Robocontrol 2012
5th WORKSHOP IN APPLIED ROBOTICS AND AUTOMATION
unesp

A BLOCK PROGRAMMING INTERFACE FOR EDUCATIONAL MOBILE ROBOTS

Antonio Valério Netto[1], *Felipe Antunes Miranda*[1], *Wesley Silva*[1], *Yan Freitas*[1], *José Tadeu Aldrigue*[1]

[1] XBot, São Carlos (SP), Brazil, valerio@xbot.com.br, felipe@xbot.com.br, wesley@xbot.com.br, tadeu@xbot.com.br

Abstract: This paper describes Curumim Program Interface, the software used to control the robot Curumim with block programming. Curumim is a tool for educational robotics, at first commanded by blocks programming, which will be described in this article, but also provides the possibility to expand the horizon of students to other technological fields. Using this, even those who don´t have enough knowledge about program languages can determinate some action to the robot. In order to write the program, it is only necessary to click on the icon in menu and the block will be inserting in program's window. Then a sequence of blocks will be created which can be executed by Curumim. It's possible to change any parameter of the block, delete it and insert a new block between among others already fixed. For expert users, there is the possibility to use C/C++ languages to program Curumim.

Keywords: educational software, robotic education, mobile robot, educational technology, robotics interface program.

1. INTRODUCTION

In order to promote the educational development, teaching and learning of basics logic idea through a robot, we developed the Curumim Kit. This kit, illustrated in Figure 1, includes the Curumim robot, a couple of batteries with charger, the radio for communication between the robot and the computer, camera receiver, besides Curumim Software and their user's guide.

Fig. 1. Curumim Kit

Curumin's Software is the interface between the robot and the user. With this the user will be able to send all the commands required to control Curumim. For this purpose, we present two options: block programming and C/C++.

The first option of programming is formed by a set of blocks to choice and builds an algorithm, then the robot performs the actions planned sending instructions by radio.

Thus, when assembling a program the user will have to think about which blocks to use and in which order "fit" them, as well as use creativity to achieve the intended goal.

The software also has a "translator" of blocks programming for C/C++. For each block included in your program there is a code snippet that will build the equivalent code program in C/C++. This will be available to the user so it's possible to acess and even change or rebuild it. The purpose is take the user to a higher level, since this functionality provides self-learning about C/C++ programming. So, it's also possible to build more elaborate programs for Curumim.

As seen in Figure 2, Curumim robot is composed of some items which facilitate the execution of their tasks. A clamp in front of the robot which works vertically up and down, where the user can put a pen and with his positioning down have marked the trajectory of the robot.

Fig. 2. Curumim's details

There is a camera that provides images of the front of the robot, with which the user can request a Figure or reading a QRCode during the execution of each action. Curumim also has five infrared sensor used to detect obstacles near to the robot;

Besides, a set of three motor designed to move the robot, with swedish wheels, allowing a composition of different movements to the robot [1] [2] [3], which could not be made by conventional wheels. This characterizes the Curumim like an omnidirectional robot. Thus, the locomotion of the robot is shown in Figure 3 below:

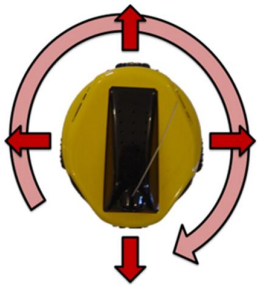

Fig. 3. Curumim's locomotion

2. MODELING AND DESIGN

Order to adapt the needs of the software's project, we used the same procedure established by [4]. First there was the analysis and specification of requirements, raising those necessary to perform the requirements set to the software. Listed are the software requirements using the UML notation [5] (Unified Modeling Language). Also, cases of use were developed, being one generic with the various features of the software illustrated in Figure 4, and others detailed for each case.

