Physical Bits: Un entorno de programación para robótica educativa

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. *Este artículo describe un proyecto de doctorado que tiene como objetivo el diseño y la implementación de un entorno de desarrollo integrado para robótica educativa. Este entorno, denominado Physical Bits, posee un conjunto de características que permiten ofrecer a los alumnos una experiencia de programación interactiva basada en un lenguaje híbrido bloques/código. Se realizó un relevamiento de herramientas similares y se compararon sus características, demostrando el carácter único de este entorno. Finalmente, se describen los resultados luego de aplicar el entorno en un taller de robótica con alumnos de secundaria.*

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.

1.1 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.

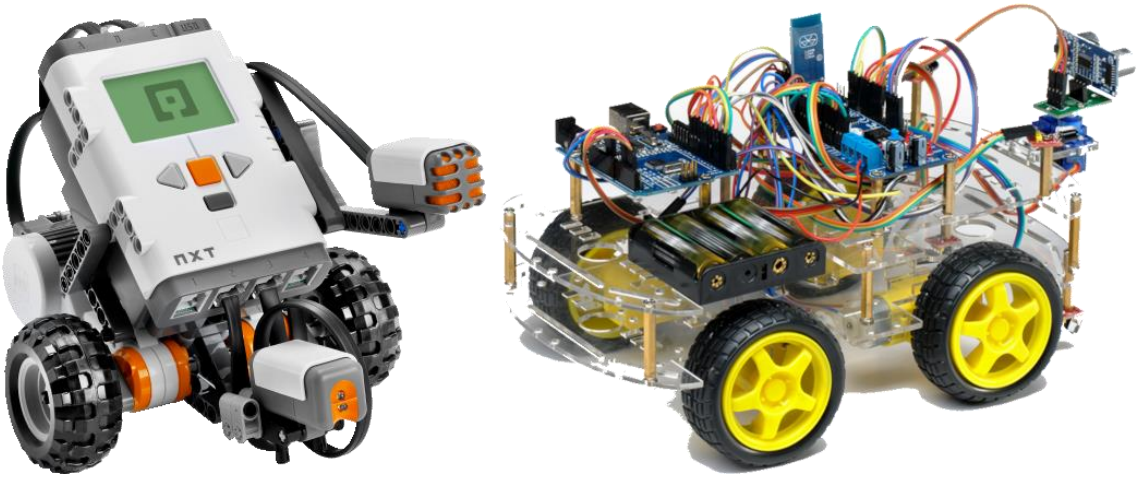


Fig 1 Ejemplos típicos de kits de robótica: Lego Mindstorms NXT (izquierda) y Arduino (derecha)

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).

