Journal of
Sensor and Actuator Networks

MDPI

Article

IoRT-Based Middleware for Heterogeneous Multi-Robot Systems

Emil Cuadros Zegarra [1,*,†], Dennis Barrios Aranibar [1,*,†] and Yudith Cardinale [1,2,*,†]

[1] Department of Electrical and Electronic Engineering (DEEE), Universidad Católica San Pablo, 04001 Arequipa, Peru
[2] Centro de Estudios en Ciencia de Datos e Inteligencia Artificial (Centro ESenCIA), Universidad Intenacional de Valencia, 46002 Valencia, Spain
* Correspondence: emil.cuadros@ucsp.edu.pe (E.C.Z.); dbarrios@ucsp.edu.pe (D.B.A.); ycardinale@universidadviu.com (Y.C.)
† These authors contributed equally to this work.

Abstract: The concurrence of social robots with different functionalities and cyber-physical systems in indoor environments has recently been increasing in many fields, such as medicine, education, and industry. In such scenarios, the collaboration of such heterogeneous robots demands effective communication for task completion. The concept of the Internet of Robotic Things (IoRT) is introduced as a potential solution, leveraging technologies like Artificial Intelligence, Cloud Computing, and Mesh Networks. This paper proposes an IoRT-based middleware that allows the communication of different types of robot operating systems in dynamic environments, using a cloud-based protocol. This middleware facilitates task assignment, training, and planning for heterogeneous robots, while enabling distributed communication via WiFi. The system operates in two control modes: local and cloud-based, for flexible communication and information distribution. This work highlights the challenges of current communication methods, particularly in ensuring information reach, agility, and handling diverse robots. To demonstrate the middleware suitability and applicability, an implementation of a proof-of-concept is shown in a touristic scenario where several guide robots can collaborate by effectively sharing information gathered from their heterogeneous sensor systems, with the aid of cloud processing or even internal communication processes. Results show that the performance of the middleware allows real-time applications for heterogeneous multi-robot systems in different domains.

Keywords: IoT; heterogeneous robots; social robots; communication protocols; IoRT; multi-robot system; distributed communication; cloud robotics

check for updates

Citation: Cuadros Zegarra, E.; Barrios Aranibar, D.; Cardinale, Y. IoRT-Based Middleware for Heterogeneous Multi-Robot Systems. *J. Sens. Actuator Netw.* **2024**, *13*, 87. https://doi.org/10.3390/jsan13060087

Academic Editor: Lei Shu

Received: 28 October 2024
Revised: 28 November 2024
Accepted: 8 December 2024
Published: 16 December 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the growth of Industry 4.0, the development of robotic solutions is increasingly expanding into various areas of daily human activities, where changes are being introduced to make them more competitive and sustainable [1,2]. While several robots operate in controlled and homogeneous environments, many of the areas where robotic systems are being applied correspond to dynamic environments or situations where they need to collaborate with other different intelligent devices [3,4], such as in airports, museums, domestic environments, and hospitals, demanding high interaction between users and machines or machine-to-machine interaction [5]. This interaction creates a synergy among people, the environment, and machines, with the latter two being blended thanks to the Internet of Things (IoT) and used from a social assistance perspective, such as helping elderly with different needs by connecting domestic service robots with smart environments to provide location-based assistance services [6]; assisting students by integrating robots to help with reading, considering emotions for interaction, and sharing relevant information [7]; and guiding tourists within museums through robots that can relate to users, establishing communication between cyber-physical and virtual environments [8].

These dynamic scenarios increase the complexity of the multiple tasks performed by robots and affects their versatility in executing such as tasks [9,10], as their resolution goes hand in hand with the complexity of the robot itself. Consequently, a solution that is being developed is the use of robots with different characteristics, whose heterogeneity is leveraged collaboratively to cover the weaknesses of one with the strengths of another, generating so-called heterogeneous multi-robot systems [11–13].

In this context, one of the objectives in these multi-robot systems is to establish and integrate communication frameworks between different physical and digital structures as part of machine-to-machine and machine-to-environment interactions [9]. This communication can occur in centralized environments or distributed systems, and sometimes, integrate both. To do so, different aspects need to be considered, such as communication protocols, interoperability methods, information transmission and reception, and modelling and coordination capabilities, which must ensure reaching all robots without information loss and in an agile and fast manner [12]. For this reason, technologies are being investigated for integration into these systems, offering more relevant advantages in accessing resources and data, considering the services provided by IoT, which has led to a new paradigm called the Internet of Robotic Things (IoRT) [14].

New developments in smart connectivity allow robotic devices to connect anytime, anywhere, and with anything and anyone through different networks and services. The purpose of IoRT is to combine robotic systems with IoT, intelligent connectivity, Cloud Computing, Artificial Intelligence, Digital Twins, Virtual/Augmented Reality, and swarm technologies [15–17]. These technologies allow smart objects to interact in a unique way and communicate with each other through the Internet and through other connectivity protocols [18], considering that the IoRT paradigm provides smart devices with sensing, actuation, processing, information acquisition, movement capabilities, and communication, which allow them to interact and collaborate in a distributed manner [19]. As a consequence, the interaction of multiple robots requires a system that can not only monitor the status of each robot, but also manage to collect information from them and determine the best options to communicate and act for each of the agents, and also derive the decisions made integrally, even in separate environments and taking advantage of the unique or different properties that different robots possess.

Within the development of solutions that have been worked on, several robotic systems present flexibility in allowing for interoperability, interaction, and collaboration among heterogeneous robots working in the same line of operation or in specific domains, based on standardization of modules [20,21] or by integrating interfaces that allow the translation of instructions that are uniform and lead to the understanding of the robots [22]. Unfortunately, many of these solutions are tied to the same framework, such as the Robot Operating System (ROS), or limit themselves to the hardware available in the robots with which they work, generating discrepancies in the perception of the environment or decreasing robots' processing capacities [23].

To overcome the limitations of existing solutions, in this paper, a communication middleware is proposed based in a cloud communication protocol that manages to integrate multiple robots with different characteristics, taking advantage of the resources provided by the cloud to streamline the task assignment, training, and planning processes. Additionally, the middleware offers communication via WiFi connection in order to comply with distributed communication for all the devices (e.g., PC, robots, smart phones) that represent each node of the connection and to allow the transmission and reception of information from these devices. The middleware includes two control modes: the local mode, directed by the information that the robots have, and the remote mode from a server in the cloud, which distributes the information to the robots and allows flexible and adaptable communication for cases in which remote connection is not possible. The information collected by the devices is taken to a cloud server and each device is able to request the data to be processed or also ask the server for actions to be carried out based on the data already processed in the cloud.

To demonstrate the suitability and functionality of the proposed middleware, we develop a proof-of-concept for an application case in tourist spaces, supported by various guide robots that collaborate by effectively sharing information gathered from their heterogeneous sensor systems which is sent to the middleware, showing that the middleware allows for the communication and control of cloud resources for the assignment of tasks, training, processing, planning, and decision making, among multiple robots with heterogeneous characteristics, achieving effective communication with low latency. Thus, the performance of the middleware allows real-time applications for heterogeneous multi-robot systems in different domains.

In summary, the contribution of this work is three-fold:

- Demonstrate the benefits of combining IRoT, Cloud Robotics, and Mesh Communication protocols for the implementation of flexible and efficient heterogeneous multi-robot systems.
- Propose a middleware for multi-robots systems that allows heterogeneous robots located in different domains communicate and share information.
- Develop of a proof-of-concept based on a tourism environment populated by heterogeneous robots.

The remainder of this paper is organized as follows. Section 2 shows the different environments in which our proposed middleware is involved. In Section 3, we describe some works that are related to the development of this paper. In Section 4, the characteristics of the proposed middleware and its implementation are indicated, while in Section 5, we present the implementation of the middleware in an application case, as well as the tests carried out and their results. General discussions are presented in Section 6. Finally, in Section 7 the conclusions of this paper are mentioned.

2. Multi-Robot Systems: Preliminaries

Multi-robot systems imply a concurrence of robotic agents, which can be of different sizes, shapes, and capabilities—e.g., Unmanned Aerial Vehicles (UAVs), Unmanned Ground Vehicles (UGVs), humanoids, etc.; thus, to allow for greater flexibility, reliability, versatility, scalability, and resilience to failures, they must be designed to cooperate with each other and with people to successfully execute their tasks [11,24]. The tasks assigned to multi-robot systems are based on the requirement of a level of cooperation to complete tasks, ranging from lightly coordinated tasks, such as subtasks of the exploration and mapping of large locations or surveillance, to closely coordinated tasks that demand significant interaction among robots during activities, such as robot soccer, object transportation, and construction [25].

To manage the execution of cooperative tasks in multi-robot systems, the workflow involves breaking down complex tasks into simple subtasks based on the robots' capabilities, which are grouped accordingly, to then perform the scheduling, planning, and control of tasks among the robot groups, which autonomously carry out the assigned tasks [26], as shown in Figure 1.

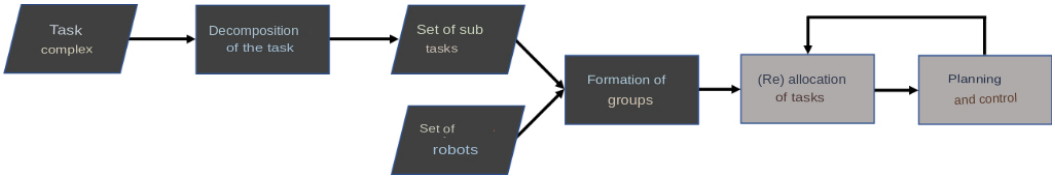


Figure 1. Workflow for task assignment in multi-robot systems [26].

