Revised Selected Papers

Accademia Musicale Studio Musica
Michele Della Ventura, *editor*

2019

# Proceedings of the International Conference on New Music Concepts and Inspired Education

Vol. 6

ACCADEMIA MUSICALE STUDIO MUSICA

**Accademia Musicale Studio Musica**

# International Conference on New Music Concepts and Inspired Education

Proceeding Book
Vol. 6

Accademia Musicale Studio Musica
Michele Della Ventura
Editor

Printed in Italy
First edition: April 2019

www.studiomusicatreviso.it
Accademia Musicale Studio Musica – Treviso (Italy)
ISBN: 978-88-944350-0-9

## Preface

This volume of proceedings from the conference provides an opportunity for readers to engage with a selection of refereed papers that were presented during the International Conference on New Music Concepts and Inspired Education. The reader will sample here reports of research on topics ranging from mathematical models in music to pattern recognition in music; symbolic music processing; music synthesis and transformation; learning and conceptual change; teaching strategies; e-learning and innovative learning. This book is meant to be a *textbook* that is suitable for courses at the advanced undergraduate and beginning master level. By mixing theory and practice, the book provides both profound technological knowledge as well as a comprehensive treatment of music processing applications.
The goals of the Conference are to foster international research collaborations in the fields of Music Studies and Education as well as to provide a forum to present current research results in the forms of technical sessions, round table discussions during the conference period in a relax and enjoyable atmosphere.
36 papers from 16 countries were received. All the submissions were reviewed on the basis of their significance, novelty, technical quality, and practical impact. After careful reviews by at least three experts in the relevant areas for each paper, 12 papers from 10 countries were accepted for presentation or poster display at the conference.

I want to take this opportunity to thank all participants who have worked hard to make this conference a success. Thanks are also due to the staff of "Studio Musica" for their help with producing the proceedings. I am also grateful to all members of Organizing Committee, Local Arrangement Committee and Program Committee as well as all participants who have worked hard to make this conference a success.
Finally I want to appreciate all authors for their excellent papers to this conference.

April 2019 Michele Della Ventura

# Contents

# Educational Non-visual Environment for Symbolic Programming of Cartesian Motion to include Children with Visual Impairment into Robotic Sciences

Francisco J. Ruiz-Sanchez, Enrique Mireles-Rodriguez, Gustavo Guzman Solis

Grupo de Robótica y Manufactura Avanzada, CINVESTAV-Saltillo, México
fruiz@cinvestav.mx

**Abstract.** Robotic Sciences, as a fundamental research area in modern technology, is an attractive opportunity for professional-fulfilment that must be instilled from a young age. Unfortunately, teaching and didactic material are mainly visual, excluding blind or partially sighted children of an early inclusion in robotics that results in a critical disadvantage with respect to their peers in the labor market. In this paper, we present a non-visual programming environment for visual impaired children. This environment is designed to foster the ability of coding motion tasks by teaching basic concepts of programming with an experimental reinforcement. The environment is a robotic platform for Cartesian motion with a block-programming interface and its didactic strategy of gradual complexity. We introduce a symbolic language of the egocentric Cartesian movements and we describe its implementation for visual impaired users as safe and friendly tokens with tactile images and auditory assistance. The programming interface includes a set of assembling tokens and a desk-board with multi-ports to connect strings of tokens coding algorithms. The sequence of tokens in a string is identified and verified before being sent to the robot platform, introducing the users into the general process of coding and debugging. The environment is implemented as a centralized network with a main computer and branches of master-slave microprocessors in a wireless connection for the robot. We broadly describe the didactic strategy to teach the basic concepts of programming using the environment and we conclude showing its final implementation.

**Keywords.** Accessibility to Disabled Users, Blind inclusion in Robotics, Educational non-visual programming interface, Robotics for visually impaired

## 1 Introduction

Robotic Sciences are a fundamental multidisciplinary approach in Modern Technology. They are interested in the formal study of intelligent automated mechanisms, controlled by electronic devices, and their interaction with humans [1]. Their study requires of a spatial perception of the mechanical motion and the ability to describe it in terms of analytic and logic tools whose learning must be instilled from young age to allow a proper mental representation of the motion in the space and an abstract capacity to describe motion tasks as algorithms.

