Physical Artifacts for Agents in a Cyber-Physical System: A Case Study in Oil & Gas Scenario (EEAS)

1st Fabian Cesar Pereira Brandão Manoel
Federal Center for Technological Education (CEFET/RJ)
Rio de Janeiro, Brasil
0000-0003-0614-0592

2nd Carlos Eduardo Pantoja
Federal Center for Technological Education (CEFET/RJ)
Rio de Janeiro, Brasil
0000-0002-7099-4974

3rd Leandro Marques Samyn
Federal Center for Technological Education (CEFET/RJ)
Rio de Janeiro, Brasil
0000-0002-0733-4172

4rd Vinicius Souza de Jesus
Federal Center for Technological Education (CEFET/RJ)
Rio de Janeiro, Brasil
souza.vdj@gmail.com

***Abstract*—Physical devices have been integrated with artificial intelligence to create Cyber-Physical Systems (CPS). Multi-Agent Systems (MAS) can provide pro-activity and autonomy using agents, social organizations, and environment modeling by means of artifacts. Usually, some works that use MAS for interfacing physical environments employ agents accessing directly all the available data of the environment, which could overload this agent. This issue could be avoided if there were tools to facilitate the integration of sensors and actuators as artifacts into the physical environment. Therefore, the objective of this work is to create physical artifacts capable of accessing hardware devices from a physical environment to be used by agents in a MAS. As the Oil & Gas industry demands robustness in its equipment and an ability to do predictive maintenance, a case study including MAS and CPS was developed and some tests were carried out to validate the functioning of physical artifacts.**

***Index Terms*—Physical Artifact, Physical environment, Oil & Gas Industry**

I. Introduction

In the last years, the agent approach has been switching from simulated to physical applications where Multi-Agent Systems (MAS) have been used to interact and control devices working in dynamic environments [1] [2] [3] [4]. In general, some approaches define four dimensions that guide a MAS implementation: agency, environmental, organisational [5], and interaction [6]. Agents interact in an environment according to their implemented beliefs, desire, and intentions (BDI); Artifacts provide operational functions and observable properties for agents, and they represent non-cognitive entities situated in workspaces; organizational dimension models the society notion and the collective norms of the agent's behavior; interaction dimension models the interaction between the three dimensions (agent, environment, and organization). In parallel, when connecting computing elements to physical elements, such as embedded computers connected in a network, it maintains a system known as Cyber-Physical Systems (CPS) [7].

When considering physical environments, rarely they are explored considering other dimensions aside from the agent one. In an agent application in the oil domain, only the agency dimension is considered [8]. The agent performance depends directly on the amount of information that an environment has to offer. There is an approach called ARGO that allows agents to collect data directly from sensors and process them as beliefs in their Belief-Desire-Intention (BDI) reasoning cycle [9]. This process requires reading all the sensors during every cycle execution even if the data are not necessary for the agent, at that moment. Some filtering techniques are available, but they can only be applied after the data has been collected [10].

Initial laboratory experiments for BDI agents in a Web-of-Cell context [11] and a proposed model of many resources of the factory following the A&A [1] are works that consider physical environments using the notion of artifacts. However, both implementations are domain-specific. Artifact is a suitable notion for agents to interact with physical objects in a CPS. When MAS employs artifacts, agents are able to access the physical environment according to their need. It avoids the agents to collect unnecessary data. However, traditional agent-oriented programming languages do not provide direct approaches to access physical environment and they are limited to a particular application domain.

Some initiatives, like the Predictive Maintenance Program (PMP) reveal the importance of collecting data from sensors in the environment to perform predictive maintenance [12]. This importance can also be seen in the Oil & Gas industry because predictive maintenance can minimize economic and environmental losses from poor preventive maintenance.

The objective of this work is to provide physical artifacts for interfacing hardware devices from a physical environment to be accessed by MAS in a CPS. In order to develop theses Physical Artifacts, it will be created an extension of

DOI reference number: 10.18293/SEKE2020-154

CArtAgO artifacts that communicates with microcontrollers using serial interfaces. A case study will be presented in a scenario considering a physical engine as an artifact in the Oil & Gas field.

This paper is organized as follows: Section 2 presents the theoretical background to understand the idea; Section 3 shows the methodology used to implement Physical Artifacts, Centralized Layer, and the scenario of study; Section 4 presents the related works and the Section 5 concludes this work.