2.1. Internet of Robotic Things

Taking into account that several robotic systems request scalable services for the computational processes they execute, these systems are can be integrated into a Cloud Computing infrastructure, amounting to what is known as Cloud Robotics [27], which combined with IoT, gives rise to IoRT. Thus, IoRT can be conceptualized as an integration

of the capabilities of intelligent environments with autonomous agents to implement applications and operations that monitor and control the states and processes of robots, assisting in their communication and optimizing their intelligence to handle complex and challenging situations within intelligent environments.

Figure 2 illustrates the concept of IoRT as an integration of IoT, robotics, and Cloud Robotics to support heterogeneous multi-robot systems [14,28]. Communication among heterogeneous robots maximizes its applicability, but a dynamic communication system is needed to allow both the sending and receiving of different types of information quickly and with tolerance for losses. This is covered by Cloud Robotics technologies that provide different communication protocols and carry out several of the processes and decisions of the involved robots.

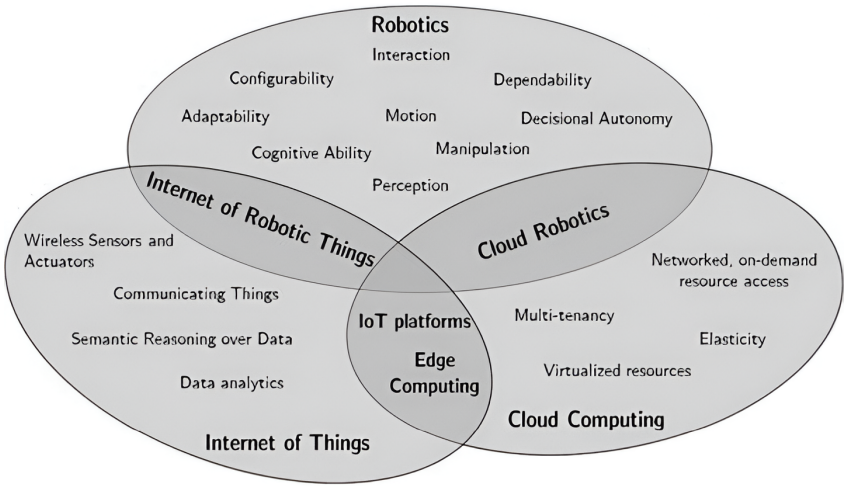


Figure 2. IoRT = IoT + Cloud Robotics [14].

Cloud Robotics overcomes classical robot computing approaches, as shown in Table 1 [29]. Cloud-based robotic infrastructure comprises a set of robot clusters or dynamic set of neighboring robots that work cooperatively to accomplish complex tasks. As a robot could work from one environment to another, it can seamlessly join a different cluster of robots and can still benefit from cloud resources. A robot can communicate with its peers using messages transmitted over a local wireless network interface. The link between any two peer robots might be direct or indirect. If two peer robots can directly send and receive messages to and from each other, then there is a direct link between them. Otherwise, intermediary peers are required to forward transmitted messages, leading to an indirect link between the two peer robots. In this case, the intermediary peers act as routers [30].

Table 1. Different types of robot computing approaches.

Category	Computing Model	Resource Level	Latency Mainly Caused by	Intelligence Level	Application
Stand-alone robots	On-board computing	Weak	Computing workload	Low	Static and structured tasks
Networked robotic system	Ad hoc cloud computing	Medium	Computing and communication workloads	Medium	Real-time processing
Cloud robotic system	Hybrid computing	Strong	Communication workload	High	Both real-time processing and resource applications

2.2. Communication in Heterogeneous Multi-Robot Systems

Heterogeneity and efficiency are just two of the key issues that are relevant when choosing or developing the integration of multi-robot systems [31]. Given that the environments where robots and different environmental sensors operate are highly interconnected and distributed, the need to integrate robotic systems at both the software and hardware levels must be considered; that is, each functional part becomes an active and even independent member in the system [32]. Moreover, if the robotic systems work under different communication architectures, they might not be capable to establish a connection between other systems due to the different protocols and subsystems with which they work, which generates an incompatibility problem.

Hence, in heterogeneous multi-robot systems, the development of a middleware arises to alleviate the complexity of communication and control. A common alternative is to use the "publication and subscription" model, in which the exchange of labeled information by topics takes place, allowing multiple connectivity between different devices [33], as shown in Figure 3.

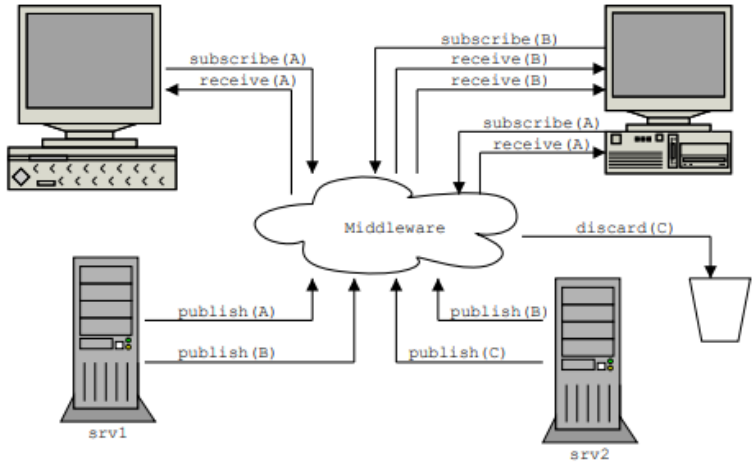


Figure 3. Publisher/subscriber model [34].

3. Related Work

In order to allow message transmission and communication in multi-robot systems, some initiatives implement a mesh topology, using a communication protocol that provides monitoring and search within the mesh of devices connected to an ROS [23,35–37], with some of them based on other technologies, such as Bluetooth Low-Energy (BLE) technology [38]. However, the results of such studies indicate that it is always necessary to consider both the number of robots involved and the characteristics that can cause each robot to have a slightly different perception of the environment compared to other robots [35], affecting the performance of the multi-robot system. Likewise, there are variants of ROSs, not all of which are designed with characteristics and algorithms necessary for various robots [36], and although there are current versions that allow greater scalability in communication between robots [37], these versions focus mostly on a single work domain, and even have increased time costs when it comes to meeting the standards of their Data Distribution Service (DDS) protocol [23].

There are also platforms that are able to manage communications between devices through dedicated systems, enabling the interconnection of control agents, the interface, and recognition and localization functions [12]. Other studies involve data transmission in a mesh using a communication protocol based on time-scheduled frame modes, allowing faster data transmission among robotic devices [39]. One piece of research has managed to implement a central system of communication, in which devices are registered to interpret the information that is sent from different robots, after which said system designates useful orders for astronauts within the International Space Station [21]. Other initiatives have

incorporated additional technologies, such as ontologies, to implement a subscription system to manage high-level instructions for robot control [22]. However, failures in synchronization and compatibility for the interactions among heterogeneous robotic systems may occur, which suggest the need for a middleware that can overcome these limitations, as our proposed middleware would be able to do.

One recent study, presented by Dey et al. [40], proposes a middleware similar to our proposal that is agnostic to specific robotics and network simulators, uses the publish–subscribe transport protocol of DDS, and is based on the ROS 2 architecture (although it can be integrated with ROS 1), relying on its package auto-discovery mechanism. However, it is not compatible with other systems that do not belong to the ROS, unlike our proposal, which is intended to work with any type of robot system and architecture.

All these studies demonstrate the need for and efforts towards a communication platform allowing interconnection among multi-robots. However, most of the existing solutions are limited to a single work domain, to homogeneous robots, or to a unique operating system. Our proposed middleware overcomes all these limitations. A comparative analysis of the referenced studies is shown in Table 2.

Table 2. Comparative analysis of related studies.

Reference	Objective	Communication	Heterogeneity Level
Casals et al., 2018 [12]	Control and management of heterogeneous devices for healthcare environments	Through the ROS but does not exchange information between devices, only a central communication system	Heterogeneous robots and devices with ROS
Sewtz et al., 2024 [21]	Control different robots remotely, simplifying their communication	Implementation of a communication model based on identifiers targeting a central system	Heterogeneous robots for International Space Station
Rajapaksha et al., 2021 [22]	Middleware that allows connectivity and control of heterogeneous robots	Publication/subscription model and ontologies. Instructions are assigned to each robot separately	Heterogeneous robots
Shi et al., 2021 [23]	Propose a communication system based on the hierarchical needs of drones	Communication based on ROS nodes and the DDS protocol	Aerial drones with similar characteristics
Yan et al., 2017 [35]	Achieve communication between heterogeneous robots for collaboration in navigation	The ROS system is the communication middleware	Applicable for collaboration between aerial and terrestrial robots
Macenski et al., 2022 [36]	Analyze the use of ROSs in multi-robot environments	Based on ROS2 and the DDS communication protocol	Support homogeneous robots
Gao et al., 2020 [37]	Collaborative communication between robots and other devices	Communication scope between ROS1 and ROS2	Heterogeneous robots
Ferranti et al., 2019 [38]	Heterogeneous communication system for robots with communication limitations	Communication via Wi-Fi, Cellular, satellite, and BLE	Drones with same equipment
Loncar et al., 2019 [39]	Network of sensors distributed in a swarm of heterogeneous robots	Finite state machine as a communication model	Interaction between two types of robots
Dey et al., 2023 [40]	Robot-agnostic middleware for speed control and transmission	Based on ROS1 and ROS2	Heterogeneous robots with ROS system

4. A Middleware for Heterogeneous Multi-Robot Systems: Our Proposal

Our solution for heterogeneous multi-robot systems is a middleware based on a generalized publish–subscribe communication model that can be adapted according to the robot systems involved, with a vision of Cloud Robotics in mind, as shown in Figure 4.

