This document is published in:

Robotics and Autonomous Systems (2012), 60 (12), 1607–1624.
DOI: 10.1016/j.robot.2012.09.019

A behaviour-based control architecture for heterogeneous modular, multi-configurable, chained micro-robots

A. Brunete [1], M. Hernando [2], E. Gambao [2], J.E. Torres [2]

[1] *Universidad Carlos III de Madrid, Av. Universidad, 30, 28911 Leganés (Madrid), Spain*

[2] *Robotics and Cybernetics Group, Centre for Automation and Robotics (CAR), UPM-CSIC, José Gutiérrez Abascal 2, 28006 Madrid, Spain*

E-mail: abrunete@ing.uc3m.es, alberto.brunete@gmail.com; (A. Brunete), miguel.hernando@upm.es (M. Hernando), joseemilio.torres@upm.es (E. Gambao), ernesto.gambao@upm.es (J.E. Torres).

Abstract: This article presents a new control architecture designed for heterogeneous modular, multi-configurable, chained micro-robots. This architecture attempts to fill the gap that exists in heterogeneous modular robotics research, in which little work has been conducted compared to that in homogeneous modular robotics studies. The architecture proposes a three-layer structure with a behaviour-based, low-level embedded layer, a half-deliberative half-behaviour-based high layer for the central control, and a heterogeneous middle layer acting as a bridge between these two layers. This middle layer is very important because it allows the central control to treat all modules in the same manner, facilitating the control of the robot. A communication protocol and a module description language were also developed for the control architecture to facilitate communication and information flow between the heterogeneous modules and the central control. Owing to the heterogeneous behaviour of the architecture, the system can automatically reconfigure its actions to adapt to unpredicted events (such as actuator failure). Several behaviours (at low and high levels) are also presented here.

Keywords: *Modular, Multi-configurable, Heterogeneous, Behaviour-based, Control Architecture*

1. Introduction

The control architecture described in this article was designed for chained, modular micro-robots. These micro-robots are composed of different types of drive module (heterogeneous modules) that can be arranged in different configurations, a feature called multi-configurability. Thus, a robot can be manually assembled in different configurations depending on the chosen task. Because this architecture was designed for the robot MICROTUB [1], it will be called MICROTUB architecture.

Heterogeneous modular robots are robots composed of different types of module and are called *n*-modular [2], where *n* is the number of different modules. Conversely, homogeneous modular robots are robots composed of one single type of module. "Chain" means that the modules are connected in a row, as opposed to "lattice" robots in which modules can be connected in a lattice.

Most modular robot designs are homogeneous [3–8], at least in a locomotive sense. Polypod [9] and I-Cubes [10] are composed of two types of module, one of which is passive and mainly functions to carry the power supply. Molecules [11,12] feature two different modules that perform the same type of movement. Thus, there is a lack of heterogeneous drive modules. Consequently, there is no clear state of the art regarding heterogeneous chained, modular robots. The closest robot to the state of the art relies on homogeneous modular robot architectures.

Among homogeneous modular robot architectures, three of them have been found to be of special interest: CONRO [13,5], PolyBot [14,15] and M-TRAN [16,17]. All of these feature control algorithms for distinct locomotion modes. CONRO presents the concept of hormones, which are special messages that can trigger different actions in different modules.

M-TRAN presents a distributed control and a three-layer architecture with a low-level middle layer for communication and high-level control (used for reconfigurations). PolyBot presents the attribute/service model, which is specially designed for complex tasks that require communication between different modules.

Chain-type robots have several features of bio-inspired robots that also suggests the incorporation of biologically based behaviours into their architecture. Behaviour-based architectures are specifically appropriate for the design and control of semi-autonomous artificial robots based on biological systems (because biological models often serve as the basis for the design of behaviour-based robotic systems [18]), for modular robots and for systems integrating both low- and high-level control [19].

Regarding behaviour-based architectures, apart from the well-known Motor Schemas [20], Activation Networks [21] or DAMN [22], it is necessary to mention CAMPOUT [23,24], which is a very interesting architecture because it integrates different types of behaviour (e.g., primitive, composite, communication and coordination) and different arbitration mechanisms (e.g., priority-based, state-based, voting and fuzzy). It is a clear example of the great flexibility that enables the use of behaviour-based architectures. Recent publications like [25,26] (presenting the iB2C architecture) show the relevance of behaviour-based control architectures.

The architecture proposed in this paper attempts to combine some of the previously presented concepts with some new features, thus providing a control solution for chained, modular robots composed of different types of module (heterogeneous modules). These new features include a middle layer similar to that of M-TRAN's, but with some heterogeneous capabilities that permit the triggering of similar actions in different modules (as opposed to CONRO's hormones), a communications model that allows heterogeneous modules to communicate amongst themselves and to communicate its capabilities, and a set of behaviours to cover low- and high-control layers.

The MICROTUB architecture is mainly based on behaviours and is divided into three layers: a low-control layer that is embedded in the modules and makes decisions for the modules, a high-control layer that makes decisions that concern the entire robot, and a heterogeneous middle layer that acts as an interpreter between the central control and the modules. The heterogeneous layer is particularly important because it allows the central control (CC) to treat all modules in the same manner, thus facilitating the control of the modules.

Because it is not desirable to reprogram each module every time a new configuration or a new task is chosen, the control architecture provides a mechanism for the CC and the modules to know the current configuration of the micro-robot and behave according to this configuration. Thus, the robot can be manually assembled into different configurations (depending on the task), and it will be able to perform the most appropriate movements and coordinate sensors and actuators. Because of this control architecture, the robot is controlled as a whole, and it can receive simple and complex commands (e.g., go, stop, turn and explore) and execute them irrespective of its configuration. There is no need to send specific commands to each module every time. Every module can perform individual actions and react in a different manner to the same commands.

Two features of the architecture are of special relevance: the module description language (MDL) and the offline control. MDL has been developed to describe the capabilities of the modules (both the movements that they can perform and sensors that they may have). Because of MDL, each module can report to the CC what it is able to do (their capabilities, e.g., rotate, push forwards, measure temperature and measure distance), and the CC can set up actions for the entire robot.

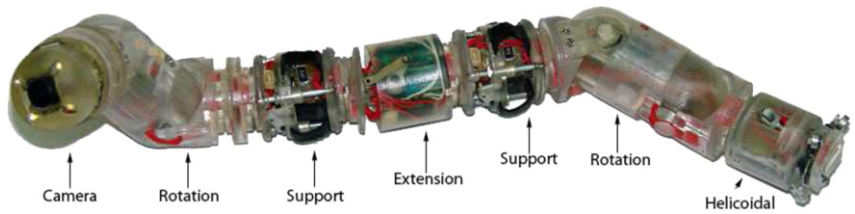
Offline control permits the development of rules and movement patterns based on the optimisation of the movements of the heterogeneous configurations of the robot while running in a simulator that is specifically designed for the type of robot and has been previously validated with real data.

This architecture has been tested in the robot MICROTUB (Fig. 1), a chained, modular micro-robot designed for pipe and small cavity exploration, in which the following modules have been developed: rotation, support, extension, helicoidal, and camera/contact [1]. These modules allow for different movements such as snake-like, worm-like, helicoidal-driven and any combination of these. The MICROTUB robot is described in Section 2.

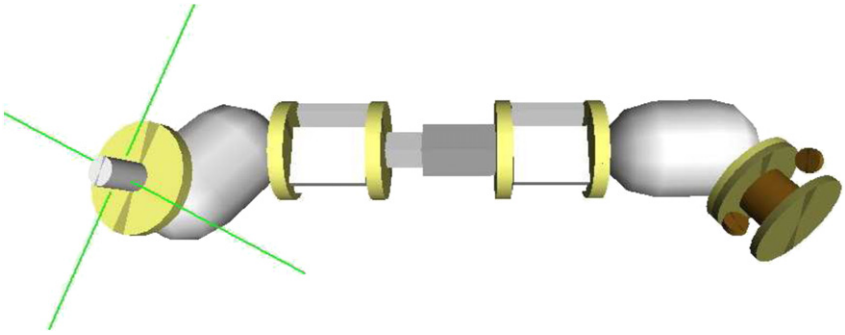
Section 3 provides an overall description of the architecture, and the three layers of the architecture will be explained in Sections 4–6. Section 7 is dedicated to the offline control and the simulation. Finally, in Section 8, sample applications of the architecture in different situations are provided.

2. Robot description

MICROTUB is a semi-autonomous multi-configurable micro-robot for small-diameter pipe inspection and maintenance. It has been designed to explore pipes with a camera to detect breakages, holes, leaks and any type of defect [27]. This micro-robot is composed of different modules, each of which performs a different task. Thus, multi-configurability is an essential characteristic that allows these modules to be easily interchanged depending on the task, without the need for reprogramming the micro-robot.



(a) Real.



(b) Simulated.

Fig. 1. Micro-robotic modules.

The different types of module developed (Fig. 1) will be briefly described next, but more information can be found in [28,1]. The diameter of each module is 27 mm. The thickness of some parts is less than 1 mm. Fig. 2 shows a chart comparing the dimensions of all of the modules.

The micro-robot is heterogeneous and modular, meaning that it is composed of different types of active (they are able to move) and passive (they have to be acted on) modules.

To assemble them together, a common interface has been built. This interface allows for the mechanical and electrical connection between modules. The electrical bus is composed of eight wires:

- Power (5v) and ground.
- I^2C communication: data and clock.
- Two synchronism lines (in and out) for low-level communication between adjacent modules (some examples of its use can be found in Sections 4.1.6 and 5.2).
- Two auxiliary lines for general purposes (for example, to transmit the video signal from the camera).

Each module includes an electronic control board that performs the following tasks: control actuators, communication via I^2C or with adjacent modules, manage several types of sensor, auto-protection and adaptable motion, self-orientation detection, and low-level embedded control.

The different modules of the micro-robot have to be manually assembled at the beginning owing to the characteristics of the mechanical connectors. The use of electromechanical latches or magnets (as in [16,14]) for future modules so that they are able to attach and detach by themselves is being considered.

2.1. Camera/Contact module

This module plays two roles. First, as a camera, it is used to acquire information about the environment and detect holes, breakages or cracks in the pipes. Second, as a contact sensor, it is able to determine whether the micro-robot is facing an obstacle.

The module features a 320 × 240 pixels composite video CMOS B&W camera, which permits the visualisation of the inner part of a pipe, and three contact sensors, which permit the detection of obstacles.

