Original research / Artículo original / Pesquisa original - Tipo 1

Programmable mobile robots as hands-on platform for basic programming

Sergio García[1] / sergio.garcia@correounivalle.edu.co

Sebastián Rueda[2] / sebastian.rueda@correounivalle.edu.co

Beatriz Florián-Gaviria[1] / beatriz.florian@correounivalle.edu.co

Bladimir Bacca-Cortés[2] / bladimir.bacca@correounivalle.edu.co

[1]Escuela de Ingeniería de Sistemas y Computación, Universidad del Valle, Cali-Colombia

[2]Escuela de Ingeniería Eléctrica y Electrónica, Universidad del Valle, Cali-Colombia

ABSTRACT Nowadays, mobile robots platforms are being used in different education contexts. The state of the art shows that 197 papers have been published in this area knowledge over ten years. Latin America faces a problem regarding the enrolled students in engineering programs. The SPADIES program (Colombia) affirms that the lack of motivation and interaction with real artifacts relating theory and practice is an important aspect for dropout. In this work, a platform composed by a set of programmable mobile robots, and a WEB-responsive software tool for programming at different levels of knowledge were implemented. The set of mobile robots were implemented with proximity, trajectory, light, inertial, and vision sensors; also, tools such as Bluetooth and LEDs-ring are included; and, a mechanical support for an erasable marker. The WEB-responsive tool supports graphical programming for novice, Python programming for middle, and ANSI-C for advanced level learners.

KEYWORDS Mobile robots, engineering education, Web-responsive.

Robots móviles programables como una plataforma hands-on para programación básica

RESUMEN Las plataformas robóticas móviles se usan en diferentes contextos de la educación. El estado del arte muestra 197 artículos publicado al respecto en los últimos diez años. Latinoamérica enfrenta problema de baja matrícula de estudiantes de ingeniería. En Colombia, el programa SPADIES destaca a la ausencia de motivación e interacción con artefactos reales como un aspecto importante para abandonar los estudios. En este trabajo se reporta la implementación de una plataforma compuesta por un set de robots móviles y de herramientas software web-responsives para programación a diferentes niveles de conocimiento. La serie de robots se implementó con sensores de proximidad, trayectoria, iluminación, inercia y visión, e incluyó herramientas como bluetooth y LEDs-ring y un soporte mecánico para un marcador borrable. Las herramientas web-responsives soportan la programación básica para novatos, mientras que Python y ANSI-C hacen lo propio para estudiantes de los niveles intermedio y avanzado.

PALABRAS CLAVE Robots móviles, educación en ingeniería, web-reponsive.

Robôs móveis programáveis como uma plataforma hands-on para programação básica

RESUMO As plataformas robóticas móveis são usadas em diferentes contextos de ensino. O estado da arte mostra que 197 artigos foram publicados nesta área de conhecimento em um período de dez anos. A América Latina enfrenta um problema de baixa de inscrição de estudantes de engenharia. Na Colômbia, o programa SPADIES destaca como a falta de motivação e interação com artefatos reais relacionados com a teoria e prática é um aspecto importante na hora de abandonar os estudos. Este artigo descreve a implementação de uma plataforma composta de um conjunto de robôs móveis e de ferramentas software web-responsives para programação em diferentes níveis de conhecimento. Os robôs móveis foram implementados com sensores de proximidade, trajetória, iluminação, inércia e visão, sendo incluídas ferramentas como bluetooth e LEDs-ring e um suporte mecânico para um marcador apagável. As ferramentas web-responsives suportam a programação básica para novatos, enquanto Python e ANSI-C fazem o mesmo para alunos de níveis intermediário e avançado.

PALAVRAS-CHAVE Robôs móveis, ensino de engenharia, web-responsive.

Received / Recepción: March 21, 2017 - Accepted / Aceptación: April 30, 2017 *doi: 10.18046/syt.v15i41.2455*

I. Introduction

Robotics is a valuable learning resource which allows students building their own concepts on Science, Technology, Engineering and Math [STEM]. The main idea of using robotics on education is learning STEM concepts through hands-on learning activities (Cristóforis et al., 2013), and in this way encouraging the team work within a multi-disciplinary environment. Robotics on education has been becoming an interesting research topic over last years, in Benitti (2012) and Major, Kyriacou and Brereton (2011) report 197 different scientific works over the last 10 years. These works show that robotics on education is an important tool to improve the learning process, not only at secondary level but at Universities.

Nowadays, Latin America faces a serious problem of students enrollment into engineering programs (Ministerio de Educación, 2014). In Colombia, according with the Ministry of Education statistics (Mineducación, 2014) the Engineering graduate student to population ratio is 0.0005364 which means 1 Engineering student for 1864 people. In addition, these statistics show that only the 5% of enrolled students in Engineering programs finish their studies. These statistics are an important concern of the Ministry of Education, which has implemented the Desertion Prevention System for High Education [SPADIES]. An important observation made by this program was focused on lacking of student interaction with real learning artifacts into the first semesters of engineering programs.

Therefore, this work is focused to develop a platform composed by a set of mobile robots, and a development environment with the aim of allowing students more interaction with real learning artifacts using programming languages with increasing complexity.

A. Related works

Currently, many mobile robotic platforms are available and ready to use in engineering education. A summary of those more representative platforms is shown in **Table 1**. This table makes a comparison considering the development environment used, the programming languages supported, the operative system, the learner's level of knowledge required, and the type of licensing. The latter is essential to guarantee long-term use of the learning platform. Observing **Table 1**, most platforms strictly constraint the type of users able to deal with the mobile robot, i.e. basic or advanced or intermediate. In addition, most of them have commercial licenses, which could limit their use. **Table 1** also shows there

I. Introducción

La robótica es un valioso recurso de aprendizaje que permite a los estudiantes construir sus propios conceptos en Ciencia, Tecnología, Ingeniería y Matemáticas [*Science, Technology, Engeneering and Mathematics, STEM*]. La idea principal del uso de la robótica en la educación es el aprendizaje de conceptos de STEM a través de actividades de aprendizaje práctico (Cristóforis et al., 2013), para de esta manera fomentar el trabajo en equipo en un entorno multidisciplinario. La robótica en la educación se ha convertido en un tema interesante de investigación en los últimos años. Benitti (2012) y Major, Kyriacou y Brereton (2012) reportan 197 diferentes trabajos científicos en los últimos diez años. Estos trabajos muestran que la robótica es una herramienta importante para mejorar el proceso de aprendizaje, no sólo a nivel de secundaria, sino también universitario.

Hoy en día, América Latina se enfrenta a un grave problema con respecto del número de estudiantes que se matriculan en programas de ingeniería (Ministerio de Educación, 2014). En Colombia, de acuerdo con las estadísticas del Ministerio de Educación Nacional (MEN2014), la proporción de estudiantes graduados de ingeniería es de 0,0005364, lo que significa un estudiante de ingeniería por cada 1.864 personas. Las estadísticas muestran además que sólo el 5% de estudiantes matriculados en los programas de ingeniería terminan sus estudios. Estas estadísticas representan una preocupación importante para el MEN, quien ha implementado el Sistema para la Prevención de la Deserción de la Educación Superior [SPADIES]. Una observación importante, hecha por este programa, se centra en la falta de interacción del estudiante con artefactos de aprendizaje reales durante los primeros semestres de los programas de ingeniería.

En vista de lo anterior, el presente artículo se enfoca en el desarrollo una plataforma compuesta por un conjunto de robots móviles y un entorno de desarrollo, que tiene como objetivo permitir a los estudiantes un mayor nivel de interacción con artefactos de aprendizaje reales usando lenguajes de programación con una complejidad creciente.

A. Trabajos relacionados

Actualmente, muchas plataformas robóticas móviles están disponibles y listas para su uso en la formación en ingeniería. En la **Tabla 1** se muestra un resumen de las más representativas, en ella se hace una comparación considerando el entorno de desarrollo usado, los lenguajes de programación permitidos, el sistema operativo, el nivel de conocimiento requerido del estudiante y el tipo de licencia. Esta última es esencial para garantizar el uso a largo plazo de la plataforma de aprendizaje.

