A Robotic-agent Platform For Embedding Software Agents using Raspberry Pi and Arduino Boards

Nilson Mori Lazarin, Carlos Eduardo Pantoja

[1]CEFET/RJ – UnED Nova Friburgo – Av. Gov. Roberto da Silveira, 1900 – Nova Friburgo – RJ – Brasil

pantoja@cefet-rj.br, nlazarin@cefet-rj.br

Abstract. *This paper presents a robotic-agent platform to embed software agents into hardware devices. The platform consists in embed Jason framework in Raspberry Pi, allowing directly control of its pins, in order to control hardware devices, and Arduino to control sensors/actuators. So, it is necessary to prepare both hardware and software, and establish a communication between them. For this, it was developed the Javino library, which is a communication protocol for exchange messages between Java and Arduino using a serial port. It is also presented a three step methodology for supporting the robotic-agent development. An example using the proposed platform and methodology is presented. It was chosen a vehicle chassis, as the hardware robot, that is able to move and deviate from obstacles.*

1. Introduction

Intelligent Agents are autonomous entities capable of reasoning and are situated into an environment which can be physical or virtual [Wooldridge, 2000]. Nowadays, robotics is one field application for intelligent agents where a robotic agent can have actuators and sensors to interact with the physical environment. Besides, it is necessary a reasoning system embed into hardware to provide some intelligence to the robot. Multi-agent systems (MAS) are software or hardware systems that deals with subjects such as cooperation, distributed control, communication and fault-tolerance, in real-time situations. So, to implement a robotic agent microcontrollers are necessary, in order to control the actuators and sensors; an agent reasoning system using a program language; and a bridge between the hardware and the software layers. .

There are several microcontrollers like the ATMEGA328, part of Arduino board [Gertz; Justo, 2012], which is a widespread board for small automation projects; and the Raspberry Pi board that is a tiny computer with high processing capability [Upton; Halfacree, 2012]. However, the Arduino processing is very slow for an embed agent reasoning; and the Raspberry Pi although has a higher processing and memory power, it does not have an analogic interface, avoiding some sensors to be used. For the development of reasoning systems, there are several agent-oriented program languages like Jade [Bellifemine et al., 2013], a Java-based agent-oriented programming language, and JaCaMo [Boissier et al., 2011], which integrates three technologies for a Multi-Agent System (MAS) development: the Jason for agent reasoning programming; the CArtAgO for environments' artifacts programming; and the Moise+ for organizational programming.

Some projects try to integrate and embed a robotic reasoning into hardware such

as [Barros et al., 2014] that is an automated grounded vehicle, which uses the ATMEGA328 microcontroller to program the hardware basic functions; a java library for serial communication between the hardware and the simulated environment programmed in Java; and Jason framework for the agent programming. In [Calce et al., 2013] it is proposed an autonomous aquatic robot, which uses Arduino together with BeagleBoard and can move point-to-point deviating from obstacles.

The objective of this paper is to propose a platform for robotic agents, which uses the Raspberry Pi and Arduino together to provide the hardware controls, and uses Jason framework for the agent reasoning. Besides, an improvement of [Barros et al., 2014] platform is also presented. For the platform construction it will be developed: an action-ready hardware file for the 4WD Robot Chassis vehicle; The Javino, a library for serial communication between the Arduino and the Java Environment; a direct link between Jason and the Raspberry Pi using Pi4J library (available at http://pi4j.com/); and a simple example using the robotic-agent programmed with Jason controlling the 4WD Robot Chassis vehicle.

This paper is structured as follows: section 2 presents the methodology for supporting the robotic agent programming; in section 3, a simple example using the 4WD Robot Chassis vehicle and Jason framework is presented; section 4 discusses some related work; section 5 concludes the work; and finally the references used in this paper are shown.

