# Learning Control Architectures by Robotics Application

R. MONTUFAR-CHAVEZNAVA, V. DE LA CUEVA HERNANDEZ AND M. ALI YOUSUF
GIRATE Group, Engineering Division
ITESM Santa Fe
Av. Carlos Lazo 100, Col. Santa Fe, Del. Álvaro Obregón, México D. F.
MEXICO

*Abstract:* - Engineering students starting their studies in engineering find it difficult to understand and appreciate the value of undergraduate education. Our group has experimented with various robotic projects for diverse engineering students designed specifically to improve their learning and understanding of the subjects taught. In particular, we are interested in teaching the basics of control architectures, and robotics has given us the clue to get this objective. The wide variety of projects, their quality and originality gives us confidence that these students can become far more fruitful engineers if we employ the right strategy of education. In this work we present some developed projects related to control architectures applied to robotics, the results obtained and the gained experiences.

*Key-Words:* -Control Architectures, Robotics, Control Applications.

## 1 Introduction

Actually, the control of many systems is carried out by control architectures. We can find in literature diverse architectures such as behavior based, reactive, deliberative or multiagent architectures. In this work we present some instances to motivate the learning of control architectures by a specific robotic application. The instances we present were developed by diverse undergraduate students.

At first, considering robot teleoperation on Internet is a complex problem, we have proposed and developed a **multiagent control architecture**, where information streams, including video data and voice commands, transmitted on Internet, are processed and managed using intelligent agents to successfully execute specific robotic tasks.

The second application deals with the design and construction of an educational indoor cleaner robot named Crabot. Crabot is a Lego based autonomous mobile robot, which represents the prototype of a real robot for house cleaning, which requires a minimum of intelligence. Crabot is provided with necessary items for floor cleaning such as a small vacuum cleaner and a brush. The control system is based on the **subsumption architecture** with memory and offers a repertoire of behaviors for navigation and cleaning. This kind of control system allows mixing real-time distributed control with behaviors triggered by robot sensors. The processing is carried out in a pair of Lego RCX microcomputers. Crabot was tested in an artificial world and in competition where its performance was very acceptable.

Finally we present a control architecture based on **membrane systems** emulating a behavior based architecture. The idea is to exploit the parallel computing characteristic of membranes in a specific robotic application: house cleaning. We try to map behaviors to membranes to build the proposed architecture. The developed architecture was tested in an educational Lego robot: Crabot M.

## 2 Control Architectures Essentials

From robotics perspective, control architecture is the interaction between sensors, processors and actuators to produce an intelligent behavior according the world where the robot interacts. The importance of control architecture relies on the fact that mobile robots are systems with a high degree of complexity, and then, if the system is more complex, the organization of the components is more important. A mobile robot is also considered as a system where the inputs are sensorial data and the outputs are all possible commands to actuators. In brief, the architecture is the interface between sensors and actuators.

The architecture determines the robot behavior when interacting with the world. In this sense, the architecture selects the stimuli to accept, search or produce; it also determines how to select the most appropriate action to execute according the perception. Briefly, the architecture must have the capacity for selective perception and adequate action.

Traditionally, there are three robotic architectures:

*Deliberative, Reactive* and *Hybrid.* There is an additional class inspired on ethology: *Behavior Based*.

The different responses produced by the architectures allow establishing comparisons evaluating each approach, attending criteria such as: robustness, reactivity, finality and scalability.

Besides, we can find new control architectures proposals, which can fit perfectly in any of the three architectures mentioned above, such as the membrane architecture presented in this work.

Finally, the multiagent architecture, employed in one of the developed projects presented in this work, corresponds to a Belief-Desire-Intention (BDI) architecture, which is one of the most successful agent architecture. The properties of this architecture resemble a behavior based architecture where agents can be considered behaviors.

Next, we briefly describe the principal control architectures for robots previously mentioned.

### 2.1. Deliberative Architectures

Architectures traditionally considered deliberative are heiress to classic artificial intelligence research. They emerge first in the world of mobile robotics generating behavior, and being dominant until the end of 80's. They produce behavior executing a plan, and reasoning on certain symbolic model of the world. The robot intelligence in locate in a module named the planner, which deliberates on the world model symbols and produces an explicit actuation plan. The plan is a sequence of actions to execute to pursuit the goal.