II. Theoretical Background

Multi-Agent technologies provide tools for distributed control, decentralization, adaptation, and openness. These characteristics can be found in four MAS domains: (i) agent-oriented programming languages, (ii) interaction languages and protocols, (iii) environment frameworks, architectures and infrastructure, and (iv) organizational systems. These perspectives lead MAS to the four dimensions of development, such as described by JaCaMo approach [13] and complementary works [6]: *organization*, where rules and missions are defined to ensure the society behavior; *agent*, where BDI agents are implemented; and *environmental*, responsible to integrate the external environment and agents using artifacts with operational functions; *integration*, that represents program languages responsible to ensure integration between agents, artifacts and organization rules.

In the Multi-Agent field, artifacts are Activity Theory and Distributed Cognition-based computational devices existing in environments and capable of performing a particular function or service that agents can explore. Regarding the agent/artifact relationship, there are two different types of external objectives attached to an artifact: (i) *use value*, where external goals head the artifact selection by agents; and (ii) *use*, which is associated with agent's internal goals [14]. Therefore, three distinct aspects characterize the relationship between agents and artifacts: agents can *select*, *use*, and *construct/manipulate* artifacts, where the latter occurs when the artifact does not exist and needs to be created.

Artifacts are composed of four elements [13]: *User Interface (UI)*, *Operating Instructions (OI)*, *Function*, and *Structure and Behavior*. *User Interface (UI)* is a set of operations that agents can call to use the artifact; *Operating Instructions (OI)* describe how the artifact should be used to access its functionality; *Function* is the purpose of the artifact's existence; and *Structure and Behavior* are the internal characteristics of artifacts that define how it is implemented [15].

For programming the environmental dimension for agents, there is the CArtAgO framework, which is based on three main pillars. The (i) Agent Body is the part of an agent where artifacts represent some behaviors that it can access and control but it is not part of their internal reasoning; (ii) Artifacts are the components identified in a Workspace that agents or any part of their body can interact with; A (iii) Workspace is used to define the desktop topology. Artifacts and Agent Bodies are stored in these Workspaces, where the relation between them is established. Then, artifacts must be within a specific Workspace so that agents can use. Consequently, events generated by these artifacts can only be seen by agents living in the same Workspace [16].

Using artifacts that are only accessible within their workspaces may not represent the best approach to be employed in dynamic scenarios since it restricts agents that are not originally from these workspaces to access the environment's resources. In dynamic scenarios agents can come and go freely and they can compete for each available component. Moreover, the environment should be open for any entity that intends to enter it. However, even CArtAgO, and other languages and frameworks that consider the development of artifacts do not provide a distributed and open characteristics for environments.

III. Methodology

In CPS, the use of environmental objects by computational entities is an essential factor that helps these entities to adapt to environments with dynamic characteristics. Besides, these environments are increasingly demanding automation, pro-activity, and cognition. While the agents layer promotes computational intelligence and the Organization layer promotes social rules, the Artifacts layer encourages the modeling of objects from the external environment. Although the environment layer is ideal for representing objects from the external MAS environment, there are approaches that still transfer this responsibility to agents. Therefore, this work presents a solution to apply MAS in physical, dynamic, and intelligent environments using Physical Artifacts to connect MAS artifacts to ATMEGA microcontrollers. A scenario will be presented with instrumented engines in the Oil & Gas industry with a focus on predictive maintenance implemented in MAS with Physical Artifacts.

A. Oil & Gas Engine Scenario

When it comes to equipment maintenance, the natural approach is prevention, which aims to replace defective components or parts from time to time. However, this type of maintenance can be costly from a financial and environmental point of view. From an economic point of view, the periodic replacement of a specific component can make the process more expensive; from the environmental point of view, the equipment may present failure situations before the replacement period and cause accidents to the environment. On the financial side, prediction is better than prevention because predicting that the equipment's life cycle will be longer than usual can avoid spending on unnecessary maintenance. On the environmental side, predicting that equipment is being damaged can result in support before it is damaged. Following this idea, the Oil & Gas industry benefits in the economic and environmental fields with predictive maintenance.

The Strategic Petroleum Reserve (SPR) - that is an Oil emergency fuel storage unit - is composed of several engines that supply power to the pumps that move a large amount of oil in the unit. As the SPR does not have a continuous operation, the motors do not remain connected at all times,

which hinders the temporal precision that is necessary to carry out preventive maintenance. Therefore, prediction techniques on engines such as vibration analysis, thermography, and oil analysis can be useful to reduce maintenance costs and prevent accidents. In addition to sensors for analysis, the motors have actuators that define their operation and can also be activated intelligently to minimize the risk of equipment degradation.