Children are encouraged to be involved in Robotic activities since their early education by means of ludic platforms (Cobetto by Primo Toys, Coding Awbie by OSMO, Kick-starter by Algobrix, Lego Boost by Lego, etc.). These platforms are composed of a symbolic programming interface of basic movements or actions, and a safe robotic device, either real or virtual, in a structured environment. The robots perform actions, mainly related to Cartesian movements, whose description and control in terms of instructions allows an intuitive understanding of coding algorithms even before the children acquire their first reading skills. Unfortunately, these platforms are mainly visual hampering the participation of blind or partially sighted children in conventional courses [2] [3].

Developers and researchers visually impaired regret the lack of non-visual didactic material in their early years of their education. They consider that would had been very appreciated to develop their abilities before their middle and high school years. The lack of teaching material that translates graphical information in alternate formats accessible to blind students, forces them to wait until College to receive a formal education in Robotics and Computer Sciences; demotivating their interest in these areas (less than 4 % of the total number of freshmen with visual impairments are engaged in computer sciences [2][4]).

Visual impaired people develop alternative sensory capacities that have been used to design assistance devices for the daily living activities [5][6]. Nowadays, modern technological paradigms provide the means to enhance the use of these capacities by multisensory interfaces that translate visual information into auditory/tactile images [3]. These interfaces allow the transmission of scientific information, inherently visual, and encourage the participation of visual impaired people in Science and Technology. Braille and tactile displays, together with auditory word processors, already provide the basic interaction tools to get involve with the use of computers [7]. Nevertheless, the ability of coding algorithm requires knowledge and comprehension of the abstract concepts and structures of programming and, when they are applied to describe robotic tasks, this knowledge is strongly related to the perception of motion and its representation with respect to its environment.

In this paper, we present a programming environment for children with visual impairments, conceived to teach the basic concepts of coding in robotics with an experimental reinforcement. It is aimed to foster the abstract ability to code algorithms stimulating the spatial orientation. The environment is composed of a non-visual programming interface of symbolic language, a robotic platform for Cartesian motion, and a didactic learning strategy of gradual complexity. The interface is a block-programming interface of a graphic language, implemented with assembling tokens of tactile and auditive images, describing the basic egocentric movements and actions in a mobile robot. It provides of a non-visual multi-port desk-board to connect the sequences of tokens, or programs, with auditory assistance to help the user in the process of coding and debugging, before executing the program in the robot. We describe the conceptual idea of the programming environment and discuss its designing characteristic concerned with visually impaired users and the way it can be used in a didactic strategy to teach the basic concepts of programming, and we show its final implementation.

## 2 Educational Non-visual Programming Environment

The proposed non-visual programming environment is a robotic platform with a programming interface designed for blind and visually impaired children. It is aimed to teach the basic concepts of programming by coding Cartesian motion tasks regarding the importance of the motion perception in Robotics (Fig. 1).

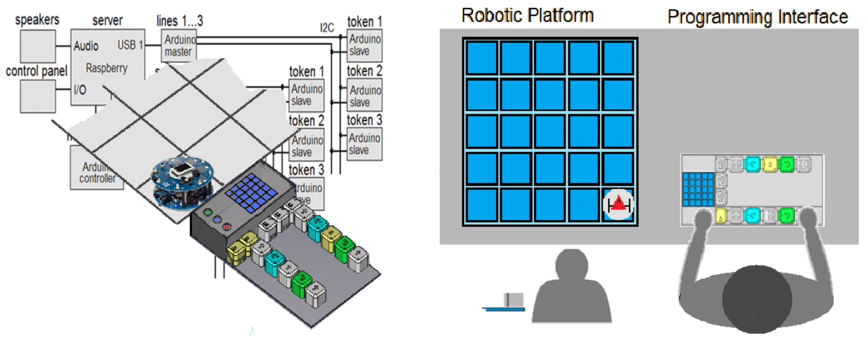


Fig. 1 Non-visual robotic environment for Symbolic Programming of Cartesian 2D motion.

### 2.1 Programming Interface

The programming interface is a non-visual interface for block programming with auditory assistance. It is implemented using assembling tokens describing commands of the basic Cartesian movements and the number of times they are executed. The programming interface also includes a multi-ports desk-board as a working space to connect programmed algorithms as string of tokens with auditory assistance to code and debug the program before being sent to the robotic platform (Fig. 1).

#### 2.1.1 Command Tokens for Block Programming

Tokens in the programming interface define the commands and their modal attributes for the block-programming interface. They describe the basic Cartesian movements in a plane from an egocentric frame of reference (a natural description of motion for blind persons) and determine the number of times a movement is executed.

