Integrating Simulated Exogenous Environments to Support the Learning Process of the Embedded MAS Approach

Bruno Policarpo Toledo Freitas[12], **Nilson Mori Lazarin**[12], **Carlos Eduardo Pantoja**[1], **José Viterbo**[2]

[1]Federal Center for Technological Education Celso Suckow da Fonseca (CEFET/RJ)

[2]Fluminense Federal University (UFF)

{bruno.freitas,nilson.lazarin,carlos.pantoja}cefet-rj.br, viterbo@ic.uff.br

Abstract. *A multiagent system (MAS) is a group of autonomous software entities capable of collaboration and cooperation. It is a common practice to use simulators in MAS development. However, in the case of Embedded MAS education, integrating simulators is difficult because the agent's reasoning is tightly coupled to hardware communication interfaces. This work then presents a simulator integration methodology based on serial channel emulation that fully decouples the agent's reasoning from the environment, allowing flexible integration of simulators. We present a report on an Embedded AI undergraduate course in two academic semesters, showing that the student's projects completion rate increased from 33% to 100%.*

1. Introduction

An agent is an autonomous software entity capable of reasoning that, based on the system's perceived state, executes plans to achieve a determined goal. Agents differ from conventional software because they show independence, cognition, and collaboration skills, while being proactive and adaptable. Consequently, a Multi-agent system (MAS) reunites a group of agents that could collaborate or compete for common or conflicting goals (Michel et al. 2009). Agents can use cognitive models to guide their reasoning process. One of the most adopted models is the Belief-Desire-Intention (BDI), which considers the mental state based on human reasoning composed of beliefs, desires, intentions, plans, and actions (Bratman et al. 1988). Furthermore, nowadays, the use of BDI agents embedded in hardware platforms has become a reality (Onyedinma et al. 2020; Gavigan and Esfandiari 2021a; Karaduman et al. 2023; Pantoja et al. 2023; Lazarin et al. 2024).

From an educational point of view, the BDI agent approach allows the use of abstractions closer to the human mind to construct computational entities, leading to a natural learning process of building intelligent applications (Lazarin et al. 2023). Additionally, adopting educational robotics and visual development environments can support the learning process in Artificial Intelligence (AI) fields, such as embedded systems with MAS (Yim and Su 2024).

For example, a visual IDE for Agents and the Web of Things (Burattini et al. 2022) focuses on the industrial scenario, particularly interested in enabling domain experts without programming experience to create or modify systems in Web of Things scenarios. It has also been proposed a specific toolkit for teaching distributed and embedded AI (Lazarin et al. 2023), providing a do-it-yourself kit containing methods, software, teaching materials, hardware schematics, and codes for experiments with embedded BDI agents. This toolkit intends to support the professor and the student in the device design by following a methodology in which the student must focus on the reasoning layer by abstracting low-level layers. For this, the professor previously assembles hardware platforms and components, such as sensors and actuators, into a single device to speed up the learning process.

However, although hardware such as robotics and physical artifacts are generally effective in learning, they can be too expensive to build, which does not scale well for larger classrooms (Yim and Su 2024). In addition, it requires specific electronic abilities and assembly of components to prepare them to communicate with agents (Lazarin et al. 2023). The professor could then be limited by the insufficient number of devices and the broad knowledge necessary to assemble them.

This work proposes an architecture that allows flexible exogenous environments to support the development and learning processes of embedded MAS. In this generic approach, the physical environment can be switched by any simulator without changing the reasoning code or adding new layers between the agent and the environment. This guarantees that the reasoning and architecture of the agents remain intact during deployment. Because agents use the serial port to gather sensor data and send commands to actuators, we present a novel serial channel emulator dedicated to embedded MAS using Jason (Bordini and Hübner 2006; Pantoja et al. 2023) or JaCaMo (Boissier et al. 2013) to integrate the agent's reasoning with external software.

We conducted several proof-of-concept experiments. In the first experiment, the MAS manipulates SimulIDE (González 2023), an Arduino simulator, to interact with the simulated hardware. Subsequently, we built an educational application to illustrate agents exchanging perceptions and commands with the emulated ports. Finally, we created a real robot using an embedded MAS and switched the hardware of the Webots robot simulator (Michel 1998). We also present a report on a real classroom where students had to develop embedded MAS as part of the requirements for approval. The contributions of our architecture are as follows: i) A novel serial channel emulator dedicated to agents that interface hardware at the kernel level of the operating system; ii) The extension of an existing toolkit for engineering distributed and embedded AI systems; iii) The integration of two well-known simulators for suppressing the absence of hardware technologies.