The epistemological origin of deliberative approach is found in the Cartesian philosophy that considers the soul as the being, which decides the behavior [1]. It sinks its roots in the cognitive and represents a complete theory of the intelligent operation. Naturally, the artificial intelligence will look for approbation, testing the capacity to generate intelligent behavior.

The principal characteristics of these architectures are the necessity of planning and the modeling of the world, the use of hierarchy and the execution of the traditional cycle sense-model-plan-act.

### 2.2. Reactive Architectures

At the end of 80's, the lack of flexibility in robots controlled by deliberative architectures was notable. Perhaps it was the reason many researchers began to think again how to generate behavior, and the use of plans, which was the masterpiece of the deliberative paradigm [2, 3, 4]. For instance, Firby speaks about the necessity to monitor the plan execution continually because there are events that cannot be anticipated due to failures when acting, or to the dynamic changes of the world. As result of this rethinking, the reactive approach was born. The reactive architecture, guided by the works of Rodney Brooks [5, 6] caused an important turn in the way to produce behavior

The new approach points at the necessity of a most direct link between sensors and actuators, avoiding intermediate layers employed by deliberative robots. In this way, events reaction is faster. The world model and management of symbols were considered unnecessary. In this way the reactive approach is sub symbolic, and argues that it is not necessary the symbolic representation or the reasoning of symbols to produce behavior. According to Murphy [7], this approach reduces the essential primitives to sense and act, which are directly linked, and left besides the planning.

If epistemologically the deliberative paradigm is related to the classic cognitive and the introspection, then, the reactive approach is related to conducts, and specifically to connectionism. In fact, the fundamental hypothesis is that behavior can be produced as an amalgam of reflex, which connects sensorial data to actuator values.

### 2.3. Behavior Based Architectures

Behavior based systems (BBS) emerge inside the reactive approach, and the most representative models are the works of Rodney Brooks [5]. Similar to Mataric [8], the reactive aspects have been separated from distribution and emergency behavior aspects. The principal BBS's characteristics are the control distribution and the perception in many units operating in parallel. This decomposition contrasts with the functional decomposition and the monolithic control, typical of deliberative architectures.

The underlying idea is that in a system complex behavior is an emergent property, which arises from the interactions between the basic components and the environment. Each component is a rapid cycle, reactive, from sensors to actuators, which have their own objective. In general, each unitary component implements the adequate reactions according the stimuli, accessing directly to necessary sensorial data and emitting control to actuators. Then, this paradigm also proposes the perception distribution. In this sense, this architecture does not need a central representation of the world or symbolic models to produce behavior.

### 2.4. Hybrid Architectures

As a consequence of the limitations observed in the reactive and deliberative approaches [6, 9] many hybrid approaches emerged, pretending to mix the advantages of both architectures as a unique architecture. Actually, hybrid architectures are the most employed. It is difficult to find an autonomous robot, which does not have some reactive components and certain degree of deliberation. In fact, there is an extended consensus in the fact that the reactive part is the right way to perform low-level control, perhaps supported by its practical success.

The differences inside the hybrid architectures are principally the way to introduce the deliberative part without fissures and breaking the reactive operation. In this way, there are hierarchic systems that employ symbolic deliberation as principal motor, but they allow the activation of some basic reactive routines as primitive actions.

This descendent essence appears in concrete architectures as Saphira [10] or TCA [11]. Other hybrid approaches have a marked ascendant essence as RAP by Firby [12], where small actuation units are built in the reactive part, introducing re-planning, or the DAMN architecture [13] that introduces some deliberative behaviors in a reactive system.

The principal characteristics of the deliberative systems (objective oriented, planning, tasks decomposition in subtasks and modeling) and those offered by the behavior-based architectures (reactivity, world oriented) are recommended for a real robot. A robot in the real world needs to have both capacities, it needs to be reactive to work in higher dynamic worlds but they also need to be able to carry out complex tasks that may need certain planning.

