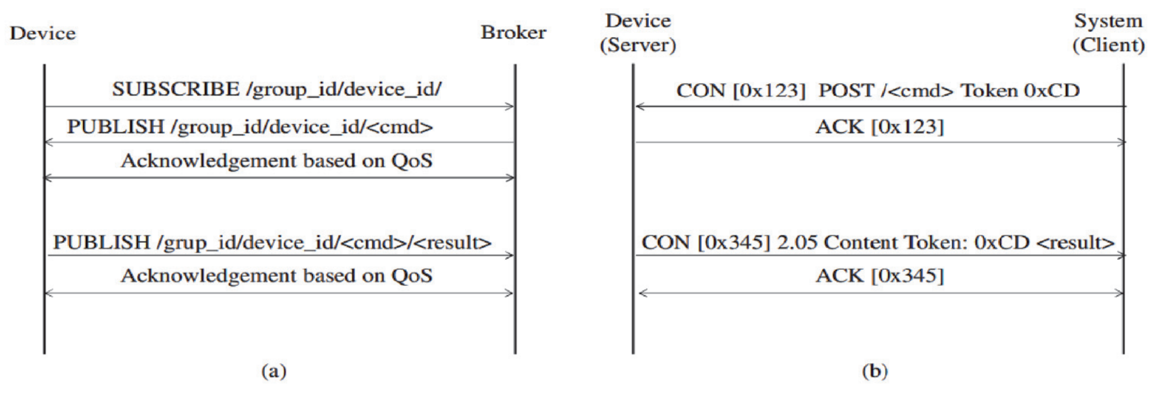

Device
Broker
SUBSCRIBE /group_id/device_id/
PUBLISH /group_id/device_id/<cmd>
Acknowledgement based on QoS
PUBLISH /grup_id/device_id/<cmd>/<result>
Acknowledgement based on QoS
(a)
Device (Server)
System (Client)
CON [0x123] POST /<cmd> Token 0xCD
ACK [0x123]
CON [0x345] 2.05 Content Token: 0xCD <result>
ACK [0x345]
(b)

Figure 11.
Communication pattern example command for (a) MQTT, (b) CoAP.

certain difficulty of implementation in the *Consultation* pattern because it has the need to define a response topic for the communication since there is no way for it to be readily constructed (**Figure 10**).

For the *Command* pattern, both protocols face difficulties. CoAP faces the same addressing problems detailed in textit Telemetry and MQTT does not support native result paths, thus requiring a results topic (**Figure 11**).

Finally, in the *Notification* pattern, the CoAP addressing problems, also listed in the Command and Telemetry patterns, are present. On the other hand, the model MQTT publishes/subscribes to the notification architecture, presenting problems only if better flow control is needed for a large amount of data at high rates [4].

5. Conclusion and related works

Based on the information listed in the previous sections, it can be concluded that both protocols (MQTT and CoAP) are considered for use in restricted environments and on devices with battery, processor and limited memory. However, although the two protocols were designed for application in limited environments, the MQTT exchange protocol has the following advantages over CoAP:

- The transport with small overhead makes MQTT an interesting solution for networks with resource constraints, low bandwidth and high latency;
- The MQTT is more geared for communication “many to many” (using the TCP/IP protocol (**Table 3**)), since the COAP is more geared for

communication "one to one" for information transfer between client and server, using the UDP/IP protocol;

- Because the MQTT protocol has an exchange of messages based on the publish/subscribe model (**Table 3**), decoupling the sender and receiver from the message both in space and time. Thus the sensors that produce the data do not need to know the identity of the clients who are interested in that information. While in CoAP it is the opposite, the protocol requires the identification of both parties;

According to [14], Due to these characteristics, and mainly the characteristic of being a protocol of communication "many to many", the MQTT protocol, has a greater relevance and use in the existing researches that use scenarios where the number of devices communicating is great. The protocol is used in systems that seek to monitor industrial environments, comparative performance analysis with other industrial protocols of the Internet of Things and M2M and in situations of latency estimation in communication.