The remainder of this paper is organized as follows. Section 2 reviews the concepts necessary to understand this proposal. Section 3 presents a serial interface emulation proposal for an Embedded MAS. Section 4 presents a proof of the concept of this proposal. Section 5 shows the results when applying the methodology in classroom. Section 6 presents and discusses related work and tools. Finally, Section 7 presents a discussion and future work.

2. Background

The agent perception and action models are usually tied to an *endogenous* or *exogenous* environment. When the MAS adopts an environment dimension, which is optional in some programming languages, the agent can perceive and act upon this environment, modifying it to fulfill its goals. When using an endogenous approach, it is necessary to project and program the environment logical abstraction where agents will gather the finite set of possible perceptions, actions, and consequent reflections in this environment. On the other hand, such abstraction is not necessarily mandatory when using an exogenous approach since the source of perceptions and target for actions can stand external to the agent. Besides, the exogenous environment is continuous, which means the set of perceptions and actions tends to infinity (Ricci et al. 2012). This is the case for embedded MAS: the environment is continuous since it is the real world, and the agents can interact with it by gathering perceptions from the sensors, and the actions are commands sent to actuators. Then, the available hardware limits the sets of perceptions and actions.

One can use a four-layer architecture shown in Figure 1 to build an embedded MAS (Pantoja et al. 2016): the Reasoning layer is where the MAS is hosted; the Interfacing layer is where the agents' commands and the environments' perception flow between the MAS and the devices' microcontrollers; the Firmware layer interprets the raw data coming from sensors as useful perceptions for agents and vice-versa, and finally; the Hardware layer defines the boundary between the device and the real-world since it is where sensors and actuators exist.

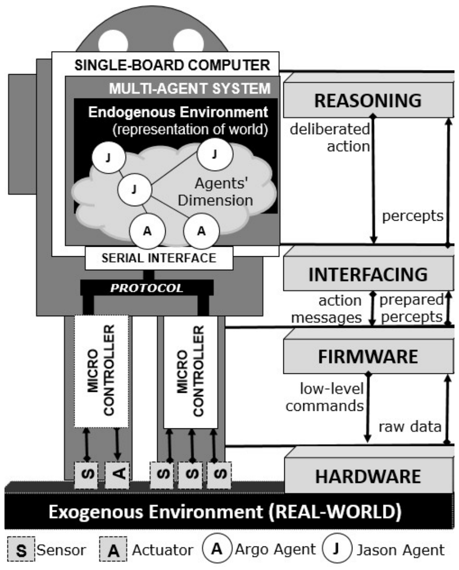


Figure 1. The architecture of an embedded MAS.

In the Reasoning layer, the ARGO architecture (Pantoja et al. 2016) extends standard Jason agents (Bordini and Hübner 2006) to enable direct interaction in the exogenous environment by accessing sensors and actuators. This approach is centered on serial communication, where microcontrollers connected to the serial ports can send perceptions directly to agents' minds. In the same way, agents can send commands using these serial ports. This is possible because a double library middleware abstracts this communication for both layers and the device's designers (Lazarin and Pantoja 2015).

As can be seen, serial channel communication plays an important role in this architecture since sensors and actuators are physically connected to the MAS by using serial ports. An ARGO agent can then use this channel to communicate with the microcontroller, sending commands to act upon the real world using Jason's specific internal action. Also, the microcontrollers' perceptions are updated on the agent's belief based on each reasoning cycle.

The designer could take advantage of a simulation approach based on an indistinguishable serial communication, which could lead to physical components or simulators. In this case, the agent coding will not need to change since it is indifferent to the agent if it is accessing hardware or not. The agent's behavior manages to stay the same.

3. The Exogenous Environment Simulation Approach

This work presents an architecture to integrate an exogenous environment simulation that supports the learning process by avoiding the need for devices' construction during the MAS development and teaching. The proposed approach uses pseudo serial interfaces to replace the firmware and hardware layers with a simulation layer, where the hardware components can be represented. The main advantage of the proposed approach is that the agents gather perceptions and act naturally, as the MAS code does not require changes to function properly in both situations.

In the teaching and learning process of the embedded MAS approach, the professor can replace the devices with simulated representations. These digital twins are useful to avoid damage to hardware components, microcontrollers, and boards during early contact with the technologies. Besides, it reduces costs in constructing new devices or emergency needs for new ones. For example, in a classroom where the professors do not have enough resources, they need only one physical prototype for the whole class. At the same time, the students can use the simulator to represent the physical prototype.