2. The Robotic-Agent Platform

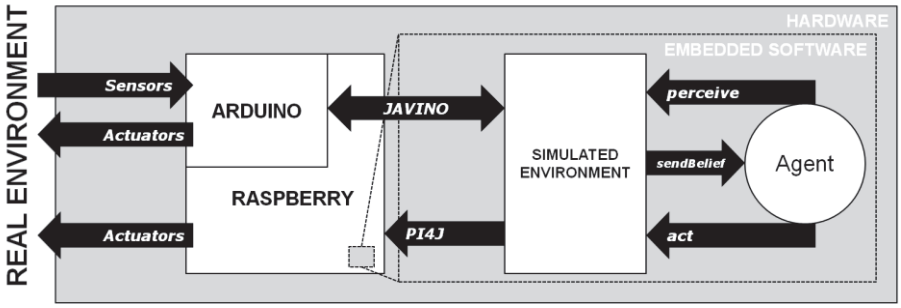
In this section, it is presented the proposed robotic-agent platform and a methodology to support the robotic-agent programming. The robotic-agent platform consists in an embedded software (agent) into a hardware platform (robot). The hardware platform is composed of the Arduino and Raspberry Pi boards, where Arduino is connected on the top of Raspberry Pi (using a USB port) to provide analogic hardwares to be used once Raspberry does not provide analogic pins. Therefore, it is possible to connect actuators devices using both boards, but is only allowed to use sensors connected with Arduino. One of the advantages of using Raspberry Pi is that it is possible to connect up to 127 Arduino boards (the USB device limit) in a single board.

The software agent is programmed using Jason framework and it is embedded into Raspberry because of its processing power and storage capacity. The platform uses the Javino to provide bidirectional serial communication between the hardware (Arduino) and the software agent; and the PI4J library for a direct control over Raspberry's digital pins. So, the agent is able to switch the Raspberry's pins values without any microcontroller intervention (the Raspberry uses the Python programming language to control digital pins), and senses the real environment or act upon it using Javino which transfers data perceived or sends a command for some action to be executed in real world. The platform is initialized once Raspberry starts. The proposed platform can be seen in figure 1.

The Raspberry can be used to manage both actuators and sensors, however this is not the main objective of the board since the GPIO are delicate and do not accept analogic inputs on its project. The Arduino's pins are more resistant then GPIO, besides it has available analogic pins. Furthermore, it is possible to communicate with Raspberry from an USB port. Therefore, in this platform, the Arduino manages sensors and actuators,

while Raspberry hosts the reasoning and control only some actuators with GPIO.

Figure 1. The proposed robotic-agent platform.



2.1. The Communication Protocol Using Javino

Here it is introduced the Javino which is a double-side library protocol for exchanging messages between an Arduino and Java program using a serial port. When a software agent needs to act in the environment, it is necessary to transmit its command to the microcontroller (Arduino ATMEGA328) where the actuators are connected. Since the agent cannot perform an action directly to the Arduino, because it is not possible to embed an agent software into a limited microcontroller (Arduino storage is due to 32Kb), it is necessary to provide a bridge between agent's external actions and microcontroller's functions. In the same way, when the agent needs to sense the environments, the data perceived from sensors has to be sent to the agent.

There are some libraries that use serial port to deal with one-side messages such as RxTx and IO library (both for Java). However, these libraries just provide message treatment for one platform side (environment) leaving the other side to the programmer. The Javino aims to fill this gap because it offers a double-side communication library based on the platform functioning: the Javino for Arduino and Javino for Java. They work together to provide a higher level of correctness in message exchange.

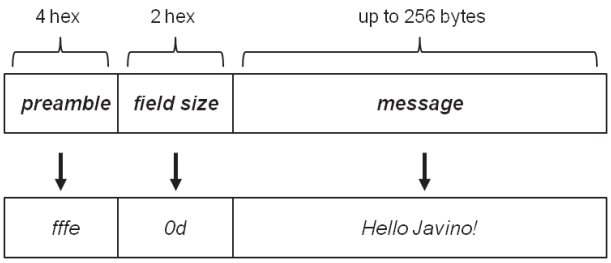


Figure 2. The message format.

For this, every message is composed of a preamble, a field size and the message content (figure 2). The preamble is a field composed of four hexadecimal characters that are used to identify the beginning of a message sent by an agent. The field size is composed of two hexadecimal characters that are used to calculate the message extension.