Observando la **Tabla 1**, es evidente que la mayoría de las plataformas restringen de manera estricta el tipo de usuarios capaces de tratar con el robot móvil, es decir, básico, avanzado o intermedio. Además, la mayoría de ellos tienen licencias comerciales, lo que, como se indicó, puede

Table 1. State of the art summary of mobile robotics platforms for education and research / Resumen del estado del arte de las plataformas robóticas para educación e investigación

Mobile robot / *Robot móvil*	**Development environment** / *Ambiente de desarrollo*	**Programming languages** / *Lenguajes de programación*	**Operative systems**/ *Sistemas operativos*	**Learner level** / *Nivel de aprendizaje*	**License**/ *Licencia*
Hemisson (K-Team, 2016)	BotStudio	Graphic	Multiplatform	Advanced	Commercial
Khepera (K-Team, 2016)	WebBots	C, C++ Matlab, Labview	Linux, Windows, Mac	Advanced	Commercial
E-Puck (K-Team, 2016)	WebBots	C, Matlab	Linux, Windows, Mac	Advanced	Commercial
AmigoBot (MobileRobots, 2013)	ARIA	C++, Java Phyton	Multiplatform	Advanced	Commercial
Pioneer 3DX (MobileRobots, 2013)	ARIA	C++, Java Phyton	Multiplatform	Advanced	Commercial
iCreate (Corporation, 2013b)	ROS	C++, Java, Python	Multiplatform	Advanced	Commercial
Pob-bot (Corporation, 2013a)	RisBee	Graphic, Basic and C	Multiplatform	Basic / intermediate / advanced	Commercial
ArduinoBot (Arduino, 2013)	Arduino IDE	Arduino	Multiplatform	Intermediate	GNU
Thymio II(Aseba, 2013)	Aseba Studio	Graphic and Aseba	Multiplatform	Basic / Intermediate	Commercial
Robotino (Festo, 2013)	Robotino API	C, C++, Java, Labview	Multiplatform	Advanced	Commercial
The Finch (BirdBrain-Technologies, 2016)	Finch Dreamse	Graphic, Java, Python, Scratch, C, C++, Matlab, VBasic, Labview…	Linux, Windows, Mac	Basic / Intermediate / advanced	GNU
Pololu M3Pi (Pololu-Corporation, 2016)	AVR IDE	C, C++, Arduino	Linux, Mac, Windows	Intermediate / advanced	GNU
mBot (MakeBlock, 2016)	mBlock Software	Graphic, Scratch, Arduino	Windows, Mac	Basic	Commercial
Edison (Meet-Edison, 2016)	EdWare	Graphic	Linux, Windows, Mac	Basic	GNU
Scribbler 2 (Parallax, 2014))	S2 GUI	Graphic, Basic, C	Linux, Mac, Windows	Basic / Intermediate	GNU
TurtleBot2 (Open-Source-Robotics-Foundation, 2016)	TurtleBot API	C, C++, Python	Linux, Mac, Android	Advanced	GNU
MarxBot (Boanani et al., 2016)	Aseba Studio	Graphic, Scratch, Aseba	Linux, Mac, Windows	Advanced	Commercial
K-Junior (K-Team, 2016)	AVR IDE	C, C++	Windows	Advanced	Commercial

limitar su uso. La **Tabla 1** también muestra que hay dos posiciones marginales, la plataforma requiere de estudiantes avanzados o de los básicos. Las plataformas robóticas avanzadas normalmente tienen un proceso lento de aprendizaje, pero por su parte las plataformas robóticas básicas se convierten en juguetes caros, una observación importante hecha en el Mobile Robotics Seed Bed desarrollado por el grupo de investigación Percepción y Sistemas Inteligentes (Jimenez, Caicedo, & Bacca-Cortes, 2010).

Recientemente, plataformas como Pob-Bot (Awabot, 2013), Thymio 2 (Aseba, 2013), The Finch (BirdBrain-Technologies, 2016) y UVBots1 (Gómez, Muñoz, Florián, Giraldo y Bacca-Cortés, 2008; Giraldo, Florian, Bacca, Gómez, & Muñoz, 2012) ofrecen a los usuarios propiedades como: soporte multiplataforma, programación en varios lenguajes y licencias GNU.

Hoy en día los conceptos STEM, específicamente los conceptos básicos de programación, también se aprenden

exists two marginal positions, the platform require advanced learners, or basic ones. Advanced robotic platforms normally have a slow process of learning; however basic robotic platforms turn into expensive toys. Which is an important observation made in the Mobile Robotics Seed Bed developed by the Perception and Intelligent Systems research group (Jimenez, Caicedo, & Bacca-Cortes, 2010).

Recently, platforms such as Pob-Bot (Awabot, 2013), Thymio 2 (Aseba, 2013), The Finch (BirdBrain-Technologies, 2016), and UVBots1 (Gómez, Muñoz, Florián, Giraldo, & Bacca-Cortés, 2008; Giraldo, Florian, Bacca, Gómez, & Muñoz, 2012) offer users properties such as multiplatform support, programming using various languages, and GNU licensing.

Today, STEM concepts and specifically basic programming concepts are also learnt using WEB applications, and

on-line mini-course. Most of them do not need a real world platform to execute those programs implemented by learners; then, software applications which use robotic platforms are described first, afterwards those applications intended for teaching programming only. RoboLab (LEGO, 2016) is a free graphical programming environment which supports RCX 2.0 and NXT Lego bricks, this software need those robotic platforms to execute the implemented program. Bot-Studio (K-Team, 2016) is desktop application which uses GRAFCET language to program Hemisson and K-Junior robots, it is a licensed application, and it is only distributed buying these robots. Scribbler GUI (Parallax, 2014) and Risbee (Awabot, 2016) are free graphical programming environments for the Scribbler2 and POB robots respectively, they are desktop and platform dependent applications. Scratch (MIT-Media-Lab, 2016) is a free graphical programming tool to introduce learners to STEM concepts using images, sounds and multimedia resources, and then, to create animated stories. CODE (Code.org, 2016) is a mini-course of twenty hours where learners acquire the fundamental concepts of computer science using animations and popular movies or game characters for kids, depending of the learner knowledge level there are many mini-course to choose. App Inventor (MIT, MIT-Media-Lab, & MIT-CSAIL, 2016) is an innovative programming environment for beginners app developers, it uses drag-and-drop blocks to create simple apps for Android SO. It is worth noting that all these software applications are oriented only to one user profile, they do not support an increasing level of knowledge complexity.

This work proposes a hands-on platform composed by a set of programmable mobile robots, and a WEB responsive development environment; both of them, supporting increasing level of complexity in order to include learners with any or previous knowledge on mobile robotics or programming concepts; supporting multiplatform and GNU licensing; and with the aim of having a mobile robot as real artifact to learn STEM concepts at early semesters of engineering programs.

B. System configuration

Figure 1a shows the proposed platform. Here, eight mobile robots are able to be programmed considering three levels of complexity depending on the learner expertise. The mobile robots support Bluetooth communication each other, but they are programmed using a PC or Laptop through a wired serial link. At the PC or Laptop runs a WEB responsive application which holds a development environment considering three levels of complexity. From the programming point of view, these levels of complexity offer to learners program-

usando aplicaciones WEB, y asistiendo a mini cursos en línea. La mayoría de ellos no necesitan una plataforma de mundo real para ejecutar los programas implementados por los estudiantes; entonces, las aplicaciones de software que usan plataformas robóticas se describen primero, y luego las aplicaciones destinadas específicamente a la programación de enseñanza. RoboLab (LEGO, 2016) es un entorno de programación gráfico gratuito que soporta RCX 2.0 y ladrillos Lego NXT; este software necesita de esas plataformas robóticas para ejecutar el programa implementado. Bot-Studio (K-Team, 2016) es una aplicación de escritorio que usa el lenguaje GRAFCET para programar los robots Hemisson y K-Junior, es una aplicación con licencia que sólo se distribuye comprando estos robots. Scribbler GUI (Parallax, 2014) y Risbee (Awabot, 2016) son entornos de programación gráfica gratuitos para los robots Scribbler2 y POB respectivamente, son aplicaciones dependientes del escritorio y la plataforma. Scratch (MIT-Media-Lab, 2016) es una herramienta de programación gráfica gratuita, útil para introducir a los estudiantes en los conceptos STEM usando imágenes, sonidos y recursos multimedia, para luego crear historias animadas. CODE (Code.org, 2016) es un mini curso de veinte horas en el cual los estudiantes adquieren los conceptos fundamentales de la informática usando animaciones y películas populares o personajes de juego para niños, dependiendo del nivel de conocimiento del estudiante, hay muchos mini cursos disponibles para elegir. App Inventor (MIT, MIT-Media-Lab, & MIT-CSAIL, 2016) es un innovador entorno de programación para desarrolladores de aplicaciones principiantes, usa bloques de arrastrar y soltar para crear aplicaciones simples para Android SO. Vale la pena señalar que todas estas aplicaciones de software están orientadas solamente a un perfil de usuario, no soportan un nivel creciente de complejidad del conocimiento.

Este trabajo propone una plataforma práctica compuesta por un conjunto de robots móviles programables y un entorno de desarrollo adaptativo WEB [*Web responsive*], ambos apoyan el nivel creciente de complejidad, con el fin de incluir a los estudiantes con conocimientos previos sobre robótica móvil o conceptos de programación y permiten las licencias multiplataforma y GNU, con el objetivo de tener un robot móvil como artefacto real para aprender conceptos STEM en los primeros semestres de los programas de ingeniería.

B. Configuración del sistema

La **Figura 1a** muestra la plataforma propuesta. Aquí, ocho robots móviles se pueden programar considerando tres niveles de complejidad, dependiendo de la experiencia del estudiante. Los robots móviles permiten la comunicación Bluetooth entre sí, pero son programados usando un computador, de escritorio o portátil, a través de un enlace serial por cable. En el computador se ejecuta una aplicación adaptativa WEB que tiene un entorno de desarrollo que considera tres niveles de complejidad. Desde el punto

de vista de la programación, estos niveles de complejidad ofrecen a los estudiantes una programación usando interfaces gráficas (nivel básico), Python (nivel intermedio) y C (nivel avanzado). En todos los niveles de complejidad, los estudiantes son capaces de usar todas las herramientas de percepción, movimiento, comunicación e interacción de los robots móviles. Incluye: seis sensores de proximidad infrarrojos, cuatro parachoques, cuatro fotoceldas, un sensor de línea, una Unidad de Medición Inercial [IMU] de nueve grados de libertad [DOF], un PixyCam, dos codificadores incrementales de 562 señales por revolución, dos motores DC en configuración diferencial, un dispositivo de comunicación Bluetooth, un zumbador y ocho LEDs de color.

