Diseño e implementación de un entorno de desarrollo integrado para robótica educativa soportado por una máquina virtual para plataformas de computación física

Ricardo Moran

Universidad Abierta Interamericana, Centro de Altos Estudios en Tecnología Informática, CABA, Argentina

Comisión de Investigaciones Científicas, La Plata, Buenos Aires, Argentina

ricardo.moran@uai.edu.ar

Resumen. *El uso de dispositivos físicos como herramientas de aprendizaje presenta diversos desafíos. Una de las dificultades más grandes a las que se enfrentan los alumnos es la construcción de un modelo mental que permita entender la ejecución de los programas y su relación con el código fuente. El uso de lenguajes visuales no alcanza a resolver este problema y, adicionalmente, puede complicar la eventual transición a lenguajes basados en texto. En este artículo se presenta un entorno de desarrollo integrado para robótica educativa que ofrece una experiencia de programación interactiva basada en un lenguaje híbrido bloques/código.*

1. Introducción

La robótica educativa es una disciplina que tiene como objetivo la utilización de dispositivos robóticos como material pedagógico, poniendo a la enseñanza de la tecnología no como un fin en sí mismo sino como un medio para entender y conocer mejor el mundo que nos rodea (Barrera Lombana, 2015). En este sentido, la robótica educativa presenta numerosos beneficios al proceso de aprendizaje, entre otros podemos mencionar: impacto positivo en la motivación de los estudiantes, reducción del ausentismo escolar (Vega-Moreno et al., 2016), fortalecimiento de las habilidades vinculadas al trabajo en equipo y compañerismo, desarrollo de habilidades que exceden lo meramente intelectual y ayudan en el pasaje del pensamiento concreto al pensamiento abstracto (Zabala et al., 2010).

Asimismo, la robótica educativa fomenta la formación de ciudadanos capaces de interactuar de forma productiva con la tecnología (Factorovich & Sawady O'Connor, 2017), ubicando a los estudiantes en una posición de creador y no de mero consumidor. Considerando además la demanda insatisfecha por profesionales en el área de sistemas (*El Gobierno presentó un plan para formar 111.000 programadores jóvenes y adultos - LA NACION*, s. f.) (*Argentina Programa*, 2020) y el potencial de la robótica educativa para ayudar en la formación de programadores, es comprensible el crecimiento significativo durante los últimos años que ha tenido la inserción de robots en las aulas, tanto en Argentina como en el resto del mundo (Lopes Guedes et al., 2015).

Dada la importancia del tema y los esfuerzos existentes por introducir la robótica en las aulas, resulta fundamental optimizar el uso de los recursos para obtener así el mayor beneficio posible que representa la robótica educativa para la sociedad. Para ello

es necesario identificar y resolver los problemas que se observan en las herramientas de programación para robótica educativa disponibles en la actualidad. En este artículo se presenta un proyecto de doctorado cuyo objetivo es aportar soluciones a dichos problemas.

2. Problema

La explosión en popularidad y repentina demanda de soluciones tecnológicas para implementar robótica en las aulas tuvo como consecuencia la proliferación de kits de robótica orientados a usuarios no expertos.

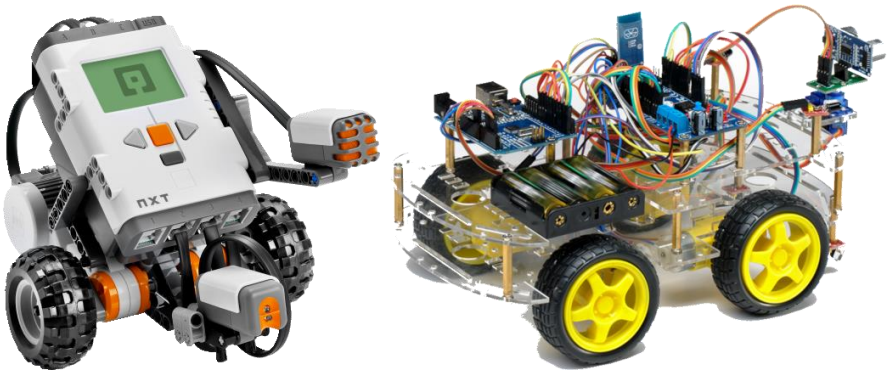

Figura 1. Kits de robótica: Lego Mindstorms NXT (izquierda) y Arduino (derecha)

En casi todos los casos, los kits de robótica incluyen algún tipo de software focalizado en el aprendizaje de la programación, de forma que usuarios no expertos (e incluso niños pequeños) puedan no sólo construir un robot sino además programarlo para realizar diferentes tareas. A pesar de que estas herramientas están diseñadas especialmente para ayudar a introducir la robótica y la programación, la mayoría sufre de diversos problemas que reducen su efectividad.