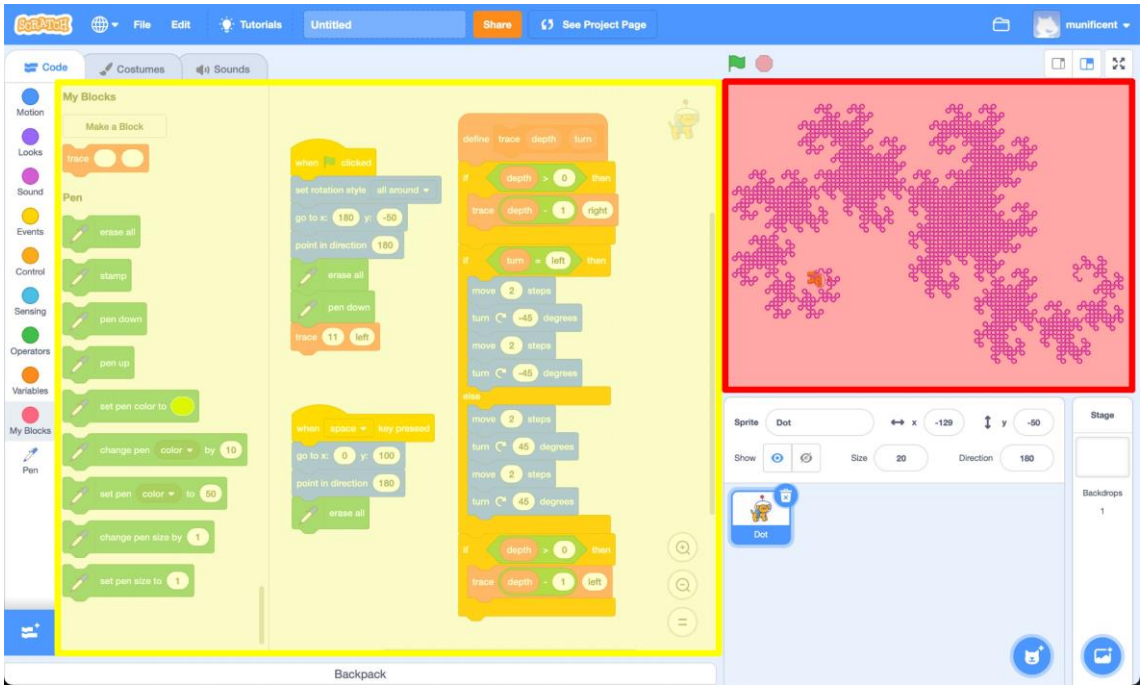


Fig 2 En Scratch tanto el entorno de programación (amarillo) como el espacio de simulación (rojo) forman parte del mismo programa. Esto no es posible en entornos de programación para robótica educativa.

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.).

1.2 Solución

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 et al., 2021). 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.

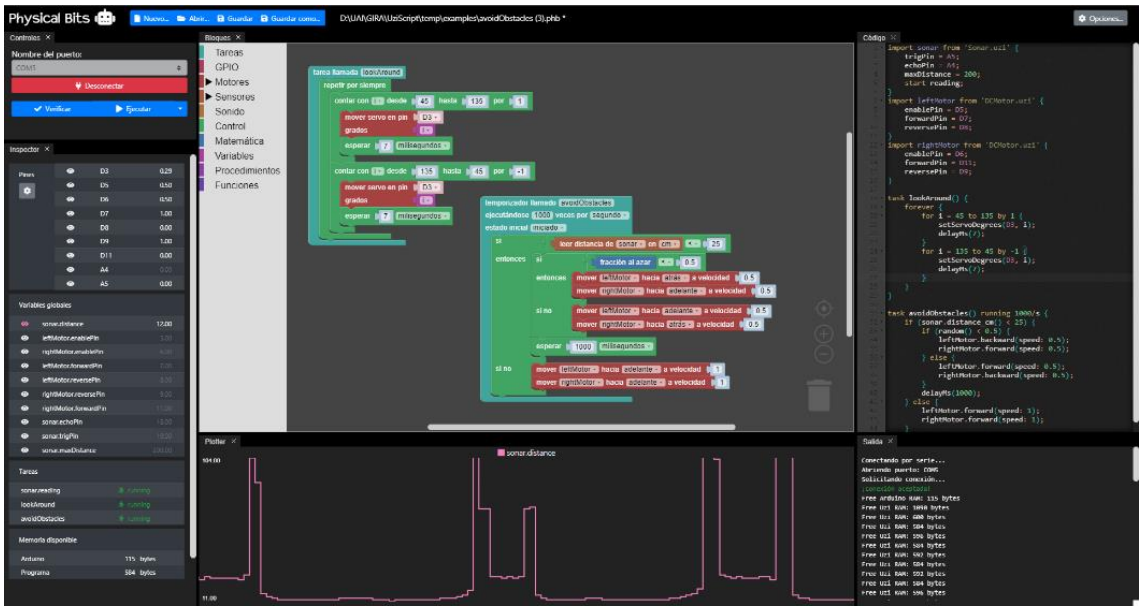


Fig 3 Entorno de desarrollo Physical Bits

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.

2. Trabajo Relacionado

En los últimos años se ha observado una introducción gradual de la robótica en las aulas, motivado en parte por el creciente interés por el aprendizaje de la programación. Este fenómeno ha dado lugar al surgimiento de diversos kits y herramientas diseñados específicamente para usuarios no expertos. Sin embargo, aún quedan preguntas sin respuesta en cuanto a las características de estas herramientas. Para responder estas preguntas, se realizó un relevamiento de la literatura buscando artículos describiendo el uso de entornos de programación para robótica educativa. En la siguiente tabla se presentan todos los entornos agrupados en función de las características soportadas por cada lenguaje.

