Physical Bits: un Entorno de Programación Web para Robótica Educativa

Ricardo Moran[1,2], Gonzalo Zabala[1], Matías Teragni[1]

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

[2]Comisión de Investigaciones Científicas de la Provincia de Buenos Aires, Calle 526 e/ 10 y 11, La Plata, Buenos Aires, República Argentina

{ricardo.moran, gonzalo.zabala, matias.teragni}@uai.edu.ar

Abstract. El uso de dispositivos físicos como herramientas de aprendizaje presenta diversos desafíos. Muchos entornos de programación ayudan en el aprendizaje eliminando la posibilidad de errores de sintaxis por medio de un lenguaje visual. Sin embargo, la sintaxis es sólo uno de los aspectos del proceso de aprendizaje. 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 Physical Bits, un entorno de programación web para robótica educativa que intenta solucionar estos problemas mediante una experiencia interactiva que permite aprovechar tanto programación visual como basada en texto.

Keywords: Robótica Educativa, Programación Visual, Arduino.

1 Introducción

El uso de robots en las aulas ha crecido significativamente en los últimos años. Parte de este crecimiento se debe a políticas públicas que han sabido reconocer la importancia y los beneficios de la introducción de material concreto como recurso didáctico. Podemos mencionar experiencias como el plan "Aprender Conectados" [1] propuesto por el Ministerio de Educación Nacional, "Todos a la robótica" [2] en la provincia de San Luis, o el programa de Robótica de la Provincia de Buenos Aires [3], entre otros.

A su vez, la proliferación de kits de robótica orientados a usuarios no expertos facilitó la implementación de dichas propuestas. Entre los kits más populares se destaca la plataforma de hardware Arduino [4] que, siendo abierta y de bajo costo, se ha convertido muy rápidamente en una de las tecnologías más utilizadas para proyectos de este estilo.

La plataforma Arduino provee un entorno de desarrollo simplificado (basado en el lenguaje de programación C++) en el que muchos conceptos avanzados de programación de microcontroladores están "escondidos" al usuario. Sin embargo, dado que este entorno es todavía demasiado complejo para algunos de los usuarios menos experimentados (sobre todo los niños más pequeños), han surgido múltiples intentos de proveer un entorno de programación más adecuado para principiantes [5] [6] [7] [8]. Aunque estas herramientas ayudan a introducir la robótica y la programación, la mayoría sufre de diversos problemas que reducen su efectividad.

En primer lugar, muchas de las herramientas ofrecen interfaces de programación visual, pero fallan en considerar la eventual transición a lenguajes de programación basados en texto. Esto tiene como consecuencia un efecto contrario al esperado. En lugar de motivar a los alumnos a aprender programación, el salto en dificultad es tan grande que la experiencia resulta frustrante [9] [10] [11].

En segundo lugar, la mayoría de las herramientas disponibles se pueden clasificar en una de dos categorías: o soportan lenguajes compilados con mínimas posibilidades de interacción, o son entornos completamente interactivos, pero sin la capacidad de ejecutar programas de forma autónoma en el robot. Aquellos que caen en la primera categoría carecen de los mecanismos básicos de análisis de ejecución que permiten hacer frente a los eventuales errores de código, mientras que los segundos pueden presentar estas capacidades, pero al costo de reducir ampliamente el tipo de proyectos aplicables ya que siempre se requiere una computadora conectada.

En tercer lugar, uno de los principales problemas que se observan al trabajar en la enseñanza de la programación es la dificultad que encuentran los alumnos para predecir correctamente el efecto que los cambios en el código tienen en el comportamiento del programa. Algunos estudios muestran que muchos errores que cometen los estudiantes al aprender programación surgen por una conceptualización imprecisa del modelo de ejecución que presenta el lenguaje [12]. Para resolver estos problemas muchas herramientas de programación introductorias ofrecen entornos virtuales interactivos donde los cambios en el programa tienen un efecto inmediato visible para el usuario, fomentando así la experimentación y la prueba [13]. Sin embargo, esta interactividad no está presente en la misma medida en entornos de programación para robótica educativa [14], y los casos donde sí se soporta se lo hace en detrimento de la autonomía del robot.