El principal problema que se observa es que la interfaz de programación visual que presentan muchas de estas herramientas es efectiva para el aprendizaje en un estadío inicial, pero puede resultar contraproducente una vez que el estudiante deba incursionar en un lenguaje de programación de propósito general. Al concentrarse exclusivamente en la interfaz visual muchas herramientas introductorias provocan consecuencias negativas:

1. Los estudiantes tienden a considerar los lenguajes visuales (en particular, los lenguajes de bloques) como "programación de juguete" y no cómo "programación real" (Moors et al., 2018).
2. Los estudiantes que pasan de un lenguaje de bloques a aprender un lenguaje de programación basado en texto pueden sufrir un problema denominado "sobrecarga de sintaxis" (Moors et al., 2018).
3. Los estudiantes que aprenden inicialmente con lenguajes basados en bloques pueden potencialmente adquirir malos hábitos de programación que dificultan su capacidad para trabajar satisfactoriamente con lenguajes basados en texto (Meerbaum-Salant et al., 2011).

El segundo problema identificado está vinculado a la necesidad de programar dispositivos que se mueven e interactúan con objetos del mundo real. Normalmente el

entorno de programación para robótica educativa se ejecuta en una computadora que, mediante algún mecanismo de comunicación, permite transmitir el programa desarrollado al robot para su posterior ejecución. Al separar el entorno de programación (la computadora del estudiante) del entorno de ejecución (el robot), resulta difícil fomentar un estilo de programación interactivo que reduzca el tiempo entre que el usuario realiza un cambio al programa y se observan los efectos del mismo. Acortar este "feedback loop" tiene efectos positivos sobre el aprendizaje ya que fomenta la experimentación y la prueba a la vez que disminuye los efectos de la frustración ante los errores (Lodi et al., 2019) (Weintrop & Wilensky, 2015).

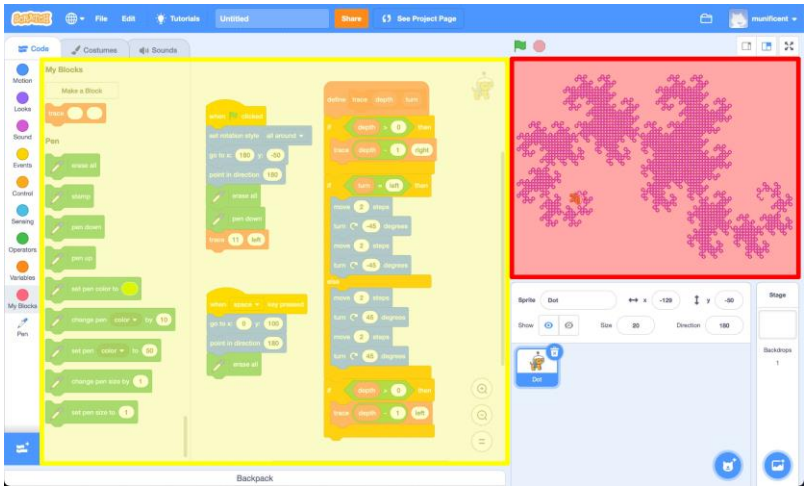

Figura 2. En Scratch tanto el entorno de programación (amarillo) como el espacio de simulación (rojo) forman parte del mismo programa. Esto no es posible en entornos de programación para robótica educativa.