As a motivation to use prediction as an approach, the Predictive Maintenance Program (PMP) proposed in 1994 sets targets for reducing maintenance costs by 20% by the third year of operation of this PMP [12]. With PMP, it is possible to offer accuracy to equipment operators as to when intervention should occur. In this case, expenses with unnecessary maintenance and the risk of accidents would be reduced.

B. The Physical Artifacts

A Physical Artifact is an extension of the standard MAS Artifact capable of integrating with a physical Device in the environment to collect its sensor data or send actuation commands to actuators. For Physical Artifacts, a Device is an object in the physical environment composed of a microcontroller with sensors or actuators. Besides, a Device must have communication functions between the microcontroller and another external computational entity to provide readings on its sensors and receive commands for its actuators. Therefore, to become a Physical Artifact, an object in the physical environment must assume the characteristics of a Device. In this case, the Operation Functions of this Artifact can be implemented to read the sensors and operate directly on the actuators of this Device. For example, in the scenario of engines in the Oil & Gas industry, it is necessary a microcontroller in them that sends the data from the vibration sensors, thermography and oil to this Artifact.

To create Physical Artifacts, the Artifacts implementation of CArtAgO framework was employed. We chose CArtAgO because it is used to create the environmental dimension of the JaCaMo framework for Jason and because both CArtAgO and Jason are widely used in the academia. In the hierarchical structure of CArtAgO, the Physical Artifact is a child class of the Artifact class. Therefore, physical artifacts can also implement Observable Properties and Operations. It is expected with this integration to allow MAS integration with CPS without overloading agents.

Once incorporated as CArtAgO Artifacts, Physical Artifacts must be able to communicate with Devices in the physical environment. For this, the serial interface Javino [17] was employed, which is a library that implements a protocol for exchanging messages between low-level hardware (microcontrollers) and high-level software (Java). The choice for Javino is justified because it is a serial communication library that handles error detection, unlike libraries based on serial ports, such as RxTx and JavaComm. The messages exchanged between hardware and software follow a format composed of 3 fields: 2 bytes of a pre-scope that is used to identify the beginning of the message, 1 byte to represent the size of the main content of the message, and finally, 256 bytes containing the content of the message to be passed. The loss or collision of information from the past message is verified through the pre-scope field and the size field: the receiver validates the content in the pre-scope; if the preamble is correct, the size field helps to verify that the message arrived at the correct size. If all verification is validated, the message is used; otherwise, the message is discarded. Javino offers three modes of operation: **Send**, **Request**, and **Listen** modes. The **Send** mode provides simplex message transmission from software to hardware; the **Request** mode offers half-duplex message transmission (the hardware responds to the message sent); the **Listen** mode allows the transmission of simplex messages from hardware to software. Another factor that justifies the choice of Javino is the possibility that it is designed to be multi-platform and can be used in ATMEGA, PIC, or Intel Families microcontrollers.

The use of Javino as a connection bridge must be analyzed both on the side of the abstraction (Physical Artifact) and the embedded hardware (Device). On the Physical Artifact side, the Javino implementation class for high-level software is added as an attribute to the *PhysicalArtifact* class and instantiated directly in the constructor. All child classes of *PhysicalArtifact* must define, via abstract method, the following values: **Serial Port** that will be used to connect the artifact to the microcontroller (method *String definePort()*), a **Number of Attempts** to send a message (method *int defineAttemptsAfterFailure()*), and **Timeout** in milliseconds between one attempt to send a message and another *int defineWaitTimeout()*). In addition, the class *PhysicalArtifact* has the implementation of the *String read()* method, which performs reading from a physical device in Javino **Listen** mode; and also has implementation of the *void send (String message)* method, which sends messages in **Send** mode to the microcontroller. Figure 1 shows the architecture of Physical Artifacts in a MAS communicating with a physical environment.

