ROBOTIC TOOL WITH SCRATCH LANGUAGE

Marco Tálio Chella

1 Universidade Federal de Sergipe (UFS), Aracaju, Brasil, chella@ufs.br

Abstract: Robotics as an educational tool has been used at all ages levels to motivate interest in further study in science and technology. To create activities with robots in educational context are needed suitable software and hardware tools. We present the development of a robotic control board and the integration with Scratch language in a system that allows building and programming a robot.

Keywords: robotic, educational, tool.

1. INTRODUCTION

Robots have a long history as educational tools. Used at all ages levels, they are generally used to motivate student interest in further study in science and technology [1]-[3]. Activities with robots in the classroom usually involve defining a problem to be solved, building and programming the robot, and finally evaluating the result. If necessary, changes can be made after evaluation to meet the specified in the initial project.

Working with robotics in an educational context can give students:
• Hands-on experience with real problems.
• Experience in understanding and implementing robotics principles from primary research literature.
• Confidence in their ability.
• Help to develop teamwork skills.

In a common approach students develop the control board under supervision of the computer instructors and the system software to communicate with hardware. This approach tends to focus on technical aspects and not in the project itself and its implementation. The result is to devote more time to implementation and debugging, and less to developing solutions.

Another aspect to consider is that a control board, even simple devices made out of cheap and readily available electrical components (switches, sensors, solenoids, motors) is hard to build, instructor and students need to have a background in electronics, circuit design or electrical engineering and embedded systems. This is a competence that most computer scientists and human computer interaction specialists lack.

Commercially available devices can be another alternative but most may have no published application-programming interface (API). As a result, an outsider cannot program them unless the device is ‘hacked’ or reverse-engineered.

To solve this drawback we developed a control board called ERB (Educational Robotic Board) with resources to control actuators and capture analog and digital data from sensors. To allow a PC computer to communicate with an ERB, a protocol was developed in a way that any software development tool that can access hardware like Python, DotNet, Java, can be used to control ERB.

Dialects of Logo language like Imagine [4], Microworl[5]s and MSWLogo[6] have been used in computer literacy since 80’s decade. Recently Scratch Language was introduced by the Lifelong Kindergarten Group at the MIT Media Lab [7].

Scratch uses a building-block metaphor, in which students build programs by connecting graphical blocks that look like pieces of a jigsaw puzzle.

Scratch allows students to program with a mouse: programmatic constructs are represented as puzzle pieces that only fit together if “syntactically” appropriate. We consider that this environment allows students not only to master programmatic constructs before syntax but also to focus on problems of logic before syntax.

Scratch offers a rich environment to develop media and networked applications. We consider that this environment can expand the potential of Scratch if it allows the integration between physical objects and media contents.

The robotic extension proposed in this work intends to offer resources to explore curricular contents by the construction and sharing of physical objects.

Activities in workshops with teachers and students suggest that developing applications that provide integration between physical objects like robots with sensors, motors, and multimedia contents (sounds, videos) is suitable and can stimulate and motivate students in the learning process. The intended public is 8th graders or above but Scratch has been used in introductory computer science courses [8],[9].

2. PURPOSE

Robotic systems used in educational context are constituted by a control board, a set of sensor and actuators, and a software to program and communicate with the robot via USB or serial RS232. In Brazil Lego kits like Mindstorms and NXT are used in some schools in robotic educational activities.

Lego platform constituted by a programmable brick and a software running in PC computer allows to build and program a variety of robots. The drawback of Lego products is the relatively high cost and software and hardware proprietary which difficulties extended applications.

Arduino[10], an open hardware platform, has recently been introduced as a robotic and automation tool. Constituted by an electronic board and a software, it allows to control devices like analog and digital sensors, and actuators like relays and motors. Developing robots with Arduino involves manipulating and knowing the characteristics of electronic devices, and creating programs in a language with a syntax that looks like Java. This need can intimidate non experts in electronics and computer programming.

In this scenario we noticed the need to develop a robotic tool with the following characteristics:
• Low cost of hardware and software
ROBOTIC TOOL WITH SCRATCH LANGUAGE
Marco Túlio Chella

• easy to use by non experts in computer programming and electronics devices
• Use tools with focus in educational process
• Could be freely distributed, studied and extended by users
To attend this purpose, a hardware control interfaces and a system software that allows integration between hardware and the authoring tool Scratch were developed.
This system enables non experts in robotic users to build and program many kinds of robots.