2 Solución Propuesta

Para resolver estos problemas se creó Physical Bits, un entorno de desarrollo integrado (IDE) para robótica educativa basado en tecnología web y completamente open source. 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.

Esto es posible debido a que la arquitectura de la solución está compuesta por: un firmware instalado en el robot que se encarga de la ejecución de los programas del usuario; un middleware que contiene las herramientas de compilación, comunicación y

depuración del dispositivo; y un entorno de desarrollo web que permite hacer uso de dichas herramientas para programar los robots desde una computadora o dispositivo móvil.

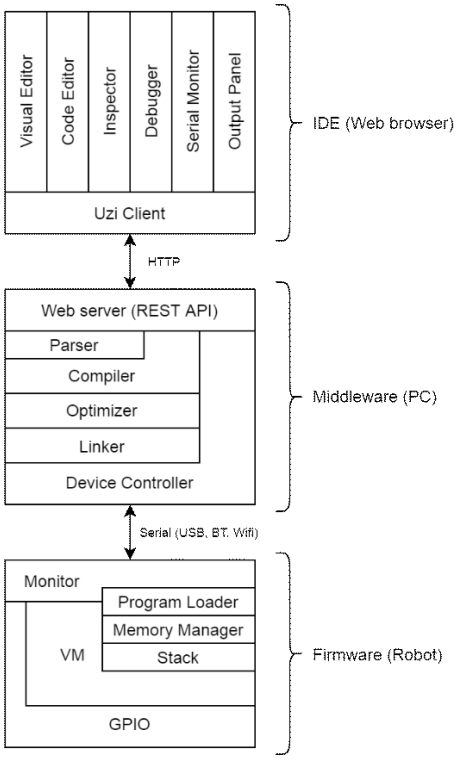

Fig. 1. La arquitectura de la solución

En lo que respecta a la interfaz de programación, se diseñó un lenguaje específico de dominio (DSL) para robótica educativa. Se optó por implementar un DSL de diseño propio en lugar de usar un lenguaje de programación de propósito general para poder controlar los elementos del lenguaje que se presentan al usuario, facilitando una introducción gradual a la programación. El lenguaje propuesto cuenta con una selección muy limitada de conceptos que se consideran esenciales para el aprendizaje tanto de robótica como de los fundamentos de la programación [15]. Entre ellos se incluyen: estructuras de control básicas (secuencia, decisión e iteración), variables, funciones y procedimientos. Para facilitar su uso en robótica, se agregó al lenguaje la posibilidad de definir tareas concurrentes y funciones primitivas que facilitan el acceso a los pines del Arduino. El lenguaje fue diseñado para permitir extender la funcionalidad del programa mediante librerías externas. Se han desarrollado librerías para simplificar el uso de sensores y actuadores comunes como, por ejemplo: motores de corriente continua, motores paso a paso, sensores ultrasónicos, botones, joysticks, sensores de temperatura, entre otros. Una descripción completa del lenguaje, ejemplos de código, y la definición

de la gramática exceden el alcance del presente artículo, pero los mismos pueden encontrarse en el sitio web del proyecto[1].

Se implementaron dos interfaces de programación: una visual (basada en programación por bloques, permitiendo la programación sin la necesidad de lidiar con las complejidades de una sintaxis rígida) y otra textual (inspirada en el lenguaje C, simplificando el salto que deben realizar los estudiantes para utilizar luego lenguajes de más bajo nivel). Ambas interfaces fueron diseñadas para usarse en simultáneo, proveyendo generación automática de una representación a partir de la otra. El usuario puede modificar tanto los bloques como el código y el sistema se encarga de mantener ambas representaciones sincronizadas de forma automática.