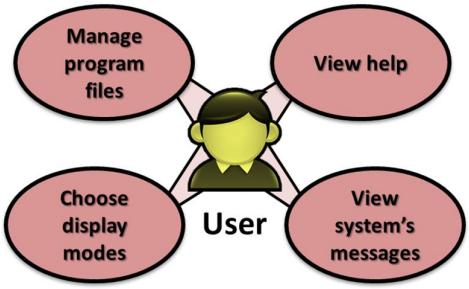

Fig. 4. General use case

During the design and development, was elaborated a model of the software also through UML diagrams. To gradually increase the information and detailing, the design of such tasks was divided into two stages, the physical design and the interface. The first was about modeling, and information related to implementation and the choice of programming language. The second represents information of the interface since it needs a higher degree of detail. Finally various tests were performed according to the needs of software.

3. SOFTWARE ARCHITECTURE

The software architecture is divided into three different modules: Interface, Basic Structure and Communication. This division was proposed order for the internal structure and communication stayed separate from the interface, not only to facilitate code comprehension, but also interface changes, as a development of another version to Linux/Unix, since the only part to be reviewed would be the interface, keeping the structure. The interface module is accountable for communication among software and the user. Through this module, there are the options to build the program (blocks or C/C++) for command Curumim.

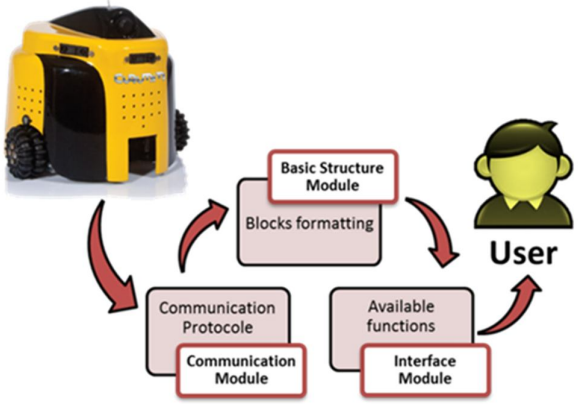

Fig. 5. Curumim's software architecture overview

The basic structure module is responsible for managing the information sent to the robot related to tasks defined by the user. The communication module is the one that performs the interaction between the robot and Curumim Software. It's through this module that is sent tasks to the robot, like the software were the voice command for the activities to be performed by the robot. In a general overview, shown in Figure 5, the activities of the robot are determined by the commands attributed to basic structure module through interface module. Using the commands of the protocol into the communication module, it sends commands to the robot, so that "understands" and execute the programed task.

4. BLOCK PROGRAMMING

The software is the Curumim's environment programming [6], where the user determines the actions for the robot. It has a friendly interface [7] [8], and is divided into five sections, as illustrated in Figure 6.

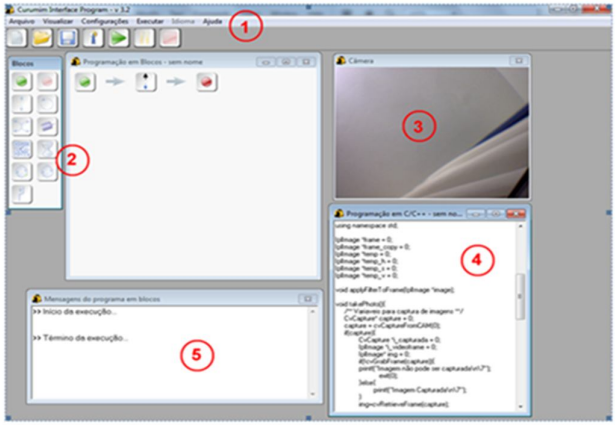

Fig. 6. Curumim's software interface

1. Toolbar and Menu: The menu is formed by the general software's options available, since managing the program generated by the user, as long settings, views and help to better use of the software. Some of these options also have the combination of keys for direct access. The toolbar has some of the options in the menu in order to act as quick shortcut buttons, illustrated with figures that identify them easier.