Un tercer problema se refiere a la utilización de abstracciones inadecuadas para el dominio del problema o el nivel de aprendizaje de los alumnos. Un ejemplo claro de este problema es el soporte inadecuado para la concurrencia que tienen algunos entornos. A pesar de que la mayoría de las actividades en robótica educativa requieren la programación de un dispositivo capaz de realizar dos o más tareas en forma simultánea, muchos lenguajes no proveen ningún mecanismo para expresar esta concurrencia de forma sencilla, dejando a los usuarios la responsabilidad de organizar y coordinar el trabajo, muchas veces mediante complicadas máquinas de estado. Si bien el limitado soporte para concurrencia es una de las falencias más visibles en los entornos de programación para robótica educativa, no es el único caso de abstracciones inadecuadas. Otro ejemplo son los detalles del hardware soportado por el entorno, que en algunos lenguajes exponen complicaciones adicionales que podrían evitarse si el entorno los manejara de manera automática (opcionalmente ofreciendo una implementación por defecto que satisfaga los casos más comunes de uso). Se pueden mencionar: la configuración de los pines del GPIO, las interrupciones de hardware (para el control de encoders, por ejemplo), la comunicación con dispositivos I2C, entre otros. Por supuesto, la implementación de dicha funcionalidad es relevante para maximizar la utilidad de la herramienta, pero exponer los detalles de la implementación no necesariamente es pertinente a la educación del alumno. Es importante resaltar que la inserción de robots en las aulas es una excusa para ayudar a cumplir un objetivo mayor, no un fin en sí mismo (Barrera Lombana, 2015), por lo que es crucial evitar distraer al

alumno con conceptos innecesarios para que pueda concentrarse en aquellos fundamentales para el aprendizaje (Lopez et al., s. f.).

3. Solución Propuesta

Habiendo identificado entonces los principales problemas que afectan, en mayor o menor medida, a todos los entornos existentes para robótica educativa, la contribución principal del presente trabajo es doble: por un lado, se formularon una serie de principios de diseño y objetivos fundamentales que deberían servir de guía para futuros desarrollos en la materia y, por otro lado, se implementó un entorno de desarrollo integrado (IDE) para robótica educativa que sigue dichos principios y sirve como prueba de concepto de las ideas presentadas.

Los principios de diseño formulados son los siguientes:

1. Autonomía: El entorno debe permitir el desarrollo de programas que se almacenen y ejecuten en el robot de forma independiente, sin requerir una conexión constante con la computadora que se utilizó para la programación.
2. Interactividad: En caso de estar el robot conectado con el entorno, el mismo debe ser capaz de mandar cada cambio realizado al programa de forma automática, permitiendo ver sus efectos inmediatamente y acortar así el "feedback loop".
3. Monitoreo y adquisición de datos: El entorno debe ser capaz de mostrar y almacenar de forma clara y precisa el estado interno del robot y del programa.
4. Modelo híbrido bloques/código: El entorno debe presentar al usuario un lenguaje híbrido basado tanto en bloques como en código, de forma que se fomente una transición gradual hacia el aprendizaje de lenguajes de propósito general.
5. Concurrencia: El lenguaje de programación debe permitir expresar tareas concurrentes sin requerir del usuario la coordinación manual de las diferentes acciones del robot.
6. Depuración: El entorno debe incluir herramientas de depuración que permitan detener la ejecución del programa en cualquier momento, ejecutar las instrucciones paso a paso, y observar cómo el estado del programa cambia ante la ejecución de cada instrucción individual.

Como prueba de concepto, se desarrolló un entorno de programación para robótica educativa denominado Physical Bits (Moran, s. f.). Su implementación es completamente open source, está basado en tecnología web, es compatible con Arduino (una de las plataformas más populares para robótica educativa) y además cuenta con un diseño portable que permitiría soportar otros kits con cambios mínimos al código. Physical Bits fue diseñado bajo los principios formulados previamente, por lo que cuenta con características que lo distinguen de otras herramientas similares.

