Ingeniería e Investigación
ISSN: 0120-5609
revii_bog@unal.edu.co
Universidad Nacional de Colombia
Colombia

Giraldo, C. A.; Florian-Gaviria, B.; Bacca-Cortes, E. B.; Gómez, F.; Muñoz, F.
A programming environment having three levels of complexity for mobile robotics
Ingeniería e Investigación, vol. 32, núm. 3, diciembre, 2012, pp. 76-82
Universidad Nacional de Colombia
Bogotá, Colombia

Available in: http://www.redalyc.org/articulo.oa?id=64324939015

How to cite
Complete issue
More information about this article
Journal's homepage in redalyc.org

Scientific Information System
Network of Scientific Journals from Latin America, the Caribbean, Spain and Portugal
Non-profit academic project, developed under the open access initiative

A programming environment having three levels of complexity for mobile robotics

Entorno de programación con tres niveles de complejidad para robótica móvil

C. A. Giraldo[1], B. Florian-Gaviria[2], E. B. Bacca-Cortes[3], F. Gómez[4], F. Muñoz[5]

ABSTRACT

This paper presents a programming environment for supporting learning in STEM, particularly mobile robotic learning. It was designed to maintain progressive learning for people with and without previous knowledge of programming and/or robotics. The environment was multi-platform and built with open source tools. Perception, mobility, communication, navigation and collaborative behaviour functionalities can be programmed for different mobile robots. A learner is able to programme robots using different programming languages and editor interfaces: graphic programming interface (basic level), XML-based meta-language (intermediate level) or ANSI C language (advanced level). The environment supports programme translation transparently into different languages for learners or explicitly on learners' demand. Learners can access proposed challenges and learning interfaces by examples. The environment was designed to allow characteristics such as extensibility, adaptive interfaces, persistence and low software/hardware coupling. Functionality tests were performed to prove programming environment specifications. UV-BOT mobile robots were used in these tests.

Keywords: Programming environment, mobile robot, STEM, meta-language.

RESUMEN

Este artículo presenta un entorno de programación concebido para apoyar la enseñanza en STEM y en particular la enseñanza de robótica móvil. Este ha sido diseñado para soportar un aprendizaje progresivo, desde personas sin conocimientos en programación o robótica, hasta expertos. El entorno es multiplataforma y edificado con herramientas de software libre. Las funcionalidades de percepción, movilidad, comunicación, navegación, y los comportamientos colaborativos, se pueden programar para diferentes robots móviles. El usuario puede programar los robots utilizando diversos tipos de lenguajes e interfaces de edición: 1) desde un ambiente gráfico de programación por bloques (nivel básico); 2) empleando un metalenguaje basado en XML (nivel intermedio); o 3) usando el lenguaje de programación nativo del robot ANSI C (nivel avanzado). El entorno soporta la traducción de los programas entre los distintos lenguajes de forma transparente al usuario o de forma explícita si este lo desea. El usuario dispone de interfaces para la solución de retos propuestos y la capacitación por medio de ejemplos. El diseño del entorno permite extensibilidad, adaptabilidad de interfaces, manejo de persistencia y bajo acoplamiento software/hardware. Se realizaron pruebas funcionales para comprobar las especificaciones de acuerdo con las cuales fue construido el entorno. Para las pruebas se utilizaron los robots móviles UV BOTs.

Palabras clave: Entorno de programación, Robots móviles, STEM, Metalenguaje.

Received: November 23rd 2011
Accepted: November 9th 2012

Introduction

Despite economic crisis, job demand related to science, technology, engineering and mathematics (STEM) has been growing. For instance, in Washington DC, USA, increase in such labour supply was 11% between 2001 and 2011 (Koebler, 2011). It has been estimated that there were eight million STEM-related jobs in the USA in 2011. The European Union has bonded STEM-related jobs to its strategic plan concerning education level and as a motor for competitiveness, productivity and environmental sustainability (European Center for the Development of Vocational Training - CEDEFOP, 2010). The South Korean research institute KIST has implemented an educational programme for assisting learners using mobile robots (Sang-Rok, 2011). However, America and Europe have been hit by educational crisis in STEM-related fields; science, technology, engineering and mathematics are considered boring and very demanding. Moreover, the number of students has decreased in these professional fields

[1] Carlos Alberto Giraldo. Affiliation: ATOS International, Spain. MSc in Artificial Intelligence, Universidad Politécnica de Madrid, Spain. BSc in Systems Engineering Universidad del Valle, Colombia. E-mail: carlos.giraldo@atos.net

[2] Beatriz Florian-Gaviria. Affiliation: EISC, Universidad del Valle, Colombia. PhD in Technology. Universitat de Girona, Spain. MSc in Computation and Systems Engineering, Universidad de los Andes, Colombia. BSc Systems Engineering, Universidad del Valle, Colombia. E-mail: beatriz.florian@correounivalle.edu.co

[3] Eval Bladimir Bacca-Cortes. Affiliation: EIEE, Universidad del Valle, Colombia. PhD in Technology, Universitat de Girona, Spain. MSc in Automatic, Universidad del Valle, Colombia. BSc Electronic Engineering, Universidad del Valle, Colombia. E-mail: bladimir.bacca@univalle.edu.co

[4] Felipe Gómez. Affiliation: Universidad del Valle, Colombia. BSc Electrical Engineering, Universidad del Valle, Colombia. E-mail: felgoriz@univalle.edu.co

[5] Francisco Muñoz. Affiliation: Universidad del Valle, Colombia. BSc Electrical Engineering, Universidad del Valle, Colombia. E-mail: pacho_munoz@hotmail.com

How to cite: C. A. Giraldo, B. Florian-Gaviria, E. B. Bacca-Cortes, F. Gómez, F. Muñoz. (2012). Programming environment with three levels of complexity for mobile robotics. Ingeniería e Investigación, Vol. 32, No. 3, pp, 76-82.

(Ulloa, 2008). Mobile robotics has thus emerged as a popular concept for engaging learners in STEM-related fields, due to the fact that it has been shown to be a tool producing tangible results regarding how learners acquire new knowledge about technology and related abilities (Brauner, Leonhardt, Ziefle and Schroeder, 2010). Robotic seedbed experience in Cali, Colombia (Jimenez Jojoa, Caicedo Bravo and Bacca-Cortes, 2010) has corroborated a large increase in learners' motivation and therefore their results when children and young people built, programmed and tested a mobile robot for accomplishing a specific task involving concepts regarding sensors, actuators, programming and mainly a lot of hands-on tasks.