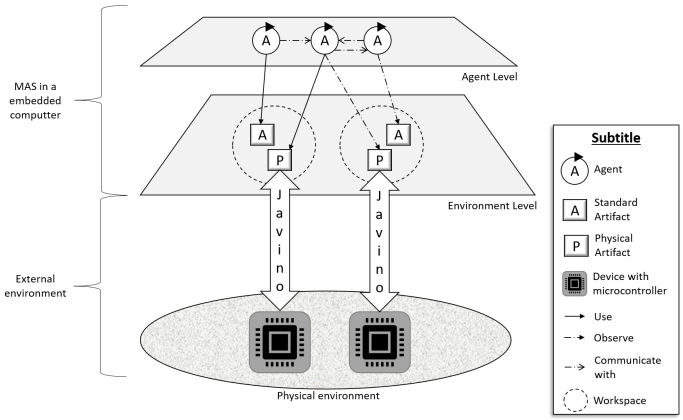

Fig. 1. The architecture of a MAS integrated with a physical environment showing only the Agent level and the Environment level. At the Environment level, Standard Artifacts are together to Physical Artifacts that connects to a Device in the physical environment using Javino middleware.

C. Engine Scenario Prototype

To represent the engine scenario in the Oil & Gas industry and test the Physical Artifacts approach, a prototype of an instrumented engine with a sensor was created and connected to a MAS that will control it, as shown in the Figure 3. The physical prototype consists of a fan to represent the motor actuator, a temperature sensor, and LEDs that indicate the state of operation of the motor. An Arduino Mega was used as a microcontroller that contains all the sensors and actuators of the prototype. Besides, Arduino Mega is responsible for exchanging messages with MAS. This physical configuration configures the physical prototype as a Device that can be used by a Physical Artifact.

The engine designed in the prototype has the following operations: turn on, off, block use, unlock use. In particular, the blocking operation is used by operators when the engine is in an abnormal condition and should not be operated. In this prototype, the motor has three possible states: **Ready to be Operated**, represented in the first lower frame of the Figure 3, where the prototype is turned off and unlocked; **On**, represented in the second lower frame of the Figure 3 (represented by the connection of two of the three LEDs); **Blocked**, represented in the third lower frame of the Figure 3, where the motor is blocked for use (indicated by the red led).

On the Arduino side, the Javino library is imported and used as a support in sending and receiving messages to the Physical Artifact. The Arduino was programmed to send data from the temperature sensor whenever a message arrives from the Artifact that requests it. In addition, the Arduino operates the engine whenever the Physical Artifact requests one of the available forms of operation.

On the MAS side, the Motor Artifact described in Figure 2 was created that extend a Physical Artifact. In this Motor Artifact, operations are implemented to read the temperature sensor, turn on, off, lock, and unlock the motor. In addition, an Agent Manager was created in MAS to control operations. For this, this Agent creates the Artifact Motor and starts a basic cycle of activities to test all the operations provided by Artifact. In the upper left corner of Figure 3, the running agent log is displayed.

When modeling the class diagram in Figure 2, a representation of the engine for the system with the respective registration information can be seen. Besides, a model of sensors and sensor measurements was created to record the data in a MySQL database. With this, an application was developed to allow monitoring at the level of the engine operator so that it can visually diagnose the engine situation. In the upper right corner of the Figure 4, is showed a graph with temperature measurements of the environment where the prototype is located. These measures is in degrees Celsius unit and are provided by the Physical Artifact that reads the engine. The variation in the graph can be analyzed by an operator in the field of work and serve as a variable in the generation of a prediction diagnosis, for example, associating that the increase in temperature crossed with other data, means loss of engine life.

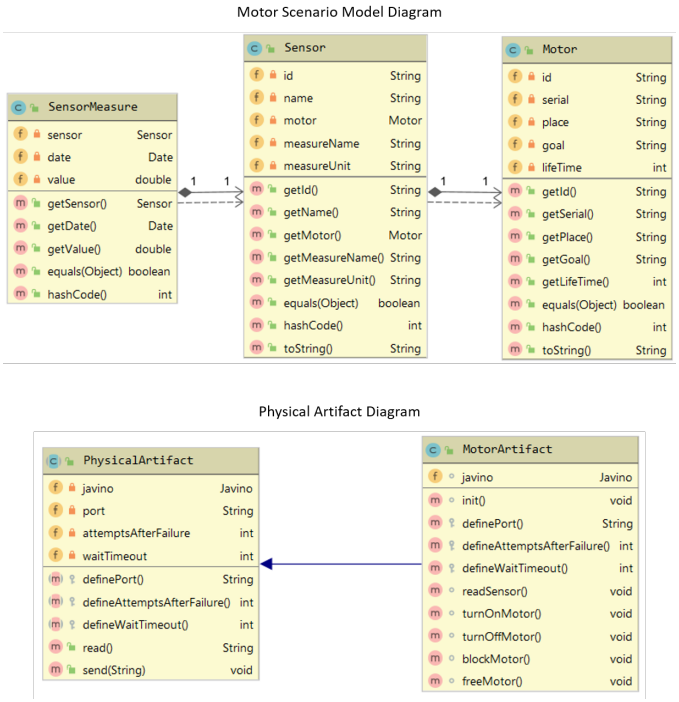

Fig. 2. Class diagram of the model made to represent the Engine scenario.