Finally, the last field is the message content up to 256 bytes. The preamble and the field size are used together to avoid errors in case of loss of information during the message transmission. For the sake of practice, Javino automatically mounts the message.

When a message is sent (either from agent to Arduino or vice versa), the Javino library starts to listen the serial port for arriving char-to-char messages. If there is any information arriving, the Javino stores this character analysing if it is part of the expected preamble. So, this process is repeated until the message has been completely received. Once the preamble is not confirmed, the Javino discards all information received until it finds a valid preamble. Otherwise, the Javino verifies the field size value to identify the message length. This process avoids error insertions and defines where a message starts and ends. Javino still has a pre-defined time-out to every character received, ending the actual process (if the time-out has been reached) and starts a new message listening.

After all done, the string message is mounted and returned. For the Arduino-side Javino, the message received will activate a hardware function or ask for data from sensors while for the environment-side Javino, the received message can be transformed into beliefs and sent to the software agent.

2.2. The Platform Methodology

In this section it is explained the methodology to support the robotic-agent programming. The methodology aims to guide the programmer between phases that will need some programming intervention, since the platform uses Arduino, Raspberry Pi and Jason. The methodology is composed by three steps that can be seen in figure 3.

The STEP 1 phase is composed by the selection of the hardwares (robot, sensors and actuators) to be connected in the Raspberry and/or Arduino boards. The selection consists in choose one pre-defined file for Arduino robots available with the methodology. Until now, there are two pre-defined robot chassis available: the Rover 5, a tank style vehicle; and the 4WD, a four-wheel drive vehicle. These pre-defined files guarantee all basic movements (forward, reward, left, right, turn, etc.) for those robot chassis. In the same way, the sensors and actuators selection consist in import a communication file where functions for connecting some devices (like GPS devices and compass) are defined. The methodology tries to provide a plug-and-play style, where the programmer do not need to interfere too much into the programming.

However, if the programmer desires to develop his own robot (or improve an existing one), it has to be programmed, in the Arduino board, all functions for the connected sensors and the actions that will be performed when the software agent executes an external action (both using the Javino library).

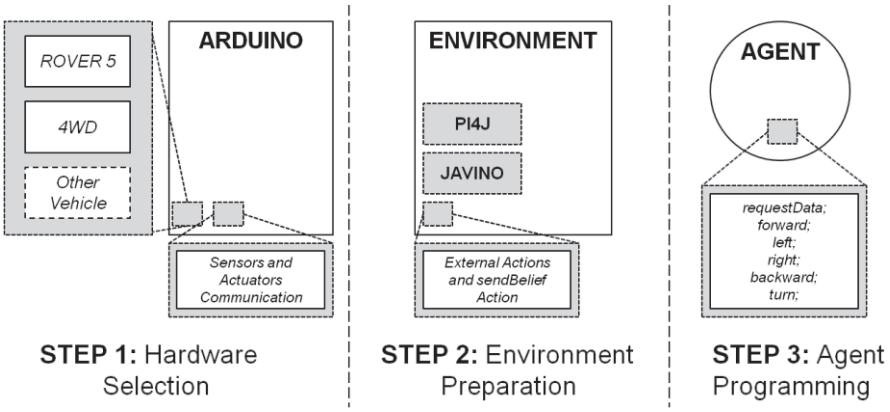


Figure 3. The robotic-agent programming methodology.

In STEP 2 it is necessary to prepare the agent's simulated environment. As the platform uses the Jason framework, it is used the basic Java environment where the agent's external actions (the actions that an agent wants to execute in real environment through the hardware layer) are programmed. So, when an agent wants to activate the actuators to move left for example, it is necessary to program the message (it is used "left" for the robot pre-defined files) that will be sent to the hardware. If the agent wants to get data from sensors, it is necessary to request it using an external action, where the hardware will return the data via Javino.

In order to have the communication between the software agent and the hardware it is necessary to import two libraries into Jason's simulated environment: i) The Javino library for Java which provides communication between Arduino and the environment, and; ii) The Pi4J library which provides functions for directly controlling the Raspberry pins (without any intervention in the microcontroller programming).

