Multi-Agent Systems Integration in Embedded Systems with Limited Resources to Perform Tasks of Coordination and Cooperation

Ángel Soriano, Leonardo Marín, Ángel Valera, Marina Vallés
Instituto Universitario de Automática e Informática Industrial, Universitat Politècnica de València, Camino de Vera s/n, 46022, Valencia (Spain)
ansovi@ai2.upv.es, leomarpa@upv.es, {giuprog, mvalles}@ai2.upv.es

Keywords: Multi-agent systems, embedded systems, robot cooperation, distributed architecture, systems with limited resources, mobile robot, robot control, intelligent systems.

Abstract: This article describes in detail the steps for the integration of embedded systems with limited resources within multi-agent systems. Platforms and procedures are described and the development of three applications that bring the benefits offered by these systems is accomplished. The developed applications demonstrate the ability of the multi-agent systems to solve problems that require a high degree of coordination and cooperation between agents also adding automatic trading strategies and deliver results including optimal resolution of missions. These experiments results also show the flexibility, scalability, robustness and efficiency in environments with very different characteristics and easily applicable to real environments such as industry, collaboration between mobile robots or animal behaviour. The robotic agents of the applications submitted in this paper, socialize among them, have intelligence and ability to perform work individually and collectively.

1 INTRODUCTION

The exposition and development of multi-agent systems integration in different research areas related to robotics or autonomous heterogeneous systems is a widespread issue in the area of mobile robots research. The area of artificial intelligence has made its way through different areas and has found very useful, particularly in the field of robotics. As it is well known, the main topic of AI is the concept of intelligent agent defined as an autonomous entity which observes through sensors and acts upon an environment using actuators (Russell, Norvig, 2009). This definition is very close to the services that a robot can provide, so the concept of agent often is related with robots, (Bruce, Cardelli, Pierce, 1997), (Van Leeuwen, 1995), (Michalewicz, 1996). The ability to use different types of collaborative robots and unify the diversity of architectures and communication protocols that implements each one, offers the advantage of being able to take advantage each specific functionalities of robots. Thus, the multi-agent system is proposed as a performance control layer and processing communications system located above and abstracted from different programming languages of different hardware platforms. Besides it offers independence and adaptability to unforeseen own these systems.

Nowadays, more types of small robots are increasingly coming to the market, which offer more limited resources than professional robots but are oriented as educational robotics and facilitate their acquisition economically.

There are many different multi-agent platforms, but they all need, by definition, a minimum level of computation and above, of connectivity; since one of the most important characteristics of these systems is the ability to communicate between agents to report on something, negotiate, ask, etc.

In this paper, the used robots provide slow communication protocol as Bluetooth 2.0 with a transfer rate of 100ms. In case the physical hardware could be integrated directly into the multi-agent system, these can slow down transfer times and make the system unstable when trying, for example, a negotiation between agents.

This paper describes the integration of multi-agent systems in embedded systems which alone are

not able to run the multi-agent platform because they have limited computing capacity and slow communication protocols used as little as possible to ensure their integration into the platform.

2 MULTI-AGENT SYSTEMS

A multi-agent system (MAS) is a set of autonomous agents able to work together to solve a problem (Ana Mas, 2005). These systems must meet certain conditions (Feber, 1999), MAS consists of an environment, a set of objects, a set of agents, a set of relationships that link objects and agents, a set of operations and the definition of operators that represent the employment of operations on the environment and its reaction when it is disturbed.

The functional unit of MAS is the agent. A large number of proposals for the definition of agent can be found in the literature but none has been fully accepted by the scientific community. One of the simplest was stated in (Russell, 1995), which considers an agent as a physical or abstract entity that perceives and acts on an environment. An agent must be able to evaluate such perceptions and decisions by simple or complex reasoning mechanisms, as well as act on the environment in which it operates. MAS can be compared to a group of people with different domains of knowledge, trying to solve a common problem.

Three important concepts are crucial to the integration of a set of agents of different types to form a MAS: communication, cooperation and coordination.

When a group of individual agents is part of MAS, the need for a mechanism to coordinate the group of agents and a language to allow communication between them, arises. There are two main mechanisms of coordination: cases in which agents have common goals and, therefore, cooperate, and cases in which the agents are competitive and have conflicting goals with others, where trading mechanisms are required (Michael N. Huhns, Anuj K. Malhotra, 1999), (Munindar P. Singh, Michael N. Huhns, 1999) . Among the most used negotiation mechanisms in the literature are the coalition formation, market mechanisms, bargaining theory, voting, auctions and tasks allocation between agents. More specifically, there are three main approaches to automated negotiation (Fatima S., Wooldridge M., Jennings N. R., 2001), (Rahwan I., Sonenberg L., Dignum F., 2004): techniques based on game theory, techniques based on heuristics and techniques based on argument.