3. DEVELOPMENT

3.1. Scratch Language

The Scratch language developed as an educational tool applied to literacy in computer programming offers functionalities to build computer games, interactive electronic books and any kind of programs that explore interaction and multimedia resources from PC computers. Besides, Scratch doesn’t provide functionalities to access computer hardware resources, especially communication devices like USB (Universal Serial Bus) and serial RS 232.

Scratch language was developed within Mitchel Resnick’s “Life Long Kindergarten” group. Consequently, the idea of “imagine”, “program”, “share” is at the core of how Scratch pedagogy occurs in the classroom. Resnick [7] elaborates on a metaphor introduced by Seymour Papert [11]: that a computing environment should have a low floor (e.g. easy access) and a high ceiling (vast potential.). The Scratch development team added the metaphor “wide walls.” Scratch allows rapid development of 2-D event driven, asynchronous concurrent processes. It supports importing of personalized graphics (via jpeg), and audio files. Consequently it is a tool for creating 2-D games.

Programmers have used it to compose music, create greeting cards, parodies of TV shows, instructional tutorials, science simulations, interactive history, animated poetry, procedural art, animation studies, including mapping to 3-D.

Scratch is constituted by blocks that represent fundamentals as statements, Boolean expressions, conditions, loops, and variables. Other blocks, meanwhile, offer pseudo-randomness as well as multithreading and event-handling. With these blocks students can program one or more “sprites” (i.e., characters) on a “stage,” the end result of which is Scratch’s promise of some animation, game, or interactive art. In effect, Scratch lowers the bar to programming, empowering first-time programmers not only to master programmatic constructs before syntax but also to focus on problems of logic before syntax.

Textual languages like Logo need a complete knowledge of syntax and a fundamental programmatic construct to produce some result. To often do semicolons and their syntactical cousins delay, if not downright discourage students. Below a demonstration that compares a “hello world" procedure in MSWLogo (table 1) and that same in Scratch (fig 1).

```
To hello_example
  Show [Hello World]
End
```

Table 1 – MSWLogo implementation

Scratch with a robotic extension allows a complete robotic application to be developed by association of conditions, loop and multimedia blocks.

3.2. Scratch Robotic Server - SRS

The “remote sensor connections” scratch functionality provides a socket connection that sends and receives messages. This feature was used to create a middleware application software (Scratch Robotic Server - SRS) to link Scratch to hardware system. The “remote sensor connections” are enabled in Scratch by double clicking sensor block in sensing section (fig 2).

![Fig. 1. Scratch hello world example](image1)

This middleware application SRS receives and sends messages to Scratch over sockets. This messages are decoded and sent to ERB hardware. If a message is to control an actuator like an electrical motor, a command is sent to hardware. To read a sensor, a message is sent and the status of sensor is returned. A message is created and sent to Scratch to be used by the user script in Scratch. The block diagram (fig. 3) shows the architecture of system developed.

![Fig. 2. Scratch hello world example](image2)

![Fig. 3. System architecture](image3)
In Scratch the primitive “broadcast” and “broadcast and wait” send and receive text messages in asynchronous and synchronous mode respectively. By default the port 4999 is used to socket send and receive message over TCP protocol. In a Client-Server architecture Scratch acts as a server and the middleware as a client.

Two groups of messages were created: sensor or actuator. Sensor category sends messages that ask ERB about status of connected analog and digital sensors. Messages in actuator category controls motors, relays, and other devices in ERB. As an example a message “servo 1 189” sets the position in a servomotor connected in output 1 to position 189 (fig 4).

Middleware functionalities are:
- Detect and configure communication with hardware device,
- Receive and send messages over socket in TCP protocol,
- Send commands to ERB,
- Receive messages from ERB,
- Parse messages received from ERB and code to send to Scratch,
- Error check and recovery.

Figure 5 shows a block diagram of SRS software.

![Fig. 5. SRS software diagram](image)

SRS application has a GUI that allows to manage and monitor all activities with Scratch and ERB. Messages indicate status of all operations and make it easy to detect and correct problems (fig 6).

### 3.3. ERB - Educational Robotic Board

The projects requisites for hardware development of ERB control board were preliminary mind a low cost, use of off-the-shelf components, secure and easy to use.

To provide mobility in robotics projects, communications with PC and ERB are over a wireless data. The system is composed by a base communication board (fig. 7) connected to PC in USB port and a robot controller board. Radio modules with ZigBee[12] technology are used to implement the wireless link. Base and controller board have a microcontroller with a firmware that provides all functionalities of the system.