In [15], the MQTT and CoAP protocols were responsible for connecting sensors and controlling devices on channels with low bandwidth and little robustness. In this case, they were used in conjunction with the Narrowband-IoT standard (NB-IoT), which has the characteristic of allowing mobile phone communication to be used by devices with limited capacities.

In [13], the authors analyze the feasibility of using ciphertext policy attribute-based encryption (CP-ABE), to allow the security of IoT devices, using the ´MQTT protocol and its variants SMQTT and SMQTTSN.

The authors in [16], the authors use the Prognostic Health Management (PHM) system to detect anomalies in industrial systems. It is proposed in this way the integration of the PHM system to the industrial environment of IoT, based on MQTT and Cloud Computing in order to allow the assessment of the state of the equipment in real time, thus improving the performance of the PHM.

In [17] a cloud-based architecture based on machine learning, for condition monitoring, fault detection and process optimization in industrial environments. The implemented system uses the Dempster-Shafer Evidence Theory (DSET) (**Figure 12**).

In **Figure 12**, the main components proposed can be seen. In this work, the OPC-UA/MQTT gateway is used to communicate between OPC servers on the automation platform and the CORESYS CLOUD broker.

	MQTT	**MQTT-SN**	**CoAP**
Network Protocol	TCP/IP	Not specified	UDP
Useful data type	Binary	Binary	Binary
Suitable for microcontrollers	Yes	Yes	Yes
Security	SSL/TLS	Not specified	DTLS
Scalability	Simply	Simply	Complex
Network architecture	Broker-based (publishes/subscribes)	Broker-based, Client/Server, Client/Server	Client/Server (request/response)
Network architecture	Broker-based (publishes/subscribes)	Topic-based	REST architecture
QoS Options	Yes	Yes	Yes

Table 3.
Comparison table between MQTT and CoAP protocols.

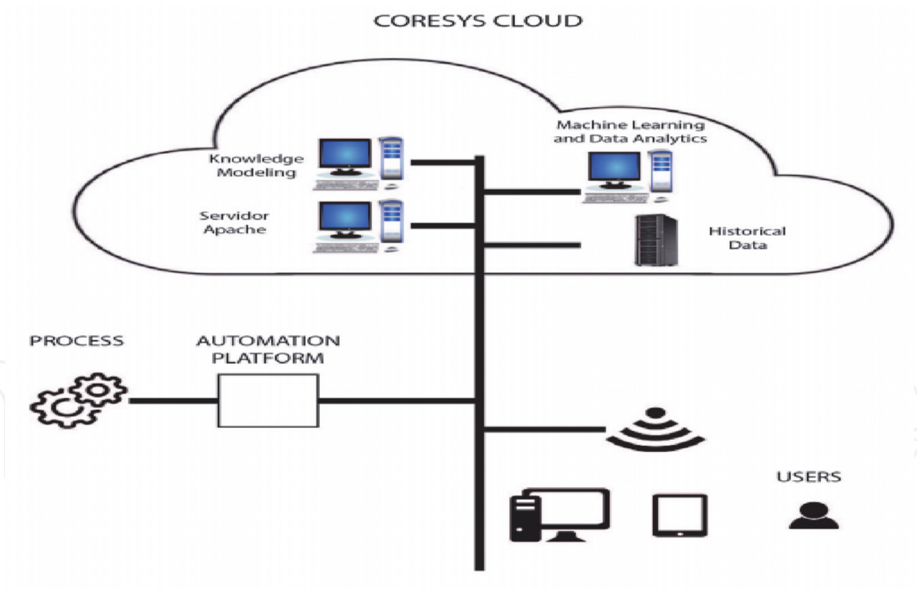


Figure 12.
Framework proposed CORESYS CLOUD. Source: [17].

According to [18], CoreSys-Cloud four services were implemented, namely:

- Machine Learning e Data Analythics (MLDA): Module responsible for learning and analyzing data for state assessment and monitoring;
- Flask: Software used for web application development;
- MySQL database server: responsible for storing the results MLDA evaluation;
- Server MQTT: the MQTT server is responsible for establishing communication between the OPC-UA/MQTT gateway and CoreSys-Cloud.

