Development of Industrial Agents in a Smart Parking System for Bicycles

Lucas Hiroaki Sakurada

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

Work oriented by:

Professor PhD Paulo Leitão

Professor PhD José Barbosa

Professor PhD Roberto Neli

Bragança

2018-2019

Development of Industrial Agents in a Smart Parking System for Bicycles

Lucas Hiroaki Sakurada

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

Work oriented by:

Professor PhD Paulo Leitão

Professor PhD José Barbosa

Professor PhD Roberto Neli

Bragança

2018-2019

Dedication

I dedicate this work to my parents who helped me get where I am today.

Acknowledgement

Here I would like to thank everyone who contributed to the development of this work.

Firstly, to my supervisors at the Polytechnic Institute of Bragança (IPB), Professor PhD Paulo Leitão and Professor PhD José Barbosa, for guidance, knowledge transmitted, opportunity, motivation and advice. I never imagined that I would have such experience. My eternal thanks.

To my supervisor at the Federal University of Paraná (UTFPR), Professor PhD Roberto Neli, for the opportunity to participate in the Dual Diplomacy program.

To my friends and laboratory colleagues, for all the moments spent together, sharing experiences and knowledge.

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

Abstract

Smart parking systems are a suitable solution to avoid some recurring problems in large cities, such as traffic congestion and air pollution. These systems aim to guide in real time the driver to find free parking spots, in the shortest time possible and as close as possible to the desired location. In this sense, such smart parking constitutes a Cyber-Physical System (CPS) and requires the use of appropriate Internet of Things (IoT) technologies and Artificial Intelligence (AI) techniques, such as Multi-Agent System (MAS), that allow the incorporation of intelligence into the CPS through autonomous, proactive and cooperative entities. This work presents the development of an agent-based CPS for parking of bicycles, where the interface between the software agents and the low-level control devices plays a fundamental role in the global functioning of the system. For this purpose, different interface practices were implemented and tested to determine which best fits for the current application. Based on the results obtained, an agent-based CPS was implemented, showing an efficient, modular, adaptable and scalable operation.

Keywords: Cyber-Physical System; Interface Practices; Internet of Things; Multi-Agent System; Smart parking system.

Resumo

Os sistemas inteligentes de estacionamento são uma solução adequada para evitar alguns problemas recorrentes nas grandes cidades, como o congestionamento do tráfego e poluição do ar. Esses sistemas visam orientar em tempo real o motorista a encontrar vagas livres de estacionamento, no menor tempo possível e o mais próximo do local desejado. Nesse sentido, um sistema inteligente de estacionamento constitui um sistema ciber-físico e requer a utilização de tecnologias relacionadas a internet das coisas e técnicas de inteligência artificial, como os sistemas multi-agentes, que permitem a incorporação de inteligência através de entidades autônomas, proativas e cooperativas. O presente trabalho apresenta o desenvolvimento de um sistema ciber-físico baseado em agentes para um parque de bicicletas, onde a interface entre agentes de software com dispositivos de controle de baixo nível desempenham um papel fundamental para o funcionamento global do sistema. Para esse propósito, diferentes interfaces práticas foram implementadas e testadas para determinar qual é a mais adequada para a atual aplicação. Baseado nos resultados obtidos, foi implementado um sistema ciber-físico baseado em agentes, mostrando uma operação eficiente, modular, adaptável e escalável.

Palavras-chave: Sistema ciber-físico; Interfaces práticas; Internet das coisas; Sistema multi-agente; Sistema inteligente de estacionamento.

Contents

List of Tables

List of Figures

Acronyms

ACL Agent Communication Language. 11–13

ADU Application Data Unit. 15

AI Artificial Intelligence. ix, 5

CNP Contract Net Protocol. 12

CPS Cyber-Physical System. ix, 2–5, 9, 12, 19, 20, 22–24, 39, 41, 46, 49, 50

FIPA Foundation for Intelligent Physical Agents. 11, 12

GPS Global Positioning System. 2, 7

ICT Information and Communications Technology. 2, 5, 49

IoT Internet of Things. ix, 2, 3, 5, 7, 12, 13, 19, 49, 50

IPB Polytechnic Institute of Bragança. vii

IT Information Technology. 8

JADE JAVA Agent Development Framework. 27, 29, 43

MAS Multi-Agent System. ix, 2, 3, 5, 9, 10, 12, 19, 20, 27, 46, 49

MBAP Modbus Application Protocol. 15

MILP Mixed-Integer Linear Programming. 7

MQTT Message Queuing Telemetry Transport. 12, 13, 25, 28, 37, 43

OPC UA Open Platform Communication Unified Architecture. 12, 16, 17, 25, 28, 31, 32, 35, 37

PDU Protocol Data Unit. 15

PLC Programmable Logic Controller. 25, 27, 28, 31, 32, 35

QoS Quality of Service. 13–15

RFID Radio Frequency Identification. 2, 7

RTT Round Trip Time. xix, 30–33, 35–37, 47

TCP Transmission Control Protocol. 15

UTFPR Federal University of Paraná. vii

WSN Wireless Sensor Network. 7

Chapter 1

Introduction

This work is framed in the context of Industry 4.0, focusing the new concept of "Cyber-Physical Systems". These systems are revolutionizing the interaction between the physical world and computational processes, proposing significant improvements in various fields of application, such as traffic control, energy conservation, communication systems, robotics, medicine, agriculture and security.

1.1 Motivation and Framework

In recent years, the intense urbanization process and consequently the increase of the circulation of vehicles are factors that prevent the sustainable development in the large cities. It is estimated in generally that 30% of the traffic congestion is caused by the demand for parking spots, which in addition to causing frustration to the driver who consumes a significant amount of time, on average 7.8 min until finding a free spot, also contributes to air pollution [1].

These problems are not only limited to drivers and the mobility sector, affecting all sectors of a city, such as economy, health, governance, people and environment. In this sense, smart parking systems are a suitable solution to solve these problems, avoiding the traffic congestion through efficient systems that can guide the driver to find a free spot to park its vehicle, in the shortest time possible and as close as possible to the desired

location [1], [2]. Such smart parkings constitute a CPS and require the integration of some emergent technologies such as Information and Communications Technology (ICT), IoT and MAS.

In the literature, there are several approaches in development and already implemented, using different strategies, namely reservation and search systems, Global Positioning System (GPS), Radio Frequency Identification (RFID), IoT and MAS technologies [1], [3], [4]. All approaches present optimistic results, showing the potential of a smart parking systems for smart cities, in particular MAS, which allows the development of a decentralized architecture and incorporation of intelligence into the system, through autonomous, cooperative and proactive entities.

Having this in mind, this work addresses the development of an agent-based CPS for parking of bicycles using publish-subscribe model, where the interface between the software agents and the low-level control devices plays a fundamental role in the global functioning of the system. For this purpose, different interface practices need to be implemented and tested to determine which best fits for the current application.

Taking this into consideration, the develop of an agent-based CPS for parking of bicycles aims to facilitate the daily use of bicycle drivers, stimulate the practice of physical activity and promote the sustainable development by encouraging the use of bicycles rather than vehicles responsible for carbon dioxide emissions. In addition, the bicycle parking system may be an alternative solution for the traffic congestion, since the space required for parking a vehicle (e.g., car or truck) may be intended for parking several bicycles, as shown in Figure 1.1.

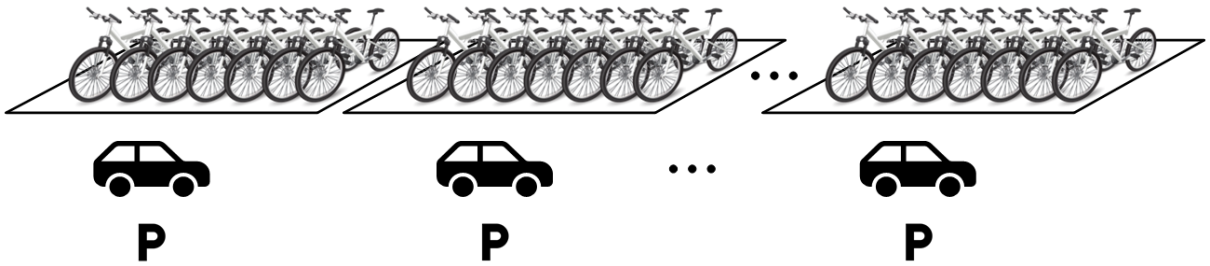

Figure 1.1: Parking of bicycles.

1.2 Objectives

The main objective of this work is to study and develop the interconnection between the software agents with the low-level control devices, constituting an agent-based CPS for parking of bicycles that allows the access to the parking spots. In order to achieve this objective, secondary objectives have been established:

- Deployment of software agents in a low cost computational platform.
- Implement and test different ways to interconnect software agents with low-level control devices in terms of location, interaction mode, computational platform and communication protocols.
- Analyze the implemented interface practices based on parameters such as response time, technology, scalability and re-usability. Through these results, identify which interface practice is a suitable solution for the smart parking system.
- Contribute to define an architecture for the smart parking system based on the MAS technology and allied with the holonic principles.
- Define a strategy to control the access to the parking spots through reservation mechanisms and IoT technology.

1.3 Document Structure

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

The Chapter 2, entitled "Related work", presents a review of the related work, addressing the fundamental contents for the development and understanding of the work.

The Chapter 3, entitled "Smart Parking Architecture", describes the system architecture used in this work, demonstrating the global functioning of the system, the use of negotiation processes, the inclusion of holonic principles and highlights the importance of

the interconnection between the software agents and the low-level control device to allow the access to each parking spot.

The Chapter 4, entitled "Interface Practices", presents an overview of recommended practices for interfacing the software agents and the low-level control devices, performs the deployment of different interface practices, the methodology to test each interface and, discusses and shows the results obtained.

The Chapter 5, entitled "Cyber-Physical Interconnection", uses the results obtained in Chapter 4 to develop an agent-based CPS for a bicycle park. In this way, it proposes a structure for the agent-based CPS, describes how the reservation mechanism and communication between the cyber and physical part works. Finally, it presents how this structure was implemented for a parking system of bicycles and discusses the achieved results.

Finally, the last chapter, entitled "Conclusions and Future Work", rounds up the paper with the conclusions and points out the future work.

The document also contains 2 appendixes, one comprising the results of the experimental tests regarding the implemented interface practices, and the second including the electrical diagrams to implement of latch system of the parking spot.

Chapter 2

Related Work

This chapter aims to guide the reader through a literature review in the field of smart parking systems, and particularly in the interconnection between cyber and physical counterparts of the such CPS. Initially will be explained the smart cities and smart parkings concepts. Then, the fundamental technologies for the development of this work, such as CPS and MAS will be introduced. Finally, it will be provided a brief explanation of some communication protocols.

2.1 Smart City

The high population density in urban centers, the structure not suitable for the citizens and the environmental pollution, are issues constantly discussed in recent years. In this context, the concept of "smart city" emerges as a solution to these problems, a term that has been gaining strength due to the technologies available today, such as ICT, IoT and AI, which has enabled the implementation of these smart cities [5].

Although it is a recurring term, there is not a single definition for a smart city, being possible to find in the literature different points of views. However, the goal of a smart city is common for all of these works, and mainly related to the use of the technology as a tool to ensure the well-being of the population [6], [7]. Another important point that the smart cities care about is the environmental changes. For a long time, the human

activity has used the natural resources without worrying about the consequences. Today, there are several problems in the world, such as global warming, air, ground and water pollution, food shortages and drastic reduction of biodiversity [8].

In this way, for the smart cities to be able to promote a sustainable development in social, economic and environmental terms, ensuring the well-being of the population and preserving the planet resources, it is necessary to adopt planning and management strategies, as well as to use ICT, IoT and AI technologies in all sectors that constitute a city [8], [9]. Figure 2.1 illustrates the model of a smart city proposed by the European smart cities project [10].

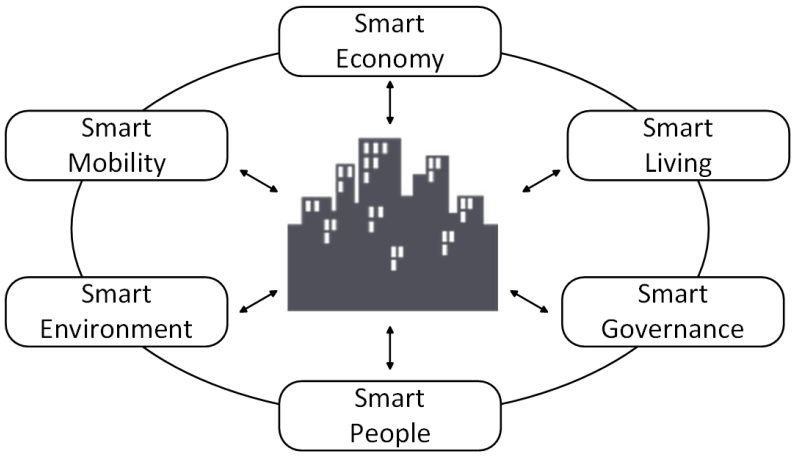

Figure 2.1: Smart city model (adapted from [10]).

This model establishes several sectors to define a smart city, namely smart economy, living, governance, people, environment and mobility. In this work, the focus is on the smart mobility sector, more specifically on the intelligent management of a parking system.

2.2 Smart Parking System

The increase of the circulation of vehicles and limited parking in large cities, are factors that the intelligent parking systems focus on solving. The main problem generated by the intense flow of vehicles and the shortage of parking spots is the traffic congestion, which generates as a consequence the frustration to the driver, who consumes a lot of

time searching for a free parking spot and causes damages to the environment through the air pollution [2], [11].

Currently the most common way of looking for a parking spot is manual, through several attempts and consuming a lot of time. Still, this process does not guarantee the meeting of the parking spot and when it finds, it is far from the desired location. In this sense, the smart parking system is a suitable solution to solve these problems [3], [12].

A smart parking is an intelligent parking system, basically performs the management between the physical parking and the driver. Its purpose is to ensure that each driver can find a free parking spot to park their vehicle, in the shortest time possible and as close to the desired location. The search for a free parking spot can be initiated by the driver, or by the smart parking, offering access to the spots that are available, promotions to the drivers and using different negotiation strategies [2].

In the literature, there are several approaches to implement a smart parking system, basically using reservation and search systems, IoT technologies, sensors and GPS.

In [3] a smart parking system is proposed based on parking search algorithms and IoT technologies. This algorithm identifies a parking spot at the lowest cost, considering the distance and number of free parking spots, and also offers a new parking spot if the current parking lot is full. To monitor, the system uses the Wireless Sensor Network (WSN), which consists of RFID technology to check which parking spots are free. Based on the results, this approach increases the chances and decreases the time to find a free parking spot.

In [1] a smart parking system is proposed based on mathematical modeling using Mixed-Integer Linear Programming (MILP) with the objective of minimizing the monetary cost for drivers and maximizing the utilization of parking spots. This system uses two parameters, proximity to the desired location and parking cost, to allocate reservations to the drivers. The simulation results compared to uncontrolled parking, allowed to verify that this approach reduces the average time to find a free parking spot.

In [4] a smart parking system is proposed based on GPS, which uses the location of the driver to find free parking spots. In this case, the system compares the location

of the driver and the location of parking spots, and informs the driver through a user application, the nearest spots. Comparing this approach with a traditional parking lot, the results show a reduction in search time.

2.3 Cyber-Physical Systems and Industry 4.0

Today we live the 4th Industrial Revolution, also called Industry 4.0, which is introducing several transformations in industry and other sector, particularly driven by the digital transformation. The past industrial revolutions were [13] (see Figure 2.2):

- 1st Industrial Revolution: marked by the use of the steam engine in the production process.
- 2nd Industrial Revolution: characterized by the mass production.
- 3rd Industrial Revolution: incorporation of electronics and Information Technology (IT) in production lines.

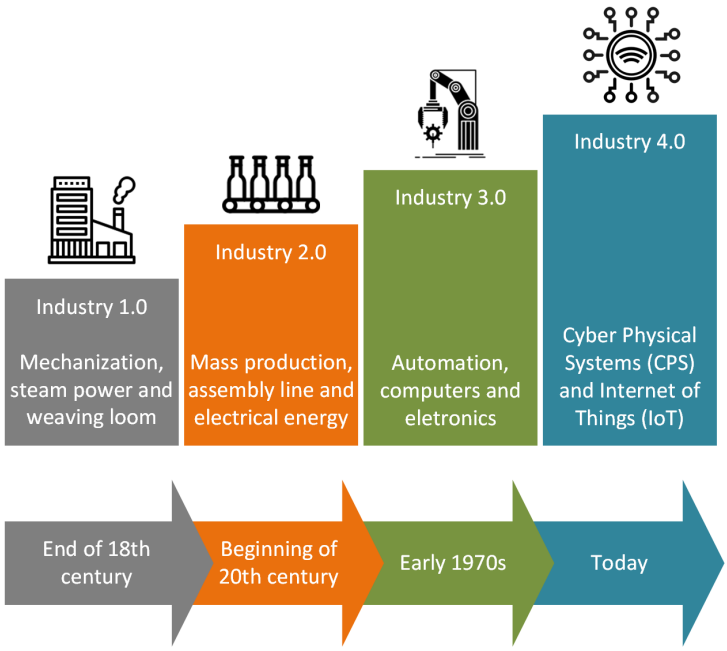

Figure 2.2: Chronology of the industrial revolution (adapted from [14]).

As seen in the previous phases, where they were marked by a paradigm shift, Industry 4.0 also innovates in the way the production process is performed. With the introduction of a new concept, the CPS, the 4th Industrial Revolution is moving towards building intelligent factories [13], [15]. CPS are systems that integrate computational resources with physical processes [16], [17], constituting a network of cyber and physical entities that are integrated to achieve a particular objective. Just like the Internet that has modified the way that the people interact with each other, the CPS are revolutionizing the interaction between the physical world and the society. This revolution extends to various fields of application, such as manufacturing, traffic control, advanced automotive systems, energy conservation, communication systems, robotics, medicine and agriculture.

The operating principle of a CPS is based on three elements: communication, control and computation. Based on these elements, the CPS is able to perform actions such as: detecting external events through sensors, interacting and acting on the environment through actuators, collecting and evaluating data, making decisions based on collected data and interacting with other CPS [18].

In fact, there are no limits to applications based on CPS, the difficulty for its implementation is the unpredictability of the physical world, i.e., the control of an environment with unpredictable conditions.

2.4 Multi-Agent System

MAS becomes a suitable solution to assist in the performance of the CPS, incorporating intelligence to the system. According to [19], the agent is an autonomous software component and endowed with an interoperable interface. This component may interact with other agents cooperatively, i.e., to benefit all parties, or may compete for self-interest. What makes the agent a singular and intelligent entity is its characteristics [19], [20]:

- Autonomy: the agent controls its behavior, i.e., its able to make decisions without human intervention.

- Responsiveness: the agent can be equipped with sensors and actuators, so it can perceive the physical world through sensors and respond by means of actuators.

- Proactivity: the agent is able to take the initiative to reach its objectives or to anticipate possible changes in the environment.

- Social skills: the agent can interact with humans or other agents to achieve their goals by sharing or requesting information.

- Learning ability: to be autonomous and flexible, the agent must have the ability to learn, adapting to the environment and the needs of the user.

The problems of the physical world require solutions that are at the same level of complexity, so the use of a single agent is not enough. In these cases, the effort of more agents is necessary, constituting a MAS. Due to the advantages offered, the MAS has been used in several applications, the characteristics that ensure a wide level of applicability are: scalability, flexibility, robustness, responsiveness, decentralized architecture and reusability [20], [21].

Although there are various advantages, there are some consequences when working with several agents, which makes their implementation a challenge, such as the environment, since the action taken by an agent can modify the environment shared by the agents. The perception, as the agents are distributed over an extensive environment, it is impossible for a single agent to be able to have the knowledge of every environment, being necessary the exchange of information among the agents. Conflict resolution, the action taken by one agent can harm the others, so it is necessary to establish a high level of cooperation between the agents, avoiding the conflict in the decision making [21]. Figure 2.3 illustrates the structure of MAS.

The operating principle of MAS is based on the communication between the agents, from this it is possible the exchange of information between the agents of the system. For this purpose, just as humans have their own language of communication, the agents

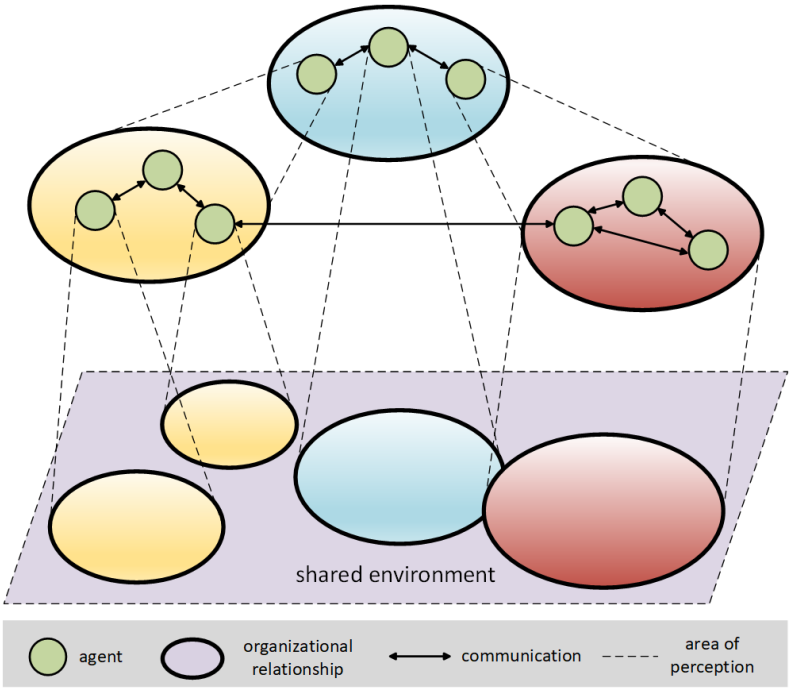

Figure 2.3: Structure of MAS [22].

are also endowed with one. Currently the most commonly used language for agent-to-agent communication is the Agent Communication Language (ACL) established by the Foundation for Intelligent Physical Agents (FIPA) [19].