At last, the STEP 3 is the agent programming using AgentSpeak. In this phase it is just necessary to program normal agents using Jason framework. The agent's plan should have external actions to perform actions into the simulated environment. The methodology do not provide how to connect electronic devices for neither the robots chassis nor sensors and actuators. For more information about how to use the devices and pre-defined files cited in this paper look for http://sourceforge.net/projects/javino/.

3. Controling a 4WD vehicle with Jason

In this section it is presented a simple example using both proposed platform and methodology to implement a vehicle robotic-agent. The vehicle is able to move forward until it finds an obstacle when it will change its directions turning to right, left or reward, depending if there is another obstacle or not. For this, it was chosen the 4WD vehicle and an ultrasonic distance sensor. The 4WD chassis is equipped with four 7.2V motors encoders. By means of exemplification and to fully follow methodology, was chosen to connect the vehicle and the sensor at Arduino board using the Raspberry only for embed the software agent.

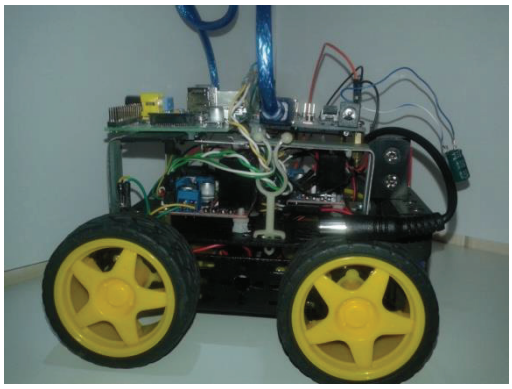


Figure 4. The vehicle robotic-agent chassis.

Following the methodology, in STEP 1 was selected the pre-defined file for the 4WD vehicle and the communication file for the ultrasonic distance sensor (both available at http://sourceforge.net/projects/javino/.). The pre-defined files are ready to use and have the Javino library setted. The pre-defined 4WD file provides movements functions (move, reverse, left, and right).

After that, the environment has to be prepared for the agent's external actions. First of all, it is mandatory to import Javino library into Jason's environment. The Pi4J library is not necessary in this example because it was chosen to connect the hardwares in Arduino board. All movements functions present in 4WD file must be programmed (if the programmer desires to use all possible movements). It was chosen to use the request-send method to update the agent's belief base: the agent requests to update its beliefs using an external action. Finally, the agent is programmed using Jason and AgentSpeak. The environment programming and the agent code can be seen in figure 5.

```
if (action.getFunctor().equals("front")) {
    logger.info(agName + ": sent front to Arduino.");
    javino.sendmsg("front");
}

if (action.getFunctor().equals("refresh")) {
    logger.info(agName + ": refreshing sensors.");
    javino.sendmsg("refresh");
    if (javino.availablemsg()) {
        addPercept(Literal.parseLiteral(javino.getmsg()));
    }
}

if (action.getFunctor().equals("turnLeft")) {
    javino.sendmsg("turnLeft");
    logger.info(agName + ": sent TurnLeft to Arduino.");
}
```

```
!start.

+!start: connected <-
    !move.

-!start: not connected <-
    .print("Waiting for hardware response...").

+!move : not obstacle <-
    .print("Go ahead.");
    front;
    !refreshSensor;
    !move.

+!move : obstacle <-
    .print("Obstacle ahead... turning left!");
    turnLeft;
    !move.

+!refreshSensor <-
    refresh.
```

Figure 5. The environment and agent implementation.

Now, the Raspberry Pi has to be configured to start with the embedded software (the project's jar file). It was realized two basic experiments: using a room without obstacles; and a room with three obstacles. The examples could show that the platform

works as expected, providing a reasoning for a hardware trying to reduce the programmer interventions. Some problems could be identified such as the agent executing time to send messages while it was still waiting data from sensors. The serial buffer could loss the information if the agent sends another serial command to the hardware. For now, this problem can be solved by both programming: *wait* action in Jason or *Thread.Sleep* in Java.