IDE

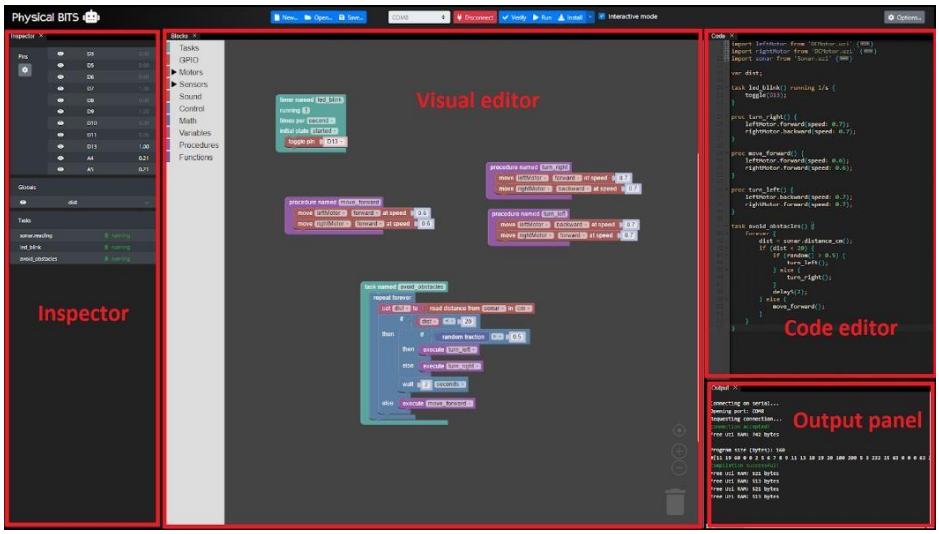

Fig. 2. El IDE y sus distintos paneles

El entorno de desarrollo integrado unifica bajo una única interfaz web las distintas herramientas, incluyendo:

1. Un editor de código y un editor de bloques que permiten al usuario desarrollar programas y enviarlos al robot,
2. Un inspector que permite observar las tareas en ejecución, el valor de las variables del programa, y el estado de los pines del robot,
3. Un depurador que facilita la resolución de errores mediante la posibilidad de pausar el programa y ejecutar el código paso a paso,
4. Y una consola de salida que notifica al usuario de los mensajes del sistema.

[1] https://github.com/GIRA/UziScript

Para la implementación del lenguaje visual se decidió utilizar la librería Blockly, que facilitó la creación y manipulación de los bloques. El lenguaje visual se diseñó con el objetivo de facilitar la programación de los robots y de ayudar en la posterior transición a lenguajes basados en texto. Para cumplir este objetivo se agregó la opción de visualizar el texto de los bloques en modo código en lugar de lenguaje natural. Cuando esta opción está activada el alumno está efectivamente escribiendo código, sólo que el entorno permite hacerlo arrastrando y soltando bloques. Esta opción, sumada a la visualización automática del código generado, ayuda a realizar una transición gradual a lenguajes de programación basados en texto.

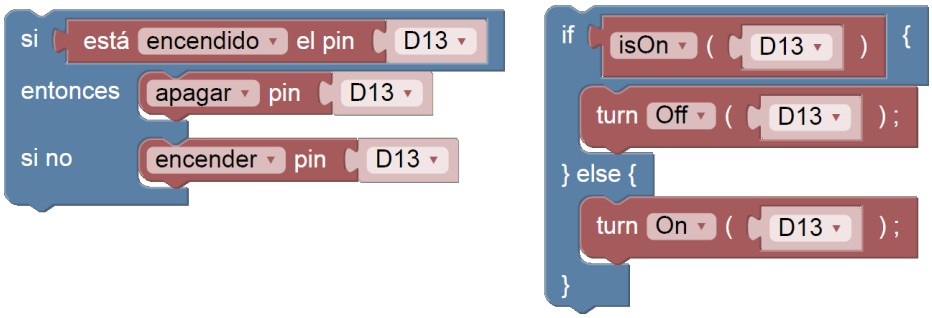

Fig. 3. Bloques en modo texto y en modo código

Por otro lado, para facilitar el uso del entorno en alumnos que recién están aprendiendo a leer se agregó también una opción que muestra en mayúsculas todo el texto de la interfaz gráfica (incluyendo botones, etiquetas, títulos, y también el texto en los bloques).