Communication plays a very important role since the negotiations depend directly on its effectiveness. There are different agent communication languages (Austin, 1962), (Searle, John, 1969) like FIPA Agent Communication Language (FIPA-ACL) or Knowledge Query and Manipulation Language (KQML).

In the development of new methodologies for the design of multi-agent systems, researchers have focused their efforts on extending the existing ones, mainly on the areas of object-oriented methodologies and on engineering knowledge, (Iglesias, Garijo, Gonzalez, 1999). A MAS is inherently multithreaded since each agent has at least one thread of control, (Wooldridge, 2002). These characteristics make MAS appropriate for development operations in complex, dynamic and unpredictable, systems or environments.

The chosen development environment for programming multi-agent systems that are presented in this paper is Java Agent Development Framework (or JADE).

2.1 JADE – Java Agent Development Framework

JADE (http://jade.tilab.com) is a free software platform for agent development, under development since 2001 and fully implemented in Java. JADE supports the coordination of multiple agents according to FIPA specifications and provides a standard implementation of agent communication language FIPA-ACL. It was originally developed by Telecom Italia and is distributed as free software, being fully compatible with Java Development Kit (JDK) 1.4 or higher. It includes the functionality for agent basic creation, the programming of agents' behaviour, the implementation of FIPA-ACL specification for sending and receiving messages, classes useful for programming FIPA protocols, information management using ontologies, etc… Also, the platform provides FIPA (AMS, Facilitator and MTS directory) that enables the execution of one or more instances of the Java Virtual Machine (JVM). Each JVM is seen as an environment, where agents run concurrently and exchange messages, organizing containers.

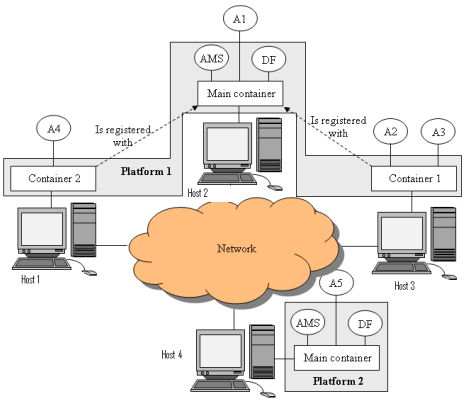

Figure 1: Example of JADE platform schema.

The functions that each of the agents should perform is within their behaviours. They represent a task that an agent can perform, and it is implemented as an object that extends the jade.core.behaviours class. Each agent has a stack of active behaviours, which in real time can be modified by an agent or by other behaviour. In addition, an agent can execute several behaviours concurrently; however, the planning of agent behaviours isn't preventive, but cooperative. This means that the developer is who decides when an agent changes the execution of behaviour to another.

Another feature offered by JADE is the directory service or yellow pages service. It allows agents to publish one or more services that they provide, so other agents can search for a particular service. The directory service in JADE (according to FIPA specification) is provided by an agent called DF (Directory Facilitator). Each platform has a FIPA DF agent by default, but more can be activated to provide a yellow page catalogue, distributed throughout the network. Although it is possible to interact with the DF agent by exchanging ACL messages like any other agent, JADE simplifies these interactions using the jade.domain.DFService class, through which it is possible to publish, edit and search for different services. Thus, agents must provide a template description of the services they offer to the DF agent, and keep that information updated at all times. Other agents may require a service provided by another agent, and the DF agent facilitate them the name of the agent (or agents) that offer that service.

One of the most important features that JADE agents possess is the ability to communicate. The adopted communication paradigm is the asynchronous message exchange. Each agent has a small stack of input messages, where the JADE runtime stores the messages sent by other agents. Also, each time a message is added to the queue, the receiving agent is notified. However, the instant at which the agent collects the message from the queue and processes it, depends only on the programmer.

An agent can get a message of its own message queue using the receive() method. This method returns the first message in the queue (and removes it) or returns null if the message queue is empty. Similarly, it has a blocking function (blockingReceive()) for receiving messages, which stops the behaviour until a message with the necessary characteristics is received.

JADE also incorporates the concept of ontology that simplifies the communication semantics. Ontology is based on two levels of content language: Lightweight Extensible Authentication Protocol (LEAP), unreadable by humans because it's byte-encoded, and String-encoded Language (SL), readable by humans because encodes the expressions like strings.

3 METHODOLOGICAL APPROACH

This section describes the integration of JADE in robotic systems with limited resources, the used platforms for experimental validation to problems based on missions and robot cooperation, the mobile robot kinematic control of these robots which includes the integration of data fusion filter locally within different platforms to avoid typical problems as imperfections in the location or positioning, and finally the general scheme of the control architecture developed.