Concerning the device's architecture of a MAS, agents at the reasoning layer can interface hardware using a protocol to exchange serial messages in the interfacing layer using serial interfaces. This approach extends the interfacing layer to communicate with pseudo-serial interfaces implemented as kernel modules. ARGO agents connect to the pseudo-serial port in the same way they connect to physical serial ports. Once connected, agents can receive prepared percepts from the simulators or any virtualized application. Besides, it can send action messages (commands) to be executed in simulators or other applications. The architecture is shown in Figure 2.

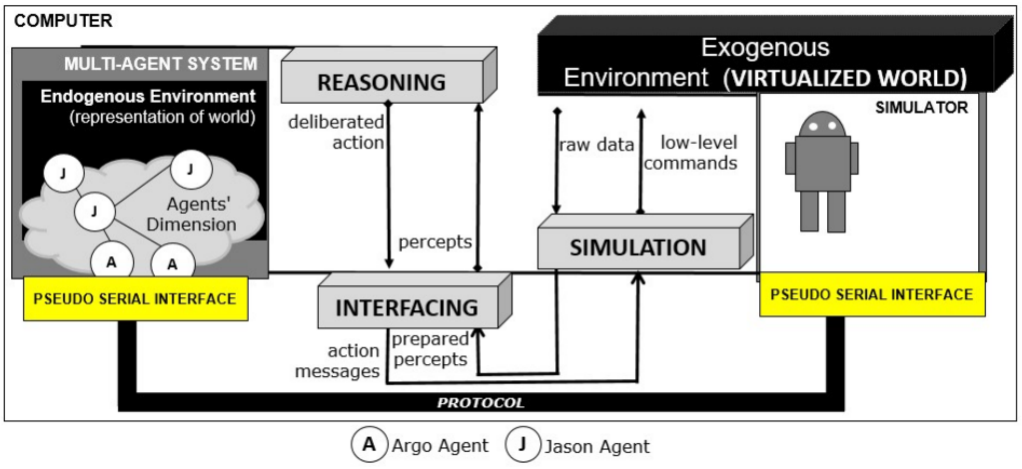
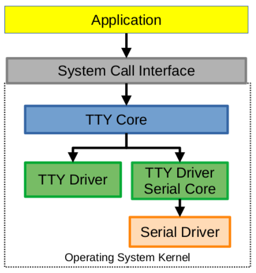


Figure 2. The architecture, considering the exogenous environment simulation.

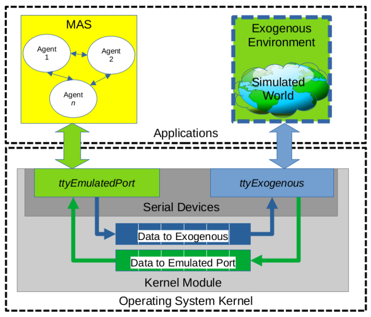
Another benefit of pseudo-serial interfaces is that they are generic for any simulator or application developed in any programming language. A simulation developed in a simulator or any programming language must provide raw data simulating sensors and prepare them as perceptions in a belief-like style since the embedded MAS used the BDI cognitive model. The opposite direction works the same; commands coming from agents must be translated to low-level commands to activate simulated actuators.

3.1. Implementation

The proposed methodology is implemented using a Linux kernel module. Inside the kernel of Linux operating systems, the TeleTYpewriter (TTY) layer manages all serial device class (Linux Kernel Organization, Inc. 2023). Figure 3a shows this subordination. Because any type of serial device is also a TTY device, one can build a TTY device that does something different, which will still be seen as a serial device for user processes.



(a) Serial devices architecture, adapted of (Linux Kernel Organization, Inc. 2023).



(b) Emulated serial interface, adapted of (Freitas et al. 2023).

Figure 3. Implementation of serial device emulation

The kernel module instantiates two pseudo serial devices: the ttyEmulatedPort and ttyExogenous. The MAS is connected to the ttyEmulatedPort device, and the simulated environment is connected to the ttyExogenous device. Whenever the agent wants to send data, it writes it on the ttyEmulatedPort, which the simulated environment can retrieve by reading the ttyExogenous port. The process is the same when the exogenous environment sends data: the agent can read its content from the ttyEmulatedPort. Figure 3b shows the main idea behind the module.

By using a serial port emulator as a kernel module, we achieve transparency between the ARGO agent and the simulator: it would still use a serial port, making the process of migration to a physical prototype just changing from the emulated port to the real one.

