Article

Design of a Cyber-Physical System-of-Systems Architecture for Elderly Care at Home

José Galeas, Alberto Tudela, Óscar Pons, Juan Pedro Bandera and Antonio Bandera *

Departamento de Tecnología Electrónica, University of Málaga, 29071 Malaga, Spain; jgaleas@uma.es (J.G.); ajtudela@uma.es (A.T.); opfernandez@uma.es (Ó.P.); jpbandera@uma.es (J.P.B.)
* Correspondence: ajbandera@uma.es

Abstract: The idea of introducing a robot into an Ambient Assisted Living (AAL) environment to provide additional services beyond those provided by the environment itself has been explored in numerous projects. Moreover, new opportunities can arise from this symbiosis, which usually requires both systems to share the knowledge (and not just the data) they capture from the context. Thus, by using knowledge extracted from the raw data captured by the sensors deployed in the environment, the robot can know where the person is and whether he/she should perform some physical exercise, as well as whether he/she should move a chair away to allow the robot to successfully complete a task. This paper describes the design of an Ambient Assisted Living system where an IoT scheme and robot coexist as independent but connected elements, forming a cyber-physical system-of-systems architecture. The IoT environment includes cameras to monitor the person's activity and physical position (lying down, sitting…), as well as non-invasive sensors to monitor the person's heart or breathing rate while lying in bed or sitting in the living room. Although this manuscript focuses on how both systems handle and share the knowledge they possess about the context, a couple of example use cases are included. In the first case, the environment provides the robot with information about the positions of objects in the environment, which allows the robot to augment the metric map it uses to navigate, detecting situations that prevent it from moving to a target. If there is a person nearby, the robot will approach them to ask them to move a chair or open a door. In the second case, even more use is made of the robot's ability to interact with the person. When the IoT system detects that the person has fallen to the ground, it passes this information to the robot so that it can go to the person, talk to them, and ask for external help if necessary.

Keywords: cyber-physical system of systems; ambient assisted living; social assistance robot

check for updates

Citation: Galeas, J.; Tudela, A.; Pons, Ó.; Bandera, J.P.; Bandera, A. Design of a Cyber-Physical System-of-Systems Architecture for Elderly Care at Home. *Electronics* **2024**, *13*, 4583. https://doi.org/10.3390/electronics13234583

Academic Editor: Praveen Kumar Donta

Received: 21 October 2024
Revised: 15 November 2024
Accepted: 18 November 2024
Published: 21 November 2024

1. Introduction

Population ageing is already a reality in much of the world and is causing a profound demographic change. Thus, taking into account the population as a whole (not just those over 64 years), globally, the number of workers per older person will fall from 7 in 2015 to only 4.9 in 2030. In Western European countries, it will fall from 3.5 workers per older person in 2015 to 2.4 in 2030 [1]. In view of these figures, the near future will require greater investment in health and social resources for older people. At the same time, the percentage of health professionals and temporary carers who can assist older people in their daily lives will decrease. To cope with these changes, it will be necessary to develop new action plans to ensure the well-being of older people. Among these plans is the concept of Active Ageing, defined as "the process of optimising opportunities for health, participation and security to improve quality of life as people age" [2]. According to the Active Ageing criteria, people should try to remain as independent and active as possible for as long as possible. This policy aims to increase the person's sense of well-being but also to reduce the costs of care, which are delayed or even avoided. In order to implement Active Ageing policies, it is necessary to guarantee that people have adequate protection, security, and care. This implies personalised treatment, continuous assessment of their capabilities, and the use of monitoring, communication, and therapeutic technologies, both at home and in hospitals or nursing homes. These technologies for Active Ageing, as part of a concept termed Ambient Assisted Living (AAL), are intended to not only contribute to the independence of older people but also alleviate the workload of health professionals and caregivers, without replacing them in any way.

The design of an AAL environment involves integrating different technologies, and the Internet of Things (IoT), given its ability to connect everything from everyday physical items to medical devices to the Internet, can be seen as the normal basis on which to build it. Adding a robot to this ecosystem of sensors and actuators fixed to a certain position presents the interesting possibility of having an element that can move wherever it is needed to interact with the person using channels that are natural and intuitive (voice, gestures…), to help them physically (especially if the person can lean on it or if the robot has arms and hands), or to capture specific data using the sensors it carries [1]. In order for the AAL ecosystem to handle high-level tasks involving some level of cognition, it must have a working memory or state representation, which will typically combine metric information (e.g., the person's position or heart and breathing rates) and symbolic information (e.g., whether he/she is sad). If the IoT system and the robot itself are considered a single cyber-physical system, they will share the same state representation [3]. This option facilitates the performance of the entire ecosystem, which can now carry out a given task in a highly coordinated manner. However, validating their performance can be a complex process, as both systems update and interact using one representation. In addition, the two systems lose their autonomy, making it difficult for them to tackle tasks independently (e.g., when the robot decides to interact with the person, it will have, in the representation, the knowledge that the IoT system is capturing from all the rooms it monitors).

The option we explore in this article is different: maintain the independence of both systems and build a cyber-physical system of systems (CPSoS). This CPSoS is built on the basis of two constituent systems (CSs) that maintain their independence. The IoT system is equipped with sensors and computing elements, making it capable of monitoring the condition of a person living in a small apartment, and an assistive robot capable of interacting with the person and assisting him/her in certain daily tasks. Both systems are deployed independently and provide services autonomously. In this proposal, both CSs will be controlled by a common target (directed CPSoS [4]) but will maintain their ability to work independently. This article focuses on the integration of both CSs at the knowledge level. Both systems maintain their own state representations, which are built and maintained within each CS, and there is a flow of knowledge that moves information between them to achieve the global objectives for which the CPSoS will be designed. The Deep State Representation (DSR) graph [5] will be used as the state representation in both systems, and the two CSs will also be organised according to the guidelines of the CORTEX software architecture [5,6].

To demonstrate the validity of the proposal, a couple of demonstrators have been designed. In the first one, the metric map that the robot has to trace routes that allow it to move to a target is augmented with the knowledge that the IoT system can provide about the positions of chairs or other moving objects or the open/closed state of doors. This allows the robot to determine when a route to a target is not possible and to ask the person to help it move a certain chair or open a door. In the second demonstrator, the presence of a fallen person on the floor is detected by the IoT system, causing the robot to abandon the task it is performing and attend to this emergency. The IoT system has, in this case, its own ability to detect the situation, while the robot, once alerted, can also operate autonomously.

1.1. Contributions

The most common solution when integrating a robot into an AAL environment is to consider it as just another device in the ecosystem, connected, like all other deployed devices, to a single, centralised knowledge representation. The aim of our work, however, is to evaluate the possibilities that arise when considering the robot and IoT system as independent elements that maintain their own models or representations of the context. In both entities, the software is organised using the CORTEX architecture [5,6], using a directional graph as a runtime model. From an implementation point of view, both representations share knowledge by encoding it in topics of the ROS 2 robot operating system. The proposal has been validated with several use cases implemented in a small flat. This article shows two examples in which the IoT system and robot share knowledge in order to solve a given task.

1.2. Organisation of the Paper

The rest of this paper is organised as follows: Section 2 discusses previous work in which a robot was, like a CS, part of a CPSoS. Previous work in which CORTEX was used in assistive environments is also presented. Section 3 describes the proposed new framework for representing knowledge in which the robot and the IoT system remain independent elements, sharing a continuous flow of information. The experimental setting and the results obtained are presented in Section 4. Finally, Section 6 presents the conclusions and future work.

2. Related Work

A system of systems (SoS) is defined as a set of autonomous and independent systems (so-called constituent systems (CSs)) providing a certain service [7]. Building a system of systems has the advantage that, in addition to the services that each system provides, new services emerge, which are based on the collaboration between the different systems deployed [8]. Briefly, a cyber-physical system (CPS) is a physical system controlled or monitored by computer-based algorithms. The physical deployment will typically have sensors and actuators distributed in a given environment. The computational deployment will also typically be distributed, with processing nodes connected to a network, which will manage the exchange of information. Decision-making will thus be supported by the internalisation of information captured from the real environment. A cyber-physical system of systems (CPSoS) is an integration of independent CPSs and is capable of providing services that can go beyond those provided by these same CPSs acting in isolation [4].

With the popularisation of the Industry 4.0 concept, the physical reality of a factory has come to be organised as a CPS. It is not uncommon that, in order to carry out intralogistics, pick-and-place, or assembly tasks, such a CPS integrates an autonomous robot [9]. Assistive living environments (AAL) are also examples of CPSs. The work of Bocicor et al. [10] describes an AAL system following the guidelines of a CPS with a wireless sensor network, into which the home monitoring software is integrated. In the CPS proposal by De Venuto et al. [11], the network includes wearable sensors. The proposal by Calderita et al. [3] presents a CPS-AAL for caregiving centres. In addition to static sensors or actuators (environmental (to measure temperature, humidity, CO_2…) or person-centred (microphones, loudspeakers, cameras…)), the system integrates a Social Assistance Robot (SAR). The integration of the robot into an AAL ecosystem allows it to offer a wide range of services. In the GiraffPlus project [12], sensors deployed in the environment provide insight into the person's emotional state. When nervousness or anxiety is detected in the person, the robot is used as a way to put them in contact with family or friends. In the MORPHIA project [13], the robot is also used to connect users and caregivers. In addition, the robot can remember whether a person needs to take medication, transport staff items from one part of the house to another, or help family members see, via the robot's cameras, how things are going at home. Using sensors in the house and on the robot, the RiSH proposal [14] aims to monitor the movement and activity of the person at home. The detection of normal or abnormal behaviours in the person's activity is the basic objective of the work of Mojarad et al. [15]. Integrated into the AAL ecosystem, a robot is also employed as a virtual therapist, which proposes physical or cognitive exercises.

In the proposal by Lin et al. [16], the robot is a CPS whose architecture follows the 5C architecture model and is structured in five layers (Component, Intelligence, Cyber, Configuration, and Deployment) (see Figure 1). In contrast to the previous examples, where the sensor network is the core of the CPS and the robot is just another element, in this proposal, the robot is the centre of the CPS, and there are external computational elements, based on Raspberry Pi4 and Intel NCS2, for intelligent inference and data exchange. In general, a moderately complex robot will include various processing elements (CPU, GPU, etc.) as well as sensors and actuators. All of these will need to be connected and organised to perform different tasks. In addition, in certain cases, it will be interesting for the robot to be able to abstract from reality, using models to evaluate how a certain task can be approached (reaching a position, picking up an object, etc.) or a digital twin of itself to assess whether a mission can be carried out successfully [17]. When the robot and the environment maintain their own structures and tasks, the solution can be considered a CPSoS [18].