Teaching STEM concepts nowadays requires a shift in educational thinking and appropriate learning tools, such as programming interfaces which adapt in terms of complexity depending on a particular learning purpose (entertainment, education and/or research).

Table 1 gives the characteristics of several programming interfaces for mobile robots: SR1 (INTPLUS, 2011), Robolab (the LEGO Group, 2011), Mindstorms (the LEGO Group, 2012), BrickOS (Noga, 2004), leJOS (Solorzano, 2012), BotStudio (K-Team Corporation, 2011), Cricket Logo (Handyboard, 2009), Webots (Cyberbotics Ltd., 2012), ARIA (Adept MobileRobots, 2012) and Pekeel (Wany Robotics, 2012). These programming interfaces were selected based on criteria such as programming language, supported operating systems, required level of user expertise in robotics and software license type. However, based on Universidad del Valle and Valle Departmental Library robotic seedbed experience, programming environments typically used in education have a very short life-cycle, suddenly ceasing to provide possibilities of more experiences for children or young people (Gobernación del Valle del Cauca & Universidad del Valle, 2006). By contrast, programming interfaces orientated towards research are more complicated to handle for someone lacking prior knowledge of robotics, and they are designed for audience prepared in robotics. Table 1 gives programming environments interacting with mobile robots teaching concepts related to mobile robotics and STEM (Kammer *et al*., 2011; Brauner *et al*., 2010; Eggert, 2009); however, some handle a single learner level (Kammer *et al*., 2011; Brauner *et al*., 2010). The programming environment life-cycle thus becomes shortened and prevents learners achieving more advanced knowledge and skills.

This paper proposes a programming environment involving three levels of complexity. It offered learners the opportunity to gain experience through a growing range of knowledge; the programming environment life-cycle thus became extended. This work dealt with mobile robotics because they represent a multidisciplinary field, promote teamwork and are an attractive field for developing scientific and technological knowledge. The three proposed programming environment levels were basic (for users lacking experience in robotics), intermediate (for users having previous robotics experience) and advanced (for users with previous knowledge in robotics). UV-bot mobile robots were used for testing, specifications being detailed in Gómez, Muñoz, Florian-Gaviria, Giraldo and Bacca-Cortes (2008). The proposed programming environment was multi-platform, developed with free software tools, extensible, had adaptable interfaces (Oppermann, Rashev and Kinshuk, 1997), persistence management and low software-hardware coupling (robot).

Table 1 Characterising educational programming environments and frameworks for mobile robotics

Environment	Mobile robot	Programming language	Operating system	User level	Licence
SR1 Explorer	SR1	BasicX	Windows	Middle	Commercial
Robolab	Block RCX 2.0	LabView	Windows, MAC	Basic	Commercial
Mindstorms 2.0		Gráfica	Windows, MAC	Basic	Commercial
BrickOS		C y C++	Linux, Windows	Middle	GNU
LeJOS		Java	Multiplatform	Middle	GNU
Mindstorms VDK		Java	Multiplatform	Middle	GNU
Torsen Sim.		Gráfica y C	PC, MAC, Android	Middle	GNU
BotStudio	Hemisson	Gráfica	Multiplatform	Basic	Commercial
Cricket Logo	Handy Crickets	Scripts	Windows, MAC	Basic	Commercial
WEBOTS	Khepera	C, C++, Matlab, Labview	Linux, Windows, MAC	Advanced	Commercial
	E-Puck	C, Matlab, Perl, Phyton, Player/Stage	Linux, Windows, MAC	Advanced	Commercial
ARIA	AmigoBot	C++, Java, Phyton	Multiplatform	Advanced	Commercial
	Pionneer 3DX	C++, Java, Phyton	Multiplatform	Advanced	Commercial
Pekee APIs	Pekee	C, C++, Java, C#, Matlab	Multiplatform	Advanced	Commercial

Design requirements

The programming environment proposed in this work was designed and implemented to offer users a continuous learning framework ranging from basic to advanced level and keeping software engineering complementary requirements in mind. The following sub-sections describe the firmware and programming environment requirements around eight cornerstones: mobility, perception, communications, programming and hands-on challenges (mobile robot) and usability, adaptability and data persistence (graphic user interface).

Basic complexity level

Mobility, perception, communications and programming functionalities include simple tasks at this level. However, they are focused on offering users overall knowledge of a robotic perception system and its actions regarding the environment.

Medium complexity level

Medium complexity includes basic level perception and mobility and allows users to deal with other concepts, such as 2D robot movement, distance-based and angle-based movements, communicating with other robots using IR modules and using arithmetic operations.

Advance complexity level

Basic and medium level requirements are used to build robot control architecture in the advanced complexity level, based on behaviour programming (Brooks, 1986). Sensor data are fed into perceptual schemas (Arkin, 1987); seven basic behaviours can be used individually or cooperatively in this work. A cooperative

Table 2. Basic complexity level requirements

Mobility	DC motors turn on / off Mobile robot wheels control speed Pre-programmed wheel turns.
Perception	IR and contact sensors acquire data Light sensors acquire data
Communications	Sounds and songs Transmitting and receiving messages from IR
Programming	XML schema for the mobile robot description Block-based programming (graphical programming) Graphical block configuration access Basic arithmetic operations Control flow blocks (while, if, for, repeat, etc.) (*) XML validation, programme compiling and downloading to the robot's memory Explicit switchin to XML programming (medium level user) Challenge selection and programming Challenge evaluation according to expected results
Examples	Basic programmes for finding light or dark places, following objects and avoiding obstacles
GUI usability and adaptability	Showing programming blocks sorted by type XML schema to adapt the tool palette Zoom controls in the graphical programming editor (*) Editing many programmes simultaneously (*) Showing user challenges and programming examples with their corresponding explanation (*) Once an example is selected, its corresponding main programme is shown
User management	To register, modify and delete users User authentication
Data persistence	(*) Open, save and close user programmes (*) XML-based schema for describing and storing user programmes (*) XML-based schema for describing and storing user data