Para poder comparar con Physical Bits, se analizaron 5 características clave, relacionadas con los principios de diseño formulados previamente:

1. Autonomía (A): ¿El programa del usuario se almacena y ejecuta en el robot?
2. Concurrencia (C): ¿Las herramientas utilizadas soportan la ejecución de múltiples tareas de forma concurrente?
3. Monitoreo (M): ¿Puede el usuario observar el estado interno del robot (valores de variables y sensores) desde el entorno de programación?
4. Interactividad (I): ¿Las herramientas utilizadas soportan programación interactiva sin intervención manual del usuario?
5. Depuración (D): ¿Puede el usuario detener la ejecución en un punto arbitrario del programa y continuar ejecutando paso a paso?

Adicionalmente, se clasificó cada entorno por el tipo de lenguaje soportado, pudiendo ser éste visual, textual, o híbrido.

Como se puede observar en la tabla, no se encontraron entornos de programación que combinen las 5 características analizadas. Y de los entornos que combinan al menos 3 características, ninguno presenta una interfaz híbrida visual/textual (los entornos que soportan un lenguaje de programación híbrido se muestran subrayados). A partir de este relevamiento, se puede concluir que Physical Bits es una herramienta única en su tipo.

	Herramientas	**A**	**C**	**M**	**I**	**D**
1	ScratchX, MicroBlocks	✔	✔	✔	✔	✘
2	Aseba, LabView	✔	✔	✔	✘	✔
3	Phogo	✘	✔	✔	✔	✔
4	Choregraphe, BlocklyTalky, Enchanting, LEGO Mindstorms, XOD, Microsoft VPL	✔	✔	✔	✘	✘
5	Talkoo toolkit, Scratch/Snap/S4A, LEGO WeDo	✘	✔	✔	✔	✘
6	Thymio VPL	✔	✘	✔	✘	✔
7	Snek	✔	✘	✔	✔	✘

	Herramientas	A	C	M	I	D
8	Modkit, PyNXC, MakeCode, NQC/NXC, RoboLab, RobotC	✔	✔	✘	✘	✘
9	Sphero Edu	✘	✔	✔	✘	✘
10	KinderBot (iPad app)	✔	✘	✘	✔	✘
11	BIPES	✔	✘	✔	✘	✘
12	Arduino IDE (C/C++), Ardublockly, Crumble, PROTEAS, mBlock 3, PICAXE Programming Editor, Tern, ZR graphical editor, CHERP	✔	✘	✘	✘	✘
13	LearnBlock, VIPLE, EUD-MARS	✘	✔	✘	✘	✘
14	TITIBOTS, MODEBOTS	✘	✘	✘	✘	✘

3. Experiencias con Alumnos

Se llevaron a cabo 3 talleres para alumnos externos al laboratorio donde se pudo comprobar la efectividad del entorno en la enseñanza de la programación. Además de estos talleres, el entorno ha sido utilizado satisfactoriamente para introducir a la robótica a estudiantes universitarios dentro del laboratorio, algunos de los cuales luego formaron parte del grupo de investigación y contribuyeron al desarrollo del proyecto.

En el último taller, realizado en Agosto de 2022, participaron 19 alumnos de entre 17 y 18 años, todos provenientes del mismo curso. El taller consistió de 4 encuentros de 4 horas cada uno, y se realizó un encuentro por semana. Los alumnos se dividieron en 2 grupos:

1. el grupo experimental (Martes), que recibió la versión completa del entorno.
2. el grupo de control (Jueves), que recibió el entorno configurado con algunas características deshabilitadas: la interactividad, las herramientas de monitoreo, y las herramientas de depuración.

Además de deshabilitar algunas características para el grupo de control, se modificó el entorno utilizado de forma que permita registrar las acciones de los usuarios (de ambos grupos) y poder así sacar conclusiones de los datos recolectados.

Interesa responder las siguientes preguntas:

RQ1 ¿Cómo impactan la interactividad, las herramientas de monitoreo, y las herramientas de depuración en el progreso de los alumnos?