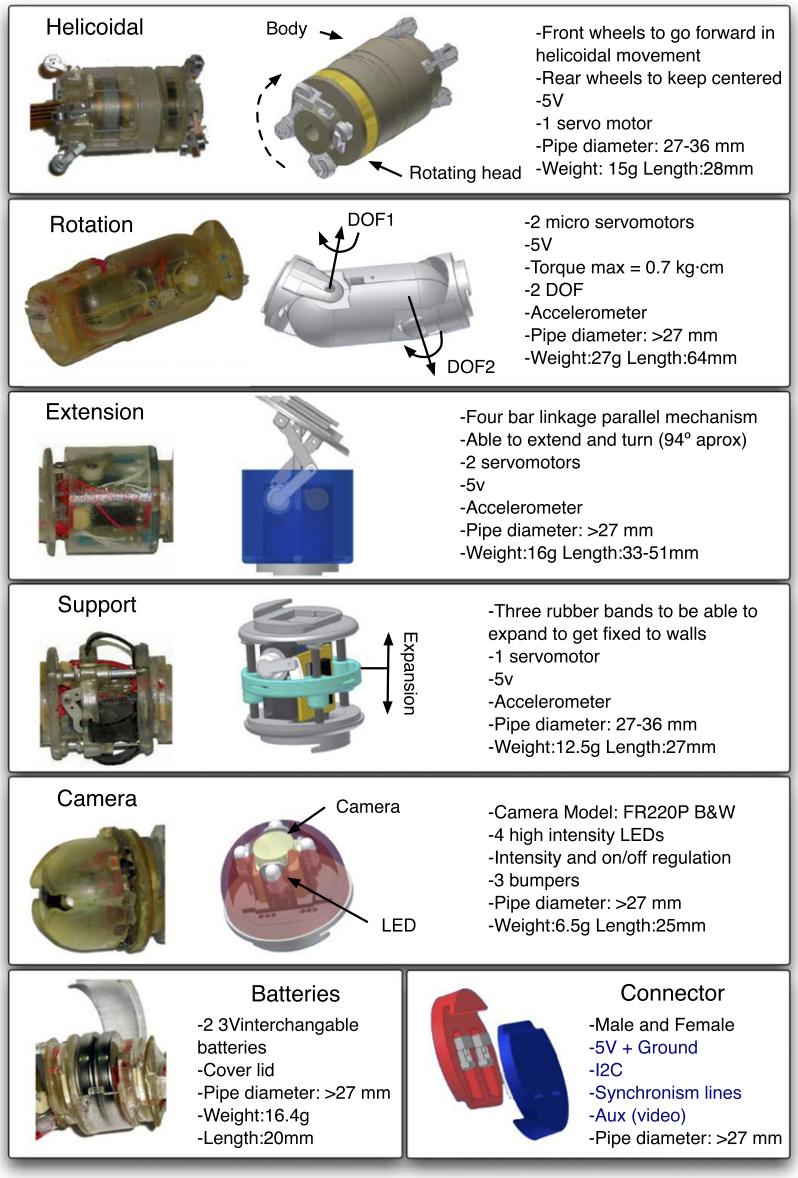


Fig. 2. Module description chart.

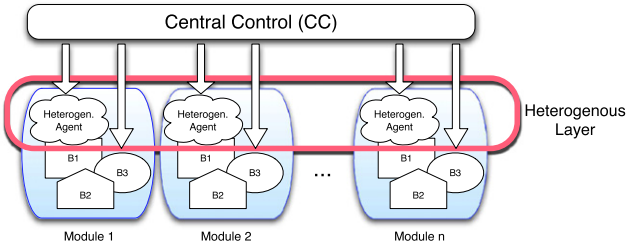


Fig. 4. Control layers.

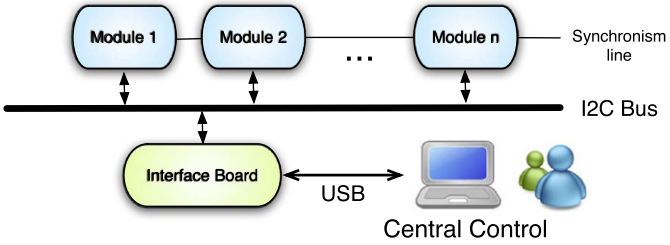


Fig. 3. Hardware architecture description.

2.2. Rotation module

The rotation module is a two-degrees-of-freedom (DOF) module that allows rotations in the horizontal and vertical planes. A combination of these modules can perform undulatory movement (snake-like) that makes the robot move forwards. It is composed of two commercial mini-servomotors.

2.3. Inchworm modules

Two modules have been developed to perform inchworm (or worm-like) movements: an extension module and a support module. The inchworm mode of locomotion allows the robot to manoeuvre in small spaces. Another advantage of this kind of motion is that the robot manages to maintain a firm grip on the surface at all times. The support module is used to fix the micro-robot to the pipe; thus, this module does not move. The extension module is used to extend the robot (make it go forwards) and to allow it to turn right and left.

2.4. Helicoidal module

The helicoidal module was designed to be a fast-drive module able to push other modules. It is composed of two parts: a body and a rotating head.

When the head turns, it goes forwards in a helicoidal movement (helped by the distribution of the wheels that form a 15° angle with the vertical) that pulls the body of the micro-robot forwards. The wheels of the body help to keep the module centred in the pipe and prevent the body from turning.

2.5. Other modules

Some other modules that have been designed (but not built) are the traveller module, the sensor module and the battery module. The traveller module is used to compute the travelled distance by using several wheels featuring encoders and an appropriate algorithm. It has been specially designed for pipes. At any time, at least one of the wheels is in contact with the pipe. The algorithm computes the measurements from all wheels and outputs an estimated distance.

The sensor module is a passive module that includes different types of sensor, such as temperature and humidity sensors.

A battery module has also been developed, but it must be improved to provide the necessary power supply to drive several modules for a reasonable period of time. Currently, the micro-robot requires a cable connection for both the power supply and video transmission.

[5] This interpreter was first thought to be a behaviour, but for clarity, it was renamed as "interpreter".

3. Overview of the control architecture

3.1. Physical layout

The hardware architecture is shown in Fig. 3; the modules hold an embedded control board, and they are connected via the I^2C bus. A PC holding the CC is connected through an interface board to the I^2C bus. Modules are also connected to their neighbours through a wire (synchronism line).

3.2. Layer structure

For the proposed architecture, a semi-distributed control has been chosen. It has a behaviour-based central control planner that makes decisions for the entire robot and an embedded behaviour-based control in every module that is capable of reacting in real time to unpredicted events. There is also an interpreter acting between the CC and the embedded behaviours: the *heterogeneous agent*.[5] The heterogeneous agents of all modules form the heterogeneous layer, which is called a middle layer because it acts between the CC (highest-level layer) and the onboard control (lowest-level layer). Regarding the physical layout, control is divided in (Fig. 4) as follows:

- CC: It could be a PC or one of the modules. Currently, it is a PC, but it could be one of the modules to make the robot autonomous. It includes the following layers:
  - High-control Layer: controls the robot as a whole. It collects information from the modules, processes it, and sends information about the situation and state of the robot, as well as commands with objectives, to the modules. It also helps the modules to make and coordinate decisions. The high-control layer is also in charge of planning and is composed of several parts, including an inference engine and a behaviour-based control.
- Onboard Control: it is embedded in each module and is based on behaviours. It includes the following layers:
  - Heterogeneous (Middle) Layer: agent that translates commands coming from the CC into specific module commands. For example, it translates the command "extend" into servo-motor movements.
  - Low-control Layer: composed of behaviours. It allows modules to react in real time (for example, to sense external and internal stimuli, such as overheating and unreachable positions, and adapt to the pipe shape) and to perform tasks that do not require the CC (e.g., movements, communication with adjacent modules and simple tasks).

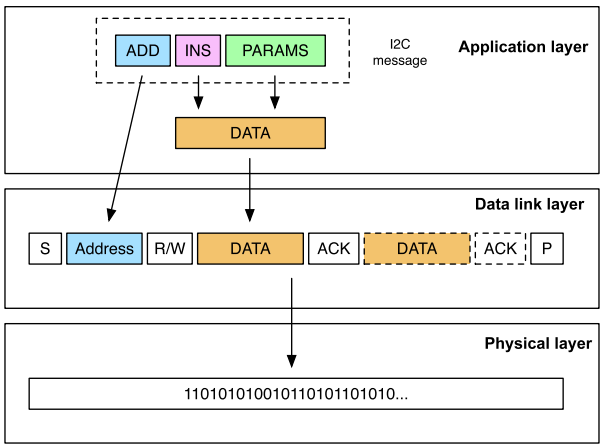


Fig. 5. I^2C frames.

A behaviour may have several definitions. Simply put, a behaviour is a reaction to a stimulus [18]. Mataric [29] defines behaviours as processes or control laws that achieve and/or maintain goals. For example, 'avoid-obstacles' maintains the goal of preventing collisions, and 'go-home' achieves the goal of reaching some home destination. Within the framework of this paper, a behaviour is considered an independent procedure or function that is in charge of a specific task. Behaviours may have states or be influenced by the state of the module.

3.3. Command exchange protocol

The command exchange protocol is used for communication between modules and the CC. It is based on I^2C upon which the message structure is built, as shown in Fig. 5. To be transmitted throughout the I^2C bus, messages have to be formatted into the I^2C format.

The two bottom layers of the protocol are the physical and data link layers of the I^2C protocol. The application data layer, which is responsible for composing the messages to be sent to the modules and the central control, is built over these two layers. In this layer, I^2C messages are structures composed of three fields: address, instruction and parameters (depending on the instruction, an I^2C message may have none, one or several parameters). Each module has a pre-defined address assigned when it is programmed (from 0 up to 63): 63 is for the CC, 0 is for broadcast messages and 1–62 are for the modules. Parameters are codified in the following manner: the first byte codes for the parameter (it is also used to determine the length of the bytes that follow), and the following bytes code for information. Parameters are shown in Table 1. Finally, instructions can be divided as follows (see Fig. 6):

- Low-level commands (LLCs): messages sent from the CC to the modules. According to the processing of the messages in the module, LLC messages can be divided into the following levels:
  - LLC level 2 (LLC2): they do not have to be translated by the heterogeneous layer.
  - LLC level 1 (LLC1): they have to be translated by the heterogeneous layer.
- High-level commands (HLCs): messages sent from the operator to the central control.

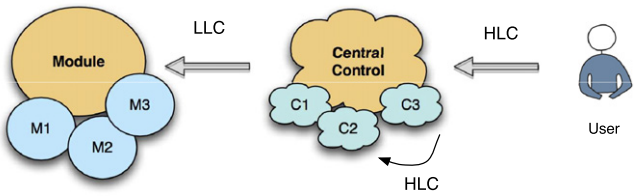


Fig. 6. HLC and LLC commands.

Table 1
Types of parameters.

Name	Code	Size	Range	Coding
Angle	1	1 byte	[−90°..90°]	Binary
Enum	2	1 byte	[0..255]	Binary
String	3	$n+1$ bytes	–	$n+1$ characters
Value	4	2 bytes	[0..65535]	Binary
Bool	5	1 bit	–	Binary
ModuleID	6	1 byte	$\{r, e, s, h, c\}$	Binary
MDL	7	String	–	Binary

3.3.1. LLCs

LLCs are commands sent by the CC to the modules and the answers to these messages. Currently implemented LLC1 and LLC2 commands are shown in Tables 4 and 5 (at the end of the article), respectively. The parameters of SIM (Send Info of the Module), AIM (Answer Info of the Module), SIE (Send information of the Environment) and AIE (Answer information of the Environment) commands (in Table 5) are shown in Table 7 (also at the end of the article).

3.3.2. HLC

HLCs are commands that can be sent to the CC by the operator to perform a specific task. The commands are specified in Table 6 (at the end of the article).

RPL parameters are composed of a first value, which indicates the type of position (Table 8 third column), plus the coordinates $[(x, y, z)]$ in millimetres (three integers) (when needed). The DO, SIR, AIR, SIE and AIE parameters are shown in Table 8.

3.4. Module description language (MDL)

MDL is a language created to describe the capabilities of one module to the CC and other modules [30].

MDL is essential for inferring functions or skills for the entire robot from the module features through rules and inference engines. With MDL, it is possible to create units (groups of modules) that are able to perform more complex tasks.

MDL is useful for developing new behaviours by combining module skills. It is based on a series of indicators that describe the tasks that the module can perform and a range of values that indicate the level of performance for each indicator.

Default MDL values are assigned in the design phase, but it is important to note that module MDL indicators are dynamic and that they may vary during the development of a task. Thus, MDL indicators can malfunction or even stop working in the module. For example, the servomotor of a module may become stuck and may be able to turn only a percentage of its nominal range of motion. When the module detects this, it can communicate the issue to the robot via MDL commands.

3.4.1. Indicators

Indicators are presented in **Table 2**. Extend/contract refers to the capability of a module to increase or decrease its length. By contrast, support refers to the capability of a module to become fixed to the pipe.

Push in pipe indicates that the module can go forwards by itself inside a pipe, whereas Push in open air refers to large spaces (including large-diameter pipes).

