STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LIII**, Number 1, 2008

ROBOSLANG – CONCEPT OF AN EXPERIMENTAL LANGUAGE

OVIDIU SERBAN

Abstract. There are very few platforms trying to unify the programming process of robots and none of them is concerned with building an extensible language that would allow anyone to have access to the hardware. RoboSlang's main goal is to allow programmers to build easily new modules without needing any hardware programming knowledge and their concern should be to create better algorithms for some well-known problems.

Keywords: Robots, Programming on embedded device, Programming languages, Communication

1. Introduction

Programming on different types of platforms (such as different hardware devices and operating systems) was always a difficult task and when trying to program non standard devices or mobile systems without a proper language or platform can be a very a very dangerous approach. Middleware platforms [12] were developed to unify the developing process, but they are only the starting point for the programming process and not all the problems were being solved. There are some problems that you may need to solve, even if you program on top of a Middleware platform, such as unifying the data feeds(communications and sensors data) or planning tasks in a distributed way.

This paper presents the concept of an experimental language and the way it could be implemented using some of the programming languages (such as Java, C, C++ or SQL). It is trying to answer some usual questions about robots communication and solve some of the basic problems of exchanging data. There are some projects oriented on the intelligent side of the robots, trying to search for a perfect algorithm that solves a problem, but there are many communication issues and hardware problems that should be solved in a different manner. RoboSlang is an entry point for every other project that

Received by the editors: October 9, 2007.

2000 *Mathematics Subject Classification.* 68T05, 91E45.

1998 *CR Categories and Descriptors.* I.2.6 [**Learning**]: – *Concept learning.*

intends to do something intelligent with some hardware and it should not be focused on the platform programming issues.

Also there should be a transparent way of exchanging information and the intelligent devices should have a platform ready for programming and for further extension. RoboSlang is trying to offer a solution for this issue. Many industrial project desire to achieve communication between the production machines, but they are, in most of the cases not able to do so. What if there existed a platform able to do these tasks such that the main concern of the industrial programmers would be a better algorithm to improve the production and not a hardware issue for some component ?

This platform is designed to be scalable by the ability to add as many dialects as needed. So if you need a domain specific dialect you will not need to implement all of the transmission mechanism from its base point, you should only extend RoboSlang for what you need.

There are many hardware incompatibilities and so many different types of architectures that only a transparent layer for every kind of communication should solve the problem, so RoboSlang proposes a unified way to transmit messages between devices.

The paper is organized as follows: The motivation and the real problems that should be handled when trying to program such platform are described in Section 1. Section 2 contains the actual description of the language. The two main dialects of RoboSlang are presented in Sections 3 and 4. Section 5 contains a few of the implementation problems and some programming suggestions. Section 6 concludes the paper.

2. Motivation of the implementation of such a platform

2.1. **What would robots say to each other?** The robots can exchange information like sensors data or positioning diagrams. This information has a huge potential in a network considering the fact that many robots and devices do not have powerful hardware or sensors to work with. An entire theory about distributed computing [13] was developed, but there is no standard protocol of communication between two devices that are able to exchange information.

Not only robots and smart devices need to communicate results, also the humans would like to communicate with the robots in a "human-like" language and if they are capable of doing that, why would not be any other smart living creature capable of communicating with a robot or a smart device.

Imagine a situation when a cat would like to ask your robot to clean all the mess inside its sleeping place. Why would not that be possible? If you consider that a cat is capable of thinking and expressing its thoughts then there should not be a problem to exchange the information.

2.2. **Hardware and other communication problems.** Hardware compatibilities are a real issue. There are so many possibilities for exchanging information that even trying to keep a collection with all the data for them is a problem. The platform will try to offer support to any further implementations so that the hardware level will be transparent.

For example if you consider different types of wireless protocols [2], you will see that some of them are incompatible even if they are supported by the same device and you do not have to mention problems related to incompatible devices that support the same protocols. Also related to wireless problems there are some issues for data transmission in a hostile environment. We should consider that our device should be able to transmit data even if the environment is unfriendly.

Suppose that all the hardware problems are not so difficult to solve, there are also some software and communication problems. For instance two platforms support different type of data, one is able to deal with big quantity of information and other is very slow and cannot process such information. What can we do ? Fortunately the network research area gives us some answers, but there are some problems that still remain unsolved.