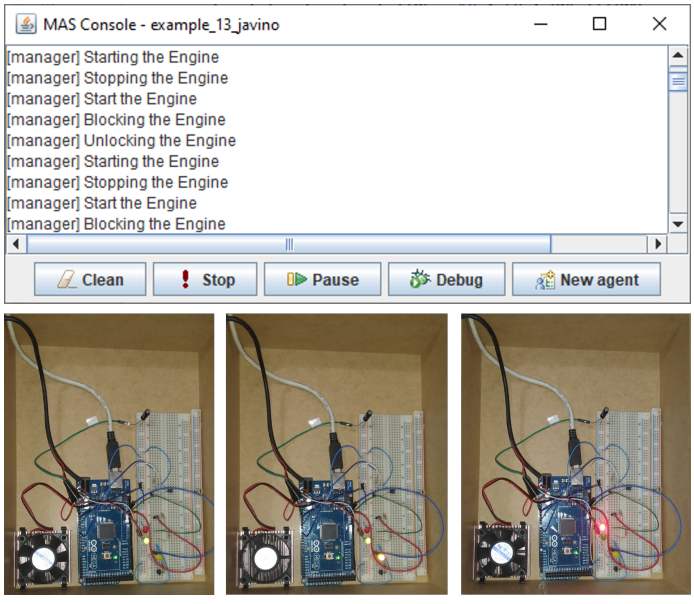

Fig. 3. Engine scenario in the Oil & Gas industry in execution: Agent Manager performing control and monitoring of the Physical Artifact, and prototype of the engine connected to the MAS.

D. Experimental Evaluation

From the elaborated scenario, tests were done to validate the functioning of Physical Artifacts in the physical environment. For this, the requirements of the framework were raised to support the experiments: (i) the Physical Artifact must be able

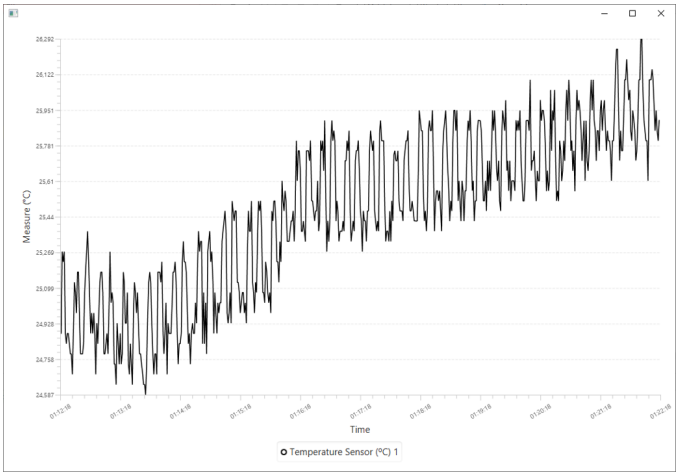

Fig. 4. An application that displays a line graph with temperature measurements of the environment in which the prototype is based on time, in degrees Celsius, provided by the Physical Artifact. The X-axis is expressed in hours, minutes, and seconds. The temperature measurement is an example, which could be replaced by measurements from other sensors.

TABLE I
THE CASE STUDY DESIGN

Design	Description
Objective	Analyze the functioning of the MAS Physical Artifact in a physical setting.
Case	The Physical Artifact will be connected to an oil and gas engine that must be monitored and controlled to perform predictive maintenance.
Questions	Is the Physical Artifact capable of sending and receiving information through Javino? Do physical Artifacts respond to agents' requests in up to one second? Do Physical Artifacts stay running for a minimum of 24 hours?
Method	Observation method with a low degree of researcher interaction.

to send and receive information using Javino; (ii) the Physical Artifact must be able to respond to agents' requests within one second, which is considered acceptable within the high-level programming field; (iii) the Physical Artifact must be able to function in a 24-hour period in the worst case.