Cyber security is treated as one of the technological pillars of industry 4.0, which in turn is associated with the protection of software, machines, equipment, network infrastructures and systems. Thinking about it [19] proposed a new approach to protect communication between networked robotic systems, providing authentication and data encryption.The Robot Operating System (ROS) is one of the best robotic software development platforms, offering low-level device control, diverse resources and many useful tools for simulating, visualizing and debugging data, making it very popular with researchers from various robotics fields. Communication in ROS is based on a publish-subscribe system, using the Remote Procedure Call Protocol (RCP) and Extensible Markup Language (XML), with the data sent in clear text over TCP/IP or UDP/IP, without any security mechanism. Based on these aspects, the authors proposed an integration between ROS and the MQTT protocol, using its security features (**Figure 13**).

The authors performed a performance analysis comparing a system without using the security systems offered by MQTT and another system using the MQTT cryptography resources. The results show that the encrypted solution adds negligible delays during communication between clients and servers.

Studies involving the implementation of ROS in industrial environments are gaining more and more evidence. The functioning of ROS is similar to the MQTT protocol in that it works about a publish/subscribe architecture, and use versions ROS of TCP and UDP protocols. Both versions called TCPROS and UDPROS. Due to the use of this architecture, ROS is composed of two elements. The master, which

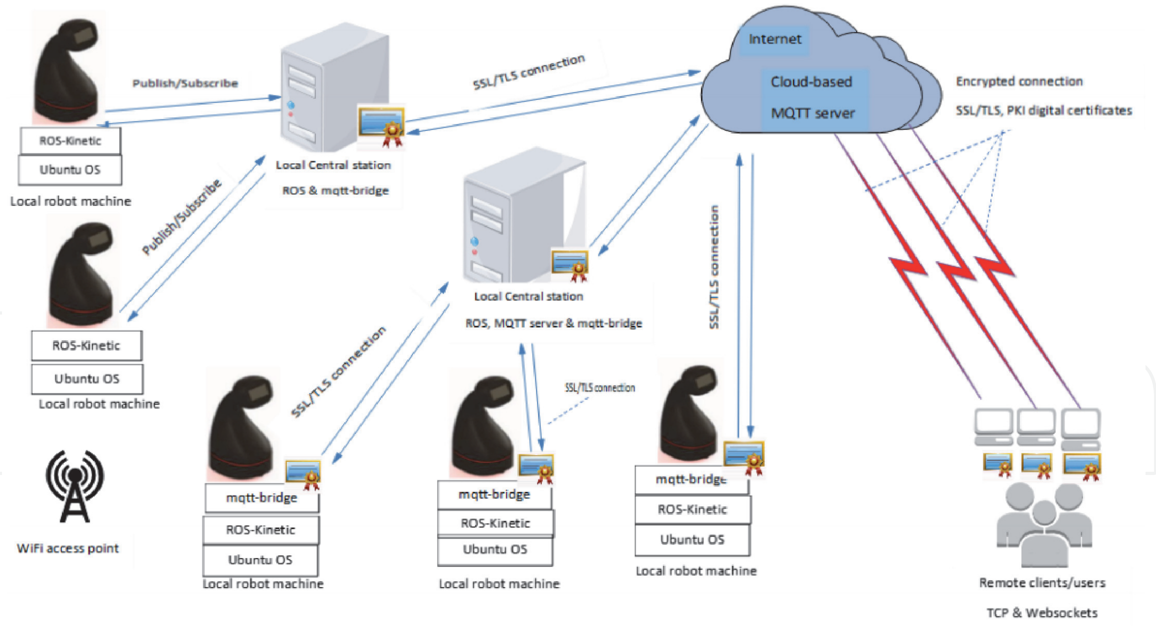


Figure 13.
Proposed architecture. Source: [17].

acts as the MQTT broker, and the nodes, which act as the clients. As o MQTT, ROS controls the data ow through the topics that are coordinated by the master [20]. The main difference between ROS and MQTT, in terms of communication, is the fact that broker has topics with defined typing, that is, a topic will be created with a single type and will receive subscribers and publishers of this type only.