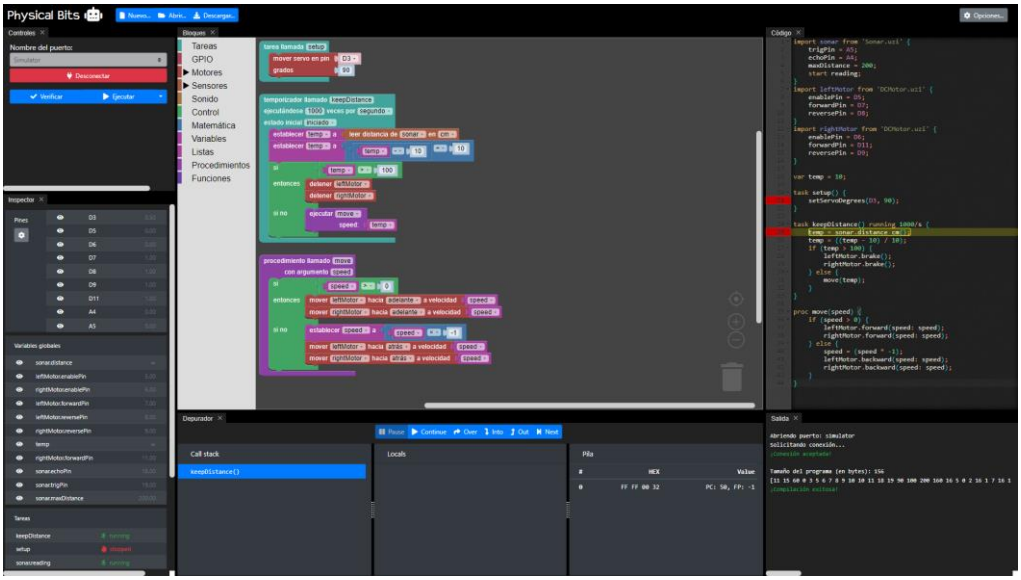

Figura 3. Entorno de desarrollo Physical Bits.

En primer lugar, el entorno soporta un modelo de programación híbrido basado en editores de bloques y de código. Este modelo está diseñado especialmente para facilitar la transición gradual de los alumnos a lenguajes de programación más sofisticados. La interfaz gráfica presenta ambos editores visibles al mismo tiempo y el usuario puede elegir cualquiera de ellos (o ambos) para desarrollar sus programas. Independientemente del editor elegido el entorno se encarga de mantener ambos editores sincronizados generando automáticamente los bloques o el código que sean necesarios.

En segundo lugar, el entorno usa un lenguaje de programación simplificado diseñado específicamente para robótica educativa. La sintaxis de este lenguaje está inspirada en la sintaxis del lenguaje C pero incluye además palabras clave y estructuras diseñadas especialmente para su uso en robótica educativa.

En tercer lugar, el entorno soporta programación interactiva sin sacrificar la autonomía del robot. Es decir que los cambios en el programa tienen un efecto visible inmediato en el comportamiento del robot y la ejecución de los programas se realiza siempre en el robot sin requerir una conexión constante con la computadora. Para ello, se desarrolló un firmware que se instala en el robot y que implementa una máquina virtual encargada de la ejecución de los programas del usuario.

Y finalmente, el entorno incluye herramientas de monitoreo y depuración, algunas de las cuales son prácticamente inexistentes en herramientas similares.

3. Trabajo relacionado

Para entender cómo se compara Physical Bits con otras herramientas similares se llevó a cabo un relevamiento de algunos entornos populares para la programación de robots con fines educativos. En el análisis se seleccionaron un conjunto pequeño de características relacionadas con los principios de diseño formulados previamente. Los resultados de dicho análisis fueron publicados previamente (Moran et al., 2021) pero se presentan a continuación en forma abreviada.

	Interactividad	Monitoreo	Depuración	Autonomía	Interfaz de programación
Physical Bits	✓	✓	✓	✓	Bloques + Código
MicroBlocks	✓	✓	✗	✓	Sólo bloques
Physical Etoys	✓	✓	✓	✗	Bloques + Código
Lego Mindstorms	✗	✓	✗	✓	Sólo bloques
Scratch4Arduino	✓	✓	✗	✗	Sólo bloques
Snap4Arduino	✓	✓	✗	✗	Sólo bloques
XOD	✗	✓	✗	✓	Sólo bloques
Ardublock	✗	✗	✗	✓	Sólo bloques
MakeCode	✗	✗	✗	✓	Bloques + Código
Arduino IDE	✗	✗	✗	✓	Sólo código

Aunque la lista de herramientas relevadas no es exhaustiva, esta selección ofrece una imagen representativa de las estrategias más comunes utilizadas en el desarrollo de entornos de programación para robótica educativa.