3.1 Integration of JADE in Robotic Systems

A JADE agent can run on any system with JVM installed and JDK 1.4 or higher; however the integration of agents in embedded systems with limited resources is not trivial. Few embedded systems incorporate the ability to run JVM or JDK. Therefore, the idea developed to incorporate these robots to MAS lies in using agent-robot communication to make transparent the concept relation between a physical robot and a software agent, and thus obtain a Multi-agent Robot System (MARS).

MARS represent a complex distributed system, consisting of a large number of agents-robots cooperating to solve a common task. Each MARS

agent is an independent system which manages subsystems like tasks execution, perception of environment by sensors, trajectory control, robots communications, etc. Each agent of MARS represents a real physical mobile robot that communicates directly with its software agent that represents it in MARS. For a good synchronization in this relation, the communication time plays an important role; however, some embedded systems offers transfer rates about a hundred milliseconds roundtrip. This precludes, for example, the idea that the kinematic control is calculated on an external computer because communication delays may make the system unstable and unpredictable. For this reason, in this work each robot usually has its own local kinematic control.

3.2 Hardware Platform Description

Below the main hardware platforms used for validation experiments are detailed.

3.2.1 LEGO Mindstorms NXT

LEGO Mindstorms NXT was introduced by on the International Consumer Electronics Show in 2006 and nowadays is often used by the research community to prove theories and carry out practical developments. The control unit of NXT is an ARM7 microcontroller with 256 Kbytes FLASH, it has 64Kbytes of RAM, it allows communications by Bluetooth v2.0 and has a wide variety of sensors such as the ultrasonic sensors, gyroscopes, accelerometers or light sensors, which can be easily connected to the brick. Furthermore, the motors incorporate encoders with 360 degrees of resolution. There are several available firmwares to program the robot, but in this work, the firmware chosen is LeJOS (http://lejos.sourceforge.net) because it offers trigonometry and complex math functions, multithreading support and object-oriented programming in JAVA.

Figure 2: LEGO Mindstorms NXT in differential drive configuration.

A more detailed description of this system can be found in (http://mindstorms.lego.com). Figure 2 shows an example of a robot that can be mounted with LEGO NXT.

3.2.2 E-puck

Another robot often used by the research community for the validation experiments is the e-puck (see Figure 3). The e-puck is a mobile robot of differential wheels drive developed by Dr. Francesco Mondada and Michael Bonani in 2006 in the EPFL, (Federal Swiss Institute of Technology in Lausanne) (http://www.e-puck.org/) and it is an open hardware and software robot. It consists of two motorized wheels with the capacity to turn in opposed directions to change the robot's direction. In addition it has several sensors, of which the infrared sensors are very useful to measure the distance to near objects. Also the activation signals of the step motors are available and can be used instead of encoders (physically not available in the e-puck). The brain of e-puck is a dsPIC processor which works like a micro-controller. It is where the control routine of the robot is programmed and where its firmware is stored. It's distributed by the Microchip Company and it allows an efficient signals processing.

Figure 3: e-puck mobile robot.

3.3 Mobile robots kinematic control

All the robots used in this work were built using a differential drive configuration. It consists of a rigid body with two non-deformable non-orientable (fixed) wheels separated a distance $\boldsymbol{b}$ and moved by two motors that generate the left $\boldsymbol{v_L}$ and right $\boldsymbol{v_R}$ linear velocities as shown in Figure 4. The wheels are conventional and they satisfy the pure rolling without slipping condition (Gampion, Bastin, Dandrea-Novel, 1996). Also, the movement of the robot is restricted to a horizontal plane. The robot pose is defined by its position $\boldsymbol{P_0=(x,y)}$ with the

heading angle $\boldsymbol{\theta}$ in the Global reference frame ($\boldsymbol{X_G}$, $\boldsymbol{Y_G}$) in Figure 4. The kinematic relationship between the linear and angular velocities ($\boldsymbol{v}$ y $\boldsymbol{\omega}$) in the Local frame ($\boldsymbol{X_L}$, $\boldsymbol{Y_L}$) and the linear velocities in the wheels is shown in (1).

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 \\ -1/b & 1/b \end{bmatrix} \begin{bmatrix} v_L \\ v_R \end{bmatrix} \tag{1}$$

With $\boldsymbol{v}$ y $\boldsymbol{\omega}$, the global velocities are defined in (2).

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{2}$$

By discretizing and recursively integrating (2) with sample time $\mathbf{T_s}$, the robot global pose is obtained in (3).