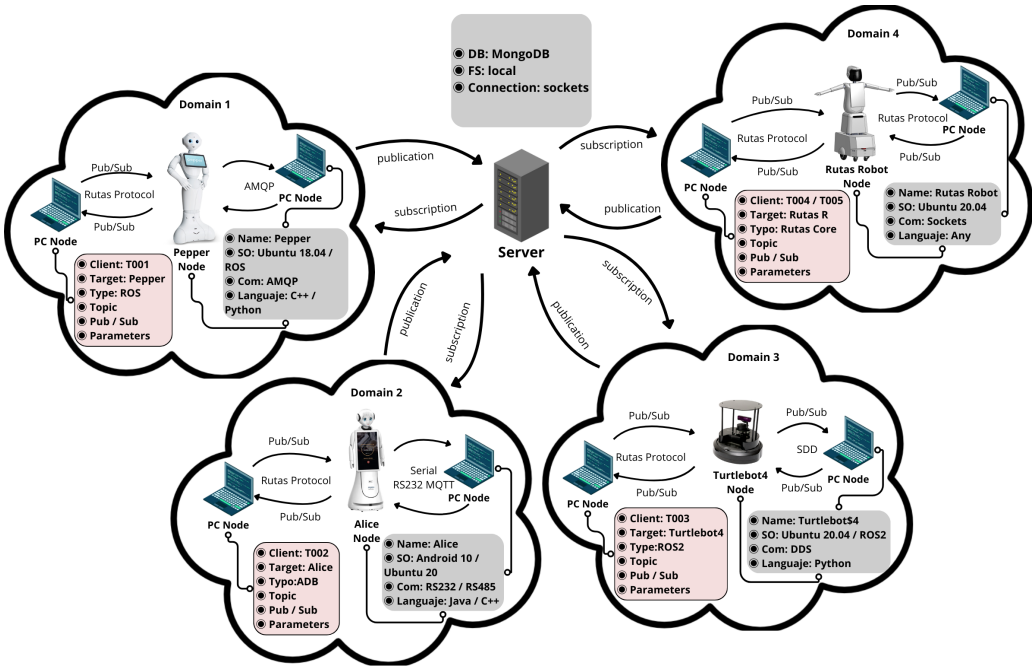


Figure 4. Proposed middleware for communication in heterogeneous multi-robot systems.

The proposed middleware is inspired by several characteristics of various IoRT communication models, described in Table 3. Like the AMQP communication model, the proposed middleware implements a client/server-based structure as an intermediary *broker* for data distribution. Following the trend of the MQTT model, which uses QoS policies, the proposed middleware supports point-to-point connectivity from one robot to several devices or from several devices to other devices. However, when the middleware does not have an auto-discovery capability when it comes to the topics that exist, it provides a message identifier by topic for the correct and direct delivery of messages in terms of their requests, as used by the DDS model, while supporting multiple publication and subscription domains.

Figure 4 describes the system architecture considered by the middleware. Each robotic system is called a *Domain*, which is composed of at least one *Node* which sends or receives data both within and outside the *Domain* and can be of two types: (i) *Robot Node*, which is characterized by being the robotic unit that interacts with the environment, and (ii) *PC Node*, which refers to any device (control unit, workstation, user interface, embedded controller, and others) that can transmit data via a wireless connection to the robot or other devices in the robotic system. The *Nodes* can have different characteristics, but they fulfill communication services with the other *Nodes* within the *Domain*. To achieve communication among different *Domains*, a *Cloud Server* provides connections, applying the same publication and subscription communication model that is used in the *Nodes*; that is, it receives and delivers data from *Nodes* that send or request information. Categorical data, such as topics and information, are stored in the Local File System of the *Nodes* and in remote repositories, such as MongoDB in the *Cloud Server*, in order to have a local control mode and remote control mode.

Table 3. Analysis of communication protocols in IoRT.

Feature	MQTT	AMQP	DDS
Architecture	Client/Broker	Client/Broker Client/Server	Without intermediary
Transport protocol	TCP (WebScoket)	TCP, SCTP (WebSocket)	UDP, TCP
Security	TLS/SSL	TLS/SSL, IPSec, SASL	DDS plugins
Connectivity	One to one, one to many, many to many	Point to point	P2P, one to one, one to many, many to many, many to one
Bandwidth consumption	Depends on number of packages	Low	High
Size header	2 Bytes	8 Bytes	Not defined
QoS	QoS 0 QoS 1 QoS 2	Settle Format (similar to At Most once) or Unsettle Format (similar to At Least once)	23 QoS levels, high reliability
Self-discovery	Topics	Does not own	Publisher, subscriber, topical
Identification of messages	By topic	Queue, topic/ routing key	Topic/key

Hence, in order to access and send the information of both *Robot Nodes* and *PC Nodes*, locally or remotely, the *Nodes* make publication requests only related to the topics of interest, and they can also subscribe to the required topics, by building message packages with the structure shown in Figure 5. As a fixed header of the message package, there is a *Control Byte* that is used to identify the package belonging to the middleware. Then, the message has a variable header with four components: (i) a *package Identifier*, which indicates the origin of the message, that is, the *Node* in which it was created; (ii) a *System Identifier*, which shows the identifier of the *Node* to which the message is addressed—i.e., the robotic system to which the message is being sent; (iii) a *Topic Identifier*, which identifies the topic of interest; and (iv) the *Message Identifier*, which indicates the type of message, **publication** message, or **subscription** message. If the *Message Identifier* indicates a publication message, *Data* are then sent within the payload section, so that another *Robot Node* or *PC Node* can subscribe and obtain that information.

The message package does not necessarily go directly to a final *Node* in the same *Domain*, but it can go outside the local network, since the middleware has a connection with the *Server* in the cloud, which allows for exchanging packages between different *Domains* and Robotic Systems. This facilitates communication among heterogeneous robot systems in different domains.

The topic's definition is related to the robots' components, including both actuators and sensors, which define their heterogeneity. Thus, each type of data that is required to know the status of a robot is defined as a topic for managing the information handled by robots, such as the information of each sensor/actuator, characteristics of the robot, and also each type of data that robots use to perform an action, such as sharing their position, or execute an order.

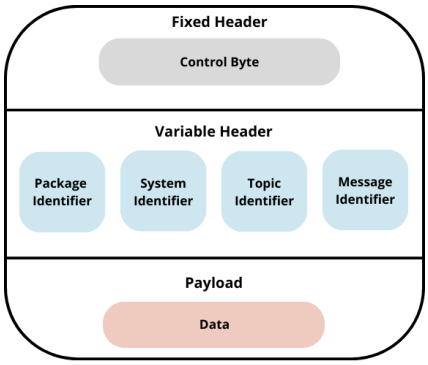


Figure 5. Message package structure of protocol model.

In summary, the communication model offered by the proposed middleware allows for communication among robots that have their own development, hardware, and system characteristics; among robots and their control *Nodes* that have different implementations, communication methods, and control models; and within all the *Domains* in which the robots operate. The middleware also allows the progressive incorporation of other robotic systems or cyber-physical systems.

Middleware Implementation

The way the middleware performs information exchange using the publish and subscribe model is described in Figure 6. The *Cloud Server* starts by initiating a connection with the MongoDB database. If the connection has been established, a communication port is opened to allow connections between the *Nodes* (*clients*) and the *Server*, and the latter waits for said connections through the established socket port. If a *client* (i.e., a *Robot Node* or a *PC Node*) enters, the communication with said *client* is prepared.

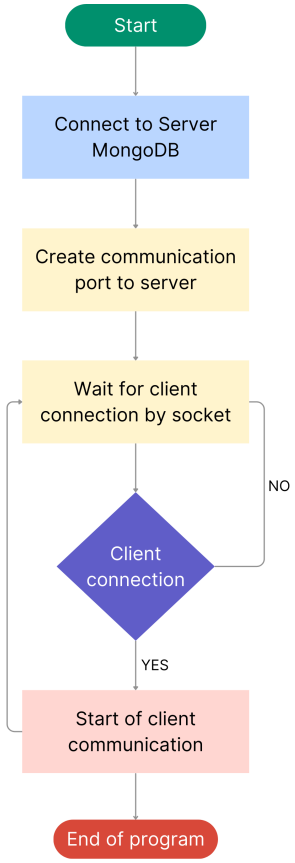


Figure 6. Main program flowchart of *Cloud Server*.

Once communication between the *client* and the *Server* has been established, the program requests the *client*'s IP and connection port, and then receives the messaging package containing the client's query. Then, the message is decoded and analyzed to determine if it is a request that meets the characteristics of the created communication protocol. If the received package does not comply with the protocol, it may be that the *client* is sending a disconnect message, so the *Server* must close the connection with the *client*, but if is not, that package is only discarded, for example, in the case of a package that was misdelivered due to the network. In the case that it is an exchange request, it must be analyzed whether the message is a publication or a subscription, as Figure 7 shows.