La mayoría de los entornos soporta programación visual en forma de bloques pero sólo algunos proveen generación automática del código. A excepción de Physical Bits y Physical Etoys (ambas herramientas desarrolladas por el mismo grupo de investigación), prácticamente ningún entorno provee herramientas de depuración. Y en lo que respecta a la autonomía y la interactividad, la mayoría de las herramientas soportan sólo una de las dos características, a excepción de Physical Bits y microBlocks que soportan ambas al mismo tiempo.

En este sentido, microBlocks (*MicroBlocks*, s. f.) es notable debido a que su implementación sigue un enfoque similar a Physical Bits: usa una máquina virtual para ejecutar programas de forma autónoma en el robot y, al mismo tiempo, brindar una experiencia interactiva y dinámica. Sin embargo, microBlocks no implementa ninguna herramienta de depuración y no es compatible con la programación basada en texto (aunque los autores expresan en su sitio web que es uno de sus objetivos para el futuro). Además, microBlocks soporta una variedad de dispositivos distinta a los que soporta Physical Bits. La máquina virtual de microBlocks está diseñada para microcontroladores de 32 bits, mientras que Physical Bits funciona en microcontroladores de 8 bits. Y microBlocks requiere al menos 12 Kb de RAM y 50 Kb de memoria Flash, mientras que Physical Bits requiere sólo 2 Kb de RAM y 24 Kb de memoria Flash.

Otro entorno interesante es XOD (*XOD*, s. f.). A diferencia de todos los demás entornos de esta lista, XOD es un lenguaje de flujo de datos. Utiliza bloques para representar nodos en un gráfico dirigido, en el que cada nodo puede representar una entrada (normalmente un sensor), algún cálculo, o una salida (normalmente un actuador). Los programas se crean vinculando nodos, que luego se utilizan para generar el programa nativo que el usuario puede cargar en la placa.

MakeCode (*Microsoft MakeCode for Micro*, s. f.) es un entorno web que admite tanto bloques como programación basada en texto utilizando Javascript, y proporciona

generación bidireccional tanto del código a partir de los bloques como de los bloques a partir del código. Aunque el entorno MakeCode no proporciona una experiencia de programación interactiva, compensa esta limitación proporcionando un simulador virtual, lo que hace posible ejecutar programas sin un dispositivo físico.

Finalmente, Physical Etoys (Zabala et al., 2010) es una extensión de Etoys desarrollada para interactuar con una amplia gama de dispositivos físicos como LEGO NXT, placas Arduino, joysticks innovadores como Microsoft Kinect o Leap Motion, etc. Permite programar robots usando bloques o un lenguaje de programación basado en texto (Smalltalk). Sin embargo, una vez que un guión (término utilizado por Etoys para referirse a los programas desarrollados por el usuario) ha sido modificado usando el modo textual, no admite volver a los bloques sin perder los cambios realizados. Este problema, sumado a la falta de autonomía, fue el disparador inicial que llevó al desarrollo del entorno Physical Bits.

4. Trabajo Futuro

Habiendo completado el desarrollo del prototipo queda pendiente una validación rigurosa de la propuesta mediante experimentos con usuarios finales. Si bien se han llevado a cabo talleres con alumnos de secundaria (los cuales han permitido mejorar sustancialmente la plataforma), todavía hacen falta más experiencias con usuarios para asegurar la efectividad de la herramienta en un contexto educativo.

Los talleres realizados han servido para resaltar diversos problemas de usabilidad en la interfaz de programación, así como identificar necesarias mejoras en las capacidades del entorno y del lenguaje. Todas estas mejoras fueron incorporadas al software, lo cual ha permitido publicar (hasta el momento) 10 versiones del entorno. Por limitaciones de espacio no se presentará en este artículo un detalle de los cambios realizados en cada versión, pero el lector interesado puede acceder a los mismos en el repositorio de código del proyecto (*Releases · GIRA/PhysicalBits*, s. f.).