The behavior (sense, planning and act) in hybrid approaches is organized as two interrelated blocks. The first one is in charge of planning, and interacts with the second one, which constantly executes associations between sense and act. Then, the organization will be: planning, sense-act.

## 3 Multiagent System Essentials

The **agent** concept is important in artificial intelligence and computer science. An agent is considered a hardware or software based computer system that posses autonomy, social ability, reactivity and proactiveness. Furthermore, an agent is envisioned or implemented using concepts that are more usually applied to human beings [14]. Mobility, veracity, benevolence and rationality are other particularities eventually discussed in the context of agency.

We can define intelligent agents as systems that execute an unsupervised labor and apply some degree of intelligence to perform their job. The intelligence can be almost insignificant but it includes some measure of learning from past experiences. In this way the agent can be trained to be more successful in the future. Some intelligent agents can interact with another. At present, intelligent agents are used to model simple rational behaviors in distributed applications.

Multiagent system (MAS) is defined as a loosely coupled network of agents that interact to solve problems that are beyond the individual capabilities or knowledge of each agent [15].

The principal characteristics of MAS are they have a limited vision due the possession of a deficient information or ability; they do not have a global system control; their information is distributed; and their computation is asynchronous. Alternatively, MAS presents other characteristics, such as the ability to solve problems considered enormous for a sole agent or allow interconnection and interoperation between multiple agents.

MAS follows the lineaments of agent architecture. Agent architecture is considered a particular methodology for building agents, which encompasses techniques and algorithms that support this methodology [16].

As mentioned in previus section, the principal kinds of control architectures are deliberative, reactive, hybrid and behavior based architectures. However, one of the most successful agent architecture is the deliberative Belief-Desire-Intention (BDI) architecture [17].

The BDI architecture is based on the correspondence between beliefs, desires and intentions with reciprocal identifiable data structures. The identification of beliefs, desires and intentions is beneficial when the system must communicate with humans or other agents and it is desired the simplification in construction, maintenance and verification of application systems.

BDI architecture consists of three dynamic data structures representing the beliefs, desires and intentions of agents, combined with an input queue of events [18]. This architecture allows updating and querying operations on the three data structures, where update operations are subject to compatibility requirements.

BDI architecture captures the theory of practical reasoning, where intentions play significant and distinct roles in practical reasoning and cannot be reduced to beliefs and desires [19]. The BDI architecture is based on a closed set of beliefs, desires and intentions and the required probability procedures are not computable.

## 4 Membrane Systems Essentials

Membrane computing was introduced as an answer to the metaphor: “A living cell is a computer and the processes taking place are computations” [20]. The principal characteristic of living cells is the complex compartmentation by means of a membrane structure where multisets of chemical substances are processed (evolve) according to some prescribed rules.

A membrane structure is a hierarchical organization of membranes contained into a skin membrane, where the skin confines the system from the environment. A membrane labels a region, considered the space between the membrane and, if any exists, the inner membranes. An elementary membrane does not contain membranes inside.

We can allocate sets or multisets of objects (data, information) into regions of a membrane structure. A multiset is a simple set containing multiplicities of the objects; it means every object can appear in a number of identical copies in a given region. Fig. 1 shows the basic membrane structure.

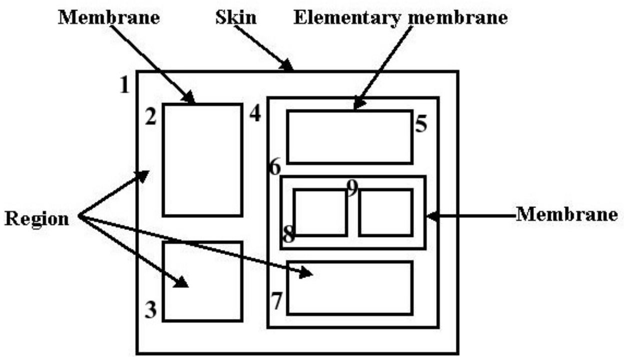

**Fig. 1.** The membrane structure. We can appreciate the skin (the main membrane) confining the system, some membranes, elementary membranes and regions. We observe that each membrane is labeled; this is useful when we design the architecture following a hierarchical representation.