Commands are represented graphically by intuitive symbols based on arrows that can be easily interpreted by kids even before they acquire reading skills. For instance, a straight arrow pointing in the forward direction with respect to the user, for the command *step* *forward*, and curved arrows pointing to the left or to the right, for the commands *turn left* and *turn right*, respectively (Fig. 2).

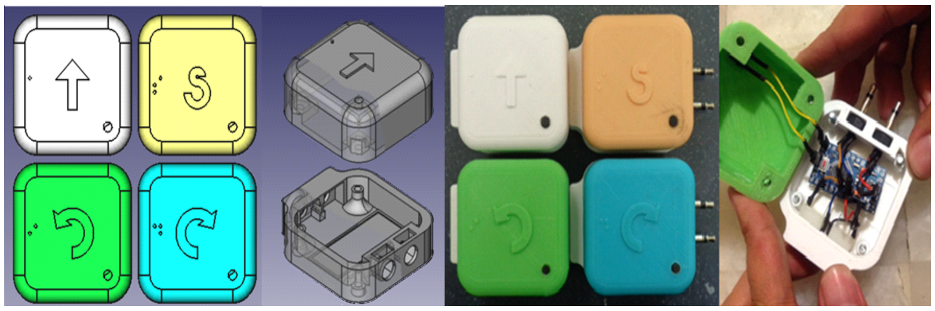


Fig. 2 Command Tokens: CAD design and instrumented 3D printed tokens.

Beside the basic Cartesian movements, or primitive commands, in the programming language, tokens can also represent two programming structures: *subroutines* or composed commands, and *conditional actions* executing a subroutine regarding the state of the system in the environment, both described by a stylized "S" and a vertical diamond recalling the conditional symbol in the flow charts, respectively. These symbols were designed for easy visual and tactile identification thinking on the users and on their instructors. With this concern, it is important to remark that blind children do not necessarily interpret these symbols in the same way as sighted people interpret them visually, thus the symbols were designed to be easily identified in a tactile inspection and the users are required to remember them and their associated actions.

Tokens are identified by the graved symbols on high relief in its upper surface that also include, at their left side, a Braille letter indicating the command -a (step forward), i (turn left), d (turn right), s (subroutine), c (conditional)-. In addition, tokens are provided of a small switch in the lower right side of the symbol as a tactile way to change modal function of the command. The switch, provided with an auditory assistance, modifies and indicates the number of times the command is executed, and, identifies the right orientation of the token.
Tokens are specially designed for a safe manipulation by children and, besides the tactile recognition, they are identified by colors to help the instructors and therapists when assisting a user. They are designed with a cover for identification and a generic base for support and connection. This last instrumented with a microprocessor to improve their functionality given the possibility to include further sensory aids for the user.Tokens were manufactured in a 3D printer (Fig. 2).

## 2.1.2 Multiport Deskboard

The desk-board of the programming interface is an ergonomic multi-ports area to connect the strings of assembled tokens. Ports are identified in a similar way as the tokens, denoting either program lines or subroutines by the letters *L* or *S*, respectively, including the Braille identifier (Fig. 3). The *L-ports* recognize the connected string of tokens as the lines of the main program where commands are interpreted from the left to the right and ordered from L1 to L3. The *S-ports* are for subroutines, either to host special sequences of commands in the program or alternative actions triggered by the conditional commands. The use of multi-ports distributes long sequences of commands in short strings. This limits the required working area into a safe region at the fingertips in front of the user avoiding unwilling collisions with arms and elbows in a non-visual manipulation.

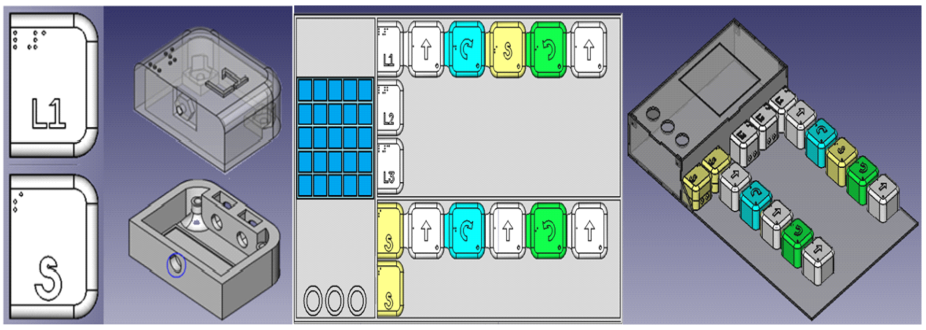


Fig. 3 Multi-ports desk-board: CAD design of the ports and desk-board

The multiport desk-board, in its left side, provides a small tactile display of the Cartesian plane and a control panel. The Cartesian display allows the user to identify the starting and final points in a motion task (*home* and *goal*), in order to determine the sequences of movements required attain un objective. The control panel is a three buttons panel to *start*, *pause* and *stop* the execution of the program. In this version of the system the Cartesian display is passive, but it is considered to provide it with extra sensory information, to allow a real time interaction with the robot along its motion.