Referencias

Argentina Programa: Más de 157 mil inscriptos en 7 días. (2020, noviembre 1). Argentina.gob.ar. https://www.argentina.gob.ar/noticias/argentina-programa-mas-de-157-mil-inscriptos-en-7-dias

Barrera Lombana, N. (2015). USO DE LA ROBÓTICA EDUCATIVA COMO ESTRATEGIA DIDÁCTICA EN EL AULA. *Praxis & Saber*, *6*(11), 215-234.

El Gobierno presentó un plan para formar 111.000 programadores jóvenes y adultos— LA NACION. (s. f.). Recuperado 2 de abril de 2022, de https://www.lanacion.com.ar/tecnologia/el-gobierno-presento-un-plan-para-formar-111000-programadores-jovenes-y-adultos-nid1969143/

Factorovich, P., & Sawady O'Connor, F. (2017). *Actividades para aprender a Program.AR* (I. Miller & T. Alberto, Eds.; Vol. 1). Fundación Sadosky. https://program.ar/descargas/manual-docente-descarga-web-v2017.pdf

Lodi, M., Malchiodi, D., Monga, M., Morpurgo, A., & Spieler, B. (2019). Constructionist Attempts at Supporting the Learning of Computer Programming: A Survey. *Olympiads in Informatics*, *13*, 99-121. https://doi.org/10.15388/ioi.2019.07

Lopes Guedes, A., Lopes Guedes, F., & Guedes Laimer, A. C. (2015). Experiencias de robótica educativa / Experiences with Educational Robot. *Revista Internacional de Tecnología, Ciencia y Sociedad*, *4*(2). https://doi.org/10.37467/gka-revtechno.v4.887

Lopez, P. E. M., Bonelli, E. A., & Sawady O'Connor, F. A. (s. f.). *El nombre verdadero de la programación*. 20.

MakeCode for micro:bit. (s. f.). Microsoft MakeCode for Micro:Bit. Recuperado 29 de mayo de 2022, de https://makecode.microbit.org/

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2011). Habits of programming in scratch. *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education - ITiCSE '11*, 168. https://doi.org/10.1145/1999747.1999796

MicroBlocks. (s. f.). Recuperado 29 de mayo de 2022, de https://wwww.microblocks.fun/

Moors, L., Luxton-Reilly, A., & Denny, P. (2018). Transitioning from Block-Based to Text-Based Programming Languages. *2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*, 57-64. https://doi.org/10.1109/LaTICE.2018.000-5

Moran, R. (s. f.). *Physical Bits*. Physical Bits - A web-based programming environment for educational robotics that supports live coding and autonomy using a hybrid blocks/text programming language. Recuperado 29 de mayo de 2022, de https://gira.github.io/PhysicalBits/

Moran, R., Teragni, M., & Zabala, G. (2021). Physical Bits: A Live Programming Environment for Educational Robotics. En W. Lepuschitz, M. Merdan, G. Koppensteiner, R. Balogh, & D. Obdržálek (Eds.), *Robotics in Education* (pp. 291-303). Springer International Publishing. https://doi.org/10.1007/978-3-030-67411-3_26

Releases · GIRA/PhysicalBits. (s. f.). GitHub. Recuperado 29 de mayo de 2022, de https://github.com/GIRA/PhysicalBits/releases

Vega-Moreno, D., Cufí Solé, X., Rueda, M. J., & Llinás, D. (2016). *Integración de robótica educativa de bajo coste en el ámbito de la educación secundaria para fomentar el aprendizaje por proyectos*. https://rio.upo.es/xmlui/handle/10433/3504

Weintrop, D., & Wilensky, U. (2015). To block or not to block, that is the question: Students' perceptions of blocks-based programming. *Proceedings of the 14th International Conference on Interaction Design and Children*, 199-208. https://doi.org/10.1145/2771839.2771860

XOD. (s. f.). Recuperado 29 de mayo de 2022, de https://xod.io/

Zabala, G., Morán, R., & Blanco, S. (2010, mayo). *Physical Etoys: Una herramienta libre para el aprendizaje de tecnología con material concreto*. V Congreso de Tecnología en Educación y Educación en Tecnología. http://sedici.unlp.edu.ar/handle/10915/18405