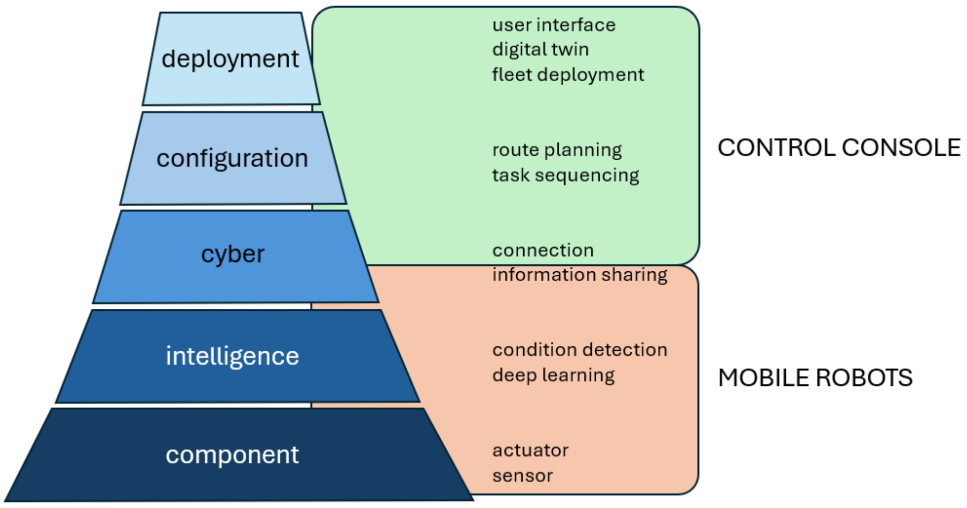


Figure 1. The five-layered CPS architecture of the robot proposed by Lin et al. [16].

As discussed above, different architectures have been proposed to handle the complexity of a CPS. The 5C architecture is a hierarchical approach that organises information processing and management into layers, from the raw data captured by sensors to the most complex abstract processing. When it comes to developing systems at design time that are capable of working at run time in a dynamic environment and in an autonomous manner, as in our case, the concept of self-adaptation must be considered. To introduce this concept into a CPS, the MAPE-K (Monitor, Analyse, Plan, Execute, and Knowledge) control loop was proposed [19]. In fact, when the complexity of the tasks is high, it may be necessary to integrate several MAPE-K control loops, which are coordinated to achieve the correct execution and adaptation of the system [19,20]. In the CPS-AAL proposed by Calderita et al. [3], a more flexible mechanism is proposed in which reactive and more deliberative processing work at the same level. The robot can evaluate how to get to a certain position and, at the same time, respond to a person waving at it as it passes by. As in Calderita et al.'s work, this article will make use of the CORTEX architecture [5,6] to manage a CPS. Identifying the blocks that define the MAPE-K loop in CORTEX, Romero-Garcés et al. [20] proposed a self-adaptation scheme that implements several control loops to control the operation of an autonomous robot operating in an intralogistics environment. However, unlike previous systems, in our case, the robot and the IoT system remain autonomous elements, each of which has its own architecture, including software agents or memories. Both CPSs collaborate to solve certain tasks.

3. Shared State Representation

Knowledge representation and reasoning is a symbolic branch of artificial intelligence that aims to design computer systems that reason using a machine-interpretable representation of the world, similar to human reasoning [21]. In this sense, knowledge is more than information, as it incorporates the agent's own experience or intuition. A knowledge-based system will have a computational model of the real world of interest [22], in which physical and virtual objects, events and actions, relationships between concepts, etc., are represented by symbols (and geometric information if necessary).

As previously mentioned, one of the main advantages of a CPSoS is that behaviours can emerge that, in individual CSs, could not. For this, it is important that CSs exchange knowledge with each other, and not merely information about their internal state [4]. In order to use these shared models, CSs must share the same formal definitions of types, properties, and relationships between entities.

In our proposal, we focus on runtime information exchange. Both the robot and the IoT system will internally use a graph-based runtime model [20]. The structure of this model is presented in Section 3.1. The CORTEX software architecture, which is built to handle such a model, is also briefly presented. Then, Section 3.2 formalises our proposal.

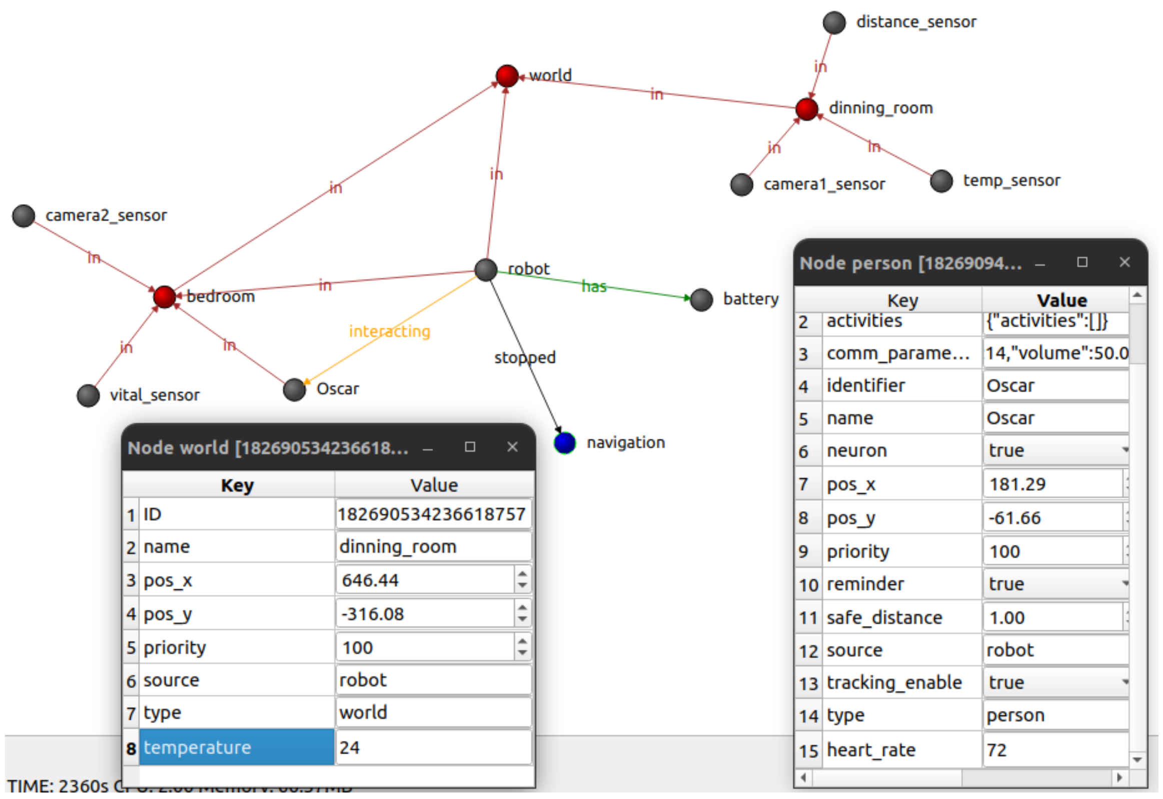
3.1. CORTEX and the Deep State Representation (DSR)

CORTEX is a cognitive robotic architecture inspired by two main ideas: modularity and internal modelling [5]. Briefly, we can describe CORTEX as a collection of agents that cooperate to solve a task. Communication between these software agents, whether reactive or deliberate, is carried out by annotating knowledge in a working memory. This working memory, called Deep State Representation (DSR), behaves as a runtime model of the knowledge acquired by the robot [20]. As a model, it includes both symbolic and geometric concepts, both of which are collected in a single graph that, mathematically, can be considered to be made up of two quivers. One contains symbolic relations and the other geometric relations, defined by rotation–translation transformation matrices.

CORTEX can therefore be classified alongside other proposals based on the blackboard concept. In CORTEX, this idea is rigidly adhered to: all relevant knowledge is written down in the DSR so that it is available to all other software agents. If a perceptual agent updates the position of the person in the DSR, the agent guiding the robot as it approaches this person uses the updated data to drive the approach movement. But, the one monitoring the person's activity also uses this same information to determine the person's daily routine. Coordination between the agents themselves is based on annotations they make in the DSR [20].

The DSR is therefore the central element of CORTEX and has been considered a digital twin of the external world itself [3]. However, the DSR is only a working memory, to which the system must link other longer-term memories (e.g., a map of the environment or information about residents in a nursing home).

Conceptually, the DSR is the runtime model that the system internalises from internal and external contexts [20]. In this working memory, which is temporally anchored to the current instant of time, entities are represented (not only physical realities (objects, rooms, people. . .) but also actions, intentions, sensations. . . [6]), along with the relationships that the system establishes between them, whether geometric (transformation matrices) or symbolic. This model is mathematically defined as a directed graph, in which there can be different links (relationships) connecting two nodes (entities). Both nodes and links have attributes, but if the link expresses a purely geometric relationship, it will only contain a transformation matrix (rotation–translation, RT). Figure 2 shows a snapshot of the state of the DSR at a certain instant in time. Specifically, the captured situation shows that the robot is standing still and interacting with Oscar. Both the robot and the person are in the bedroom. Each node in the network stores information captured by sensors installed in the environment or onboard the robot. The temperature of the dining_room, for example, is 24 degrees, and Oscar's heart rate is 72 bpm. In this example, the DSR is common to the robot and IoT environment.



Node world [182690534236618...

	Key	Value
1	ID	182690534236618757
2	name	dinning_room
3	pos_x	646.44
4	pos_y	-316.08
5	priority	100
6	source	robot
7	type	world
8	temperature	24

Node person [18269094...

	Key	Value
2	activities	{"activities":[]}
3	comm_parame...	14,"volume":50.0
4	identifier	Oscar
5	name	Oscar
6	neuron	true
7	pos_x	181.29
8	pos_y	-61.66
9	priority	100
10	reminder	true
11	safe_distance	1.00
12	source	robot
13	tracking_enable	true
14	type	person
15	heart_rate	72

Figure 2. A snapshot of the DSR. In this example, the robot is included within the IoT ecosystem.