II. Descripción del hardware de robots móviles

Cada uno de los ocho UVBots2 se diseñó por igual desde las perspectivas de hardware y firmware. En esta sección se describe brevemente el diseño del hardware (Sección I.A) y sus módulos como: la percepción (Sección I.B), el movimiento (Sección I.C) y la comunicación (Sección I.D).

A. Requerimientos de diseño

La plataforma de hardware de UVBots2 fue concebida considerando el diagrama de bloques mostrado en la **Figura 1b**. El robot móvil UVBots2 tiene los siguientes requisitos de hardware: el sistema de potencia, que consta de un paquete de baterías recargables de 12V y 1800mAh y dos subsistemas de regulación de potencia, la electrónica del robot y el sistema de movimiento; el sistema de movimiento, accionamiento diferencial, dos motores de accionamiento con 50oz/pulg (Pololu 6VDC) y dos codificadores incrementales con 562 señales por revolución; el sistema de percepción, cuatro parachoques, seis sensores de proximidad infrarrojos (GP1UM28YK), cuatro fotoceldas, una IMU (MiniIMU9v3), un sensor de línea y una cámara (CMUcam5); el sistema de comunicación y retroalimentación del usuario, comunicación PC-Robot mediante RS-232, comunicación Robot-Robot mediante Bluetooth (HC-05), un zumbador y ocho LEDs de color; y la Unidad Central de Proceso [CPU], una ATMEGA644-20PU corriendo FreeRTOS 7.4.2 (Barry, 2016).

En la **Figura 1c** se muestra la estructura mecánica del robot UVBots2, la cual tiene los siguientes requisitos: una atractiva carcasa con diferentes colores para encapsular un chasis metálico, la batería, los motores y la electrónica del hardware; tres botones superiores para encender/apagar, reiniciar y cambiar el modo de ejecución del robot móvil; un soporte para manejar un marcador en la parte inferior del robot; y tres conectores para la comunicación de PC-Robot y recarga de la batería.

B. Sistema de movimiento

Los robots móviles UVBots2 tienen un modelo cinemático diferencial (ver Ecuación 1), que toma en cuenta el espacio de velocidad del robot.

ming using graphical (basic level), Python (intermediate level) and C (advanced level) interfaces. Within all levels of complexity, learners are able to use all the perception, motion, communication and interaction tools from the mobile robots. This includes: six infrared proximity sensors, four bumpers, four photocells, one line sensor, 1 Inertial Measurement Unit [IMU] of nine Degrees of Freedom [DOF], one PixyCam, two incremental encoders of 562 ticks per revolution, two dc motors in differential drive configuration, one Bluetooth communication device, one buzzer, and eight colored LEDs.

II. Mobile robots hardware description

Each one of the eight UVBots2 was equally designed from hardware and firmware point of view. This section briefly describes hardware design (Section I.A) and its modules such as perception (Section I.B), motion (Section I.C) and communication (Section I.D).

A. Design requirements

The UVBots2 hardware platform was conceived considering the block diagram shown in **Figure 1b**. The UVBots2 mobile robot has the following hardware requirements: the power System: It consists of one rechargeable battery package of 12V and 1800mAh, and two power regulation sub-systems: the robot electronics and the motion system. The motion System: Differential drive, two drive motors with 50oz/in (Pololu 6VDC), and two incremental encoders with 562 ticks by revolution. The perception System: four bumpers, six infrared proximity sensors (GP1UM28YK), four photocells, one IMU (MiniIMU9v3), one line sensor, and one camera (CMUcam5). The Communication and User Feedback System: PC-Robot communication using RS-232, Robot-Robot communication using Bluetooth (HC-05), one buzzer, and eight colored LEDs. And the CPU: one ATMEGA644-20PU running FreeRTOS 7.4.2 (Barry, 2016).

Figure 1c shows the UVBots2 robot mechanical structure which has the following requirements: First, an attractive housing with different colors to encapsulate a metallic chassis, the battery pack, motors and hardware electronics. Second, three upper buttons to power on/off, reset and change the execution mode of the mobile robot. Third, one support to handle a marker in the bottom part of the robot. And fourth, three connectors for the PC-Robot communication and recharging the battery pack.

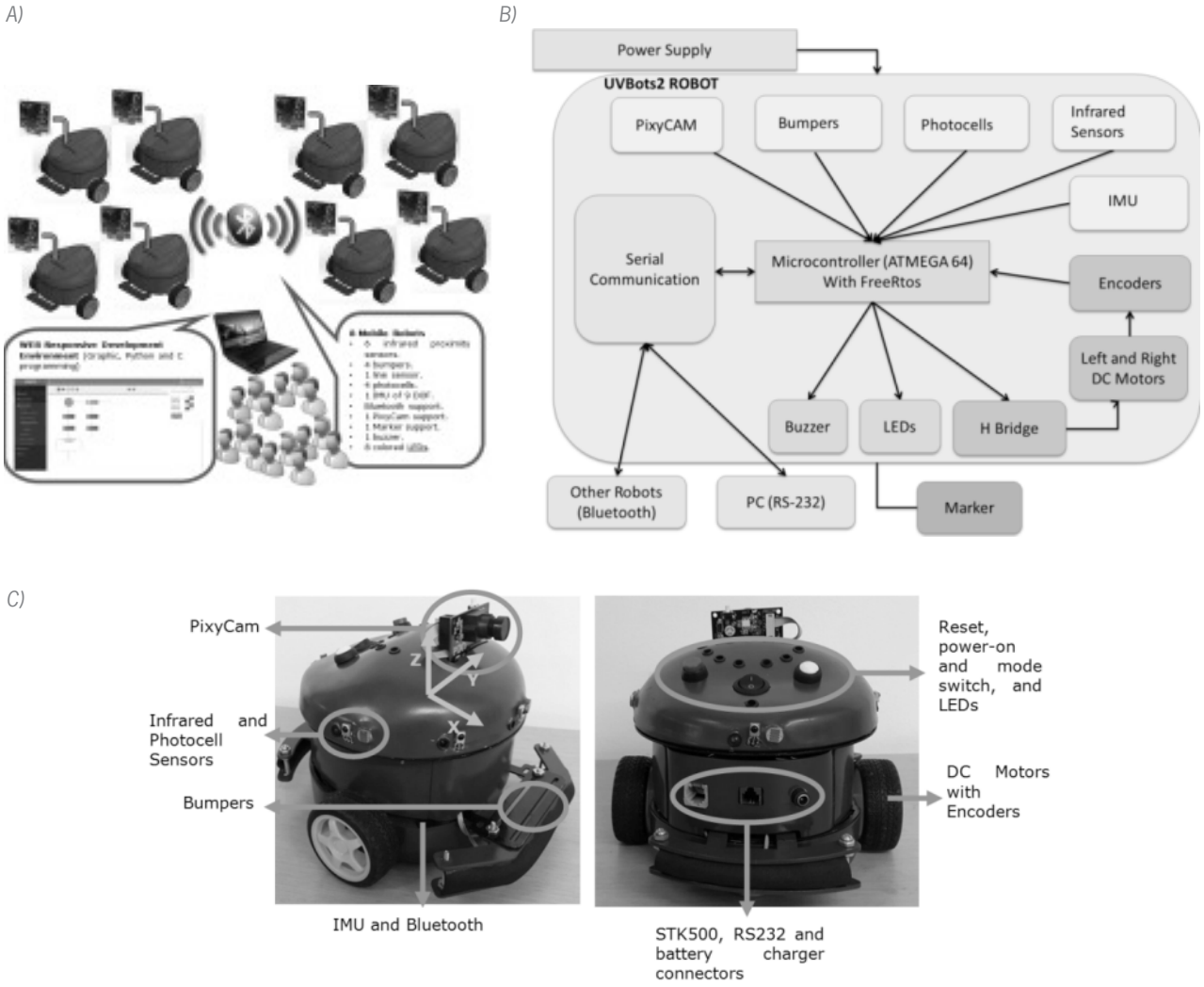


Figure 1. Hands-on platform conceptual diagram (a); hardware block diagram (b); UVBots2 mobile robot and placement of the perception system and coordinate frame (c) / Diagrama conceptual de la plataforma práctica (a); diagrama de bloques de hardware (b); robot móvil UVBots2 y la disposición del sistema de percepción y marco de coordenadas (c)

B. Motion system

The UVBots2 mobile robots have a differential kinematic model, which is shown in Ec. 1 considering the robot speed space.

$$\begin{bmatrix} V_R \\ W_R \end{bmatrix} = \begin{bmatrix} c/2 & c/2 \\ c/b & -c/b \end{bmatrix} \begin{bmatrix} w_r \\ w_l \end{bmatrix} \qquad \text{Ec.1}$$

Where, V_R and W_R are the lineal and rotational robot velocities, c is the wheel radius, b is the wheel distance, and w_r and w_l are the rotational wheel speeds. It is well known that the transfer function between the rotational speed of a DC motor and the voltage applied to it is lineal (Ogata, 2009). Then, assuming both DC motor models are equal, Ec. 2 shows the DC motor model for the lineal velocity of each robot wheel.