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} v_{k-1} T_s \cos\left(\theta_{k-1} + 0.5 T_s \omega_{k-1}\right) \\ v_{k-1} T_s \sin\left(\theta_{k-1} + 0.5 T_s \omega_{k-1}\right) \\ T_s \omega_{k-1} \end{bmatrix} \tag{3}$$

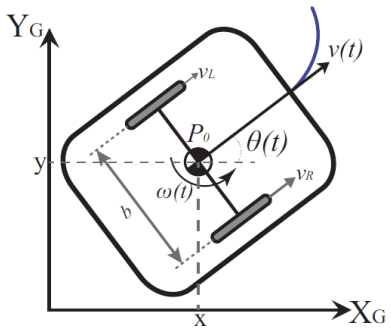

Figure 4: Kinematics differentially driven wheeled robot.

Using the wheels encoders to obtain ($\boldsymbol{v_L}$, $\boldsymbol{v_R}$) as inputs of (1) and then ($\boldsymbol{v}$, $\boldsymbol{\omega}$) in (3) the robot pose can be obtained. This is used for the e-puck, but for the LEGO NXT, the sensor fusion can be used to obtain the pose with improved accuracy.

The sensor fusion provides an efficient computational tool to estimate the state of a process, in a way that minimizes the mean squared state estimation error. In this work, the sensor fusion is based-on Kalman Filter (KF) algorithms (Welch, Bishop, 2007). The theory, algorithms and design principles of the KF is well known and is widely described in many books and articles such as (Bozic, 1994), (Grewal, Andrews, 2001). There are several variations of the filter tailored to different applications, but is the linear KF the one used to perform the fusion

The KF is used to improve the estimation of the state $\boldsymbol{x_k}=[\boldsymbol{v},\ \boldsymbol{\omega}]^{\mathbf{T}}$ by using the encoders and, in LEGO's cases, a gyroscope as measurements. But first, an experimental model is identified using several step inputs in the control action $\boldsymbol{u}$ (motor voltage) while recording $\boldsymbol{v_L}$ y $\boldsymbol{v_R}$. Using these samples in the Matlab® "System Identification Toolbox" the experimental model in (4) is obtained.

$$v_{L,R} = \frac{0.1276}{0.1235s + 1} u \tag{4}$$

Using (4) in (1) as the model for the KF (discretizing with T_s=50ms) and (5) as the measurement equation $\boldsymbol{z_k}$, the sensor fusion is performed. With this, $\boldsymbol{x_k}$ is substituted in (3) to improve the final pose estimation.

$$z_k = C x_{k-1}, \qquad c = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \tag{5}$$

Another important aspect of kinematic control of mobile robots is the path following algorithm used. For this work the navigation algorithm chosen is a pure pursuit controller. In this, a predefined trajectory is stored in the robot memory as a set of global points, independent from time, and based in the geometric description of the desired path (for example a square, a circle, etc.). The robot will follow one point after another starting from the closest one to its current position. Assuming a desired constant linear velocity of the robot $\boldsymbol{V}$, the wheels reference velocities are defined in (6).

$$v_R = V\cdot\left(1 + \frac{b}{2}\gamma\right) \qquad v_L = V\cdot\left(1 - \frac{b}{2}\gamma\right) \tag{6}$$

The constant $\boldsymbol{\gamma}$ is obtained form (7).

$$\gamma = \frac{-2\left(\left(x_{ob} - x_r\right)\cos\theta + \left(y_{ob} - y_r\right)\sin\theta\right)}{\left(\left(x_{ob} - x_r\right)^2 + \left(y_{ob} - y_r\right)^2\right)} \tag{7}$$

To obtain $\boldsymbol{\gamma}$, the robot pose is used along with the nearest point to the trajectory adding the look ahead distance $\boldsymbol{L}$ to obtain ($\boldsymbol{x_{ob}}$,$\boldsymbol{y_{ob}}$). This distance keeps the robot from reaching the objective point, making the robot follow the desired path continuously.

The fusion KF performance using the pure pursuit algorithm is shown in Figure 5. In this, the robot is set to follow a path while recording the real trajectory using a zenithal camera. From this figure, it is clear that the pose estimated using only the encoders differ from the real one. But when the

robot uses the sensor fusion with the KF, the estimated pose is more accurate as it is closer to the one measured with the camera.

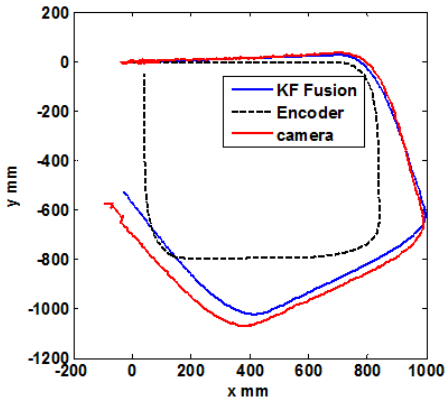

Figure 5: Pose estimation improvement using the KF sensor fusion in LEGO NXT, path followed using pure pursuit.