The Authors in [20], presented an implementation of MQTT integrated to ROS, showing the feasibility of using this protocol in the 4.0 industry scenario. [21] cited in [20], evidence the use of ROS in comparison with a traditional solution (**Figure 14**).

In [22] the authors made a comparison between the AMQP (Advanced Message Queuing Protocol) and MQTT protocols, in the context of a smart factory environment, with ROS also being used in some applications that required a more complex and heterogeneous environment.

In [23], The authors proposed os the implementation of internet technology for monitor and control industrial amr robot in industry. Was made a web-based interface for monitor motion and controlling the angle of joint arm robot in a ROS industrial simulation environment, using the mqtt protocol to communicate between the robot and the client, give low latency data transmission.

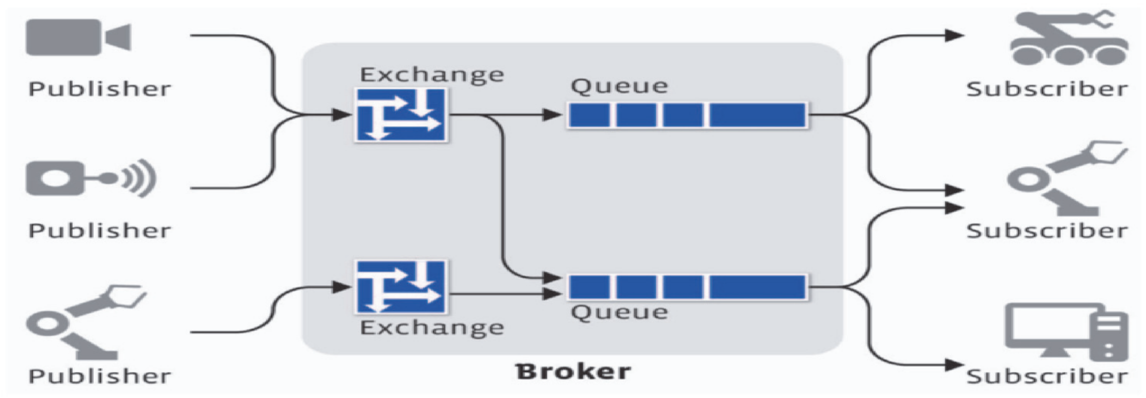


Figure 14.
AMQP architecture.

The MQTT emerges as an excellent alternative for communication between multi robot systems in several other works, as in [16] the the authors conducted a study on the use of MQTT COAP and protocols in Ubiquitous Network Robot Platform (UNR- PF) for the communication of a multi-robot system. In this work, the authors were able to verify that the MQTT protocol is easier to be implemented in the multi robot platform (UNR-PF) than CoAP, in addition to having a higher data transfer rate. Other works related to the use of the MQTT protocol in multi-robot systems are: [14, 24, 25].

Abbreviations

M2M	Machine-to-Machine
IoT	Internet of Things
IP	Internet Protocol
PLC	Programmable Logic Controller
HTTP	Hypertext Transfer Protocol
FTP	File Transfer Protocol
MQTT	Messaging Queue Telemetry Transport
CoAP	Constrained Application Protocol
PDU	Protocol Data Unit
MQTT-SN	MQTT Sensor Network
AMQP	Advanced Message Queuing Protocol
DTLS	Datagram Transport Layer Security
TLS	Transport Layer Security
CP-ABE	Ciphertext Policy Attribute-based Encryption
PHM	Prognostic Health Management
MLDA	Machine Learning e Data Analythics
ROS	Robot Operating System
RCP	Remote Procedure Call Protocol
XML	Extensible Markup Language
UNR-PF	Ubiquitous Network Robot Platform
AMQP	Advanced Message Queuing Protocol

Author details

Edi Moreira M. de Araujo[1*†], Augusto Loureiro da Costa[1†]
and Alejandro R.G. Ramirez[2]

1 Universidade Federal da Bahia, Salvador, Brazil

2 Universidade do Vale do Itajaí, Itajaí, Brazil

*Address all correspondence to: edimmaraujo13@gmail.com

† These authors contributed equally.