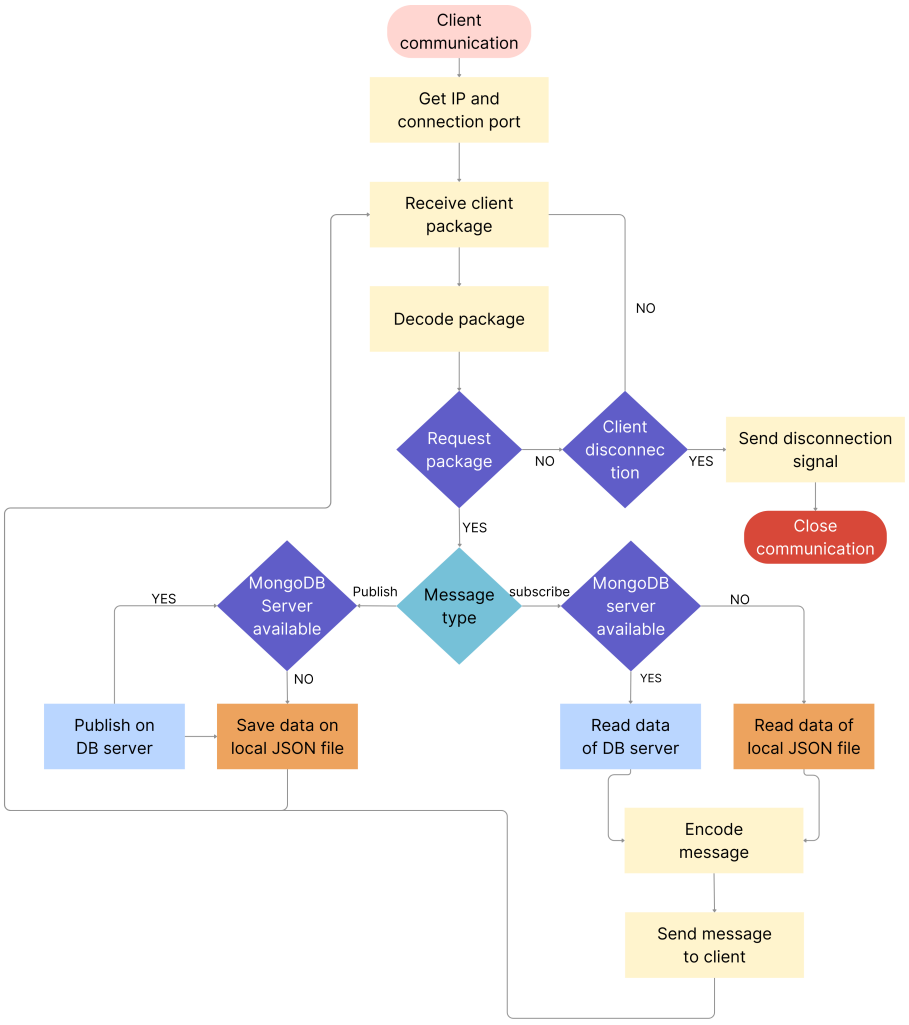


Figure 7. Client communication with server flowchart.

- If it is a publication package, the *Server* obtains the data from the *client* to store them into the remote database (MongoDB). Data are also stored in the Local File System of the *Server* and the *client Node* in JSON format. For publication in MongoDB (Figure 8), a *Collection* must first be defined using the value of the *System Identifier* within the package. If this *Collection* has not been already added to the database, it will be created. Then, a *Document* is defined using the *Topic Identifier*; in the same way, it is queried if the *Document* exists within the database *Collection*, and if it does not exist, a *Document* is created by adding the data contained in the package message. If the *Document* does exist, then the data it contains are updated.
- In the case of a subscription, the process to access the database is similar in terms of the queries that are made. A *Collection* is defined using the *System Identifier*; if it does not exist, an error message is generated. On the other hand, a database *Document*

is defined using the *Topic Identifier*; similarly, if it does not exist, an error message is generated. If the *Collection* and the *Document* exist, then the data contained in the message are extracted to be sent to the *client* (see Figure 9). Another option that the *Server* has is to use the data stored in its Local File System, in case it does not have a connection to the database.

After the middleware has delivered the required information from the database or if an error has been generated in the subscription, the messages are encapsulated and encoded to be returned to the subscribed *client*. This process is repeated for each *client* connected to the *Server*.

The middleware keeps data replicated in a Local File System and in a remote Database (MongoDB); hence, a *Node* within a *System Domain* can work as a *Local Server*, but also, the *Server* can be located in the *cloud*, thus allowing interconnection between any device within the *System's node network*, even those that are located outside the *Local Domain*.

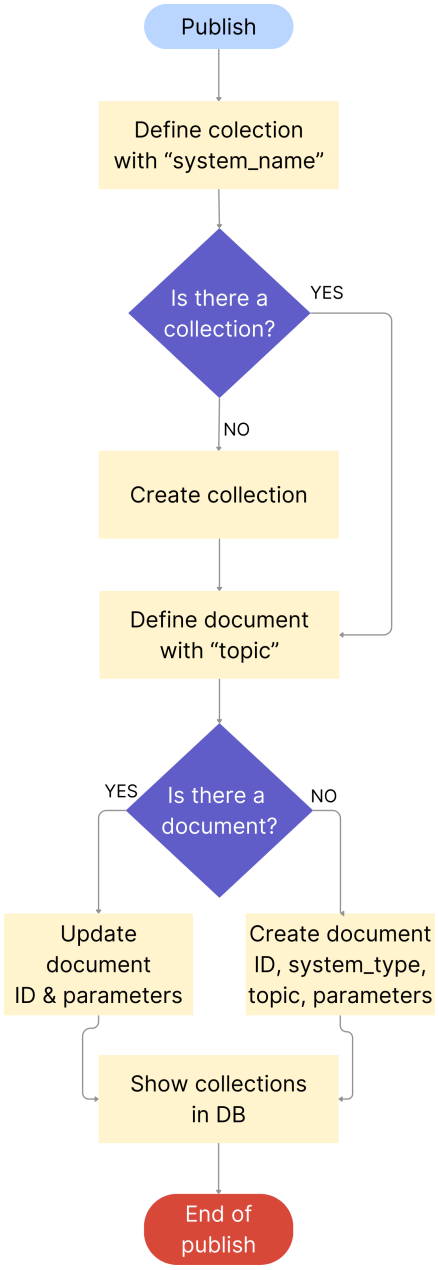


Figure 8. Publish process in *Server* flowchart.

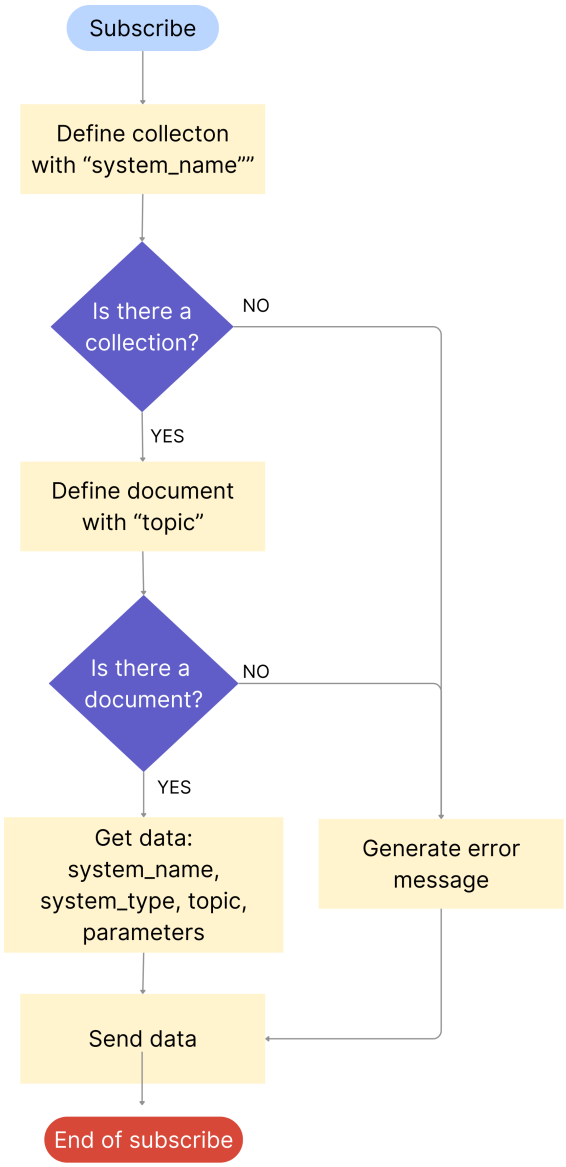


Figure 9. Subscribe process in server flowchart.

5. Application Case: Robots for Urban Tourism

To demonstrate the suitability of the proposed middleware and evaluate its performance, we developed and deployed a proof-of-concept of the middleware as the communication core in the context of a RUTAS (Robots for Urban Touristic Centers Autonomous and Semantic)-based project that considers several types of social robots performing different tasks to guide people in museums. Some of the tasks that robots must execute are Simultaneous Location and Mapping (SLAM) [41], social navigation [42], object detection [43], group identification [44], and multimodal emotion recognition [45,46], which in turn demand collaboration and robot–server, server–server, or robot–robot communication.

The scenario considered features five robots, playing *Robot Nodes*, each of which are related to a respective *PC Node*, which obtains information from the robot or can also control it via the robot's own system. Additionally, there is a *Cloud Server* that incorporates the requisite MongoDB database and is in charge of receiving packages from the *Nodes*, whether publication or subscription. The *Cloud Server* also serves as a *Robot Node*, using ***Turtlesim*** as a model, which is a simulated robot with basic characteristics; this is for making performance comparisons.

5.1. Description of the Application Case Scenario

As mentioned before, the application case scenario consists of a *Cloud Server* and five *Robot Nodes*, each one in a different *Domain*. The *Server* is executed on an Intel Core I5-10300H 2.5 Ghz computer, with 8 GB DDR4 2933 MHZ RAM, 1 TB SSD, an NVidia GeForce GTX 1650—4 GB GDDR6 graphics card, and Ubuntu Server 20.04 installed.

MongoDB version 6.0.3 was used for the storage of the data received by the *Server*; additionally, the data were stored within the Local File System of the *Nodes* involved in the communication, in order to allow local processing in cases where remote access to the *Cloud Server* was not possible or could incur a high overhead. The middleware was implemented in Python 3.8.10 by using libraries compatible with sockets, MongoDB, the relevant file system, and JSON.