## 2.1.3 Robotic Platform

The robotic platform is a test-bench of a Cartesian mobile robot in a horizontal plane which is located physically beside the programming interface, in the left side of the user (Fig. 1).

The mobile robot is a two-wheel differential drive mobile robot that moves in a step-by-step motion either to arrive to its next forward position or to turn around its axis to the next direction. The robot is controlled with an embedded computer in a move-and-wait mode using infra-red sensors to guide its motion and detect obstacles. It is also provided with sound signals indicating its position and allowing the user to trace its motion. It is wireless communicated with the programming interface to receive the program coded by the user. We provided the robot of a protection cap of rounded corners for protection and security, also with a high relief tactile and visual images on the upper side of the cover cap to identify its frontal direction and to orient its motion on the Cartesian space (Fig. 4).

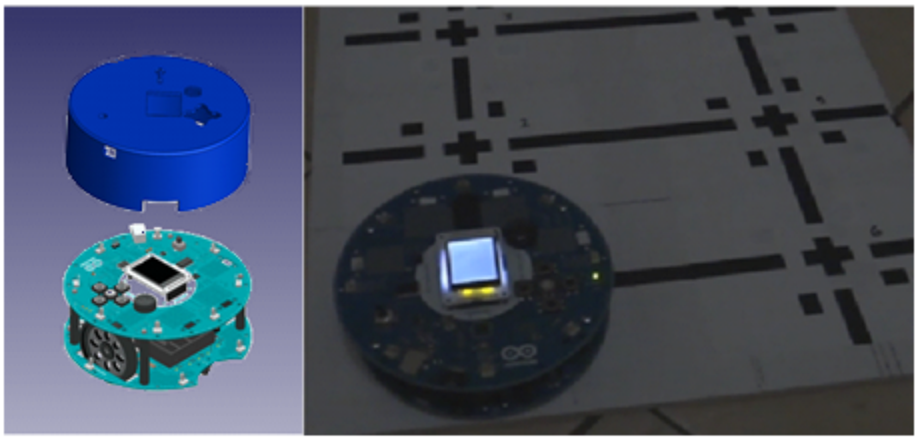


Fig. 4 Robotic Platform: Cartesian mobile robot and the horizontal plane with coded positions.

The Cartesian plane is a horizontal discrete plane of 5x5 locations. This dimension is not fixed; it can be enlarged by software according to the needs of the teaching strategy. The plane has landmarks to guide the motion of the robot with a continuous sampling of its position on the plane. Its dimension (11 cm radius) provides the users of an easy to handle robot using both hands, and of an acoustic perception of the approximate position of the robot with respect to the user. This reinforces the coded signals emitted by the robot to determine its position.

### 2.1.4 Firmware for Control and Communication

The programming interface is implemented as a distributed system that accomplishes parallel processes related to the interaction with the user and to the experimental work in the robot. Its software was developed considering it as a firmware for control the internal communication in the programming interface, and between the interface and the robot. It considers a star topology network of connected microprocessors with a central micro-computer, hosted physically in the left side of the deskboard.

Tokens are instrumented with a microcontroller to identify its function and its recorded number of repetitions. These microcontrollers are operated in a slave mode and connected to a master microcontroller located in the ports of the programming interface using a wire connection. The information of the strings of tokens is sent, via the master microprocessor, to a main single block computer where it is processed to identify the contained code. This code is verified assisting the user to debug and interprete it before being sent wireless to the mobile robot as an executable program (Fig. 5).