Session 2.2 presented some approaches for the implementation of smart parking systems commonly found in the literature. The following will be presented some approaches based on MAS, since the present work uses particularly this technology to implement an agent-based CPS for parking of bicycles.

In [23] a smart parking system is proposed to allocate parking spots through the interaction between intelligent agents, the manager and drivers. The manager manages the parking spots and the drivers use them. The allocation process takes place through a confidence level mechanism, a numerical value that increases according to the use of parking spots without violating rules, or decreases if the opposite happens.

In [24] a smart parking system is proposed with a decentralized architecture based on intelligent agents, constituting a MAS. In this system, the interaction is performed through several agents of the same type, the driver, that can act as a "buyer" or "seller",

the first option when looking for a parking spot, and the second when leaving of the parking spot. For this purpose, the authors define a model of reasoning and negotiation protocol.

In [25] a smart parking system is proposed based on different negotiation strategies between agents, namely Contract Net Protocol (CNP), English and Dutch auction. In this system there are two types of agents, manager and driver. The manager agent is responsible for negotiating and offering parking spots and the driver agents represent the drivers that wants to park their vehicles.

Although the communication between the agents is the basis for the operation of MAS, often an agent needs to communicate with, for example, some low-level control devices. In this case, the agent must be implemented with communication protocols that allow this integration, such as Modbus, Open Platform Communication Unified Architecture (OPC UA) and Message Queuing Telemetry Transport (MQTT). The next session will explain each of the cited communication protocols.

2.5 Communication Protocols

This chapter aims to address some communication protocols usually used in the interconnection of cyber and physical parts in a CPS perspective. For this purpose, the section will briefly overview FIPA-ACL, which allows the communication between software agents that may be located on different computational platforms, MQTT, commonly used in IoT applications, and Modbus and OPC UA, protocols typically used in industrial environments.

2.5.1 FIPA-ACL

FIPA is an international organization that aims to promote the use of intelligent agent by establishing standards that allow the development of agent-based solutions, as well as addressing the interoperability between heterogeneous agents. The main way to ensure such interoperability is by defining a default framework for FIPA-ACL messages [26].

The FIPA-ACL message structure is defined through parameters that indicate the type of the communicative act, the involved participants, the communication control, content and message description. The choice of parameters is defined according to the type of application [19], [26]. Table 2.1 shows the parameters of the FIPA-ACL message.

Parameter	**Category of Parameters**
performative	Type of communicative acts
sender	Participant in communication
receiver	Participant in communication
reply-to	Participant in communication
content	Content of message
language	Description of Content
encoding	Description of Content
ontology	Description of Content
protocol	Control of conversation
conversation-id	Control of conversation
reply-with	Control of conversation
in-reply-to	Control of conversation
reply-by	Control of conversation

Table 2.1: Parameters of the ACL message [26].

Through these parameters, it is possible to communicate between agents by exchanging information and take proper decisions when a message is received.

2.5.2 MQTT

The MQTT [27], [28] is a protocol based on the publish-subscribe model widely used in IoT applications, due to its flexibility and lightness. It defines two entities, the broker and clients (i.e. subscriber and publisher). In this protocol, the exchange of messages is carried out by message topics and it is necessary that all the clients are connected to the broker, which is the entity that is responsible for managing the messages. Thus, when a publisher publishes a certain message topic, the broker receives this message and forwards for the subscriber who subscribed the same topic, as shown in Figure 2.4.

The MQTT protocol has 3 levels of Quality of Service (QoS) to guarantee the reliability of messages [27], [28]:

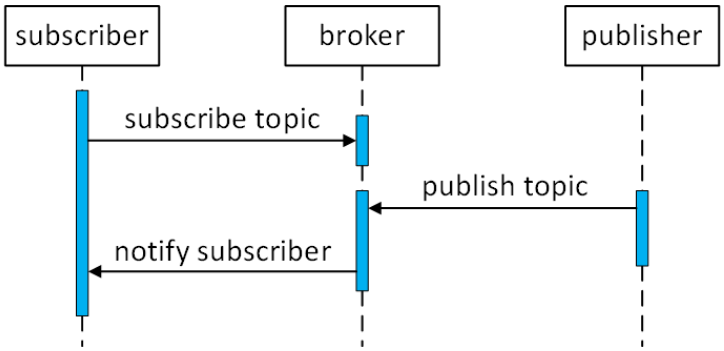

Figure 2.4: Publish-subscribe model using MQTT (adapted from [27]).

- Level 0: send the message only once and there is no guarantee of delivery.
- Level 1: send the message at least once. It is possible that the message is sent more than once.
- Level 2: send the message exactly once. At this level there are two confirmation steps, so when the process is concluded, both parties are sure that the message has been delivered. Although there is a guarantee that the message will be delivered exactly once, this level is the slowest.

In this protocol, the clients need to establish a connection with the broker, for this it is necessary to define some parameters, as described in Table 2.2.

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

Table 2.2: Parameters to connect to the broker [28].

After connecting to the broker, the client can perform a subscribe/publish in the broker through the parameters shown in the Table 2.3.

Parameter	Description
topic	The topic under which the message is subscribed/published
QoS	The Quality of Service Level
payload	Only for publish: the actual data in the message

Table 2.3: Parameters to subscribe and publish topics [28].

2.5.3 Modbus

The Modbus is an industrial protocol for communication with automation devices. Its operation is based on the client-server model, so the client initiates the communication with the server, requesting a certain service thought message, and the server receives this request and executes it [29]. In this work, the focus will be on the Modbus protocol with a Transmission Control Protocol (TCP) interface.

The Modbus TCP message structure defines how the data will be interpreted, as illustrated in 2.5. Basically, it consists of a Protocol Data Unit (PDU) that is independent of the network layer and the Modbus Application Protocol (MBAP) Header, that together form the Application Data Unit (ADU) that defines the network layer [30].

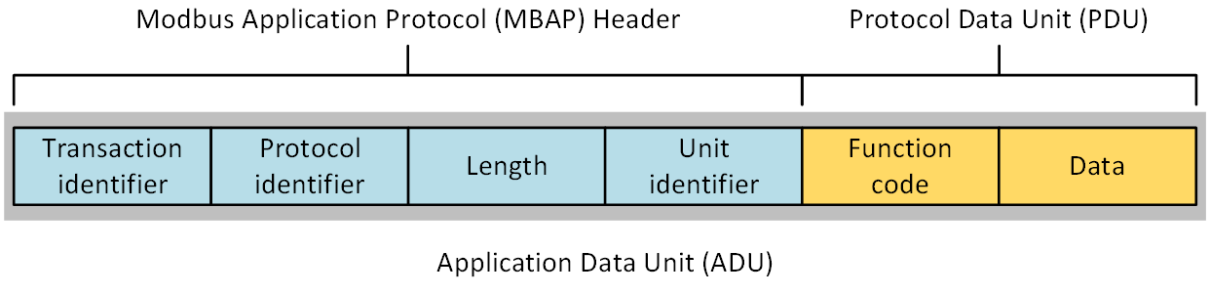

Figure 2.5: Structure of the messages for the Modbus TCP protocol (adapted from [29]).

The meaning of the each field in the message structure is the following:

- Function code: in this field the client specifies the type of service to the server.

- Data: additional information for the type of service selected in the function code, such as the quantity of items to be handled.
- Transaction identifier: this field is used to pair transactions when multiple messages are sent by the same client, so it is possible combine the request with the corresponding response.
- Protocol identifier: this field is always 0.
- Length: this field is used to count the next bytes, such as Unit identifier, Function code, and the Data fields.
- Unit identifier: used to identify a remote server on a non-TCP/IP Modbus network.

2.5.4 OPC UA

The OPC UA is a communication protocol widely used in the context of Industry 4.0, since it aims to integrate the different automation devices on the factory floor. For this purpose, this protocol defines a standard of interoperability for secure data transfer [31].

The OPC UA protocol is a client-server model with a possibility of being extended to a publish-subscribe model. Thus, in addition to the request and response mechanism, it is possible to OPC UA client to subscribe to the data updates of the OPC UA server [32].

To use this protocol there are a number of specifications that define the communication between the various parts of the system [33]. Each specification is described in Table 2.4.

Specification	**Description**
Overview and Concepts	Introduction to the OPC UA protocol
Security Model	Describes the security model
Address Space Model	Describes the addresses of an OPC UA server for the OPC UA client
Services	Describes the interfaces that servers and clients should use, expected behavior, and common data types
Information Model	Describes how the space, nodes and references of the OPC UA address are used
Mappings	Describes how the data is transferred between the servers and clients
Profiles	Describes the behavior that OPC UA servers and clients can implement
Data Access	Describes the concept of data access and its applications
Alarms and Condition	Describes the concept of alarms and conditions, and its applications
Programs	Describes the concept of programs and its applications
Historical Access	Describes how the data can be archived and retrieved from a database
Discovery and Global Services	Describes how UA products can be managed on a computer, network infrastructure, or enterprise
Aggregates	Describes the concept of aggregate functions and its applications
PubSub	Defines the OPC UA PubSub communication model and how to use

Table 2.4: Specifications to use the OPC UA protocol [33].

Chapter 3

Smart Parking Architecture

This chapter presents an overview of the smart parking architecture, describing how it will be structured, the role of each component that constitutes the system, and highlights the importance of the interconnection between the software agents with the low-level control devices to allow the access to the parking spots. The present work uses as reference the architectures for the smart parking system proposed in [34], [35].

3.1 Architecture Overview

The agent-based CPS architecture for the smart parking system is based on the MAS technology aligned with the holonic principles. Since a smart parking system is a complex, dynamic and large-scale system, the architecture needs to present a flexible, adaptable and modular structure, capable of ensuring the maximum efficiency, handling dynamic tasks and interacting with the physical world through the interconnection between the several software agents with the physical asset controllers, constituting a CPS. For this purpose, the inclusion of other technologies besides MAS and CPS, such as IoT, cloud computing and machine learning are necessary to ensure the proper functioning of the system. Figure 3.1 illustrates the agent-based CPS architecture for the smart parking system.

The architecture is composed of several agents that interact with each other, forming

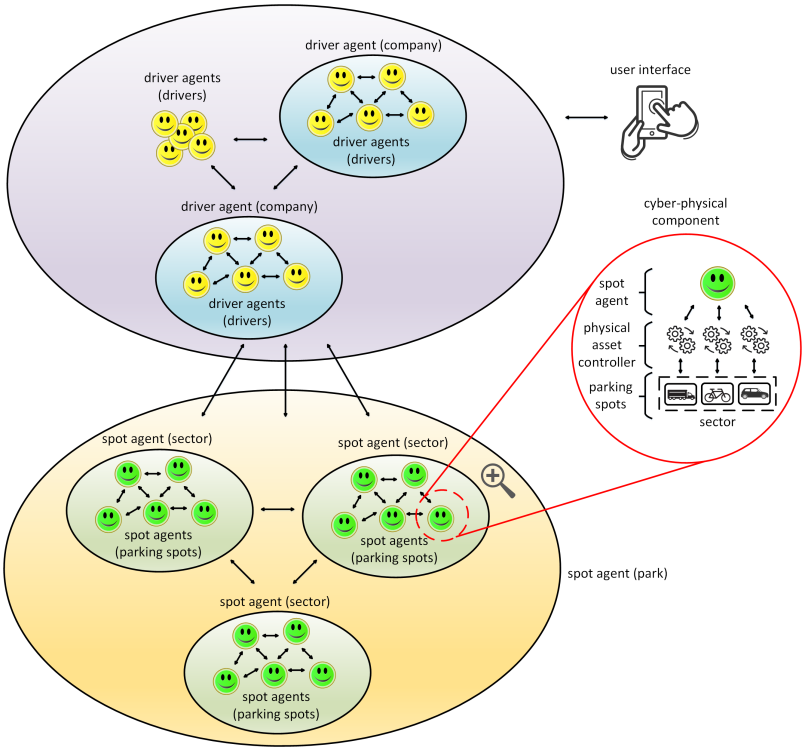

Figure 3.1: Smart parking architecture [35].

a society of intelligent, autonomous and cooperative agents. In this system, there are two types of agents, the driver and spot agents. The driver agents represents the drivers that wants to park its vehicles and are running in the cloud, accessed through a user interface, for example, a smartphone application. The spot agents are responsible for manning all the parking spots or a group of parking spots. In this way, the spot agents are interconnected with the physical devices, namely sensors and actuators, in order to enable distributed collection of data and the actuation of the physical devices.

In this architecture, the inclusion of MAS and CPS ensure inherent characteristics of these technologies, such as flexibility, modularity, adaptability and reconfigurability. In this way, the smart parking system can be adapted to any type of vehicle (e.g., a bike, car or truck), the modular structure extends the smart parking in scalable terms and facilitates its implementation through the addition of new modules (i.e., cyber-physical components), the relation between the spot agent and the physical asset controller is flexible, i.e., m spot agents can be interconnected with n physical asset controllers, and

re-configured on-the-fly, i.e., new parking spots or drivers can be easily added, removed or modified without stopping, reprogramming and restarting the entire system.

3.2 Inclusion of Holonic Principles

The concept of holon is similar to an agent, it is an intelligent, autonomous and cooperative entity, capable of interacting with the environment and making decisions to solve problems. These entities have the property of performing functions as a "whole" and a "part" at the same time, guaranteeing recursive characteristics, which simplifies the implementation of complex and large-scale systems. A holarchy is defined as a system of holons organized in a hierarchical structure, cooperating to achieve the objectives of the system, combining their individual skills and knowledge [36].

To use this architecture in a large-scale environment, the recursive capabilities of holonic principles simplify this process by building holarchies of drivers and spot agents. In this sense, a driver agent can be at the same time the "whole", e.g., representing a driving company, and the "part", e.g., representing a single driver, and a spot agent may represent a sector which in turn can comprise several parking spots, each one represented by a spot agent, as shown in the Figure 3.1.

3.3 Global Functioning

The global functioning of the system is based on the interaction between the spot and the driver agents, the holonic principles and negotiation process. In this sense, the drivers can interact with several spot agents to find a free parking spot with the desired specifications, e.g., location, price and time. Otherwise, the spot agents can recommend free parking spots based on driver profiles. For this purpose, the inclusion of machine learning techniques can maximize the intelligence capabilities of the agents. In this way, it is possible to prediction of occupancy of parking spots, taking into account participatory decision variables and atmospheric forecasting variables.

The negotiation process for a parking spot assumes a crucial interaction pattern in smart parking systems, where driver agents interact with spot agents to reserve a parking spot respecting the specifications defined by the driver, e.g., location, type and parking time. For this purpose, several negotiation strategies can be used, namely the Contract Net Protocol [37], the English auction [38] and the Dutch auction [39], which applicability to smart parking systems were tested and compared in [40].

After the desired parking spot is reserved, the driver can access the parking spot in the proper slot of time. The access to the parking spot is based on CPS technology, consisting of a cyber and physical part. The cyber part is represented by a spot agent and the physical part is composed of a physical asset controller. This configuration allows the spot agent to communicate with the physical asset controller, and then enable access to the parking spot.

Although an overview of the smart parking system has been presented, describing the fundamental points for the global functioning. The present work focuses only on the interconnection between the spot agents with the physical asset controllers to control the access to the parking spots.

Chapter 4

Interface Practices

As previously described, the main objective of this work is to study the interface practices to interconnect software agents and low-level control devices, in a CPS perspective and addressing the parking systems problem.

For this purpose, this chapter presents an overview of existing interface practices, the implementation of some interfaces in terms of location, interaction mode, computational platform and communication protocols, the methodology for testing these interfaces taking into account the response time, technology, scalability and re-usability. Finally, the obtained results are analyzed and discussed, verifying which interface practice is the most suitable solution for a smart parking system.

4.1 Recommended Practices

Agent-based systems allow the incorporation of intelligence into the CPS, decentralizing the system through autonomous entities and expanding the application domains. This new configuration, known as agent-based CPS, characterized by flexibility, robustness and responsiveness, is a suitable approach for the design of complex and large-scale systems, such as smart parkings and industrial environments. For this purpose, it is necessary to establish an interface between the software agents with the low-level control devices. However, to date there is no standardization available of how to perform this interface

[41].

In this context, the IEEE P2660.1 Working Group [42] is developing recommended practices for integration between the software agents and low-level control devices. This work is divided into four phases [41]:

- Analysis phase: performs the analysis of the existing interface practices.
- Generalization phase: defines and characterizes generic models from the interface practices of the previous phase.
- Assessment phase: defines a methodology to evaluate and classify each of the models of the generalization phase.
- Recommendation phase: based on the results from the previous phase, each practice is recommended for a specific application.

To achieve the objective of the present work, which is the interconnection of software agents with physical asset controllers, constituting an agent-based CPS for the smart parking system, the studies [34], [41], [43], [44] performed by the IEEE Working Group P2660.1 are used as references.

In the analysis phase, the research [43] about the existing practices in the fields of industrial automation, energy & energy systems and building automation, consisted of analyzing similarities between existing practices to determine the best practice. From this study, the conclusions obtained were the use of JAVA and C++ as the main programming languages for the implementation of software agents, the preference for using remote approaches and that there is no best practice, but recommended practices for each specific application.

In the generalization phase, through the practices analyzed in [43], the interface between the software agents with the physical asset controller can be grouped according to two dimensions: location and mode of interaction, as reported in [34], [41]. The location refers to where the agent is hosted in relation to the physical asset controller. There are two possibilities:

- Remote: the agent and the physical asset controller are executed in different computational platforms.
- On-device: the agent and the physical asset controller are executed in the same computational platforms.

The interaction mode corresponds to how the software agent interacts with the physical asset controller. In this case, also two possibilities were considered:

- Client-Server model: in this structure, the client is a program that requests a service, and the server is a program which provides the service. So the client requests a service and the server executes it. Based on this model, the software agent maintains a direct and synchronized connection with the physical asset controller.
- Publish-Subscribe model: this model is composed of a publisher, subscriber and a message broker. In this structure the communication is based on message topics, and the exchange of messages between the system components is intermediated by the message broker. In this way, the subscriber subscribes a topic, and when the publisher publishes the message in the same topic, the broker forwards this message to the subscriber. Based on this model, the interaction between the software agent and the physical asset controller is intermediated by a message broker, establishing an indirect and asynchronous connection.

According to the previous classification, four generic interfaces practices were considered for analysis, namely *Remote/Client-Server*, *Remote/Publish-Subscribe*, *On-device/Client-Server* and *On-device/Publish-Subscribe*. These generic interfaces can be implemented using different technologies in terms of computational platform and communication protocols.

The computational platform refers to what type of computational platform is used, e.g., a computer, a robot and a Programmable Logic Controller (PLC). And for the communication protocol the same principle is used, i.e., which protocol is used, such as FIPA-ACL, Modbus, OPC UA and MQTT.

In the Assessment phase, the IEEE Working Group P2660.1 uses ISO/IEC 25010, which defines a product quality model, as shown in Figure 4.1. From this ISO/IEC 25010, the evaluation criteria are defined to identify the recommended practices [44]. The present work follows the same principle, i.e., it defines some evaluation parameters considering the current application, the smart parking system, so the chosen parameters are: response time, technology, scalability and re-usability.

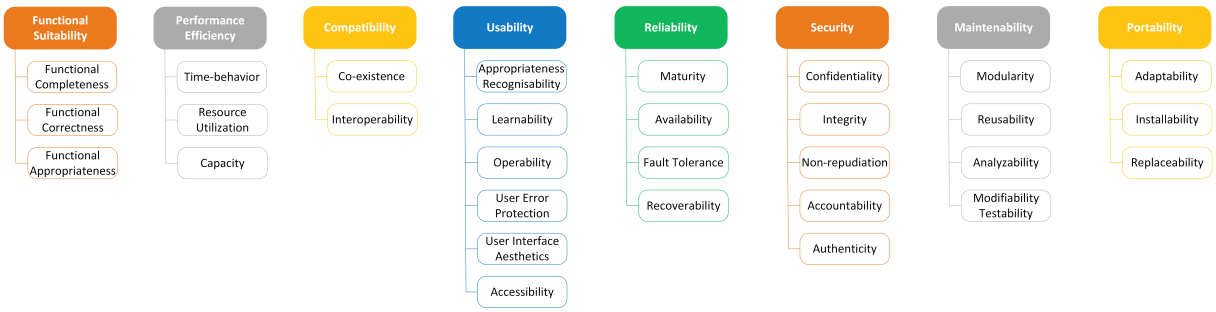

Figure 4.1: ISO/IEC 25010 quality model (adapted from [44]).

The recommendation phase will be discussed with more detail at the end of this chapter, where the results from the experimental implementation will be shown and it will be possible to determine the recommended practices for each application in terms of the parameters considered, particularly for smart parking application.

4.2 Experimental Deployment of Different Interface Practices

This section defines the structure to test the generic interfaces practices. This structure is composed of a software agent, namely spot agent, a communication channel and a logic controller. The software agent can be run in any computational platform that supports its implementation. The logical control of the physical asset controller can share the same computational platform of the software agent or different platforms. Finally, the communication channel is defined based on the model used, such as client-server or publish-subscribe. These models can be implemented using different communication

protocols. Figure 4.2 illustrates the generic structure for testing the interface practices.