The objects evolve according to some rules, which are associated to the regions. The rules specify the object transformations and the transferences between regions. The rules are employed in a non-deterministic maximally parallel mode, it means, the rules to be applied to objects are selected in a non-deterministic mode. Sometimes, the rules preference is considered to provide priority between rules.

The membrane structure and the objects define the configuration. The use of rules defines the transitions between configurations. A sequence of transitions is considered a computation.

The membrane systems consider that halting computations are successful; it means we have reached a configuration where no further rule can be applied. A successful computation is associated to the result.

Finally, we can say that membrane systems can be used to solve problems by means of generation, computation and decision, such as robotic architectures do.

## 5 Application Projects

We have designed several projects to teach control architectures. These architectures are directly applied to specific robotics tasks. The decision to employ robotics projects to teach control is due to a general observation: the materialization of the abstraction present in different theoretical topics is a sort of discovery for the students. Simply, we motivate the basic student curiosity.

Some of the designed projects are described below.

### 5.1. Telerobotics: Multiagent Architecture

Students have designed an architecture composed of five agents, as shown in Fig. 2. The modularity of MAS allows to the architecture increases their capabilities adding more agents for new tasks as necessary. Modularity provides flexibility to the system and that is the principal reason the MAS architecture was chosen. In addition to scalability, system restructuring is also possible in an easy way.

The system was developed in Java, and it is portable to almost any operating system with minimal changes.

The user interacts with the robot using a Web browser to connect to a Web server installed on the robot computer. All communication between agents is performed using a communication layer. Next we describe the developed agents:

***Receptor agent:*** This agent receives commands from the servlet, then classify and transmit them to the corresponding agent. The possible commands are *connect to simulator*, *connect to robot*, *go*, *stop*, *increase speed*, *turn left* and *turn right*. The agent also obtains data information from the robot, such as speed or position. It also can send commands to start or stop the video transmission.

***Video agent:*** This agent captures video from the webcam located on the robot and establishes the client communication to send the video. It responds to commands from the receptor agent.

***Interpreter agent:*** The interpreter agent establishes communication to the client to receive oral commands using RTP (Real Time Protocol). It recognizes and interprets speech, and then sends messages to controller agent for robot movement.

***Controller agent:*** It executes all received commands from interpreter or receptor agents. It is incharge of robot locomotion. It is continuously monitoring sensors to detect approaching objects and avoid collisions or dangerous situations.

The students also have included a ***servlet agent*** in the architecture. A servlet is a program that runs on a Web server, acting as a middle layer between a requests coming from an HTTP client and databases or applications on the HTTP server. In this case, the servlet mediates between the user interface and the receptor agent.

## 5.2. Crabot: Subsumption Architecture

In this section we describe the architecture developed for Crabot, a subsumption based architecture with memory; and some additional aspects as behaviors, communication and hardware. The subsumption architecture is a particular behavior based architecture.

***Control Architecture.***

The architecture was programmed in NQC, which is a reduced C language specific for the RCX (named bricks). Students employ a pair of bricks, one for cleaning and other for navigation. As mentioned above, the developed architecture is the subsumption with memory, based on the approach of Brooks [6]. We developed a set of actuation modules to sense relevant stimuli and produce a set of observable behaviors.

The subsumption with memory architecture offers a strategy to combine real time distributed control with behaviors triggered by sensors. The principal objective of this architecture is decomposing tasks in serial modules or simple behaviors.

The modules operate in asynchrony and are arranged as layers in a behaviors hierarchy: the upper level module corresponds to the robot objective, which is reached by executing all mediators in the scheme. The modules link perception and action and this correspondence represents a kind of behavior in the sense that each module increases the robot spectrum of abilities.

The modules are a collection of augmented finite state machines (AFSM). An AFSM is an automaton with timer; an input signal that comes from sensors or other AFSM; and a signal output that is sent to robot actuators or other AFSM. An AFSM can inhibit the input of another, connecting its output to the registry of the input (in this case, the message in the registry is substituted by the suppressor message); an AFSM can also suppress the output of another, connecting its output to the output of the other depending on the perceived sensor data. These mechanisms are the essence of the resolution of conflicts and establish the priority of the behaviors. An ASFM cannot share states and, in particular, cannot read the registries of another.