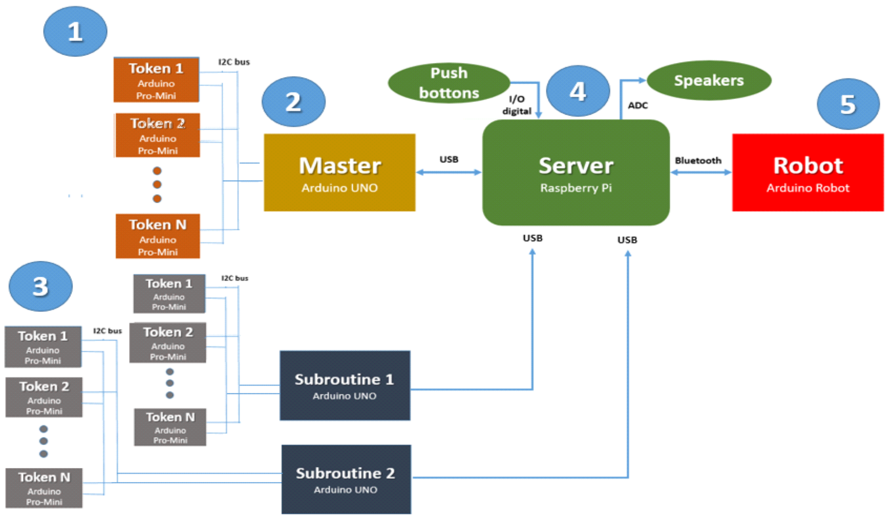


Fig. 5 Control and Communication network: 1) Lines of program, 2) Master module, 3) Subroutines, 4) Server Module, 5) Robot Module.

The main computer, among its function of server, also receives the input of the control panel (*start*, *stop* and *pause* buttons) and send, as an output signal, an auditory assistance to the user.

## 3 Symbolic programming of motion tasks

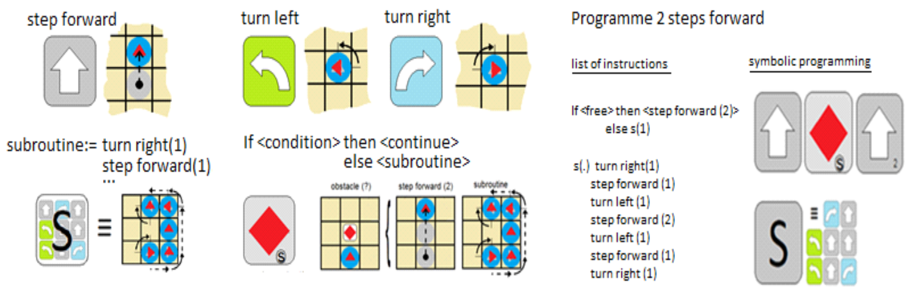
Robotics Sciences are highly concerned with the description and control of mechanical devices where the perception of motion and its interpretation in a specific context, determines the ability to code motion tasks in a student. In this way, we consider the problem of a Cartesian navigation on a horizontal plane as the simplest case of robotic motion and we use it as a case of study to introduce the basic concept of coding using a symbolic programming language.

Symbolic Programming introduced by John McCarthy and his group [9] provides a programming paradigm that synthesizes complex functions in a simple structure of action modifiable by some parameters [10], in particular, representing intuitive actions and the way these actions are executed, without a detailed description of the computing steps processing input data to get the desired result. This concept allowed the emergence of computing language for children where natural and intuitive actions are easily described by common words (or images) and numbers, and thus, providing a friendly didactic tool to teach the basic concepts of programming.

## 3.1 Basic commands for 2D Cartesian motion

On a Cartesian plane with natural numbers as coordinates, the basic actions to move all along the surface can be reduced to step and turn, and both, considered as commands, are specified by two attributes of the specific sense of the action, i.e. {*forward, backward*} for the step, and {*left, right*} for the turn. However, given the egocentric perception of motion in blind students and their preference to move in the frontal direction, we rejected the {*backward*} attribute and instead, we defined the composed action {*step forward*} as the basic command to produce a displacement, and {*turn left*} and {*turn right*} as the basic command for directional action. Thus, in our proposal we defined the set of basic commands as {*step forward (.), turn left(.), turn right(.)*} to control the navigation of a robot in a step-by-step motion.

The proposed commands for the non-visual programming environment, correspond to the basic procedures of motion defined in the programming language Logo [11]. Procedures of action and specific attributes of direction are set together in a single command, and the number of times the action is executed as the modal argument, i.e. *step forward (n):= repeat n [fr 1]*, *turn left (n):= repeat n [lt 90°]* and *turn right(n):= repeat n [rt 90°]*. This description of the basic motion actions is intuitive and allows a simple representation of the commands as graphic images using arrows that simplifies their identification and implementation in a tactile/auditory format of symbols (Figures 2 and 6).



```
If <free> then <step forward (2)>
      else s(1)

s(.)  turn right(1)
    step forward (1)
    turn left (1)
    step forward (2)
    turn left (1)
    step forward (1)
    turn right (1)
```

Fig.6 Symbolic programming of motion: Basic, subroutines and conditional commands, and an example of symbolic program and its equivalent program expressed as list of instructions.