$$\frac{v_r}{l}(s) = \frac{K}{\tau s+1} u(s) \qquad \text{Ec. 2}$$

$$\begin{bmatrix} V_R \\ W_R \end{bmatrix} = \begin{bmatrix} c/2 & c/2 \\ c/b & -c/b \end{bmatrix} \begin{bmatrix} w_r \\ w_l \end{bmatrix} \qquad \text{Ec.1}$$

Donde, V_R y W_R son las velocidades lineales y rotacionales del robot, c es el radio de la rueda, b es la distancia de la rueda, y w_r y w_l las velocidades de rotación de las ruedas. Es bien sabido que la función de transferencia entre la velocidad de rotación de un motor CC y el voltaje aplicado a él, es lineal (Ogata, 2009). Entonces, suponiendo que ambos modelos de motor DC son iguales, la Ecuación 2 muestra el modelo de motor DC para la velocidad lineal de cada rueda del robot.

$$\frac{v_r}{l}(s) = \frac{K}{\tau s+1} u(s) \qquad \text{Ec. 2}$$

Donde, $v_{r/l}$ es la velocidad lineal de la rueda derecha o izquierda, K es la ganancia del modelo y τ es el tiempo constante del modelo. Usando los criterios estándar de 2%

para identificar estos parámetros (Ogata, 2009), los valores resultantes fueron $K = 5.81$ y $\tau = 221.2$ms, lo que significa un tiempo de estado estacionario de 884,8 ms. Entonces, con el fin de mejorar el tiempo de estado estacionario para evitar sobrepasar y reducir el error de estado estacionario a cero, se diseñó un controlador PI (Proporcional-Integral) para cada rueda del robot. Los requisitos de diseño de PI incluyen: un tiempo de estado estacionario de 750ms, sin sobrepasarse, y una frecuencia de sistema natural de 5.33rad / s. Considerando un diseño de controlador PI basado en el análisis de polo-cero, las constantes proporcionales e integrales resultantes son: *Kp = 1.3595, y Ti = 0.216*. Entonces, la función de transferencia del controlador PI se muestra en la Ecuación 3.

$$C(s) = \frac{1.3595*(1+0.2161s)}{0.2161s} \qquad \text{Ec. 3}$$

C. Sistema de percepción

Como se presenta en la Sección II.A, el sistema de percepción de UVBots2 está compuesto por los siguientes módulos: detección de proximidad, mediciones de intensidad de luz, medición inercial y visión.

La detección de proximidad se usa comúnmente para detectar obstáculos, aquí los UVBots2 usan sensores de contacto y sensores infrarrojos para detectarlos. Los cuatro parachoques se colocan como se muestra en la **Figura 1c**, tres de ellos en la parte frontal del robot móvil y otro en la parte posterior. La detección de proximidad infrarroja se basa en la medición del haz infrarrojo reflejado por los obstáculos. Este haz se modula a 40kHz, ya que el receptor es un módulo Sharp GP1UD-28YK. Un total de seis pares emisor/receptor se ponen alrededor del robot, como se muestra en la **Figura 1c**. Los datos brutos proporcionados por la detección de proximidad de UVBots2 son discretos, este mide solamente la presencia de la ausencia de obstáculos.

La detección de variables ambientales es importante para los robots móviles, los robots UVBots2 son capaces de medir la intensidad de la luz alrededor del robot. Esto se hace usando un conjunto de cuatro fotoceldas colocadas como se describe en la **Figura 1c**. La intensidad de luz alrededor del robot UVBot2 es proporcionada en valores porcentuales continuos. Hoy en día, los datos inerciales se han vuelto populares en robots móviles y dispositivos móviles. UVBots2 tiene una IMU de 9DOF; esta IMU es el MiniIMU9v3 de Pololu que tiene tres sensores diferentes: acelerómetro de tres ejes, magnetómetro y giroscopio. La **Figura 1c** muestra el cuadro de coordenadas asumido para este sensor.

Como se puede ver en la **Figura 1c**, los robots UVBots2 tienen un sensor de visión. Dado que una CPU ATMEGA64 no es capaz de procesar imágenes, se usó la PixyCam CMU5. Esta tiene la gran ventaja de realizar toda la tarea de procesamiento de imágenes a bordo y emitir el resultado de procesamiento a través de varios canales de comunicación, como RS232, I2C y SPI. En este trabajo se usa el canal SPI. Básicamente, el sensor de visión se utiliza para detectar hasta siete firmas de color dentro del Campo de Visión [FOV] de la cámara.

Where, $v_{r/l}$ is the lineal velocity of the right or left wheel, K is the model gain, and τ is the model constant time. Using the standard 2% criteria to identify these parameters (Ogata, 2009), the resulting values were $K = 5.81$ and $\tau = 221.2$ms, which means a steady-state time of 884.8ms. Then, in order to improve the steady-state time, to avoid overshooting, and reducing the steady-state error to zero, a PI (Proportional – Integral) controller was designed for each robot wheel. The PI design requirements include: a steady-state time of 750ms, no overshooting, and a natural system frequency of 5.33rad/s. Considering a PI controller design based on zero-pole analysis, the resulting proportional and integral constants are: *Kp = 1.3595, and Ti = 0.216*. Then, the PI controller transfer function is shown in Ec. 3.

$$C(s) = \frac{1.3595*(1+0.2161s)}{0.2161s} \qquad \text{Ec. 3}$$

C. Perception system

As introduced in Section II.A, the UVBots2 perception system is composed by the following modules: proximity sensing, light intensity measurements, inertial measurement, and vision. The proximity sensing is commonly used to detect obstacles; here the UVBots2 uses contact sensors and infrared sensors to detect them. The four bumpers are placed as shown in **Figure 1c**, three of them in the front of the mobile robot, and another in the back. The infrared proximity sensing is based on measuring the infrared beam reflected from obstacles. This beam is modulated at 40khz, since the receiver is a GP1UD28YK Sharp module. A total of six emitter/receiver pairs are placed around the robot as shown in **Figure 1c.** The raw data provided by the proximity sensing of UVBots2 is discrete, it measures the presence of absence of obstacles only. Sensing environmental variables is important for mobile robots, UVBots2 robots are able to measure light intensity around the robot. This is done using a set of four photocells placed as described in **Figure 1c**. The light intensity around the UVBot2 robot is provided in continuous percentage values. Nowadays, inertial data have been becoming popular in mobile robots and mobile devices. UVBots2 has an IMU of 9DOF; this IMU is the MiniIMU9v3 of Pololu which has three different sensors: 3-axis accelerometer, magnetometer, and gyroscope. The **Figure 1c** shows the coordinate frame assumed for this sensor.

As can be seen in **Figure 1c**, the UVBots2 robots have a vision sensor. Since an ATMEGA64 CPU is not able to process images, the Pixy Cam CMU5 was used. It has the great advantage of performing the entire image processing

task on-board, and issue the processing result over various communication channels such as RS232, I2C and SPI. In this work the SPI channel is used. Basically, the vision sensor is used to detect up 7 color signatures within the Field of View [FOV] of the camera.

D. Learner feedback and communication system

Human – robot interaction is an important aspect in order to know what the robot is doing at any moment. The UVBots2 robots have two different types of feedback for learners: first, basic audio systems implemented using a buzzer, and second a set of eight colored LEDs. The buzzer is placed inside the mobile robot, but the set of LEDs are placed at top of the mobile robot as shown **Figure 1**c.

Today, wireless communication between mobile platforms is very popular. The UVBots2 robots are able to communicate each other using Bluetooth. To do so, the HC-05 module is used, which can be configured as master or slave. The Bluetooth communication is used to exchange data between robots. However, to perform the PC – Robot communication a RS232 link is needed. The RS232 is used to download the program from the PC, where a user interface is employed to program the UVBots2 at three different levels of complexity; this is introduced in Section III.

III. Programming environment

The programming environment was designed using WEB responsive technology. In this way, the user application is available worldwide. The development environment supports three different programming interfaces, each one for a different level of complexity namely: basic, intermediate and advanced. The basic level uses a graphical interface; it is designed for learners with basic knowledge on programming and robotics. The intermediate level uses Python programming language due it is easy to use and understand; but it assumes intermediate knowledge on programming and robotics. The advanced level uses ANSI C programming language; here it is assumed learners have advanced knowledge on programming and robotics. This section describes the design requirements, the implementation details using DJango framework (DJango-Software-Foundation, 2016), and the modules for graphical, Python and C programming.

A. Design requirements

The software development process for this project was carried out using eXtreme Programming methodology (Shore & Warden, 2008). The software functional requirements for the developed modules are described as follows:

D. Sistema de retroalimentación y comunicación del estudiante

La interacción hombre-robot es un aspecto importante para saber lo que el robot está haciendo en cualquier momento. Los robots UVBots2 tienen dos tipos diferentes de retroalimentación para los estudiantes: primero, sistemas de audio básicos implementados usando un zumbador, y segundo un conjunto de ocho LED de color. El zumbador se coloca dentro del robot móvil y el conjunto de LED en la parte superior del robot móvil, como se muestra en la **Figura 1**c.

Hoy en día, la comunicación inalámbrica entre plataformas móviles es muy popular. Los robots UVBots2 son capaces de comunicarse entre sí mediante Bluetooth. Para ello, se utiliza el módulo HC-05, el cual puede configurarse como maestro o esclavo. La comunicación Bluetooth se utiliza para intercambiar datos entre robots. Sin embargo, para realizar la comunicación PC – Robot es necesario un enlace RS232, que se utiliza para descargar el programa desde la PC, donde se emplea una interfaz de usuario para programar los UVBots2 en tres niveles diferentes de complejidad; esto se introduce en la Sección III.