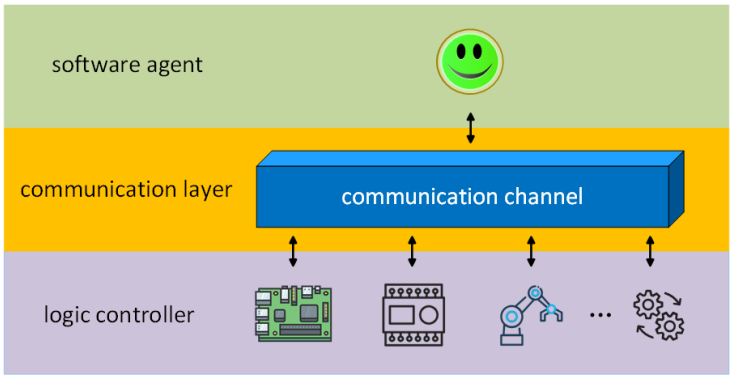

Figure 4.2: Structure of the generic interface practice.

In this work, the software agent is the spot agent of the smart parking system, which was developed in the JAVA Agent Development Framework (JADE) and deployed and executed in a Raspberry Pi 3 Model B+ [45]. JADE is a java-based framework that simplifies the development of MAS solutions [19], [46], by providing supporting functionalities like yellow pages, white pages and debug tools, namely a message sniffer and an agent introspector. The logic control of the physical asset was running in different computational platforms, namely another Raspberry Pi, an UR3 robot controller, a Modicon M340 PLC and an OPC UA server, allowing to test different technological implementations of the interface practices.

4.2.1 Remote/Client-Server

The *Remote/Client-Server* approach is characterized by using the client-server model and the separation of the computational platform, i.e., the spot agent is located in a different computational platform from the physical asset controller. So the communication between the spot agent with the physical asset controller is performed thought a direct communication channel. This interface practice is illustrated in Figure 4.3.

According to Figure 4.3, to test this interface it was necessary to use two Raspberry Pi, namely Raspberry Pi #1 and #2. The spot agent is running on the Raspberry Pi #1 together with the JADE framework, and the physical asset controller its running on

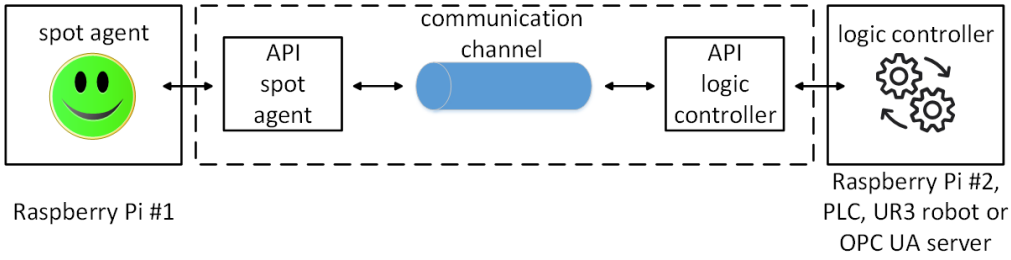

Figure 4.3: *Remote/Client-Server* interface practice [34].

the Raspberry Pi #2. For establishing the communication between the spot agent and the physical asset controller, it is necessary that both platforms be connected to the same network. In this approach, the communication with the spot agent was also tested with three different types of control devices, the M340 PLC, UR3 robot controller and OPC UA server, allowing to verify the influence of technologies in this interface. Basically, the spot agent is responsible for sending data thought messages to the logic controller, which had the function of receiving this message and sending a confirmation.

4.2.2 Remote/Publish-Subscribe

As a remote approach, this interface practice follows the same principle of the previous practice in terms of the separation of computational platforms. However, this practice uses the publish-subscribe model through the MQTT protocol. In this way, the interaction between the spot agent and the physical asset controller occurs through message topics and is intermediated by a message broker, defining an indirect communication channel, as illustrated in Figure 4.4.

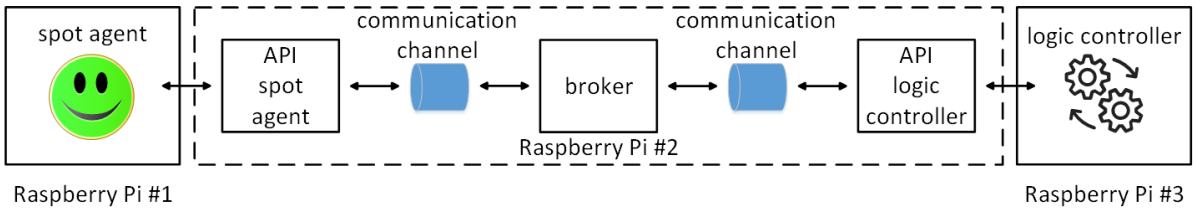

Figure 4.4: *Remote/Publish-Subscribe* interface practice [34].

This interface practice uses three Raspberry Pi, the JADE framework and the spot agent are running on the Raspberry Pi #1, the broker on the Raspberry Pi #2 and the physical asset controller on the Raspberry Pi #3. To verify this implementation, the spot agent subscribes a message topic in the broker and when the logic controller publishes the same message topic, the broker forwards this message to the spot agent.

4.2.3 On-device/Client-Server

This approach uses a client-server model and is characterized by the sharing of the computational platform between the spot agent and the physical asset controller. Thus, the spot agent, JADE framework and physical asset controller are running in the same Raspberry Pi, communicating through a local hardware communication channel, as illustrated in Figure 4.5.

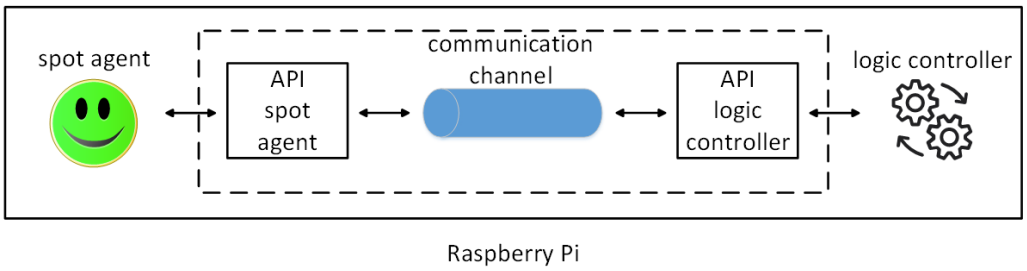

Figure 4.5: *On-device/Client-Server* interface practice [34].

4.2.4 On-device/Publish-Subscribe

In a similar way to the previous interface practice, in this approach the spot agent, JADE framework, broker and physical asset controller share the same computational platform, as illustrated in Figure 4.6.

Thus, these components are hosted in a single Raspberry Pi, now interacting through the publish-subscribe model. In this interface practice, the broker does not allow the direct access between the parts, being required a layer between them.

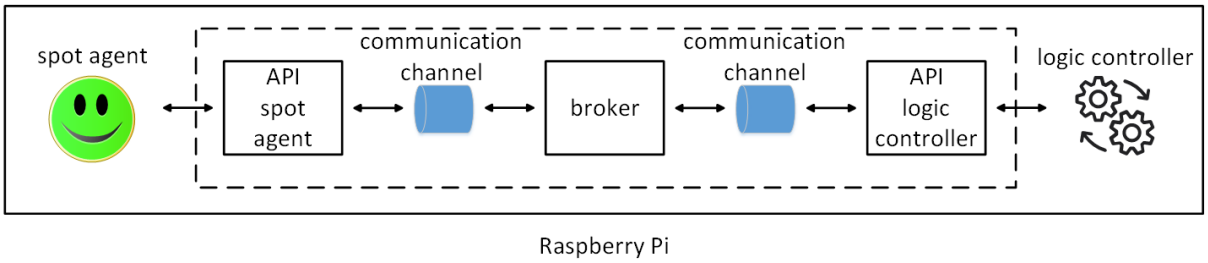

Figure 4.6: *On-device/Publish-Subscribe* interface practice [34].

4.3 Experimental Testing Methodology

The interface practices described above were tested based on a set of experiments. These tests were performed with the objective of verifying some parameters, such as response time, technology, scalability and re-usability. For this purpose, the RTT was measured, time measured from the moment that the spot agent sends a command to the physical asset controller until it receives a confirmation response. Figure 4.7 illustrates the scheme for measuring the RTT.

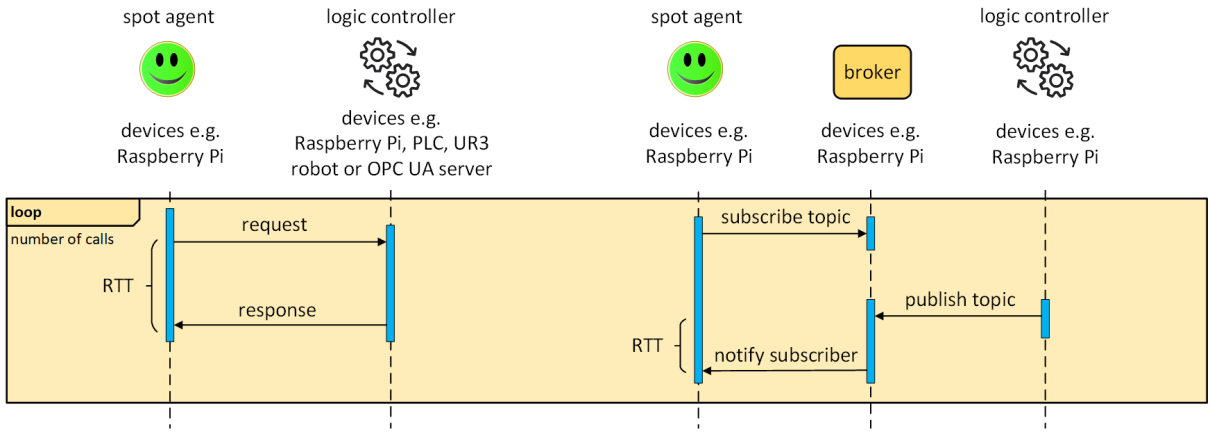

Figure 4.7: Process to measure the RTT: client-server model (left) and publish-subscribe model (right) (adapted from [34]).

In the case of client-server models, the RTT is measured from the moment that the spot agent sends a command to the physical asset controller until it receives a confirmation response. For the publish-subscribe models, the RTT is measured differently, since in this configuration there is a broker mediating the exchange of messages between the spot

agent and the physical asset controller. In this way, the RTT was measured from the moment that the physical asset controller makes a publish in the broker until the spot agent receives a notification by the broker.

In order to analyze the parameters, such as response time, technology, scalability and re-usability, variables were defined to perform the experimental tests, such as number of calls, message size and target cycle time. These variables can have a fixed value, or follow the A-B-C pattern, i.e., A is incremented in B until it reaches C, for example, if the default is 100-100-1000, first 100 calls will be made, then 200, 300 until it reaches 1000. The number of calls indicates the number of repetitions that the spot agent communicates with the physical asset controller. The target cycle time defines the time between each call, which can be n ms from the moment it receives the response or "freewheeling", that makes the next call as soon as it receives the confirmation. Finally, the message size is set to bytes.

To analyze the response time, a first scenario is defined to test the performance of each interface practice. The Table 4.1 shows the values selected for these variables.

Test	**Number of Calls**	**Message Size (Bytes)**	**Target Cycle Time (ms)**
1	100-100-1000	1000	10
2	100-100-1000	1000	freewheeling
3	1000	1000	10-10-100

Table 4.1: Summary of experimental tests for the scenario #1 (analysis of performance) [34].

For the *Remote/Client-Server* approach, the spot agent can be interconnected with different types of control devices, such as M340 PLC and UR3 robot thought Modbus protocol or OPC UA server using OPC UA protocol. Thus, the second scenario aims to verify the dependency of the interface practice according to the technological implementation in terms of communication channel and computational platform. Table 4.2 shows the selected values to test this experimental scenario.

To verify the scalability of the interface practices, the third scenario performs tests based on the number of spot agents in the system, so the other parameters are constant and only the number of spot agents is changed. In these tests, each spot agent communicates

Control Device	Number of Calls	Message Size (Bytes)	Target Cycle Time (ms)
Raspberry Pi	1000	1000	freewheeling
PLC	1000	1000	freewheeling
UR3 robot	1000	1000	freewheeling
OPC UA server	1000	1000	freewheeling

Table 4.2: Summary of experimental tests for the scenario #2 (dependency of technological implementation) [34].

simultaneously with the physical asset controller according to the values selected in Table 4.3.

Number of Agents	Number of Calls	Message Size (Bytes)	Target Cycle Time (ms)
1	1000	1000	freewheeling
10	1000	1000	freewheeling
50	1000	1000	freewheeling
100	1000	1000	freewheeling

Table 4.3: Summary of experimental tests for the scenario #3 (scalability) [34].

4.4 Analysis of Experimental Results

The results obtained with the experimental tests are described in the following sections.

4.4.1 Analysis of the Response Time

In this section will be presented the result of only some tests due to large quantity, however the tests presented here are sufficient for the analysis of the results. The rest of the tests are in the Appendix A.

The scenario #1 allowed to analyze the performance of each interface practice according to the response time. The results of test #1 are shown in Figure 4.8.

Based on the results obtained with test #1, it was possible to verify the influence of two factors on response time, network and processing power. Thus, it was verified that the on-device practices exhibited the lowest RTT values, which is justified by the inexistence of a communication network subject to delay times. Although the network

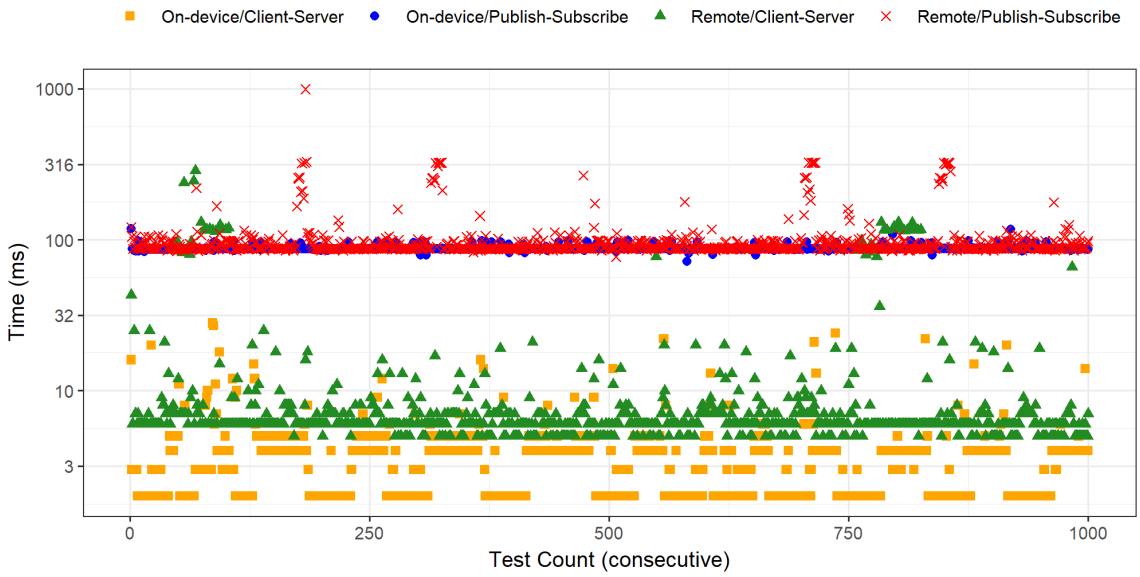

Figure 4.8: Response time for the four interface practices according to test #1 [34].

does not influence on-device practices, the spot agent and physical asset controller share the same computational platform, which requires a higher level of processing power. In some cases, this advantage ceases to be expressive, especially in the publish-subscribe models, where it is possible to observe a small difference in the RTT values.

Another factor that influences the response time is the use of a message broker. It was possible to verify that the client-server model present smaller RTT values than the publish-subscribe model, verifying the strong influence of the broker by introducing a small delay in the communication between the spot agent and the physical asset controller.

When comparing test #1 with test #2, the RTT value was verified for a target cycle time of 10 ms and freewheeling. For the approaches that use the publish subscribe model, the RTT values were smaller for a fixed time of 10 ms than freewheeling. On the other hand, for the client-server model, there were no significant changes, as shown in Figure 4.9, compared to test #1 represented by Figure 4.8.

The test #3 confirmed the influence of the target cycle time for the approaches analyzed, and just as tests #1 and #2 showed, there were only significant changes in the publish-subscribe model, as shown in Figure 4.10. In this way, it can be verified that

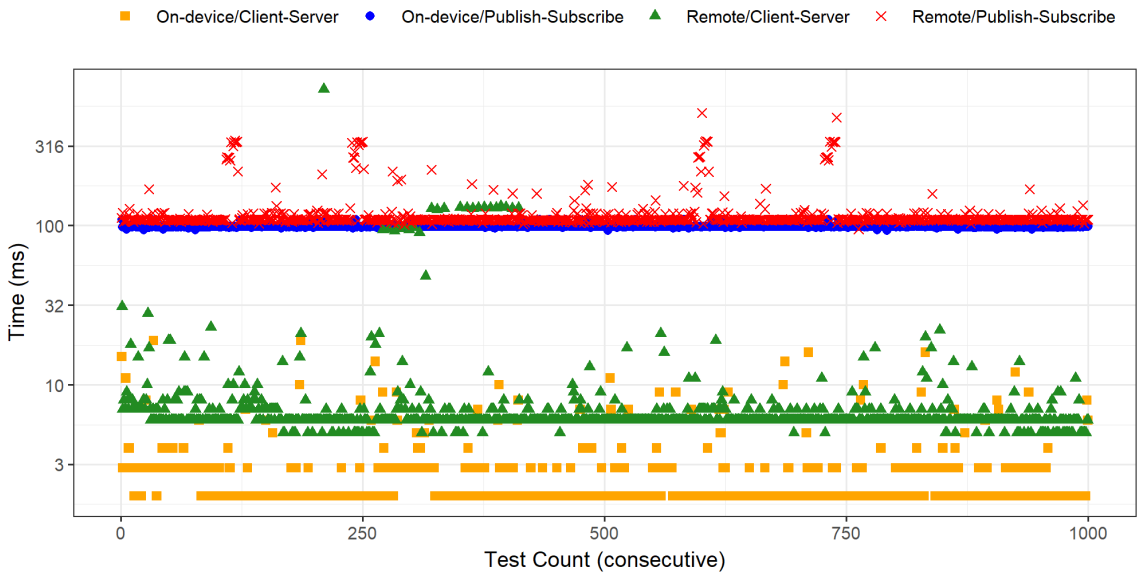

Figure 4.9: Response time for the four interface practices according to test #2 [34].

according to the target cycle time increases, the response time improves with an average of approximately 50 milliseconds, for both the *On-device/Publish-Subscribe* and *Remote/Publish-Subscribe* approaches, since the broker is not overloaded, taking a time interval between each call.

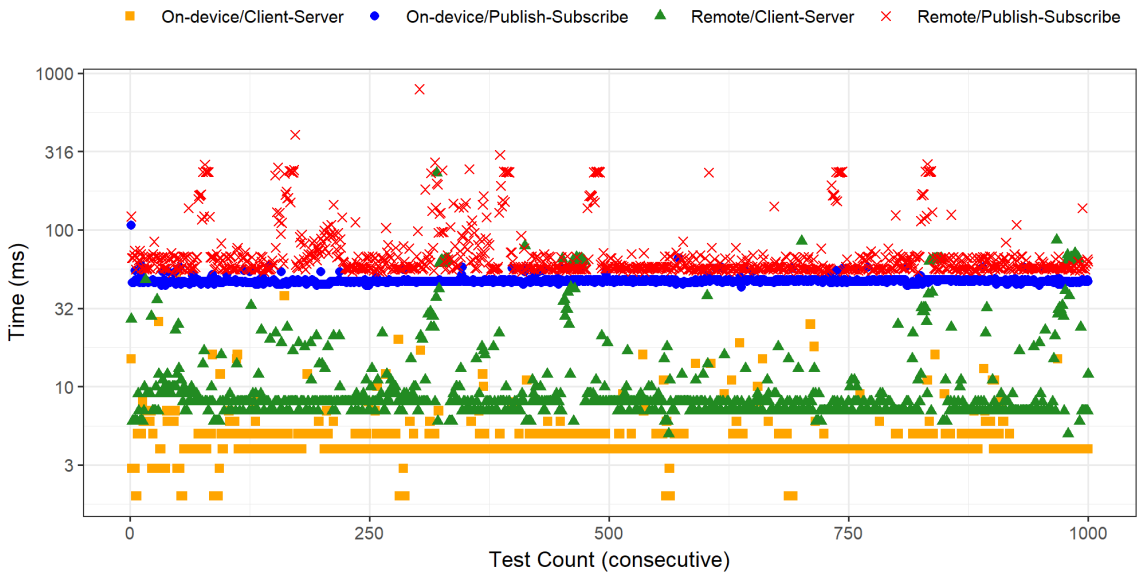

Figure 4.10: Response time for the four interface practices according to test #3 [34].

After this set of tests, the best interface practice in terms of response time was the *On-device/Client-Server*, which presents the lowest response time and the small standard deviation.

4.4.2 Analysis of the Dependency of Technological Implementations

The scenario #2 analyzed the influence of technological implementation in the response time. According to the results obtained, for the same approach, the *Remote/Client-Server*, when the spot agent was interconnected with the PLC and UR3 robot using Modbus, a lower RTT value was presented when compared to the *Remote/Client-Server* approach that uses the FIPA-ACL or OPC UA protocol, as illustrated in Figure 4.11. It was also possible to verify the influence of the computational platform in the response time. When using the UR3 robot, the response time was better than using the PLC.