Table 2
MDL indicators.

Position	Acronym	Name
1	Ext	Extend/contract
2	Sup	Support
3	Push_pipe	Push in pipe
4	Push_flat	Push in open air
5	RotX	Rotate in its x axis
6	RotY	Rotate in its y axis
7	RotZ	Rotate in its z axis
8	Att	Attach/detach to/from other modules
9	Sense_front	Sense proximity front
10	Sense_back	Sense proximity backwards
11	Sense_side	Sense proximity lateral
12	Sense_temp	Sense temperature
13	Sense_humi	Sense humidity
14	Sense_grav	Sense gravity
15	Grab	Grab
16	Drill	Drill
17	PS	Power supply

Table 3
MDL values.

Value	Indicator
0	No competence for that skill
1	Little competence
2	Medium competence
3	Good competence

Rotate in its $x/y/z$ axis means it has a DOF along that axis.

Attach and Detach to/from other modules is designed for self-reconfigurable modules with active links (SMAs or electromechanical latches) [31,32].

Sense proximity front/backwards/lateral refers to any sensor that may detect obstacles. Sense temperature/humidity refers to the capacity for measuring temperature/humidity. Sense gravity indicates that it has accelerometers.

Grab indicates that it is able to grab objects. Drill indicates that it is able to make a hole.

Power supply indicates that it has a power supply to share.

3.4.2. Values

Each indicator is associated with a value that indicates the level at which the module can perform such a task (Table 3). This value is divided into four levels from 0 to 3 (from no competence to good competence).

3.4.3. Packaging

The values corresponding to each module are packed into a single structure—an array ranging from 0 to 3. For example, for the rotation module, it would be:

```
MDL (Rot_mod) = [00003300000003000]
```

and for the helicoidal module:

```
MDL (Heli_mod) = [00310000000000000]
```

4. Low-control layer

This section is dedicated to the behaviour-based control programs running in each of the modules, also called the low-level control.

All behaviours running in the modules[6] share some common characteristics. Their goals can be to perform an activity, to attain a goal or to maintain some state. Behaviours encode time-extended processes (not atomic actions) in a relatively simple manner, are introduced into the system incrementally, from the simple to the more complex, and can be concurrently executed. Their inputs may

[6] Not all behaviours are implemented in all modules.

Table 4
LLC1 commands.

Acro	Instruction	Description/remarks	Param
MS1	Move servo 1	Indicates the position of the servo 1	Angle
MS2	Move servo 2	Indicates the position of the servo 2	Angle
GS1	Get value servo 1	Demands the position of the servo 1	None
GS2	Get value servo 2	Demands the position of the servo 2	None
EX	Expansion	For extension/support module	None
CT	Contraction	For extension/support module	None
INH	Inchworm position	Indicates the position in the inchworm gait: first support (1), extension (2) or second support (3)	[1..3]
GP	Get position	To demand what is the position in the chain	None
GPS	Get position start	Chain identification phase starts	None
GPF	Get position finish	Chain identification phase ends	None
MDS	MDL phase start	MDL phase starts	None
MDF	MDL phase finish	MDL phase ends	None
SPT	Split	Detach from the previous module	None
ATT	Attach	Attach to the previous module	None
SS1	Send value servo 1	Send the value of servo 1	Angle
SS2	Send value servo 2	Send the value of servo 2	Angle
TS	Touch sensor	It points out that the touch sensor has been activated	Enum
TSF	Touch sensor final	The elbow mode is over	None
PC1	My position in chain is first	Answers from all modules in the chain but the last	Module ID (r, e, s, etc.) or MDL parameters
PCL	My position in chain is last	Answer from the last module	Module ID (r, e, s, etc.) or MDL parameters

Table 5
LLC2 commands.

Acro	Instruction	Description/remarks	Param
MO1	1D sinusoidal gait	Vertical sinusoidal movement	None
MOS	Serpentine movement	Horizontal sinusoidal movement	None
MRO	Rolling	Lateral movement	None
MSW	Sidewinding	Lateral movement	None
TUR	Turning	Arc movement	None
TUP	Turning in pipe	Pushing against the wall	None
MWO	Move inchWOrm	Inchworm gait	None
MHE	Move HElicoidal	Move pushing forwards	None
RTC	Reset time counter	For synchronisation	None
STP	Stop	Stop the module	None
RST	Restart	Restart the module	None
CM	Change mode	To change the working mode	Enum
PO	Polling	Anybody has something to say?	None
SIE	Send information of the environment	The information demanded will be specified by the parameters	Enum
SIM	Send info of the module	Consumption, orientation, etc.	Enum
SYC	Send your capabilities	Say what you can do: MDL	None
AMC	Answer: my capabilities	MDL specific capabilities	String
AIE	Answer information of the environment	Sends the information demanded	Enum + value
AIM	Answer: info of the module	Consumption, orientation, etc.	String
–	Answer to the polling message	It depends on the module	–

Table 6
HLC commands.

Acro	Instruction	Description/remarks	Param
STP	Stop	Refers to the whole robot	None
RST	Restart	Refers to the whole robot	None
RPL	Reach a place	Go to the end of a part of the pipe, go to the next bifurcation, go to a specific coordinate, etc. The place will be specified by the parameters	Enum + value
DO	Do a task	Repair, make a hole, etc. The task is specified by the parameter	Enum
EXP	Explore		None
SIR	Send information of the robot	The information demanded will be specified by the parameters	Enum
SIE	Send information of the environment	The information demanded will be specified by the parameters	Enum
AIR	Answer information of the robot	Sends the information demanded	Enum + value
AIE	Answer information of the environment	Sends the information demanded	Enum + value

Table 7
SIM and AIM, SIE and AIE parameters.

ID	SIM and AIM parameter	SIE and AIE parameter
1	Average consumption	Temperature
2	Consumption peaks	Humidity
3	Number of working motors	Picture
4	Orientation	
5	Distance covered	
6	State of the batteries	
7	State of contraction/extension	

Table 8
SIR, SIE and RPD parameters.

ID	SIR and AIR	SIE and AIE	RPL	DO
0		Coordinates		
1	Average consumption, mA (1 integer)	Temperature (1 integer)	The end of this part of the pipe	Repair
2	Consumption peaks, mA (1 integer)	Humidity (1 integer)	The next bifurcation	Make a hole
3	Working modules, array of module IDs (string)	Picture (TBD)	Until you touch something	
4	Orientation, degrees, (3 angle)	Home		
5	Distance covered, millimetres (1 integer)			
6	State of the batteries, ok or no ok, (bool)			

come from sensors or from other behaviours, and their outputs may go to actuators or to other behaviours.

Generally, behaviours can be described as shown in Fig. 7. The activation conditions are the only conditions that must be met for a behaviour to run. If these conditions are fulfilled, the behaviour will run. Some behaviours do not have activation conditions, and they are always running. Some examples of activation conditions are command (from CC or operator), enable signals from other behaviours, and low battery signals.

Stimuli are the inputs of behaviours and include the position of the module, position of the actuators, internal variables of the modules (intensity, torque), and state of the module. Actions are the outputs of behaviours and define what the behaviours intend to do, including modifying the position and orientation of the module, blocking the motors, retrieving module state or actuator position. The outputs of behaviours must be coordinated, as will be explained in the following sections.

[7] Open air can also be a wide pipe (a pipe with a large diameter) in which the micro-robot can perform movements as if it were in the open air.

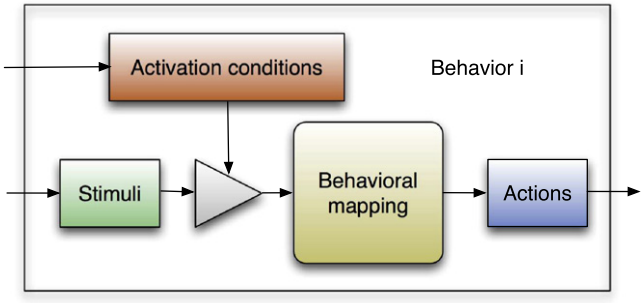


Fig. 7. Behaviour scheme.

All behaviours share the same information of the module to which they belong, such as module ID, module capabilities, position in the micro-robot, and working mode.

An interesting parameter is the working mode of a module, which refers to the situation in which the module is normally referring to environmental or system conditions or characteristics. The working mode consists of information that the CC sends to the modules after processing the information previously sent by the modules to the CC after compiling data obtained from its sensors. This working mode indicates the environment in which the micro-robot is working. The environment could be inside a pipe, open air[7] or general terrain. The working mode is information that is essential for every behaviour of the module to perform its tasks. Only some behaviours will make use of it.

4.1. Embedded behaviours

There are several types of behaviour, which have been classified into several categories (as described in [18]) according to the type and complexity of the tasks they perform. Some behaviours perform simple tasks, whereas some are based on other behaviours to perform more complex tasks.

The behaviours that have been defined are as follows:

1. Survival behaviours (seek to maintain the integrity of the module): Avoid overheating, Avoid actuator damage, Avoid mechanical damages.
2. Perceptual behaviours: (attempt to gather information about the module and its environment): Self-diagnostic, Situation awareness, Environment diagnostic.
3. Walking behaviours (move the module): Vertical sinusoidal movement, Horizontal sinusoidal movement, Worm-like movement, Push-Forward movement.

The execution of each behaviour is independent of that of the others, and it is influenced by the situation and state experienced at a particular moment. Not all behaviours can act at the same time; thus, they have to be coordinated.

A description of the implemented behaviours is given next, followed by the coordination mechanisms.

4.1.1. Avoid overheating

The purpose of this behaviour is to make sure that the accumulated heat is maintained under certain limits to prevent circuit damage. For example, the heat produced in the coil of the motors by the electric current may lead to the burning of the coil. To avoid the overheating of the circuits, the electric current has to be limited.

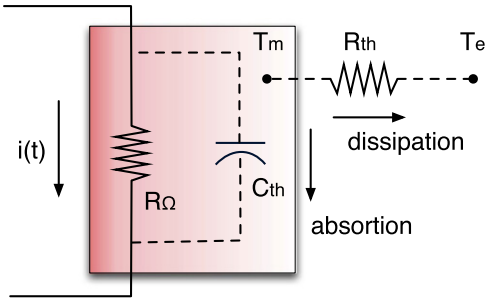


Fig. 8. Heat dissipation and absorption sketch.

The heat generated in the coil of the motor follows the Joule effect formula. Part of this heat is transmitted to the environment, and part of it is absorbed by the wire (increasing the temperature) as shown in Fig. 8 and described by Eq. (1). Eq. (1) leads to Eq. (2), where: R_Ω is the electrical resistance in (Ω), i is the electrical current in (A), t_m is the temperature of the motor in (°C), t_e is the temperature of the environment in (°C), R_{th} is the thermal resistance (°C/W) and C_{th} is the thermal capacitance (W s/°C).

$$Q_{\text{generated}} = Q_{\text{dissipated}} + Q_{\text{absorbed}} \tag{1}$$

$$R_\Omega \cdot i^2(t) = \frac{t_m(t) - t_e(t)}{R_{\text{th}}} + C_{\text{th}} \cdot \frac{\mathrm{d}t_m(t)}{\mathrm{d}t}. \tag{2}$$

In the Laplace domain, Eq. (2) is expressed as Eq. (3). To perform a Laplace transformation, a variable change has been performed: $\alpha(t) = I^2(t)$.

$$R_\Omega \cdot \alpha(s) = \frac{T_m(s) - T_e(s)}{R_{\text{th}}} + C_{\text{th}} \cdot T_m(s) \cdot s. \tag{3}$$

Thus, T_m, the temperature of the motor, which is the variable that should be under supervision, can be obtained as shown in Eq. (4).

$$T_m(s) = \frac{T_e(s) + R_{\text{th}} \cdot R_\Omega \cdot \alpha(s)}{1 + R_{\text{th}} \cdot C_{\text{th}} \cdot s}. \tag{4}$$

Applying the Euler transformation $s = \frac{1-z^{-1}}{T}$, where T is the sampling period, the following is obtained in the Z domain:

$$T_m(z) = \frac{T \cdot T_e(z) + T \cdot R_{\text{th}} \cdot R_\Omega \cdot \alpha(z) + R_{\text{th}} \cdot C_{\text{th}} \cdot T_m(z) \cdot z^{-1}}{T + R_{\text{th}} \cdot C_{\text{th}}}. \tag{5}$$

Applying the inverse transform, the discrete time equation is obtained as follows:

$$T_m[n] = \frac{T \cdot T_e[n] + T \cdot R_{\text{th}} \cdot R_\Omega \cdot I^2[n] + R_{\text{th}} \cdot C_{\text{th}} \cdot T_m[n-1]}{T + R_{\text{th}} \cdot C_{\text{th}}}. \tag{6}$$

The temperature of the environment is measured by a temperature sensor; the electrical current is obtained from the measurements of the sensors and the electrical and thermal resistance; and thermal capacitance is considered to be constant.