III. Entorno de programación

El entorno de programación se diseñó utilizando tecnología adaptativa WEB. De esta manera, la aplicación de usuario está disponible en todo el mundo. El entorno de desarrollo permite tres interfaces de programación diferentes, cada una para un nivel de complejidad diferente, a saber: básico, intermedio y avanzado. El nivel básico utiliza una interfaz gráfica y está diseñado para estudiantes con conocimientos básicos de programación y robótica; el nivel intermedio utiliza lenguaje de programación Python, debido a que es fácil de usar y entender, pero asume un conocimiento intermedio en programación y robótica; el nivel avanzado utiliza el lenguaje de programación ANSI C, en este nivel se supone que los estudiantes tienen conocimientos avanzados sobre programación y robótica. En esta sección se describen los requisitos de diseño, los detalles de implementación usando el framework DJango (DJango-Software-Foundation, 2016) y los módulos para la programación gráfica Python y C.

A. Requerimientos de diseño

El proceso de desarrollo de software para este proyecto se llevó a cabo utilizando la metodología de programación eXtreme (Shore & Warden, 2008). Los requisitos funcionales de software para los módulos desarrollados se describen a continuación.

- Módulo de usuario, útil para: realizar el registro; manejar las sesiones; acceder a la programación, desafíos y muestras; y guardar, abrir y editar programas.
- Módulo de programación gráfica, útil para: mostrar una interfaz gráfica de programación; permitir el cambio de la interfaz de programación a Python y C; mostrar una lista completa de bloques de programación organizados por categoría; permitir a los estudiantes realizar operaciones de arrastrar y soltar; y permitir a los estudiantes la configuración y borrado de bloques de programación individuales.

- Módulo de programación de Python, útil para: mostrar una interfaz de programación basada en Python; permitir el cambio de la interfaz de programación a C; y manejar los programas de Python.
- Módulo de programación C, útil para: mostrar una interfaz de programación basada en C y manejar programas C.
- Modelo de datos, útil para: definir una estructura general para las funcionalidades de UVBots2; definir una estructura para los bloques de programación gráfica y las correspondientes funcionalidades de UVBots2; y definir el modelo de persistencia para la aplicación de software y los datos de los usuarios.
- Conversión, compilación y descarga de programas, útil para: traducir programas gráficos a programas Python; traducir programas gráficos a programas C; traducir programas Python a programas C; validar sintácticamente los programas gráficos de aprendizaje en Python y C; y permitir la compilación de programas utilizando C.
- Módulo de desafíos y muestras, útil para mostrar una lista de desafíos disponibles, incluyendo el problema a resolver y los resultados esperados; permitir a los estudiantes la implementación de los desafíos seleccionados; mostrar a los estudiantes una lista de códigos de muestra; y permitir el manejo del código de muestra utilizando interfaces gráficas Python o C.

Dado que la aplicación es WEB adaptativa, hay equivalentes de servidor/cliente que deben ser considerados. Se asumieron los siguientes requisitos no funcionales para los lados servidor/cliente: en el lado servidor, todos los lenguajes de programación y herramientas de desarrollo son de código abierto; el lenguaje de programación natural de los UVBots2 es ANSI C; se instala y configura un servidor WEB con compatibilidad con DJango y se permiten hasta treinta usuarios simultáneos; JSON se usa como intercambio de datos estándar; en el lado cliente, los usuarios pueden conectarse al servidor usando Google Chrome o Mozilla, la resolución de pantalla mínima es de 640x480 píxeles.

B. Arquitectura

Este trabajo fue implementado usando DJango en el lado del servidor. La **Figura 2** muestra la arquitectura del software. Aquí, los módulos de *Editor, Muestras de aplicaciones y Desafíos* implementan todos los requisitos funcionales descritos en la Sección III.A. El módulo Editor implementa el entorno de programación gráfico Python y C. Estos módulos se detallan en las siguientes secciones.

Módulo de programación gráfica

La **Figura 3a** muestra la interfaz gráfica de programación. Los estudiantes pueden comenzar a programar usando las operaciones de arrastrar y soltar de los bloques de programación disponibles en el lado izquierdo. Todos los bloques de programación se añaden hacia abajo a la rutina

- User module: to perform registration; to handle sessions; to access to programming, challenges, and samples; to save, open and edit programs.
- Graphical programming module: to show a graphical programming interface; to allow changing the programming interface to Python and C; to show a complete list of programming blocks organized by category; to allow learners to perform drag-and-drop operations; to allow learners configure and delete individual programming blocks.
- Python programming module: to show a Python based programming interface; to allow changing the programming interface to C; to handle Python programs.
- C Programming module: to show a C based programming interface; to handle C programs.
- Data model: to define a general structure for the UVBots2 functionalities; to define a structure for the graphical programming blocks and the corresponding UVBots2 functionalities; to define the persistence model for the software application and users data.
- Program conversion, compilation, and download: to perform translation from graphical to Python programs; to perform translation from graphical to C programs; to perform translation from Python to C programs; to validate graphical, Python and C learner programs syntactically; to allow programs compilation using C.
- Challenges and samples module: to show a list of available challenges including the problem to solve, and the results expected; to allow learners implement the challenges selected; to show to learners a list of sample codes; to allow handling the sample code using graphical, Python or C interfaces.

Since the application is WEB responsive, there are server/client counterparts which must be considered. The following non-functional requirements were assumed for the server/client sides: server, all programming languages and development tools are open-source; the natural programming language of the UVBots2 is ANSI C; to install and configure a WEB server with DJango compatibility, and supporting up 30 simultaneous users; JSON is used as data standard interchange. The Client, users can connect to server using Google Chrome, or Mozilla WEB browsers; minimum screen resolution of 640x480 pixels.

B. Architecture

This work was implemented using DJango at the server side. **Figure 2** shows the software architecture. Here, the

"Editor", "App. Samples" and "Challenges" modules implement all the functional requirements described in Section II-I.A. The "Editor" module implements the graphical, Python and C programming environment. These modules are detailed in the following sections.

Module for graphical programming

Figure 3a shows the graphical programming interface. Learners can start programming using drag-and-drop operations of programming blocks available at the left side. All programming blocks are added downwards into the main routine; however, robot sensor blocks are added horizontally, and they create an independent execution threads. All learprincipal, sin embargo, los bloques de sensor del robot se agregan horizontalmente y crean hilos independientes de la ejecución. Todos los programas gráficos del estudiante se almacenan automáticamente en archivos JSON (ver **Figura 3b**), estos archivos tienen una estructura básica compuesta de dos partes: primero, se almacenan los datos del usuario, el nombre del programa y otros datos generales; y segundo, se almacena el programa en sí mismo en dos sub partes, los bloques de rutina principales y los bloques de rutina de sensores. En estas sub partes, los bloques de programación se agrupan utilizando la secuencia del programa. Se añaden estructuras independientes para cada bloque que pertenezca al programa del usuario. Estas estructuras contienen información como atributos de bloque, datos de forma gráfica y el código C correspondiente