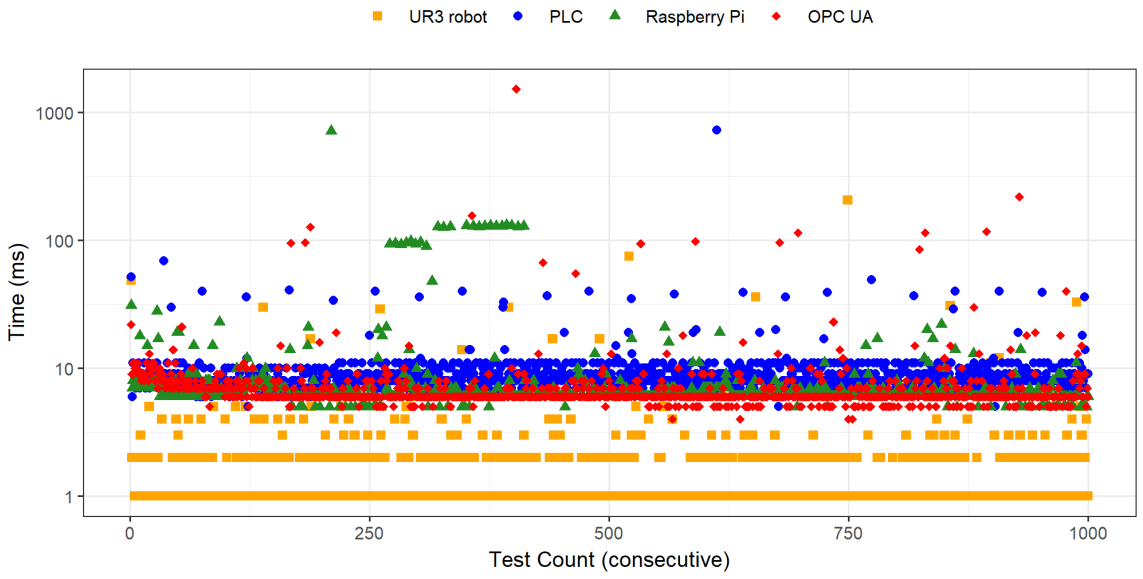

Figure 4.11: Comparison between different technologies for the *Remote/Client-Server* interface practice.

4.4.3 Analysis of the Scalability and Re-usability

The last scenario allowed to verify the scalability and re-usability of each interface through the increase of the number of agents. Figure 4.12 shows the results obtained for each interface practice.

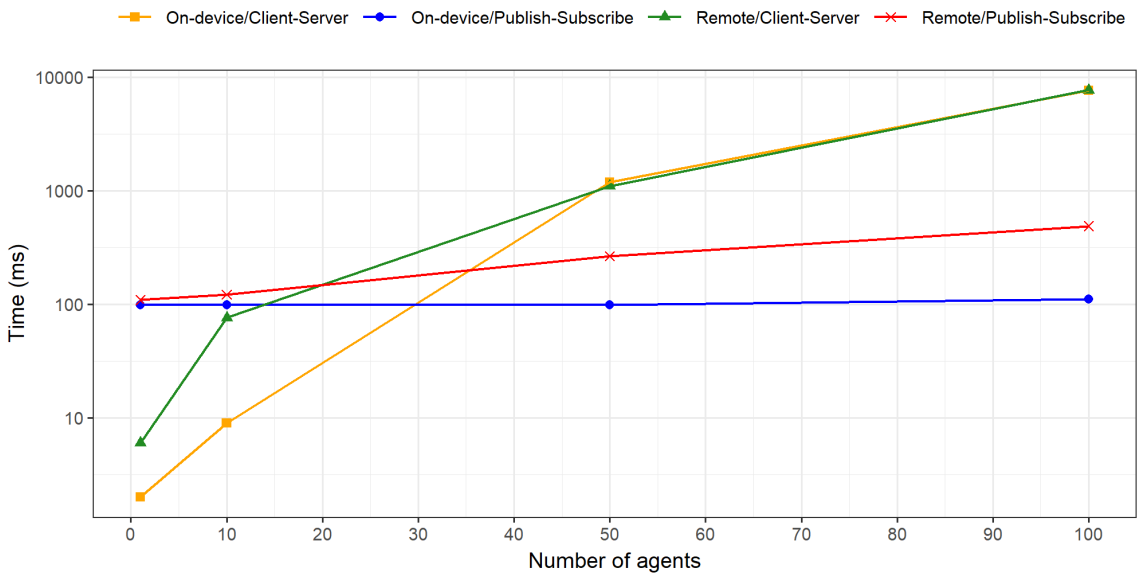

Figure 4.12: Response time for the four interface practices with the variation of the number of the spot agents [34].

In the approaches that use the client-server model it is possible to observe a greater influence on the response time with the increase of the number of agents, this is justified by the server overload that does not handle as well when it is requested by several agents at the same time. On the other hand, approaches that use the publish-subscribe model are less influenced, since the broker ensures a better management of the exchange of information among the entities.

The location is another factor to be considered in this scenario, verifying the influence of the network. For the publish-subscribe model, the remote approach always presents higher RTT values, i.e., the network plays a much more crucial role than the processing required by the on-device approach. For the *On-device/Client-Server* interface, in approximately 50 agents, the advantage of not using the network for communication is

canceled by the overhead generated by the number of agents, at this moment the RTT values are very close to the *Remote/Client-Server* interface.

Based on the results obtained, for applications that do not require many agents, the best option is the *On-device/Client-Server* that presented the lowest RTT up to 29 agents. After this number of agents, the *On-device/Publish-Subscribe* approach becomes the best solution, presenting small standard deviation and lowest RTT values.

In terms of re-usability, the approaches that use, for example, MQTT, Modbus or OPC UA, promotes the re-usability since only a few alterations are required. For interface practices that not using the standardized technologies, more adaptations have to be made, which affects the re-usability.

Chapter 5

Cyber-Physical Interconnection

This chapter discusses the low-level architecture of the smart parking system, taking into account the results obtained in chapter 4. This architecture, different from that described in chapter 3, focuses only on the interconnection between the software agents and the low-level control devices to control the access to the parking spots.

5.1 Structure of Agent-Based CPS

Based on the experimental results obtained in Chapter 4, the interface practice that best fits the smart parking system problem is the *Remote/Publish-Subscribe*. This approach provides the necessary requirements for the proper functioning of the system, where real-time responses are not required and parameters such as scalability and monitoring play a much more significant role for this application. Taking into account the working principle of this interface practice and the interconnection between the software agents and the low-level control devices to control the access to the parking spots, the proposed generic structure for the agent-based CPS is illustrated in Figure 5.1.

In this modular structure, the spot agents are responsible for managing a sector comprised of several parking spots that are controlled by physical asset controllers. The driver agents represent the drivers that want to park their vehicles, for this purpose it is necessary to reserve the parking spot through negotiation strategies. After the negotiation

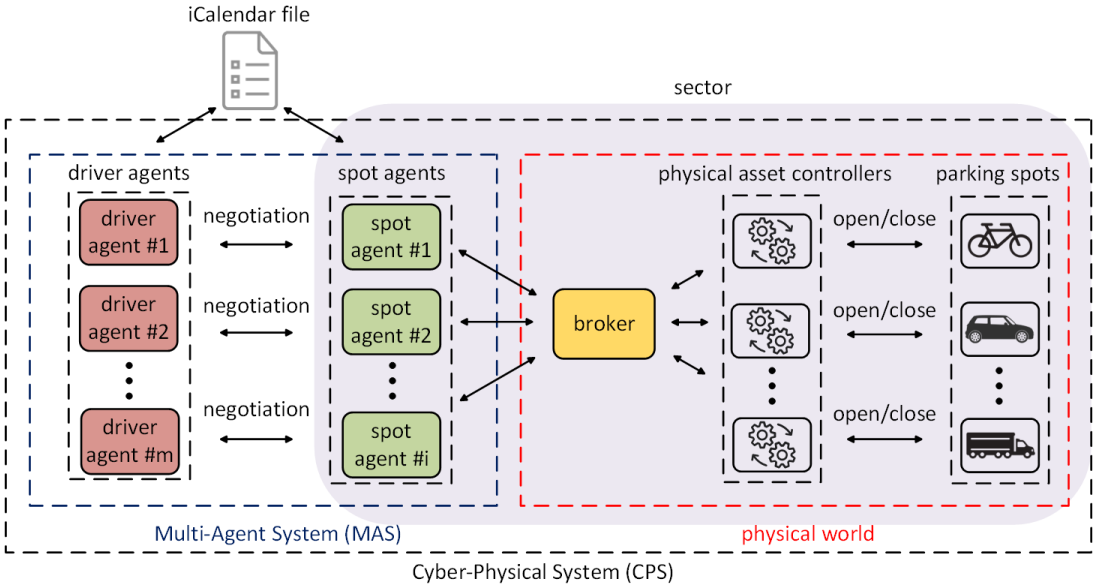

Figure 5.1: Generic structure of agent-based CPS [35].

process between the driver and spot agents, the reservation information is stored in iCalendar file [47]. Once the reservation process is completed, the parking spot is available to be used by the driver in the specific slot of time.

5.2 Reservation Registers

The use of a mechanism to register reservations for a smart parking is essential for the spot agent to manage the access to the parking spots. For this purpose, the iCalendar is used to perform these records. This file contains the reservation registers for the parking system, with the following fields: String driverId, String parkingSpotId, String status (i.e., if the parking spot is free, occupied or timeout), Date startParkingDate and Date EndParkingDate.

As a smart parking system is a dynamic system, at all times new reservations are recorded and requests to use parking spots are made. Thus, the spot agent needs to read this file whenever it is requested the access to the parking spot. By reading iCalendar, the spot agent can check if a possible driver can access the parking spot in that period.

5.3 Communication by Message Topics

The communication between the spot agent and the physical asset controllers is performed by message topics and intermediated by a broker. In this way, the use of the parking spot topics, helps the spot agent find out what message topic is necessary to interact with the physical asset controller that controls the access to reserved parking spot. Figure 5.2 illustrates how the spot agent and physical asset controllers interact with message topics.

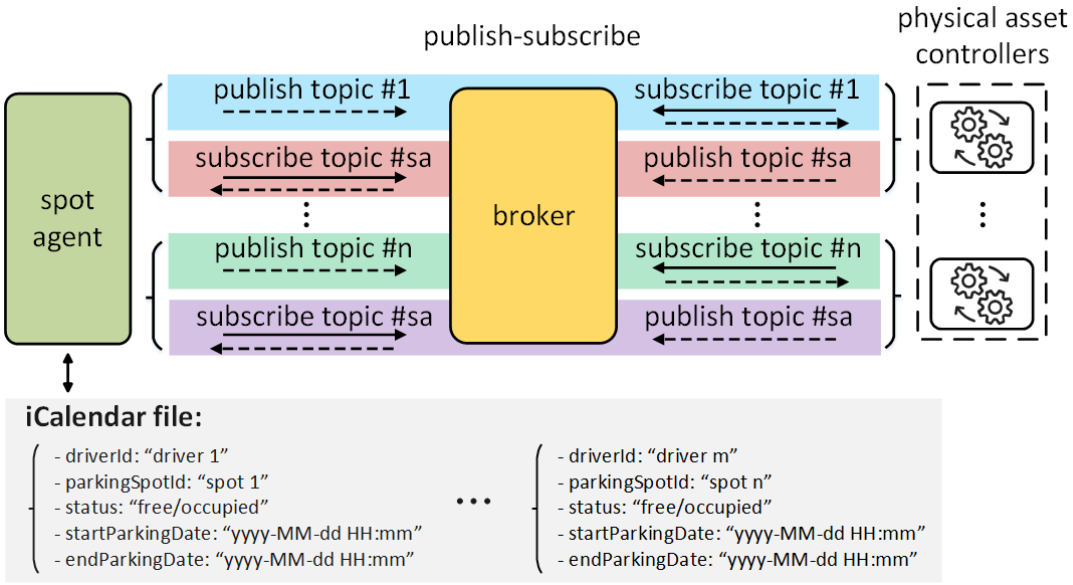

Figure 5.2: Scheme for exchanging messages by topics [35].

The spot agent uses specific topics to communicate with each physical asset controller, maintaining an independent communication. For the communication in the opposite direction, from the physical asset controllers to the spot agent it is only necessary to use a message topic for everyone, since there is only one spot agent per sector.

5.4 Interaction Scheme

The sequence diagram shown in Figure 5.3 explains the operation of the generic structure of agent-based CPS. This interaction scheme considers that the negotiation process has already been carried out and the desired parking spot has been reserved by the driver.

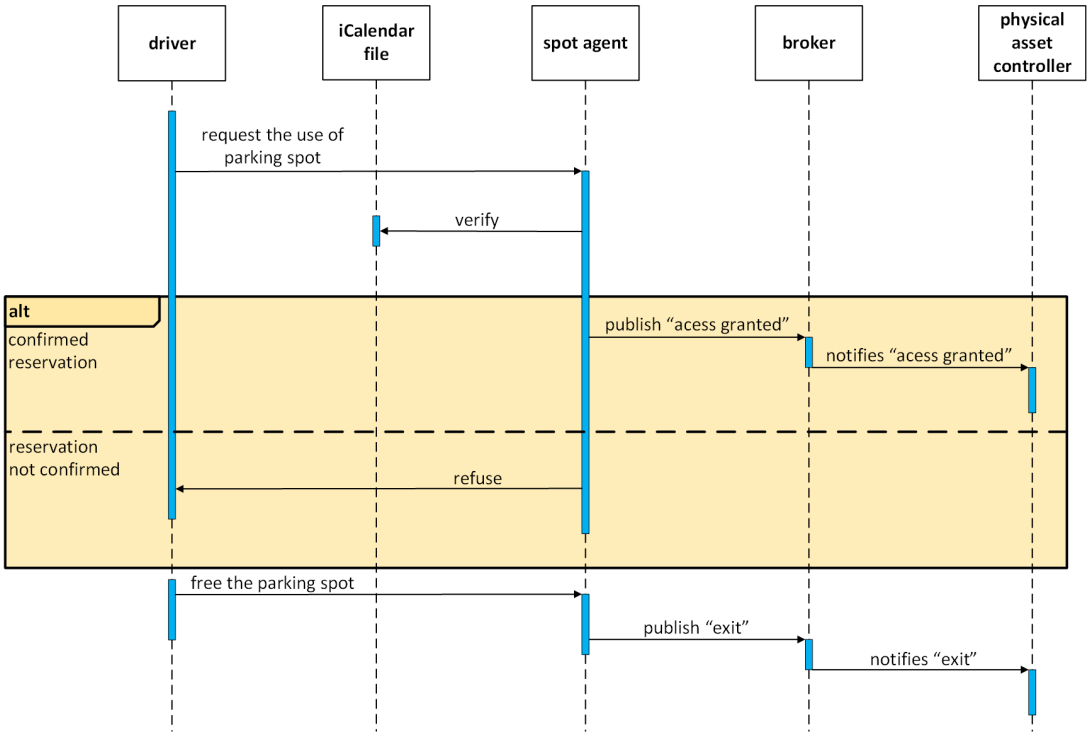

Figure 5.3: Sequence diagram for the cyber-physical interconnection [35].

Having this in mind, the sequence diagram follows the steps described below:

1. The driver sends a access command to the spot agent responsible for managing the reserved parking spot.

2. The spot agent checks the iCalendar file if the driver can access in the specific slot of time.

3. If the reservation information is correct, the spot agent sends a access command to the physical asset controller located in the reserved parking spot to allow the parking of the vehicle.

4. If the reservation information is not correct, e.g., the start time has not yet started or the reservation time has already expired, the spot agent notify the driver that can not access the parking spot in that time.

5. After using, the driver sends a exit command to the spot agent, which forwards to

its respective physical asset controller to free the reserved parking spot.

5.5 Implementation

According to Figure 5.4, the spot agent that manages a sector composed by several parking spots was developed in JADE and was deployed in Raspberry Pi 3 Model B+, namely Raspberry Pi #1. Raspberry Pi was chosen because it is a low-cost computing platform and presents the necessary requirements for implementing software agents. The physical asset controller is composed by a microcontroller, namely ESP8266, and a logical control, which together are responsible for giving or not accessing the bicycle parking spot. The iCalendar file is located in Raspberry Pi #1. Finally, the Eclipse Mosquitto [48] was used as broker, an open source broker that implements the MQTT protocol. The Eclipse Mosquitto broker was run on different Raspberry Pi, namely Raspberry Pi #2.

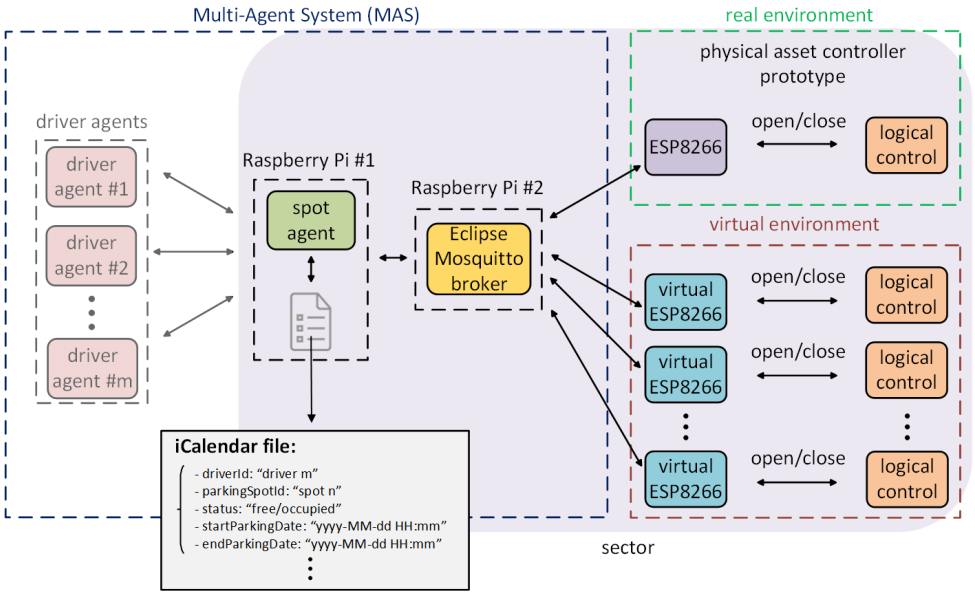

Figure 5.4: Implemented cyber-physical interconnection for the parking system of bicycles [35].

In order to test scalability, a virtual parking spot were created. For this purpose, it was necessary to develop a device capable of simulating an ESP8266, namely virtual ESP8266. This device was developed in Node-RED, a visual flow-based programming

tool that simplifies the development of IoT applications. Figure 5.5 illustrates the virtual ESP8266 developed in the Node-RED platform.

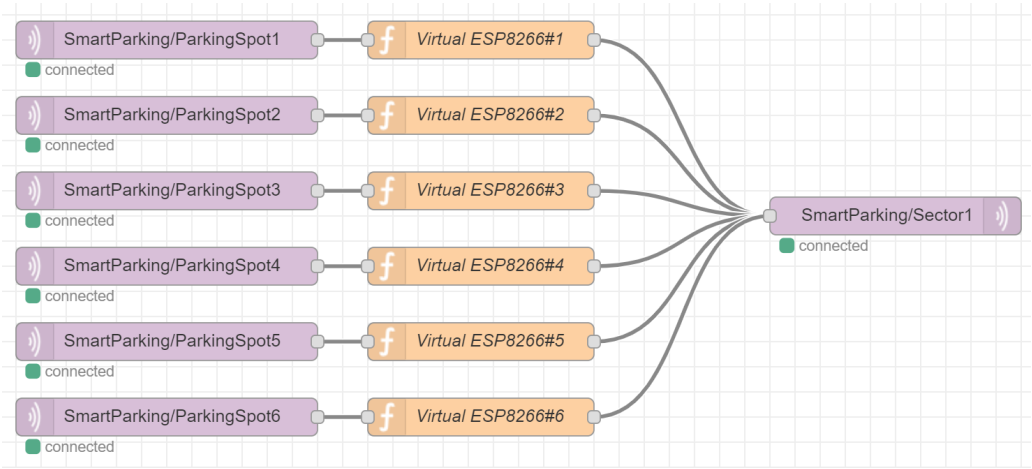

Figure 5.5: One sector composed by Virtual ESP8266 developed in the Node-RED.

This programmable component simplifies the performing of scalable tests, since it is a replicable block with the same functions as an ESP8266, i.e., it receives commands to use a parking spot, processes that information and executes it.

In this way, 7 additional sectors were added, being each one managed by one spot agent and comprising 6 parking spots, each one represented by a virtual ESP8266. Figure 5.6 shows the monitoring dashboard of the smart parking system developed in Node-RED.

This dashboard shows a map of the parking system, the driverId to know which driver is using the parking spot and the current status of each parking spot. There are three possible status for a parking spot:

- Occupied: represented by a red circle. It means that the parking spot is occupied.
- Free: represented by a green circle. It means that the parking spot is free to be used.
- Timeout: represented by a black circle. It means that the parking spot is occupied, but the reservation time has expired.