The behaviour continuously monitors the temperature and intensity, and when overheating in the servomotors is detected, the motors are stopped immediately.

Fig. 9 shows the evolution of one of the servomotors of the rotation module under different situations. In the beginning, the servomotor performs a sinusoidal movement as part of a snake-like movement, and the temperature increases to the normal working temperature (from point A to point B). After a short pause (from B to C), the servomotor is commanded to perform a movement that is blocked; therefore, the temperature starts to increase dangerously

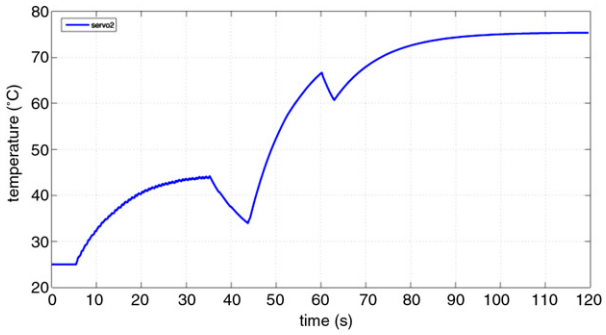


Fig. 9. Temperature evolution of one of the servomotors of the rotation module.

(from C to D). When the servomotor releases, the temperature starts to decrease (from D to E), and when the servomotor is commanded to perform the movement again, the temperature again increases (from E in advance).

Thus, when the estimated temperature exceeds a certain limit (50 ° C), the movement of the servo is stopped for safety reasons.

4.1.2. Avoid actuator damage

The purpose of this behaviour is to ensure that the torque of the motors remains under certain limits to avoid damage to the motors or actuators. If the torque exceeds a certain limit, the servomotors are immediately released.

This purpose is achieved by keeping the instant current intensity under a certain limit, which has been determined experimentally. In Fig. 10, the servomotor is trying to move from 100° to 180°, but it is blocked at 135°. The consumption increases very fast, and consequently it should be blocked below 120 mA. A limit of 100 mA is a reasonable value in our case.

4.1.3. Avoid mechanical damages

This behaviour is in charge of the mechanical security of the module, resolving any possible danger it may encounter owing to improper use of the actuators.

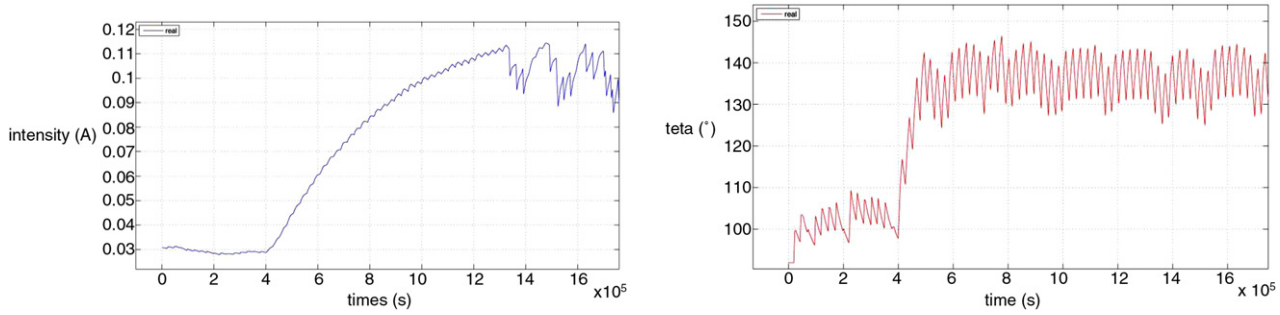
One of the tasks that this behaviour controls is avoiding mechanical singularities. Two types of singularity are considered: those at the limits of the workspace of the robot and those inside the workspace of the robot.

In the extension module, the behaviour avoids singular points in the two crank-connecting mechanisms. Singular points are produced, for example, when the links of each arm are aligned. They are produced at angles of 25° (inside the workspace) and 147° (limit of the workspace). The smaller value cannot be reached because of the mechanical configuration (see Fig. 11(a)), but the communication of that position to the servomotor must be prevented because it could break itself or the module trying to reach it. The higher value can only be avoided by using software (see Fig. 11(b)).

Because the movement of the end connector of the extension module is limited by the sliding bar placed at the centre, there are several combinations of the angles of the two actuators that are not physically possible. Positions that must be avoided are computed from the kinematic equations of each module.

The same effect occurs in the support module: singular points occur when the two links are aligned. Additionally, as described for the extension module, the mechanical design prevents that position from being reached, but that position should also be avoided.

All modules feature their specific implementation of this behaviour and execute it when needed.



(a) Intensity. (b) Angle.

Fig. 10. Maximum servomotor consumption with blocking.

(a) Lower position. (b) Higher position.

Fig. 11. Extension module at its higher and lower positions.

4.1.4. Self-diagnostic

The purpose of this behaviour is to examine the functioning of the module: whether the actuators can move, whether the levels of intensity and torque are acceptable, whether the communication bus is working, whether the synchronism line is functioning, or whether the sensors are working correctly.

This behaviour records the setpoints (desired positions) of the actuators and compares them with their real positions. If the positions are not approaching (and there is no problem with the torque and intensity, meaning it is not blocked), then there may be a problem with the actuator, and an alarm state is activated and communicated to the CC.

To verify the synchronism lines, in the configuration check phase, the behaviour checks if the signals Sin and Sout have been activated at any time. If not, there may be a problem, and an alarm state is activated and communicated to the CC.

4.1.5. Situation awareness

This behaviour attempts to determine the position of the module/micro-robot: inside a narrow pipe, a wide pipe, or open air. It makes use of the contact sensors, infrared (IR) sensors, and the intensity and torque control system of the servomotors, among other sensors.

Through the IR sensors, the module is capable of determining whether it is in an open environment or inside a pipe. If it is inside a pipe, it can detect whether the pipe is wide or narrow.

The touch (and camera) module plays a very important role because it features touch sensors to detect obstacles, which, in this case, are elbows and bifurcations. Through the contact sensor, the module can detect whether it has collided into something, and other behaviours may act accordingly.

4.1.6. Vertical and horizontal sinusoidal movements

Modules with rotational DOFs can perform several movements (some of them are similar to snake-like rolling, rotating, or lateral shifting) based on a central pattern generator [33]. The position of the actuators follows two sinusoidal waves: one for the vertical actuators and one for the horizontal actuators (Eqs. (7) and (8)).

$$\Theta v_i(t) = A_v \cdot \sin(\omega_v \cdot t + (i-1) \cdot \phi_v) \tag{7}$$

$$\Theta h_i(t) = A_h \cdot \sin(\omega_h \cdot t + (i-1) \cdot \phi_h) \tag{8}$$

where Θ is the angle of the actuator, A the amplitude, ω the angular velocity and ϕ the phase. By changing the parameters, different movements can be achieved (see [34–38]). There is a wave for the vertical joints (Θ_v) and another one for the horizontals (Θ_h), and each is managed by a behaviour.

There are two methods to synchronise t. The first is by resetting t for all modules simultaneously when a start sequence message is received from the CC. This is the easiest method, but the drawback is that modules may lose synchronisation. To avoid this, a synchronisation message must be provided at regular time intervals (i.e., every 2 s).

The second method is by using the synchronism line:

- the first module steps t and activates the output synchronism line
- the second module detects the input line activated, steps t and activates the output line

- and so on until the last module detects the input line activated, steps t, activates its input line
- the penultimate module detects the output line activated, steps t and activates the input line
- and so on.

Both methods have been implemented. The method that uses the synchronism line is more accurate, but it cannot be used when the synchronism line has to be used for other purposes. In such cases, the other method must be used.

These behaviours are also in charge of executing turns. In open spaces, turns are achieved by placing horizontal joints at a fixed position at all times. The robot is shaped like an arc. The radius of curvature of the trajectory can be modified by modifying the degree of rotation of the horizontal joints.

Inside pipes, it is also possible to negotiate elbows by pushing against the pipe walls. The sequence occurs as follows:

- The first module (M1) turns 90°.
- M1 turns up the synchronism line with module M2.
- When M2 detects the synchronism line up, M2 turns 90°.
- When M2 turns at a predetermined angle (approximately 60°), M2 turns downs the synchronism line with M1.
- M1 returns to the initial 0° position.
- When M2 has turned 90°, it then turns up the synchronism line with M3 (see Fig. 12).

4.1.7. Worm-like movement

This behaviour can be found in modules with extension–contraction capabilities. Each module knows if it has support or extension capabilities. Worm-like movement is performed by a combination of extension–contraction mechanisms.

A description of this procedure can be found in [1].

4.1.8. Push-forward movement

This behaviour can be found in modules that have self-propulsion capabilities, such as the helicoidal module. This behaviour activates the actuator to move forwards or backwards as commanded.

4.2. Behaviour fusion

Behaviour coordination is a complex task. Some behaviours collaborate to achieve their goal (cooperation), whereas others compete (competition) or act independently from each other. The scheme is explained in Fig. 13.

Behaviours are divided into sets of priorities and tasks.

Walking behaviours (e.g., vertical sinusoidal and horizontal sinusoidal) control the actuators of the module; some of them control one actuator, whereas others control two or more. They may be complementary, and thus, the modules output is combined to achieve its goal. Its output can be overridden by LLC1 commands received directly from the CC.

Perceptual behaviours act independently because they only inform and have no actuator control. However, their output feeds back to the other behaviours with information regarding broken actuators, current situation, and other parameters.

Survival behaviours have the highest priority (if they compete with other behaviours, they will win) because they try to maintain proper functioning of the module. They can inhibit the output of another behaviour if such a behaviour endangers the integrity of the module. For example, if the output of a behaviour is to move a servomotor to a specific position where power consumption is too high, the position is released.