Based on these premises, routine tests were carried out, where the agent requested the operations to start, stop, restart, lock, and unlock the engine. At each operation, the agent requests data from sensors ten times. It was concluded that the commands from the Artifact work normally. In addition, a throughput test was carried out between the agent's command and the execution of the Artifact, and it was observed that the waiting time is below one second. Finally, the Agent Manager was kept in operation for 24 hours, where it was observed that the Artifact continues to respond with a failure rate of 0%. Table I shows the case study analysis of this scenario and Table II shows the results from tests.

IV. RELATED WORKS

Physical environments have been demanding computational systems more proactive, autonomous, and adaptable to solve

TABLE II
EXPERIMENTAL EVALUATION RESULTS

Test	Description	Result
The connection between Physical Artifact and Microcontroller	Percentage of success (%) when exchanging data with the microcontroller	100%
Physical Artifact Responses to the Agent	Maximum time (milliseconds) that an Agent takes to receive data from the Artifact	1000ms
Physical Artifact execution time	Checks whether the Physical Artifact remains running for 24 hours	Yes

increasingly complex problems. The community has been developing some works using MAS in industry as an attempt to increase pro-activity and autonomy in the production chain. There is a work in the Oil & Gas industry which uses BDI agents to filter alarms that are generated by different conditions [8]. This filtering considers that an operator is not able to observe a broad set of alarms and act on them. Besides, excessive alarms can hide an important occurrence, and therefore there must be an intelligent system capable of filtering this data. For this, an alarm management system was developed using agents able of reading sensors and act on devices. However, agents were programmed directly connected to environments — in case of agents responsible for only one sensor or only one actuator — without using the notion of artifacts. As a result, agents could face bottlenecks in their reasoning due to the need to be continuously collecting data without necessarily using it.

ARGO [9] is a customized *Jason* agent's architecture that allows interactions with physical devices such as sensors and actuators. For this, a serial interface between microcontrollers and *Java* programming language was developed to collect all data from the environment to be added to the agent's belief base. The generated data flow overloads ARGO agents and filtering techniques [10] can be employed to select which perceptions the agent has to focus on. However, the sensors and actuators are available only for a specific MAS and they are not shareable. Besides, ARGO may experience a decrease in computational performance as the amount of information to be perceived increases. Both works could benefit from an approach that exposes sensors and actuators as shareable resources in the IoT.

Given the overload on the agents and aiming to take advantage of the MAS environmental modeling resources, some works developed solutions applied to physical environments. In the energy sector, a Web-of-Cell (WoC) approach [11] uses MAS to help design and test distributed solutions. For this, the Jason framework is used to develop BDI agents; environment modeling is done using the CArtAgO framework, which allows creating a bridge between the agent layer and the environment layer. The communication between the modeled environment and the physical environment was done by the communication infrastructure of the intelligent and configurable network laboratory (SYSLAB). However, this communication bridge is

strongly linked to the SYSLAB structure, which still does not help in the mission to facilitate implementations with MAS that involve environmental modeling.

MAS heterogeneity has been increasingly required in challenges that integrate CPS with environmental resources. In the Industry 4.0 concept, equipment and sensors must be integrated into the same system using the most diverse communication protocols. Camel Artifact [1] is a component that uses Java-based message routing and mediation technology (Apache Camel) in artifacts. A CamelArtifact makes it possible to transform physical devices into Artifacts in a more generic way than the WoC approach because several communication protocols can be used to create the bridge between the physical and the computational environment. For this, routing is done that directs the messages from a device to the specific artifact. However, although this work does not depend on a particular protocol of communication between physical devices and MAS, there is still a strong dependence on Apache Camel technology that guarantees message routing. Perhaps, an approach that integrates artifacts with microcontrollers can offer even more heterogeneity because it will allow configurations of these devices more directly and at a low level. If these artifacts were shareable between different MAS, the collected data would become resources of the environment that agents from any MAS could exploit.

V. Final Considerations

Normally, MAS applications using physical environments for CPS overloads agents with data coming from sensors and actuators. Besides, when they are not overloaded, the connections to these kind of artefacts are bounded to the provided solution. Based on that, this work presented an extension of CArtAgO for providing Physical Artifacts without generating overload to agents using a serial interface for communicating with heterogeneous microcontrollers.