References

[1] SCHWAB, Klaus. The fourth industrial revolution. Currency, 2017:1–5.

[2] KAGERMANN, Henning et al. Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Securing the future of German manufacturing industry; final report of the Industrie 4.0 Working Group. Forschungsunion, 2013. https://www.din.de/blob/76902/e8cac883f42bf28536e7e8165993f1fd/recommendations-for-implementing-industry-4-0-data.pdf [Accessed: 01 January 2021]

[3] de Sousa Cardoso, Fábio. UMA ARQUITETURA MULTI-AGENTE PARA COORDENAÇÃO ROBÓTICA. thesis. Rio de Janeiro, 2015. https://w1files.solucaoatrio.net.br/atrio/ufrj-pem_upl/THESIS/1839/pemufrl2015dscfbiodesousacardoso_20170112113019814.pdf [Accessed: 02 January 2020]

[4] MARTINS, Ismael Rodrigues; ZEM, José Luís. Estudo dos protocolos de comunicação MQTT e COaP para aplicações machine-to-machine e Internet das coisas. Revista Tecnológica da Fatec Americana, v. 3, n. 1, p. 24p.-24p., 2015.https://fatecbr.websiteseguro.com/revista/index.php/RTecFatecAM/article/view/41 [Accessed: 3 November 2020]

[5] PTIČEK, Marina et al. Architecture and functionality in M2M standards. In: 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2015. p. 413-418.. https://www.researchgate.net/publication/308828235_Architecture_and_functionality_in_M2M_standards [Accessed: 6 November 2020]

[6] CAO, Yang; JIANG, Tao; HAN, Zhu. A survey of emerging M2M systems: Context, task, and objective. IEEE Internet of Things Journal, v. 3, n. 6, p. 1246-1258, 2016. https://ieeexplore.ieee.org/abstract/document/7494995/ [Accessed: 10 November 2020]

[7] https://www.emqx.io/blog/mqtt-5-introduction-to-publish-subscribe-model [Accessed: 25 November 2020]

[8] CARISSIMI, Alexandre. Internet das Coisas, middlewares e outras coisas. https://www.researchgate.net/profile/Alexandre_Carissimi/publication/301298394_Internnet_das_Coisas_Middlewares_e_outras_coisas/links/5710f73308aeebe07c023a6e/Internnet-das-Coisas-Middlewares-e-outras-coisas.pdf [Accessed: 05 October 2020]

[9] MAZZER, Daniel; FRIGIERI, E.; PARREIRA, L. Protocolos M2M para Ambientes Limitados no Contexto do IoT: Uma Comparaçao de Abordagens. Inatel. Br, 2015. https://www.inatel.br/smartcampus/imgs/protocolos-para-iot-pt.pdf [Accessed: 10 October 2020]

[10] JIN, Wen-Quan; KIM, Do-Hyeun. Implementation and Experiment of CoAP Protocol Based on IoT for Verification of Interoperability. The journal of the institute of internet, broadcasting and communication, v. 14, n. 4, p. 7-12, 2014. https://www.koreascience.or.kr/article/JAKO201426636276370.page [Accessed: 15 October 2020]

[11] PANDEY, Abhishek; TRIPATHI, R. C. A survey on wireless sensor networks security. International Journal of Computer Applications, v. 3, n. 2, p. 43-49, 2010. https://arxiv.org/abs/1011.1529 [Accessed: 22 October 2020]

[12] DO ESPÍRITO SANTO, Walter; ORDOÑEZ, Edward; RIBEIRO, Admilson. Uma revisão sistemática sobre a Segurança nos Protocolos de Comunicação para Internet das Coisas. Journal on Advances in Theoretical and Applied Informatics, v. 4, n. 1, p. 1-9, 2018. https://revista.univem.edu.br/jad

i/article/view/2482 [Accessed: 28 October 2020]