In this architecture, we establish one by one the relationships between behaviors (inputs, outputs, suppression and inhibition) to obtain the desired robot behavior. Fig. 2 shows the architecture and connections between sensors, modules and actuators.

The arbitration between behaviors is carried out by inhibition and suppression mechanisms. These mechanisms do not consider the fusion of commands, it means, it is not possible to activate two behaviors at the same time.

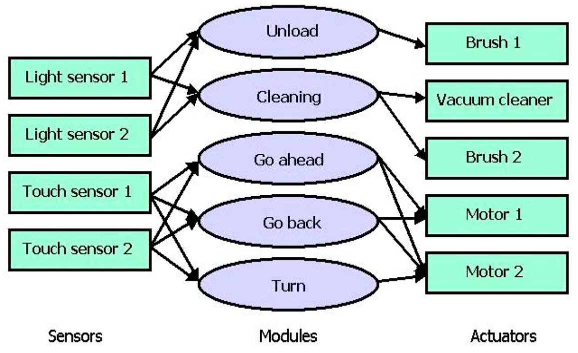

**Fig. 3.** The architecture control for Crabot: Relationships between inputs (sensors), behaviors and outputs (actuators).

*Memory:* The memory is an enhancement to subsumption. It is connected to the robot behaviors and provides memory to the robot. The memory of Crabot is a register to indicate when the garbage was sensed. If the garbage was not sensed, the Crabot executes the cleaning behavior again. The memory inclusion in the architecture is shown in Fig. 4.

### Deliberative Architectures

*1) Cleaning.* It is activated when Crabot senses garbage in the environment according the information provided by any of two light sensors. The cleaning brick performs cleaning. This behavior inhibits the flow of other behaviors when it sense garbage in the world. If the light sensor does not detects the garbage, it stops the inhibition to behaviors situated in the lower levels and the robot executes another behavior, following the rules dictated by the state. At programming level, when the cleaning brick senses garbage, it turns on the brush, the vacuum cleaner and sends the message *SendMessage(1)* to the navigation brick, which execute a sequence to go ahead and back repeatedly to clean.

*2) Unload.* It is employed to unload the garbage. The information provided by the couple of light sensors in the cleaning brick activates this behavior and inhibits the flow of other commands. When the sensors detect the trash area, the cleaning brick opens the container and sends the message *SendMessage(2),* and then navigation brick stops. After the task was completed, the inhibition to other behaviors concludes.

*3) Go ahead.* It is activated following the information provided by touch and light sensors. The navigation brick executes this behavior. “Go ahead” suppress the outputs of other behaviors when the path is free of obstacles, there are no dirty places, or the robot is not located in the trash area.

*4) Go back and turn.* It is activated by the navigation brick and depends on the information provided by the touch sensors. The robot turns every time it finds an obstacle and continues moving ahead until other behavior is activated.

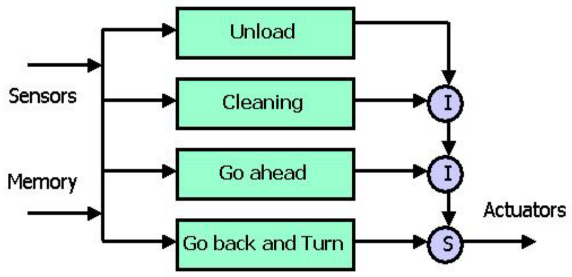

**Fig. 4.** The subsumption architecture with memory used in Crabot. We defined four basic behaviors for cleaning and navigation: *Unload*, *Cleaning*, *Go ahead* and *Go back and Turn*.

### Communication

Messaging from cleaning to navigation bricks carries out communication between bricks. The messages are sent using the infrared port of RCX.

There is a different message for every activated behavior. When the cleaning brick senses garbage, it sends the message *SendMessage(1)*, associated to the behavior *Cleaning*. If the trash area is detected, the message *SendMessage(2)* is sent, which is related to the behavior *Unload*. If none of the mentioned situations is detected, the message *SendMessage(0)* is sent. This message means *Go ahead*, and Crabot is dedicated exclusively to wander.