4. Proof of concept

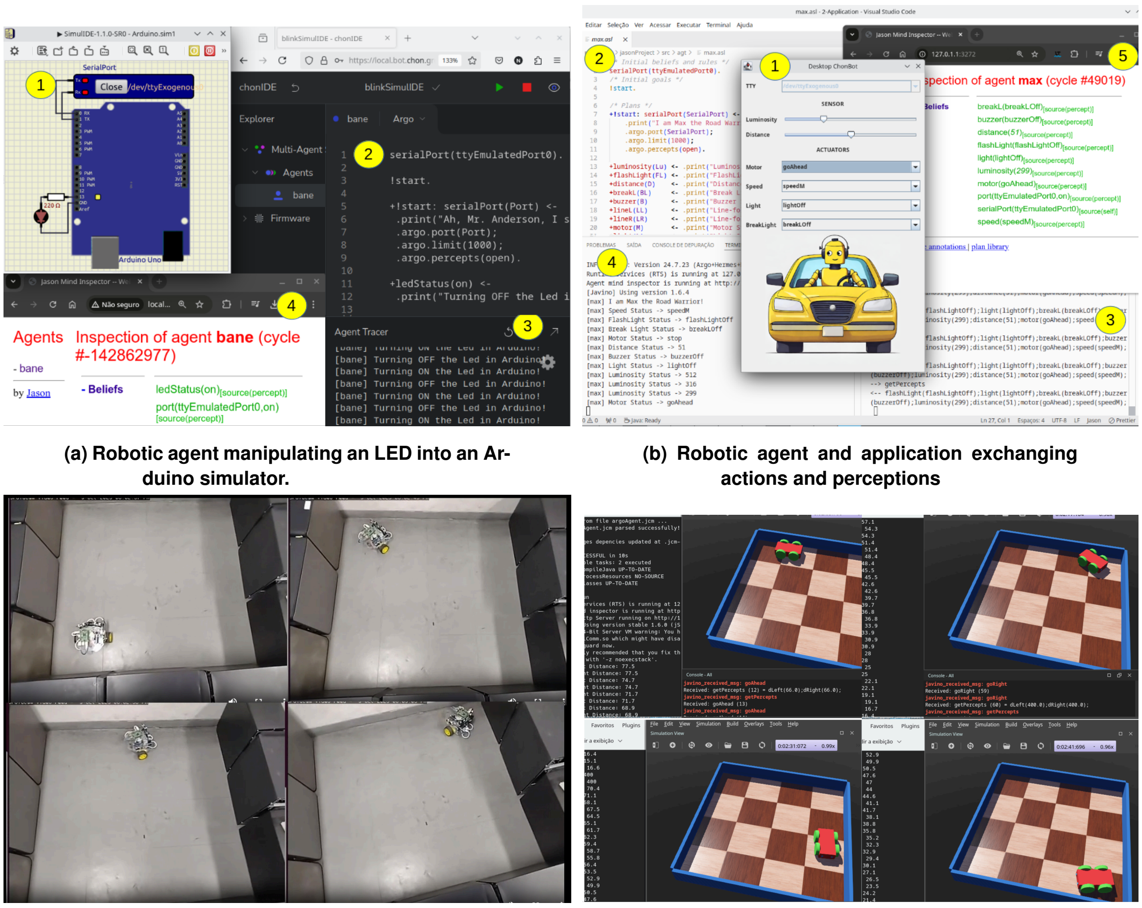
This section will present four experiments as proof of concept using simulators as an exogenous environment for embedded MAS. First, we used an Arduino simulator. In the second, we build a desktop application. The third uses a robot simulator. Finally, we build a physical prototype of the robot simulation. The first case illustrates the use of a generic hardware simulator. The desktop application shows how one can build new scenarios for MAS education. Finally, the last show how a robotic MAS developed in an exogenous simulated environment can easily be deployed to real-world scenarios.

In the first experiment, as shown in Figure 4a, a project was created in SimulIDE (version 1.0.0 R1448) where an Arduino Uno board, a 220Ω resistor, a Light Emitting Diode (LED), and a Serial Port component were added (1). We used the default Javino's blink example code to deploy the firmware layer into the simulator. After that, we used the default ChonIDE's blink project (Souza De Jesus et al. 2023), where a BDI agent constantly manipulates an LED. When the agent perceives the LED is on, he acts to turn it off. When he perceives the LED is off, he acts to turn the LED on. The only modification was to change the serial port from *ttyACM0* to *ttyEmulatedPort0* (2). When running the example, the MAS log (3) shows that the agent manipulates the LED correctly. Furthermore, the perceptions in the agent's belief base (4) are identical to those received if we were using a physical Arduino board. This example shows the potential of the proposed architecture in simulating hardware devices and integrating them with a MAS. The same code used in real Arduinos was uploaded to the simulated Arduino in SimulIDE. So, the set of possible commands and perceptions were the same in both. As Javino works based on a serial protocol for exchanging serial messages, it delivers the messages to the specified port regardless of its nature. Then, it is up to the agent to decide whether or not to connect in a physical or pseudo serial port. As a matter of fact, the agent's reasoning and code do not need to change, except for the target serial port.