Connected to the DSR is a collection of software agents, which are responsible for perceiving specific aspects of the context, whether external (identifying a person, recognising their emotions. . .) or internal (battery level of the robot), for making decisions, evaluating which tasks the robot should now carry out, or acting (on the real world or a model of it). The way in which a software agent accesses the knowledge updated in the DSR is relatively simple, allowing the design of interfaces with different operating systems or frameworks. This facilitates the deployment of sensors and actuators and the integration of computational algorithms, as the agents that manage them can be implemented in virtually any software framework. In fact, in the proposal by Romero-Garcés et al. [20], these software agents are identified as the constituent blocks of MAPE-K control loops. The DSR is the knowledge component used for synchronising the behaviour of the MAPE loops that coexist simultaneously in the robotic software architecture.

Figure 3 provides an overview of the instantiation of CORTEX in the CPS-AAL proposal [3]. The figure shows the DSR in the centre, and around this memory, services are distributed that are provided by the robot or by the IoT system. It is important to note that this figure does not correspond exactly to how CORTEX is organised internally. In CORTEX, around the DSR, agents would be placed that implement basic functions, such as navigating, talking, listening, measuring temperature or humidity, capturing heart and respiratory rates, etc. In this figure, however, the emphasis is on how both elements (robot and IoT system) share the same knowledge base and how this representation enables them to carry out different tasks.

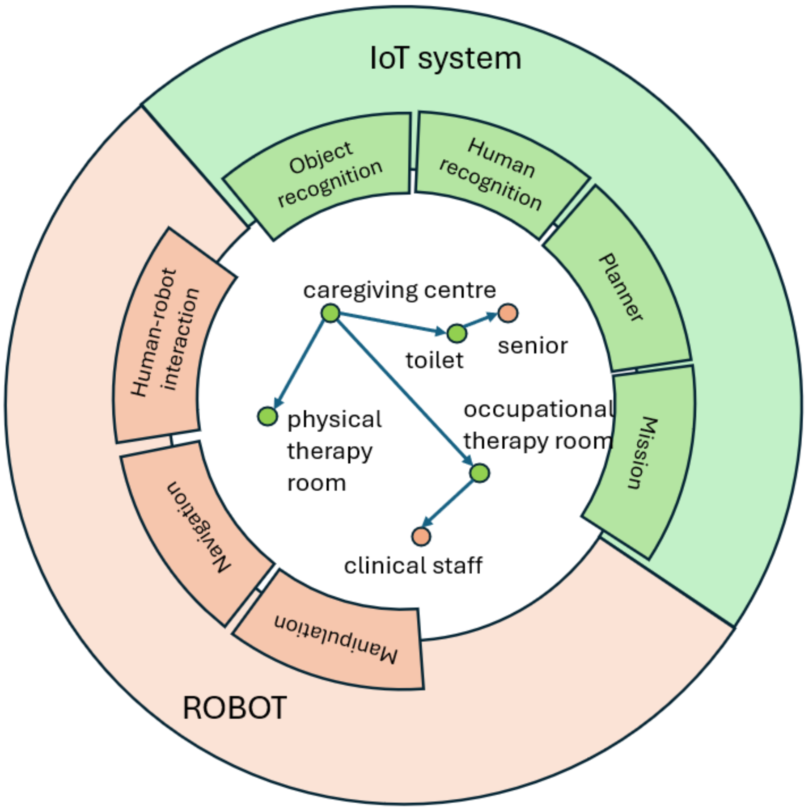


Figure 3. The CORTEX instantiation in the CPS-AAL proposal [3]: the IoT system and robot share a common DSR.

3.2. Sharing Information Between DSRs

In the implementation described in this paper, there are two distinct representations or DSRs, the one in the robot and the one in the IoT system, which must serve to enable each of these systems to carry out the tasks for which they have been designed. Thus, the robot must navigate the environment or interact with the user, and it is important that this working memory manages obstacles or maps of the environment or the user's interaction priorities. The environment monitors the person's daily routine and vital signs, such as heart and respiratory rates, as well as parameters of the environment itself (air quality, temperature). Knowledge transfer between these two CSs is carried out by exchanging parts of these two models. This transfer can be used to enable each element to optimize how it performs a certain task. For example, the IoT system can know whether doors are closed or open or know the positions of chairs, tables, or people in the environment and then transfer these data to the robot. With these data, the robot can augment the metric map of the environment that it uses to navigate, achieving routes that avoid obstacles (before the robot itself sees them) or asking for help from the person to push chairs aside or open doors for it. Figure 4 outlines this idea. The IoT system has sensors that can detect the person in any room of the environment. This information is transferred to the robot, and when the robot needs to remind the patient to take medicine, it can go directly to this room and look for the person. In our case, the transfer is continuous and involves certain entities. It is not that both representations are the same, but rather that they contain the same information with respect to certain topics. But, one could study an on-demand transfer, where one of the systems asks the other for certain information only when it needs it (in the example above, the robot would ask the IoT system for the room the person is in when it needs to remind him to take his medicine).

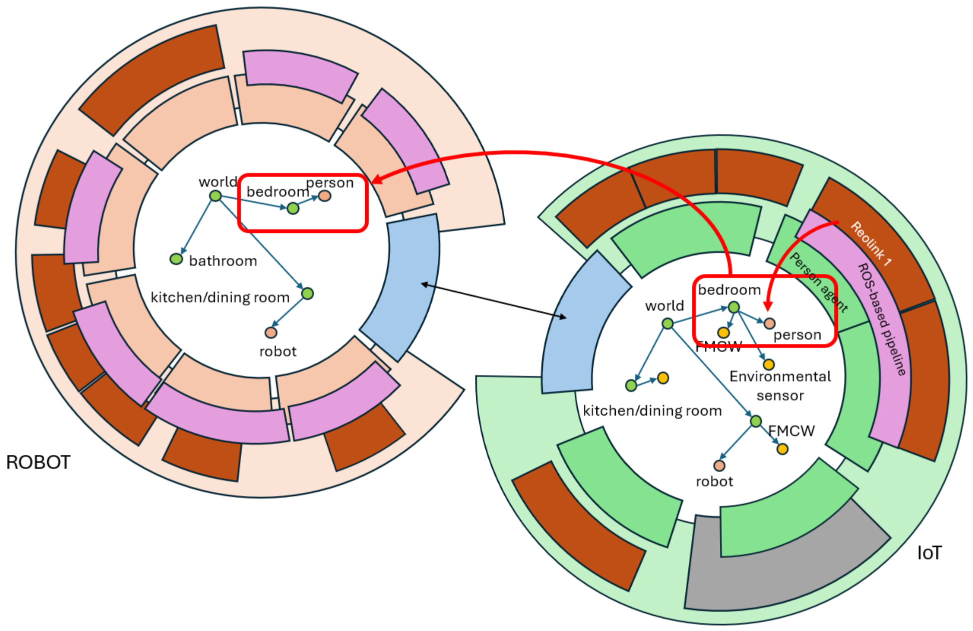


Figure 4. The IoT system and the robot share a state representation but maintain their own working memories.

Considering that both working memories are initialised with the room layout of the flat, the information transfer scheme is based on the design of agents developed in ROS 2 (*dsr_bridge*) that are instantiated in both architectures. The entities to be transferred will therefore be topics defined in ROS 2, and FastDDS will be used as the communication middleware. The knowledge transfer protocol works as follows:

- Both *dsr_bridge* agents maintain the DSR as a shared pointer, which allows them to listen for changes in the network and act as a ROS node. Depending on the tasks that have been set at a high level (e.g., person monitoring), the *dsr_bridge* activates certain communication topics. These topics are linked to nodes in the DSR. In the example of monitoring the person's state, any information that is updated in either DSR about the person node (attributes such as heart rate and respiration rate (captured by the IoT system) or the emotional state (whether the robot has spoken to the person and collected this information)) is transferred between the two DSRs. Currently, this mapping between tasks and topics is set manually at design time.
- When a modification to be transferred is detected in the DSR, it is published using ROS topics (network nodes and arcs). A 'source' attribute is added to the message that determines which system publishes the change (the robot or IoT, in our case). This avoids auto-listening to published messages.
- When one of the agents publishes a change, the other agent reads it and updates its DSR with the knowledge included in the message (node name, type, attributes, link). The change may involve adding or deleting a node or arc.
- If the published message involves a modification to the DSR (i.e., updating an attribute of an existing arc or node), the receiving agent will carry out the relevant checks (i.e., to ensure that the node or link whose attribute is to be modified exists).

This mechanism allows the alignment of the entities in the DSRs of the robot and IoT system. In all the tests carried out, the knowledge exchanged has been available to both entities almost simultaneously.

The main problem with this scheme is that conflicts may appear. Geometrical information is particularly sensitive to such conflicts, as a goal location could be different for the robot and the IoT system. To avoid this problem, making use of a ROS facility, both entities update a common reference. All messages in the ROS have a header, with its timestamp and frame associated with that message, and all frames are referenced to a frame 'map', directly or indirectly. The reference is framed in a ROS transformation tree (TF-Tree, http://wiki.ros.org/tf, accessed on 15 November 2024), which models the direct kinematics of the system. In the robot, there will be a node 'robot_state_publisher' that publishes the static transformations of the internal elements of the robot (camera, lidar, motors. . .) from a URDF (Unified Robot Description Format) file. In the IoT system, there is a similar node that publishes the static TFs of the camera with respect to the map. The relationship of the TF tree is completed by the navigation stack, which publishes the TF between the map and the odometry of the robot.

4. Experimental Evaluation

4.1. Experimental Setting

The AAL environment has been deployed in a small, three-room apartment set up within our Research Laboratory. The apartment has a kitchen/dining room, a bathroom, and a bedroom. The layout is shown in Figure 5. To monitor the movement of the person in the dining room and bedroom, two Reolink 360 panoramic cameras (Reolink, Hong Kong, China) are used, located about 2.4 m above the floor. The person's heart and respiratory rates are extracted using 60 GHz FMCW (frequency-modulated continuous wave) radar (the MR60BHA1 from Seeed Studio, Shenzhen, China). This device has been placed under the bed to capture these vital signs when the person lies in bed and thus also monitor the quality of sleep, as well as behind the back of an armchair in the dining room. There are also magnetic sensors to monitor the states of doors and windows, as well as a sensor that monitors environmental variables (temperature, humidity. . .) and a presence sensor in the bathroom.