2. Block Programming Area: mainly targeted to beginner users who have never programmed a robot and do not know a specific language to control the robot. There are a number of blocks in the menu with the actions ready for the robot, where is intuitive using each block and related commands for controlling Curumim. Each block has an illustration that facilitates their recognition. For every selection of a block, it is inserted in the window next to the programming menu.

3. Image Area: has a window that displays the image captured by the robot's camera. The user can request to show it or close whenever you want, except during the execution of the program block when their display is done automatically by the software.

4. Area Programming C/C++: with the possibility of building a program in C/C++ for the robot, there is the opportunity for progress of the novice user, where you can see the translation of the controls carried out by programming blocks in C/C++ and then change it. In addition, there is the option for more advanced programming for the robot.

5. Message Area: this window displays the information about the result of the compilation of the algorithm, as well details of each block running by the robot. This space also provided a feedback of actions taken.

Both menu and toolbar options are useful for block programming, besides display the image captured by the robot's camera and C/C++ code.

The menu is always visible, but sometimes part of this options may be disabled, according to each situation, while running the software, preventing the user from making any improper request.

The menu options are distributed in groups as shown in Figure 7:

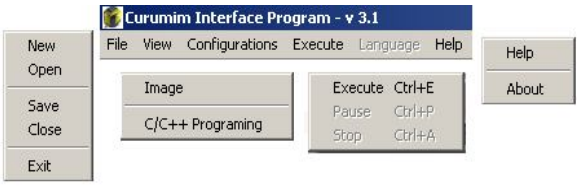

Fig. 7. Curumim's software menu

File: Have options for the management of user programs, like creating a new file, opens, save or close a program, besides the option to leave the software Curumim;

View: have the options to show the camera's image and the "translation" of blocks programming for C/C++;

Configuration: settings of the communication, using this those option to set robot's ID and test the communication, besides other options about the compiler and some camera's functions.

Execute: it has the options to compile a C/C++, since there is an open and execute a block program or C/C++ (in the case of a C/C++ program it's necessary that the program is already compiled for can be executed). It´s also possible to pause and stop the execution program, since there is a block program running;

Language: where the user can choose the language of the software. Curumim's software is available in Portuguese, English and Spanish.

Help: has a help option, which provides information about how to use the software and some tips about block programming.

Fig. 8. Software's toolbar

Among these options, those available for quick access in the Toolbar are illustrated in Figure 8.

4.1. Configurations

The configuration window is where you set parameters to be used by the software, especially in communication. Have two tabs: Communication and camera, as shown in Figure 9.

This tab has fields for filling out the appropriate serial port and source and destination addresses to be used during communication with the robot. Initially the user must fill out the serial port when is connected to radio base. If the serial port is not available, simply click the Detect Doors, the software can identify it and the user then select it. After setting the serial port, the user must fill in the source address (identification of radio) and target identification (robot) and define them as well. Then the user should be able to complete your changes, or perform a verification of communication with the robot through the Test button.

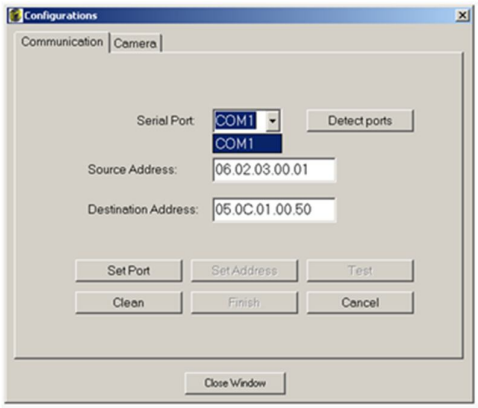

Fig. 9. Configuration window

In camera tab the user determines the default name for the figure files that may be generated by the robot during the execution of their movements. The appointment of photo files is done by prefixing the default name given by the user with a common numbering controlled by software. The user can also determine the locations where such files are stored.