## 3.2 New composed commands of a sequence of basic actions

In a coded algorithm, a sequence of basic actions can be executed repetitively with a self-meaning. This sequence can be grouped under a new command that synthesize its general action in a single call whose definition can be complemented using modal attributes.
The synthesis of new commands is a main concept in structured programming that simplifies the general structure of the programs and optimizes their coding. We introduce

this concept of composed commands in our programming environment as the function *subroutine*, called as *s(.),* which is in fact a sub-program executed as part of the main program (Fig. 6) considered as a single command whose argument determine the number of times it is executed. This concept develops in the users the ability of coding complex motion task in a clear and organized structure of growing complexity.

### 3.3 Conditional actions

The effectiveness in the execution of program can be affected by changing conditions in the environment, and the program must be adapted to deal with unexpected conditions by means of a conditional action. This action, seen as a command, verifies a logic condition before continuing the execution of the program, and according to its value *true* or *false*, continues or executes a new sequence of actions defined in a *subroutine*. The conditional command is defined as *if <condition>*, graphically represented by a diamond in the symbolic language (Fig. 6). In particular, in the conditional command applied to the motion of a mobile robot in a Cartesian plane, the condition is defined as the possibility of moving the robot to the next position, either continuing the program if the next location is free or running a subroutine to avoid an obstacle if the location is occupied (at least temporary unavailable).

### 3.4 Coding Symbolic programs

The commands for coding motion actions in a Cartesian plane were defined as a symbolic programming language for kids susceptible to be represented by graphic images either visual or tactile, and the sequence of symbols codes an algorithm that can be written in its equivalent form of a list of instruction. In Figure 6, we illustrate the use of the three kinds of commands in a simple motion task. Moving the mobile robot two locations ahead even in the presence of an obstacle. We introduce the code of the symbolic program and its equivalence in graphic representation, showing its potential to teach abstract programming concepts with tactile images of the proposed non-visual interface.

## 4 Didactic strategy for teaching the basis of coding

The proposed programming language was created to teach the basic concepts of coding by means of a didactic strategy based on exercises of growing complexity associated to a concrete experience of robotic motion. Exercises are presented as challenges to move the mobile robot from an initial position (*home*) to a final one (*goal*) accomplishing a motion task. They are organized according to the level of abstraction describing the commands presented in the previous section. The users are asked to code algorithms based on the library of commands to solve the challenges in the programming interface and encouraged to test them in the robot platform reinforcing the learning process and motivating the kids to continue their training. It is important to remark that the exercises are also concerned to stimulate the orientation ability of the user allowing to create a mental representation of the workspace and the effects of egocentric movements in the global position of the robot.

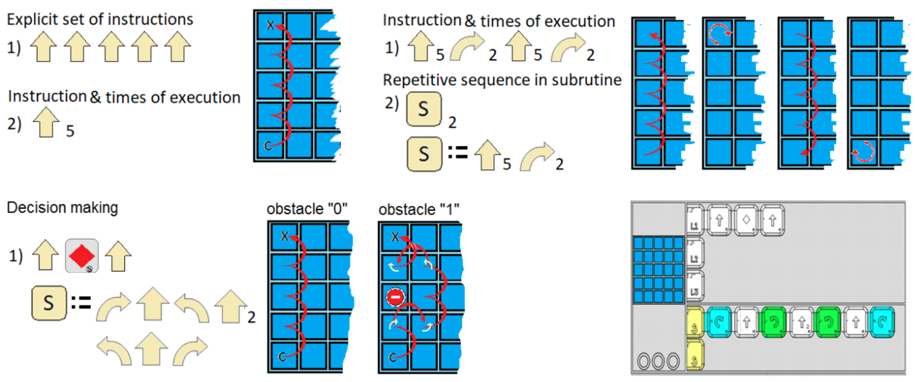

Explicit set of instructions
1)
Instruction & times of execution
2) 5
Instruction & times of execution
1) 5 2 5 2
Repetitive sequence in subrutine
2) S 2
S := 5 2
Decision making
1)
S := 2
obstacle "0"
obstacle "1"

Fig.6 Example of the growing complexity teaching strategy: series of basic commands, commands and modal attribute, subroutines, conditional command.