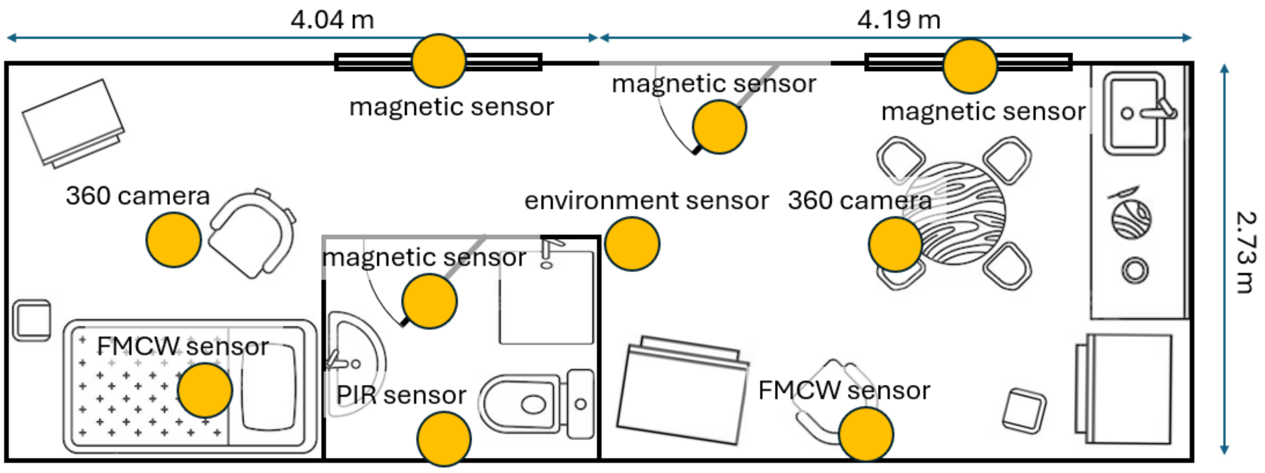


Figure 5. The layout of the small apartment and the distribution of sensors.

4.1.1. The Morphia Robot

The robot used in these tests is a Morphia from MetraLabs GmbH (Ilmenau, Germany). This robot is built on the MetraLabs TORY differential base and includes a circular bumper around its entire base and a SICK s300 range laser scanner (Sick AG, Waldkirch, Germany). For navigation, it also has an Intel RealSense D435i RGB-D camera (Intel, Santa Clara (CA), USA). Perception of the environment is captured using three 2MP Valeo cameras (Valeo SA, Paris, France) and a Microsoft Azure Kinect RGB-D camera (Microsoft Corp., Redmon (WA), USA). This equipment is completed with devices that allow interaction with the user (a tablet, speakers). Figure 6 shows the Morphia robot navigating in the small apartment. To handle all of these sensors and actuators, the robot is equipped with an Intel NUC i7 (Intel, Santa Clara (CA), USA) and an NVIDIA Jetson Orin AGX (NVIDIA Corp., Santa Clara (CA), USA).

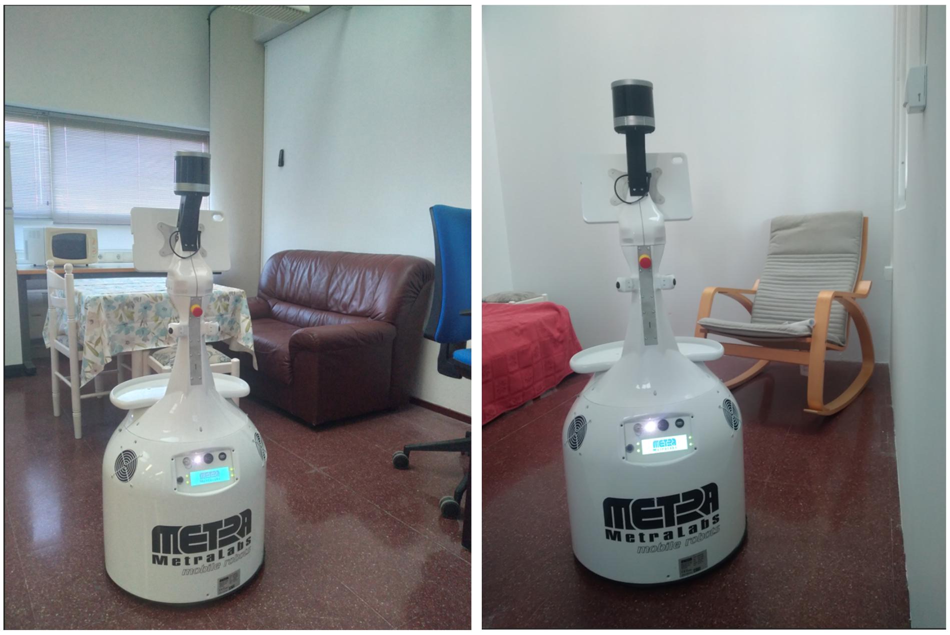


Figure 6. The Morphia robot navigating in the small apartment.

The implementations of the CORTEX architecture running in the IoT environment and the robot are shown in Figure 7. The CORTEX architecture in the robot includes actuators (such as motors, an emergency button, or speakers) and sensors (the aforementioned RGB-D cameras, the SICK s300 laser, 2MP cameras, and a touchscreen). These elements are managed by modules developed in ROS 2 (the navigation and docking stack [23]) (https://www.ros.org/, accessed on 15 November 2024), RoboComp (Speech) (https://robocomp.github.io/web/, accessed on 15 November 2024), and MIRA (the Scitos base management stack) (https://www.mira-project.org/, accessed on 15 November 2024). Thus, the CORTEX architecture in the robot has the software agents that the robot needs to navigate the environment (motor controller or battery monitoring is handled by MIRA, while the navigation stack is implemented in ROS 2). The robot also has its own agent to detect people and to communicate verbally with a person. A micro-ROS agent (https://micro.ros.org/, accessed on 15 November 2024) allows a panic button to be connected to the DSR. A WebServer agent allows it to communicate with the Chest screen. Decision-making is based on behaviour trees, which encode each of the tasks that the robot can tackle. However, in order to decide, at all times, which task to prioritise, the system includes a self-adaptation system that monitors context variables to weigh the different tasks and propose one [20]. As proposed in [20], the self-adaptation and decision-making processes implement several MAPE-K control loops.

Thanks to the fact that we have a robot deployed in the Vitalia Teatinos (Malaga, Spain) nursing home for elderly people [24], the results that these agents offer are, within the framework of their interaction with a person, in our case, immersed in a continuous process of redesign. Specifically, Table 1 shows the feedback captured through questionnaires administered to eight real users of this residence. Responses are captured using a Likert scale from 1 to 5, with 5 being the most positive response. Some are more likely to interact with the robot and others are more reluctant (in fact, as shown in the table, not all questions are always answered). When shortcomings are detected, these are corrected, and the results are recorded in the next survey.

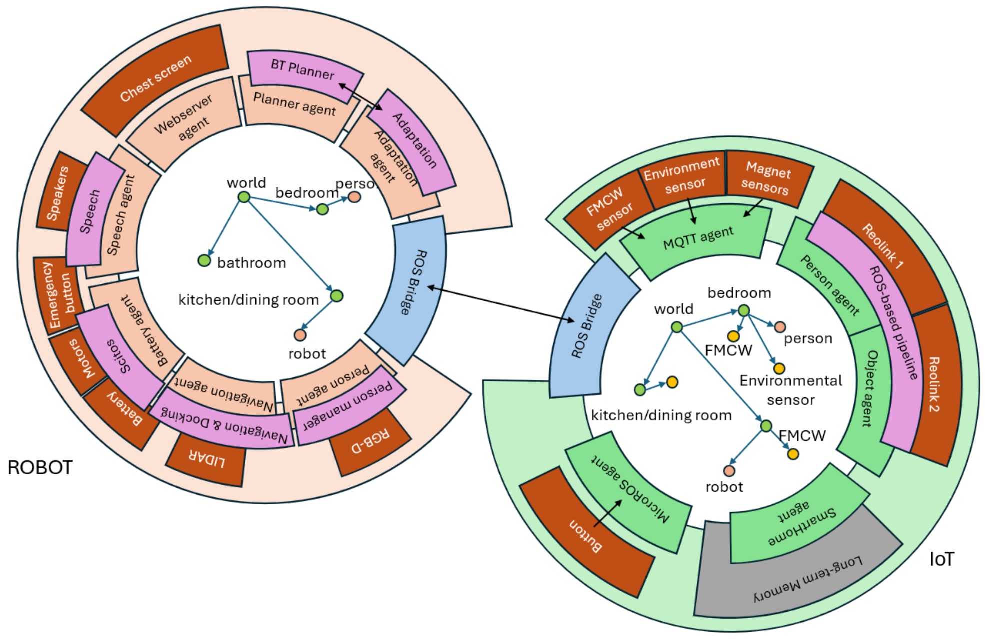


Figure 7. A schematic view of the two CORTEX architectures.

Table 1. Feedback from eight real users on different aspects related to their interactions with the robot (Vitalia Teatinos residence (Malaga, Spain), October 2024).

Person ID	(Perception) I Can Hear What the Robot Says at All Times	(Perception) I Can See What the Robot Shows on the Screen at All Times	(Operation) I Was Able to Interact with the Robot Through the Screen Without Any Problems	(Understanding) I Understood the Robot at All Times
1	4	5	3	4
2	5	5	3	5
3	5	3	2	4
4	5	5	2	5
5	5	5	2	5
6	5	5	5	5
7	3	3	–	1
8	4	–	–	2

4.1.2. The IoT System

The software agents running the IoT system are shown in Figure 7. The software architecture supporting the IoT system has an MQTT agent, through which data are received from frequency-modulated continuous-wave (FMCW) radar sensors, magnetic sensors, and environmental sensors. The FMCW sensors capture the heart and breathing rates of people lying in bed or sitting in a chair in the kitchen/dining room. Environmental sensors provide measurements of temperature, humidity, and air quality. Magnetic sensors indicate whether the two flat doors (main entrance and toilet) are open or closed. A PIR Motion sensor is used to detect the presence of a person in the bathroom (a HCSR501 sensor, DFRobot, Shanghai, China). These devices are equipped with an ESP32C microcontroller, with WiFi output, which collects the data and sends them via MQTT. Figure 8 shows what the FMCW sensor looks like. The time sequence of data captured from the environment is acquired from the DSR and stored in long-term memory (a database built using InfluxDB (https://www.influxdata.com/, accessed on 15 November 2024).