Finalmente, el IDE soporta también un "modo interactivo", en el cual se aprovecha la velocidad de compilación que ofrece el middleware para mandar al robot automáticamente y de manera inmediata todos los cambios que se realizan en el programa. Tanto el “modo interactivo” como el inspector, que provee una visualización constante y automática del estado interno del robot, fomentan la experimentación y la prueba, lo cual a su vez ayuda a los alumnos a aprender a predecir correctamente el impacto que el código fuente va a tener en las acciones del robot. De esta forma, el entorno asiste al alumno, que está aprendiendo a conceptualizar un modelo de ejecución, haciendo explícita la relación entre los cambios en el programa y el comportamiento del robot.

Middleware

El middleware tiene como responsabilidades principales: (1) gestionar la comunicación con el firmware, y por lo tanto con el dispositivo físico; (2) compilar los programas desarrollados por el usuario; y (3) exponer dichas funcionalidades a interfaces web mediante una API REST [16] y websockets [17].

El compilador se encarga de transformar el código fuente del programa escrito en nuestro DSL y generar los bytecodes en el formato que el firmware puede decodificar y ejecutar.

Dadas las restricciones de memoria que presentan las placas Arduino, es fundamental que los programas generados por el compilador sean lo más compactos posibles. Para ello se diseñó el conjunto de instrucciones teniendo en cuenta la frecuencia en que se usa cada instrucción en el contexto de la robótica educativa y se implementaron en el compilador optimizaciones orientadas a minimizar el tamaño de los programas en lugar de maximizar la eficiencia de la ejecución.

A continuación, se puede observar un programa sencillo en algunas de sus posibles representaciones: bloques, código fuente, instrucciones emitidas por el compilador, y bytes a ser transmitidos por el puerto serie.