[13] SINGH, Meena et al. Secure mqtt for internet of things (iot). In: 2015 fifth international conference on communication systems and network technologies. IEEE, 2015. p. 746-751. https://ieeexplore.ieee.org/abstract/document/7280018 [Accessed: 28 October 2020]

[14] BELLAVISTA, Paolo; ZANNI, Alessandro. Towards better scalability for IoT-cloud interactions via combined exploitation of MQTT and CoAP. In: 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI). IEEE, 2016. p. 1-6. https://ieeexplore.ieee.org/abstract/document/7740614 [Accessed: 13 November 2020]

[15] KLYMASH, Mykhailo et al. Designing the Industrial and Environmental Monitoring System based on the Internet of Things Architecture. In: 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS). IEEE, 2018. p. 187-190. https://ieeexplore.ieee.org/abstract/document/8525545 [Accessed: 13 November 2020]

[16] AYAD, Soheyb; TERRISSA, Labib Sadek; ZERHOUNI, Noureddine. An IoT approach for a smart maintenance. In: 2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET). IEEE, 2018. p. 210-214. https://ieeexplore.ieee.org/abstract/document/8379861 [Accessed: 23 November 2020]

[17] ARÉVALO, Fernando; DIPRASETYA, Mochammad Rizky; SCHWUNG, Andreas. A cloud-based architecture for condition monitoring based on machine learning. In: 2018 IEEE 16th International Conference on Industrial Informatics (INDIN). IEEE, 2018. p. 163-168. https://ieeexplore.ieee.org/abstract/document/8471970 [Accessed: 25 November 2020]

[18] SOUZA, Daniel Silva de et al. Estudo da aplicação de um sistema IoT baseado no protocolo de comunicação MQTT a área da robótica industrial. 2018. http://repositorio.ufu.br/handle/123456789/24847 [Accessed: 20 December 2020]

[19] MUKHANDI, Munkenyi et al. A novel solution for securing robot communications based on the MQTT protocol and ROS. In: 2019 IEEE/SICE International Symposium on System Integration (SII). IEEE, 2019. p. 608-613. https://ieeexplore.ieee.org/abstract/document/8700390 [Accessed: 20 December 2020]

[20] Daniel Zolett; Alejandro R.G. Ramirez. Desenvolvimento de uma Interface de Monitoração Remota para o Sistema Robótico ROBIX, Integrando o Protocolo MQTT e o ROS. Computer on the Beach. September 2020. DOI: 10.14210/cotb.v11n1.p405-412

[21] Carol Martinez, Nicolas Barrero, Wilson Hernandez, Cesar Montaño, and Iván Mondragón. Setup of the yaskawa sda10f robot for industrial applications, using ros-industrial. In Advances in Automation and Robotics Research in Latin America, pages 186–203. Springer, 2017.

[22] Bezerra, D., Aschoff, R. R., Szabo, G., e Sadok, D. (2018, November). An IoT protocol evaluation in a smart factory environment. In 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE) (pp. 118-123). IEEE.

[23] Atmoko, R. A., e Yang, D. (2018, August). Online monitoring e controlling industrial arm robot using mqtt protocol. In 2018 IEEE

International Conference on Robotics, Biomimetics, and Intelligent Computational Systems (Robionetics) (pp. 12-16). IEEE.

[24] BOTTONE, Michele et al. Implementing virtual pheromones in bdi robots using mqtt and jason (short paper). In: 2016 5th IEEE International Conference on Cloud Networking (Cloudnet). IEEE, 2016. p. 196-199. https://ieeexplore.ieee.org/abstract/document/7776601 [Accessed: 20 December 2020]

[25] BHAVSAR, Pritesh; PATEL, Sarosh H.; SOBH, Tarek M. Hybrid Robot-as-a-Service (RaaS) Platform (Using MQTT and CoAP). In: 2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). IEEE, 2019. p. 974-979. https://ieeexplore.ieee.org/abstract/document/8875263 [Accessed: 20 December 2020]