(*) Requirement available for all levels of complexity

Table 3. Medium complexity level requirements

Mobility	• Distance-based and angle-based robot movement • Robot displacements and turns based on sensor state
Perception	• To obtain the XY robot position, orientation and current speed
Communications	• Transmitting and receiving ID codes using the robot IR modules
Programming	• XML-based programming • Maths, shift and logical operations support Building XSLT files to translate from the graphical programming interface to XML-based programming • Explicit switching to ANSI C programming (advanced level user)
Examples	• Programme to move the mobile robot to a fixed XY coordinate
GUI usability and adaptability	• Embedded XML editor including XML syntax highlighter, XML syntax corrector, and debugger

Table 4. Advanced complexity level requirements

Mobility	• Modifying the robot wheels' speed and the corresponding speed controller parameters
Perception	• Vector-based representation for IR and contact sensors • Obtaining a free-obstacle angle sector around the robot with more or less illumination • Obtaining a vector orientated towards a desired region of interest on the XY frame
Programming	• ANSI C programming structures and variables definition • Building XSLT files to translate from the XML-based programming interface to ANSI C-based programming
Examples	• Behaviour-based programming examples such as obstacle avoidance, noise adding, light-based homing, pose-based homing, wall following and escape • Cooperative coordinator based on a weighted sum of priorities to fuse behaviour responses
GUI usability and adaptability	• Embedded ANSI C editor including C syntax highlighter, C syntax corrector, and debugger
Mobility	• Modifying the robot wheels' speed and the corresponding speed controller parameters

coordinator was used to coordinate behaviour response (Arkin, 1987). Actuator schemas were used to modify the robot's wheel speed, robot IR modules and buzzer. Table 4 shows this level's additional requirements.

Architecture

Programming environment architecture was based on the model-view-controller (MVC) design pattern. This pattern separated logic and data, user interface and control actions into three units (Crawford & Kaplan, 2003). Figure 1 shows the component diagram for the programming environment where grey modules were part of the model, dark grey modules were part of the view and blank modules were part of the controller. The design was modular and extensible; it involved a combination of code written in Java and XML languages.

The *controller module* received a learner's requests; it then sent queries to the data model and decided what action to execute to build an appropriate view for a learner. When the programming environment was opened, the first view was the *learner management module* validating a learner through his/her access key and created new users. Once a learner had been validated, an adaptable interface was displayed; the *graphical programming module* was displayed for beginners, the *XML programming module* for intermediate learners and the *ANSI C programming module* for advanced learners. The *graphical programming module* had two complements displayed as pop-up windows: challenges and examples.

The final programme had to be written ANSI C to be compiled and downloaded to the robot. The environment was responsible for making a transparent translation of programmes when a learner used the *graphical programming* and *XML programming* modules. If a learner wished to view his/her programmes' translation code, the environment made it possible for beginners to the highest learning level. Successive actions regarding translation, compiling and programme download to the robot were performed, respectively by *translation module*, *compiler module* and *communication with robot module*. The *persistence management module* handled access to and creation and modification of stored data regarding configuration route, user data, programmes and challenges. The *robot & interface configuration module* provided views for selecting language and organising the tool palette according to the custom robot in use.

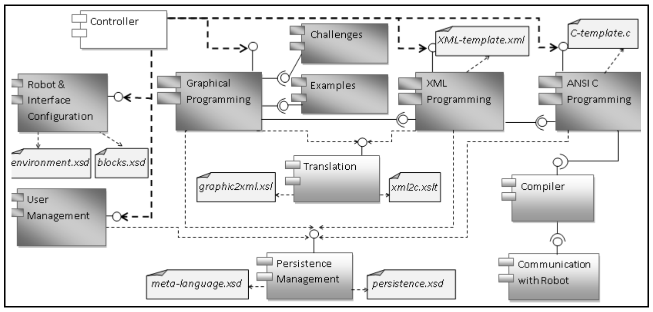


Figure 1. Diagram of programming environment components

XML schemes describe the structure and restrictions of XML documents having a high abstraction level, going beyond XML language syntactical norms (Fawcett, Ayers and Quin, 2012). XML schemes were joined to the environment architecture to support extension, adaptability, persistence and low coupling hardware / software regarding the programming environment. This programming environment had four XML schemes as its cornerstones.

Scheme for robot description (environment.xsd)

This scheme represented the robot's initial configuration for basic level programming. It defined available variables and functions and the robot's mechanical description (number and type of sensors, engine number, power, etc.). A library of functions was extensible because multiple robot descriptions could be created and the objective of low coupling software / hardware was achieved. The interface configuration presented adapted forms of functions and graphical appearance of the blocks from the data types described for function parameters. Table 5 presents the characterisation of this scheme's elements, sub-elements and attributes.

Table 5. Characterising the XML scheme environment.xsd

Element	Description
<variable-def> <variable> </variable> </variable-def>	These elements allowed enabling and setting a group of variables taking into account attributes such as name, data type, and initial value
<types-constrain> <constrain> <enum></enum> </constrain> </types-constrain>	These elements allowed describing restrictions on data types. Restrictions were visualised in each block's graphical configuration interface. Restrictions held attributes such as name, data type, range, minimum value, maximum value, associated graphical component and editing authorisation
<functions-def> <function> <param> </param> </function> </functions-def>	These elements defined the library of available functions in the graphical tool palette. Each function had attributes such as name, parameters, parameter data type and predefined parameter value These elements led to establishing a direct relation between input parameters and their graphical configuration

Graphic tool palette scheme (blocks.xsd)

This described all available functions on the graphic tool palette within the graphic programme environment. The user interface presented functions as graphic blocks. These blocks had the drag and drop interaction mechanism in the programming space. This scheme allowed personalising graphic elements by creating functional groups and setting colour, structure and input parameters. The tool palette was thus extensible. Table 6 presents the characterisation of this scheme's elements, sub-elements and attributes.

Table 6. Characterising XML scheme blocks.xsd

Element	Description
<group-block> <block></block> </group-block>	These elements described each block group on the graphic tool palette. Blocks were separated according to functionality. Each block had attributes such as background colour, block type, name of represented function, Boolean attribute to identify whether the block was a conditional one, and block icon

Data persistence scheme (persistence.xsd)

This scheme represented programming environment persistent data. It described demographic user information. It also defined challenges and examples available in the programming environment. Furthermore, it stated a learners' performance regarding challenges and examples. Table 7 presents the characterisation of this scheme's elements, sub-elements and attributes.