Another problem is related to listening process, when you search for another device to exchange information. If you do this from time to time, it is most probable to miss some of the partners. If you do this permanently then you could waste valuable energy that could be used to perform other tasks. In the real life there are some situations when you establish a synchronization moment with another partner, but in robotic like situations this is not a good way to discover other devices. You should be able to discover and exchange information as much as you can and as fast as you can, not to wait for a certain moment.

Therefore the hardware and communication problem still remains open, but in this area a lot of research is done so RoboSlang would be able to apply certain fixes as fast are discovered by other researchers. The main concern on the developing process should be the platform and not to solve certain hardware or software problems.

3. RoboSlang

This section contains the description of the proposed language (called RoboSlang).The language itself should be as light as possible. It must contain a discovery procedure, that should also involve a handshake mechanism and a procedure for choosing a dialect. The entire process can be described in a step by step manner :

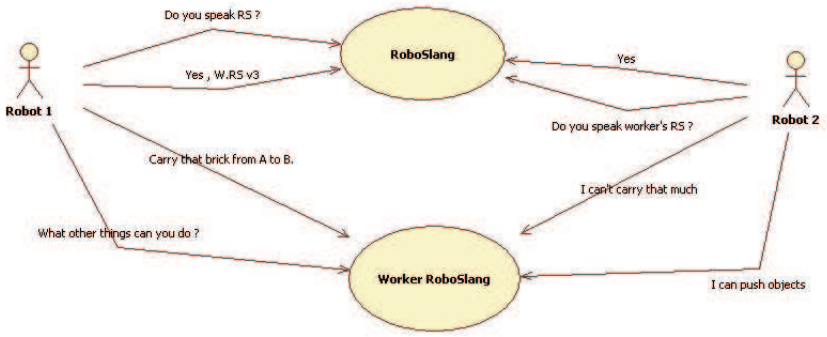

FIGURE 1. The handshake and dialect choosing mechanism

- Step 1 (Discovery procedure): Robots are inside the "visible" area of each other. They recognize some hardware signatures (such as wireless signal [3, 4] or an available Bluetooth service [1]) and try to establish a communication channel.
- Step 2 (Handshake mechanism): The handshake procedure consists of exchanging of identification data. This information contains a key (unsigned nonzero integer on 16 bits) that uniquely identifies each device. The uniqueness of the key must be assured by the developer of the platform using a key generator application. In the first version of the system the key will be assign manually at the moment of assigning name to devices, but further improvement may consist of an automatically way of assigning the keys. When a device receives identification data from other devices it must send back a confirmation message. If no confirmation message is received the identification data will be resent. The handshake process ends when each of the devices has received and accepted the identification data of the other devices. Once the communication channel between two devices is set, each of them will assign a local key to the session for the simple fact that between two devices could exist several sessions.
- Step 3 (Dialect choosing): Each robot or device has a list with some known dialects. The basic information about a dialect is:
  - the name of the dialect: 6[1] printable ASCII characters[2];
  - the version of the dialect: 8 bits integer.

[1]There are 94 characters and there 4.2 * 1015 combinations that can be used for choosing the names. The names are given in a fixed size, so if you want to specify a 3 character dialect, then you should put 3 spaces after or before the name.

[2]Printable ASCII characters are numbered from 33 to 126 in decimal notation

More information about the structure of the dialects will be discussed later in this section.

The handshake and dialect choosing mechanism is described in the Figure 1.

After choosing the dialect the two robots will decide what information should change. The structure of the exchange package remains the same for every dialect.

The package structure is as follows:

- 16 bits - the unique identification key of the destination device or 0 for a broadcast. The key is written in fixed 16 bits form.
- 32 bits - representing the length of the data transmitted in the package(number of bytes)
- 1 bit - the checksum bit of an error detection algorithm for the transmitted data the actual data

The proposed checksum algorithm is a simple one revealed by the following function:

$$checksum(x) = XOR_{i=1,..n} x_i, \tag{1}$$

where x_i is the bit representation of x, n is the length of x and XOR is the bits operator applied on multiple bits.

There are some special packages transmitted by the RoboSlang handshake and dialect negotiation protocol. For a better understanding of the concepts of the language and dialects I will use a hex base notation for all the numbers. The first of the series is the presentation package.