### Hardware

Crabot was built with a couple of Lego Mindstorms kits. These kits include two kinds of sensors: touch and light sensors. We can detect obstacles using the touch sensors. We can know if an object was detected by reading the collision status in the sensors. We can detect the light intensity or a color using the light sensors. These sensors deliver values from 0 (white) to 256 (black).

The dimensions of “Crabot” are a base of 25×30 cm$^2$ and a high of 25 cm. It weights 1.5 kg. The robot base is a differential model with two wheels controlled by independent motors at front and two fixed wheels at back for stability. Crabot has two touch sensors and two light sensors at front. Light sensor point out to the floor to detect garbage or the trash area. A couple of motors control the wheels, a third motor is employed to clean the garbage container and a fourth motor activates the brush situated at front of Crabot.

The garbage container was adapted as a slipper system to unload the garbage. The container has also a linear brush for cleaning and protection of the gages. We adapt a small vacuum cleaner at back of Crabot. The vacuum cleaner is turn on and off automatically. Crabot is shown in Fig. 5.

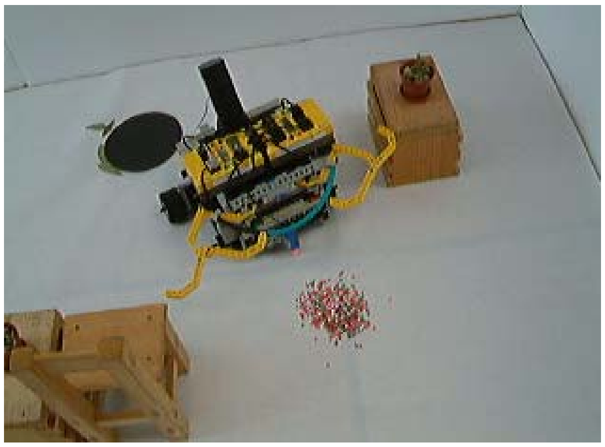

**Fig. 5.** The cleaning system and container of Crabot.

## 5.3. Crabot M: Membrane Architecture

The membrane architecture was programmed in the core of an educational robot based on the Lego Mindstorm kit: Crabot M. The architecture and the robot were configured for cleaning purposes. It was necessary to add some extra sensors and actuators suitable for our purposes. The robot was configured with a couple of bricks, one for actuators control and other for sensor readings and basic functions. The bricks are able to process up to ten tasks in parallel and can be programmed in specific Java or C languages. The use of C allows the use of threads, providing the capability of carrying out parallel processing.

In the case of the control architecture, students carefully carry out the definition of rules and the communication between membranes to preserve the schema states and arbitration. The proposed architecture is composed of the skin and eight membranes, which are defined as follows:

***Sensors membrane:*** It is essential in the architecture because it is in charge of getting all information from the world. This membrane reads data from sensors and processes it to send messages to other membranes.

***Light sensor membrane****:* It reads a value between 0 and 100 and passes it to the upper membrane. The value is got from a photo resistor and its magnitude depends on the zone where the robot is located or the kind of object detected under the body of the robot.

***Touch sensor membrane*****:** This membrane manages a boolean value, provided by a touch sensor located in the front of the robot. The value indicates when robot has reached the end of the world or has collided with an object.

***Task actuators membrane*****:** It carries out the control of the actuators corresponding to the brush, dustpan and dustpan lift. The actions depend of the information that the sensors membrane provides.

***Dustpan lift membrane*****:** If certain zone, object or garbage is detected this membrane commands to the dustpan lift to go up or down. Physically, the dustpan lift is a structure where the dustpan and the brush are fixed. This membrane manages two bits for three possible states: up, down and stop.

***Dustpan membrane:*** This membrane controls the dustpan and also receives from the upper membrane a two bits word for the same states.

***Brush membrane****:* It controls the brush through a two bits word. The commands to send are spin up, spin down and stop.

***Motion actuators membrane:*** This membrane controls the robot navigation. It uses a three bits word for five states or actions, such actions are: go ahead, go back, turn left, turn right and stop. The action is selected depending whether the robot is wandering or it has detected an object or a particular zone.

The membrane architecture is presented in Fig. 6, where we can observe the communications between some of the membranes.