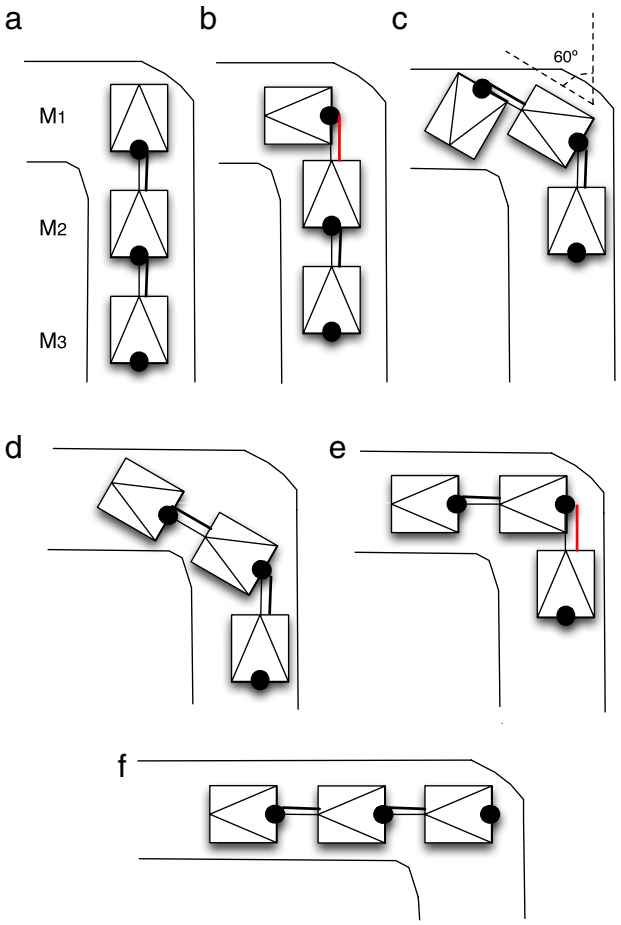


Fig. 12. Elbow negotiation inside pipes.

5. Heterogeneous middle layer

The heterogeneous layer controls several tasks that take place between the module and the CC and/or other modules, such as communication. Each time a command is received by the module, it is processed by the heterogeneous layer and translated into specific instructions for the module. Conversely, when the module needs to send a message, this is performed by the heterogeneous layer.

For example, when an action has to be executed (i.e., extend), the CC sends an I^2C message to every module with the command to follow. The heterogeneous layer of each module translates this message into proper commands for the module. It is important to note that all messages are the same irrespective of the module at which they are aimed; therefore, the module knows what actions it has to perform.

The heterogeneous layer also controls the following tasks: communications, configuration check and MDL phase.

5.1. Communications

The heterogeneous layer receives commands from the CC and sends commands to the CC when the CC asks if there is something to communicate (polling).

At certain time intervals, the CC sends a message to all modules asking if they have something to communicate (polling). This process is how the modules communicate with the CC or other modules. In the inverse procedure, the module sends a command to the CC, and if necessary, the heterogeneous layer translates the message.

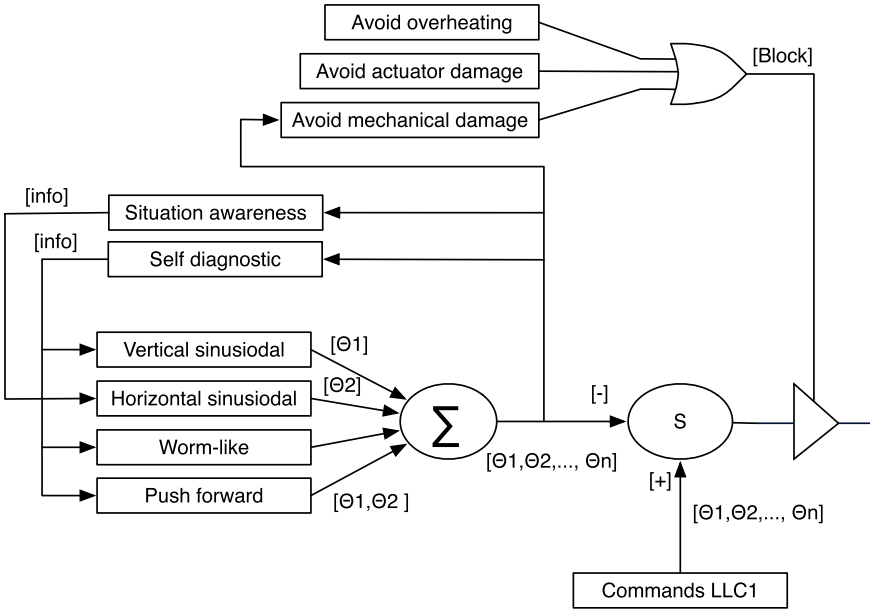


Fig. 13. Embedded behaviours fusion scheme.

5.2. Configuration check

The purpose of this task is to determine the configuration of the micro-robot and the position of the modules in the robots chain. The first time this behaviour acts is after the mechanical connection of the modules and power-up, when the phase of awareness starts: every module knows its position in the modular chain. Then, the behaviour can be activated by the CC any time it is necessary to know the configuration (e.g., after split up and if a module is broken).

This procedure occurs as follows (see Fig. 14):

- The CC sends a GPS message to all modules.
- All modules activate their synchronism lines.
- The first module (it knows that it is the first because its S_{in} synchronism line is down) replies with a PC1 message (this message is sent to the CC and includes the ID of the module: e.g., *r* for rotation, *s* for support).
- The first module puts the S_{out} synchronism line down, and thus, the second module knows it goes next (because now its S_{in} synchronism line is down).
- The second module sends a PC1 message and puts its S_{in} synchronism line down, and thus, the first module knows it has finished.
- The CC continues to collect all of the messages.
- This process repeats for all modules.
- When it is the last module's turn (it knows it is the last because its S_{out} synchronism line is down), it sends a PCL message.
- The CC sends a GPF message, and thus, the last module knows it has finished.

5.3. MDL phase

The MDL phase follows a similar mechanism to that of the configuration check phase, but instead of sending the id parameter in the PC1 and PCL messages, the module sends the MDL string that shows capabilities.

6. High-control layer

The CC represents the high-control layer in the control architecture. It makes the main decisions of the robot (i.e., what it is going to do and how) independently of the modular composition of the micro-robot. The CC can be run on an external PC – as it is now – or in a specific module. It is also based on behaviours that target the entire micro-robot.

To determine the capabilities of the robot, the CC makes use of an inference engine and a set of rules that make use of the MDL commands from each module to set the capabilities for the entire micro-robot.

Each module has several features that define what it can do, although a set of modules together can have newer features. Modules can be grouped into units to develop different capabilities; these units can in turn be grouped into super-units to possess even newer capabilities.

The capabilities of the entire micro-robot are the consequence of a combination of the capabilities of all modules and the position of the modules in the chain. It is not equivalent to have an extension module between two support modules as it is to have the extension module at the side of two support modules in a row. In the first case, the chain could perform an inchworm movement, whereas this is not possible in the second one. There are three possible locations for the modules:

- Anywhere: they can develop their capabilities independently of where are they placed.
- In sequential order (but not adjacent).
- Adjacent: one after the other.

To determine the capabilities of a micro-robot, a set of rules has been implemented (shown in Table 9). These rules can be extended either by writing new rules when new features appear or by developing new rules by learning.

In general, rules can be described as follows:

$$\sum \text{sequential(MDL)} + \sum \text{adjacent(MDL)} + \sum \text{anywhere(MDL)} \Rightarrow \sum \text{robot(MDL)}$$

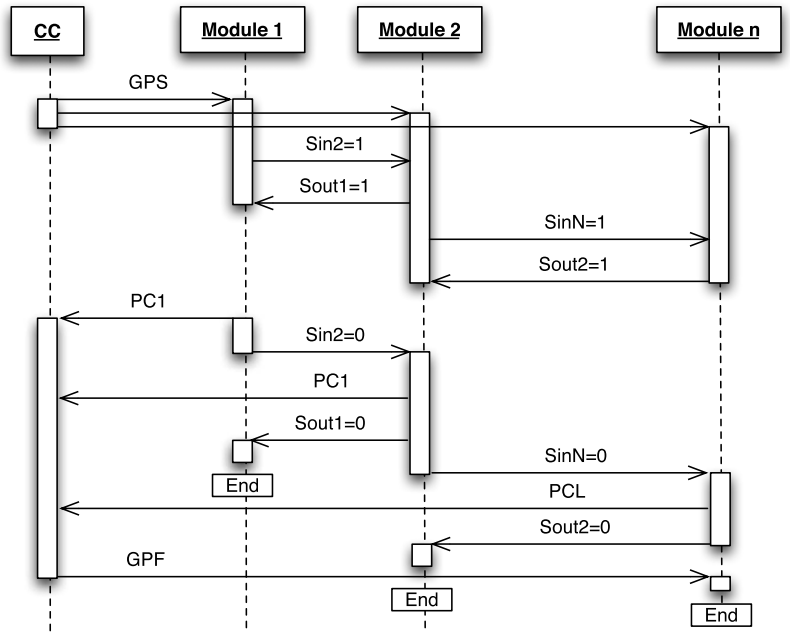


Fig. 14. Configuration check sequence diagram.

Table 9
Table of rules.

Anywhere	Sequential	Adjacent	Robot
Bat		Rot + Rot + Rot	Ext
Bat	Sup + Ext + Sup		Forward/backward movement (inchworm)
Bat	Sup +.. + Sup		Sup unit
Bat	Ext +.. + Ext		Ext unit
Bat	Sup unit + Ext unit + Sup unit		Forward/backward movement (inchworm)
Bat + Rot + Push_pipe			Turning
Bat + Push_pipe			Forward movement
Bat + Push_pipe			Backward movement
Bat		Rot + Rot + Rot	Forward movement (snake)

where $\sum$ sequential(MDL) means the combination of the MDL parameters of sequential modules, $\sum$ adjacent(MDL) the combination of parameters of adjacent modules, $\sum$ anywhere(MDL) the combination of parameters of modules where its position doesn't matter, and $\sum$ robot(MDL) are the MDL parameters of the whole robot.

For example, the rotation module does not have extension/contraction capabilities, but a unit composed of three rotation modules together does have that feature:

$$\begin{aligned}&\text{ADJACENT}(\text{Rot}_{\text{mod}} + \text{Rot}_{\text{mod}} + \text{Rot}_{\text{mod}})\\&\quad + \text{ANYWHERE}(\text{Open}_{\text{air}})\\&\Rightarrow \text{Extension/Contraction (grade 3)}\\&\text{ADJACENT}(\text{Rot}_{\text{mod}} + \text{Rot}_{\text{mod}} + \text{Rot}_{\text{mod}})\\&\quad + \text{ANYWHERE}(\text{Pipe}) \Rightarrow \text{Extension/Contraction (grade 1)}.\end{aligned}$$

To explain this clearly, let us suppose that a chain is composed of three modules with the following MDL structures:

```
MDL(module 1) = [00003203001003000]
MDL(module 2) = [00003302130003000]
MDL(module 3) = [00003000200003111]
```

Each MDL structure is merged with each of the indicator masks to determine whether the module has that specific capability. For example, the mask for indicator RotX is [00001000000000000]. Merging each of the modules' MDLs with the masks gives [00003000000000000].

Then, capabilities are inserted in the rules, and those that are fulfilled are activated. In this case, the activated rules are

$$\begin{aligned}&\text{SEQUENTIAL}(00003000000000000,\\&\quad 00003000000000000, 00003000000000000)\\&\quad + \text{ANYWHERE}(\text{Open}_{\text{air}}) \Rightarrow 30000000000000000\\&\text{SEQUENTIAL}(00003000000000000,\\&00003000000000000, 00003000000000000)\\&\quad + \text{ANYWHERE}(\text{Pipe}) \Rightarrow 10000000000000000.\end{aligned}$$

Thus, new capabilities are obtained, which can in turn create new rules to obtain newer capabilities.

Through these rules, the CC can deduce or infer the capabilities of a robot. It goes through all of the rules and selects those that are fulfilled. Then, the procedure is repeated by incorporating previously obtained conclusions. This procedure continues until there are no new rules fulfilled in a cycle.

The CC can also deduce or infer which modules are needed for a specific task. For example, if a robot needs to split into two, it can decide the optimal point at which to split such that each part of the robot keeps the necessary modules to accomplish the task to be executed.

New rules can be added or modified at any time without reconfiguring the system. The inference engine makes use of the active rules at every moment.

6.1. Behaviours

Continuing with the classification made in Section 4.1, the behaviours that have been defined for the CC are shown in the following list. As explained previously, some behaviours perform simpler tasks, whereas more complex behaviours are based on these to perform more complex tasks.