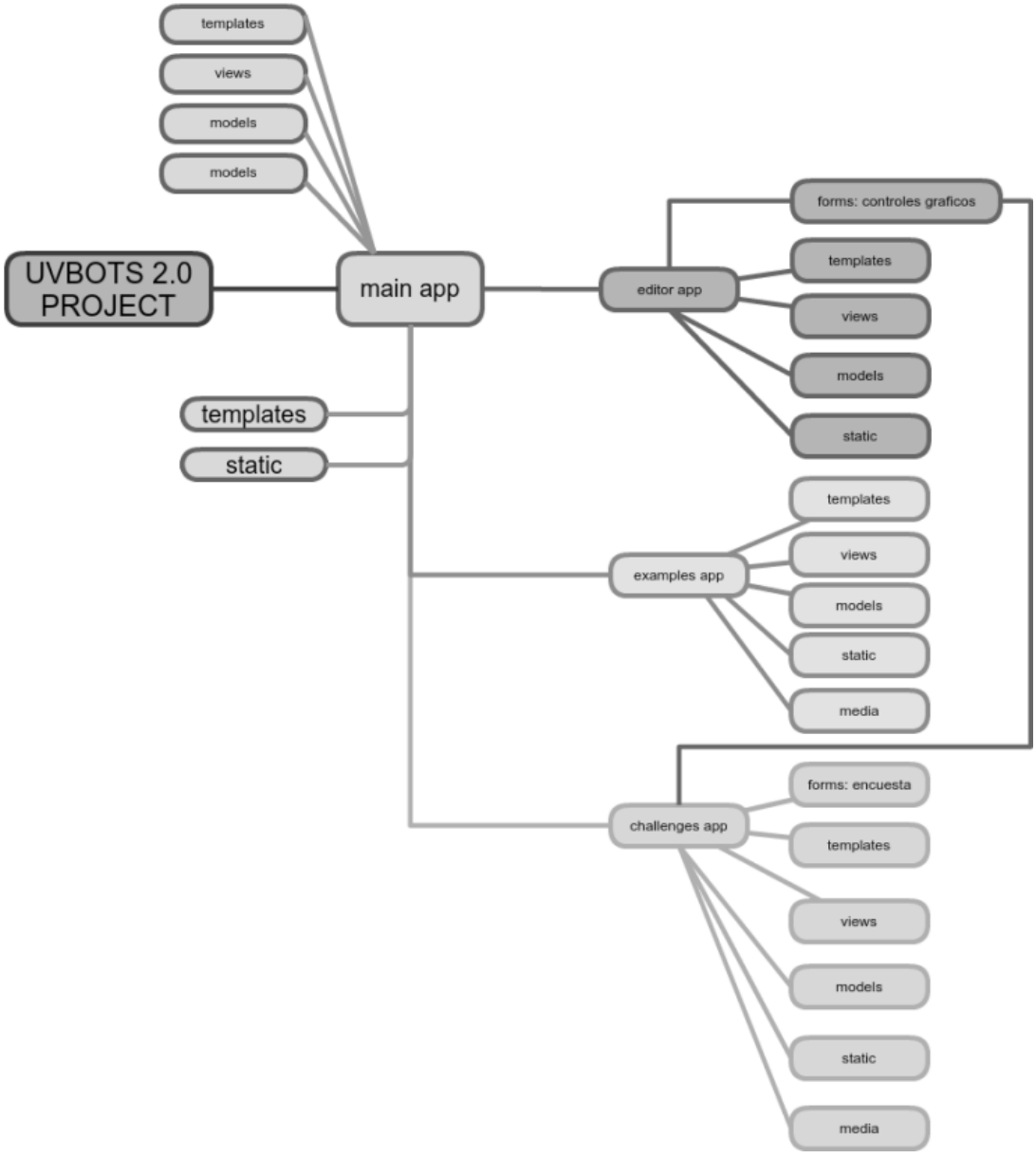
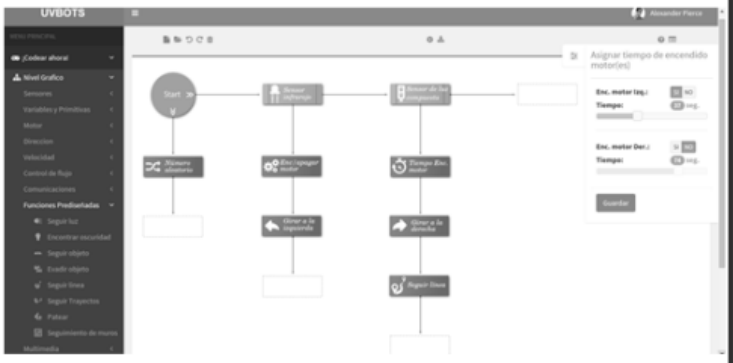


Figure 2. Components view of the UVBots2 software architecture / vista de los componentes de la arquitectura del software UVBots2

A)



B)

C)

D)

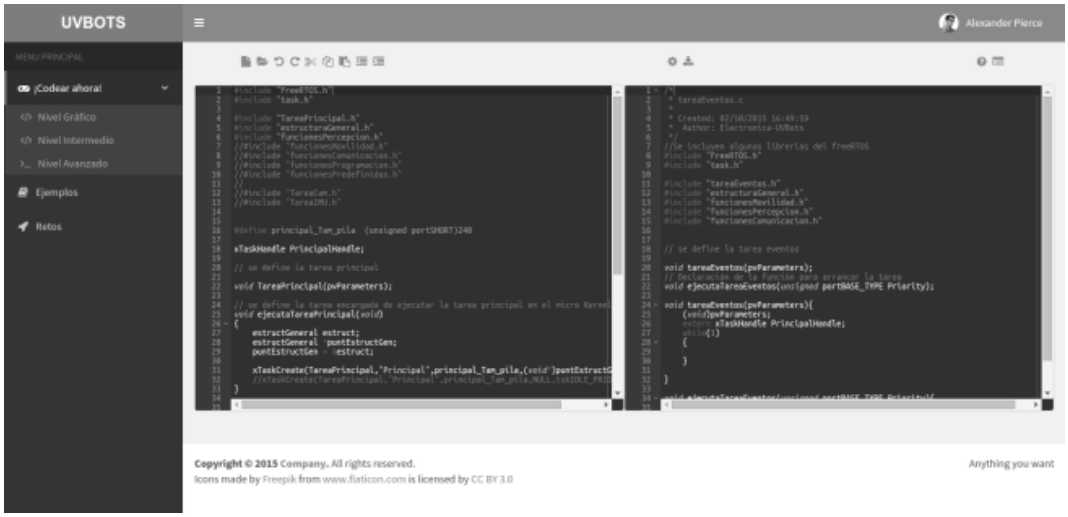


Figure 3. Graphical programming interface (a); graphical program persistence (b); Phyton programming interface (c); y ANSI C programming interface (d) / Interfaz de programación gráfica (a); persistencia del programa gráfico (b); interfaz de programación de Python (c); e interfaz de programación ANSI C (d)

ner graphical programs are automatically stored in JSON files; this can be observed in **Figure 3b**. These files have a basic structure composed of two parts: first, user data, program name and other general data are stored; and second, it stores the program itself into two subparts namely: the main routine blocks, and the sensors routine blocks. Into these subparts, programming blocks are grouped using the program sequence. Independent structures are added for each block that belongs to the user program. These structures hold information such as block attributes, graphical form data, and the corresponding C code for each block. These structures were implemented using Python dictionaries. In this work, three different dictionaries were created: first, to translate block programming to C; second, block programming to Python; and third, Python programming to C.

Module for python programming

In this work, Python was the programming language selected for the intermediate level. The main reason of this selection was language simplicity. Python is a simple and easy to understand programming language; besides, there is much syntactic equivalence with C. **Figure 3c** shows the Python programming interface, where learners can start programming using all built-in functions available for the UVBots2 robots. Here, learners must consider two main routines: "tareaPrincipal" (mainTask), and "*tareaEventosSensores*" (taskSensorEvents). The former is used to introduce all Python sentences corresponding to the main execution thread. The latter is used to introduce all Python sentences of the UVBots2 robot sensors.

Module for C Programming

Figure 3d shows the C programming interface for advanced learners. This interface shows two C programs templates: the "*tareaPrincipal*" or mainTask, and the "*tareaEventos*" or eventTask. They are used to store C code related with the main program, and C code related with the robot sensors. It can be observed from **Figure 3d** that C code is already wrote on these files, this code is a C program template taking into account the FreeRTOS Real-Time operative system (Barry, 2016). The FreeRTOS was used to program all UVBots2 robot functionalities which were described in Section II. Using this interface, C programs are directly compiled to get the binary file supported by the UVBots2 robot's microcontroller.

IV. Test and results

In order to test the proposed system, three experiments were performed. Test No. 1 shows the results obtained when the UVBots2 robot was programmed to draw different geo-

para cada bloque. Estas estructuras se implementaron utilizando diccionarios de Python. En este trabajo, se crearon tres diccionarios diferentes: el primero, para traducir la programación de bloques a C; el segundo, la programación de bloques a Python; y el tercero, la programación de Python a C.

Módulo de programación en Python

Como se indicó, Python fue el lenguaje de programación seleccionado para el nivel intermedio. La razón principal de esta selección fue la simplicidad del lenguaje. Python es un lenguaje de programación simple y fácil de entender, y tiene además mucha equivalencia sintáctica con C. La **Figura 3c** muestra la interfaz de programación de Python, donde los estudiantes pueden comenzar a programar usando todas las funciones incorporadas disponibles para los robots UVBots2. Aquí, los estudiantes deben considerar dos rutinas principales: *tareaPrincipal* y *tareaEventosSensores*. La primera se utiliza para introducir todas las oraciones de Python correspondientes al hilo de ejecución principal, la segunda, para introducir todas las oraciones de Python de los sensores del robot UVBots2.

Módulo de programación en C

La **Figura 3d** muestra la interfaz de programación C para estudiantes avanzados. Esta interfaz presenta dos plantillas de programas C, la *tareaPrincipal* y la *tareaEventos*, que se utilizan para almacenar el código C relacionado con el programa principal y el código C relacionado con los sensores del robot. Se puede observar en la **figura 3d** que el código C ya está escrito en estos archivos, este código es una plantilla de programa C que tiene en cuenta el sistema operativo en tiempo real FreeRTOS (Barry, 2016). El FreeRTOS se utilizó para programar todas las funcionalidades del robot UVBots2 que se describieron en la Sección II. Usando esta interfaz, los programas C se compilan directamente para obtener el archivo binario compatible con el micro controlador del robot UVBots2.

IV. Prueba y resultados

Para probar el sistema propuesto se realizaron tres experimentos: en la prueba Nº 1 se muestran los resultados obtenidos cuando el robot UVBots2 fue programado para dibujar figuras geométricas diferentes, los programas para estas figuras se implementaron utilizando la interfaz gráfica de programación para los estudiantes básicos; en la prueba Nº 2 se muestran los resultados obtenidos cuando el robot UVBots2 intenta seguir un objeto colocado delante de él usando la PixyCam, esta aplicación fue programada usando lenguaje Python para estudiantes intermedios; finalmente, en la prueba Nº 3 se describen los resultados obtenidos cuando el robot móvil utiliza la programación basada en el comportamiento (Arkin, 1998) para alcanzar la zona más iluminada del entorno, evitando obstáculos, esta prueba se implementó utilizando lenguaje C para estudiantes avanzados. El objetivo de estas pruebas fue mostrar las principales funcionalidades de UVBots2 a diferentes niveles de conocimiento del estudiante.