## 4.1 Primitive Commands: Basic commands and its modal arguments

We start teaching the concept of function and its arguments by moving the robot from one point to another in the Cartesian space. The aim is awake the capacity to describe a repetitive action by the abstract idea of the action itself and the number of times it is executed.

The user is asked first to trace a straight-line trajectory in a step by step sequence of actions, either in a frontal or a lateral direction, i.e., indicating explicitly by repeating the command the same number of times the command is executed. Then, introducing progressively in the desired trajectory changes in direction to stimulate the spatial orientation with respect to an absolute referential frame and finally, optimizing the code by reducing the number of calls to the command and determining the number of times it is executed by means of the argument (Fig. 6).

## 4.2 Subroutines: composed commands

New functions, defined as sub-programs executing specific actions, are introduced proposing motion task that repeat a sequence of actions to be accomplished. We take advantage of some of the programs developed in previous level to propose partial motion goals integrated as subroutines, whose definition develop the capacity to recognize common procedures for optimizing the code. In Figure 6, we show a simple motion task describing a round trip, starting and concluding at the same point, calling twice a subroutine defined as the function to go from one side to the other in the Cartesian plane.

## 4.3 Decision making: conditional commands

Decision making, with logic commands, is taught by means of motion tasks that can be realized, either partially, in different ways. In particular, the conditional subroutine determines an alternative solution when the conditions of the environment prevent the system to continue the normal execution of an algorithm. The alternative solution starts running when the conditional command detects the state of the condition as true and resumes the algorithm after the unexpected conditions is surpassed. In Figure 6, we present the example of obstacle avoidance if the frontal place is occupied. After avoiding the obstacle, the system regains its normal execution to attain its goal.

## 5 Final Implementation of the Programming Environment

The final implementation of the Programming Environment is showed in Figure 7. Tokens and ports were manufactured in a 3D printed using ABS material of different colors for easy identification, and they were instrumented with Atmega32 microcontrollers. Color and visual symbols were implemented to help the instructors and therapists to use the system and to assist the blind and visual impaired users when using the environment. The body of the programming interface was made in acrylics considering the multi-ports desk board in its right side, and a host place for the computing elements in the left side, under the reference Cartesian display (Fig. 7).

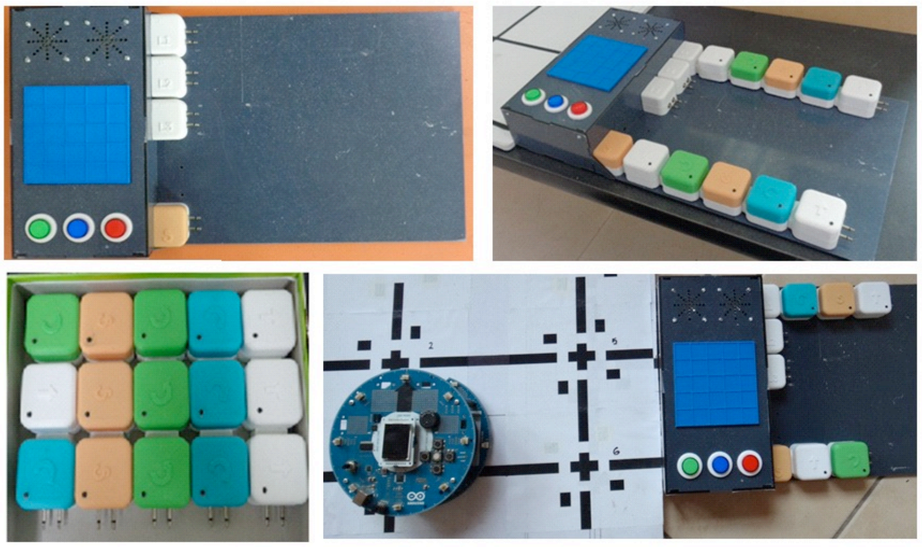


Fig.7 Images of the final implementation of the programming environment showing the non-visual programming interface and the robotic platform.

The mobile robot is an AlphaBot2 by Arduino moving in a horizontal plane of 5x5 squares. The plane is implemented in a white Eva Foam surface where the Cartesian position are marked in a black digital code to guide the motion of the robot using the lower matrix of infrared sensors included in the robot.
The Firmware of the environment was developed with the Arduino open-source C/C++ libraries using a Raspberry Pi as a main single block computer communicated to the master and slave microcontrollers by a serial I2C protocol or to the mobile robot by a Bluetooth transmitter.