In the second experiment, presented in Figure 4b, we developed a desktop application in Java that simulates a physical prototype (1). We use the default chonBot's reasoning layer example for Jason. A BDI agent named Max displays the perceptions received from the simulated exogenous environment based on what a real vehicle would do: it shows the motor and buzzer status, the speed, and the LED's status from the break light, alert, and flashlight. Besides, it simulates the road luminosity and the distance to an obstacle. In this desktop application, all these properties are manipulable at runtime. The agent will reflect every change on the console. The only modification was to change the serial port from ttyACM0 to ttyEmulatedPort0 (2). When running the example, the desktop application's log (3) displays the messages exchanged with the reasoning layer. The MAS log shows that the agent interacts correctly with the application (4). Finally, the perceptions in the agent's belief base (5) are identical to those received if we were using a physical prototype. This example shows that desktop applications can be used as simulators. In educational scenarios, students can produce their own simulated environment as an exercise to represent the real world virtually, or the professor can build a scenario for them instead.

In the last two experiments, presented in Figures 4d and 4c, we assembled a prototype based on the chonBot model using one Raspberry Pi, one Arduino, two sensors (distance), and a two-Wheel Drive (WD) robotic platform. Next, we adapted a Webot simulation example that contains two main components: one that describes the environment (world), with objects' position, sky, gravity, object masses, and so on, and another that specifies the robot's behavior (controller). Webot is a multipurpose robotic simulator widely adopted since it is free and open-source. In this chosen example, the world is a small square where a 4WD prototype will avoid collision with the walls, and the robot's controller is written in the programming language C. This experiment aims to show that simulations can replace complex robots by just swapping the connection port.

In this case, the MAS is interfacing a C-based layer in the simulated firmware (We-


(a) Robotic agent manipulating an LED into an Arduino simulator.

(b) Robotic agent and application exchanging actions and perceptions

(c) Robotic agent acting in the Exogenous real world.

(d) Robotic agent acting in the Exogenous simulated world.

Figure 4. Proof of concept experiments.

bots). Javino is a double-side middleware (reasoning and firmware) that offers Arduino, Java, and Python interfacing. So, to allow the ARGO agents to interact with the simulation, we first needed to implement a novel C-based middleware for Javino's firmware side. Hence, we add a main loop in the controller file to listen to the pseudo serial port, waiting for agents' acts or request perceptions. If an action is received, it must be mapped into original robot commands. When a perception request is received, the simulator answers with the sensor's values. Extending Javino is interesting because it expands the possibilities for firmware adoption since this Javino C-based can run in other technologies in addition to Webots.

At last, we program the reasoning layer using JaCaMo, where an ARGO agent should drive the vehicle, avoiding the wall by turning right when it finds something between a pre-defined range. Firstly, we connected the MAS to connect to Webots. In this case, the agent successfully drove the robot without collision when executing the MAS. Once the simulation validated the agent's behavior, we assembled an arena with proportional dimensions as the simulated world. Afterward, we deployed the reasoning layer to the physical prototype. The only code change performed was changing the port, where ttyEmulatedPort0 (emulated) was changed to ttyUSB0 (physical). The ARGO agent presented the same behavior when executing in the embedded MAS hosted by the physical prototype.

4.1. Reproducibility

Aiming to ensure the reproducibility of this work, the source code, the scenario implementation, the source code, and a video demonstration are available in a web page[1]. It also contains links to instructions on how to install and use the emulator created.

5. Results in an Embedded AI Course

This section presents a report on two academic semesters in an undergraduate electric engineering course on working with embedded MAS. The first report will bring the results before using the proposed methodology, while the second will report the results using it.

The course mixes theory and practice in teaching distributed and embedded AI supported by intelligent BDI-based agents. It aims to transform theoretical concepts into easily assimilated examples, stimulate learning and commitment, and improve classroom interaction at various levels of education. We used a project-based learning model, where small groups chose a problem to solve using robotic agents built with microcontrollers, sensors, and actuators available in the laboratory.

The classroom followed a six-step methodology that goes from the basics of agent theory to the development of distributed embedded AI systems (Lazarin et al. 2023): i) Introductory concepts about intelligent agents; ii) BDI architecture and agent programming language; iii) BDI reasoning cycle and interaction between agents; iv) Communication between different MAS; v) Mobile agents and open MAS; vi) Agents embedded in robotic prototypes.