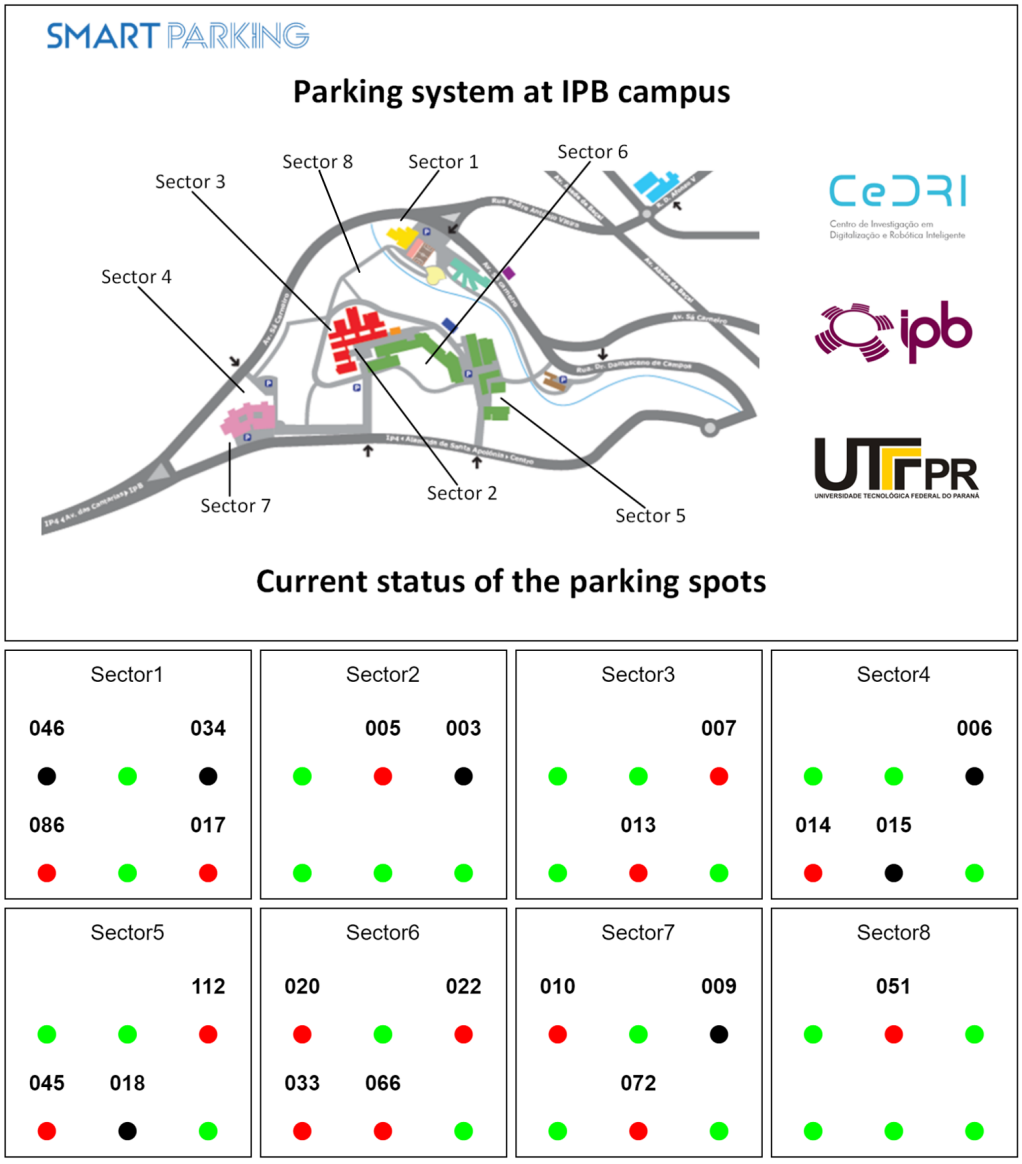

Figure 5.6: Monitoring dashboard for the bicycles smart parking [35].

To verify how the spot agent manages several parking spots, an experimental scenario was simulated. Thus, some commands were requested for the spot agent of each sector, such as the access or release the parking spot, checking through the Node-RED dashboard, the current status of each parking spot after the cyber-physical interaction to allow the access to the parking spots. These commands were performed through an user interface developed in [49], as illustrated in Figure 5.7.

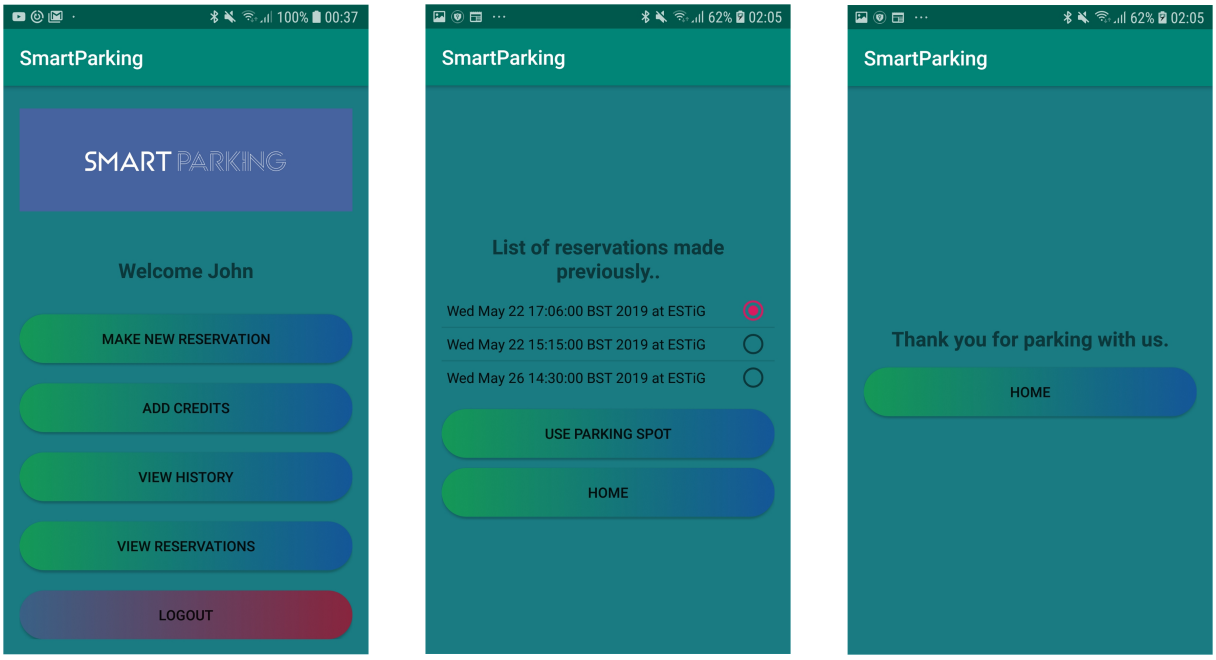

Figure 5.7: User interface: start screen (left), screen to request the use of the parking spot (center) and confirmation screen (right) [49].

5.6 Experimental Results and Discussions

The implemented agent-based CPS performed correctly during the tests, indicating that the interface chosen, the *Remote/Publish-Subscribe* is suitable for a smart parking system. It was also possible to verify the presence of some characteristics inherent to MAS, such as the decentralization of intelligence through the several sectors implemented, each one controlled by an independent spot agent. The scalability, since new parking spots can be easily added or removed on-the-fly into the parking system, and without the degradation of the performance system. And the inclusion of holonic recursive capabilities to simplify the implementation in large-scale environments.

Although the present work has implemented an agent-based CPS for a bicycle park, its structure is adaptable and can be used for other types of vehicles, changing only the access mechanism, e.g, using a gate, latch or sensor, as observed in [35] with the implementation of an agent-based CPS for parking of cars.

The Response time was also adequate for the current application, as observed through

the RTT, time measured from the moment that the spot agent publishes a message in the broker and the physical asset controller gets the notification, as illustrated in Figure 5.8.

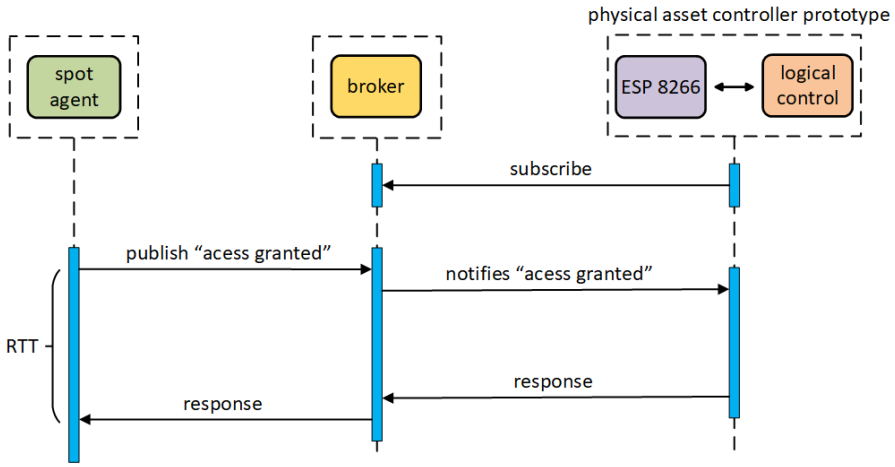

Figure 5.8: Process to measure the response time [35].

The values obtained for the RTT during the system operation can be seen in Figure 5.9. Each value corresponds to an interaction between the spot agent and the physical asset controller.

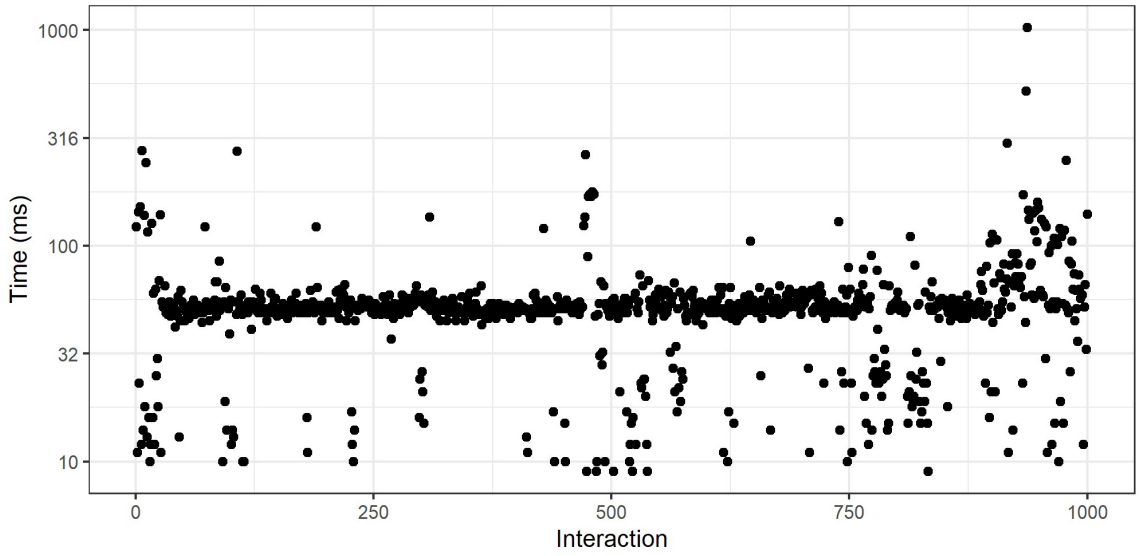

Figure 5.9: Response time for the spot agent to communicate with the physical asset controller [35].

According to the interactions, the response time presented satisfactory and deterministic values, presenting an average value of 51 milliseconds and a standard deviation of 0.04. Only some values showed deviations from the average, but this is justified by the network traffic, a factor that influences the performance of the remote interface practice approaches [34].

Chapter 6

Conclusions and Future Work

A smart parking system is a suitable solution to deal with recurring problems in large cities, such as traffic congestion and air pollution. In this context, the present work developed an agent-based CPS for parking of bicycles, an innovative approach, based on the interconnection between the software agents and the low-level control devices to control the access to the parking spots. Such cyber-physical interaction constitutes a CPS, and requires the use of emergent ICT, IoT and MAS technologies for its implementation.

To develop the agent-based CPS, several interface practices were analyzed and tested, verifying some fundamental parameters for the current application such as response time, technology, scalability and re-usability. The results obtained allowed to determine the best interface practice for a smart parking system, and also to recommend the other interfaces for specific applications. Thus, for applications that need a fast response time, the *On-device/Client-Server* approach is the best option, presenting the lowest RTT values, since the communication is direct and there is no network communication channel. For applications in scalable environments, the *On-device/Publish-Subscribe* presents the best performance among the analyzed interfaces. Although on-device approaches have the best response times and deterministic data acquisition, their use is limited, since many low-level computational platforms are unable to host software agents and remote applications are more widely used. Another important factor in these approaches is the low-level computational platform, which responds proportionally to the processing power,

even using the same communication protocol as observed with the Modbus. Finally, the re-usability is simplified if standard communication protocols are used.

According to the above considerations, it is concluded that there are several ways to interconnect a software agent with low-level control devices, and the application domain is the determining factor in choosing which interface practice should be used. In this case, for a smart parking system where the response time is not the major requirement to be considered, and parameters such as scalability and monitoring play a significant relevant role, the *Remote/Publish-Subscribe* is the interface that best fits this application. Taking this into account, the agent-based CPS parking system was implemented using proper IoT technologies and publish-subscribe model, showing important characteristics for smart parking systems, such as modularity, scalability and reconfigurability. In terms of modularity, the agent-based CPS presents a modular structure, where a parking spot comprises a spot agent and a physical asset controller, being possible to combine several parking spots into a composed parking spot, e.g. a parking sector, following the holonic principles. The scalability and reconfigurability are easily achieved on-the-fly by adding new spot and driver agents without the need to stop, reprogram and restart the entire system.

Future work will be devoted to testing and analyzing other parameters to interconnect software agents with low-level control devices, such as the security that is an important criterion for industrial applications and smart parking systems. Analyze the behavior of the system by increasing the number of agents, verifying the stability of the system. In addition, the agent-based CPS will be implemented in a real environment where drivers can park their bicycles. For this purpose, a model for the latch system that will lock the bicycle in the parking spot is already being developed (see Appendix B), allowing to validate this prototype in real environments.

Bibliography

[1] Y. Geng and C. G. Cassandras, "New "smart parking" system based on resource allocation and reservations", *Proceedings of the IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1129–1139, Sep. 2013.

[2] G. Yan, W. Yang, D. Rawat, and S. Olariu, "SmartParking: A Secure and Intelligent Parking System", *Proceedings of the IEEE Intelligent Transportation Systems Magazine*, vol. 3, pp. 569–574, 2011.

[3] T. Pham, M. Tsai, D. Nguyen, C. Dow, and D. Deng, "A cloud-based smart-parking system based on internet-of-things technologies", *Proceedings of the IEEE Access*, vol. 3, pp. 1581–1591, 2015.

[4] M. ur Rehman and M. A. Shah, "A smart parking system to minimize searching time, fuel consumption and co2 emission", *Proceedings of the 23rd International Conference on Automation and Computing (ICAC'17)*, pp. 1–6, Sep. 2017.

[5] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities", *Proceedings of the IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[6] H. Chourabi, T. Nam, S. Walker, J. R. Gil-Garcia, S. Mellouli, K. Nahon, T. A. Pardo, and H. J. Scholl, "Understanding smart cities: An integrative framework", *Proceedings of the 45th Hawaii International Conference on System Sciences (HICSS'12)*, pp. 2289–2297, Jan. 2012.

[7] N. M. Kumar, S. Goel, and P. K. Mallick, "Smart cities in india: Features, policies, current status, and challenges", *Proceedings of the Technologies for Smart-City Energy Security and Power (ICSESP'18)*, pp. 1–4, Mar. 2018.

[8] M. Naphade, G. Banavar, C. Harrison, J. Paraszczak, and R. Morris, "Smarter cities and their innovation challenges", vol. 44, no. 6, pp. 32–39, Jun. 2011.

[9] G. R. Ceballos and V. M. Larios, "A model to promote citizen driven government in a smart city: Use case at gdl smart city", *Proceedings of the IEEE International Smart Cities Conference (ISC2'16)*, pp. 1–6, Sep. 2016.

[10] europeansmartcities, *The smart city model*, Last accessed 20 May 2019, 2019. [Online]. Available: `http://www.smart-cities.eu/`.

[11] R. Grodi, D. B. Rawat, and F. Rios-Gutierrez, "Smart parking: Parking occupancy monitoring and visualization system for smart cities", *Proceedings of the Southeast-Con 2016*, pp. 1–5, Mar. 2016.

[12] G. Yan, W. Yang, D. B. Rawat, and S. Olariu, "Smartparking: A secure and intelligent parking system", *Proceedings of the IEEE Intelligent Transportation Systems Magazine*, vol. 3, no. 1, pp. 18–30, 2011.

[13] N. Jazdi, "Cyber physical systems in the context of industry 4.0", *Proceedings of the IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR'14)*, pp. 1–4, May 2014.

[14] M. Baygin, H. Yetis, M. Karakose, and E. Akin, "An effect analysis of industry 4.0 to higher education", *Proceedings of the 15th International Conference on Information Technology Based Higher Education and Training (ITHET'16)*, pp. 1–4, Sep. 2016.

[15] J. Jiang, "An improved cyber-physical systems architecture for industry 4.0 smart factories", *Proceedings of the International Conference on Applied System Innovation (ICASI'17)*, pp. 918–920, May 2017.

[16] E. A. Lee, "Cyber physical systems: Design challenges", *Proceedings of the 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'18)*, pp. 363–369, May 2008.

[17] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: The next computing revolution", *Proceedings of the Design Automation Conference*, pp. 731–736, Jun. 2010.

[18] C. K. Keerthi, M. A. Jabbar, and B. Seetharamulu, "Cyber physical systems(cps):security issues, challenges and solutions", *Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research (ICCIC'17)*, pp. 1–4, Dec. 2017.

[19] F. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*. Chichester, England: NJ: John Wiley, 2007.

[20] P. Leitão and S. Karnouskos, *Industrial Agents: Emerging Applications of Software Agents in Industry*. Elsevier, 2015.

[21] B. Gokulan and D. Srinivasan, "An introduction to multi-agent systems", in. Jul. 2010, vol. 310, pp. 1–27.

[22] W. Lepuschitz, "Self-reconfigurable manufacturing control based on ontology-driven automation agents", Technische Universität Wien, PhD Thesis, Apr. 2018.

[23] L. Souza de Castro, G. Vaz Alves, and A. Borges, "Utilização de grau de confiança entre agentes para alocação de vagas em um smart parking", *Proceedings of the 10º Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações*, May 2016.

[24] F. Ducheiko, *Implementação de um sistema multiagente com mecanismo de negociação descentralizado para um estacionamento inteligente.* Work presented to obtain bachelor's degree in computer science, Universidade Tecnológica Federal do Paraná, 2019.

[25] A. Marini, *Implementação e análise de mecanismos de negociação em um sistema multiagente aplicado a alocação de vagas em um estacionamento inteligente.* Work presented to obtain bachelor's degree in computer science, Universidade Tecnológica Federal do Paraná, 2018.

[26] FIPA, *FIPA ACL Message Structure Specification*, Last accessed 23 june 2019, 2019. [Online]. Available: `http://www.fipa.org/specs/fipa00061/SC00061G.html#_Toc26669700`.

[27] S. Lee, H. Kim, D. Hong, and H. Ju, "Correlation analysis of mqtt loss and delay according to qos level", *Proceedings of the International Conference on Information Networking (ICOIN'13)*, pp. 714–717, Jan. 2013.

[28] IBM, *Getting to know MQTT*, Last accessed 23 june 2019, 2017. [Online]. Available: `https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/`.

[29] S. Tamboli, M. Rawale, R. Thoraiet, and S. Agashe, "Implementation of modbus rtu and modbus tcp communication using siemens s7-1200 plc for batch process", *Proceedings of the International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM'15)*, pp. 258–263, May 2015.

[30] NI, *O protocolo Modbus em detalhes*, Last accessed 24 june 2019, 2019. [Online]. Available: `https://www.ni.com/pt-pt/innovations/white-papers/14/the-modbus-protocol-in-depth.html`.

[31] J. Polge, J. Robert, and Y. L. Traon, "Assessing the impact of attacks on opc-ua applications in the industry 4.0 era", *Proceedings of the 16th IEEE Annual Consumer Communications Networking Conference (CCNC'19)*, pp. 1–6, Jan. 2019.

[32] J. Pfrommer, A. Ebner, S. Ravikumar, and B. Karunakaran, "Open source opc ua pubsub over tsn for realtime industrial communication", *Proceedings of the IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA'18)*, vol. 1, pp. 1087–1090, Sep. 2018.

[33] OPCfoundation, *Unified Architecture OPC Unified Architecture Specification*, Last accessed 24 june 2019, 2019. [Online]. Available: `https://opcfoundation.org/developer-tools/specifications-unified-architecture`.

[34] L. Sakurada, J. Barbosa, and P. Leitão, "Deployment of industrial agents in a smart parking system", *Proceedings of the IEEE 17th International Conference on Industrial Informatics (INDIN'19)*, 2019, **Submitted and accepted for publication**.

[35] L. Sakurada, J. Barbosa, P. Leitão, G. Alves, A. Borges, and P. Botelho, "Development of an agent-based cps for a smart parking system", *Proceedings of the 45th Annual Conference of the IEEE Industrial Electronics Society (IECON'19)*, 2019, **Submitted and accepted for publication**.

[36] M. Calabrese, "Hierarchical-granularity holonic modelling", Università Degli Studi di Milano, PhD Thesis, 2010.

[37] R. G. Smith, "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver", *Transactions on Computers*, vol. C-29, no. 12, pp. 1104–1113, 1980.

[38] K. Omote and A. Miyaji, "A practical english auction with one-time registration", in, ser. Lecture Notes in Computer Science, vol. 2119, 2001, pp. 221–234.

[39] T. E. Rockoff and M. Groves, "Design of an internet-based system for remote dutch auctions", *Internet Research*, vol. 5, pp. 10–16, 1995.

[40] B. Alves, G. Alves, P. Leitão, and A. Borges, "Experimentation of negotiation protocols for consensus problems in smart parking systems", *Proceedings of the 9th International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS'19)*, 2019.

[41] P. Leitão, S. Karnouskos, L. Ribeiro, P. Moutis, J. Barbosa, and T. I. Strasser, "Integration patterns for interfacing software agents with industrial automation systems", *Proceedings of the 44th Annual Conference of the IEEE Industrial Electronics Society (IECON'18)*, pp. 2908–2913, Oct. 2018.

[42] P2660.1, *Recommended practices on industrial agents: Integration of software agents and low level automation functions*, Last accessed 08 june 2019. [Online]. Available: `https://standards.ieee.org/develop/project/2660.1.html`.

[43] P. Leitão, S. Karnouskos, L. Ribeiro, P. Moutis, J. Barbosa, and T. I. Strasser, "Common practices for integrating industrial agents and low level automation functions", *Proceedings of the 43th Annual Conference of the IEEE Industrial Electronics Society (IECON'17)*, pp. 6665–6670, Oct. 2017.