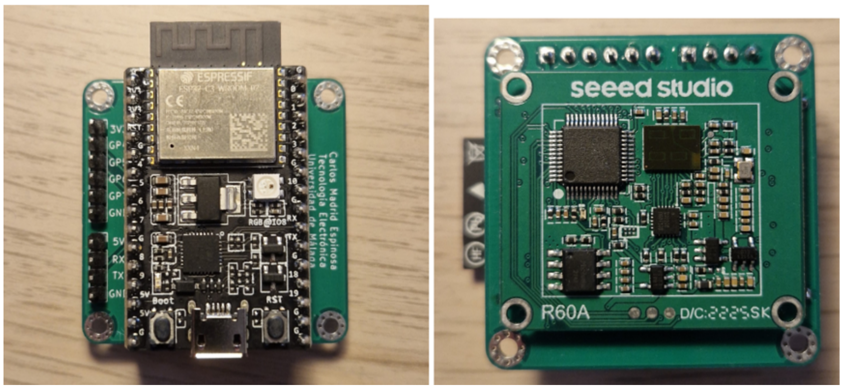


Figure 8. The device designed for mounting the FMCW 60GHz sensor (MR60BHA1) from Seeed Studio and the ESP32C3 microcontroller: (**Left**) The top layer showing the ESP32C3 (Espressif Systems, Shanghai, China) and (**Right**) the bottom layer showing the MR60BHA1 sensor.

The two 360 cameras provide information about the position of the person(s) in the apartment and about other objects in the two main rooms (see Figure 9). To perform this task, we tested different versions and sizes of yolo-object-detection models, but none of them provided correct detection results (many false positives and confusion between classes) due to the zenithal view of the cameras. For this reason, we trained a yolov11 (large) model (currently the latest version) on a custom dataset consisting of 1263 labelled images and 3031 images after data augmentation. The model was trained for 125 epochs, but in the last 10 epochs, data augmentation was turned off (this explains the final drop in loss functions). All metrics are shown in Figure 10. After this step, our 360 cameras could detect people, their positions (sitting, standing, or lying down), and a set of objects (the main ones presented in the small flat).

However, in our representation, objects and people are categorised as three-dimensional (3D) objects. To achieve this, the information from the 360 cameras is processed using the ROS 2 perception pipeline shown in Figure 11. The first step is to rectify the images captured by the cameras. This step is carried out by the *RectifyNode* node. The rectified image is used to detect people and objects (*object_detection* node) and to estimate depth (*depth_anything* node). For the latter, the Depth Anything model is used [25]. Detections and depth are merged by the *detection_to_3D* node. Information about people is extracted by the *person_posture_manager* node and updated in the DSR by the *person-agent* node. From an implementation point of view, we have created a ROS 2 wrapper for the yolo models, composed of a node with an image subscriber (original RGB image), a Detection2DArray message (https://github.com/ros-perception/vision_msgs/blob/ROS2/vision_msgs/msg/Detection2DArray.msg, accessed on 15 November 2024) publisher, and a marked-detections image publisher (bounding boxes drawn on the original RGB image). The bounding boxes associated with the detections are matched with a depth image obtained using our ROS 2 wrapper for the Depth Anything model (https://github.com/grupo-avispa/depth_anything_v2_ROS2, accessed on 15 November 2024). The *detection_to_3D* node is developed with a synchronised subscriber (for managing 2D detections and the depth image at the same time) and a Detection3DArray message publisher (https://github.com/ros-perception/vision_msgs/blob/ROS2/vision_msgs/msg/Detection3DArray.msg, accessed on 15 November 2024). Information about people is extracted from this Detection3DArray message, and a set of attributes is selected to be published in a custom person message. It can be observed that all nodes in this perception pipeline have been converted to ROS 2 composable nodes and encapsulated into a single container. We include this step because all composable nodes, so-called components, run under the same process, and when a subscription and a publisher of a topic reside in the same process, the underlying middleware can be completely short-circuited; messages are passed via smart pointers pushed into a shared ring buffer. This allows for zero-copy communication, which saves computational resources and reduces latency [26].

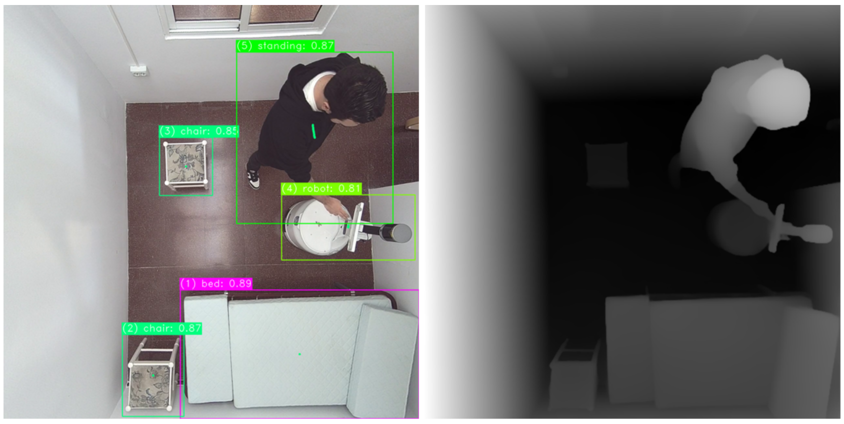


Figure 9. (**Left**) A rectified image showing a person interacting with the robot in the bedroom and (**right**) the associated depth image.

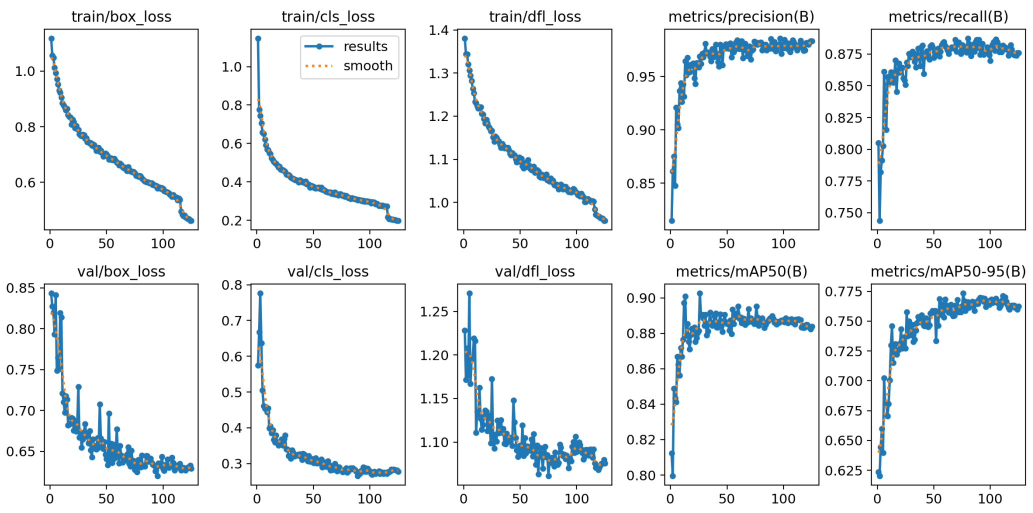


Figure 10. Resulting metrics from yolov10 model training.

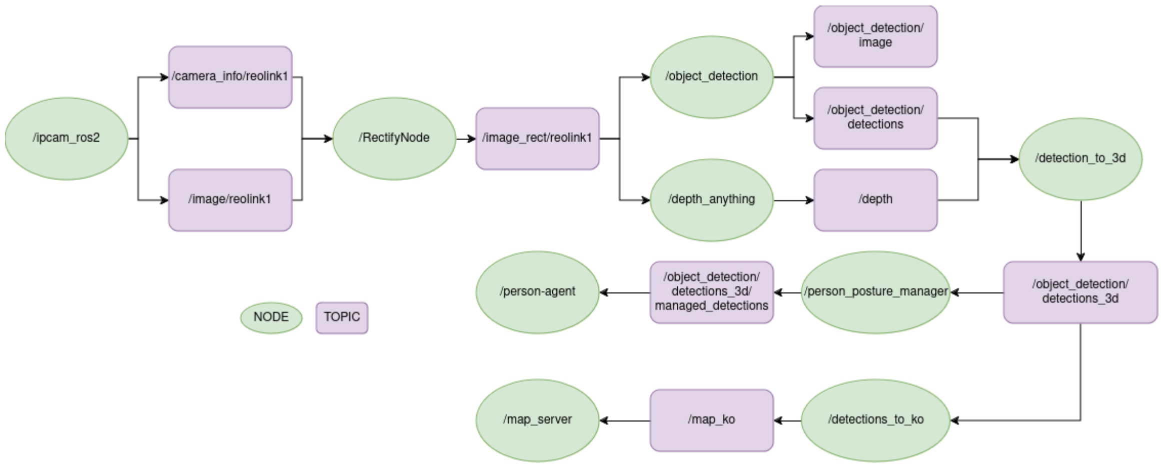


Figure 11. ROS 2 perception pipeline.

Both architectures communicate their CORTEX architectures using the aforementioned ROS bridges.

4.2. Enhanced Robot Navigation

The described framework was first evaluated in a simple use case: employing the information provided by the IoT system to augment the metric map used by the robot to navigate the environment.

Currently, the robot navigates autonomously using a custom version of the ROS 2 navigation stack 'Nav2' [27] that is managed by a software agent in CORTEX. The Nav2 stack provides the foundational capabilities for autonomous robot navigation, integrating planning, obstacle avoidance, and real-time adjustments to ensure efficient movement through an environment. Nav2 is designed as a modular system, where each component performs a dedicated role in navigation. This allows the robot to navigate with flexibility, adapting to changing surroundings.

Nav2 relies on several key components working in tandem. First, it uses a global planner, which calculates an optimal path from the robot's current position to the target, considering static obstacles and the map layout. This path is continuously updated as new information is received. Alongside the global planner, a local planner manages real-time adjustments to the path, handling dynamic obstacles and ensuring smooth motion. The local planner provides commands directly to the robot's motion controller, allowing it to adapt quickly to changes in the environment.

Another essential component is the behaviour tree architecture, which provides flexible decision-making for the robot. Through behaviour trees, Nav2 manages navigation tasks, like goal-reaching and recovery behaviours, allowing the robot to respond to various situations (e.g., being stuck or encountering an obstacle) autonomously. Additionally, the map server and costmap are integral; the map server loads the environment map, while the costmap dynamically represents both static and moving obstacles, making it possible to continuously adapt navigation.