Below is the scheme of the control architecture that's used for practical experiments developed.

3.4 Control Architecture for the Practical Experiments

Different architectures to coordinate the movement of several robots can be developed using wireless communication. Figure 6 shows the basic scheme of the control architecture implemented.

Robots are connected to their software agents (computers) via Bluetooth and those computers are part of a network that forms the overall MAS through JADE. In the scheme a camera is often used to monitoring the experiments or, in some cases, providing some global information to the system.

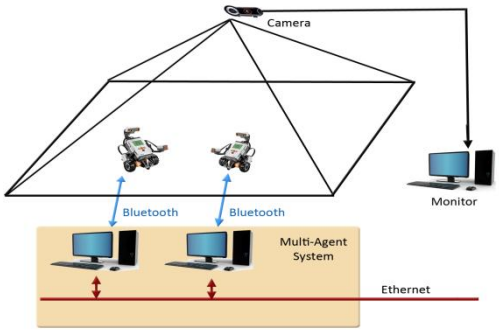

Figure 6: Architecture for the practical experiments.

4 PRACTICAL EXPERIMENTS

This section presents three applications developed with different robots based on a distributed architecture using MAS.

4.1 Hierarchical Adaptive Control

This experiment aims to recreate the behaviour of birds when they move in flocks. There are 6 e-pucks randomly moving around the stage. At some point, a hierarchy that determines for each robot who must pursue is established, thus changing its destination to trying to reach the leader robot. The robots are grouping together and avoiding potential collisions that may occur while adjusting its speed to keep the group. If a robot isn't assigned any leader, it continues to move without a fixed destination. This is the case of the robots which are on top of the hierarchy scheme. To implement this application, an overhead camera captures the scene and positions the robots through triangle-shaped decoys of different colors, placed above them. All e-pucks are connected via Bluetooth to their respective software representatives in a computer network that forms the MAS following the scheme described in Section 3.4.

In this application, the camera is acting as an agent within the MAS, and through communication between agents, each robot is able to know the positions of the rest and even correct its pose. Locally, every e-puck has three main modules in their behaviours. One is responsible for communication with the software agent representing it in the MAS, another handles the collision avoidance reactively if it detects an obstacle near (Braitenberg, 1991), and another performs the control to try to reach the target position that is has been assigned. When, at some point, the hierarchy of the formation that robots must follow is established, the agent initiates a cyclical behaviour where each cycle time locates all the robots on stage and sends, to each robot agent, the target position that has to be reached according to the leader robot that has assigned. Thus, the robots are grouped according to a hierarchy that can be changed anytime as desired, they adjust its speed and they avoid collisions between them, creating the feeling of being part of a group of birds that move in formation flight. Figure 7 shows an example of the execution of the application. A video can see at http://wks.gii.upv.es/cobami/webfm_send/10.

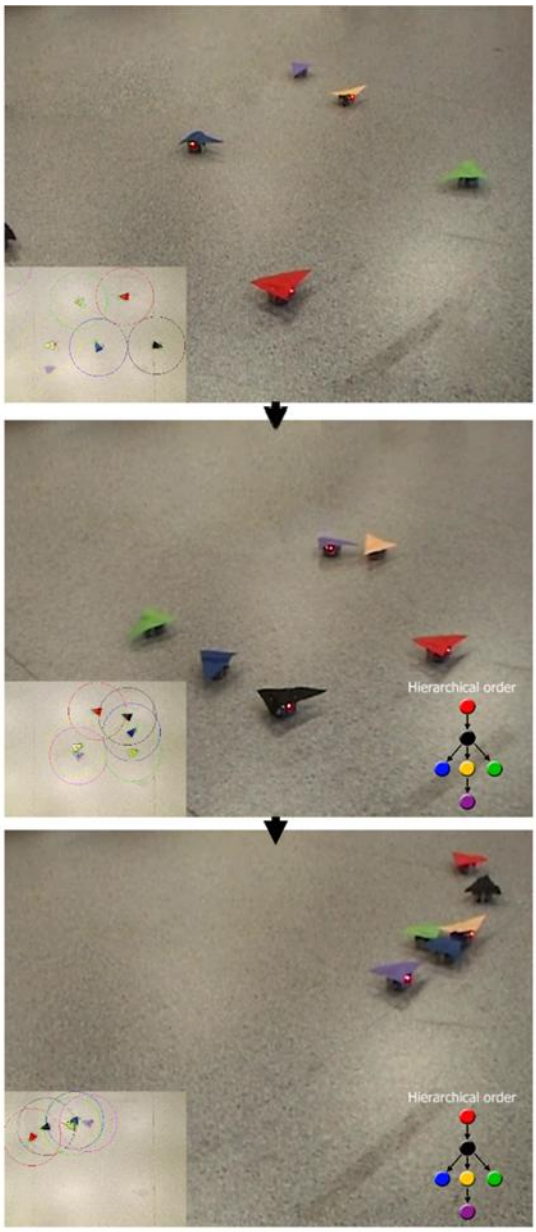

Figure 7: example of the execution of the application in three phases: initial phase where each robot moves randomly, the second phase which provides the relevant the hierarchy and the third when robots are grouped.