4.2. Block Programming Area

The area of block programming is the highlight of the software, because it was designed especially to make the robot control a simple situation, intuitive and independent of any knowledge in a particular programming language.

In the figure, which aims to be identified immediately, the blocks contained in the menu still have hints that help users identify the name of each block, when the mouse passed over them. The sleep block is a block that makes the robot stand still, without performing any action for a while.

The blocks FOR Loop, WHILE Loop and Conditional IF are conditional blocks that include other blocks, and the evaluation of the condition of the conditional block in question. In Figure 10 there is each block identified by name. The condition for each conditional statement is evaluated:

FOR loop: while the number of times determined for its repetition is not exceeded, the blocks involved by the FOR Loop are evaluated and executed;

WHILE loop: while the combined results of the sensors the robot selected is true and the maximum number of times determined for its repetition is not exceeded, the blocks involved by the WHILE Loop are evaluated and executed;

Conditional IF: if for the selected sensors, the combination is true, the blocks involved by the block IF are evaluated and executed.

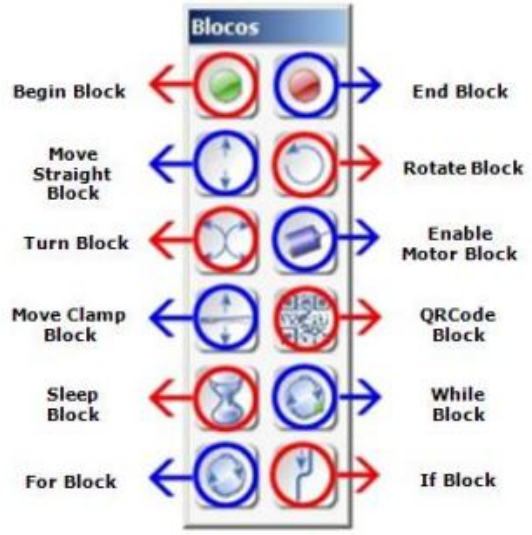

Fig. 10. Functions related to each block

When any blocks are insert, except the delimiters (Begin or End), a window with the block information is opened with default values that can be easily changed by the user. Moreover, it has completed verification of the values for the user do not perform any wrong filling. As an example, some of these windows are shown below. The move straight block window, shown in Figure 11, has the following parameters:

Fig. 11. Straight move configuration window

Direction: indicates the direction to be followed by the robot (front or back)

Speed: indicates the speed level to be used by the robot (1: low, 2: medium, 3: high);

Distance: defines the distance to be traveled by the robot;

Image Resource: offers the option of use or not of use image capture.

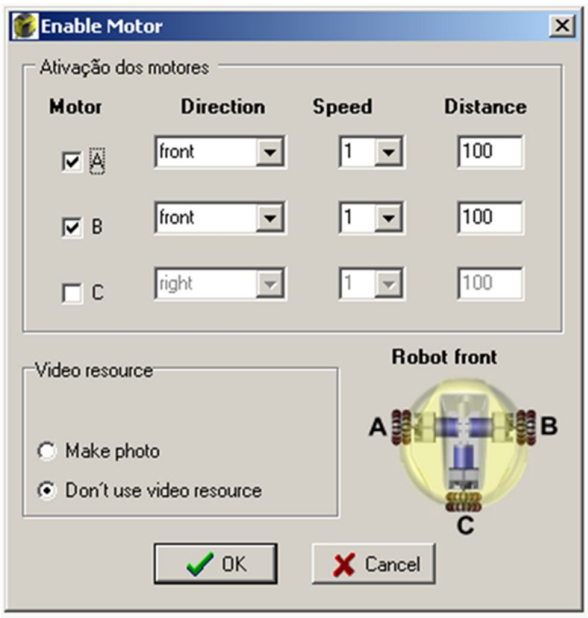

Fig. 12. Enable motor configuration window

The enable motor block window, illustrated in Figure 12, has the same parameters of the move straight block, but the control of these parameters is done individually for each of the three robot's wheels which each include a motor. This, together with the choice of engines for use allows the creation of movement for each combination of different parameters.