In order to create Physical Artifacts, several technologies were employed such as Jason and CArtAgO frameworks, the Serial Interface Javino, and microcontrollers. CArtAgO's Artifact was extended to allow Javino to interact with sensors and actuators connected to microcontrollers. The proposed extension was tested in an engine scenario for Oil & Gas domain. The results showed that our approach is suitable for designing CPS using MAS and Physical Artifacts.

A future issue to be considered is that the fact that artifacts to be accessible only within their workspaces can make it challenging to implement in dynamic scenarios because this restricts agents that are not from this MAS. If artifacts could be accessed by agents from another MAS, it could be possible to create a multi-purpose layer of physical artifacts to be consumed by different agents. As future works, we intend to create a shareable layer of artifacts to be used along with the Internet of Things. Agents from different MAS, or any other technology, could compete for Physical Artifacts. Besides, we will extend the scenario of motors for Oil & Gas to allow a middle layer capable of managing plans and rules for some situations when using those motors.

References

[1] C. Amaral, S. Cranefield, J. Hübner, and M. Roloff, "Giving camel to artifacts for industry 4.0 integration challenges," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11523 LNAI, 2019.

[2] T. Sanislav, G. Mois, S. Folea, L. Miclea, G. Gambardella, and P. Prinetto, "A cloud-based cyber-physical system for environmental monitoring," in *2014 3rd Mediterranean Conference on Embedded Computing (MECO)*, pp. 6–9, IEEE, 2014.

[3] S. Wang, J. Wan, D. Zhang, D. Li, and C. Zhang, "Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination," *Computer Networks*, vol. 101, pp. 158–168, 2016.

[4] B. Vogel-Heuser, C. Diedrich, D. Pantförder, and P. Göhner, "Coupling heterogeneous production systems by a multi-agent based cyber-physical production system," in *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, pp. 713–719, IEEE, 2014.

[5] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, and A. Santi, "Multi-agent oriented programming with jacamo," *Science of Computer Programming*, vol. 78, no. 6, pp. 747–761, 2013.

[6] M. R. Zatelli and J. F. Hübner, "The interaction as an integration component for the jacamo platform," in *International Workshop on Engineering Multi-Agent Systems*, pp. 431–450, Springer, 2014.

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

[8] N. Sanchez-Pi, L. Leme, and A. Garcia, "Intelligent agents for alarm management in petroleum ambient," *Journal of Intelligent and Fuzzy Systems*, vol. 28, no. 1, pp. 43–53, 2015.

[9] C. E. Pantoja, M. F. Stabile Jr, N. M. Lazarin, and J. S. Sichman, "Argo: A customized jason architecture for programming embedded robotic agents," *Fourth International Workshop on Engineering Multi-Agent Systems (EMAS 2016)*, 2016.

[10] M. F. S. Jr, C. E. Pantoja, and J. S. Sichman, "Experimental analysis of the effect of filtering perceptions in bdi agents," *International Journal of Agent-Oriented Software Engineering*, vol. 6, no. 3-4, pp. 329–368, 2018.

[11] D. Issicaba, M. Rosa, A. Prostejovsky, and H. Bindner, "Experimental validation of BDI agents for distributed control of electric power grids," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe, ISGT-Europe 2017 - Proceedings*, vol. 2018-January, pp. 1–6, 2018.

[12] R. J. Murry and B. F. Mitchell, "Cost savings from a practical predictive-maintenance program," in *Proceedings of Annual Reliability and Maintainability Symposium (RAMS)*, pp. 206–209, IEEE, 1994.

[13] A. Ricci, M. Viroli, and A. Omicini, "Programming mas with artifacts," in *International Workshop on Programming Multi-Agent Systems*, pp. 206–221, Springer, 2005.

[14] R. Conte, C. Castelfranchi, *et al.*, *Cognitive and social action*. Garland Science, 2016.

[15] A. Omicini, A. Ricci, and M. Viroli, "Coordination artifacts as first-class abstractions for mas engineering: State of the research," in *International Workshop on Software Engineering for Large-Scale Multi-agent Systems*, pp. 71–90, Springer, 2005.

[16] A. Ricci, M. Viroli, and A. Omicini, "Cartago: A framework for prototyping artifact-based environments in mas," in *International Workshop on Environments for Multi-Agent Systems*, pp. 67–86, Springer, 2006.

[17] N. M. Lazarin and C. E. Pantoja, "A robotic-agent platform for embedding software agents using raspberry pi and arduino boards," in 9^{th} *Software Agents, Environments and Applications School*, 2015.