4.2 Mission-Based Control

This application is developed from a group of LEGO Mindstorms robots which offer different features to perform missions that require efficient organization and cooperation of robots. From a general point of view, the application includes features to explore, manipulate, and finally collect items of interest. Extrapolation of these roles may enable the development of applications such as the management of a crop in the field, waste collection, detection and demining on land, etc.

Three models of robots are involved in the implementation, each characterized by its ability to act and sense the environment. First the robot scout, provided with sensors that detect the items of interest. Second the robot manipulator, who simulates the interaction with the item of interest. Finally, the robot collector, equipped with a linear actuator which has been added, a magnet that allows collecting the items of interest, which have metallic material to facilitate their collection.

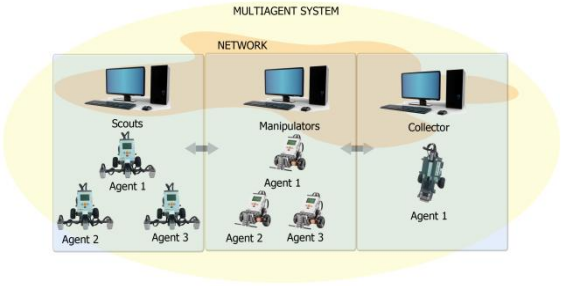

Figure 8: Architecture developed for the application.

For this application three levels of hierarchy or, more specifically, three holarchies have been defined. One formed by three scout robots, another by three manipulator robots and another by a single collector robot. Each robot is connected to a computer as shown in Figure 8. The scheme is very similar to that described in section 3.4.

The application procedure can be divided into two levels. In a first-level, scout agents robots run through the stage in search of the elements of interest. Each time that an item of interest is detected, they obtain the position of it in the shared coordinate plane by all robots and communicate with some manipulator agent robot who will act on the element. Here begins the first performance optimization strategy. With several manipulators, the explorer who found the object, communicates with the rest to obtain their positions and status (they may be available, busy acting on some object or simply unavailable) and according to the information received, decides to whom it assigns that item. This automated trading has been implemented using mainly heuristic based strategies combined with the occupancy estimated time when the manipulators are acting. The optimization strategy is defined by minimizing the time required for each candidate to reach from its position to the position of the element of interest. To carry out such reasoning the scout agent estimates the best candidate in terms of the factors that make the equation (8),

$$t = V\sqrt{\left(x_c - x_e\right)^2 + \left(y_c - y_e\right)^2} + T_0 + Busy\left(\left(T_0 - T_e\right) + T_m N_e\right) \quad (8)$$

where V is the average speed at which the candidate robot moves, x_c, y_c are its coordinates, x_e, y_e are the coordinates of the element of interest, T_0 is the average time it takes to perform the candidate, *Busy* is a boolean which defines the status of the candidate (if busy one, zero if not), T_e corresponds with the time that has already taken on the item that is acting now, T_m is the average time it takes to reach and act on an object and N_e is the number of objects that the candidate has already assigned to manipulate. This reasoning is fully dynamic and cannot be deterministic from the viewpoint of candidates, because the allocation performed by scout agents can be reorganized by the manipulators agents through compensation algorithms workload or because of some unforeseen situation that requires that an agent has to delegate its assignments to other manipulator.

At the second level, the manipulators should contact the collector agent and provide the coordinates of the elements of interest that it has to collect. The dynamism of the system is reflected in the collection order of the objects, which also responds to a strategy based on heuristics, where the priority of collection depends directly on the distance between the element of interest and collecting agent.

Figure 9: Picture of the beginning of actual application.

Figure 9 shows a picture of actual scenario and a video of the application is available at http://wks.gii.upv.es/sidireli/webfm_send/227.

4.3 Emulation of an Industrial Environment

A recreation of an industrial environment through embedded systems with limited resources has been developed in order to demonstrate the usefulness of MAS in such environments.