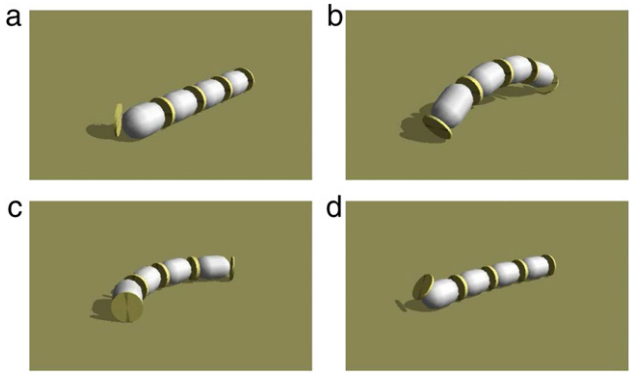


Fig. 15. Example of orientation behaviour.

1. Postural behaviours: Balance/stability.
2. Walking behaviours: Move straight forwards/backwards, Turn to left/right, Move laterally, Rotate.
3. Protective behaviours: Obstacle negotiation.
4. Path following behaviours: Edge following, Pipe following, Stripe following.
5. Exploration behaviours: Wandering.
6. Goal Oriented behaviours: Reach a landmark, Reach a place, Find a pipe break, Repair.

6.1.1. Balance/stability

This behaviour is responsible for determining the orientation of the robot and changing it when necessary in order to be in the correct position for the current task.

The information regarding the orientation of the module is taken from the accelerometer, from the servomotors or from information received from other modules or the CC.

For example, if a module with two rotational DOFs wants to turn to the right, depending on its orientation, it will use one of the DOFs. If neither of them is in the right position, the behaviour will make the necessary movements to place the module in the right position.

For certain movements, it is important to maintain a specific posture. For example, in the vertical sinusoidal wave movement, if the robot lays down, it is necessary to recover the position before continuing with the vertical sinusoidal movement.

In Fig. 15, an example of the performance of the orientation behaviour is shown. In (a), it is possible to see that the first DOF is horizontal. In (b), the robot makes an arc and consequently falls down as shown in (c). Then, it orients itself into a straight position, leaving the first degree of freedom vertical.

6.1.2. Walking behaviours

The move straight forward/backward behaviour controls the forward and backward movements of the micro-robot. There are several types of movement it can perform, such as serpentine, caterpillar, and inchworm. The use of one or another depends on the type of modules comprising the robot, the predominant modules, the environment through which it is moving and the state of the module (e.g., in terms of power supply and mechanical viability).

If the predominant modules are rotation modules, a snake-like gait is performed. Propagation of sinusoidal waves in the horizontal plane is called serpentine motion, whereas that in the vertical plane is called caterpillar locomotion. Serpentine motion is more suitable for open spaces, whereas caterpillar can be used in both pipes and open spaces. Other possible gaits in open spaces are rolling and sidewinding, although these require the micro-robot to first change its orientation. If the predominant modules are the support and the extension modules, an inchworm gait is performed.

If the predominant modules are the rotation modules, it is also possible to perform an inchworm locomotion. A group of three modules has contraction–extension capabilities, and this group could act as a unit similar to a support or extension module by following the previous procedure.

The helicoidal module has only one DOF. It can go forwards or backwards by pushing other modules. Thus, a helicoidal module can be added to other modules, and its push will be added to the other modules' push. If the locomotion of other modules is not possible or desired, the modules acquire a configuration of minimum friction that would facilitate straight forward movements.

Other modules that have no actuators act as pig modules, and they are performed by the drive modules. They only have to relay the signals coming from the synchronism line.

The move laterally behaviour controls the sideways movements of the micro-robot. As mentioned previously, there are several types of movement that a micro-robot can perform, such as lateral shifts and rolling gaits. The use of one over the other depends on the types of module comprising the robot, the predominant modules, the environment through which it is moving and the state of the module (e.g., in terms of power supply and mechanical viability). In the rolling gait, the robot can roll around its body axis. In the lateral shift gait, the robot moves parallel to its body axis. For these movements, some of the modules must have two perpendicular DOFs.

The rotation behaviour rotates the robot parallel to the ground clockwise or anti-clockwise. The robot can change its orientation in the plane. Rotation can only be performed if the micro-robot has modules that have one rotational DOF. Vertical and horizontal sinusoidal movements are used.

Finally, the turning behaviour changes the heading of the robot. To turn, there are at least two possibilities: performing a turning gait (caterpillar locomotion combined with rotation in the other DOF actuator) or to stop, rotate to a certain degree and go forwards.

The type of gait to use is defined by each behaviour according to the rules, the types of modules of the robot and the situation and destination of the micro-robot. Depending on the modules of the micro-robot, each behaviour knows which movements can be performed.

6.1.3. Obstacle negotiation

Obstacle negotiation is a complex behaviour because it makes use of other behaviours. When something is detected in the path of the micro-robot, the behaviour is in charge of selecting the appropriate actions to move around the obstacle.

If the robot is in a pipe when it encounters an obstacle, it is probably at an elbow or bifurcation. The robot must then select the actions to negotiate the turn.

When an obstacle is encountered in the open air, it is slightly more complicated because there are many available options. The easiest way is to go back and then slightly turn and go forwards. If the object is detected again, the same algorithm is performed.

6.1.4. Path following behaviours

Edge following makes use of distance (IR) and touch sensors and seeks to keep the micro-robot from operating too close to a wall or object. Depending on the measurements received from the IR sensors of the modules, the behaviour will output the coordinates where the robot should go.

Pipe following controls the movement of the robot inside a pipe, trying to keep the best movement gait and negotiating elbows and bifurcations.

6.1.5. Wandering

This behaviour controls the movement of the robot when there is no specific task selected. It is especially useful in pipes.

The robot moves around, looking for possible damage and trying to avoid colliding into an obstacle. It also may follow the pipes by making a map of the path using the travelled distance measuring system.

Fig. 16 shows an example of a micro-robot executing the wandering behaviour. This includes going forwards and negotiating an elbow when a bifurcation is detected by the contact module. The micro-robot is composed of the following modules: one contact, two rotation, one helicoidal, two rotation and one passive module. The helicoidal module is responsible for the main driving force. The rotation modules help to a small extent in going forwards, but their main task is to turn.

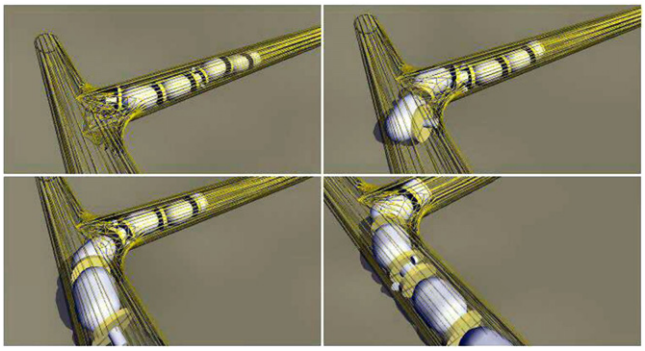


Fig. 16. Contact, rotation, helicoidal and passive modules inside a pipe.

6.1.6. Goal oriented behaviours

These behaviours are the highest-level behaviours. They make use of other behaviours to complete their tasks.

The behaviours reach a place and reach a landmark function in a similar manner. Starting from its own position, the behaviour estimates where the objective is and moves the robot in that direction. In a pipe, the objective could be to go to the next bifurcation, go forwards/backwards a certain distance, or go up/down.

In open air, possible objectives include go to position (x, y), go to the next corner, or move into the pipe in front.

The behaviour 'find a pipe break' utilises the wandering behaviour to move inside the pipe while looking for breaks or holes with the camera and IR sensors.

The repair behaviour is an example of what will be possible when repairing tools are developed and added to the robot. The behaviour will control moving the robot while it repairs the damaged pipe.

6.2. Behaviour fusion

The behaviour fusion scheme for the CC algorithms is shown in Fig. 17. Higher-level behaviours (i.e., path following, obstacle negotiation, exploration (wandering) and goal oriented) follow a subsumption-like procedure in order to coordinate. If no behaviour wants to take control, wandering is the active behaviour, but it can be subsumed by path following, which in turn can be subsumed by goal oriented; this final behaviour can then be subsumed by obstacle negotiation. Thus, obstacle negotiation is the highest-level behaviour.

Each of the path-following and goal-oriented behaviours contributes to the selection of the destination. Thus, the bottom part of Fig. 17 shows a summation of all of the individual outputs.

The output of all of the previous behaviours is the coordinates of or directions to where the robot seeks to go. This output is received by the walking behaviours, which compete amongst themselves for the control of the modules. The output of the action

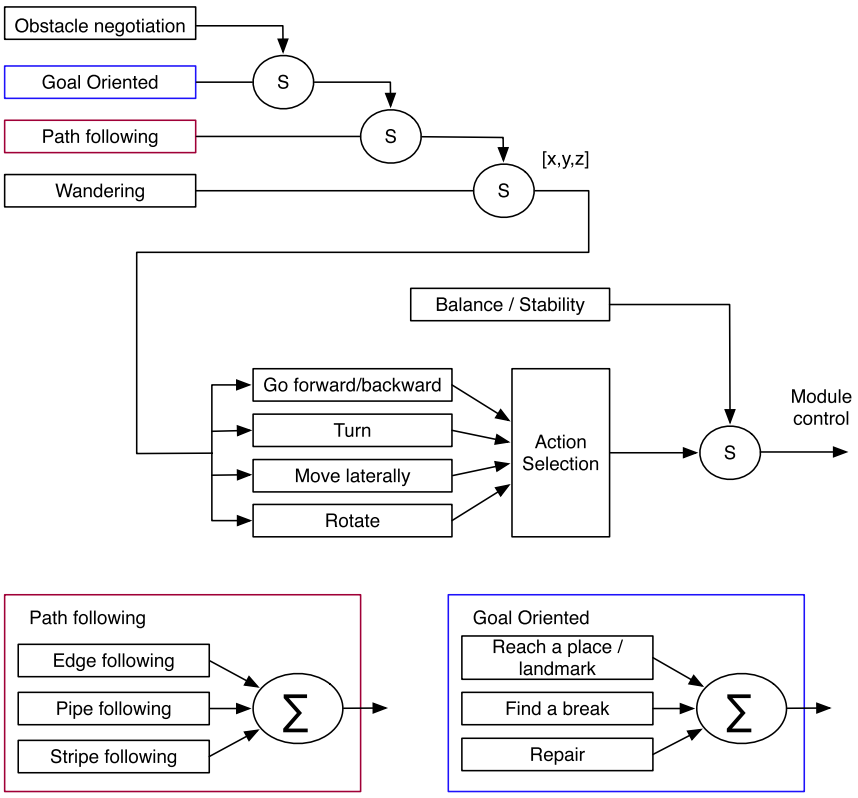
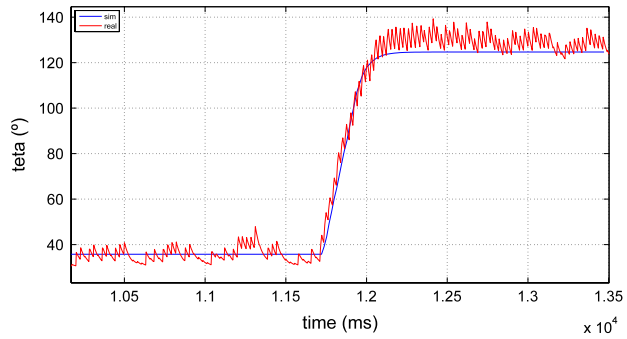
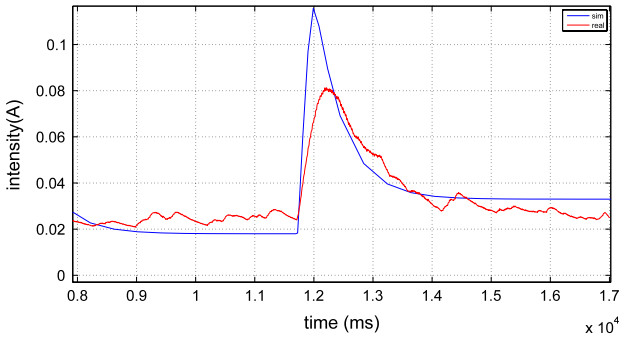


Fig. 17. Behaviour fusion scheme for central control behaviours.