The navigation stack also integrates with sensor fusion and localisation components. For instance, it uses AMCL (Adaptive Monte Carlo Localisation) to keep track of the robot's position on the map by fusing data from sensors like lidar. This localisation, combined with robust path planning and obstacle avoidance, allows the robot to operate reliably in complex environments.

The stack uses a metric map of the environment—initially captured during design—to plan paths toward goals. This metric map includes both a global costmap, which spans the entire environment and accounts for static elements such as walls, worktops, and wardrobes, and a local costmap, which dynamically updates to account for obstacles like people or movable furniture within a 5 m radius of the robot.

Within the framework of the tasks to be performed by the robot, this stack enables it to navigate to the required positions in the flat. The robot can plan a path to these positions using the static, global metric map of the environment and then solve the problems it encounters on the fly using the local map. However, given the small sizes of the rooms in the flat (which can also be found in a typical house), it is very often the case that the paths traced by the robot using the global map are not valid afterwards. Sometimes, the robot finds the doors closed (in the global map, they are always open). In other cases, the movement of chairs or tables prevents it from reaching the target, and it does not find alternative routes. In these cases, the robot spends a significant amount of time trying to find these alternatives, which, in reality, do not exist. It is worth noting that the IoT system does know the status of doors and the distribution of objects in the environment. If this information is transferred to the robot, it can have an augmented map of the environment, in which the position of not only the person but also moving obstacles such as chairs or tables can be located, in addition to knowing whether doors are open or closed. In this way, the robot could know in real time whether there are available passageways. If it needs to ask the person to open a door that is closed or to move chairs blocking its path, it can proactively request this help as a preliminary step to navigate to the desired position.

In the framework of the proposed CPSoS, the robot's DSR is updated with the semantic information of the objects present in each room (chairs, tables, people) or the opening/closing of doors, as well as with the necessary geometric information (i.e., the projection on its own metric map of the position of each detected person or object). This geometric information is placed on top of the global costmap as a polygon filter layer and allows the robot to know whether, for example, chairs are in the path towards the medicine dispenser in the kitchen. In order to find out how the objects detected by the cameras are projected on the two-dimensional ground plane, we use depth estimation using the aforementioned ROS 2 wrapper for the Depth Anything model (see Figure 12). Using the detections and depth estimation, the IoT system can detect the various objects or people present in the rooms and project them onto the global metric map. All items (cameras, 3D detections. . .) are located in the same 3D framework.

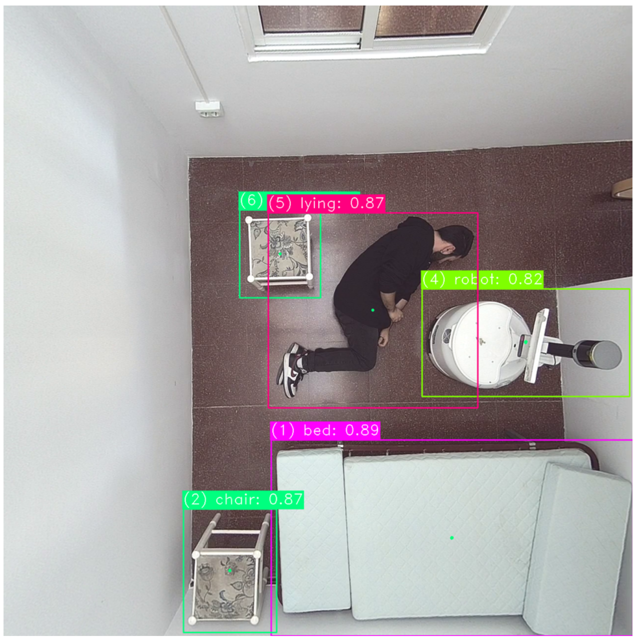


Figure 12. The detection of a person lying on the floor and objects (chairs and a bed) in the bedroom.

Finally, Figure 13 shows the metric map augmented with information on the objects/people present in the two main rooms of the flat and the status of the two doors. As mentioned above, this map lets the robot know that a certain object, or a closed door, makes it impossible to reach a certain goal. In Figure 13(top), the door is open. The robot knows the positions of people, chairs, and tables and plans a path to the bedroom. When the robot starts to navigate, the main door of the house closes (Figure 13(bottom)). The IoT system immediately detects this situation, and this knowledge is transferred to the robot, which stops and should inform the person that it cannot continue its mission unless the main door of the flat is opened.

It is important to note that the robot will have this information in real time, as well as the possible relationships that the IoT system detects between the objects and people present in the flat. If the mission is to locate the person to remind him/her to take a certain medicine, the geometric position of the person will be available in real time as long as he/she is in one of the two monitored rooms. The robot will also be able to know whether the person has entered the bathroom and must wait at the bathroom door for the person to come out. In the following example, this knowledge will be used to come to the person's aid as soon as a possible emergency is detected.

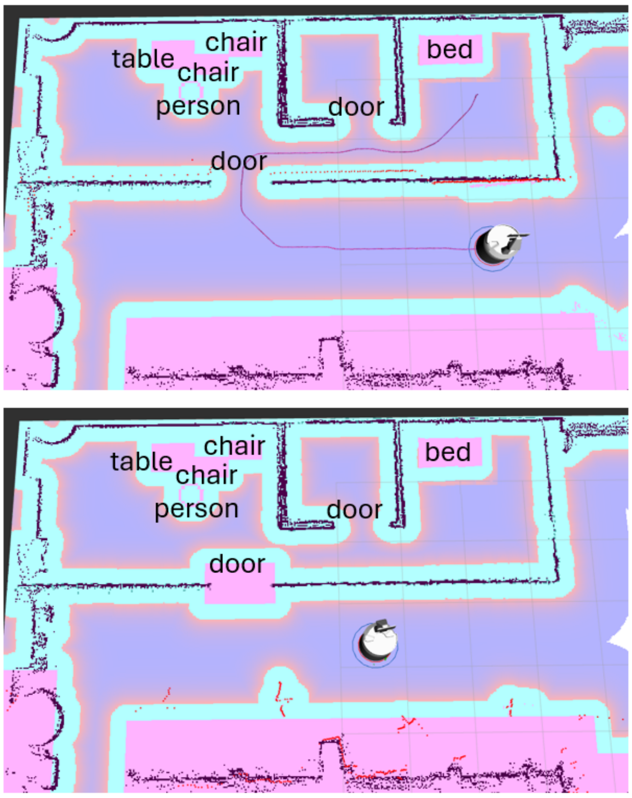


Figure 13. (**Top**) The robot plans a path towards the bedroom. There is no problem navigating this route. (**Bottom**) When the robot is moving, the main door is closed. This event is immediately detected by the AAL system and communicated to the robot. The robot must ask the person in the home to open this door.

4.3. Care for Fallen Persons

In this use case, the overall objective of the complete CPSoS is to properly handle a situation in which a possible fall of the person at home is detected. The IoT system is responsible for the detection of this situation, for which it will make use of two agents integrated into its software architecture: the Person agent and the SmartHome agent. As previously mentioned, the first one will be responsible for detecting, using 360 cameras, the presence of a person in a certain room. The second one manages a database that stores the person's activity and routine (when the person enters or leaves a room and whether the person has been sitting, standing, or lying on the bed in the bedroom). It will also be responsible for detecting that the person is lying down but not in bed, generating an alarm that is annotated in the IoT system's DSR. This alarm is shared between the IoT system and robot representations, so its occurrence triggers the use case in the robot to address this risk situation.

In the robot, attention to the person's fall alarm is encoded in a behaviour tree, the execution of which is prioritised over any other task the robot is executing. This task prioritisation is implemented in the Adaptation agent, mentioned when briefly presenting the CORTEX architecture running on the robot (more details are provided in [20]. The behaviour tree is shown in Figure 14. Basically, the robot captures the position of the person who may have fallen and navigates to it. Once it is next to the person, a message is displayed on the touch screen on the robot's torso asking if the person is OK. The same question is asked by voice. If, after a reasonable time (initially set to 30 s), the robot receives no response, it sends an emergency e-mail to the caregiver or contact person, and an audible alarm is triggered. The way in which the robot handles the situation (number of times the person is asked, waiting time) can be easily modified in the behaviour tree.

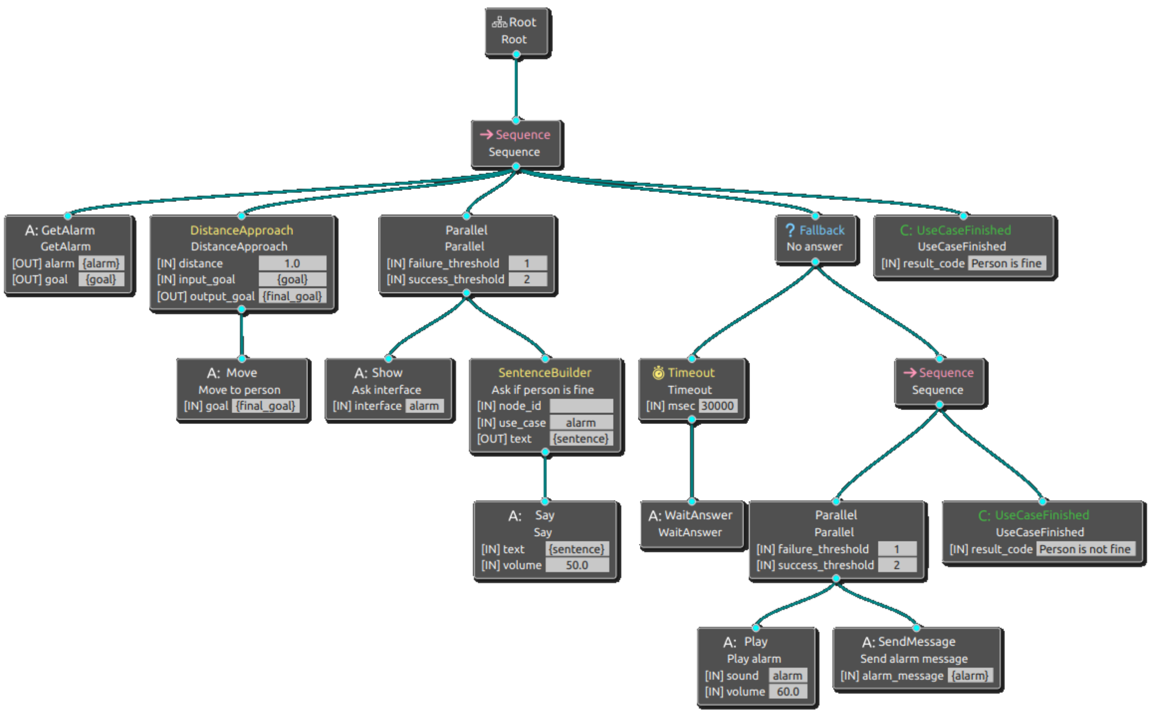


Figure 14. The behaviour tree encoding attention to a fall alarm.