[44] S. Karnouskos, R. Sinha, P. Leitão, L. Ribeiro, and T. I. Strasser, "Assessing the integration of software agents and industrial automation systems with iso/iec 25010", *Proceedings of the IEEE 16th International Conference on Industrial Informatics (INDIN'18)*, pp. 61–66, Jul. 2018.

[45] RaspberryPiFoundation, *Raspberry Pi*, Last accessed 30 May 2019, 2019. [Online]. Available: `https://www.raspberrypi.org/`.

[46] JADE, *JAVA Agent DEvelopment Framework is an Open Source Platform for Peer-to-Peer Agent Based Applications*, Last accessed 04 May 2019, 2019. [Online]. Available: `http://jade.tilab.com/`.

[47] iCalendar, *The iCalendar Standard*, Last accessed 20 May 2019, 2019. [Online]. Available: `https://icalendar.org/`.

[48] Mosquitto, *Eclipse Mosquitto An open source MQTT broker*, Last accessed 19 May 2019, 2019. [Online]. Available: `https://mosquitto.org/`.

[49] S. Kadariya and M. Tiwari, *Smart parking*, Report for the Curricular Unit of Project, Instituto Politécnico de Bragança, 2019.

Appendix A

Analysis of the Response Time

This appendix comprises the results of the several tests performed during the experimental evaluation of the interface practices to interconnect software agents and the low-level control devices.

According to Table 4.1, test #1, #2 and #3 aims to verify the response time of each interface practice based on three variables, number of calls, message size and target cycle time.

For test #1, the number of calls was varied and the other variables remained constant. Test #2 follows the same principle, however the target cycle time is defined as "freewheeling". Finally, test #3 varies the target cycle time, and keeps the number of calls and message size constant.

A.1 Test #1

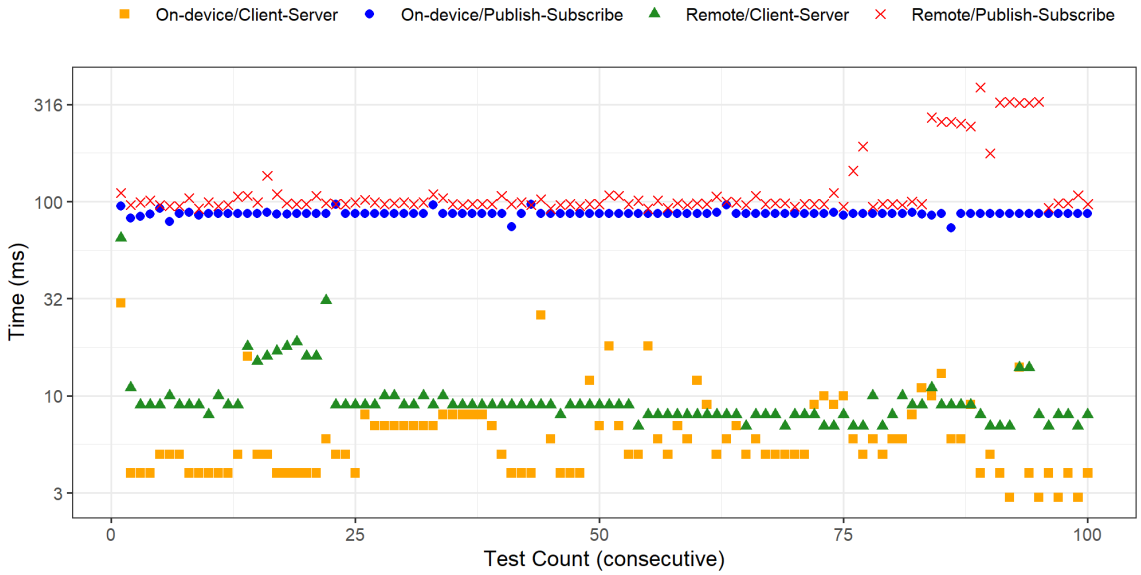

Figure A.1: Number of Calls: 100; Message Size: 1000 bytes; Target Cycle Time: 10 ms.

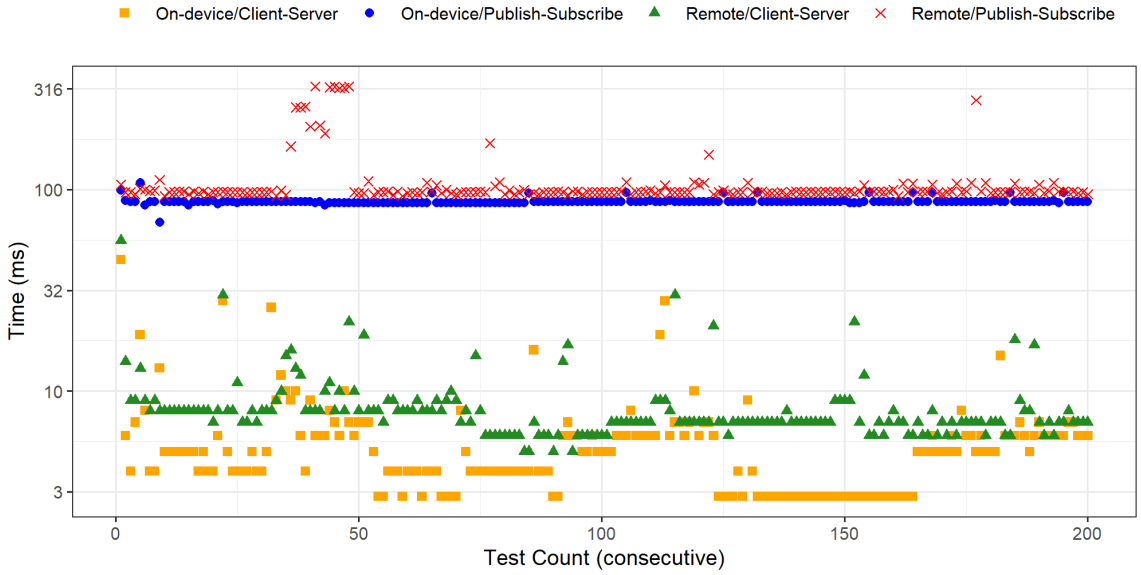

Figure A.2: Number of Calls: 200; Message Size: 1000 bytes; Target Cycle Time: 10 ms.

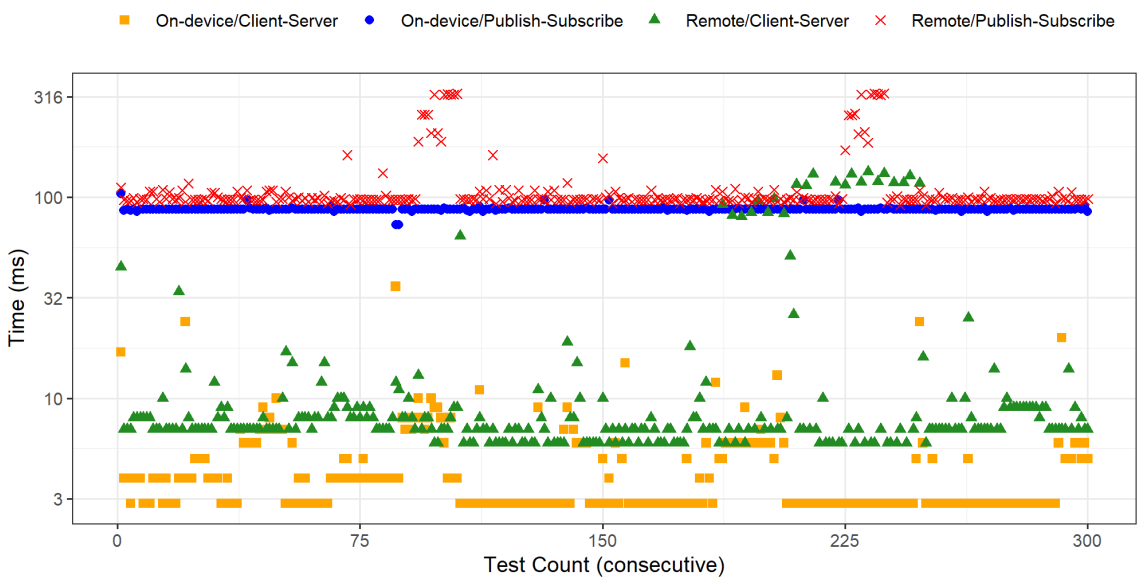

Figure A.3: Number of Calls: 300; Message Size: 1000 bytes; Target Cycle Time: 10 ms.

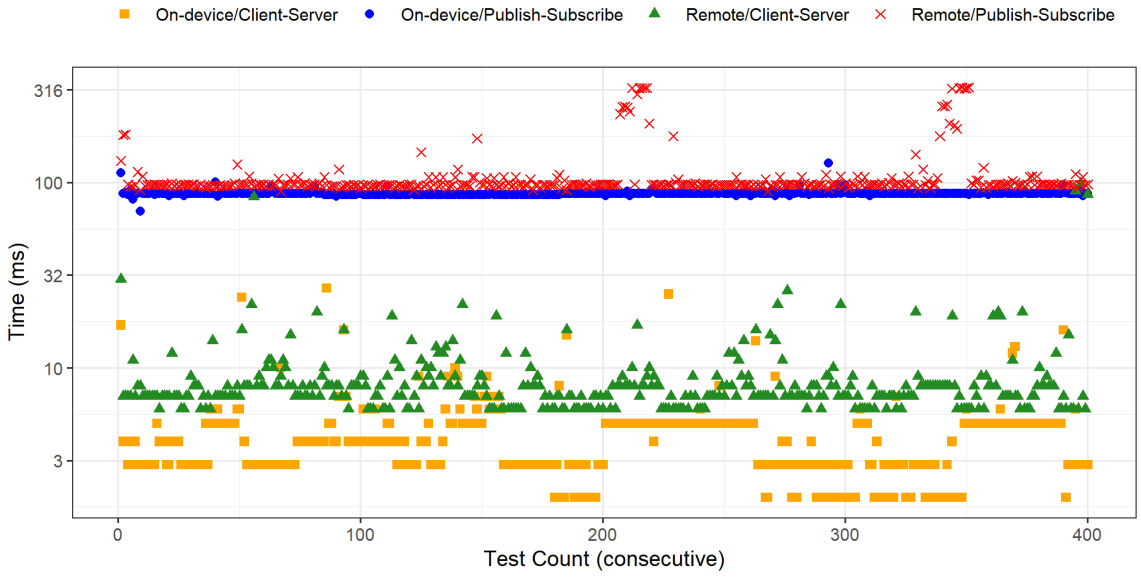

Figure A.4: Number of Calls: 400; Message Size: 1000 bytes; Target Cycle Time: 10 ms.

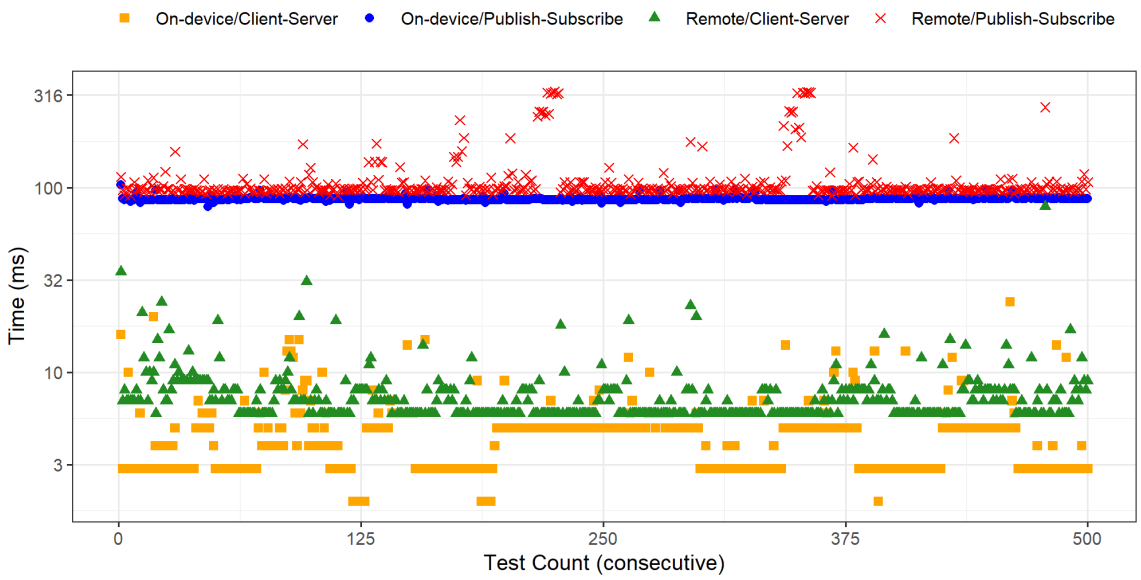

Figure A.5: Number of Calls: 500; Message Size: 1000 bytes; Target Cycle Time: 10 ms.

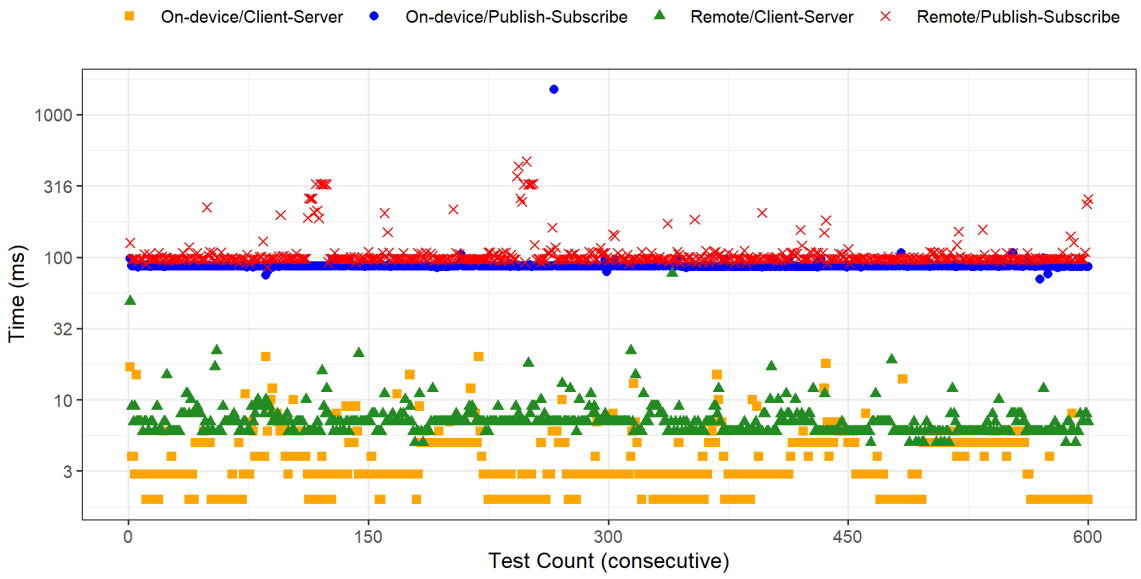

Figure A.6: Number of Calls: 600; Message Size: 1000 bytes; Target Cycle Time: 10 ms.

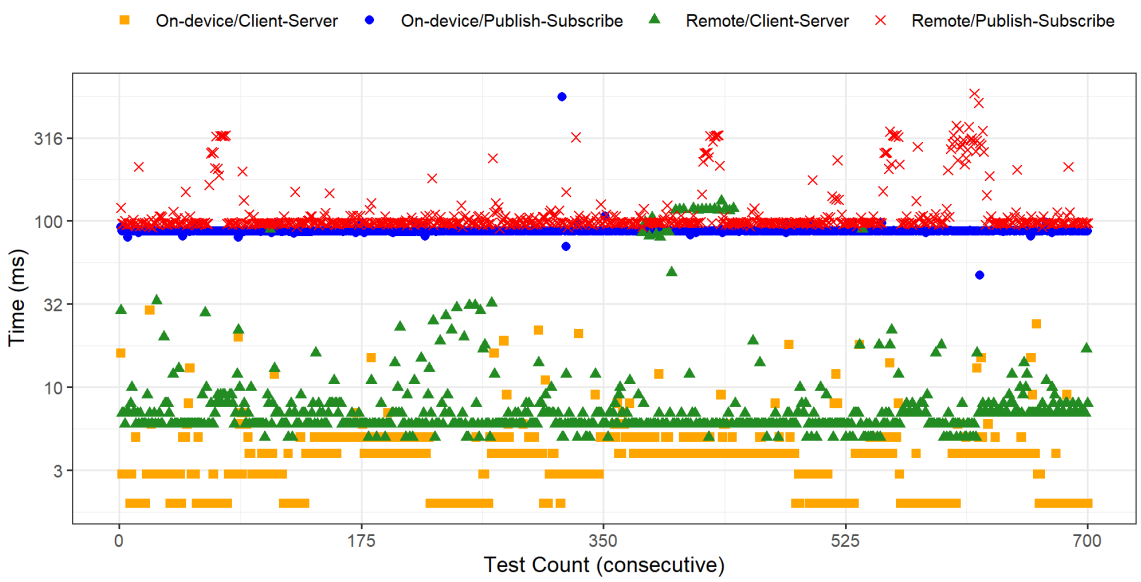

Figure A.7: Number of Calls: 700; Message Size: 1000 bytes; Target Cycle Time: 10 ms.

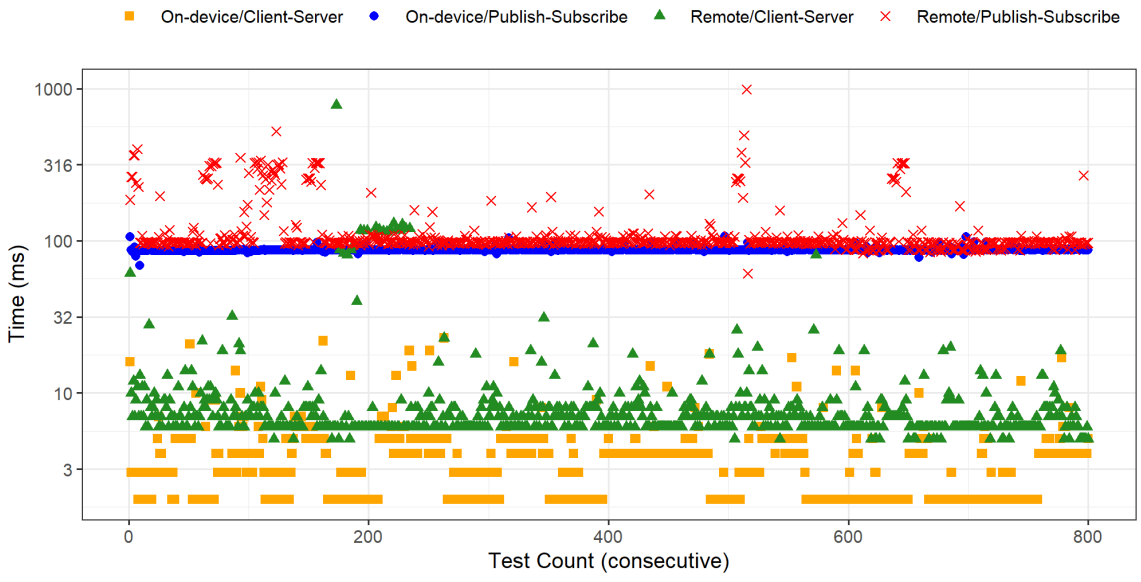

Figure A.8: Number of Calls: 800; Message Size: 1000 bytes; Target Cycle Time: 10 ms.

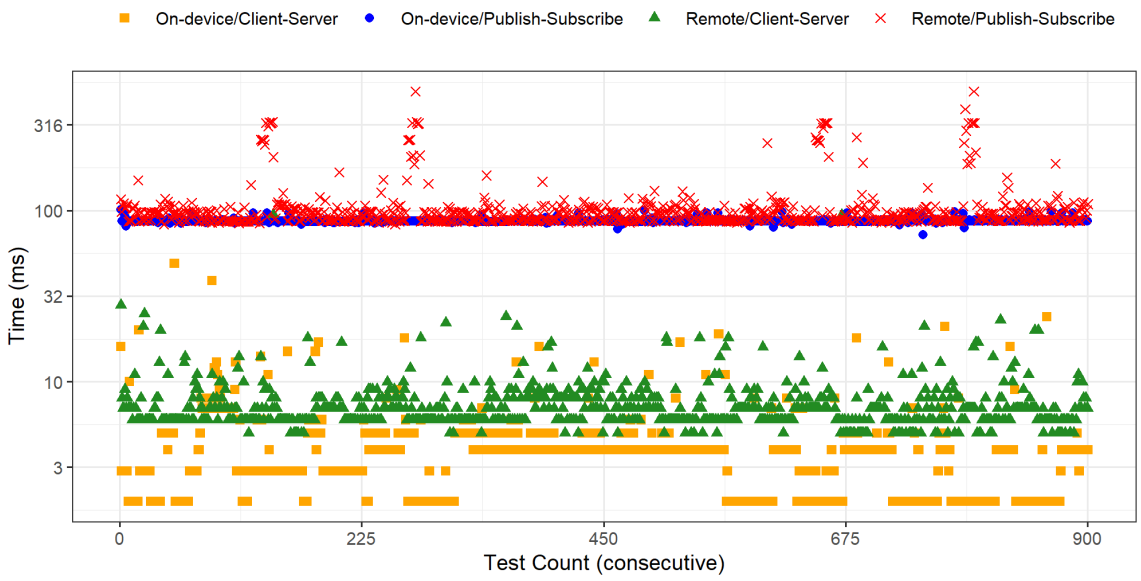

Figure A.9: Number of Calls: 900; Message Size: 1000 bytes; Target Cycle Time: 10 ms.