Crabot M is presented in Fig. 7, where we can observe their actuators, sensors, processors and tools for cleaning tasks.

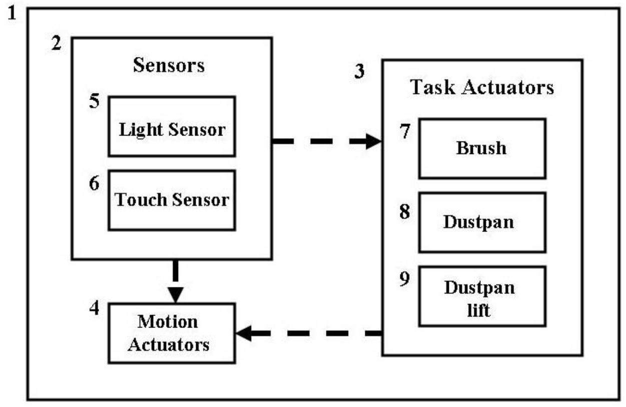

**Fig. 6.** The membrane architecture for robot control. The robot is designed for cleaning tasks and the membranes are implemented following the rules of the objectives. The architecture has a skin and eight inner membranes.

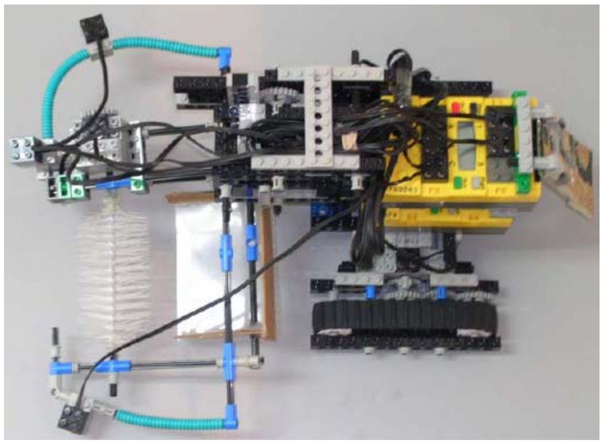

**Fig. 7.** Crabot M. It has two bricks where the control architecture resides. We note the dustpan lift structure where the brush and the dustpan are fixed. Some

arrangements were made to make sure the cleaning task was successfully carried out.

Crabot M was used in a competition for house cleaning. Initially, it wanders around the world, a white square room, where there are several kinds of garbage, some objects to avoid and garbage dumps. During wandering the sensors and motion actuators membranes are working. If some garbage or a garbage dump is detected an interchange of information is produced activating some membranes and halting or keeping in stand by others.

Rules and goals were determined in the architecture. The system is started and when it completely halts, we know that the goals have been reached. In this case, it means that the world has been cleaned and the garbage has been put in the corresponding dump.

## 6 Conclusions

In this work we have presented three projects developed for students. The idea behind each project is to materialize the control architecture theory through the development of them.

We note that students appreciate at best the knowledge when it has a practical application. We also observe the challenge in a competition is a motivational factor for learning, and that is the reason we look for different events where the results of the students work be presented and in competition with other students work.

The results obtained from the projects presented in this work are excellent. The objectives proposed for each project were reached in time and form. In fact, a set of articles were accepted and presented in international forums [21, 22, 23].

Considering the results we have obtained [24], we will continue proposing different projects, not only for advanced studies as control, but also for basic courses such as mathematics or physics, where the students usually are very unmotivated and do not have any interest in them.

***References:***

[1] McFarland, D. and Bosser, T. *Intelligent behavior in animals and robots.* The MIT Press, 1993. ISBN-0-262-13293-1.

[2] Payton, D. W. "Internalized plans: a representation for action resources". *Robotics and Autonomous Systems*, 6:89–103, 1990.

[3] Agre, P. E. and Chapman, D. "What are plans for?" In *Designing Autonomous Agents: theory and practice from Biology to Engineering and Back*, Pattie Maes, editor, pp. 17–34. MIT Press, 1990.

[4] Firby, R. J. "An investigation into reactive planning in complex domains". In *Proceedings of the 6th AAAI National Conference on Artificial Intelligence*, pp. 202–206, Seattle, WA, 1987.