A relevant question that may arise at this point is how to deal with unexpected events. In this sense, it is important to distinguish two aspects: situations that are perceptually complex, as they incorporate elements that are not usually contemplated, and situations that alter the plan described in a behaviour tree or that simply cannot be resolved by any of them. The first situation needs to be addressed with more in-depth training. Figure 15, for example, shows a person correctly detected as having fallen on the floor despite the chair hiding his head. This is not considered a normal situation, but training allows it to be correctly resolved. It would not be a problem. With respect to truly unexpected events, it is important to note that it does not seem reasonable to let the robot make an action decision without having considered the complete context information or without having an action protocol (i.e., a behaviour tree) designed by an expert and intensively validated in a controlled environment. If a situation is detected that is not contemplated, it is best for the system to alert the caregiver, provide the details (images, sensor data) available to it, and leave the caregiver in charge of what to do. The robot can behave in this situation as a fully teleoperated element, which can see and hear, as well as say, whatever the caregiver decides.

To validate the robustness of the system, a total of 24 tests were carried out with four different users. These tests evaluate the success of the execution of the mission. The detections are considered successful if, during the time the user is in the room, the track ID is maintained and the different postures of the user are correctly identified. On the other hand, two different missions were evaluated, one for a fall case and one for a wandering case. In the case of a fall, the mission is considered successful if the alarm is generated by the system and the robot goes to the location of the fall. If there is no fall, the mission is considered successful if the system correctly locates the person in the room (e.g., locates the person on the bed when he/she lies down on it). In the 24 tests performed, each of the four people simulated three falls and three wanderings around the room. The success rate of the missions was 95.45%.

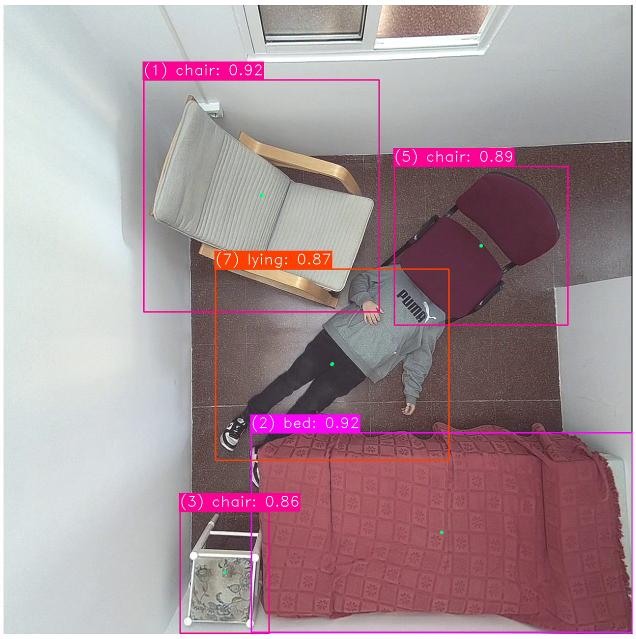


Figure 15. A person correctly detected as having fallen on the ground (see text).

The use of CORTEX makes it relatively easy to extend this use case. Thus, when the SmartHome agent detects that the person is lying in bed and records this in the DSR of the IoT system (person in bed), the FMCW sensor under the bed wakes up and starts to periodically measure the heart and respiratory rates of the person. The MQTT agent records these parameters in the DSR, and they are stored in the database associated with the person. The latest available data can be sent to the contact person in the e-mail sent to him/her when managing the risk situation described above. Another emergency situation, which is managed by the robot like the scenario described for a possible fall, is that associated with a person entering the bathroom and remaining in this room for an excessive amount of time. In this case, the presence is detected using a simple presence detector, which is connected to the DSR by the MQTT agent. The robot will approach the bathroom door and ask the person, as before, if he or she is well. The robot's handling of the situation is the same as described above, with the same top priority being maintained over any other ongoing tasks.

4.4. Comparison with Other IoT Systems

This section provides a qualitative comparison of our proposal with other popular IoT platforms: Home Assistant and Node-RED. Home Assistant (https://www.home-assistant.io/, accessed on 15 November 2024) is a complete open-source operating system, which allows us to integrate home automation devices of many different brands into the home, giving us full control of the automation of the home. Once all the devices have been added to the system (integrations), different automation processes can be carried out so that they perform certain actions that can be configured in detail. Node-RED (https://nodered.org/, accessed on 15 November 2024) is a flow-based open-source development tool designed to program Internet of Things (IoT) applications with ease.

Table 2 summarises the compliance or non-compliance with an important set of characteristics that might be required of an IoT platform. As a major disadvantage, our system does not currently have a graphical interface, nor does it allow YAML or GUI configuration or automation using drag-and-drop. As far as the rest of the features are concerned, they are all satisfactorily fulfilled. In addition, one of the main advantages of our system in supporting dependent people is that we do not need any kind of invasive sensor that the person has to have on their body during the day. Therefore, the acceptance and ease of integration are of high value in this proposed system.

Table 2. A qualitative comparison of the proposed system vs. Home Assistant and Node-RED.

Feature	Home Assistant	Node-RED	Our Proposal
Graphical interface	YES	NO	NO
Support for multiple devices	YES	YES	YES
Automation by visual flows (drag-and-drop)	NO	YES	NO
YAML or GUI configuration	YES	NO	NO
Connectivity with external services (APIs)	YES	YES	YES
Home automation-oriented	YES	NO	YES
Developer-oriented	NO	YES	YES
Scalability for complex integration	YES	YES	YES
Advanced automation between services and protocols	NO	YES	YES
Support for various communication protocols (Zigbee, Z-Wave, MQTT)	YES	NO	YES
Local server installation	YES	YES	YES
Open-source code	YES	YES	YES
Communication between systems hosted on different local servers (robot–environment)	NO	NO	YES

5. Discussion

The deployment of new assistive technologies, including Socially Assistive Robots (SARs), will become an increasingly common reality to help older people lead better lives in their own homes. However, it is important to note that there are still many questions about adherence, acceptability, and real long-term usefulness. In most cases, the design process has been carried out with very limited consideration of the needs, preferences, or values of potential users. Another limitation is related to the limited number of long-term evaluations carried out in real-life settings. Although the present work also suffers from this last problem, the design of use cases involving people follows the scheme described, for a specific case, in previously published work by our research group [24,28]. In this manuscript, the emphasis is on the integration of two different technological approaches that can perform specific tasks independently but that, when deployed in the same environment, can solve new tasks that require coordination. Tests were carried out in a Living Lab, the design of which reflects the characteristics of a real environment in terms of space distribution and furniture. The users who took part do indeed fall outside the required profile, but we have not considered use cases in which the person plays an active role.

Protecting private data in IoT applications, such as a smart home, remains a relevant challenge due to the distributed nature of the deployed networks [29]. The use of these home automation systems to deploy smart healthcare solutions, in line with what has been addressed in this paper, exacerbates security issues in terms of the privacy of the owner of the data handled and the very confidentiality of these data, which can be particularly sensitive. There are different options to address this problem. In our implementation, data move in the network without being linked to a specific person. The user ID is also not moved in the network. The captured images are processed by the system, and the semantic data are stored in the representation without being stored or sent externally.

6. Conclusions and Future Work

This article describes the integration of a robot and an IoT system to form an AAL environment, where both systems continue to operate independently and autonomously and can maintain their own tasks. The integration takes place at the level of knowledge exchange. This transfer allows both systems to have increased knowledge, where the relevant part of the other is incorporated.

As an example of a use case to initially validate this proposal, we use the detection of a possible fall of a person living alone at home; upon the detection of a possible fall, the robot comes to ask and assess whether he/she needs external help. In this use case, the robot acts as an interface for the interaction between the AAL environment and the person, based on the resources it is equipped with, while the IoT system provides continuous information on the person's position, being able to distinguish whether the person is sitting, standing, or lying down. As discussed in this article, the IoT system offers other capabilities, such as the ability to monitor the person's heart or respiratory rate when lying in bed or sitting on the sofa in the dining room. These data will provide the caregiver with relevant information to determine the patient's condition and the relevance of the possible fall.

Although it could be argued that it is a better solution to maintain a single representation deployed in the IoT system, with which the agents instantiated in the robot and in the IoT system itself communicate, this proposal has the advantage that the two systems that make up the CPSoS remain functionally independent so that verification remains modular and, therefore, easier. The use cases are divided into clearly differentiated parts (the environment detects a possible fall, and the robot manages its evaluation and possible request for help), whose individual traceability is easier. The problem arises when deciding which part of the representation to share with the other.

Future work will consist of incorporating a mechanism to deal with possible inconsistencies or conflicts that arise between the knowledge that either of the two CSs has captured from the context and the knowledge they receive from the other entity. Perceptions may, in many cases, be characterised by a probability value (this is the case, for example, for object or person position detections). Information is also gained from the active interaction with the context itself. If the system reports that there is a person who has fallen on the ground and, when the robot approaches that position, it detects that this is not the case and that the person is actually sitting, the final decision could rest on probabilities, but what is more interesting is that the robot simply asks the person what he/she is doing and whether he/she is OK. In other cases, given that the DSR graph allows it, a double symbolic relationship between two nodes could be maintained. If this double relation involves a contradiction, the system could ask the robot to execute a certain task specifically designed to assess the real situation of the context. Additionally, the factor that contributes most to reducing the percentage of correctly resolved missions is the percentage of correct detections of the person's posture or the correct tracking of the person (who should always have the same identifier). This success in detecting and tracking the person is only 83.33%. This success rate must be improved by running a second training stage with a larger dataset. In this augmented dataset, data acquisition should focus on those contexts where the system has shown poorer detections, such as people with geometric patterns on their sweaters. Finally, it is also important to design the control system of the SoS itself, which will allow it to achieve its own objectives, distinct from those of the CSs that make it up. In the example shown in this article, the objectives are those of the CSs themselves: the IoT system is able to detect a possible fall, and the robot is able to determine whether the person has actually fallen and whether they need external help. It is also our aim to incorporate the person into the CPSoS as a component. There is previous work in this direction, the so-called human-in-the-loop CPS (HiLCPS) [30,31] or social cyber-physical system (SCPS) [32], which we will have to study in depth before undertaking this step.