(a) Rotation angle.



(b) Intensity.

Fig. 18. 30°–120°.

selection mechanism can be suppressed by the balance/stability behaviour, which is in charge of keeping the micro-robot in the most appropriate position.

6.2.1. Action selection mechanism

The outputs of the four walking behaviours (go forwards, turn, move laterally and rotate) have to be merged into a unique output. Because these are all competing behaviours, one behaviour must be selected to follow. The selection criteria depend on two factors: the situation and the destination of the micro-robot.

The situation is very important because it is not the same to move inside as it is to move in the open air; the same applies to movement across a plain terrain/pipe with respect to that across an uphill terrain/vertical pipe.

The destination of the robot is of the greatest importance. Destinations may include the following: in front of or to the left/right of or diagonal to the robot. These also encode distance scales, such as near or far. Depending on all of these, one behaviour will be chosen to take control.

7. Offline control

Offline control refers to the control algorithms that occur when the micro-robot is not running (as opposed to online control, which has been covered in the previous sections). These offline algorithms aim to select the best configuration of the micro-robot (regarding both module position and parameter configuration) for later use in the online control.

Offline control occurs in a simulator specifically developed for the micro-robot described in Section 2 (more information in [39]). A physically accurate robotic simulation system has been developed to provide a very efficient method of prototyping and verifying control algorithms, hardware design, and exploration system deployment scenarios. It is also used to verify the feasibility of system behaviours using realistic morphology, body mass and servomotor torque specifications.

Because of this simulator, it is possible to perform tests with a simulated robot based on genetic algorithms (GAs). Two types of GA have been developed:

- configuration demand: in heterogeneous configurations, for a given task, the GA has to determine which modules to use for an optimal configuration and/or the optimal position (order) of the modules in the chain.
- parameter optimisation: for a given configuration, the GA has to determine the optimum parameters for the best performance. This is particularly useful in configurations in which the micro-robot is performing a snake or inchworm movement (to optimise amplitude, angular velocity, or phase).

The results of these offline algorithms feed the action selection mechanisms and the inference engine rules of the CC, helping to develop new rules.

Table 10
Speed test of helicoidal module.

Angle (°)	0	30	60	90
Speed (cm/s) (real)	3	2, 1	1, 5	1, 2
Speed (cm/s) (simulation)	3	2, 3	1, 6	1, 3

8. Results

8.1. Validation

Validation experiments have been performed regarding not only external parameters (position, velocity) but also internal variables (torque, intensity). The simulator has been validated with data taken from real modules to adjust its parameters as much as possible, to be able to generate new movement patterns and gaits, and to test new module concepts.

As an example, the movement of one of the DOFs of the rotational module from 30° to 120° was compared to the simulated module in Fig. 18(a) (angle) and (b) (intensity).

Table 10 shows some tests performed in the helicoidal module at different slopes.

8.2. Simulation

In this section, several examples of the use of the architecture are provided.

Rotation plus support plus extension plus helicoidal modules.

One key feature of the architecture is that it allows heterogeneous modules to work together. By combining several types of module, several types of movement can be performed simultaneously: snake-like, worm-like and helicoidal. Each of them is well suited for each respective situation in pipes or open air.

Fig. 19 shows an example of the touch, rotation, helicoidal, extension and support modules working together and performing vertical sinusoidal, helicoidal and worm-like movements simultaneously.

After interconnecting the modules, the system is aware of the configuration of the robot. Modules with sensors (in this case IR) send information to the CC, which is used to determine whether the micro-robot is inside a pipe. When the command to explore is given, behaviours at the CC select the best way to move, and the behaviours of each module perform the necessary gaits.

When an elbow is detected by the first module (contact sensor), it is communicated to the CC, which informs all modules that they have to turn, after which each of them takes the necessary actions. All of this procedure is independent of the type of module and is based only on the modules characteristics, which are immediately transmitted to the CC upon operation.

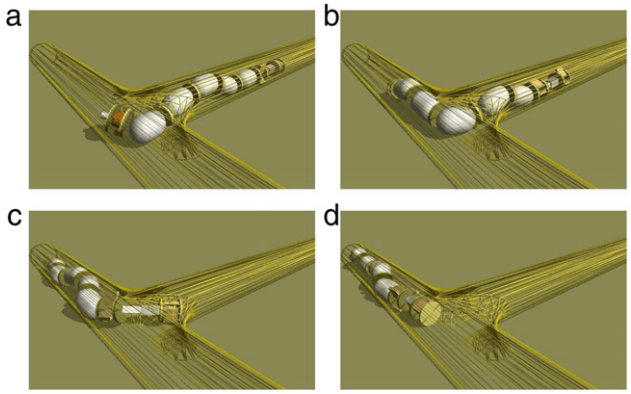


Fig. 19. Example of heterogeneous configuration.

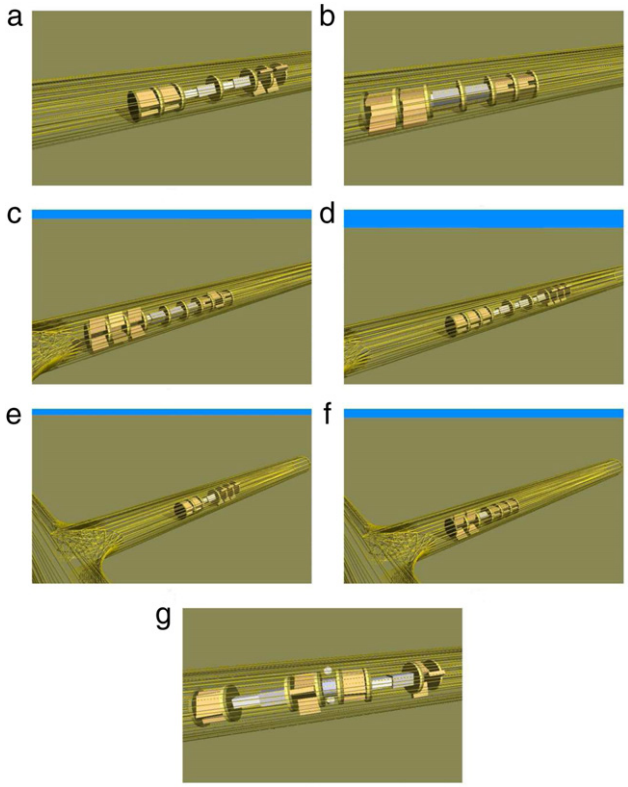


Fig. 20. Inchworm locomotion composed of several extension and support modules.

Several support plus extension modules.

The next example (Fig. 20) shows that it is possible to combine several support modules to create support units and several extension modules to create extension units. After the MDL phase, the CC can detect the support and extension modules and identify support and extension units.

These units can be composed of different numbers of modules, either two or more. Fig. 20 shows an example of a unit composed of two modules ((a) and (b)), three modules ((c) and (d)) and a combination of these ((e) and (f)). In (g), an example is shown of two different inchworm drive units working together to demonstrate that modules are aware of their positions (although this configuration does not work because the support modules of one unit avoids the movement of the extension module of the other unit).

Rotation plus inchworm unit plus rotation modules.

The third example demonstrates the flexibility of the architecture and how the same configuration of modules performs differently depending on the environment (expressed by the working mode) without reprogramming any module or the CC.

The combination of several rotation modules plus an inchworm unit leads to several possible movements. This configuration allows the micro-robot to perform an inchworm movement inside pipes and other movements (making use of the rotational DOF of the extension module) in the open air.

Figs. 21 and 22 show a chain composed by two rotation, one support, one extension, one support and two rotation modules. In the first figure, the micro-robot is performing an inchworm movement inside a pipe, whereas in the second figure, it is performing a rolling gait in open air.

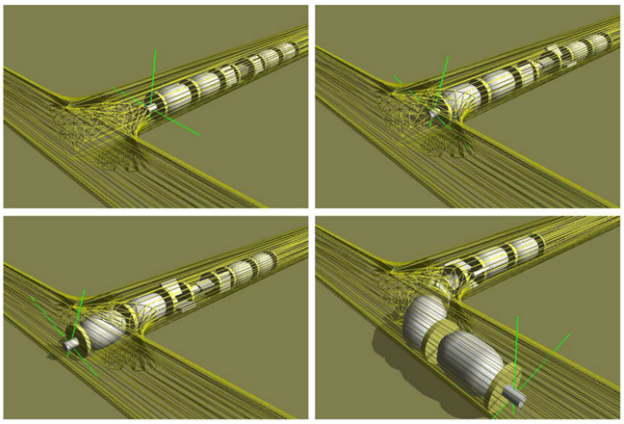


Fig. 21. Example of locomotion inside a pipe.

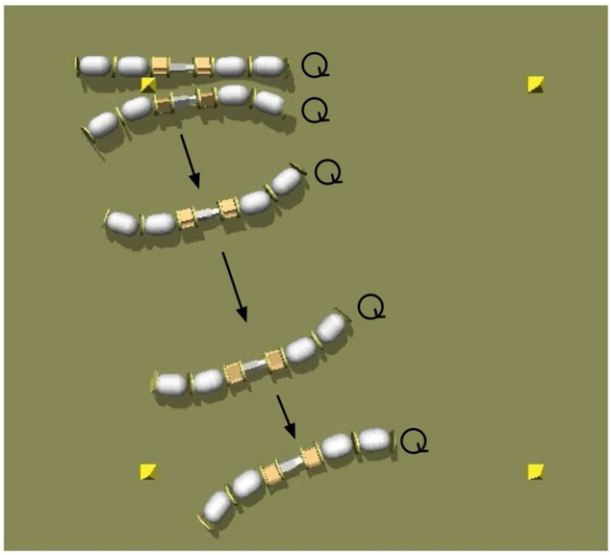


Fig. 22. Example of rolling movement in heterogeneous configuration (rotation and inchworm modules).

8.3. Optimisation

In this experiment, a micro-robot composed of six rotation modules performed a snake-like gait. The algorithm goal was to optimise its sinusoidal wave parameter to move as fast as possible. The chromosome was composed of 21 bits: seven each for the amplitude, angular velocity, and phase. The results of the algorithm (for which the parameters can be seen in Table 11) are shown in Fig. 23.

Starting with a population of 40 randomly selected chromosomes, the best individual was obtained in generation 172 (111100111111111010000), corresponding to the following values: 85° amplitude, 15 rad/s angular velocity, 1.97 rad phase.

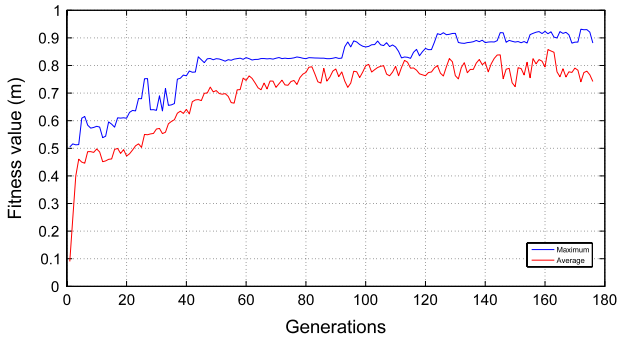


Fig. 23. Results of GA algorithm in open air (parameter optimisation).

Table 11
GA experiment PO 1: Open air.