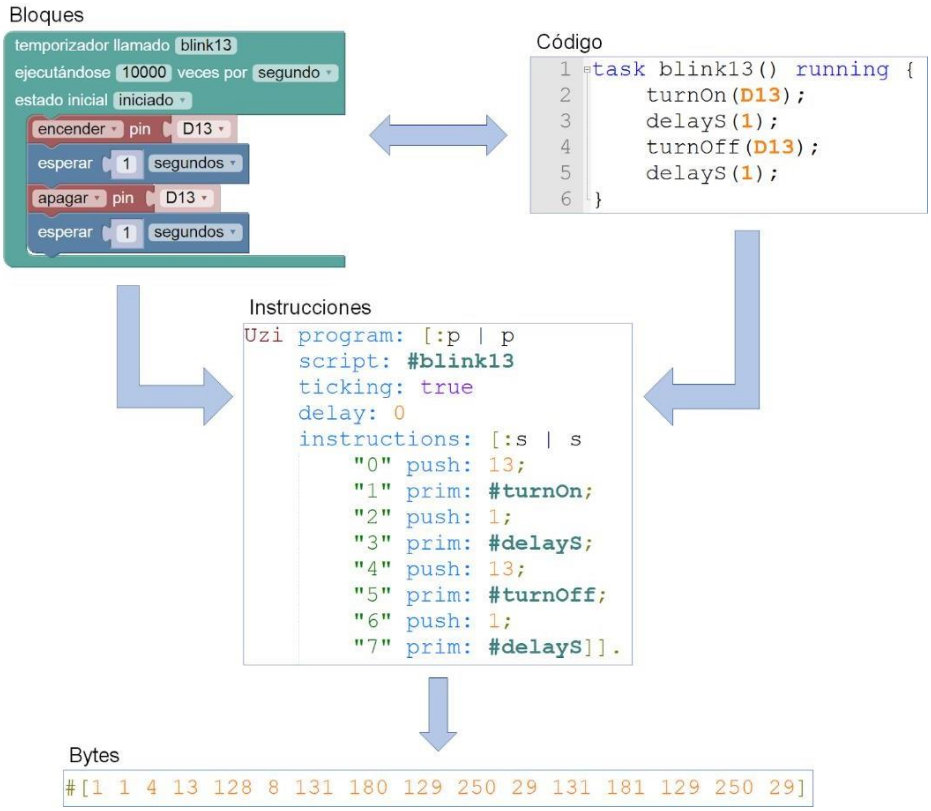

Fig. 4. Diferentes representaciones de un mismo programa

Firmware

El firmware es un programa tradicional de Arduino escrito en C++ que se puede instalar en el dispositivo usando las herramientas estándar de Arduino. Se diseñó de esta forma para minimizar la complejidad de la instalación, permitiendo que cualquiera pueda, con un esfuerzo mínimo, instalar el firmware en su placa usando las herramientas y la documentación estándar de la plataforma.

La implementación actual soporta múltiples placas Arduino diferentes, desde la UNO (la más popular y estándar), pasando por Micro y Nano (versiones en miniatura), MEGA 2560 (con mayor capacidad y cantidad de pines) y Yun (que cuenta con soporte para wifi). También se han recibido reportes de pruebas realizadas de forma exitosa en otras placas compatibles con Arduino como: DuinoBot [18], Educabot [19], y Totem-DUINO [20].

Como se puede observar en el diagrama de la arquitectura, el firmware incluye varios componentes. Entre ellos, los tres más importantes son: el Monitor, la máquina virtual (VM), y la GPIO.

El Monitor se encarga de la interacción con el middleware (normalmente una computadora) mediante el puerto serie, ya sea conectando al dispositivo mediante USB o Bluetooth. Este programa implementa un protocolo de comunicación que le permite al firmware reaccionar a comandos enviados por el middleware e informar, periódicamente, el estado interno del dispositivo (incluyendo valores de los pines y variables, tareas en ejecución, uso de memoria, etc.).

La VM es una implementación de un intérprete de bytecodes [21]. Dado que este proyecto tiene fines educativos, se eligió esta implementación privilegiando su simplicidad por sobre el rendimiento. La VM implementa, a su vez, la administración automática de la memoria del programa y un scheduler cooperativo que permite coordinar la ejecución de múltiples tareas concurrentes en un solo microcontrolador.

Finalmente, el componente GPIO provee una abstracción del hardware que simplifica el manejo de los pines del dispositivo y facilita la portabilidad.

Aprovechando estos tres componentes, el sistema puede ejecutar programas de forma autónoma en diversos dispositivos sin sacrificar la interactividad característica de los lenguajes educativos y, a su vez, permitiendo la implementación de herramientas de depuración comúnmente disponibles en lenguajes de propósito general.

3 Experiencias con usuarios

Se llevó a cabo un taller introductorio de programación y robótica usando Physical Bits en un colegio secundario en la localidad de La Plata. En el taller participaron 12 estudiantes de entre 15 y 18 años. Los estudiantes tenían poco conocimiento de programación. Habían desarrollado algunas páginas web con HTML, pero nunca habían programado con Arduino ni con ningún otro tipo de robot.

El taller duró aproximadamente una hora y media, durante la cual se introdujeron los conceptos principales de la robótica, los tipos de robots y los componentes que los conforman. Luego se mostró el entorno de Physical Bits usando como ejemplo entradas digitales y analógicas.

Luego de la demostración se dividió a los estudiantes en grupos de 3 a 5 personas y a cada grupo se le dio un robot basado en Arduino. Debido a la disponibilidad de materiales cada robot tenía características diferentes de modo que hubo que ajustar los ejercicios para cada grupo de acuerdo con el robot que les hubiera tocado.

Estos ejercicios fueron diseñados de forma que fomenten la experimentación con la plataforma en lugar de buscar un objetivo concreto. Los estudiantes debían trabajar por

sí mismos todo lo que pudieran, pero tenían permitido consultar a los docentes cuando necesitaran ayuda.

Los resultados de la actividad variaron entre los diferentes participantes. El grupo más avanzado llegó a aprender cómo controlar tanto servomotores como motores de corriente continua mientras que el grupo que menos progreso obtuvo sólo logró prender y apagar una serie de LEDs formando patrones rítmicos. Sin embargo, no se llevó una cuenta precisa del progreso de cada grupo dado que no era ese el objetivo de la actividad. Por el contrario, interesaba más la experiencia general y la reacción de los alumnos al entorno. En ese sentido, el grupo que sólo pudo manipular LEDs fue también el que más se divirtió y el que reaccionó con mayor entusiasmo ante las posibilidades que les ofrecía la plataforma.