RQ2 ¿Cómo impactan la interactividad, las herramientas de monitoreo, y las herramientas de depuración en el tiempo de resolución de cada ejercicio?

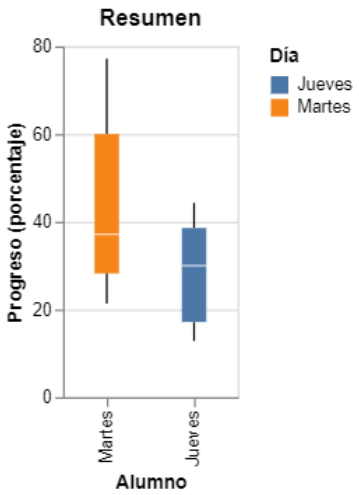
RQ3 ¿Cómo impactan la interactividad, las herramientas de monitoreo, y las herramientas de depuración en la forma de trabajar de los estudiantes?

RQ4 ¿Cómo impactan la interactividad, las herramientas de monitoreo, y las herramientas de depuración en la duración del “feedback loop”?

Dada la cantidad limitada de participantes y la breve duración de la actividad, los datos recolectados no son suficientes como para obtener un resultado estadísticamente significativo. Sin embargo, se observan tendencias interesantes para analizar.

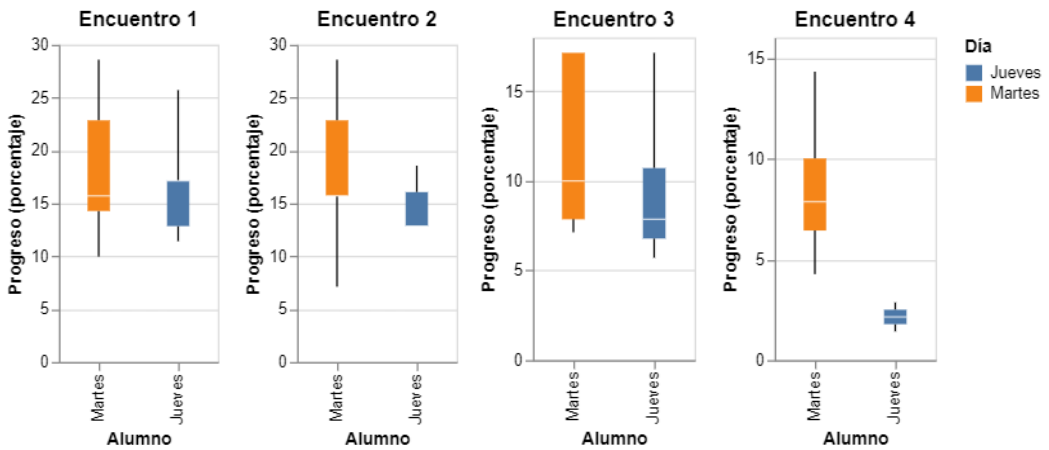
RQ1 Impacto en el progreso de los alumnos

Para calcular el progreso de cada estudiante se calculó el porcentaje de ejercicios resueltos sobre el total (70 ejercicios). El siguiente gráfico muestra un resumen del progreso de los dos grupos durante los 4 encuentros que tuvo el taller.