On stage, a conveyor belt runs until a piece which carries, comes at the end of the conveyor and activates a sensor. At that time the conveyor belt stops and a robotic arm picks up the piece for depositing it on a point called store. Between the store and the beginning of the same conveyor belt, a forklift transports the pieces from one point to another, respectively, such that the pieces are located again on the conveyor belt and the cycle repeats. There are three pieces moving on stage but the store only has the capacity to stock one of them, so when a piece is waiting in store to be picked up, until the forklift does not take it, the robotic arm cannot leave another. Also between the store and the conveyor belt there is a semaphore. If the light is green, when the forklift arrives at the traffic light can continue, but if the light is red, it must stop.

In this case, the forklift mobile robot, the conveyor belt, the semaphore and the robotic arm, are the agents of this application. The specific scheme for this application is shown in Figure 10.

The conveyor and the presence sensor are controlled by a LEGO Mindstorms brick that is connected via Bluetooth to a computer running software agent representative. Similarly the robotic forklift built with LEGO pieces, and the semaphore, which is just one more NXT brick, join the MAS. The robotic arm is controlled via serial commands to computer where the corresponding software agent executes.

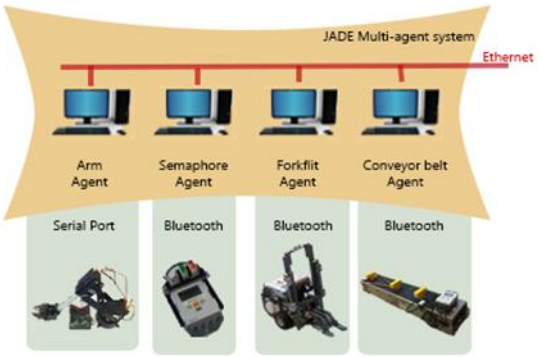

Figure 10: Architecture developed for the industrial environment application.

Unlike the sequential programming typically used in industrial environments, the operation in this application is completely distributed. Each agent executes its own behaviours that define the actions that can be performed at any time. MAS can solve, by communication between agents, some critical actions that require collaboration and cooperation among agents.

One important fact is that only if there is a piece on the conveyor belt waiting to be picked up, the robot arm must go to pick it up. The arm agent executes a cyclical behaviour which at all times is waiting for a message to report on this. Whenever the sensor of the conveyor detects a piece, the conveyor agent informs the arm agent that it has a prepared piece and, only then, it can go and pick it up.

Another critical point for the arm is that, before placing the piece in the store, it must first ensure that the forklift is not near to avoid colliding with it. It also needs to know if, when it is going to leave the piece, there is still a piece in the store waiting to be picked up by the forklift. This is solved by communication between arm and forklift agents. Before arm leaves the piece in the store, the arm agent must ask permission to forklift agent to perform the action. The forklift agent may deny permission because it has not gone to pick up the last piece yet, or because it is within the zone of danger of collision with the arm. In any case, when the store can only hold one piece and the forklift is out of danger, the arm can perform the action of leaving the piece in the store without any danger.

An additional critical point is the semaphore. The semaphore obliges the forklift agent to ask permission to cross every time they meet. The response of the semaphore can be changed in real time pressing the buttons of the brick. This makes the whole system could be locked at this point. (The conveyor belt waiting for the arm to pick up a piece, arm awaiting confirmation from the forklift to leave a piece in the store and the forklift blocked because the semaphore isn't allowed to continue). Recovery of normal system operation when the light turns green is implicit in the application deployment.

This application succeeds in demonstrating the flexibility, efficiency and robustness of MAS in this type of environment, from its recreation through platforms with limited resources. Figure 11 shows a picture of actual scenario and a demonstration video can be viewed http://wks.gii.upv.es/cobami/webfm_send/8.

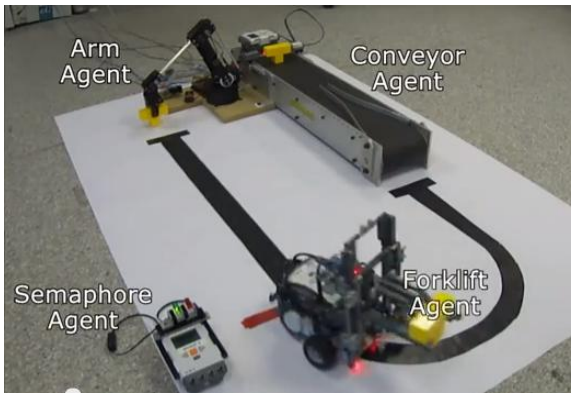

Figure 11: Picture of actual industrial environment application.