At the end of the course, a general presentation was organized, where all groups described their experience, challenges encountered, and results obtained. Also, they presented a report describing their experience, considering the four different stages in the embedded MAS development process: **R**easoning (logical part of the agents), **H**ardware (construction of the prototype), **F**irmware (logical part of the hardware), and **I**nterfacing (integration of the logical part with the hardware).

Table 1 describes the proposed projects the first time the course was held, in the second semester of 2023. The table's columns relate to the initial letters of each stage mentioned in the last paragraph. As can be seen, only 2 out of 6 groups were able to fully finish their projects. Other 2 groups didn't even make it to the hardware development phase. In this iteration, the most common challenge reported was difficulties with building hardware with Arduino, seconded by difficulties with C programming. It is also noteworthy to mention that, while some groups managed to complete the Reasoning layer, they weren't capable of testing it properly without the hardware completed.

Table 2 shows the first iteration of the class using the integrated Arduino shown in Figure 4a, in the second semester of 2024. Here, most groups opted to use the simulated environment, the main reason being the inexperience with Arduino itself. By using the integrated simulator, all these groups managed to complete their projects within the course's duration, showing functional prototypes.

[1]`https://papers.chon.group/CSBC/2025/serialChannelEmulator/`

Table 1. Description of the practical projects developed by the students on 2023/2, without any simulator

Project	**Description**	**H**	**F**	**I**	**R**
Fire alert	A dynamic fire monitoring, when turning on, the agent communicates with the firefighters' MAS, asking for official configuration parameters for the sensors.	√	√	√	√
Health monitoring	A solution for monitoring cardiac arrhythmia based on agents.	half	X	X	√
Home security	A system to manage seasonal room rentals. Embedded agents manage the unit, controlling access by password and rental period.	half	X	X	√
Landslide monitoring	A landslide monitoring system based on agents integrating civil defense and smart homes.	√	√	√	√
Pet feeder	An agent deposits food and water in a receptacle and monitors consumption, notifying the human responsible whenever it is low.	half	X	X	√
Vegetable garden	An irrigation system for domestic vegetable gardens.	half	half	X	√

Table 2. Description of the practical projects developed by the students in 2024/2, with an integrated simulator

Project	**Description**	**H**	**F**	**I**	**R**
Parking management	Cognitive agents are responsible for parking space reservation, access control, and automated billing	√	√	√	√
Animal dietary control	An automated system to identify animals and setup their diets according to their weight	**Simulation**		√	√
Dam level control	An automated system to detect dam levels and warn authorities in case of a dangerous level	**Simulation**, then physical prototype		√	√
Market stock control	An automated system to detect item inventory in markets and automatically refill them	**Simulation**		√	√

6. Related works

Given the plethora of available simulation platforms, many works in the literature propose ways of integrating agent reasoning with existing platforms. It has been proposed a 3-tier architecture (Singh et al. 2016) to develop such integration: a generic layer, a system layer, and an application layer. This architecture, however, is aimed to complex systems simulations with hundreds or thousands of agents. It is not meant to simulate control over real hardware platforms and its interfaces. Also, our work avoids the need to program any specific integration layer since it uses the operating system itself as an integration layer.

Similarly, another 3-layer architecture (Davoust et al. 2020) composed by a simulated environment, a state synchronization, and a BDI framework aims to solve this challenge. This architecture has been also extended to work with ROS2 (Gavigan and Esfandiari 2021b) by changing the simulated environment with ROS, which interfaces to application nodes that finally interact with the environment, which can be a simulator. This way, it uses 4 layers to implement the integration: the agent dimension, ROS, specific application nodes, and the environment. In contrast, our solution integrates the BDI framework directly to the simulator by using the emulated serial channel, avoiding any kind of integration layers between them.

Regarding educational approaches for teaching and learning agents theory and practice, many works have proposed simulators or self-made tools in the literature. An intelligent agents course, whose main objectives were to introduce the notion of MAS, areas of applications, and its advantages for developing complex software systems, uses the NetLogo as a simulation tool to assign assessed work to the students, citing their main benefits of being easy to install and having many examples for the students to run (Sakellariou et al. 2008). The work extended the NetLogo simulator with libraries to implement BDI agents since this capability was not native to that simulator. The SimFleet (Palanca et al. 2021) is used for a Master's degree in an AI course. The goal is to represent open fleets of vehicles and to allow the implementation of different negotiation and cooperation algorithms. The tool uses SPADE3 (Palanca et al. 2020) for programming agents, which defines a set of programmable behavior types that these agents can run. Then, the students are tasked with implementing different customers' behaviors.