[5] Brooks, R. A. "A robust layered control system for a mobile robot". *IEEE Journal of Robotics and Automation*, 2(1):14–23, March 1986.

[6] Brooks, R. A. "Intelligence without reason." In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 569–595, 1991.

[7] Murphy, R. R. "Dempster-shafer theory for sensor fusion in autonomous mobile robots". *IEEE Transactions on Robotics and Automation*, 14(2):197–206, April 1998.

[8] Mataric, M. J. "Behavior-based control: main properties and implications". In *Proceedings of the IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems*, pp. 46–54, Nice (France), May 1992.

[9] Brooks, R. A. "A hardware retargetable distributed layered architecture for mobile robot control". In *Proceedings of the 1987 International Conference on Robotics and Automation*, pp. 106–110, Raleigh-NC, March 1987. Computer Society Press. 645–649a, Perth (WA, Australia), 1999.

[10] Konolige, K. et al., "The Saphira Architecture: A Design for Autonomy," *J. of Experimental and Theoretical AI*, vol. 9, no. 1, pp. 215-235, 1997.

[11] Simmons, R. G. "Structured control for autonomous robots". *IEEE Journal of Robotics and Automation*, 10(1):34–43, February 1994.

[12] R. James Firby. "Building symbolic primitives with continuous control routines". In *Proceedings of the 1st International Conference on AI Planning Systems* AIPS'92, pp. 62–69, College Park, MD (USA), June 1992.

[13] Julio K. Rosenblatt and David W. Pyton. "A fine-grained alternative to the subsumption architecture for mobile robot control". In *Proceedings of IEEE International Joint Conference on Neural Networks*, vol. 2, pp. 317–323, Washington D.C., June 1989.

[14] Wooldridge, M. and Jennings, N. R. "Intelligent Agents: Theory and Practice". *The Knowledge Engineering Review. 10 (2).* 1995. pp. 115-152.

[15] Sycara, K. P. "Multiagent Systems". *Artificial Intelligence Magazine. 10 (2).* 1998. pp. 79-93.

[16] Maes, P. "The Agent Network Architecture (ANA)". *SIGART Bulletin, 2 (4)*, 1991. pp. 115-120.

[17] Busetta, P., Ronnquist, R., Hodgson, A. and Lucas, "A. JACK Intelligent Agents – Components for Intelligent Agents in Java.", *AgentLink Newsletter 2.* January 1999.

[18] Rao, A. S. and Georgeff, M. P. "BDI Agents: From Theory to Practice". *Technical Note 56. Australian Artificial Intelligence Institute*. Victoria, Australia. 1995.

[19] Bratman, M. E. *Intentions, Plans and Practical Reason*. Harvard University Press, Cambridge, MA, USA. 1987.

[20] Păun, Gh. "Computing with Membranes", *Journal of Computer and System Sciences*, Vol. 61:1, pp.108-143, August 2000.

[21] Montúfar-Chaveznava, R. and Fernández, Y. "Crabot: Educational Robot Prototype for Cleaning." *Proceeedings of 12th Portuguese Conference on Artificial Intelligence (EPIA 2005)*: Covilha, Portugal, December 5-8 2005, pp. 266-271.

[22] Montufar-Chaveznava, R and Méndez-Polanco, J. A. "Multiagent Architecture for Telerobotics", *Proc. Of 15th International Conference on Electronics, Communications and Computers - CONIELECOMP 2005*: Puebla, México. February 28 to March 2nd 2005, pp. 149-153.

[23] Montufar-Chaveznava, R. and Méndez-Polanco, J. A. "Multiagent Technology for Web Robot Teleoperation," in CD of the *Seminario Anual de Automática, Electrónica Industrial e Instrumentación 2004*: Toulouse, France. September 15-17 2004.

[24] M. Ali Yousuf, R. Montúfar Chaveznava, and V. de la Cueva Hernández. "Robotic Projects to Enhance Student Participation, Motivation and Learning". Accepted in *IV International Conference on Multimedia and ICTs in Education* (m-ICTE2006), Sevilla (Spain), 22-25 November 2006.