It has the following structure:

$$0000h \qquad 000000000000000Fh \qquad checksum(key) \qquad key$$

And the response should be:

$$key \qquad 000000000000000Fh \qquad checksum(key) \qquad key$$

After this procedure the robots will ask each other the interested dialect list:

$$key \qquad 0000000000000040h \qquad checksum(command) \qquad command$$

command = { transmission error == 0h
retransmit data == 1h
do you know == 3h dialect version
yes == 7h
no == 15h
begin == 31h
end == 63h }

This codes should be enough for device to establish communication and begin the whole exchange process. The RoboSlang language should be kept as light as possible because it should be used only for initiating the data exchange process.

The codes are chosen to solve the transmission error problem in a easy way, so if you consider the bit representation of these commands you will see that each of it begins with a variable number of zeros followed by a variable number of ones. When receiving a command, a simple check should be made to verify if the data conforms with the specification. If not both error commands (transmission error and retransmit data) will be send.

4. CommonRoboSlang (CRS)

4.1. **The tasks - breaking into small pieces.** CommonRoboSlang is a dialect of RoboSlang and its main goal is to handle the tasks in a uniform manner. Imagine a huge task that a single robot could not handle, but a lot of them can. So you need a proper language to exchange the information about the task and the way it can be divided. Also you would need an algorithm to divide these tasks.

The main principles of CRS are:

- *If you can do a task without any help, do it!*
- *If you can't do a task without any help, then consider the most suitable robot to do it.*

So first of all you would analyze that it is better to split the task and than to do it. A second argument for this should come when there are not any robots available in the area and you need to search them. Also you should consider the transmission time. When all the answers are negative then you should consider splitting as an option.

When you split something you take the task that you can do in an affordable time, but then it comes another question: "How many units of time are affordable¿'. So CRS answers this question in a simple manner. Try to split the task in as many pieces as the known robots are. Also you should consider the task that you are able to perform.

We talked about splitting tasks into small pieces, but what the tasks are? A task is a collection of abstract steps that you should follow to reach at the end. Abstract steps are known algorithms and they cannot be divided into small pieces. For instance an abstract step can be "go from point x to y" and also "climb a mountain". I suggest keeping the steps as small as possible, but if it is necessary you can construct them in a more complex way. These steps are encoded so the transmitted information is as well, as small as possible. The encoding of the steps is chosen by the programmer and it does not have a standard form. If a robot does not know what an abstract step means then it should reject the whole task. The rejection cause can be one of the following: "unknown resource needed", "unknown steps" or "task too big".

The robot should keep a log of the current task so that every time it would know how much time remains.

Nobody should consider that splitting the tasks and describing the abstract steps is an easy job. In fact the intricacy of the problem is equivalent to designing a very complex programming language. The description of the abstract steps is a difficult task and putting them in the task collection can be also very complex. You must decide if to create a task as an abstract step or to split it into small pieces and let the devices decide their flow. In the final stage, when a device is getting and mixing all the results can be also considered a task and could be designed as an abstract step collection.

4.2. **The main commands of CommonRoboSlang.** After switching the conversation to CRS the dialect will have a standard form of question(Q) and answer(A):

Q:	status	
A:	busy	means that the other party does not accept any task for the moment
	idle	this means that the robot is available and ready for any task
	none	is an undefined state, meaning that the robot have some problems or could not serve any task
Q	can you perform [task]	
A:	yes	means that the robot is sure that it can perform the given task
	no	means that the robot is not able to perform the entire task
	possible	means that the robot is not sure about the task so it can be rejected later

The other phrases of the dialect have an imperative form:

- job done [task] [result]
- job rejected [cause]

Task assignment should be considered immediately and the results should be given to the robot that assigned the task. Also when all the sub tasks are done the main robot should finish the task also.

5. HumanRoboSlang (HRS)

This dialect is one of final goals of RoboSlang as smart devices should be able to communicate with humans in a natural manner. There are some problems related to this subject and that include human language semantics, natural language processing [11], the criteria to choose a certain language and the list could continue. If we speak of getting the semantics of a certain language we would see that current research in this area is not very surprising [14]. Unfortunately this subject is far away from human dream that we should be able to talk to robots and they would be able to perform tasks given by us. There is some hope also because there are some algorithms capable of answering and learning from humans, but they are still in research and used only for experimental purposes or fun.

The HumanRoboSlang is proposing o way of communicating only certain task to a robot using an SQL like language [15]. So suppose that every task and its description and steps are described in a database. Your only problem now is to figure out what task should be done at a certain time and you take the ID of the task from the TASKS table and insert it into your REQUESTS table. The device will query the table from time to time and perform the tasks ordered by the priority. I know that this is far from a human to robot language, but is a simple way to communicate and it is very easy and low costing solution to implement.