A.2 Test #2

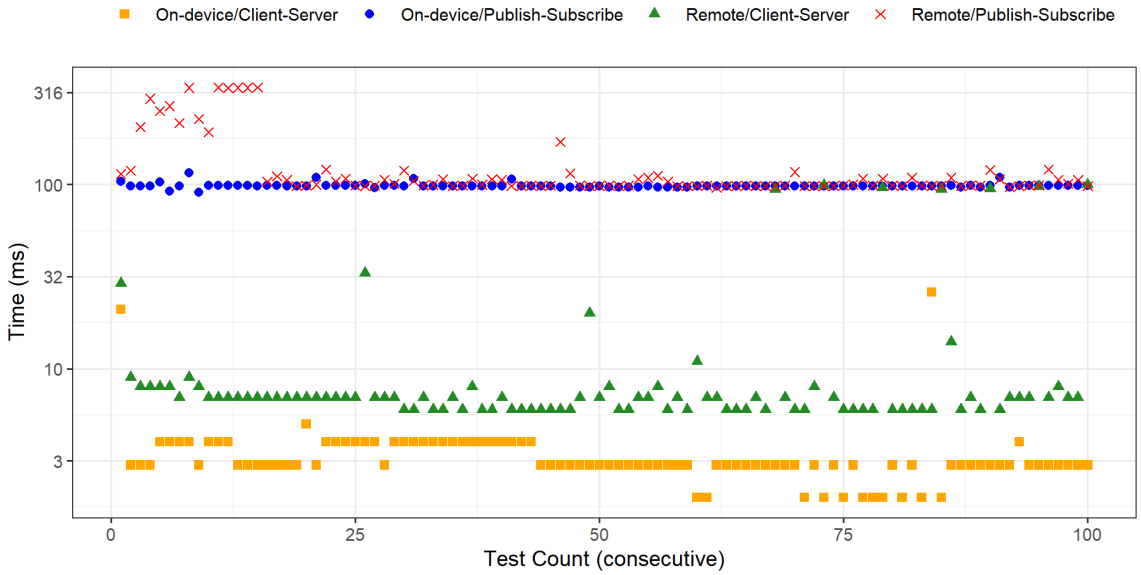

Figure A.10: Number of Calls: 100; Message Size: 1000 bytes; Target Cycle Time: free-wheeling.

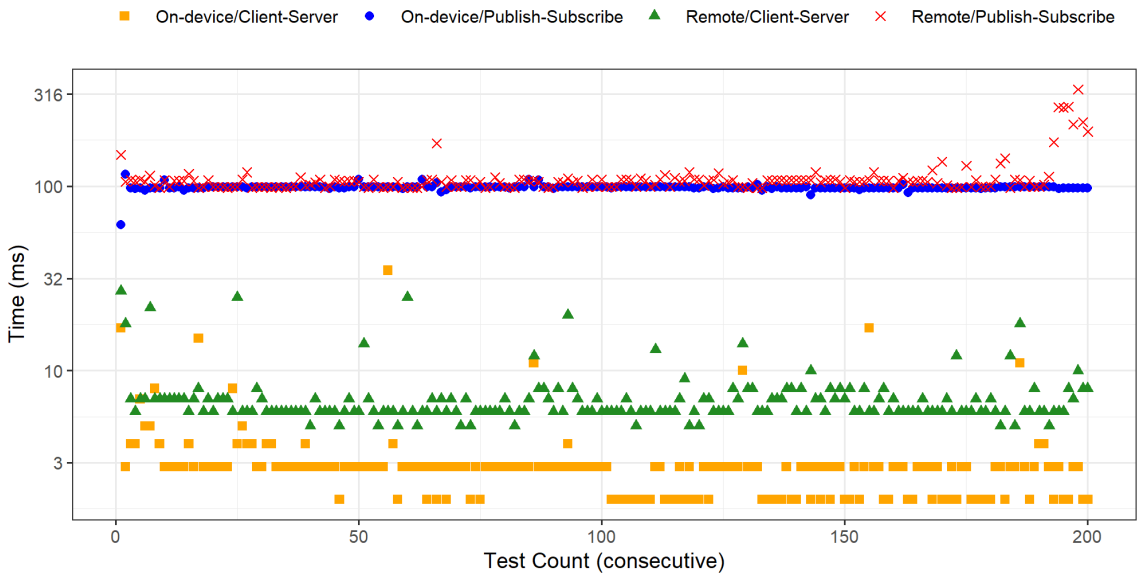

Figure A.11: Number of Calls: 200; Message Size: 1000 bytes; Target Cycle Time: free-wheeling.

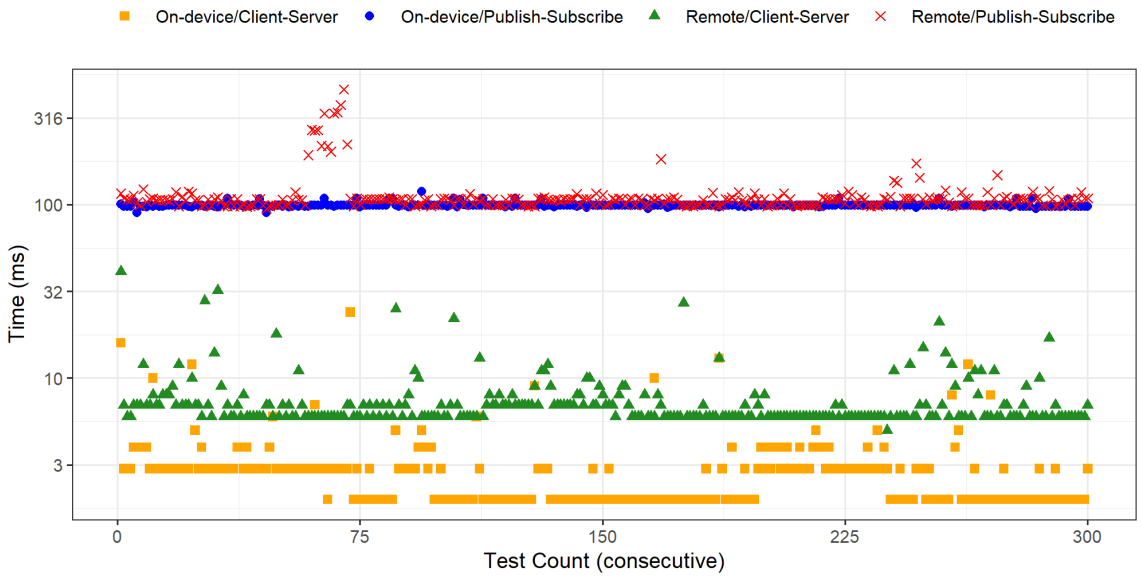

Figure A.12: Number of Calls: 300; Message Size: 1000 bytes; Target Cycle Time: free-wheeling.

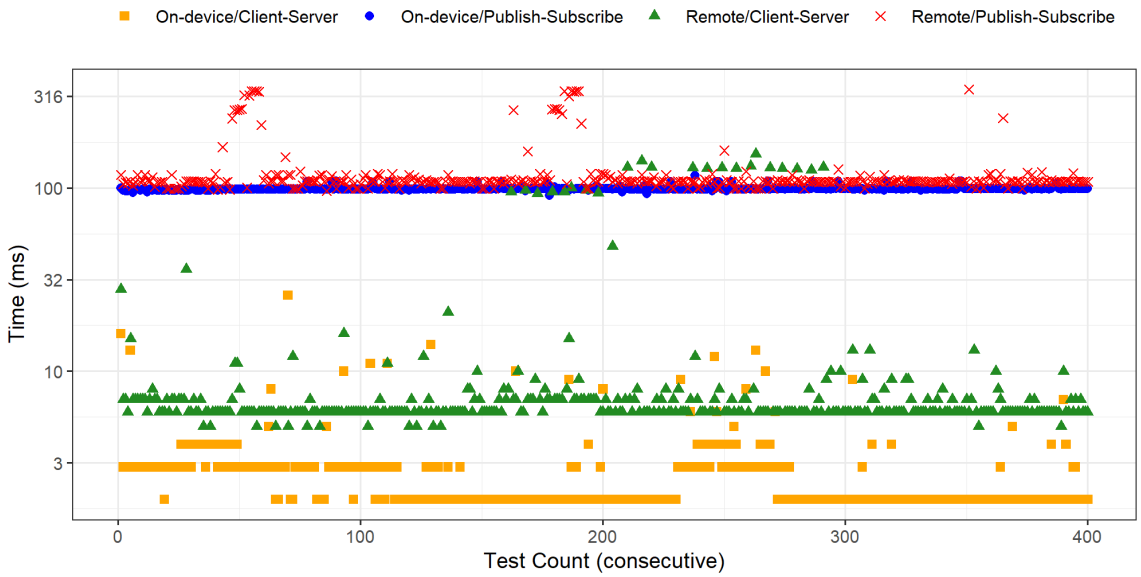

Figure A.13: Number of Calls: 400; Message Size: 1000 bytes; Target Cycle Time: free-wheeling.

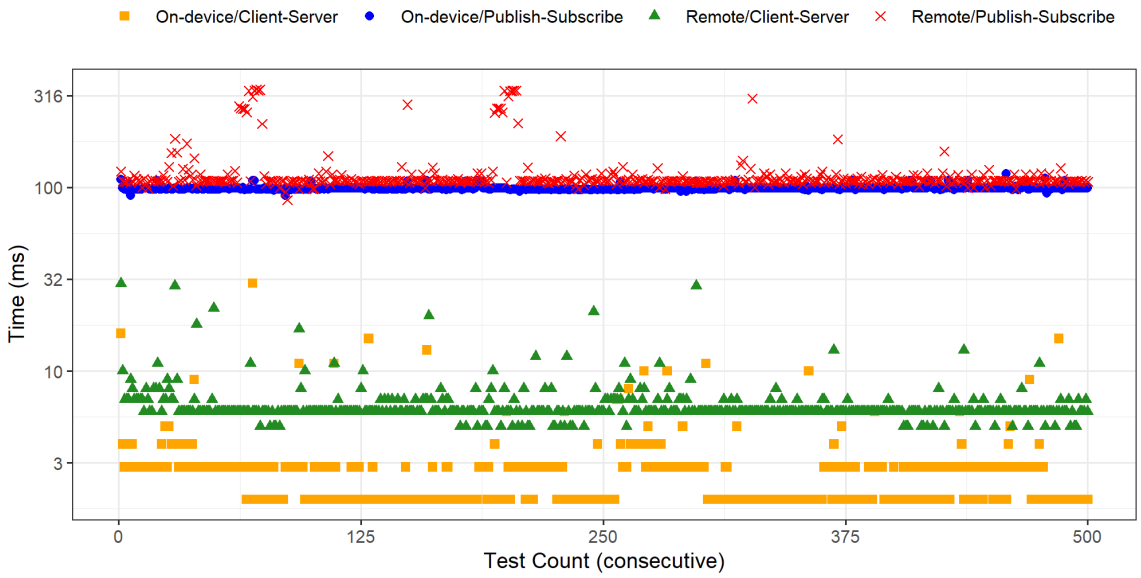

Figure A.14: Number of Calls: 500; Message Size: 1000 bytes; Target Cycle Time: free-wheeling.

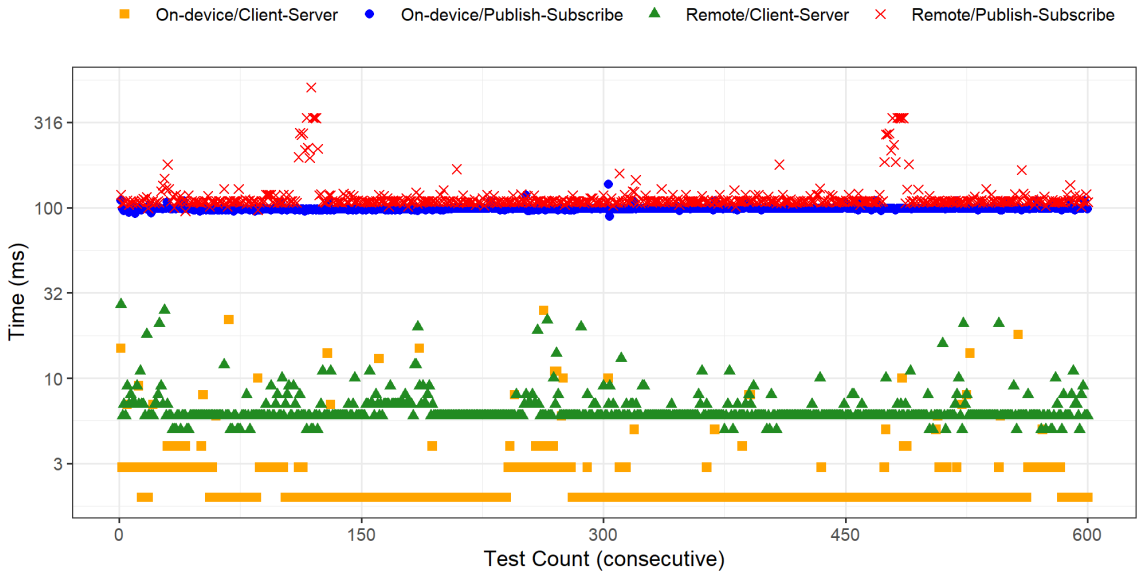

Figure A.15: Number of Calls: 600; Message Size: 1000 bytes; Target Cycle Time: free-wheeling.

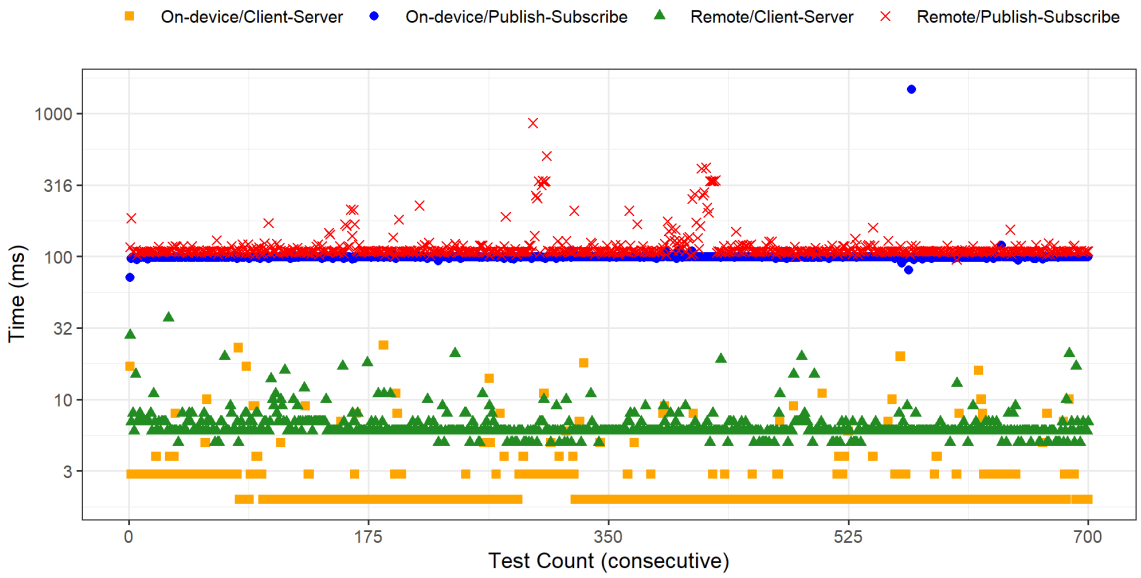

Figure A.16: Number of Calls: 700; Message Size: 1000 bytes; Target Cycle Time: free-wheeling.

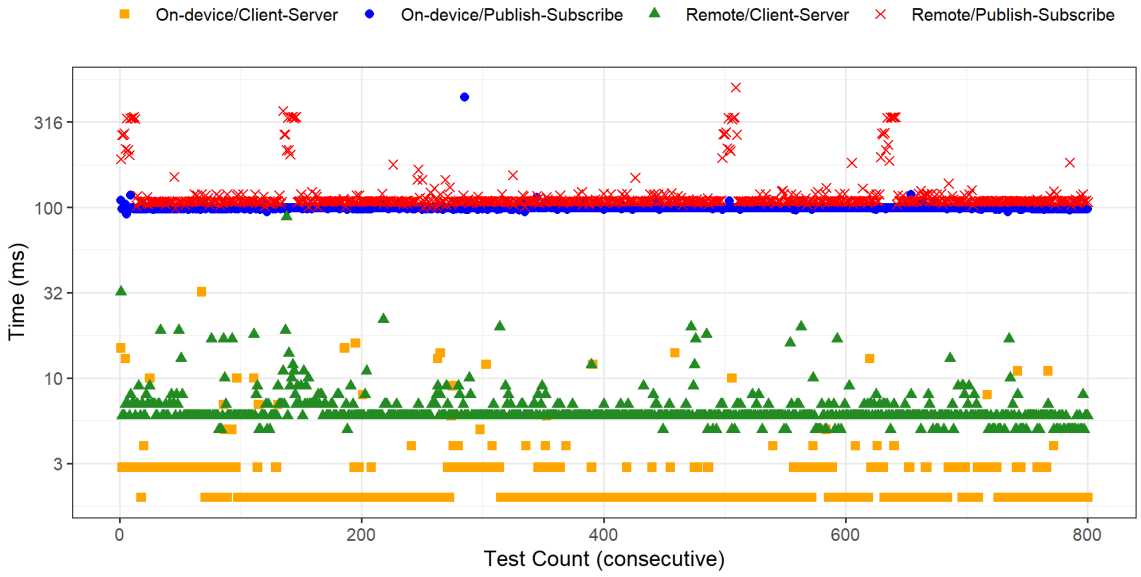

Figure A.17: Number of Calls: 800; Message Size: 1000 bytes; Target Cycle Time: free-wheeling.

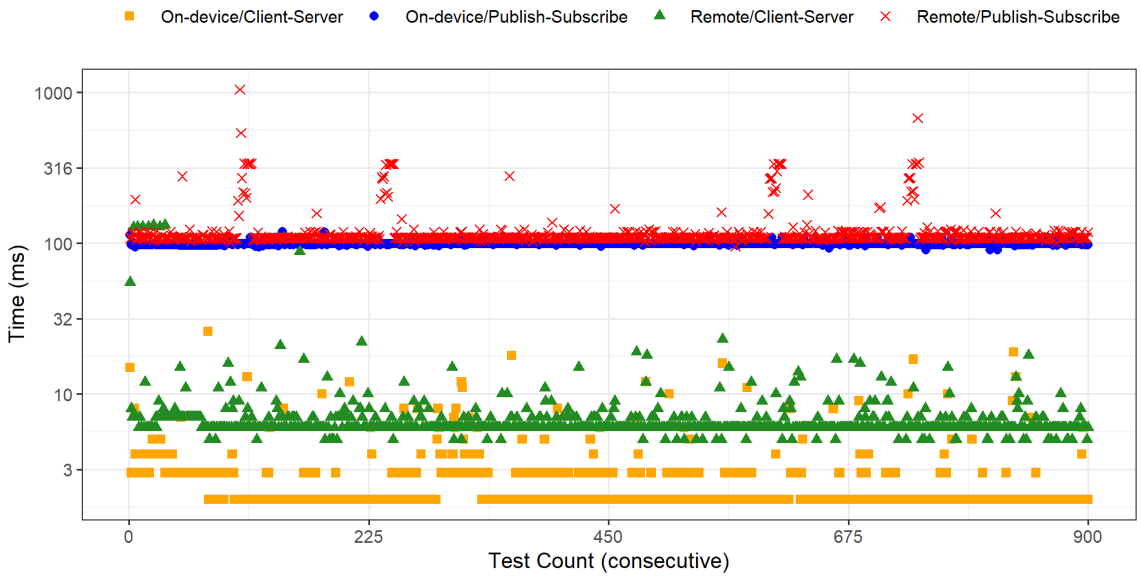

Figure A.18: Number of Calls: 900; Message Size: 1000 bytes; Target Cycle Time: free-wheeling.

A.3 Test #3

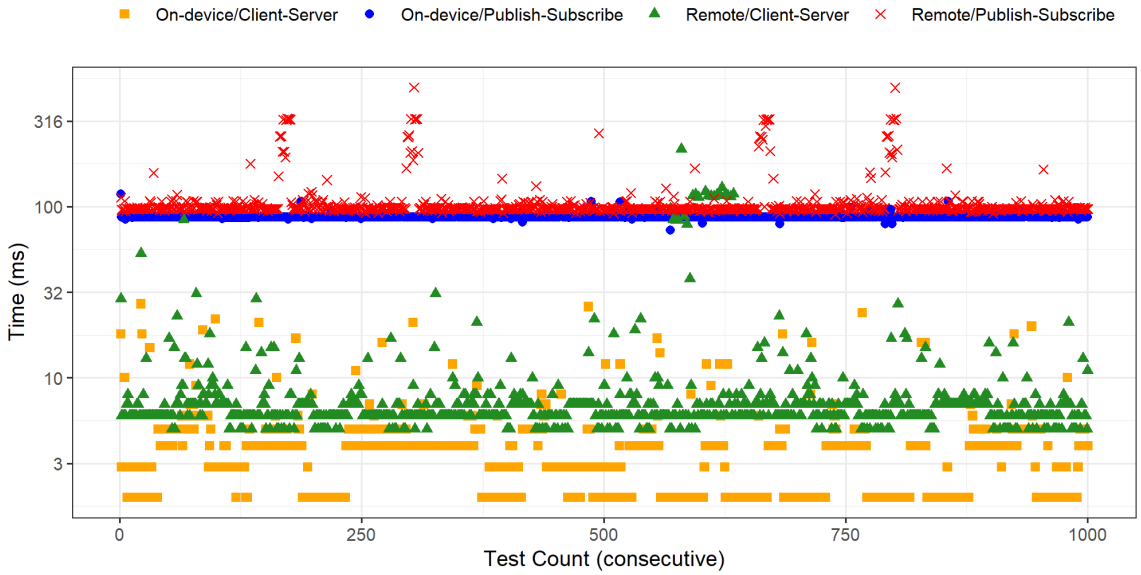

Figure A.19: Number of Calls: 1000; Message Size: 1000 bytes; Target Cycle Time: 10 ms.