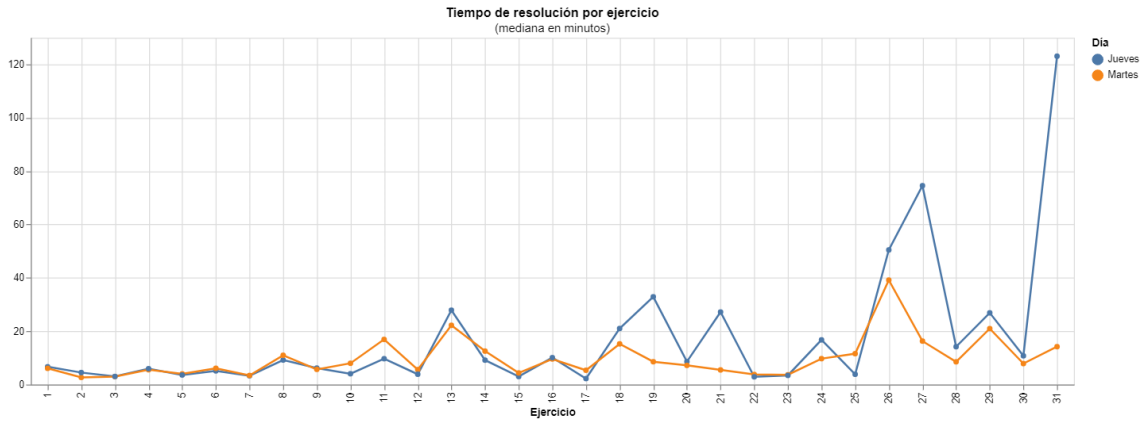
Tanto para el grupo de control como para el experimental se observa un progreso similar, siendo en promedio superior para el grupo experimental. Se puede destacar también que el alumno que llegó más lejos del grupo experimental alcanzó a resolver el 77.14% de los ejercicios, mientras que el más avanzado del grupo de control llegó tan sólo a resolver el 44.28%.

Dado que los ejercicios realizados en el taller fueron ordenados siguiendo una curva de dificultad ascendente (con algunas excepciones, por ejemplo, cuando se introducía un nuevo concepto) tiene sentido analizar el progreso de los alumnos encuentro por encuentro. De esta forma, se puede observar que la mayor diferencia se presenta en el último encuentro, donde el grupo experimental tuvo en promedio un progreso 4 veces mayor que el grupo de control.



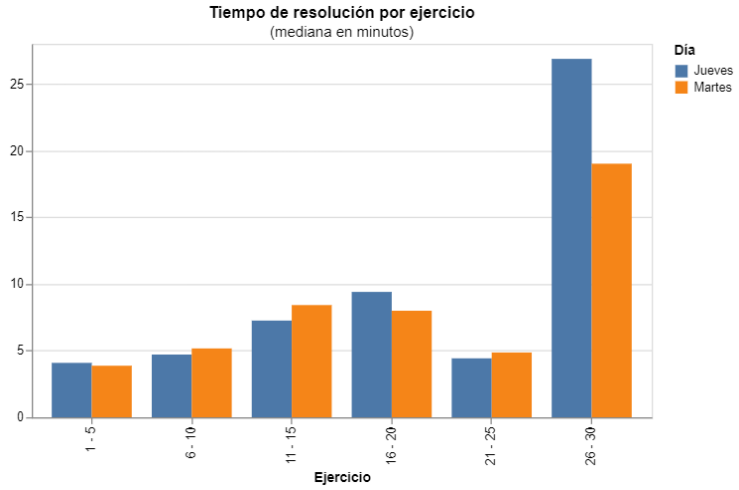
RQ2 Impacto en el tiempo de resolución de cada ejercicio

Para responder esta pregunta se puede observar el siguiente gráfico, que muestra el tiempo de resolución para cada ejercicio hasta el ejercicio 31 (el último que fue resuelto por al menos 1 alumno de ambos grupos).



Como se puede observar, no hay una diferencia significativa en el tiempo de resolución de los ejercicios iniciales, pero a medida que se incrementa la dificultad comienzan a aparecer diferencias entre los grupos.

El siguiente gráfico muestra la misma información que los gráficos anteriores, pero con los ejercicios en grupos de 5 y excluyendo el ejercicio 31 (cuyo valor atípico para el grupo de control distorsiona los datos).



Como se puede observar, para los ejercicios más sencillos no se observan diferencias notables en el tiempo de resolución. Pero conforme avanza la complejidad de los ejercicios también se observa que el grupo de control demora más en resolverlos, sugiriendo que el beneficio de las características de Physical Bits es mayor cuanto más complejo es el problema a resolver.

RQ3 Impacto en la forma de trabajar de los estudiantes

Para analizar el trabajo de los estudiantes se utilizaron principalmente 2 métricas: el nivel de actividad y la eficiencia.