Once the research in the natural language processing is getting some useful results, then a certain solution for HumanRoboSlang should be found, but until then we will be able to communicate with smart devices thought some pseudo natural languages.

6. Implementation

Regarding to the programming languages, there is no restriction for using one or other. If you consider one language better than other, then search for the RoboSlang architecture implemented in your language and begin the programming process. Also there are some cases when you do not really need a RoboSlang implementation and you only want to send or receive some data from a certain language using a known data exchange method.

The platform will be implemented using Java [7] and C on a Linux [5, 6] platform because of the portability and numerous Linux distributions that can be installed on every kind of platform. RoboSlang is not intending for the moment to build a platform of its own, but in time, if it is considered necessary, it would be considered as an option.

As an approach for the platform implementation we can use network programming after we establish an IP connection with a server. The whole scanning process and wireless network [10] detection can be done by the operating system. Another way of implementing the connection between robots is based on bluetooth programming [9]. It can be easily done in Java and Java ME [8] and it does not need any more platforms. There are a few settings to be done under Linux for using bluetooth, but there are a lot of materials on the Internet describing how to do it.

7. Conclusions and further work

The current researches done by some companies suggested that there isn't a certain platform available on the market that could solve the problem of communication between two intelligent devices. So every product comes with its own firmware that it is able to do some tasks that would be designed for, but nobody is concerning in exchanging results and important data between two or more devices. Even if the industrial usage of the robots is extending, the companies propose non distributed solutions for task management and production improve. There is some research for algorithms to control robotic swarms, but there is not a stable platform to test those algorithms. Swarm OS from IRobot Industries is a platform for swarms, but that operating system is still under research, it would not be available for free and their purpose is to control only their hardware (IRobot swarms) [16]. After performing some research you would see that even is a critical problem there are few companies that are implementing some platforms for robots programming, but none of them is dealing with the robots in an autonomous manner and none of them is proposing a distributed solution and a communication platform. RoboSlang will begin as an Open Source project and will became a solution for some of the companies that would like to have real intelligent devices, because they will be able to exchange data and work as a team.

References

[1] Kenneth C. Cheung, Stephen S. Intille, and Kent Larson : An Inexpensive Bluetooth-Based Indoor Positioning Hack, Massachusetts Institute of Technology Press, 2003
[2] Andrew S. Tanenbaum: Computer Networks 4rd Edition, Prentice Hall ,2003
[3] Halk Gmskaya, and Hseyin Hakkoymaz : WiPoD Wireless Positioning System based on 802.11 WLAN Infrastructure, ENFORMATIKA, 2005

[4] Y. Wang, X. Jia, H.K. Lee : An indoors wireless positioning system based on wireless local area network infrastructure, Satellite Navigation and Positioning Group (SNAP),School of Surveying and Spatial Information Systems ,2005
[5] Daniel Bovet , Marco Cesati: Understanding the Linux Kernel, Third Edition, O'Reilly Media, 2005
[6] Linux kernel docs and FAQ http://www.kernel.org/pub/linux/docs/lkml/, last visited on 05.05.2008
[7] Sun Java API and Docs http://java.sun.com/, last visited on 06.05.2008
[8] Sun Java ME API and References http://java.sun.com/javame/reference/apis.jsp, last visited on 06.05.2008
[9] C. Bala Kumar : Bluetooth Application Programming with the Java API , Morgan Kaufmann Publishers, August 2003
[10] Andreas Lerg : Wireless Networks: LAN and Bluetooth, Data Becker, August 2002
[11] Natural language processing
http://en.wikipedia.org/wiki/Natural_language_processing,
last visited on 08.05.2008
[12] Hans-Arno Jacobsen: Extensible Middleware, Kluwer Academic Publishers, 2004
[13] Jim Farley : Java Distributed Computing, O'Reilly Media, 1998
[14] I. Heim; A. Kratzer : Natural Language Semantics, Springer Netherlands, 2002
[15] Alex Kriegel, Boris M. Trukhnov : SQL Bible,John Wiley & Sons Inc, 2003
[16] IRobot swarms http://www.irobot.com/filelibrary/GIvideos/swarm3%20ad.html, last visited on 12.05.2008

Babeş Bolyai University, Department of Computer Science, 1, M. Kogălniceanu St., Cluj-Napoca, Romania
E-mail address: so29769@scs.ubbcluj.ro