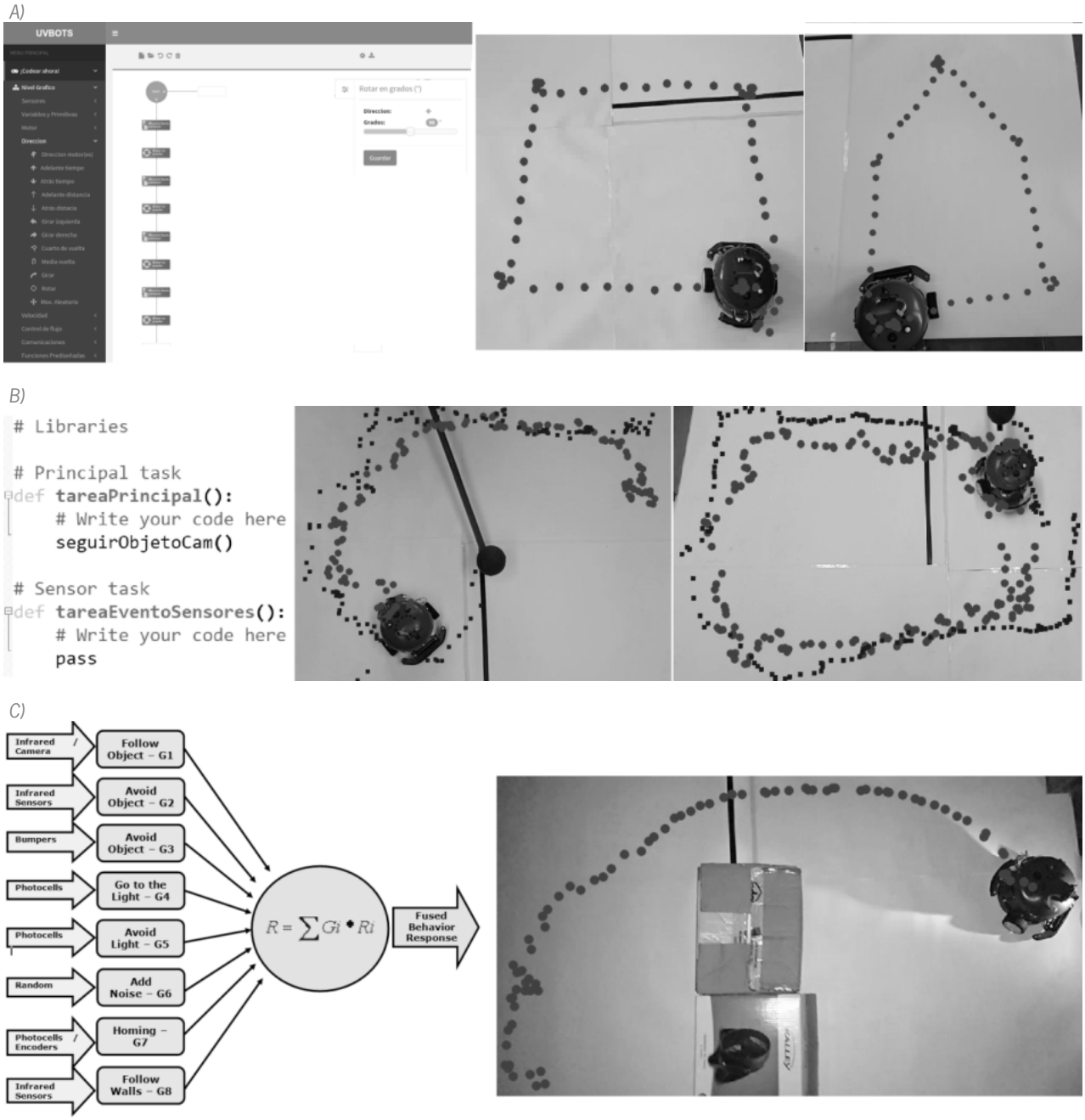


Figure 4. Geometric figures [square and house] drew using basic programming interface (a); using a Python program, the mobile robot is able to follow a blue object using the PixyCam (b); homing and obstacle avoidance behaviors programmed using the advanced programming interface (c) / Figuras geométricas [cuadrado y casa] dibujadas utilizando la interfaz de programación básica (a); utilizando un programa Python, el robot móvil puede seguir un objeto azul usando la PixyCam (b); comportamientos de guiado y evitación de obstáculos programados usando la interfaz de programación avanzada (c)

Prueba N ° 1. Figuras geométricas

El objetivo de esta prueba es programar el UVBot2 utilizando el entorno gráfico para realizar pasos secuenciales básicos con el fin de dibujar figuras geométricas. En la **Figura 4a** se muestra el programa utilizado para construir un cuadrado. El cuadrado resultante se muestra en el centro de la **Figura 4a**, y un resultado adicional que construye una casa se muestra a la derecha de la **Figura 4a**. Debido a la resolución de las imágenes no se puede observar el rastro del marcador, por este motivo se desarrolló un programa

metric figures; the programs for these figures were implemented using the graphical programming interface for basic learners. In test No. 2 shows the results obtained when the UVBots2 robot tries to follow an object placed in front of it using the PixyCam; this application was programmed using Python language for intermediate learners. Finally, test No. 3 describes the results obtained when the mobile robot uses the behavior based programming (Arkin, 1998) in order to reach the most illuminated zone of the environment, while avoi-

ding obstacles; this test was implemented using C language for advanced learners. The aim of these tests is to show the main UVBots2 functionalities at different levels of learner's knowledge.

Test No. 1. Geometric figures

The aim of this test is programming the UVBot2 using the graphical environment to perform basic sequential steps in order to draw geometric figures. **Figure 4a** shows the program used to build a square, the resulting square is shown in the middle of **Figure 4a**, and one additional result building a house is shown at right of **Figure 4a**. Due the images resolution the marker trace cannot be observed, for this reason a Matlab program was developed to process the video captured while the test was performed. The circles in **Figure 4a** show the UVBot2 robot trajectory.

Test No. 2. Camera based object following

In this test, the Python programming environment was used to perform an object following application using the PixyCam. Here, the program in charge of perform this task has a more complicated syntax in comparison with the graphical programming environment. **Figure 4b** shows the Python program and two different trajectories performed while the UVBot2 robot was following the ball. The circles show the robot trajectory, and the squares show the ball trajectory. Basically, the UVBot2 robot firmware captures the ball coordinates with respect to the image principal point, this is used by the seguirObjCam() procedure to drive the mobile robot.

Test No. 3. Light based homing with obstacle avoidance

In order to test the advanced programming interface which handles ANSI C programs, a behavior based application was implemented. The robot homing behavior is very important in robotics, it allows to mobile robots returning to recharging stations, and going to specific goals in the environment. Here, the robot goal is the more illuminated region in **Figure 4c**, which is placed at the right. However, avoiding obstacles is important while going to goal. Then, two simultaneous tasks must to be coordinated in order to perform this task. Behavior based programming (Arkin, 1998) is a control architecture used to handle this problem. At left side of **Figure 4c** is shown the set of behaviors supported by the UVBots2 robots. All the behaviors motion commands are coordinated using a weighted sum, each weight represents the behavior importance into the behavior stack. In this test, the Avoid Object (using the infrared sensors), and the Go to the Light (using the photocells) behaviors were emplo-

ma Matlab para procesar el video capturado mientras se realizaba la prueba. Los círculos de la **Figura 4a** muestran la trayectoria del robot UVBot2.

Prueba No. 2. Seguimiento del objeto basado en cámara

En esta prueba, el entorno de programación de Python se utilizó para desarrollar una aplicación de seguimiento del objeto utilizando la PixyCam. Aquí, el programa encargado de realizar esta tarea tiene una sintaxis más complicada en comparación con el entorno de programación gráfica. En la **Figura 4b** se muestra el programa Python y dos trayectorias diferentes realizadas mientras el robot UVBot2 estaba siguiendo la bola, los círculos muestran la trayectoria del robot y los cuadrados la trayectoria de la bola. Básicamente, el firmware del robot UVBot2 captura las coordenadas de la bola con respecto al punto principal de la imagen, esto es usado por el procedimiento seguirObjCam () para conducir el robot móvil.

Prueba N ° 3. Guiado basado en luces con evitación de obstáculos

Para probar la interfaz de programación avanzada que maneja los programas ANSI C, se implementó una aplicación basada en el comportamiento. El comportamiento de guiado del robot es muy importante en la robótica, pues permite a los robots móviles volver a las estaciones de recarga e ir a objetivos específicos en el entorno. Aquí, el objetivo del robot es la región más iluminada de la **Figura 4c**, que se coloca a la derecha. Sin embargo, es importante evitar los obstáculos mientras se va al objetivo, por lo que se deben coordinar dos tareas simultáneas para realizar esta tarea. La programación basada en el comportamiento (Arkin, 1998) es una arquitectura de control utilizada para manejar este problema. En el lado izquierdo de la **Figura 4c** se muestra el conjunto de comportamientos permitidos por los robots UVBots2. Todos los comandos de comportamientos de movimiento se coordinan usando una suma ponderada, cada peso representa la importancia del comportamiento en la pila de comportamiento. En esta prueba, se emplearon los comportamientos de Avoid Object (utilizando los sensores de infrarrojos) y Go to the Light (utilizando las fotoceldas) con pesos de 0.7 y 0.3 respectivamente. La configuración de la pila de comportamiento se realiza utilizando el lenguaje C en la interfaz de programación avanzada. En la **Figura 4c** también se muestra la trayectoria resultante del UVBot2. Inicialmente el robot móvil está ligado a la luz, sin embargo, hay cajas en el camino que debe evitar para alcanzar la región más iluminada del entorno.