Author Contributions: Conceptualization, J.G., A.T., J.P.B. and A.B.; methodology, A.T. and A.B.; software, J.G., Ó.P. and A.T.; validation, J.G., Ó.P. and A.T.; formal analysis, J.G. and A.T.; investigation, J.G., Ó.P. and A.T.; resources, A.B.; data curation, J.G., Ó.P. and A.T.; writing—original draft preparation, J.G., A.T., Ó.P. and A.B.; writing—review and editing, J.G., A.T., Ó.P. and A.B.; supervision, J.P.B. and A.B.; project administration, J.P.B. and A.B.; funding acquisition, J.P.B. and A.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by grants PDC2022-133597-C4X, TED2021-131739B-C2X, and PID2022-137344OB-C3X, funded by MCIN/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR (for the first two grants), and "ERDF A way of making Europe" (for the third grant).

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request. We consider that the database currently used for the CNN training is not of adequate depth to make it publicly available in a dedicated repository.

Acknowledgments: The authors would like to thank the people who were involved in the evaluation of the system at Vitalia Teatinos retirement home.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Cruces, A.; Jerez, A.; Bandera, J.P.; Bandera, A. Socially Assistive Robots in Smart Environments to Attend Elderly People—A Survey. *Appl. Sci.* **2024**, *14*, 5287. [CrossRef]
2. WHO. Active Ageing: A Policy Framework. *Aging Male* **2002**, *5*, 1–37. [CrossRef] [PubMed]
3. Calderita, L.V.; Vega, A.; Barroso-Ramírez, S.; Bustos, P.; Núñez, P. Designing a Cyber-Physical System for Ambient Assisted Living: A Use-Case Analysis for Social Robot Navigation in Caregiving Centers. *Sensors* **2020**, *20*, 4005. [CrossRef] [PubMed]
4. Sanz, R.; Bermejo, J.; Rodriguez, M.; Aguado, E. The role of knowledge in cyber-physical systems of systems. *TASK Q.* **2021**, *25*, 355–373. [CrossRef]
5. Bustos, P.; Manso, L.; Bandera, A.; Bandera, J.; García-Varea, I.; Martínez-Gómez, J. The CORTEX cognitive robotics architecture: Use cases. *Cogn. Syst. Res.* **2019**, *55*, 107–123. [CrossRef]
6. Marfil, R.; Romero-Garces, A.; Bandera, J.; Manso, L.; Calderita, L.; Bustos, P.; Bandera, A.; Garcia-Polo, J.; Fernandez, F.; Voilmy, D. Perceptions or Actions? Grounding How Agents Interact Within a Software Architecture for Cognitive Robotics. *Cogn. Comput.* **2020**, *12*, 479–497. [CrossRef]
7. Nielsen, C.B.; Larsen, P.G.; Fitzgerald, J.; Woodcock, J.; Peleska, J. Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions. *ACM Comput. Surv.* **2015**, *48*, 1–41. [CrossRef]
8. Kopetz, H.; Bondavalli, A.; Brancati, F.; Frömel, B.; Höftberger, O.; Iacob, S. Emergence in Cyber-Physical Systems-of-Systems (CPSoSs). In *Cyber-Physical Systems of Systems: Foundations—A Conceptual Model and Some Derivations: The AMADEOS Legacy*; Bondavalli, A., Bouchenak, S., Kopetz, H., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 73–96. [CrossRef]
9. Venancio-Teixeira, J.; da Silva Hounsell, M.; Wildgrube-Bertol, D. How CPS and Autonomous Robots are Integrated to other I4.0 Technologies: A systematic literature review. *Prod. Manuf. Res.* **2023**, *11*, 2279715. [CrossRef]
10. Bocicor, M.I.; Frau, D.C.; Draghici, I.C.; Goga, N.; Molnar, A.J.; Pérez, R.V.; Vasilateanu, A. Cyber-physical system for assisted living and home monitoring. In Proceedings of the 2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 7–9 September 2017; pp. 487–493. [CrossRef]
11. De Venuto, D.; Annese, V.F.; Sangiovanni-Vincentelli, A.L. The ultimate IoT application: A cyber-physical system for ambient assisted living. In Proceedings of the 2016 IEEE International Symposium on Circuits and Systems (ISCAS), Montreal, QC, Canada, 22–25 May 2016; pp. 2042–2045. [CrossRef]
12. Coradeschi, S.; Cesta, A.; Cortellessa, G.; Coraci, L.; Galindo, C.; Gonzalez, J.; Karlsson, L.; Forsberg, A.; Frennert, S.; Furfari, F.; et al. GiraffPlus: A System for Monitoring Activities and Physiological Parameters and Promoting Social Interaction for Elderly. In *Human-Computer Systems Interaction: Backgrounds and Applications 3*; Hippe, Z.S., Kulikowski, J.L., Mroczek, T., Wtorek, J., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 261–271. [CrossRef]
13. Wengefeld, T.; Schuetz, B.; Girdziunaite, G.; Scheidig, A.; Gross, H.M. The MORPHIA Project: First Results of a Long-Term User Study in an Elderly Care Scenario from Robotic Point of View. In Proceedings of the ISR Europe 2022, 54th International Symposium on Robotics, Munich, Germany, 20–21 June 2022; pp. 1–8.
14. Do, H.M.; Pham, M.; Sheng, W.; Yang, D.; Liu, M. RiSH: A robot-integrated smart home for elderly care. *Robot. Auton. Syst.* **2018**, *101*, 74–92. [CrossRef]
15. Mojarad, R.; Chibani, A.; Attal, F.; Khodabandelou, G.; Amirat, Y. A hybrid and context-aware framework for normal and abnormal human behavior recognition. *Soft Comput.* **2023**, *28*, 4821–4845. [CrossRef]
16. Lin, H.Z.; Chen, H.H.; Choophutthakan, K.; Li, C.H.G. Autonomous Mobile Robot as a Cyber-Physical System Featuring Networked Deep Learning and Control. In Proceedings of the 2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Sapporo, Japan, 11–15 July 2022; pp. 268–274. [CrossRef]
17. Mazumder, A.; Sahed, M.; Tasneem, Z.; Das, P.; Badal, F.; Ali, M.; Ahamed, M.; Abhi, S.; Sarker, S.; Das, S.; et al. Towards next generation digital twin in robotics: Trends, scopes, challenges, and future. *Heliyon* **2023**, *9*, e13359. [CrossRef] [PubMed]
18. Mitchell, D.; Blanche, J.; Zaki, O.; Roe, J.; Kong, L.; Harper, S.; Robu, V.; Lim, T.; Flynn, D. Symbiotic System of Systems Design for Safe and Resilient Autonomous Robotics in Offshore Wind Farms. *IEEE Access* **2021**, *9*, 141421–141452. [CrossRef]
19. Ben Halima, R.; Hachicha, M.; Jemal, A.; Hadj Kacem, A. MAPE-K patterns for self-adaptation in cyber-physical systems. *J. Supercomput.* **2022**, *79*, 4917–4943. [CrossRef]

20. Romero-Garcés, A.; Hidalgo-Paniagua, A.; González-García, M.; Bandera, A. On Managing Knowledge for MAPE-K Loops in Self-Adaptive Robotics Using a Graph-Based Runtime Model. *Appl. Sci.* **2022**, *12*, 8583. [CrossRef]
21. Gaber, M.M. *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*, 1st ed.; Springer Publishing Company, Incorporated: New York, NY, USA, 2009.
22. Zaraté, P.; Liu, S. A new trend for knowledge-based decision support systems design. *Int. J. Inf. Decis. Sci.* **2016**, *8*, 305–324. [CrossRef]
23. 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]
24. Jerez, A.; Iglesias, A.; Pérez-Lorenzo, J.; Tudela, A.; Cruces, A.; Bandera, J. An User-Centered Evaluation of Two Socially Assistive Robots Integrated in a Retirement Home. *Int. J. Soc. Robot.* **2024**, 1–21. [CrossRef]
25. Yang, L.; Kang, B.; Huang, Z.; Xu, X.; Feng, J.; Zhao, H. Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data. In Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–22 June 2024; pp. 10371–10381. [CrossRef]
26. Macenski, S.; Soragna, A.; Carroll, M.; Ge, Z. Impact of ROS 2 Node Composition in Robotic Systems. *arXiv* **2023**, arXiv:abs/2305.09933. [CrossRef]
27. Macenski, S.; Martín, F.; White, R.; Clavero, J.G. The Marathon 2: A Navigation System. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 2718–2725. [CrossRef]
28. Iglesias, A.; Viciana, R.; Pérez-Lorenzo, J.M.; Ting, K.L.H.; Tudela, A.; Marfil, R.; Qbilat, M.; Hurtado, A.; Jerez, A.; Bandera, J.P. The Town Crier: A Use-Case Design and Implementation for a Socially Assistive Robot in Retirement Homes. *Robotics* **2024**, *13*, 61. [CrossRef]
29. Popoola, O.; Rodrigues, M.; Marchang, J.; Shenfield, A.; Ikpehai, A.; Popoola, J. A critical literature review of security and privacy in smart home healthcare schemes adopting IoT & blockchain: Problems, challenges and solutions. *Blockchain Res. Appl.* **2024**, *5*, 100178. [CrossRef]
30. Gil, M.; Albert, M.; Fons, J.; Pelechano, V. Engineering human-in-the-loop interactions in cyber-physical systems. *Inf. Softw. Technol.* **2020**, *126*, 106349. [CrossRef]
31. Tehrani, B.M.; Wang, J.; Wang, C., Review of Human-in-the-Loop Cyber-Physical Systems (HiLCPS): The Current Status from Human Perspective. In *Computing in Civil Engineering 2019*; American Society of Civil Engineers: Reston, VA, USA, 2019; pp. 470–478. [CrossRef]
32. Calinescu, R.; Cámara, J.; Paterson, C. Socio-Cyber-Physical Systems: Models, Opportunities, Open Challenges. In Proceedings of the 2019 IEEE/ACM 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS), Montreal, QC, Canada, 28 May 2019; pp. 2–6. [CrossRef]