5 CONCLUSIONS

The area of research in robotics moves increasingly toward the development of cooperative robots groups. This implies that new ways to solve problems should be proposed; the old approaches of individual agents do not work for this. Access to a group of high-end commercial robots to experiment with them is sometimes complicated because the economic costs. Nowadays, educational robots offer very similar characteristics to the high-end robots, but with severely limited resources. This article has described in detail the steps for the integration of embedded systems with limited resources within multi-agent systems. Platforms and procedures have been described and the development of three applications that bring the benefits offered by these systems has been accomplished. The developed applications demonstrate the ability of the MAS to solve problems that require a high degree of coordination and cooperation between agents also adding automatic trading strategies and deliver results including optimal resolution of missions. These experiments results also show the flexibility, robustness and efficiency in environments with very different characteristics and easily applicable to real environments such as industry, collaboration between mobile robots or animal behaviour. The robotic agents of the applications submitted, socialize among them, have intelligence and ability to perform work individually and collectively.

There is no doubt that over time there will be new ways to understand how agents should behave in a group or a holarchy, and new strategies for reconciliation of theories of MAS.

ACKNOWLEDGEMENTS

This work has been partially funded by the Ministerio de Ciencia e Innovación (Spain) under research projects DPI2011-28507-C02-01 and DPI2010-20814-C02-02.

REFERENCES

Austin, J. L., 1962. *How to Do ThingsWith Words*. Oxford University Press: Oxford, England.

Braitenberg V, 1991. *Véhicules: expériences en psychologie synthétique*, 171, Presses Polytechniques Romandes, Lausanne, Switzerland.

Bruce, K.B., Cardelli, L., Pierce, B.C., 1997. *Comparing Object Encodings. Theoretical Aspects of Computer Software*. Lecture Notes in Computer Science, volume 1281. Springer-Verlag, Berlin Heidelberg New York.

Bozic S., 1994. *Digital and Kalman filtering: an introduction to discrete-time filtering and optimum linear estimation*. Halsted Press.

Campion G., Bastin G., Dandrea-Novel B., 1996. *Structural properties and classification of kinematic and dynamic models of wheeled mobile robots*. Robotics and Automation, IEEE Transactions on, 12(1):47–62.

e-puck educational robot web page. http://www.e-puck.org/

Fatima, S., Wooldridge, M., Jennings, N. R. 2001. *Optimal negotiation strategies for agents with incomplete information*. In Intelligent Agent series VIII: Proceedings of the 8th International Workshop on Agent Theories, Architectures, and Languages (ATAL-2001), volume 2333 of lecture Notes in Computer Science, pages 53-68. Springer Verlag, Berlin, Germany.

Feber J., 1999. *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence.* Addison Wesley, London.

Grewal M.S., Andrews A. P., 2001. *Kalman Filtering: Theory and Practice Using Matlab.* John Wiley & Sons.

Iglesias C., Garijo M, Gonzales J. A. 1999. *Survey of Agent-Oriented Methodologies*. 1999.

Java Agent Development Framework. http://jade.tilab.com

LEGO Mindstorms home page. http://mindstorms.lego.com.

LeJOS: Java for LEGO Mindstorms. http://lejos.sourceforge.net.

MAS, A., 2005. *Agentes Software y Sistemas Multiagente: conceptos, arquitectura y aplicaciones.* Pearson-Prentice Hall.

Michael N. Huhns, Anuj K. Malhotra, 1999. *Negotiating for Goods and Services.*IEEE Internet Computing, vol. 3, no. 4, pp. 97-99.

Michalewicz, Z., 1996. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, Berlin Heidelberg New York.

Munindar P. Singh, Michael N. Huhns, *Multiagent Systems for Workflow*. International Journal of Intelligent Systems in Accounting, Finance and Management, Volume 8, John Wiley & Sons, Ltd., pp. 105-117.

Rahwan, I., Sonenberg, L. Dignum, F. 2004. *On interest-based negotiation*. Advances in Agent Communication Workshop held in conjunction with AAMAS-03, volumen 2922 de Lecture Notes in Artificial Intelligence, Berlin: Springer-Verlag, pp. 383-197.

Russell, S., 1995. *Rationality and Intelligence. Artificial Intelligence*, 94, 57–77.

Russell, S.J., Norvig, P., 2009. *Artificial Intelligence: A modern approach*. Prentice Hall Series in Artificial Intelligence, Upper Saddle River, New Jersey.

Searle, John, 1969. *Speech acts: An essay in the philosophy of language.* Cambridge, England: Cambridge University.

Van Leeuwen, J. (ed.), 1995. *Computer Science Today. Recent Trends and Developments*. Lecture Notes in Computer Science, volume 1000. Springer-Verlag, Berlin Heidelberg New York.

Welch G., Bishop G., 2007. *An Introduction to the Kalman Filter*, University of North Carolina at Chapel Hill, http://www.cs.unc.edu/~welch/kalman

Wooldridge M., 2002. *An Introduction to MultiAgent Systems*. John Wiley & Sons: Chicester, UK.