Robot Nodes publish or subscribe to their topics in the middleware. The communication with the *Cloud Server* does not interfere with the communication systems that *Robot Nodes* already have with their respective *PC Nodes* within their own *Domain*.

The five *Robot Nodes* considered in the application case were a Pepper robot, an Alice robot, a Turtlebot, a RUTAS Robot (developed as part of the RUTAS project), and a simulated Turtlesim robot, whose characteristics are described as follows:

1. **Pepper robot.** Pepper is a semi-humanoid robot from the Aldebaran company. Among the sensors it includes, there are microphones, one front camera, two stereo depth cameras, contact sensors, shock sensors, lasers, infrared, sonars, one internal IMU, and a battery capacity sensor. As actuators, it uses the motors in the wheels and joints, one speaker, and LEDs. The information provided by these components can be accessed through topics. The following characteristics are presented in the configuration with which the Pepper robot works within the RUTAS project:
    - Hardware: Intel ATOM E3845 Quad core, 1.91 GHz, 4 GB DDR3 RAM
    - Operating System: Ubuntu 18.04
    - Working system: NAOqi 2.9 and ROS Melodic.
    - Communication model: AMQP 1.0
    - Programming language used: C++ and Python 2.7
    - Communication route: WiFi
    - Transport protocol: TCP
2. **Alice robot.** Alice is also a semi-humanoid robot from the company CSJbot. Unlike the Pepper robot, this robot has fewer components and among the sensors, it has one lidar for mapping, one camera located at the top of the touch screen, microphones, one ultrasonic sensor for distances and a battery status controller. Another factor that makes this robot different is its use of two main control boards, one of them for the use of the Android operating system and another for the Ubuntu operating system, both communicating with each other. Most notable configuration features used in the Alice robot are:
    - Hardware: Rockchip RK3399 Dual Cortex-A72 + Quad Cortex-A53, 2.0 GHz, 8 GB DDR3 RAM;
    - Operating System: Android 10 and Ubuntu 20.04;
    - Working system: COS (CSJbot operative system) and ADB;
    - Communication model: RS232 between mainboards and sockets;
    - Programming language used: Java and C++;
    - Communication route: WiFi;
    - Transport protocol: TCP.
3. **Turtlebot4.** This is a mobile robot with a circular shape, which is mainly used to perform SLAM; it has several sensors to fulfill its purpose: a charging station guidance system, an OAK -D spatial AI stereo camera, 2D Lidar, IMU, optical floor tracking sensor, wheel encoders, and infrared, cliff, bump, and slip detection. Likewise, the characteristics configured in this robot are the following:

- Hardware: Raspberry pi 4 Quad core Cortex-A72, 1.5 GHz, 4 GB DDR3 RAM;
- Operating System: Ubuntu 20.04 and 22.04;
- Job system: ROS2 Galactic and Humble;
- Communication model: DDS;
- Programming language used: Python 3;
- Communication route: WiFi;
- Transport protocol: TCP.

4. **RUTAS robot.** This is a robot created for the RUTAS project, which combines several characteristics that the previous robots have, but for the same purpose of applying its systems in social environments. It has one depth camera, one lidar, infrared sensors, distance sensors, an internal IMU, encoders on its two wheels, and six degrees of freedom located on its head and arms. The characteristics of the system used with this robot are as follows:
   - Hardware: 8-Kern Arm Cortex-A78AE, 2 GHz, 16 GB DDR4 RAM;
   - Operating System: Ubuntu 20.04;
   - Work system: ROS Noetic;
   - Communication model: AMQP 1.0;
   - Programming language used: Python 3;
   - Communication route: WiFi;
   - Transport protocol: TCP.
5. **Turtlesim.** This is a tool integrated within ROSs, which encapsulates many of the fundamental topics handled by ROSs and some robots. Turtlesim is executed as a *Robot Node* that can receive and send messages. That is why, within the *Server*, ROS Noetic was installed to execute this *Robot Node* and can be used in comparison with the communication of other *Robot Nodes*. The characteristics that they present are the same as those of the *Server* and ROS Noetic:
   - Work system: ROS Noetic;
   - Communication model: TCPROS;
   - Programming language used: Python 3;
   - Communication route: WiFi;
   - Transport protocol: TCP.

Some robotic systems have topics with the same name and make reference to the same type of information; such is the case of the Pepper robot, Turtlebot4, Turtlesim, and RUTAS robot, which use an ROS as an internal system; however, the communication mode and protocols they use vary due to the versions of their internal systems, as is the case of the Pepper robot with the ROS Melodic version and Ubuntu 18.04 and Turtlebot4 with ROS 2 Galactic and Ubuntu 20.04. In addition, a robot can also have a controller based on another type of operating system, such as the Alice robot, which works with Android 10 and whose communication with the *PC Node* that belongs to its *Domain* is given under an Android Debug Bridge (ADB) and socket connection, being in majority very different from the communication modes of the other robots.

The aforementioned characteristics of the robots are those that provide heterogeneity in the system and influence both the topics sent and the transmission speed from device to device. The topics make up the information that is needs to be transmitted or used, such as sensor or engine position information.

5.2. Tests and Results

We performed two types of tests. In the first one, Turtlesim, Turtlebot4, the RUTAS robot, and the Pepper robot publish the topic "direction&velocity", named *cmd_vel*, and in the case of the Alice robot, the topics published are *NAVI_ROBOT_MOVE_REQ* (for lineal movements) and *NAVI_GO_ROTATION_REQ* (for rotational movements), as shown in Table 4. The topic *cmd_vel* consists of two vectors, which correspond to the linear velocity (x, y, z) and the angular velocity (a, b, c), with their corresponding values in the three axes.

The topics used by the Alice robot only have one value corresponding to the form of movement, with 0 for *forward*, 1 for *backward*, 2 for rotating *left*, 3 for turning *right*, and 4 for *stopping*, in the case of lineal movement topics, (*NAVI_ROBOT_MOVE_REQ*), and both a positive and negative number indicating the angle of rotation in the case of rotational movement topics, (*NAVI_GO_ROTATION_REQ*).

This test consists of transmitting packages from each *Robot Node* to the *Server*. In the experiment, 1000 publication and 1000 subscription packages are sent by each *Robot Node* to the *Server*, including from the *Turtlesim Node*, to calculate the average per package. In the case of the subscription packages, the *Server* must respond to the subscribed *Node* with a package containing the required information. In Figure 10, the messages that each *Robot Node* sends to the *Server* can be appreciated. As part of the created protocol, the packet sent, both for subscription and publication, starts with the Control Byte and is followed by the packet, system, and topic identifiers. The packet continues with the type of message being sent, either subscription ("s") or publication ("p"); in the latter case, the topic parameters must also be added. For all publication cases, where the topics are indicated in Table 4, values that produce no movement were used to evaluate the performance of the packet transfer. The packets that use the ROS system follow the following pattern: *%01/System_Identifier/cmd_vel/p/0.0,0.0,0.0,0.0,0.0,0.0%*, indicating the absence of linear and angular velocity; and packets using the COS system take the form of *%01/ALICE/COS/NAVI_ROBOT_MOVE_REQl/p/stop%*, indicating the robot has stopped. All *Robot Nodes* that work with an ROS have similar messages, varying only in respect to the Robot System Identifier, and every message has a Topic Identifier to differentiate between publish or subscribe messages.

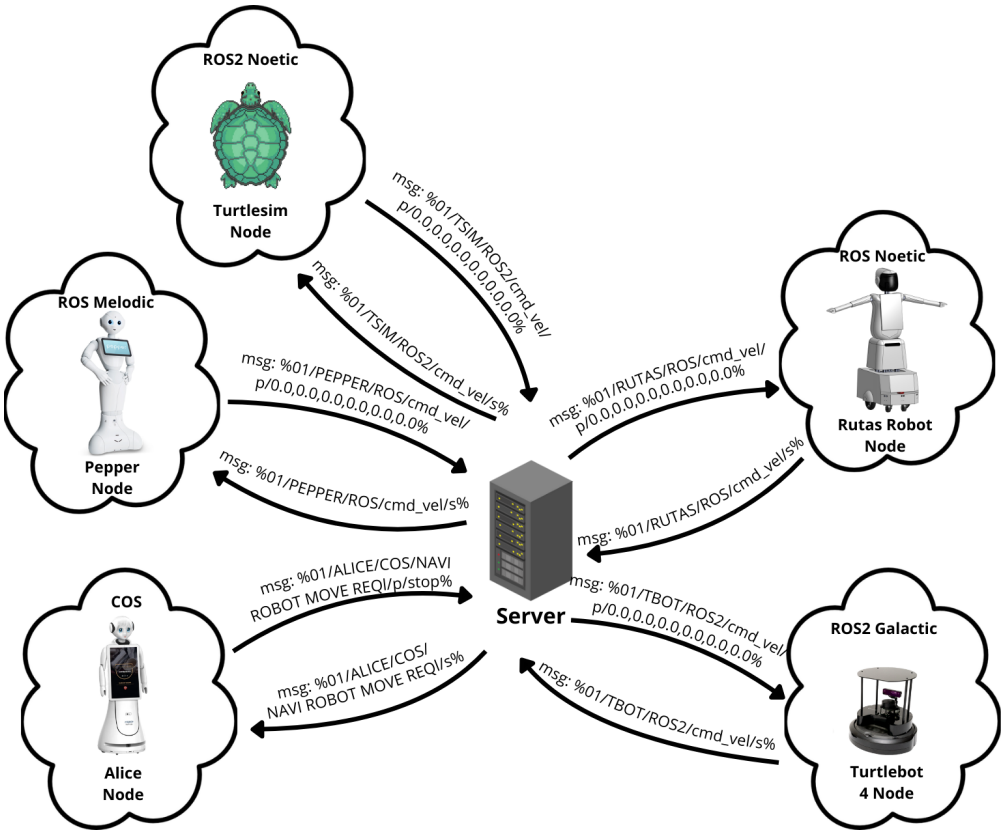


Figure 10. Messages published and subscribed by each Robot Node.