4. Related works

In this section it is discussed the proposed platform and methodology in comparison with the platform for grounded-vehicles [Barros et al., 2014] and the aquatic robotic using and BeagleBoard [Calce et al., 2013]. The platform for grounded-vehicles proposes an integration between Jason framework and the Arduino that uses RxTx library for communication between Arduino board and Jason's environment. Although the platform can operate a ground-vehicle and be adaptable for any kind of vehicles, it uses transmitters and receivers for each side (Jason and Arduino) that could cause data loss, interferences and depends on an external processing station (e.g. computer). The proposed platform truly embed Jason into a hardware without using external devices or transmitters/receivers for communication since it uses Raspberry, which provides an operational system that can be directly connected to an Arduino board.

The methodology for the grounded-vehicle platform consists in four programming steps: the hardware selection; the firmware programming (for the sensors/actuator actions and transmitters/receivers actions); the environment preparation (using RxTx and Jason's external actions); and the agent programming. The proposed methodology reduces the programmer interference into the code, because it offers pre-defined files for some hardwares, that can be changed based on the hardware selection; it uses Javino which provides a ready-to-use communication protocol between Java and Arduino; and can use the Pi4J library, that directly controls the Raspberry digital pins (decreasing the methodology in one step). Basically, the programmer only has to worry with the agent and the simulated environment.

In [Calce et al., 2013], an aquatic robot (boat) using Arduino BeagleBoard is presented, where a robot can move from one point to another deviating from obstacles. The robot platform consists in connecting via USB port Arduino and BeagleBoard, which holds a standard robot middleware called ROS (for movements). The proposed robotic-platform uses Raspberry Pi, a lower price board with similar specifications. It also provides pre-defined files for Arduino (such as ROS), which communicates with a message protocol library that can be used to exchange messages between hardware and software. The aquatic robot does not use agent-reasoning, being only an action-reaction robot while the proposed methodology uses Jason framework.

5. Conclusion

This paper presented a robotic-agent platform that uses Arduino and Raspberry to automate hardware functions and Jason to provide intelligent reasoning. Besides, it also presented the Javino library which is a communication protocol to exchange messages between Java and Arduino using serial port. A simple example using 4WD chassis was implemented to evaluate both platform and methodology.

For future works it will be proposed an architecture and several internal actions

for Jason, in order to provide to agents directly control vehicles movements. It will reduce the programmer interference into code, since he will not need to program the simulated environment anymore (for vehicles movements). It is also possible to directly update the agent's belief base without any interference into environment. The platform will be extended to guarantee communication between two or more robotic-agents. Furthermore, more pre-defined files for other kind of vehicles and sensors/actuators will be developed too.

References

Barros R. S., Heringer V. H., Pantoja C. E., Lazarin N. M., and Moraes L. M. (2014) "An Agent-oriented Ground Vehicles Automation Using Jason Framework" In: Proceedings of 6th International Conference on Agents and Artificial Intelligence: volume 1, ICAART'14, Angers, France.

Bellifemine, F., Caire, G., and Greenwood, D. (2007). "Developing multi-agent systems with JADE". Wiley series in agent technology.

Boissier, O., Bordini, R. H., Hubner, J. F., Ricci, A. e Santi, A. (2011) "Multi-agent oriented programming with jacamo" Science of Computer Programming.

Bordini, R. H., Hubner, J. F. e Wooldridge, W. (2007) "Programming Multi-Agent Systems in AgentSpeak using Jason" Jonh Wiley and Sons, London.

Calce A., Forooshani P. M., Speers A., Watters K. ,Young T., and and Jenkin M. (2013) "Autonomous Aquatic Agents" In: Proceedings of 5th International Conference on Agents and Artificial Intelligence: volume 1, ICAART'13, Barcelona.

Gertz E. and Justo P. D. (2012). Environmental monitoring with Arduino. EUA: Maker Press, 2012.

Upton, E. and Halfacree, G. (2012). "Raspberry pi user guide". United Kingdom: John Wiley & Sons Ltd,.

Wooldridge, M. (2000). "Reasoning about rational agents". Intelligent robotics and autonomous agents, MIT Press.