V. Conclusiones

En este trabajo se presentó la plataforma UVBots2 que incluye un entorno de programación y un conjunto de ocho robots móviles con tres niveles de conocimiento. El entorno de desarrollo permite el uso de los lenguajes de programación gráfica, Python y ANSI C, cada uno e en un nivel de conocimiento distinto, básico, intermedio o avan-

zado, respectivamente, sobre conceptos de programación o robótica. Los robots móviles están equipados con seis sensores de proximidad, cuatro parachoques, cuatro fotoceldas, una unidad diferencial con codificadores, un 9DOF IMU, comunicación Bluetooth, un zumbador, ocho LEDs de color y una cámara PixyCam. La plataforma UVBots2 ofrece un importante objeto de experimentación utilizado principalmente para ejecutar todos los programas de los estudiantes, que permite aprender conceptos básicos de programación, robótica o STEM.

El entorno de desarrollo se implementó teniendo en cuenta las siguientes propiedades: utiliza tecnologías WEB adaptativas, lo que aporta a los usuarios independencia de la plataforma; es capaz de traducir programas basados en bloques a Python o código ANSI C y de los programas Python a ANSI C usando diccionarios; los estudiantes pueden guardar, editar, descargar y ejecutar programas utilizando el robot móvil; finalmente, este entorno de desarrollo considera los conocimientos previos de los estudiantes sobre la programación y la robótica, ya que ofrece una metodología de aprendizaje de complejidad creciente. El firmware de los robots móviles se implementó utilizando el kernel en tiempo real FreeRTOS para ejecutar simultáneamente todas las tareas de percepción, comunicación, movimiento, retroalimentación y usuario.

Hoy en día existen esfuerzos gubernamentales europeos (CEDEFOP, 2010) y norteamericanos ("Informática para...", 2016) para mejorar y desarrollar nuevas estrategias educativas en los diferentes niveles del proceso educativo. Las experiencias previas del semillero de investigación en Robótica (Jimenez et al., 2010) y la plataforma práctica presentada en este trabajo representan una interesante opción para motivar a los niños y jóvenes hacia el estudio de la ciencia, la tecnología, la ingeniería y las matemáticas. S&T

yed using weights of 0.7 and 0.3 respectively. The behavior stack configuration is done using C language at the advanced programming interface. **Figure 4c** also shows the resulting trajectory of the UVBot2. Initially, the mobile robot bound to the light, however, the boxes were in the way, and then it avoids them to reach the most illuminated region of the environment.

V. Conclusions

In this work, the UVBots2 platform was presented which includes a programming environment, and a set of eight mobile robots having three levels of knowledge. The development environment supports graphical, Python and ANSI C programming languages, each one corresponding for basic, intermediate and advanced levels of knowledge on programming concepts or robotics. The mobile robots are equipped with six proximity sensors, four bumpers, four photocells, differential drive with encoders, one 9DOF IMU, Bluetooth communication, one buzzer, eight colored LEDs and one PixyCam camera. The UVBots2 platform offer an important experimentation object mainly used to execute all learner programs; however basic programming, robotics or STEM concepts can be learnt.

The development environment was implemented considering the following properties: it uses WEB responsive technologies, which brings to users platform independence; it is able to translate block based programs to Python, or ANSI C code, and from Python programs to ANSI C using dictionaries; learners are able to save, edit, download and execute programs using the mobile robot; finally, this development environment considers the learners previous knowledge on programming and robotics, since it offers an increasing complexity learning methodology. The mobile robots firmware was implemented using the FreeRTOS real-time kernel in order to concurrently execute all the perception, communication, motion, feedback, and user tasks.

Nowadays, there are European (CEDEFOP, 2010) and North American ("Computer Science for…", 2016) governmental efforts to improve, and develop novel educational strategies at different levels of the education process. Considering the previous experiences on the Robotics seedbed (Jimenez et al., 2010), and the hands-on platform presented in this work, they represent an interesting option to bring children and young people to science, technology, engineering and math. S&T

References / Referencias

Arkin, R.C. (1998). *Behavior-based robotics.* Cambridge, MA: MIT Press.

Aseba (2013). *Thymio II Robot* [online]. Retrieved from: https://aseba.wikidot.com/en:thymio

Awabot (2016). *Awabot education* [online]. Retrieved from: http://www.pob-tech.com/

Awabot. (2013). *Pob-Bot robot* [online]. Retrieved from: http://education.awabot.com/

Barry, R. (2016). *FreeRTOS - Market leading RTOS* [online]. Retrieved from: http://www.freertos.org/

Benitti, F.B.V. (2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education, 58*(3), 978-988.

BirdBrain-Technologies. (2016). *The finch robot* [online]. Retrieved from: http://www.finchrobot.com/

Code.org. (2016). *Anybody can learn* [online]. Retrieved from: https://code.org/

De Cristoforis, P., Pedre, S., Nitsche, M., Fischer, T., Pessacg, F., & Di Pietro, C. (2013). A behavior-based approach for educational robotics activities. *IEEE transactions on education, 56*(1), 61-66.

DJango-Software-Foundation. (2016). *The Web framework for perfectionists with deadlines - DJango* [online]. Retrieved from: https://www.djangoproject.com/

European Centre for the Development of Vocational Training [CEDEFOP]. (2010). Skills for green jobs: European synthesis report).Luxemburg: Publications Office or the European Union.

Giraldo, C., Florian, B., Bacca, B., Gómez, F., & Muñoz, F. (2012). A programming environment having three levels of complexity for mobile robotics. *Ingeniería e Investigación, 32*(3), 76-82.

Gómez, F., Muñoz, F., Florián, B.E., Giraldo, C.A., & Bacca-Cortés, E.B. (2008). Design and testing of a mobile robot with three levels of complexity for robotics experimentation. *Ingeniería y Competitividad, 74*(2), 53-74.

Jimenez, M, Caicedo, E., & Bacca-Cortes, E. (2010). Tool for experimenting with concepts of mobile robotics as applied to children's education. *IEEE Transactions on Education, 53*(1), 88-95.

K-Team (2016). *Mobile robotics* [online]. Retrieved from: http://www.k-team.com/

LEGO. (2016). *Robolab on-line WEB site* [online]. Retrieved from: http://www.robolabonline.com/home

Major, L., Kyriacou, T., & Brereton, O. P. (2012). Systematic literature review: teaching novices programming using robots. *IET software, 6*(6), 502-513.

Ministerio de Educación (2014). *Plan estratégico de ingeniería 2012-2016*. Buenos Aires, Argentina: Ministerio de Educación. Available at: http://pefi.siu.edu.ar/

Ministerio de Educación Nacional [MEN]. (2014). *SPADIES.* Bogotá, Colombia: MEN. Available at: http://www.mineducacion.gov.co/1621/w3-article-156292.html

MIT, MIT-Media-Lab & MIT-CSAIL. (2016). *MIT App Inventor* [online]. Retrieved from: http://appinventor.mit.edu/explore/about-us.html

MIT-Media-Lab, 2016. *Scratch: Imagine, programming, share*. Retrieved from: https://scratch.mit.edu

Ogata, K. (2009). *Sistemas de control en tiempo discreto.* México: Prentice Hall.

Parallax. (2014). *Scribbler robot* [online]. Retrieved from: http://www.parallax.com/product/28136

Shore, J. & Warden, S. (2008). *The art of agile development*. Sebastopol, CA: O'Reilly.

Smith, M. (2016, January 30). *Computer Science for All*. Retrieved from: https://www.whitehouse.gov/blog/2016/01/30/computer-science-all

CURRICULUM VITAE

Sergio García Electronics Engineer from Universidad del Valle (Cali, Colombia) / Estudiante de la Facultad de Ingeniería de la Universidad del Valle (Cali, Colombia).

Sebastián Rueda Electronics Engineer from Universidad del Valle (Cali, Colombia) / Estudiante de la Facultad de Ingeniería de la Universidad del Valle (Cali, Colombia).

Beatriz Florián-Gaviria Systems and Computational Engineer (Universidad del Valle, Cali-Colombia), Master in Systems and Computation Engineer (Universidad de los Andes, Bogotá, Colombia), and PhD in Technologies of Information at the Universidad de Girona (Spain) / Ingeniera de Sistemas y Computación de la Universidad del Valle (Cali-Colombia), Máster en Ingeniería de Sistemas y Computación de la Universidad de los Andes (Bogotá, Colombia) y Doctora en Tecnologías de la Información de la Universidad de Girona (España).

Bladimir Bacca-Cortés Electronics Engineer with a Master degree in Automatics from the Universidad de Valle (Cali, Colombia) and Ph.D in Technology of the Universidad de Girona (Spain) / Ingeniero Electrónico con Maestría en Automática de la Universidad del Valle (Cali, Colombia) y Doctorado en Tecnología de la Universidad de Girona (España).