![Fig. 7. base communication board](image)

The ERB electronic circuit is based in a microcontroller PIC18F2550 a RISC 12 MIPS chip with transceiver for full speed USB 2.0, peripherals like analog to digital converter, pulse with modulation (PWM) module, pins for digital input/output and USART (serial communication). This chip is produced by Microchip and can be get at low cost (~3 dollars).

In a base communication board, the microcontroller manages data communications and provides a layer for hardware interface with USB port. The hardware is constituted by microcontroller, a voltage regulator, a XBEE module and the USB connector, a 20 MHz crystal is used to generate the clock. The power is supplied by USB cable.

Figure 8 shows a block diagram from a base communication board.

![Fig. 8. ERB board block diagram](image)
packets and sent to PC host. Figure 9 shows a block diagram from a base communication board firmware.

The controller board has functionalities to control actuators, sensors, decode commands and manage communication with base board. This is a brief functionalities description:

- **Servomotor output**: These outputs allow to control hobby servomotors. These servomotors are suitable because they are cheap, have high torque, small dimension and weight. A servomotor is a small device that has an output shaft. This shaft can be positioned to specific angular positions by sending the servomotor a coded signal. As long as the coded signal exists on the input line, the servomotor will maintain the angular position of the shaft. As the coded signal changes, the angular position of the shaft changes. Servomotor can be modified to execute continuous rotation. ERB has eight servomotors outputs
- **Analog inputs**: in number of four can read sensor data with 10 bits resolution.
- **Input/Output pins**: pins that can be configured as digital input or output by user command are in number of three.
- **Output with relay**: this output allows to control two devices like lamps, motor and other devices that require higher energy consumption.

Auxiliary classes are implemented to read and write to serial hardware, parse commands, and catch hardware and operation errors. Figure 10 shows a block diagram from an ERB firmware.

An open and documented ASCII based protocol enables applications and API’s development to communicate with ERB in any computer language with resources to communicate over serial port RS 232 and USB virtual serial ports.

The protocol is composed by a header character, a command character, followed by one or more characters with command parameters. Two characters indicate an end message.

Physically, ERB has a set of connectors that allow to connect motors and other actuators, and analog or digital sensors (fig. 11).

Command sets allow to access individual resources from hardware. For example, servomotor control allows to select servomotor to be controlled and set the position that it must assume. Associating commands allows the user to build complex robotics applications.

4. Robots design with Scratch and ERB

The process to build a robot consists of connecting the actuators and sensors in ERB, creating the physical robot, and implementing applications that provide functionalities to robots.

To exemplify, a robot created with ERB and Scratch will demonstrate a called Braitenberg Vehicle [13].

Braitenberg vehicles are simple automata, and illustrate the abilities of reactive robots, thus representing the simplest form of Behavior Based Artificial Life or embodied cognition, i.e. intelligent
behavior that emerges purely from sensor and motor interaction between the agent and its environment, without any need for an internal memory, representation of the environment, or inference.

The Braitenberg Robot developed can autonomously move around. It has a light sensor and two wheels (each driven by its own motor) that function as actuators or effectors. The values read in light sensor determine the control that will be sent to motor and movement the wheels. Depending on how sensors values are used to send commands to wheels the robot exhibits different, goal-oriented behaviors. The following rules will be implemented:

- More light produces away movement
- Less light produces far away movement

This behavior can be interpreted as a robot that is afraid of the light and that moves fast to get away from it. Its goal is to find a dark spot to hide. The Scratch implementation of these rules is demonstrated in figure 13.

![Fig. 13. Scratch implementation of Braitenberg Robot](image)

The physical implementation of robot is show in figure 14.

![Fig. 14. Braitenberg Robot implementation](image)

We are in the initial stage of presenting workshops for students aged 8 to 12 years old. In these workshops they can build and program a robot according to the school curriculum. The pedagogical approach is that suggested by Papert[11] and based on Piaget[14] theories in which the learner occupies a central position in the learning process. Examples of created robots are a similar industrial manipulator and a dancing robot that moves in the rhythm of music.

5. CONCLUSION

In the process to build a robot, students face challenges with the physical design, the adequate algorithm, the behavior of motor and sensor. Our proposal is to offer hardware and software tools in which the student could concentrate in the conceptual aspects of the project and could experiment quickly the various alternatives in development.

A control board that encapsulates the hardware functionalities and clear software could make easy the implementation of robots by non experts in computer and electronics students.

The first result of our experiments is that ERB and Scratch could be a tool applied to educational robotic with effectiveness.

ACKNOWLEDGMENTS

Thanks to FAPITEC (Fundação de Apoio à Pesquisa e à Inovação Tecnológica do Estado de Sergipe), by partial sponsor.

REFERENCES

1989.