El rol de los docentes en el taller consistió principalmente en observar a los estudiantes a medida que interactuaban con el entorno, tomando nota de los errores comunes que cometían. Esta experiencia resultó muy útil para identificar algunos errores de usabilidad en la interfaz gráfica que ya fueron corregidos.

Al finalizar la actividad se les solicitó a los participantes que llenaran una encuesta anónima con la finalidad de entender lo que sintieron sobre la experiencia y sobre el software utilizado.

Sorpresivamente, el 50% de los estudiantes manifestó que los ejercicios eran muy difíciles, pero aproximadamente el 80% expresó que disfrutaron mucho de la actividad. Un 60% manifestó interés por la robótica y un 80% interés por la programación en general. Considerando que esta fue su primera experiencia programando robots se cree que los resultados son prometedores.

Además de este taller, el entorno ha sido utilizado para introducir a la robótica a estudiantes universitarios dentro del laboratorio, algunos de los cuales ahora forman parte del grupo de investigación.

A partir de estas experiencias se ha podido mejorar sustancialmente la plataforma y, aunque todavía hacen falta más experiencias con usuarios para asegurar la efectividad de la herramienta, se considera que la misma está lista para ser utilizada en el aula por cualquier docente interesado.

4 Limitaciones

La implementación actual cuenta con tres grandes limitaciones. La principal limitación resulta ser la performance. Se estima un overhead de entre 60x y 100x respecto del mismo programa ejecutado de forma nativa, aunque el impacto de esta lentitud resulta poco importante dados los fines educativos del proyecto.

En segundo lugar, el tamaño de los programas se encuentra severamente limitado. Dado el poco espacio de almacenamiento presente en microcontroladores como el Arduino UNO (2 KB de RAM y 1 KB EEPROM), los programas del usuario no pueden superar ciertos límites en lo que refiere a cantidad de tareas, cantidad de instrucciones, cantidad de variables, etc. Si bien el tipo de problemas de robótica educativa a los que apunta esta solución no requiere programas demasiado complejos, esto restringe la utilidad del entorno para resolver problemas más sofisticados.

Finalmente, el lenguaje fue diseñado con el objetivo de servir como herramienta introductoria a la programación y, por lo tanto, no soporta algunas características comunes en lenguajes de programación más sofisticados. Por ejemplo, no hay soporte para estructuras de datos ni objetos. Si bien no se ha terminado de explorar la utilidad del lenguaje en su versión actual se cree que la implementación de dichas características podría ser interesante como trabajo futuro.

5 Conclusión y trabajo futuro

En este artículo se identificaron los problemas más comunes en las herramientas de programación para robótica educativa y se presentó Physical Bits, un entorno de desarrollo integrado diseñado especialmente para resolver dichos problemas. Si bien se ha llegado a un punto en el desarrollo en que se puede empezar a usar esta herramienta con alumnos, el proyecto aún no está terminado. Los resultados expuestos en este artículo sugieren que el enfoque elegido es prometedor, pero todavía hacen falta más experiencias con usuarios para garantizar su efectividad en el aula.

En cuanto a mejoras a la implementación, se está considerando incorporar nuevas placas además de Arduino, en particular ESP8266 y ESP32, las cuales permitirían su uso en contextos conectados a internet.

Se están evaluando también algunas mejoras tanto al lenguaje de programación como al protocolo de comunicación. Sería especialmente útil que el lenguaje ofreciera mecanismos para extender su repertorio de operaciones primitivas.

Finalmente, se cree que esta solución tiene potencial para convertirse en una plataforma de robótica educativa integral. Con este objetivo se está empezando el desarrollo de un repositorio en la nube donde docentes de todo el país puedan compartir sus proyectos y actividades.