Scheme for BOT-XML meta-language (meta-language.xsd)

This scheme represented a programming meta-language called BOT-XML. Its meta-language was built for the programming environment middle level. BOT-XML was based on o: XML (Klang, 2007). BOT-XML had some simplifications for evaluating expressions calculating variable values. BOT-XML supported a greater number of data types than o:XML, such as int, distance, angle, direction, sensor, light, motor, speed, angular_speed, state, song, time, sensor_number.

Table 7. Characterising XML scheme persistence.xsd

Element	Description
<users> <user></user> </users>	These elements designated register users with data such as name, nickname, image and user role. The programming environment had two user roles: learners and teachers with administration authorisation
<challenges> <challenge> </challenge> </challenges>	These elements defined challenges. Each challenge had a name, a unique identification, a short description, a long description, some hints for supporting learners during solution, a video showing the desired solution and a questionnaire for learners who finished a challenge
<examples> <example> </ example > </ examples >	These elements explained examples (a collection of selected pairs of programmes). Each example had a name, a summary, a complete description of the solution programme and the path of the programme file to be displayed in the programming environment
<config> <config-path> </config-path> </config>	These elements held environment commands and configuration file paths. Each command had identification, a name, a type and a text field with the path file

Based on this scheme, learners' programming files were validated; programming files were thus saved in XML format. Translation took place at the end the programme to be downloaded to the mobile robot.

This scheme made it possible to build middle languages for each robot a user might want to use with the programming environment (i.e. the mechanism allowing extensibility and low coupling software/hardware). It also allowed the persistence definition for learner programmes. Table 8 presents this scheme's elements, sub-elements and attributes.

Table 8. Characterising the meta-language.xsd scheme (details of robot action functions have been skipped due to their extension)

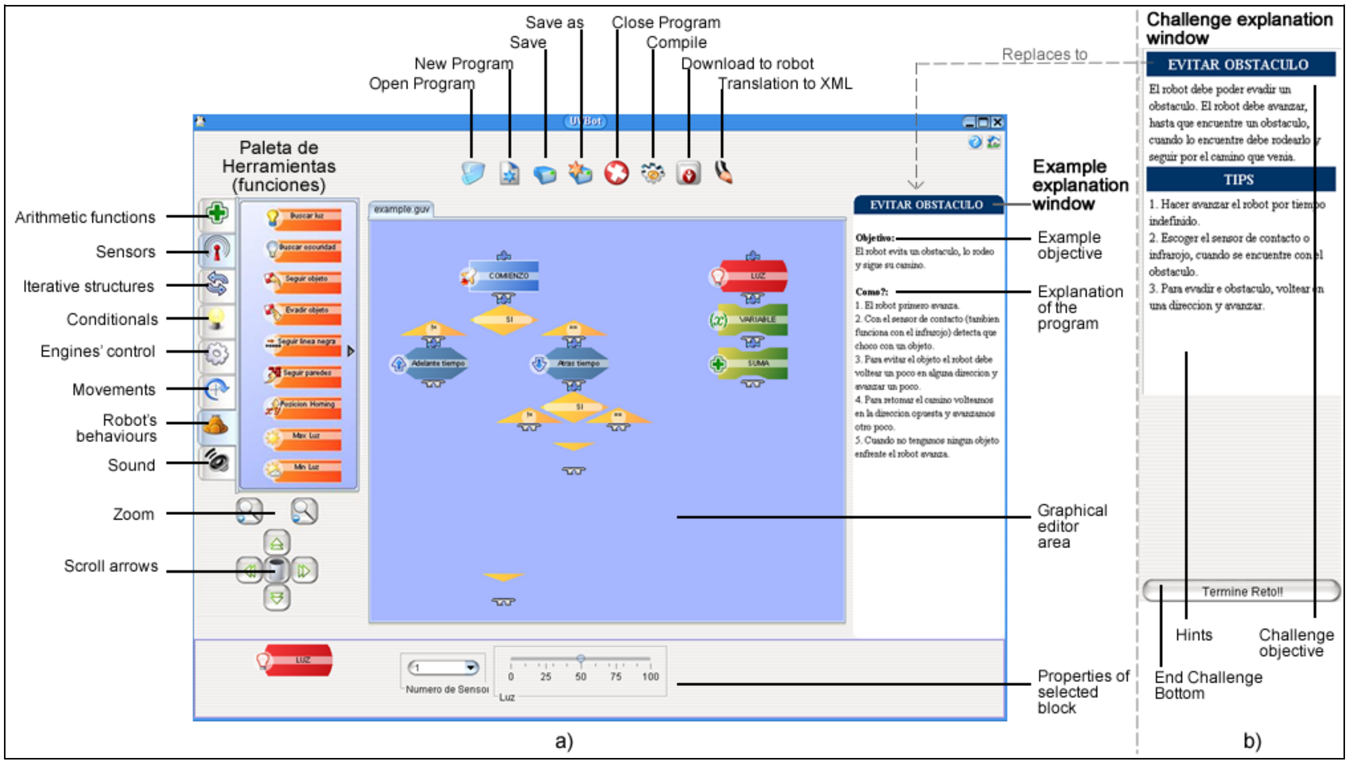
Element	Description
Arithmetic operation	
<add result="" var1="" var2=""/> <sub result="" var1="" var2=""/> <mult result="" var1="" var2=""/> <div result="" var1="" var2=""/> <sqrt result="" var1=""/> <square result="" var1=""/> <set name="" select=""/> <variable name="" type="" select=""/>	These elements defined procedures allowing simple arithmetic operations Each operation had input operators. The result was stored in the result attribute
Logical structure (logical operations on integers)	
<if var1="" op="" var2=""> </if>	Allowed arithmetic comparisons between two integer values: == equal, > greater, < less, >= greater equal, <= less equal, and != different
<choose var=""> <when test=""> </when> <else> </else> </choose>	Comparing an input value (var) and adding a condition to each possible value which could take that variable by using the element when
Iterative structures (allow repetitive sequences)	
<for from="" step="" to=""> </for>	A determined number of repetitions, controlled by an initial value (from) to a final value (to)
<while var1="" op="" var2=""> </while>	Making a number of repetitions controlled by an arithmetic condition

Translation between programming languages

XSLT transformations were used for translating programmes between programming languages (Tennison, 2005). Transformation files graphic2sml.xslt and xml2c.xslt collaborated in the task of code translation within programming environment architecture.