However, the WHILE block window shown in Figure 13 provides an area for the construction of a logical expression.

The user can use the results of the sensors combining them by operators NOT, AND, OR, NAND and XOR, besides having the possibility of using parentheses.

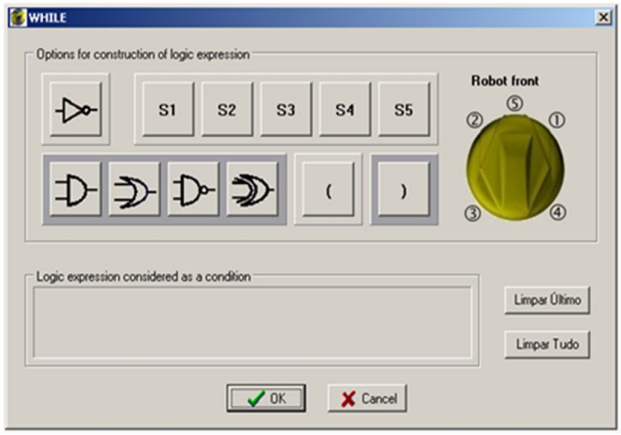

Fig. 13. While block configuration

This expression is evaluated during the execution of the program and if it was found to be true, the blocks that are involved in this conditional will be executed. Otherwise, they are ignored.

As the user is selected blocks in the program, they are included in the blocks programming window. Furthermore, the user can still change the values of the parameters of their blocks through the windows already mentioned and / or removing a block of their program.

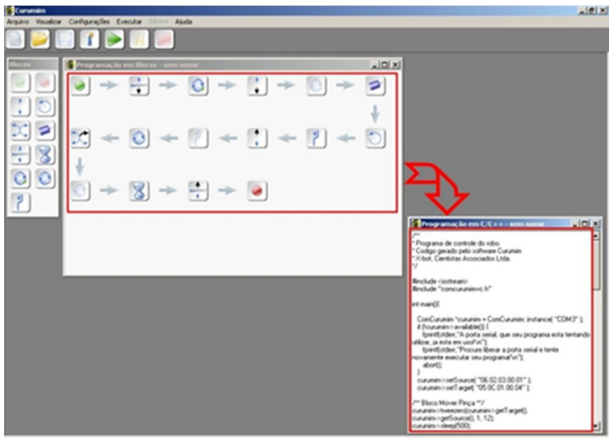

Fig. 14. Example of a block program and translation to C/C+ +

With the exception of the insertion sequence blocks, the other features mentioned are available in the popup menu, differentiated according to each block: for the begin block is "Insert a block after", to the motion blocks and conditional are "Change the values of the parameters of the block", "Delete the block", Insert a block before "and insert a block after" and for end block are "Delete block" and "Insert a block before".

By adding to the program block by block, the user determines the options for the robot and in which order these actions must be performed.

In Figure 14, there is an example of program blocks that can be built into the software. This program was developed by request of seven steps to be completed by the robot:

1. Put Down the clamp;
2. While there aren't obstacles behind the robot, ride 30cm back, with average speed. Consider this condition only valid for up to four times its implementation;
3. Rotate 90 degrees to the left, then take a Figure;
4. If there any obstacles to the right, go ahead;
5. Repeat for 4 times the curve to the right front;
6. Wait for 2 seconds before continuing the next step;
7. Finally, put the clamp up.

Although Figure 14 doesn't show directly the values of changed parameters in a request, the user can check each one sending a hint with the information.

If somebody want to change their values, its possible delete or insert new blocks, just click the right mouse button on the desired block and requesting the required action.