Exploring the process of learning MAS is not new. However, they all have similar approaches: they use a coupled endogenous environment to provide simulations or create extensions to methodologies and agent-programming languages for existing simulators. In our proposal, we decouple the environment from the reasoning layer, transferring the responsibility of communicating with simulators and self-made applications to the OS's serial layer. The student focuses on the agent reasoning, abstracting simulation properties in most cases.

Educational robotics activities have also been widely explored with MAS. For example, three case studies help to understand how agents and hardware interfacing work: a firefighting scenario, a robot drama, and using social robots to enhance student's learning (Bravo and Páez 2023). The firefighting scenario provides a programming interface for students and physical robots to deploy solutions. The programming interface allows the description of the robot's behaviors with a user interface (an application). However, the solutions are generally dedicated to a specific robotic platform and do not adopt BDI in the learning process. The BDI always imposes challenges when adopted in embedded systems. The amount of perceptions and beliefs can lead to bottlenecks in processing the upcoming events that agents must comply with. In our proposal, the reasoning layer is decoupled from the environment, which means that both sides of the system can be exchanged if desired. Furthermore, the Jason Embedded distribution has already tackled several challenges in dealing with BDI agents and hardware interfacing.

7. Conclusion

This work presented an exogenous environment simulation approach using an emulated serial communication where the agent can directly manipulate virtual devices. Furthermore, several proofs-of-concept of this approach were presented: one with an Arduino simulator, a specific desktop application, and a robot simulator. Then, we built a real-world robot, showing how easily we can deploy its MAS reasoning to it. Finally, we also presented a report on an undergraduate course during an academic semester before and after using the proposed approach, showing that it helped students implement and validate their ideas easily. In educational embedded MAS scenarios, the students can now develop the reasoning layer independent from the environmental layer, focusing on the learning process of how the agents' mind works. For future work, it is important to make available other simulator examples to provide more options to the students.

REFERENCES

Boissier, O., Bordini, R. H., Hübner, J. F., Ricci, A., and Santi, A. (2013). Multi-agent oriented programming with JaCaMo. *Science of Computer Programming*, 78(6):747–761. DOI: 10.1016/j.scico.2011.10.004.

Bordini, R. H. and Hübner, J. F. (2006). BDI Agent Programming in AgentSpeak Using Jason. In *Computational Logic in Multi-Agent Systems*. Springer, Berlin. DOI: 10.1007/11750734_9.

Bratman, M. E., Israel, D. J., and Pollack, M. E. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(3). DOI: 10.1111/j.1467-8640.1988.tb00284.x.

Bravo, F. A. and Páez, J. J. (2023). Exploring the use of MAS in Educational Robotics Activities. *IEEE Transactions on Learning Technologies*, 16(6). DOI: 10.1109/TLT.2023.3277715.

Burattini, S., Ricci, A., Mayer, S., Vachtsevanou, D., Lemee, J., Ciortea, A., and Croatti, A. (2022). Agent-Oriented Visual Programming for the Web of Things. In *Proceedings EMAS 2022*, Auckland, New Zealand. IFAAMAS. DOI: 20.500.14171/109205.

Davoust, A., Gavigan, P., Ruiz-Martin, C., Trabes, G., Esfandiari, B., Wainer, G., and James, J. (2020). An Architecture for Integrating BDI Agents with a Simulation Environment. In Dennis, L. A., Bordini, R. H., and Lespérance, Y., editors, *Engineering Multi-Agent Systems*, pages 67–84, Cham. Springer International Publishing.

Freitas, B., Lazarin, N., and Pantoja, C. (2023). A Proposal for a Serial Port Emulator for Embedded Multi-Agent Systems. In *Proceedings of the 17th Workshop-School on Agents, Environments, and Applications*, Porto Alegre, RS, Brasil. SBC. DOI: 10.5753/wesaac.2023.33437.

Gavigan, P. and Esfandiari, B. (2021a). Agent in a Box: A Framework for Autonomous Mobile Robots with BDI. *Electronics*, 10(17). DOI: 10.3390/electronics10172136.

Gavigan, P. and Esfandiari, B. (2021b). Agent in a Box: A Framework for Autonomous Mobile Robots with Beliefs, Desires, and Intentions. *Electronics*, 10(17):2136.

González, S. (2023). SimulIDE Circuit Simulator. `https://simulide.com`.

Karaduman, B., Tezel, B. T., and Challenger, M. (2023). Rational software agents with the BDI reasoning model for Cyber–Physical Systems. *Engineering Applications of Artificial Intelligence*, 123. DOI: 10.1016/j.engappai.2023.106478.