Para estimar el nivel de actividad de cada alumno se definió la siguiente fórmula:

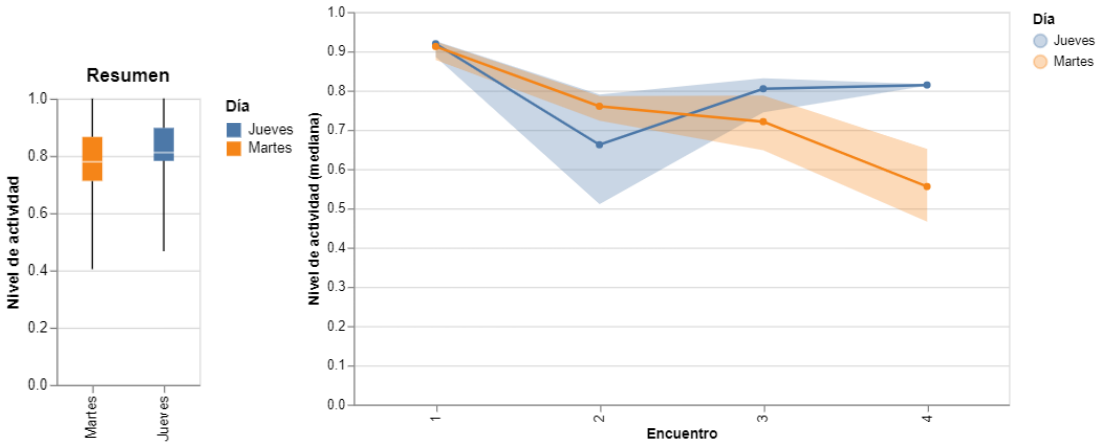
$$a = \frac{ws}{ts}$$

Siendo:

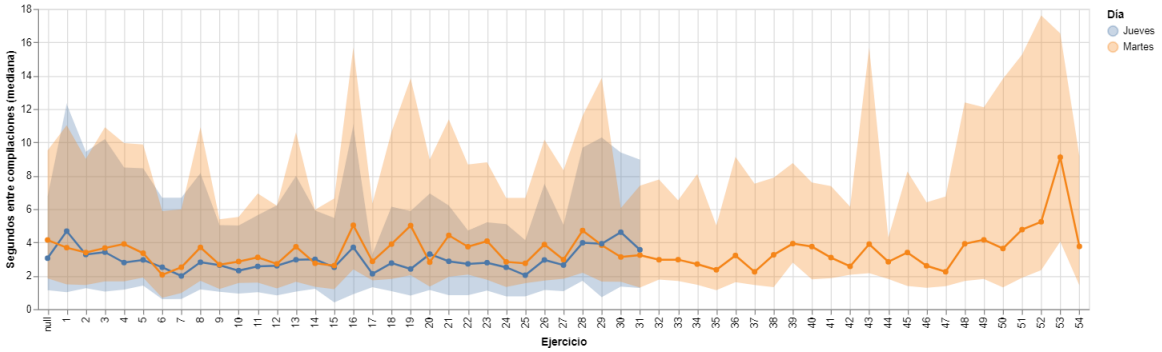
- ts (total seconds) la sumatoria del tiempo total del encuentro (incluyendo sólo los encuentros que el alumno haya asistido).
- ws (working seconds) el resultado de sustraer del total aquellos intervalos donde no se registraron eventos durante un período mayor a 1 minuto.

De esta forma, un valor cercano a 1 implica que el alumno no se distrajo en ningún momento, y un valor cercano a 0 significa que perdió el tiempo durante la mayor parte de la actividad.

Cómo se puede observar, el nivel de actividad de ambos grupos fue muy similar, llegando incluso en el último encuentro a ser inferior para el grupo experimental.