4.3. Running a block program

When the user requests the execution of the block program, the software opens a window just below the window programming block which indicates the actions taken. This allows the user to follow one by one of the actions done by the robot, as the Figure 15.

Also during the block program execution, it's possible to pause and then request the return to the program, or stop the execution.

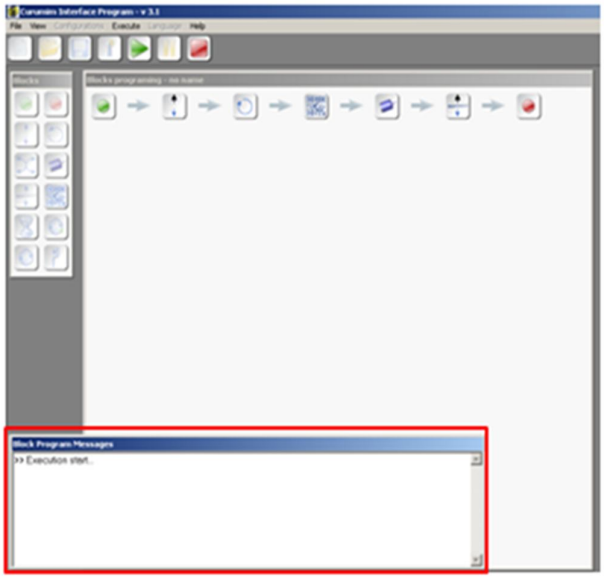

Fig. 15. Running window

4.4. Translating into C/C++ a block program

The user can request the "translation" of the program in block C/C++. This makes learning the syntax of the language more accessible.

Even for those who already understand the programming logic, it's possible to check the methods which send commands to the robots, and change their parameters directly. Thus, the user can include any other more complex structures in the created algorithm. The compilation of this program may be made in the software itself, simply by setting one of two compilers required by the software. It is also possible to run this program, already compiled.

However, if the user prefers to edit it in another environment more specific of the language, just save this program, open it in the desired environment and import their library of robot's commands. Figure 16 illustrates an example an algorithm project to control Curumim with Microsoft Visual Studio:

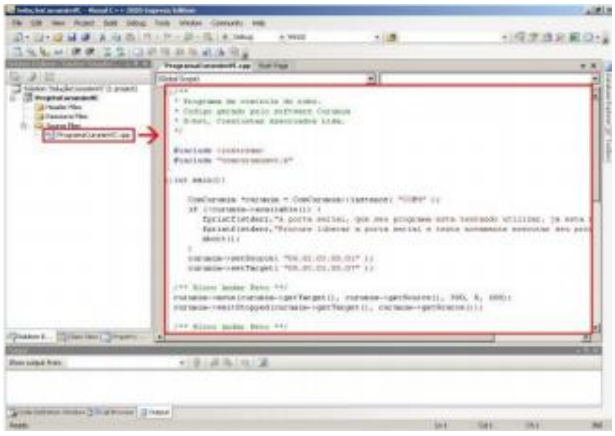

Fig. 16. Programming Curumim with Microsoft® Visual Studio

Below, there is a simple example of an algorithm ready to use with the robot, performing a rotation.

```
  //basic input-output libraries
  #include <iostream>
  #include <fstream>

  //Curumim's library
  #include "comcurumimvc.h"

  //main program
  int main(){

  //creating an instance of ComCurumim
already determining the port where is
connect the radio
  ComCurumim *curumim =
ComCurumim::instance( "COM7" );

  //source address setting
  curumim->setSource( "06.02.03.00.01"
);
  curumim->setTarget( "05.0C.01.00.07"
);

  //moving Curumim, in this case we use
as example the rotate function
  curumim->rotate(curumim-
>getTarget(),curumim->getSource(),-45,
0);
  curumim->waitStopped(curumim-
>getTarget(),curumim->getSource());

  // finishing the main function
  system("PAUSE");
  return 0;.
  }
```

5. FUTURE WORKS

After some opportunities where we could test the ability of Curumim as a teaching tool, we are preparing Curumim for make easier some hardware customizations of the robot. Curumim proved to be useful in classes on Boolean logic, programming concepts, algorithms, C/C++ and image analysis.