Lazarin, N. and Pantoja, C. (2015). A Robotic-agent Platform For Embedding Software Agents using Raspberry Pi and Arduino Boards. In *Proceedings of the 9th Workshop-School on Agents, Environments, and Applications*, Porto Alegre, RS. SBC. DOI: 10.5753/wesaac.2015.33308.

Lazarin, N., Pantoja, C., and Viterbo, J. (2023). Towards a Toolkit for Teaching AI Supported by Robotic-agents: Proposal and First Impressions. In *Proceedings of XXXI Workshop sobre Educação em Computação (WEI 2023)*, Porto Alegre. SBC. DOI: 10.5753/wei.2023.229753.

Lazarin, N. M., Pantoja, C. E., and Viterbo, J. (2024). Dealing with the Unpredictability of Physical Resources in Real-World Multi-agent Systems. In Rocha, A. P., Steels, L., and Van Den Herik, J., editors, *Agents and Artificial Intelligence*, volume 14546, pages 48–71, Cham. Springer Nature Switzerland. DOI: 10.1007/978-3-031-55326-4_3.

Linux Kernel Organization, Inc. (2023). TTY — The Linux Kernel documentation. `https://www.kernel.org/doc/html/latest/driver-api/tty/`.

Michel, F., Ferber, J., and Drogoul, A. (2009). Multi-Agent Systems and Simulation: A Survey from the Agent Community's Perspective. In *Multi-Agent systems: Simulation and applications*. CRC Press.

Michel, O. (1998). Webots: Symbiosis Between Virtual and Real Mobile Robots. In Goos, G., Hartmanis, J., Van Leeuwen, J., and Heudin, J.-C., editors, *Virtual Worlds*, volume 1434, pages 254–263. Springer, Berlin, Heidelberg. DOI: 10.1007/3-540-68686-X_24.

Onyedinma, C., Gavigan, P., and Esfandiari, B. (2020). Toward Campus Mail Delivery Using BDI. *Journal of Sensor and Actuator Networks*, 9(4):56. DOI: 10.3390/jsan9040056.

Palanca, J., Jordán, J., and Julián, V. (2021). Using learning by doing methodology for teaching multi-agent systems. In *INTED2021 Proceedings*. DOI: 10.21125/inted.2021.0794.

Palanca, J., Terrasa, A., Julian, V., and Carrascosa, C. (2020). SPADE 3: Supporting the New Generation of Multi-Agent Systems. *IEEE Access*, 8. DOI: 10.1109/ACCESS.2020.3027357.

Pantoja, C. E., Jesus, V. S. D., Lazarin, N. M., and Viterbo, J. (2023). A Spin-off Version of Jason for IoT and Embedded Multi-Agent Systems. In *Intelligent Systems*, volume 14195, pages 382–396. Springer Nature Switzerland, Cham. DOI: 10.1007/978-3-031-45368-7_25.

Pantoja, C. E., Stabile, M. F., Lazarin, N. M., and Sichman, J. S. (2016). ARGO: An Extended Jason Architecture that Facilitates Embedded Robotic Agents Programming. In *Engineering Multi-Agent Systems*, pages 136–155. Springer, Cham. DOI: 10.1007/978-3-319-50983-9_8.

Ricci, A., Santi, A., and Piunti, M. (2012). Action and Perception in Agent Programming Languages: From Exogenous to Endogenous Environments. In *Programming Multi-Agent Systems*, pages 119–138. Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-642-28939-2_7.

Sakellariou, I., Kefalas, P., and Stamatopoulou, I. (2008). Teaching Intelligent Agents using NetLogo. In *Proceedings of the ACM-IFIP IEEIII 2008 - Informatics Education Europe III Conference*, pages 209–221, Venice. Ca' Foscari. URL: https://scholar.google.com.br/scholar?cluster=11874270740786254465.

Singh, D., Padgham, L., and Logan, B. (2016). Integrating BDI Agents with Agent-Based Simulation Platforms. *Autonomous Agents and Multi-Agent Systems*, 30(6):1050–1071.

Souza De Jesus, V., Mori Lazarin, N., Pantoja, C. E., Vaz Alves, G., Ramos Alves De Lima, G., and Viterbo, J. (2023). An IDE to Support the Development of Embedded Multi-Agent Systems. In *Advances in Practical Applications of Agents, Multi-Agent Systems, and Cognitive Mimetics. The PAAMS Collection*. Springer, Cham. DOI: 10.1007/978-3-031-37616-0_29.

Yim, I. H. Y. and Su, J. (2024). Artificial intelligence (AI) learning tools in K-12 education: A scoping review. *Journal of Computers in Education*. DOI: 10.1007/s40692-023-00304-9.