Referencias

1. Ministerio de Educación - Presidencia de la Nación, 2017. [En línea]. Available: http://www.bnm.me.gov.ar/giga1/documentos/EL005852.pdf. [Último acceso: 19 Junio 2020].
2. R. A. Munizaga, C. Moleker, G. Yonzo y G. Zabala, «Everyone to Robotics: an educational robotics project in Argentina,» *RoboCup 2013,* 2013.
3. C. M. Banchoff Tzancoff, S. Martín, S. Gómez y F. E. M. López, «Experiencias en robótica educativa: diez años trabajando con escuelas argentinas,» de *XIV Congreso Nacional de Tecnología en Educación y Educación en Tecnología (TE&ET 2019)*, Universidad Nacional de San Luis, 2019.
4. Arduino, «Arduino Playground - Structure,» [En línea]. Available: http://playground.arduino.cc/ArduinoNotebookTraduccion/Structure. [Último acceso: 23 Julio 2017].
5. A. Pina y C. Iñaki, «Primary Level Young Makers Programming & Making Electronics with Snap4Arduino,» *Educational Robotics in the Makers Era,* pp. 20-33, 2017.

6. Citilab, «About S4A,» 2015. [En línea]. Available: http://s4a.cat/. [Último acceso: 15 Junio 2017].
7. Grupo de Investigación en Robótica Autónoma del CAETI (GIRA), «Physical Etoys,» 2010. [En línea]. Available: http://tecnodacta.com.ar/gira/projects/physical-etoys/. [Último acceso: 15 Junio 2017].
8. «Ardublock | A Graphical Programming Language for Arduino,» [En línea]. Available: http://blog.ardublock.com/. [Último acceso: 15 Junio 2017].
9. D. Weintrop y U. Wilensky, «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,* p. 199–208, 2015.
10. L. Moors, A. Luxton-Reilly y P. Denny, «Transitioning from Block-Based to Text-Based Programming Languages,» *2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE),* pp. 57-64, 2018.
11. K. Powers, S. Ecott y L. Hirshfield, «Through the looking glass: teaching CS0 with Alice,» *ACM SIGCSE Bulletin,* vol. 39, nº 1, p. 213–217, 2007.
12. M. Lodi, D. Malchiodi, M. Monga, A. Morpurgo y B. Spieler, «Constructionist Attempts at Supporting the Learning of Computer Programming: A Survey,» *Olympiads in Informatics: An International Journal, Vilnius University, International Olympiad in Informatics,* pp. 19-121, 2019.
13. P. Rein, S. Ramson, J. Lincke, R. Hirschfeld y T. Pape, «Exploratory and Live, Programming and Coding: A Literature Study Comparing Perspectives on Liveness,» *The Art, Science, and Engineering of Programming,* vol. 3, nº 1, 2019.
14. L. Cabrera, J. Maloney y D. Weintrop, «Programs in the Palm of your Hand: How Live Programming Shapes Children's Interactions with Physical Computing Devices,» *Proceedings of the 18th ACM International Conference on Interaction Design and Children,* p. 227–236, 2019.
15. P. E. Martínez López, E. A. Bonelli y F. A. Sawady O'Connor, «El nombre verdadero de la programación: Una concepción de ense˜nanza de la programación para la sociedad de la información,» Universidad Nacional de Quilmes.
16. M. Masse, REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces, O'Reilly Media, Inc., 2011.
17. I. Fette y A. Melnikov, The websocket protocol, RFC 6455, December, 2011.
18. A. Rojas, «Reporte Robótica Educativa,» *Universidad Nacional de La Pampa (UNLPam),* 2017.
19. «Educabot,» [En línea]. Available: https://educabot.org/. [Último acceso: 13 12 2019].
20. Totem, «TotemDUINO | Totemmaker.net,» Totemmaker.net, [En línea]. Available: https://totemmaker.net/product/totemduino-arduino/. [Último acceso: 13 12 2019].
21. J. Smith y R. Nair, Virtual Machines: Versatile Platforms for Systems and Processes, San Francisco, CA: Morgan Kaufmann Publishers Inc., 2005.