Results

Figure 2a shows basic GUI level programming and Figure 2b explains the challenge, involving the following tools: programming control flow blocks, graphical programming editor, zooming GUI controls, editor navigation GUI controls and programme editing tools (delete, add and programming block properties).



EVITAR OBSTACULO

Objetivo:
El robot evita un obstaculo, lo rodeo y sigue su camino.

Como?:
1. El robot primero avanza.
2. Con el sensor de contacto (tambien funciona con el infrarojo) detecta que choco con un objeto.
3. Para evitar el objeto el robot debe voltear un poco en alguna direccion y avanzar un poco.
4. Para retomar el camino volteamos en la direccion opuesta y avanzamos otro poco.
5. Cuando no tengamos ningun objeto enfrente el robot avanza.

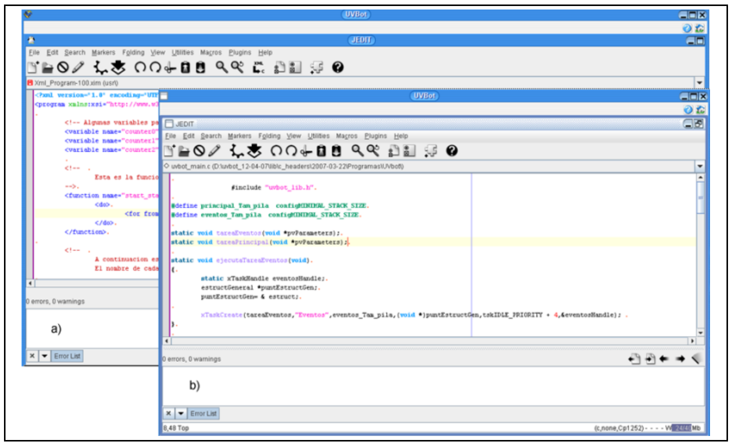
EVITAR OBSTACULO

El robot debe poder evadir un obstaculo. El robot debe avanzar, hasta que encuentre un obstaculo, cuando lo encuentre debe rodearlo y seguir por el camino que venía.

TIPS

1. Hacer avanzar el robot por tiempo indefinido.
2. Escoger el sensor de contacto o infrarojo, cuando se encuentre con el obstaculo.
3. Para evadir e obstaculo, voltear en una direccion y avanzar.

Figure 2. a) Basic level programming GUI showing a built-in exercise. b) Challenge explanation window


UVBot
JEDIT
File Edit Search Markers Folding View Utilities Macros Plugins Help
xml_Program-100.xim
uvbot_main.c (D:\uvbot_12-04-07\lib\c_headers\2007-03-22\Programas\UVbot)
0 errors, 0 warnings
Error List
8,48 Top
a)
b)

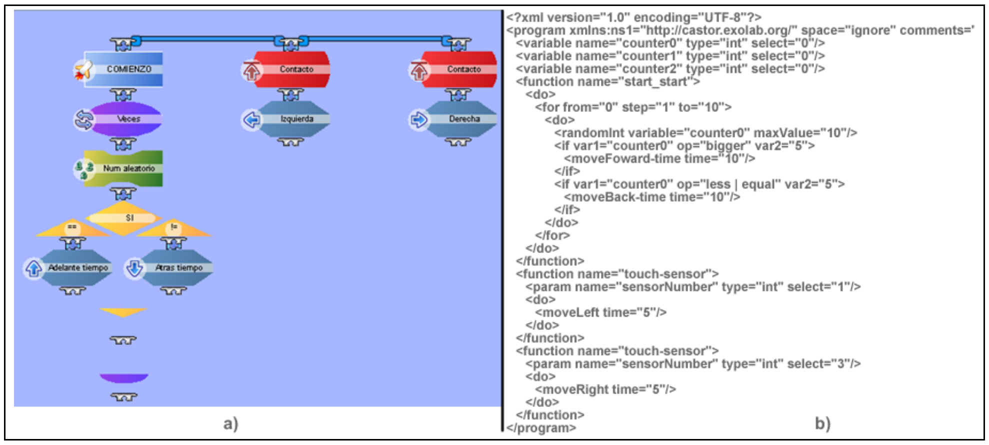
```
        #include "uvbot_lib.h".
.
#define principal_Tam_pila  configMINIMAL_STACK_SIZE.
#define eventos_Tam_pila  configMINIMAL_STACK_SIZE.
.
static void tareaEventos(void *pvParameters);.
static void tareaPrincipal(void *pvParameters);.
.
static void ejecutaTareaEventos(void).
{.
        static xTaskHandle eventosHandle;.
        estructGeneral *puntEstructGen;.
        puntEstructGen= & estruct;.
.
        xTaskCreate(tareaEventos,"Eventos",eventos_Tam_pila,(void *)puntEstructGen,tskIDLE_PRIORITY + 4,&eventosHandle); .
}.
```

Figure 3. a) XML programming GUI (medium level). b) ANSI C programming GUI (advanced level)

Figure 3a shows XML programming GUI where a XML editor was used. Using this XML programming GUI, the mobile robot could be programmed using BOT-XML meta-language. The XML editor included a console for syntax error log and another console for compiling results and showing programme download status.

The ANSI C programming GUI is shown in Figure 3b. An ANSI C editor was used in this GUI and included the most common ANSI C editor tools, a console where the compiling results and download status were shown. It started with a code template in which users had to add the C main programme and enable or disable sensor events.

Users could start programming at any level (basic, medium, advanced), but medium and advanced levels had important advantages: deeper access to mobile robot functionalities (robot behaviour, sensor events and perception schemas) and being able to add custom utilities.