Meanwhile, the current structure difficult a further approach to electronic aspects of the robot. Thus, among the possible modifications, we planned the possibility of integrating the core of the robot - currently controlled by a MSP480 - with other microcontrollers, such as Arduino. This fact would increase the use of the platform as a way of teaching embedded systems, electronics, among other topics related to robot's hardware.

6. CONCLUSION

Our society has experienced a growing technological evolution, which brings the necessity for new practices of teaching, learning and access to knowledge. Thus, with the proposal to allow more interaction with technology tools, enabling an increase in the retention of knowledge, currently there is a high demand for Robotics Education [4]. However, there are few studies about how it all began [9]. In education, the robotics is shown as a tool that enhances the multidisciplinary approach, combining a lot of technological topics.

With that, students have an environment where they can handle, create, program and, by this ludic practice, develop the logical reasoning, which is really important in many areas of knowledge. It's notable that robotics allied to education proposes a greater interaction between teacher and student, allowing both to experiment, by searching, a constant learning.

Thinking about that, Curumim's software was designed to be a simple and friendly interface, in order to reach as many users as possible. It also provides the evolution of the

novice users, which in a first contact only would program the robot through blocks and then could to start using the language C/C++. This objective is expected because it is believed the interest in learning programming is easily motivated through the robot Curumim.

Is worth emphasizing that with the robot the user is not limited to study only programming languages, but also to all other technologies included in robotics, such as mechanics, electronics, sensing, among others. Thus, we believe we have developed a highly effective way of learning, dynamic and practical.

ACKNOWLEDGMENTS

We would like to thank all development team of XBot and other members of the company, as well as those who have contributed to the company's activities, such as Anderson, Mauro, Celso, Roberto, Valdir and all those who used to be involved at some stage of Curumim's project. We thank CNPq and FAPESP for supporting this research too. Finally, we also thank all customers who have chosen to know the robot, and help in the continuous improvement of our product.

REFERENCES

[1] Siegwart, R., Nourbakhsh, I. R., (2004). Introduction to Autonomous Mobile Robots. The MIT Press.

[2] Nehmzow, U., (2003). Introduction to Autohnomous Mobile Robots. 2nd Edition, Springer.

[3] Braunl, T., (2003). Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems. 1st Edition, Springer.

[4] Francisco, S. D. (2006). Sistema Integrado de Robótica para as áreas de Educação, Pesquisa e Entretenimento. Cientistas Associados Desenvolvimento Tecnológico Ltda. Relatório técnico de pesquisa apresentado à FAPESP referente à bolsa de trabalho nível IV do projeto PIPE-FAPESP Fase 2 (12 meses) – Ano I.

[5] Furlan, J. D. (1998). Modelagem de Objetos através da UML: The Unified Modeling Language. Makron Books

[6] Rocha, H. V., Baranauskas M. C. C. (2000). Design e Avaliação de Interface Humano-Computador. XII Escola de Computação, São Paulo – SP.

[7] Konzen, I. M. G., Cruz, M. E. J. K. (2007). Kit de Robótica Educativa: desenvolvimento de aplicação metodológica. II Escola Regional de Licenciatura em Computação, Universidade Federal de Santa Cruz do Sul, Santa Cruz do Sul – RS.

[8] Pretto, N,. Costa Pinto, C., (2006). Tecnologias e novas educações, Revista Brasileira de Educação.

[9] Melo, C. K. S., Azoubel, M. A., Padilha, A. S. P (2009). A metodologia da robótica no ensino fundamental: o que dizem os professores e alunos? III Simpósio Nacional ABCiber, São Paulo.