Parameter	Value
Population	40
Gene type	Bit
Chromosome	21 genes
Generations	175
Crossover prob	0.7
Mutation prob	0.005
Fitness function	Distance covered
Terrain	Open air

8.4. Real experiments

In this section two experiments made with the real robot are presented. In the first one (Fig. 24), a worm-like configuration is shown, where the microrobot is going forward in a 40 mm diameter pipe at two different slopes, 0° and 30°, and also negotiating an elbow.

In the second experiment (Fig. 25), a snake-like configuration is presented. In the figure the micro-robot is moving inside a 40 mm diameter pipe and is able to negotiate straight stretches and elbows.

9. Conclusions

A control architecture for chained, modular robots composed of heterogeneous modules has been presented. This architecture attempts to fill the gap in the modular robotics field regarding heterogeneous configurations.

The control architecture is structured in three levels. It is similar to a hybrid architecture, and indeed, both share many features; however, the control architecture behaviours can be found in both low- and high-level control layers and not only in the reactive layer.

The lower level is entirely based on behaviour, and it includes behaviours related to the module, such as reactive behaviours that take care of the health of the module, walking behaviours in charge of the movement of the robot and perceptual behaviours in charge of gathering information about the module and its environment.

Fig. 25. Negotiating an elbow inside a 36 mm diameter pipe.

On the contrary, the higher level has two main parts. One is also behaviour-based, composed of behaviours related to the whole micro-robot. The other is an inference engine in charge of making decisions based on the information provided by the modules. Behaviours in this layer control the stability of the robot and its movements, allow it to reach goals, help it to avoid obstacles, and to perform other tasks.

Behaviour fusion is a very important feature at both low and high levels. Both coordination and competition are selected depending on the behaviours. It is also worth noting that the coordination of walking behaviours is an independent function, as the combination of heterogeneous drive movements is one of the distinguishing elements of this architecture.

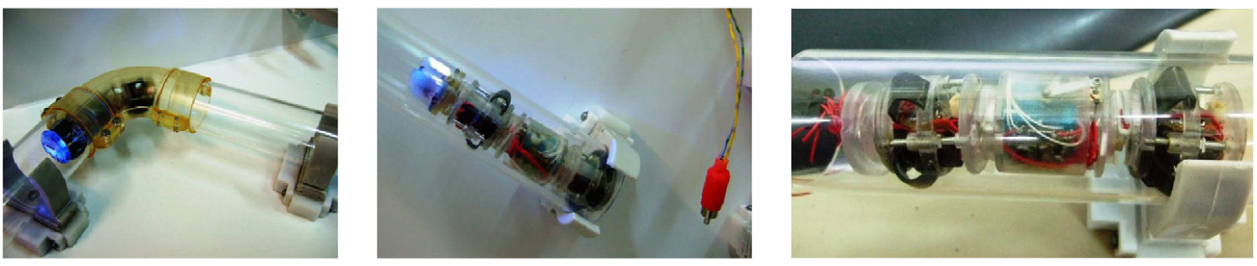
It is important to highlight the role of the intermediate layer, which allows the central control to treat all modules in the same manner because the heterogeneous layer translates its commands into module-specific commands.

An important part of the control architecture is the MDL, a language developed to allow modules to transmit their capabilities to the CC so that the CC can process this information and choose the best configuration and parameters for the micro-robot.

A simulator has been developed to serve as a prototype verification tool and to develop new algorithms. Through this simulator, offline algorithms based on GAs have been developed to optimise the parameter configuration of the robot and to improve its module configuration.

The architecture has been tested in a micro-robot composed of several modules (rotation, extension, support, helicoidal and camera), and some examples of its use have been provided.

In summary, the control architecture described in this paper presents a new solution for control of chained, modular micro-robots composed of several drive units, contributing to the line of research of heterogeneous modular robotics.



(a) Worm module at an elbow. (b) Worm module at 30°. (c) Worm module at 0°.

Fig. 24. Worm module tests.

Acknowledgment: The research leading to these results has received funding from RoboCity2030-II-CM (S2009/DPI-1559), funded by Programas de Actividades I+D en la Comunidad de Madrid and cofunded by Structural Funds os the EU.

References

[1] A. Brunete, M. Hernando, E. Gambao, Modular multiconfigurable architecture for low diameter pipe inspection microrobots, in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA, Barcelona, Spain.

[2] M. Yim, D. Duff, K. Roufas, Evolution of polybot: a modular reconfigurable robot, in: 2001 COE/Super-Mechano-Systems Workshop, Tokyo, Japan.

[3] H. Kurokawa, K. Tomita, A. Kamimura, E. Yoshida, S. Kokaji, S. Murata, Distributed self-reconfiguration control of modular robot m-tran, in: Proceedings of the 2005 IEEE International Conference on Mechatronics and Automation, vol. 1, pp. 254–259.

[4] Y. Zhang, K.D. Roufas, M. Yim, Software architecture for modular self-reconfigurable robots, in: Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS.

[5] W.-M. Shen, Y. Lu, P. Will, Hormone-based control for self-reconfigurable robots, in: Proceedings of the 2000 International Conference on Autonomous Agents, Barcelona, Spain.

[6] V. Zykov, E. Mytilinaios, B. Adams, H. Lipson, Self-reproducing machines, Nature 435 (2005) 163–164.

[7] W.-M. Shen, M. Krivokon, H. Chiu, J. Everist, M. Rubenstein, J. Venkatesh, Multimode locomotion via superbot reconfigurable robots, Autonomous Robots 20 (2006) 165–177.

[8] M. Jorgensen, E. Ostergaard, H. Lund, Modular atron: modules for a self-reconfigurable robot, in: Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, Japan.

[9] M. Yim, New locomotion gaits, in: Proceedings of the 1994 IEEE International Conference on Robotics and Automation, pp. 2508–2514.

[10] C. Unsal, P.K. Khosla, Mechatronic design of a modular self-reconfigurable robotics system, in: Proceeding of the 2000 IEEE International Conference on Intelligent Robots and Systems, pp. 1742–1747.

[11] K. Kotay, D. Rus, M. Vona, C. McGray, The self-reconfiguring robotic molecule, in: Proceedings of the 1998 IEEE International Conference on Robotics and Automation, pp. 424–431.

[12] K. Kotay, D. Rus, Efficient locomotion for a self-reconfiguring robot, in: Proc. of 2005 IEEE International Conference on Robotics and Automation, ICRA, Barcelona, Spain.

[13] M. Rubenstein, W.-M. Shen, Scalable self-assembly and self-repair in a collective of robots, in: Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'09, IEEE Press, Piscataway, NJ, USA, 2009, pp. 1484–1489.

[14] M. Yim, D. Duff, K. Roufas, Polybot: a modular reconfigurable robot, in: Proceedings of the 2000 IEEE International Conference on Robotics and Automation, pp. 514–520.

[15] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, G.S. Chirikjian, Modular self-reconfigurable robot systems [grand challenges of robotics], IEEE Robotics and Automation Magazine 14 (2007) 43–52.

[16] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, S. Kokaji, M-tran: self-reconfigurable modular robotic system, IEEE/ASME Transactions on Mechatronics 7 (2002) 431–441.

[17] S. Murata, K. Kakomura, H. Kurokawa, Toward a scalable modular robotic system—navigation, docking, and integration of m-tran, IEEE Robotics and Automation Magazine 14–4 (2008) 56–63.

[18] R.C. Arkin, Behavior-Based Robotics, MIT Press, 1998.

[19] M.M. Mataric, Behavior-based control: main properties and implications, in: Proceedings of the 1992 IEEE International Conference on Robotics and Autonomation, Workshop on Architectures for Intelligent Control Systems.

[20] R. Arkin, Motor schema based navigation for a mobile robot: an approach to programming by behavior, in: Proceedings of the 1987 IEEE International Conference on Robotics and Automation, vol. 4, pp. 264–271.

[21] P. Maes, Situated agents can have goals, in: P. Maes (Ed.), Designing Autonomous Agents, MIT Press, 1990, pp. 49–70.

[22] J.K. Rosenblatt, Damn: a distributed architecture for mobile navigation, in: AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents, AAAI Press, Menlo Park, CA, 1995.

[23] P. Pirjanian, T. Huntsberger, P. Schenker, Development of campout and its further applications to planetary rover operations: a multirobot control architecture, in: Proceedings of the 2001 SPIE Sensor Fusion and Decentralized Control in Robotic Systems.

[24] P. Pirjanian, T.L. Huntsberger, A. Trebi-ollennu, H. Aghazarian, H. Das, S.S. Joshi, P.S. Schenker, Campout: a control architecture for multi-robot planetary outposts, in: Proc. SPIE Conf. Sensor Fusion and Decentralized Control in Robotic Systems III, pp. 221–230.

[25] C. Armbrust, L. Kiekbusch, K. Berns, Using behaviour activity sequences for motion generation and situation recognition, in: Proceedings of the 2011 International Conference on Informatics in Control, Automation and Robotics, ICINCO, Noordwijkerhout, The Netherlands, pp. 120–127.

[26] M. Proetzsch, T. Luksch, K. Berns, Development of complex robotic systems using the behavior-based control architecture iB2C, Robotics and Autonomous Systems 58 (2010) 46–67.

[29] M.J. Mataric, Interaction and intelligent behavior, Ph.D. Thesis, Massachusetts configurable chained micro-robot for exploration of small cavities, Automation in Construction Magazine 21 (2011) 184–198.

[28] A. Brunete, J. Torres, M. Hernando, E. Gambao, A proposal for a multi-drive heterogeneous modular pipe-inspection micro-robots, International Journal of Information Acquisition (IJIA) 5 (2008) 111–126.

[29] M.J. Mataric, Interaction and intelligent behavior, Ph.D. Thesis, Massachusetts Institute of Technology, MIT, 1994.

[30] A. Brunete, M. Hernando, E. Gambao, J. Torres, A. Castro-Gonzalez, MDL: a module description language for chained heterogeneous modular robots, in: 2011 IEEE International Conference on Robotics and Biomimetics, ROBIO, pp. 2706–2711.

[31] H. Kurokawa, A. Kamimura, E. Yoshida, K. Tomita, S. Kokaji, S. Murata, M-tran II: metamorphosis from a four-legged walker to a caterpillar, in: Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, 2003, pp. 2454–2459.

[32] S. Murata, H. Kurokawa, Self-reconfigurable robots, IEEE Robotics and Automation Magazine 14 (2007) 71–78.

[33] J. Conradt, P. Varshavskaya, Distributed central pattern generator control for a serpentine robot, in: Proceedings of the 2003 International Conference on Artificial Neural Networks, ICANN, Istanbul, Turkey, pp. 338–341.

[34] K. Dowling, Limbless locomotion: learning to crawl, in: Proceedings of the 1999 IEEE International Conference on Robotics and Automation, vol. 4, pp. 3001–3006.

[35] H. Schempf, E. Mutschler, B.C.S. Boehmke, Boa II and pipetaz: robotic pipe-asbestos insulation abatement systems, in: Proceedings of the 1997 IEEE International Conference on Robotics and Automation, vol. 1, pp. 52–59.

[36] J. Gonzalez, H. Zhang, E. Boemo, J. Zhang, Locomotion of a modular robot with eight pitch-yaw-connecting modules, in: 2006 International Conference on Climbing and Walking Robots.

[37] J. Gray, H. Lissmann, The kinetics of locomotion of the grass-snake, Journal of Experimental Biology 26 (1950) 354–367.

[38] M. Sato, M. Fukaya, T. Iwasaki, Serpentine locomotion with robotic snakes, IEEE Control Systems Magazine 22 (2002) 64–81.

[39] A. Brunete, Design and control of intelligent heterogeneous multi-configurable chained microrobotic modular systems, Ph.D. Thesis, Universidad Politecnica de Madrid, 2010.