Table 4. Topic "direction&velocity" used from *Robot Nodes*.

Robot	System	Topic	Parametes
Turtlesim	ROS2	cmd_vel	x/y/z/a/b/c
Turtlebot4	ROS2	cmd_vel	x/y/z/a/b/c
RUTAS robot	ROS	cmd_vel	x/y/z/a/b/c
Pepper robot	ROS	cmd_vel	x/y/z/a/b/c
Alice robot	COS	NAVI_ROBOT_MOVE_REQ NAVI_GO_ROTATION_REQ	direction

Results presented in Figure 11 show the time in microseconds that it takes for each *Robot Node* to communicate with the *Server*. The average time of each communication between a *Node* and the *Server* is affected by the hardware characteristics of each *Node* and the means of connectivity, which, in this application case, was carried out by local WiFi.

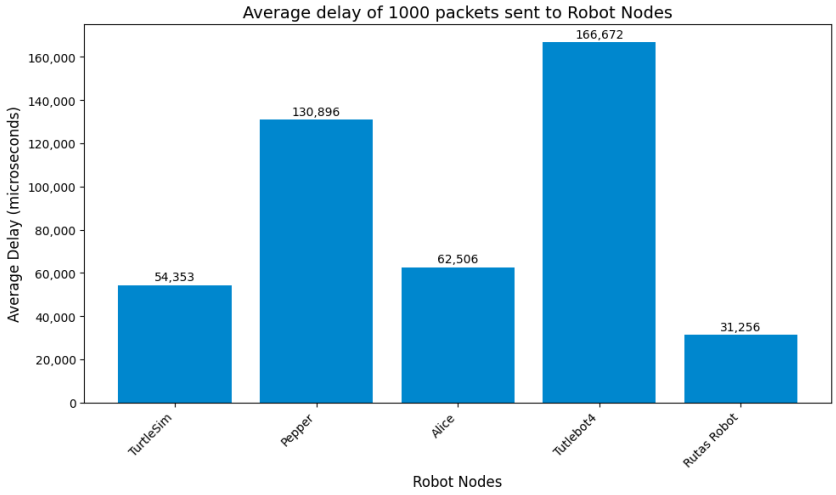


Figure 11. Average time for publishing package to be sent from *Robot Node* to *Server*.

Figure 12 shows the average time in microseconds that each subscription request took, taking into account that not only the client *Node* sends a message package, but also the *Server* must respond the subscribed topic with a response package; thus, with respect to time, this process is longer compared to the publication process.

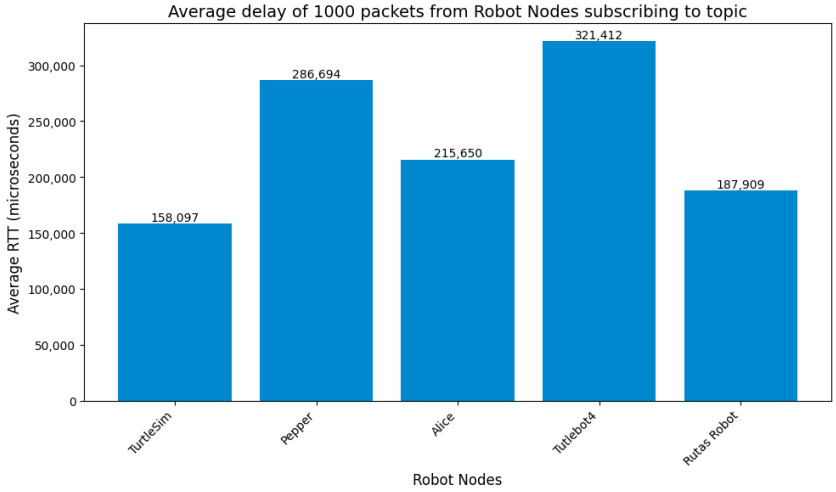


Figure 12. Average time for subscribing package to be sent from *Robot Node* to *Server*.

In both processes, the fastest communication times are those for *Robot Nodes* that have better hardware, such as Turtlesim, the *Server*, the RUTAS robot, and the Alice robot, in contrast to Turtlebot4 and the Pepper robot, which have more limited hardware. However, the worst average time is near 320 ms, which is considered a good time for real-time applications [47,48].

It was also analyzed how many packages in total were delivered from the *Robot Nodes* and received by the *Server*, both in publication mode and subscription mode. As shown in Figure 13, delivered messages from *Robot Nodes* had a high percentage of packages successfully received by the *Server* (up to 99.49%), with the lowest success rate associated with those delivered from the *Turtlesim Node* (95.18%), this being due to a possible saturation in the communication, since this *Robot Node* works inside the same *Server*.

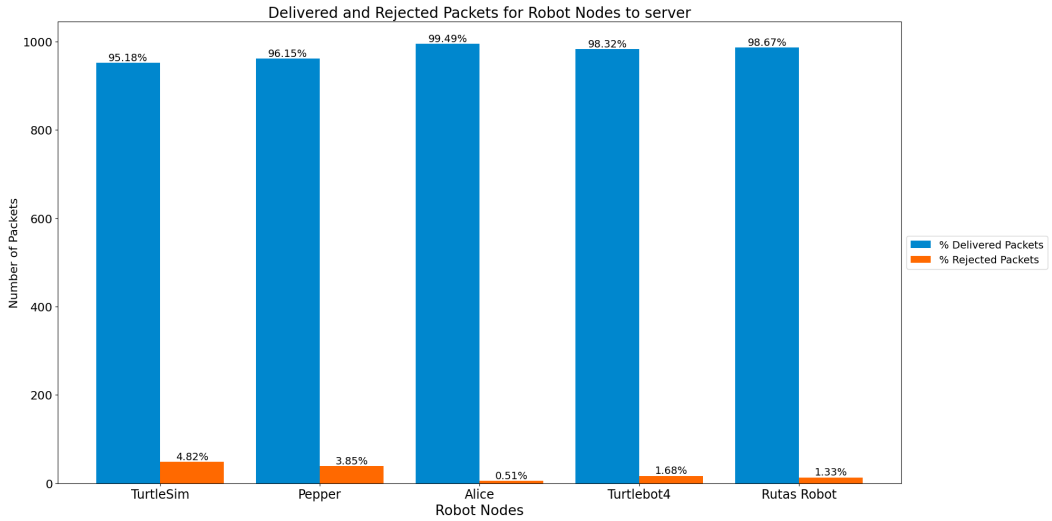


Figure 13. Percentage of correct packages delivered from *Robot Nodes* to *Server*.

For the second test, an application to control robots' movements by pressing directional keys on a keyboard was created. This program was executed on a *PC Node* to perform a publishing process, in which the speed and direction parameters for the movement of the robots are established through the topic *cmd_vel*, applicable to most RUTAS project robots.

For this case, two *Domains* were used, one involving *Turtlebot4*, which has the ROS 2 Galactic system, and the other *Domain* including *Turtlesim*, which uses the ROS Noetic system. Both *Robot Nodes* are subscribed to the topic *cmd_vel*. The *PC Node* that publishes the topic *cmd_vel* was situated in the *Domain* of Turtlesim. The program simultaneously sends two publication packages to the *Server* for both *Robot Nodes*, containing the same topic and the same parameters; the information from these packages is stored by the *Server* in MongoDB. Each time the *up arrow* or *down arrow* key is pressed, each package sends within its parameter component a fixed positive or negative value of linear velocity on the *X* axis, and when the *right arrow* or *left arrow* key is pressed, a fixed negative or positive value of angular velocity on the *Z* axis is sent. Then, a program that sends *subscription* messages is executed by both the *Turtlebot4 Node* and the *Turtlesim Node*, which both subscribe to the topic found on the *Server* to obtain the speed and directional parameters they need to move.

Figure 14 illustrates the movements performed by Turtlebot4 (images above) and Turtlesim (images below). Both *Robot Nodes* start at rest (Figure 14a); then, the topic indicates a forward movement for 2 s; thus, they move in a straight line for a certain distance (Figure 14b). For both cases, the distance traveled varies given the robots' properties (one of them can move faster than the other); afterwards, the command is to turn—approximately 45 degrees (Figure 14c); the angle of the turn is not exact due to the angular velocity; then, the topic indicates a forward movement in a straight line with the same speed for 1 s (Figure 14d); and finally, the robot makes a turn of approximately 180 degrees (Figure 14e).

Both systems subscribe to the *Server* to obtain the corresponding speed data, thus achieving similar movement within the environments they are in, demonstrating that different systems manage to communicate in order to accomplish the same tasks simultaneously.

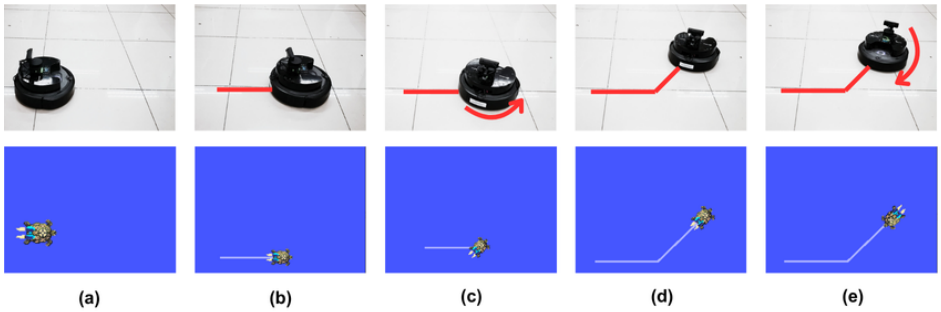


Figure 14. Moving Turtlebot4 (images above) and Turtlesim (images below) using publish and subscribe in middleware.