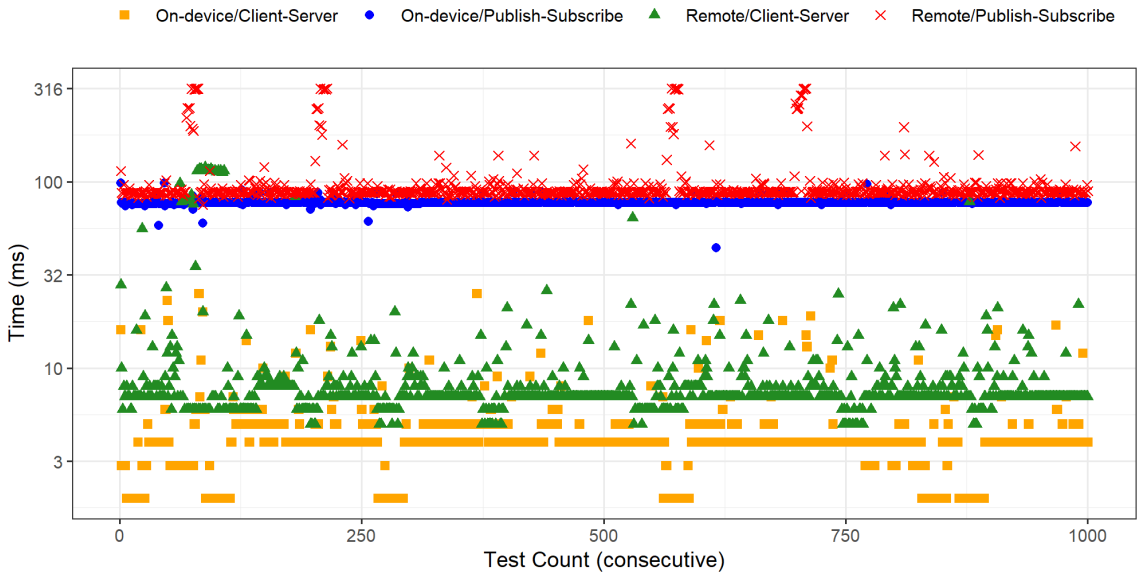

Figure A.20: Number of Calls: 1000; Message Size: 1000 bytes; Target Cycle Time: 20 ms.

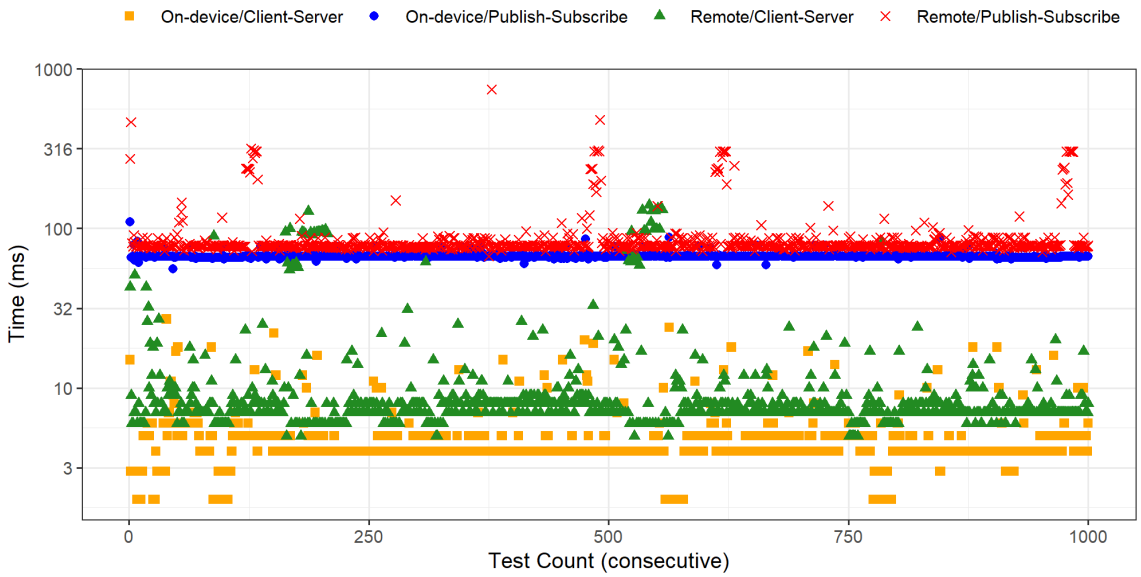

Figure A.21: Number of Calls: 1000; Message Size: 1000 bytes; Target Cycle Time: 30 ms.

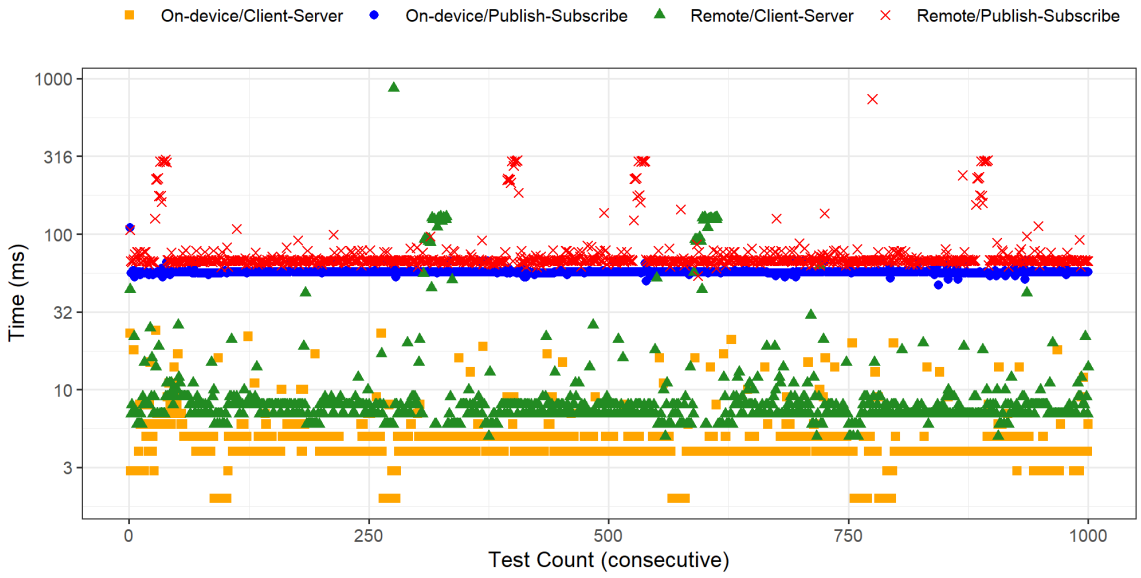

Figure A.22: Number of Calls: 1000; Message Size: 1000 bytes; Target Cycle Time: 40 ms.

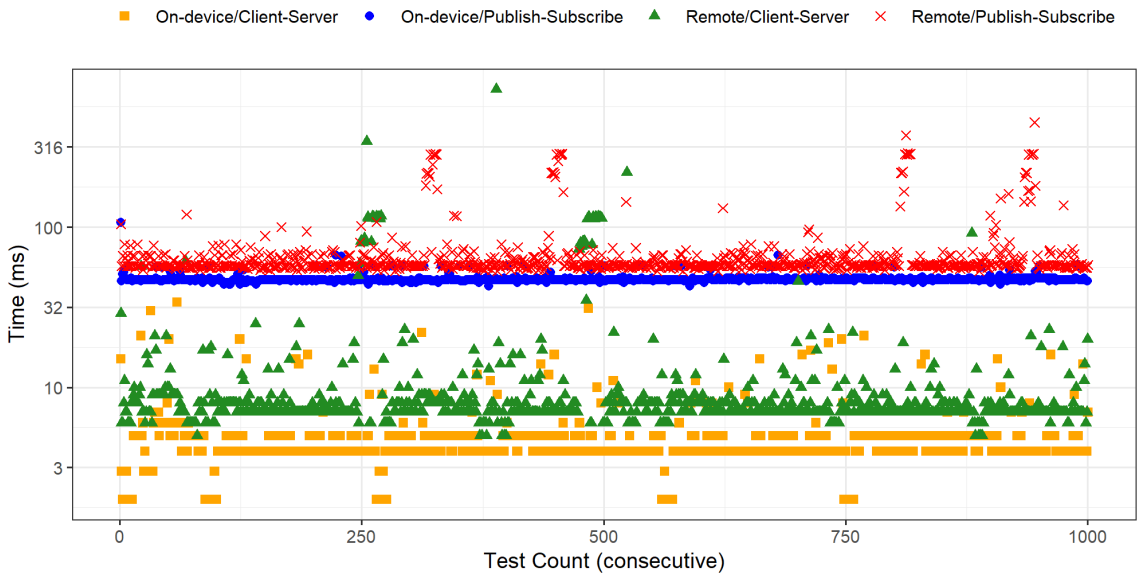

Figure A.23: Number of Calls: 1000; Message Size: 1000 bytes; Target Cycle Time: 50 ms.

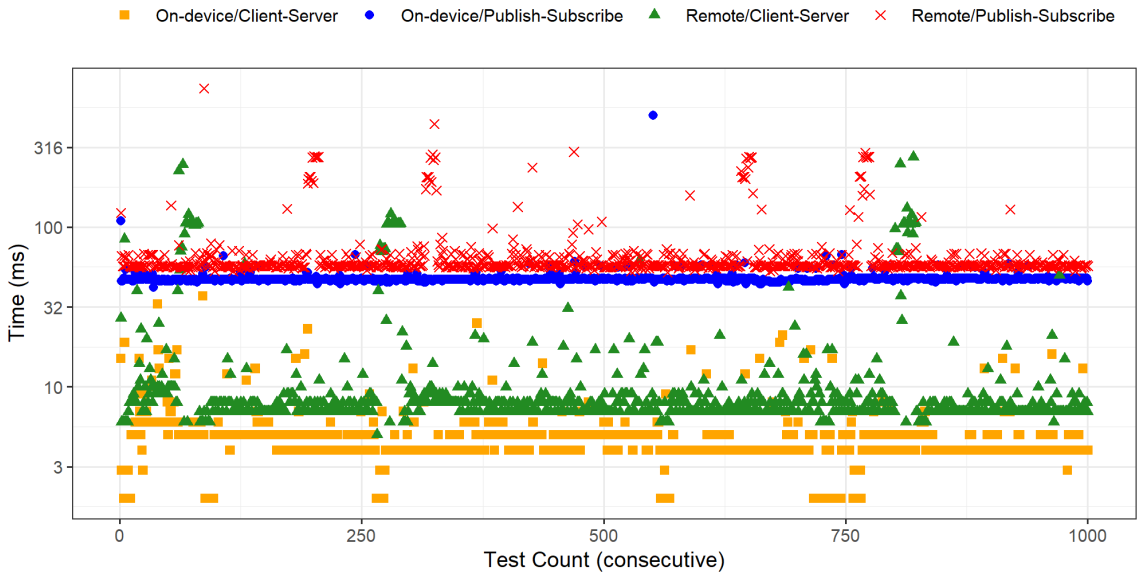

Figure A.24: Number of Calls: 1000; Message Size: 1000 bytes; Target Cycle Time: 60 ms.

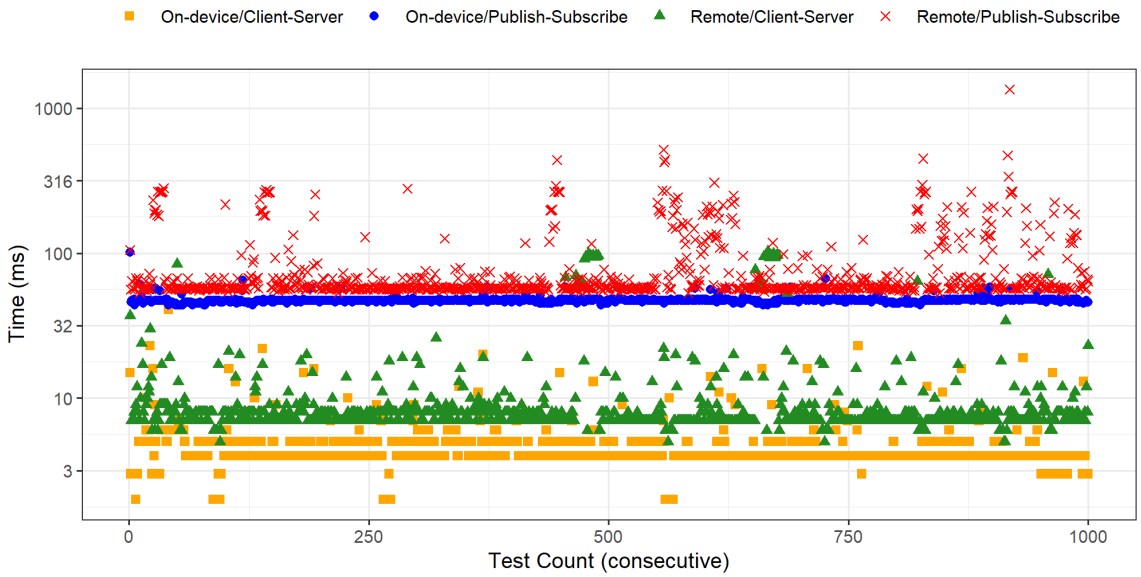

Figure A.25: Number of Calls: 1000; Message Size: 1000 bytes; Target Cycle Time: 70 ms.

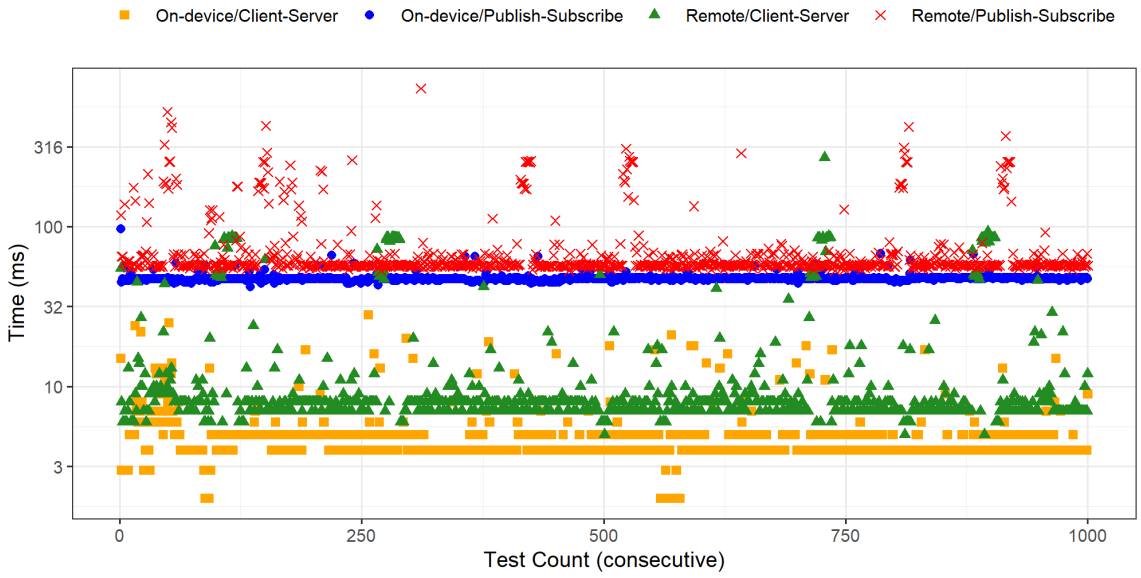

Figure A.26: Number of Calls: 1000; Message Size: 1000 bytes; Target Cycle Time: 80 ms.

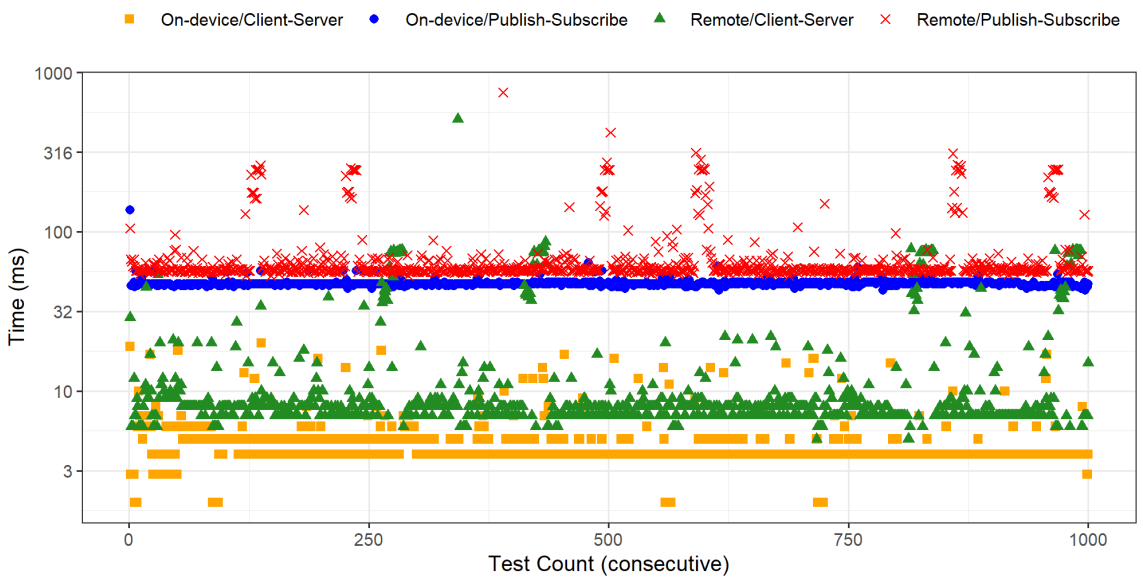

Figure A.27: Number of Calls: 1000; Message Size: 1000 bytes; Target Cycle Time: 90 ms.

Appendix B

Latch System for Parking of Bicycles

This appendix shortly illustrates the developed diagrams of the electrical circuits for the latching system for parking of bicycles.

Figure B.1 shows the latch drive circuit, basically composed of an ESP8266 microcontroller and connections for connecting a solenoid type latch and a 28BYJ-48 stepper motor.

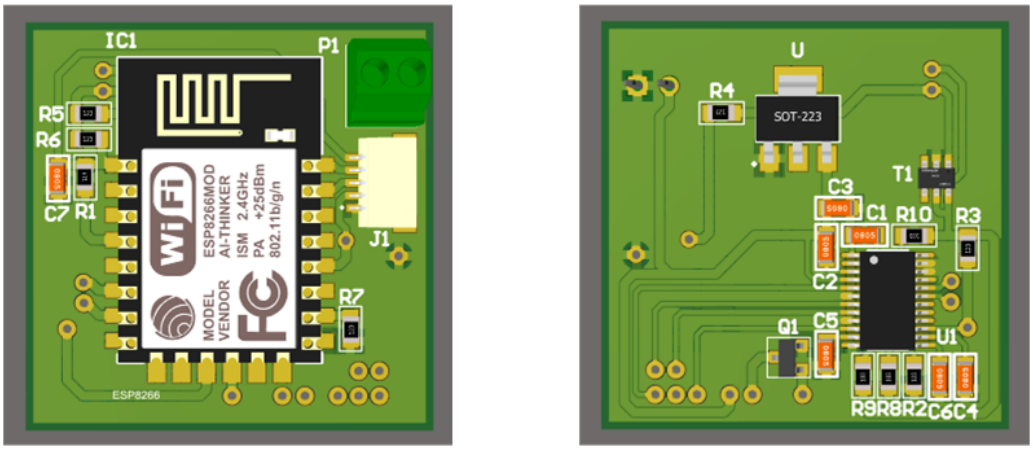

Figure B.1: Electronic circuit for the latch system: top view (left) and bottom view (right).

Figure B.2 shows the sketch developed to place the latch drive circuit.

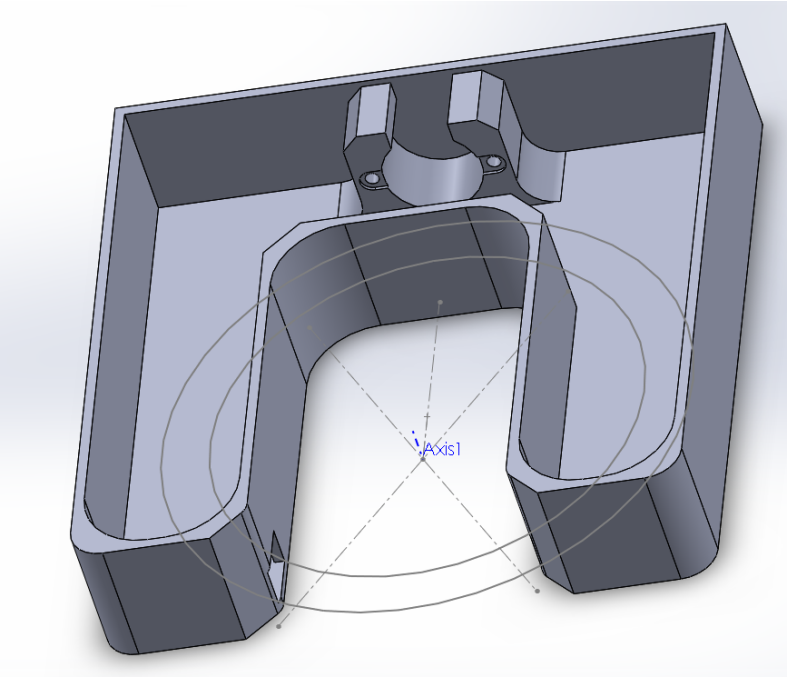

Figure B.2: Sketch for the latch system.