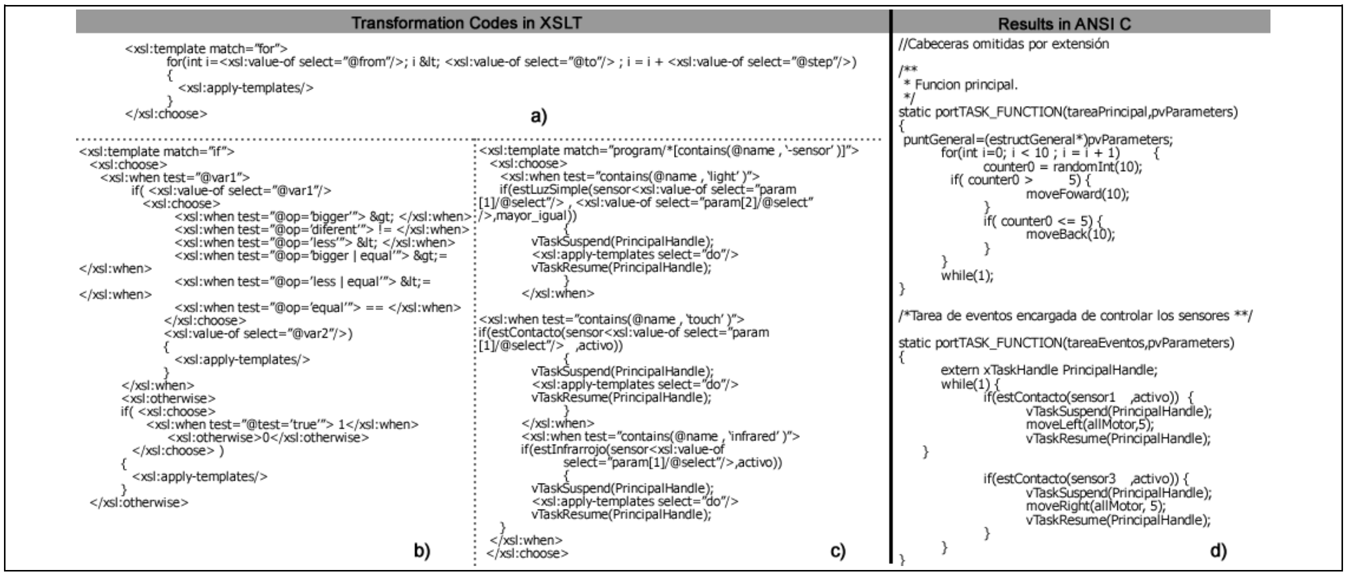


```
<?xml version="1.0" encoding="UTF-8"?>
<program xmlns:ns1="http://castor.exolab.org/" space="ignore" comments='
  <variable name="counter0" type="int" select="0"/>
  <variable name="counter1" type="int" select="0"/>
  <variable name="counter2" type="int" select="0"/>
  <function name="start_start">
    <do>
      <for from="0" step="1" to="10">
        <do>
          <randomInt variable="counter0" maxValue="10"/>
          <if var1="counter0" op="bigger" var2="5">
            <moveFoward-time time="10"/>
          </if>
          <if var1="counter0" op="less | equal" var2="5">
            <moveBack-time time="10"/>
          </if>
        </do>
      </for>
    </do>
  </function>
  <function name="touch-sensor">
    <param name="sensorNumber" type="int" select="1"/>
    <do>
      <moveLeft time="5"/>
    </do>
  </function>
  <function name="touch-sensor">
    <param name="sensorNumber" type="int" select="3"/>
    <do>
      <moveRight time="5"/>
    </do>
  </function>
</program>
```

Figure 4. Obstacle avoidance programme. a) Basic programming level (graphical code). b) Medium programming level

The tests were divided into two main parts. Translation between three programming languages was tested writing a programme using the graphical GUI; it was then transformed to its BOT-XML equivalent and translated to its ANSI C equivalent, downloaded and run on the mobile robot. A behaviour-based program was coded (available for the advanced level) to use and test all inherited functionalities from the basic and medium levels.

An obstacle avoidance application



Transformation Codes in XSLT

```
<xsl:template match="for">
        for(int i=<xsl:value-of select="@from"/>; i &lt; <xsl:value-of select="@to"/> ; i = i + <xsl:value-of select="@step"/>)
        {
          <xsl:apply-templates/>
        }
</xsl:choose>
```

```
<xsl:template match="if">
  <xsl:choose>
    <xsl:when test="@var1">
          if( <xsl:value-of select="@var1"/>
            <xsl:choose>
                  <xsl:when test="@op='bigger'"> &gt; </xsl:when>
                  <xsl:when test="@op='diferent'"> != </xsl:when>
                  <xsl:when test="@op='less'"> &lt; </xsl:when>
                  <xsl:when test="@op='bigger | equal'"> &gt;=
</xsl:when>
                  <xsl:when test="@op='less | equal'"> &lt;=
</xsl:when>
                  <xsl:when test="@op='equal'"> == </xsl:when>
                </xsl:choose>
                <xsl:value-of select="@var2"/>)
                {
                  <xsl:apply-templates/>
                }
        </xsl:when>
        <xsl:otherwise>
        if( <xsl:choose>
             <xsl:when test="@test='true'"> 1</xsl:when>
                 <xsl:otherwise>0</xsl:otherwise>
          </xsl:choose> )
        {
          <xsl:apply-templates/>
        }
  </xsl:otherwise>
```

```
<xsl:template match="program/*[contains(@name , '-sensor' )]">
  <xsl:choose>
    <xsl:when test="contains(@name , 'light' )">
    if(estLuzSimple(sensor<xsl:value-of select="param
[1]/@select"/> , <xsl:value-of select="param[2]/@select"
/>,mayor_igual))
                {
        vTaskSuspend(PrincipalHandle);
        <xsl:apply-templates select="do"/>
        vTaskResume(PrincipalHandle);
                }
      </xsl:when>

<xsl:when test="contains(@name , 'touch' )">
if(estContacto(sensor<xsl:value-of select="param
[1]/@select"/>   ,activo))
                {
        vTaskSuspend(PrincipalHandle);
        <xsl:apply-templates select="do"/>
        vTaskResume(PrincipalHandle);
                }
      </xsl:when>
      <xsl:when test="contains(@name , 'infrared' )">
      if(estInfrarrojo(sensor<xsl:value-of
              select="param[1]/@select"/>,activo))
                {
        vTaskSuspend(PrincipalHandle);
        <xsl:apply-templates select="do"/>
        vTaskResume(PrincipalHandle);
   }
 </xsl:when>
 </xsl:choose>
```