The results show that the middleware allows for establishing communication between various robotic systems with different characteristics applied in the RUTAS project, enabling a system like Trutlebot4 to extract information from another system that does not have the same connection platform. Likewise, any *Robot Node* can publish the collected information, not only to a *Server*, but also to another device when it requires it, only requiring that one of the two can be established as a *Server* and the other as a *client*.

6. General Discussion

The proposed middleware enables seamless communication between heterogeneous robotic systems, accommodating differences in hardware and software. This capability provides a flexible foundation for developing diverse applications that leverage various robotic platforms within the same network, even in different domains, enhancing the scope of potential use cases.

Additionally, the middleware supports both cloud-based and local communication systems, offering adaptability for different deployment scenarios. This dual approach ensures efficient operation in environments requiring global connectivity or localized environments, and even cases involving offline functionality.

Its hybrid communication model further enhances scalability, allowing dynamic switching between centralized and distributed modes. This flexibility optimizes resource utilization and maintains performance as the system scales, making it suitable for complex and expanding robotic networks. Combining a communication protocol with integrated robotic systems applied to cloud resources contributes to better interoperability and collaboration in robotic environments.

In social tourism environments such as the RUTAS project, this scalability becomes crucial, as the middleware enables diverse robots and devices to communicate seamlessly. The ability to integrate heterogeneous systems allows robots to collaborate on tasks like SLAM, where they share sensor data to build distributed maps. For instance, ground robots can map detailed floor-level areas while humanoid robots provide a higher image perspective. By synchronizing maps in real-time through the middleware, the system reduces redundancy and enhances mapping accuracy, addressing the complexities of expansive or crowded tourist locations. Another example of applicability in the tourist domain is the possibility of dynamically integrating other devices in the systems, such as tourists' mobile devices, in order to let them access information services regarding the relevant museum. Overall, this system can improve visitor interaction in tourist spaces, operational efficiency, and scalability, making it an adaptable solution for managing complex tourist environments.

Challenges may arise from latency in distributed systems or connectivity issues in remote areas. However, the middleware mitigates these issues by maintaining local data

replication, ensuring continuous operation even during outages. Integrating heterogeneous robots may also present complexities in standardizing data formats, but this is addressed by the middleware's unified communication protocol.

7. Conclusions

Heterogeneous multi-robot systems are becoming solutions for developing robotics applications in dynamic environments that demand the execution of collaborative tasks. In these scenarios, a communication platform is needed to allow heterogeneous robots to share information and collaborate in the execution of tasks. Existing solutions are limited to robots with the same characteristics or to the communication of robots in the same physical space (i.e., in the same domain). To overcome the limitations of existing solutions, in this work, we proposed a communication middleware that supports interaction among robotic systems that have different characteristics or are located in different domains. The proposed middleware is based on a generalized publish–subscribe communication model, on the principles of IoRT, and on Cloud Computing technologies, leading to a Cloud Robotics platform. The middleware can be adapted according to robotics systems interacting in the same domain or across different domains, allowing multi-robot systems to be more robust when interacting, as well as in a more flexible and scalable way.

To evaluate the functionalities and performance of the proposed middleware, we developed a proof-of-concept of the middleware in an application case with guide robots in museums with heterogeneous systems and characteristics, featuring different domains. Results show that the middleware is able to support such heterogeneity in an efficient way, allowing for real-time applications.

In order to mitigate the performance disparity when it comes to the communication of heterogeneous robots with low hardware performance, we are currently working on implementing optimization strategies to improve communication, such as local processing or data replication. The current capability of the middleware regarding keeping the shared data in local file systems as well as in the *Cloud Server* makes it possible to use functions that can be executed locally, in addition to the transmission of information among robotic systems, to reduce communication or accommodate cases in which it is not possible to reach the remote *Server*. To do so, it is also necessary to capture optimization measures that can help evaluate the state of the network and the system, as well as evaluate the aspects that impact communication overhead the most. This capacity will increase the robustness and scalability of the middleware in more robotic environments. We also are deploying the middleware as the communication core in the RUTAS project in order to implement several services for cooperative heterogeneous social robots, such as cooperative SLAM and human–robot interactions based on emotion recognition wherein robots are designed to behave and express themselves accordingly.

Author Contributions: Conceptualization, E.C.Z., D.B.A. and Y.C.; methodology, E.C.Z., D.B.A. and Y.C.; software, E.C.Z.; validation, E.C.Z., D.B.A. and Y.C.; investigation, E.C.Z., D.B.A. and Y.C.; resources, D.B.A.; writing—original draft preparation, E.C.Z.; writing—review and editing, E.C.Z. and Y.C.; supervision, D.B.A. and Y.C.; project administration, D.B.A.; funding acquisition, D.B.A. and Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by PROCIENCIA as an executing entity of CONCYTEC under grant agreement no. PE501080073-2022-PROCIENCIA as part of the project RUTAS 2.0: Robots for Urban Tourist centers with Advanced Social interaction—Towards the preservation and diffusion of cultural heritage.

Data Availability Statement: Data contained in this article.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Baratta, A.; Cimino, A.; Gnoni, M.G.; Longo, F. Human Robot Collaboration in Industry 4.0: A literature review. *Procedia Comput. Sci.* **2023**, *217*, 1887–1895. [CrossRef] [PubMed]
2. Stavropoulos, P.; Alexopoulos, K.; Makris, S.; Papacharalampopoulos, A.; Dhondt, S.; Chryssolouris, G. AI in manufacturing and the role of humans: Processes, robots, and systems. In *Handbook of Artificial Intelligence at Work*; Edward Elgar Publishing: Cheltenham, UK , 2024; pp. 119–141.
3. Glas, D.F.; Kanda, T.; Ishiguro, H.; Hagita, N. Teleoperation of Multiple Social Robots. *IEEE Trans. Syst. Man Cybern. Part Syst. Hum.* **2012**, *42*, 530–544. [CrossRef]
4. Chakraa, H.; Guérin, F.; Leclercq, E.; Lefebvre, D. Optimization techniques for Multi-Robot Task Allocation problems: Review on the state-of-the-art. *Robot. Auton. Syst.* **2023**, *168*, 104492. [CrossRef]
5. Rosete, A.; Soares, B.; Salvadorinho, J.; Reis, J.; Amorim, M. Service Robots in the Hospitality Industry: An Exploratory Literature Review. In *Exploring Service Science*; Nóvoa, H., Drăgoicea, M., Kühl, N., Eds.; Springer : Cham, Switzerland, 2020; pp. 174–186.
6. Luperto M.; Monroy J.R.J. Integrating Social Assistive Robots, IoT, Virtual Communities and Smart Objects to Assist at-Home Independently Living Elders: The MoveCare Project. *Int. J. Soc. Robot.* **2023**, *15*, 517–545. [CrossRef] [PubMed]
7. Rosenberg-Kima, R.B.; Koren, Y.; Gordon, G. Robot-Supported Collaborative Learning (RSCL): Social Robots as Teaching Assistants for Higher Education Small Group Facilitation. *Front. Robot.* **2020**, *6*, 148. [CrossRef]
8. Nisiotis, L.; Alboul, L. Initial Evaluation of an Intelligent Virtual Museum Prototype Powered by AI, XR and Robots. In *Augmented Reality, Virtual Reality, and Computer Graphics*; De Paolis, L.T., Arpaia, P., Bourdot, P., Eds.; Springer: Cham, Switzerland, 2021; pp. 290–305.
9. Msala, Y.; Hamlich, M.; Mouchtachi, A. A new Robust Heterogeneous Multi-Robot Approach Based on Cloud for Task Allocation. In Proceedings of the 5th International Conference on Optimization and Applications, Kenitra, Morocco, 25–26 April 2019; pp. 1–4. [CrossRef]
10. Shorinwa, O.; Halsted, T.; Yu, J.; Schwager, M. Distributed Optimization Methods for Multi-robot Systems: Part 1—A Tutorial. *IEEE Robot. Autom. Mag.* **2024**, *31*, 121–138. [CrossRef]
11. Rizk, Y.; Awad, M.; Tunstel, E.W. Cooperative heterogeneous multi-robot systems: A survey. *ACM Computer Survey (CSUR)* **2019**, *52*, 1–31. [CrossRef]
12. Casals Gelpi, A.; Vinagre Ruiz, M.; Aranda López, J.; Amat Girbau, J. Plataforma para un entorno asistencial inteligente heterogéneo. In Proceedings of the Jornadas Nacionales de Robótica—Spanish Robotics Conference. Comité Español de Automática (CEA-IFAC), Valladolid, Barcelona, 14–15 June 2018; pp. 1–6.
13. Barber, R.; Ortiz, F.J.; Calatrava, F.M.; Garrido, S.; Alfonso, L.M.J.; Vera, A.M.; Prados, A.; Roca, J.; Jiménez, M.; Mendez, I.; et al. Himtae: Sistema heterogéneo multirobot para ayuda de personas mayores en un ambiente asistido en el hogar. *XII Jornadas Nac. RobóTica (MáLaga Univ. MáLaga)* **2022**, 106 –117. ISBN 978-84-09-41095-8
14. Batth, R.S.; Nayyar, A.; Nagpal, A. Internet of Robotic Things: Driving Intelligent Robotics of Future—Concept, Architecture, Applications and Technologies. In Proceedings of the 4th International Conference on Computing Sciences, Jalandhar, India, 30–31 August 2018; pp. 151–160. [CrossRef]
15. Bhat, K.U.; Kumar, N.; Koul, N.; Verma, C.; Enescu, F.M.; Raboaca, M.S. Intelligent Communication for Internet of Things (IoRT). In *Proceedings of the International Conference on Recent Innovations in Computing*; Springer Nature: Singapore, 2023; pp. 313–328.
16. Kabir, H.; Tham, M.L.; Chang, Y.C. Internet of robotic things for mobile robots: Concepts, technologies, challenges, applications, and future directions. *Digit. Commun. Netw.* **2023**, *9*, 1265–1290. [CrossRef]
17. Odirichukwu, J.; Ndigwe, C.; Njoku, O. Internet of Things (IoT), Internet of Robotic Things (IoRT), IoT Security (IoTS) and Machine Learning Algorithms: A Review Perspective. *Univ. Ib. J. Sci. Log. Ict Res.* **2023**, *9*, 74–81. ISSN: 2714-3627 .
18. Odirichukwu, J.; Asagba, P.; Onuodu, F. Interoperable Protocols of the Internet of Things and Internet Of Robotic Things: A Review. *Int. J. Comput. Intell. Secur. Res.* **2021**, *1*, 101–123.
19. Vermesan, O.; Bahr, R.; Ottella, M.; Serrano, M.; Karlsen, T.; Wahlstrøm, T.; Sand, H.E.; Ashwathnarayan, M.; Gamba, M.T. Internet of Robotic Things Intelligent Connectivity and Platforms. *Front. Robot.* **2020**, *7*, 509753. [CrossRef] [PubMed]
20. Bi, Z.; Miao, Z.; Zhang, B.; Zhang, C.W.J. Framework for Performance Assessment of Heterogeneous Robotic Systems. *IEEE Syst. J.* **2021**, *15*, 1191–1201. [CrossRef]
21. Sewtz, M.; Lay, F.S.; Luo, X.; Chupin, T.; Lii, N.Y. Enabling Communication between Heterogeneous Robots and Human Operators in Collaborative Missions. In Proceedings of the IEEE Aerospace Conference, Big Sky, MT, USA, 2–9 March 2024; pp. 1–8. [CrossRef]
22. Rajapaksha, U.K.; Jayawardena, C.; MacDonald, B.A. ROS Based Heterogeneous Multiple Robots Control Using High Level User Instructions. In Proceedings of the IEEE Region 10 Conference (TENCON), Auckland, New Zealand, 7–10 December 2021; pp. 163–168. [CrossRef]
23. Shi, L.; Marcano, N.J.H.; Jacobsen, R.H. A review on communication protocols for autonomous unmanned aerial vehicles for inspection application. *Microprocess. Microsyst.* **2021**, *86*, 104340. [CrossRef]
24. Tuci, E.; Alkilabi, M.H.M.; Akanyeti, O. Cooperative Object Transport in Multi-Robot Systems: A Review of the State-of-the-Art. *Front. Robot.* **2018**, *5*, 1–31. [CrossRef] [PubMed]