## 6 Conclusion

In this paper, we presented the design and implementation of a didactic platform to include children with visual impairments in robotics. It is intended to foster the ability of coding abstract algorithms from specific motion tasks and, implicitly, to stimulate the spatial orientation in a global frame of reference as the result of a sequence of egocentric movements. The platform provides a non-visual programming environment composed of a tactile/auditive programming interface and a robotic test-bench for experimental reinforcement.
The design and implementation of the non-visual interface imposed the challenge of creating a practical device, intuitive and interesting enough, for children with visual impairments, i.e., provided of non-visual information, perceptible and understandable for visual impaired users, and friendly for the instructors that assist the users. For instance, tokens are marked with colors and visual/tactile images of symbols representing the basic egocentric movements in a Cartesian plane easily identifiable either by touch or by sight. Their size and form, cubic with rounded edges, and their parallel connectors respond to requirements detected of safety and easy manipulation for visually impaired children. Tokens implemented with microcontrollers offers the possibility of a better interaction with the user by means of the hardware already available in the market to enhance the sensory information provided to the users. In this work we presented the use of these microprocessors to determine the number of times the command is repeated; however, this functionality can be expanded to describe the motion of a more complex devices beyond the motion of a mobile robot in the plane, for instance of a manipulator in its articular space.
Currently the prototype is tested by volunteers before being used in a classroom, the results will be presented in future works. It is important remark that the proposed programming environment is designed for children with visual impairments; however, its concept as a didactic material is useful in general to introduce children in Robotic Sciences during their first years of education.

## Acknowledgement

This work has been supported by Industrias Plásticas Martin and Cinvestav-IPN under the grant *1er premio innovación y diseño de juguetes 2017*. The authors are indebted with the instructors of the *Centro Atención a la Discapacidad y Rehabilitación Integral* for their comments and advise.

## References

[1] K. Rajan and A. Saffiotti, "Editorial: Towards a science of integrated AI and Robotics", Artificial Intelligence no. 247, Elsevier, pp. 1-9, 2017.

[2] A. M. Howard, Ch H Park and S. Remy, "Using Haptic and Auditory Interaction Tools to Engage Students with Visual Impairments in Robot Programming Activities", IEEE Transaction on Learning Technologies, vol. 5, No. 1, pp 87-95, January-March 2012.

[3] N. M. Al-Ratta and H. S. Al-Khalifa, "Teaching Programming for Blinds: A Review", Forth International Conference on Information and Communication Technology and Accessibility (ICTA), Hammamet, Tunisia, 2013.

[4] B. Beck-Winchatz and M. A. Riccobono, "Advancing participation of blind students in Science, Technology, Engineering, and Math, Advances in Space Research, no. 42, pp. 1855-1858, 2008.

[5] P. W. Nye and J. C. Bliss, "Sensory Aids for the Blind: A Challenging Problem with Lessons for the Future", Invited paper, Proceeding of the IEEE, vol. 58, No. 12, 1970.

[6] J. Liu and X Sun, "A Survey of vision Aids for the Blind", Proceedings of the 6th World Congress on Intelligent Control and Automation, Dalian, China pp.4312-4316, 2006.

[7] V. G. Chouvardas, A. N. Miliou, M. K. Hatalis, "Tactile display: Overview and recent advances", Displays, Elsevier vol. 29, pp. 185-194, 2018.

[8] J. McCarthy, "Recursive Functions of Symbolic Expressions and Their Computations by Machine, Part 1", Communication of the ACM 3:4, pp 184-195, April 1960.

[9] D. S. Touretzky, "COMMON LISP: A Gentle Introduction to Symbolic Computation", The Benjamin/Cummings Publishing Company, Inc., 1999.

[10] S. Delaney, "A Very Basic Introduction to MSW Logo Programming, http://sean-delaney.com/wp-content/uploads/2012/01/A-Very-Basic-Introduction-to-Logo-Programming.pdf, 2012.

This book presents a collection of selected papers that present the current variety of all aspect of music research, development and education, at a high level. The respective chapters address a diverse range of theoretical, empirical and practical aspects underpinning the music science and teaching and learning, as well as their pedagogical implications. The book meets the growing demand of practitioners, researchers, scientists, educators and students for a comprehensive introduction to key topics in these fields. The volume focuses on easy-to-understand examples and a guide to additional literature.

Michele Della Ventura, editor

**New Music Concepts and Inspired Education**

Revised Selected Papers

ISBN: 978-88-944350-0-9

9 788894 435009

www.studiomusicatreviso.it