Results in ANSI C

```
//Cabeceras omitidas por extensión

/**
 * Funcion principal.
 */
static portTASK_FUNCTION(tareaPrincipal,pvParameters)
{
 puntGeneral=(estructGeneral*)pvParameters;
        for(int i=0; i < 10 ; i = i + 1)        {
                counter0 = randomInt(10);
          if( counter0 >        5) {
                        moveFoward(10);
                }
                if( counter0 <= 5) {
                        moveBack(10);
                }
        }
        while(1);
}

/*Tarea de eventos encargada de controlar los sensores **/

static portTASK_FUNCTION(tareaEventos,pvParameters)
{
        extern xTaskHandle PrincipalHandle;
        while(1) {
                if(estContacto(sensor1   ,activo))  {
                        vTaskSuspend(PrincipalHandle);
                        moveLeft(allMotor,5);
                        vTaskResume(PrincipalHandle);
    }

                if(estContacto(sensor3   ,activo)) {
                        vTaskSuspend(PrincipalHandle);
                        moveRight(allMotor, 5);
                        vTaskResume(PrincipalHandle);
                }
        }
}
```

Figure 5. BOT- XML to ANSI C translation. a) <for> tag translation, b) <if> tag translation. c) Sensor event translation. d) ANSI C result

was used to test translation between the three programming languages (this task is crucial in mobile robotics). Figure 4 shows the graphical code written using the basic level and its corresponding translation to BOT-XML language. The obstacle avoidance algorithm was pretty simple; it modified the left and right motor speed according to right or left contact sensor activation. Figure 4a shows three columns; the first is the main programme and the other show each sensor thread modifying the main programme's behaviour.

Considering the XML code shown in Figure 4b (basic programming level GUI) and the translation file called xml2c.xslt, defining translation instructions to ANSI C, Figure 5 shows examples of translation from BOT-XML to ANSI C for different programming tags. The <for> tag and its properties (from, step and to) are shown in Figure 5a; the corresponding translation from the <if> tag to the ANSI C if control flow sentence is shown in Figure 5b; Figure 5c shows all the sensor functionalities translated to conditional sentences within an ANSI C function controlling sensor events. Figure 5d shows the ultimate ANSI C code resulting from such translation.

The obstacle avoidance programme was then downloaded into the robot memory and the resulting robot path along the experiment is shown in Figure 6. This robot path was extracted off-line using a digital image processing tool specifically developed for this project. Two obstacle configurations were tested: a maze-based environment and a corridor.

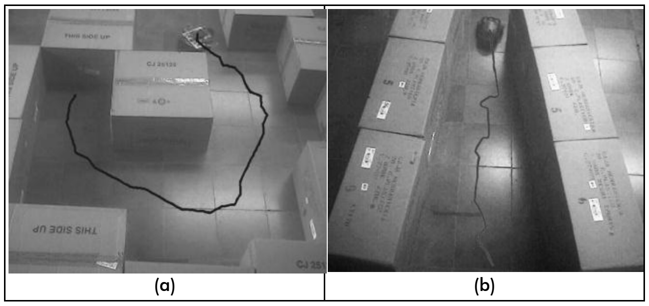


Figure 6. Robot paths in different environmental configurations. a) Maze. b) Corridor.

The second batch of tests was focused on using and testing all inherited functionalities from basic and medium levels achieved by programming the mobile robot in advanced level, particularly using the behaviour-based application. Homing is a typical behaviour in mobile robotics whose main goal is orientating the mobile robot towards a region of interest. This behaviours can be combined with other behaviours such as obstacle avoidance and emergent behaviours thus becomes more complex (Arkin, 1989). In our case, the region of interest was defined using a metric position in 2D (x, y) or using an environm kinds of behaviour.

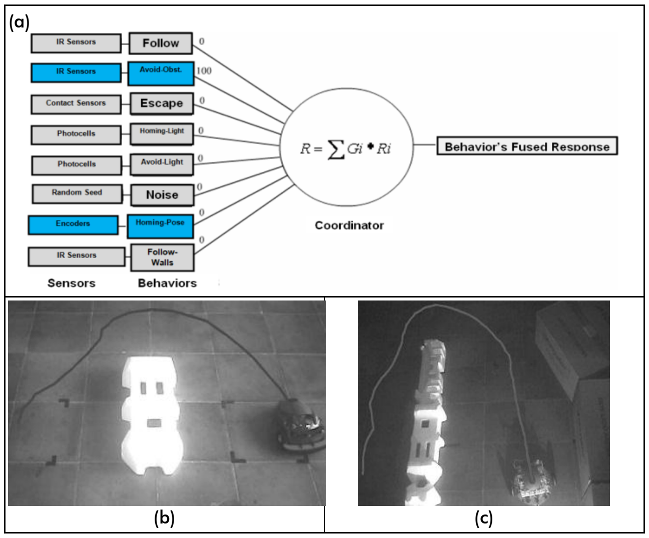


Figure 7. Position-based and light-based homing behaviour and obstacle avoidance. a) Mobile robot behaviour stack. b) Robot path for position-based homing. c) Robot light-based homing path.

- Position-based homing and obstacle avoidance (Figure 7a) shows the behaviour stack where homing and obstacle avoidance were only enabled. Homing behaviour aimed at an XY position (100cm, 0cm) in the overall framework. The mobile robot was stopped at the point when it approached a ~5cm region. Figure 7b shows the robot path using a digital image processing tool developed for this project.

- Light-based homing and obstacle avoidance. The only behaviours changed in the behaviours stack was homing behaviour, changing the criteria used to define the region of interest to sense light intensity. It is worth noting that the mobile robot did not know the stimulus position in advance, nor the obstacles' distribution around the environment. The perceptual schema obtained high light intensity orientation using an array of photocells. It was assumed that the mobile robot had a line of sight towards the light source. The resulting robot path is shown in Figure 7c.

Conclusions and future research work

This work has shown a programming environment having three complexity levels to enable users to experiment with mobile robotics using differing degrees of knowledge. Compared to other mobile robot learning tools considering only one user profile, this work had the advantage of extending the lifetime of the programming environment and the hardware platform. Not only concept-based mobile robotics can be learnt, but also hands-on STEM-based concepts.