25. Sahni, Y.; Cao, J.; Jiang, S. Middleware for multi-robot systems. In *Mission-Oriented Sensor Networks and Systems: Art and Science: Volume 2: Advances*; Springer: Cham, Switzerland, 2019; pp. 633–673.
26. Anderson, J.E. Humanoid Multi-robot Systems. In *Humanoid Robotics: A Reference*; Goswami, A., Vadakkepat, P., Eds.; Springer: Dordrecht, The Netherlands, 2019; pp. 2473–2481. [CrossRef]
27. Wan, J.; Tang, S.; Yan, H.; Li, D.; Wang, S.; Vasilakos, A.V. Cloud robotics: Current status and open issues. *IEEE Access* **2016**, *4*, 2797–2807. [CrossRef]
28. Grieco, L.; Rizzo, A.; Colucci, S.; Sicari, S.; Piro, G.; Di Paola, D.; Boggia, G. IoT-aided robotics applications: Technological implications, target domains and open issues. *Comput. Commun.* **2014**, *54*, 32–47. [CrossRef]
29. Shakya, S. Survey on Cloud Based Robotics Architecture, Challenges and Applications. *J. Ubiquitous Comput. Commun. Technol.* **2020**, *2*, 10–18. [CrossRef]
30. Chen, W.; Yaguchi, Y.; Naruse, K.; Watanobe, Y.; Nakamura, K.; Ogawa, J. A Study of Robotic Cooperation in Cloud Robotics: Architecture and Challenges. *IEEE Access* **2018**, *6*, 36662–36682. [CrossRef]
31. Halsted, T.; Shorinwa, O.; Yu, J.; Schwager, M. A Survey of Distributed Optimization Methods for Multi-Robot Systems. *arXiv* **2021**, arXiv:2103.12840.
32. Gielis, J.; Shankar, A.; Prorok, A. A critical review of communications in multi-robot systems. *Curr. Robot. Rep.* **2022**, *3*, 213–225. [CrossRef] [PubMed]
33. Jawhar, I.; Mohamed, N.; Al-Jaroodi, J. Secure Communication in Multi-Robot Systems. In Proceedings of the IEEE Systems Security Symposium, Crystal City, VA, USA, 1 July–1 August 2020; pp. 1–8. [CrossRef]
34. Matteucci, M. Publish/Subscribe Middleware for Robotics: Requirements and State of the Art. Tech. Report N 2003.3. Citeseer, 2003, pp. 1–36. Available online: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=82e61c3c0a3c5719e0312b7f5f39fff17040ffe0 (accessed on 27 October 2024).
35. Yan, Z.; Fabresse, L.; Laval, J.; Bouraqadi, N. Building a ROS-Based Testbed for Realistic Multi-Robot Simulation: Taking the Exploration as an Example. *Robotics* **2017**, *6*, 21. [CrossRef]
36. Macenski, S.; Foote, T.; Gerkey, B.; Lalancette, C.; Woodall, W. Robot Operating System 2: Design, architecture, and uses in the wild. *Sci. Robot.* **2022**, *7*, eabm6074. [CrossRef]
37. Gao, Z.; Wanyama, T.; Singh, I.; Gadhrri, A.; Schmidt, R. From Industry 4.0 to Robotics 4.0—A Conceptual Framework for Collaborative and Intelligent Robotic Systems. *Procedia Manuf.* **2020**, *46*, 591–599. [CrossRef]
38. Ferranti, L.; D'Oro, S.; Bonati, L.; Demirors, E.; Cuomo, F.; Melodia, T. HIRO-NET: Self-Organized Robotic Mesh Networking for Internet Sharing in Disaster Scenarios. In Proceedings of the IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks", Washington, DC, USA, 10–12 June 2019; pp. 1–9. [CrossRef]
39. Lončar, I.; Babić, A.; Arbanas, B.; Vasiljević, G.; Petrović, T.; Bogdan, S.; Mišković, N. A Heterogeneous Robotic Swarm for Long-Term Monitoring of Marine Environments. *Appl. Sci.* **2019**, *9*, 1388. [CrossRef]
40. Dey, E.; Walczak, M.; Anwar, M.S.; Roy, N.; Freeman, J.; Gregory, T.; Suri, N.; Busart, C. A Novel ROS2 QoS Policy-Enabled Synchronizing Middleware for Co-Simulation of Heterogeneous Multi-Robot Systems. In Proceedings of the 32nd International Conference on Computer Communications and Networks, Honolulu, HI, USA, 24–27 July 2023; pp. 1–10. [CrossRef]
41. Cornejo-Lupa, M.A.; Cardinale, Y.; Ticona-Herrera, R.; Barrios-Aranibar, D.; Andrade, M.; Diaz-Amado, J. Ontoslam: An ontology for representing location and simultaneous mapping information for autonomous robots. *Robotics* **2021**, *10*, 125. [CrossRef]
42. Daza, M.; Barrios-Aranibar, D.; Diaz-Amado, J.; Cardinale, Y.; Vilasboas, J. An approach of social navigation based on proxemics for crowded environments of humans and robots. *Micromachines* **2021**, *12*, 193. [CrossRef]
43. Tejada-Mesias, A.; Dongo, I.; Cardinale, Y.; Diaz-Amado, J. Odrom: Object detection and recognition supported by ontologies and applied to museums. In Proceedings of the XLVII Latin American Computing Conference (CLEI), Cartago, Costa Rica, 25–29 October 2021; pp. 1–10.
44. Quiroz, M.; Patiño, R.; Diaz-Amado, J.; Cardinale, Y. Group emotion detection based on social robot perception. *Sensors* **2022**, *22*, 3749. [CrossRef]
45. Graterol, W.; Diaz-Amado, J.; Cardinale, Y.; Dongo, I.; Lopes-Silva, E.; Santos-Libarino, C. Emotion detection for social robots based on NLP transformers and an emotion ontology. *Sensors* **2021**, *21*, 1322. [CrossRef]
46. Heredia, J.; Cardinale, Y.; Dongo, I.; Aguilera, A.; Diaz-Amado, J. Multimodal emotional understanding in robotics. In *Workshops at 18th International Conference on Intelligent Environments (IE2022)—1st International Workshop on Sentiment Analysis and Emotion Recognition for Social Robots (SENTIRobots)*; IOS Press: Amsterdam, The Netherlands, 2022; pp. 46–55.
47. Miller, R.B. Response time in man-computer conversational transactions. In Proceedings of the 9–11 December 1968, Fall Joint Computer Conference, Part I, San Francisco, CA, USA, 9–11 December 1968; pp. 267–277.
48. Stonebraker, M.; Çetintemel, U.; Zdonik, S. The 8 requirements of real-time stream processing. *ACM Sigmod Rec.* **2005**, *34*, 42–47. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.