The programming environment proposed in this work, which was implemented using the MVC pattern, provided a set of important properties such as a training module for inexperienced users, exercises at different levels of complexity (basic, medium and advanced), basic level programming GUI orientated towards users lacking previous experience in robotics, medium level XML-based GUI programming and advanced level programming GUI based on ANSI C, extensibility, interface adaptability, data persistence, and low software/hardware coupling due to BOT-XML language. This work was able to translate programmes at three complexity levels using XSLT translations.

Considering robotic seedbed experience (Jimenez Jojoa *et al.*, 2010) and current academic and governmental efforts to develop novel educational strategies (European Center for the Development of Vocational Training - CEDEFOP, 2010), the work presented here represents an interesting option for stimulating children and young people's science, technology, engineering and maths learning.

A challenge management module should be developed to evaluate user learning. Simulating mobile robot behaviour is an important tool prior to downloading the application into a mobile robot. Adding remote mobile robot operation and programming environment represents an interesting direction for future work in this field.

References

Adept MobileRobots. ARIA. Retrieved from http://robots.mobile robots.com/wiki/ARIA#Download_Aria. 2012

Arkin, R., Motor schema based navigation for a mobile robot: An approach to programming by behavior. IEEE International Conference on Robotics and Automation, Vol. 4, 1987, pp. 264 – 271.

Brauner, P., Leonhardt, T., Ziefle, M., & Schroeder, U. (). The Effect of Tangible Artifacts, Gender and Subjective Technical Competence on Teaching Programming to Seventh Graders. In J. Hromkovič, R. Královič, & J. Vahrenhold (Eds.), Teaching Fundamentals Concepts of Informatics, Vol. 5941, 2010, pp. 61–71. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-11376-5

Brooks, R. A., A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation, RA-2, Vol. 1, 1986, pp. 14–23.

Crawford, W., & Kaplan, J., J2EE Design Patterns. O'Reilly Media, Inc. 2003

Cyberbotics Ltd., Webots: the mobile robotics simulation software. Retrieved from http://www.cyberbotics.com/. 2012

Eggert, D. W. (). Using the Lego Mindstorms NXT Robot Kit in an Introduction to C Programming Class. Journal of Computing Sciences in Colleges, Vol. 24, No. 6, 2009, pp. 8–10.

European Center for the Development of Vocational Training (CEDEFOP). Skills for Green Jobs (European Synthesis Report). 2010.doi:10.2801/31554

Fawcett, J., Ayers, D., & Quin, L. R. E., Beginning XML (5th Editio.). John Wiley & Sons, Inc. 2012

Gobernación del Valle del Cauca, & Universidad del Valle. (). Curso "Semillero de Robótica" en la Biblioteca Departamental. Retrieved from http://www.valledelcauca.gov.co/publicaciones.php?id=1089, 2006.

Gómez, F., Muñoz, F., Florian-Gaviria, B., Giraldo, C. A., & Bacca-Cortes, E. B., Diseño y prueba de un robot móvil con tres niveles de complejidad para la experimentación en robótica. Ingeniería y Competitividad, 10(2), 53–74. Retrieved from http://ingenieria.univalle.edu.co:8000/revistaingenieria/index.php/inycompe/article/view/151. 2008

Handyboard., Cricket Logo. Retrieved from http://handyboard.com/cricket/program/, 2009

INTPLUS., SR1 Robot Movil Multifuncional. Retrieved from http://www.superrobotica.com/sr1_Robot.htm, 2011

Jimenez Jojoa, E. M., Caicedo Bravo, E., & Bacca-Cortes, E. B. Tool for Experimenting With Concepts of Mobile Robotics as Applied to Children's Education. IEEE Transactions on Education, Vol. 53, No. 1, 2010, pp. 88–95. doi:10.1109/TE.2009.2024689

K-Team Corporation., BotStudio. Retrieved from http://www.k-team.com/mobile-robotics-products/hemisson, 2011.

Kammer, T., Brauner, P., Leonhardt, T., & Schroeder, U., Simulating LEGO Mindstorms Robots to Facilitate Teaching Computer Programming to School Students. In C. D. Kloos, D. Gillet, R. M. Crespo García, F. Wild, & M. Wolpers (Eds.), Towards Ubiquitous Learning, Vol. 6964, 2011, pp. 196–209. Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-23985-4

Klang, M., o:XML - object-oriented XML. Retrieved from http://www.o-xml.org/about/, 2007

Koebler, J. (). D.C., Maryland and Washington State Hold Highest Concentration of STEM Jobs. Retrieved from http://www.usnews.com/news/blogs/stem-education/2011/09/30/dc-maryland-nd-washington-state-hold-highest-concentration-of-stem-jobs, 2011

Noga, M. L., BrickOS. Retrieved from: http://brickos.sourceforge.net/, 2004

Oppermann, R., Rashev, R., & Kinshuk, Adaptability and adaptivity in learning systems. Knowledge transfer, 2, 1997, pp. 173–179. Retrieved from:http://torcaza.uis.edu.co/~clarenes/docencia/3501MO0660/pdfs/adaptability-adaptivity-kinshuk.pdf

Sang-Rok, O., The real facts about Korea's kindergarten teaching robots. Retrieved from http://www.everything-robotic.com/2011/03/real-facts-about-koreas-kindergarten.html, 2011.

Solorzano, J., leJOS, Java for LEGO Mindstorms. Retrieved from http://lejos.sourceforge.net/links.php, 2012

Tennison, J., Beginning XSLT 2.0: From Novice to Professional. C. Mils, Ed., Apress, 2005.

The LEGO Group., RoboLab. Retrieved from http://www.legoengineering.com/robolab-submenusupport-141.html, 2011

The LEGO Group., LEGO MINDSTORMS. Retrieved from http://mindstorms.lego.com/en-us/products/default.aspx, 2012

Ulloa, G., ¿Qué Pasa con la Ingeniería en Colombia? Interacción, Vol. 28, No.4, 2008, pp. 3–5. Retrieved from http://www.icesi.edu.co/revista_interaccion/edicion042007/index.htm

Wany Robotics., Pekee Mobile Robot Platform. Retrieved from http://www.wanyrobotics.com/pekeel.html, 2012