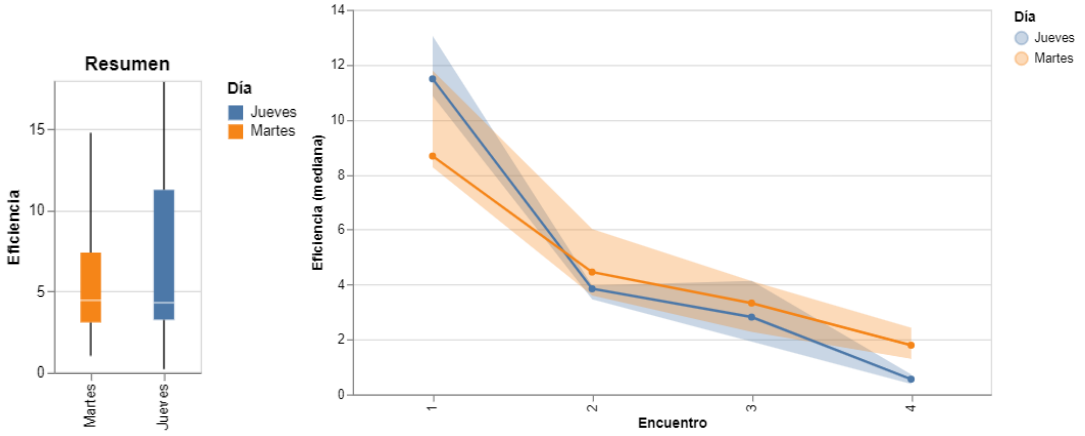
Si en cambio se toma en cuenta el tiempo entre compilaciones la historia es muy similar, como se ve en el gráfico siguiente. Todo esto parece sugerir que los alumnos de ambos grupos trabajaron aproximadamente al mismo ritmo.



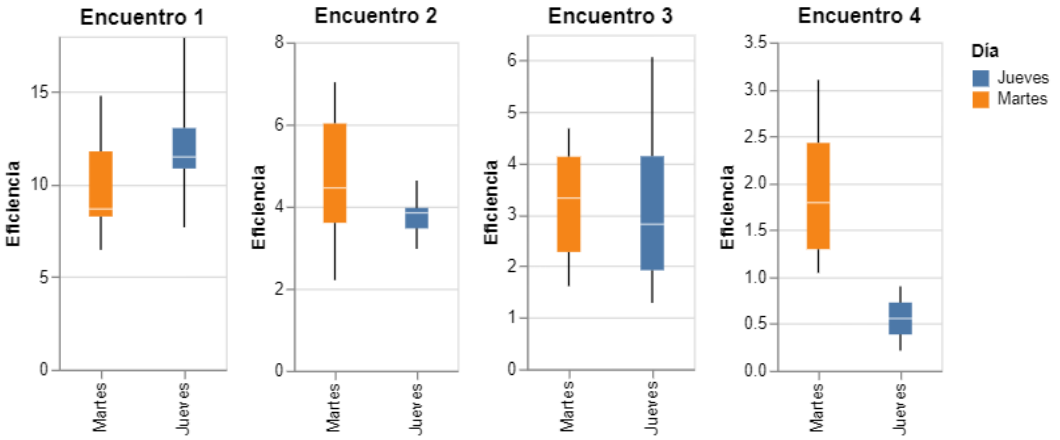
Por otro lado, la estimación de eficiencia de cada alumno se calculó contando cuántos ejercicios pudo resolver por hora.

En este caso, si se observa el resumen de todos los encuentros se ve que en promedio la eficiencia fue muy parecida entre ambos grupos, resultando en un rápido descenso en la eficiencia conforme avanzan los encuentros. En consonancia con los

gráficos anteriores, tiene sentido considerando el incremento en dificultad de los ejercicios.



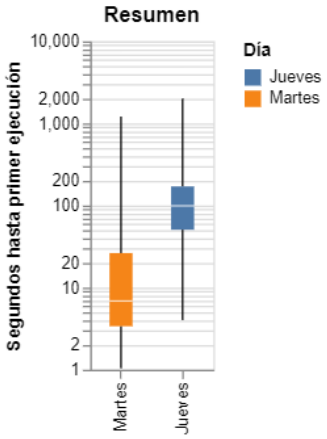
Sin embargo, cuando se analiza cada encuentro por separado se puede observar que aunque en el primer encuentro el grupo de control muestra una eficiencia superior, en los próximos encuentros su eficiencia media está siempre por debajo del grupo experimental, incrementándose la brecha sobre todo en el último encuentro (donde la eficiencia del grupo experimental triplicó a la del grupo de control).



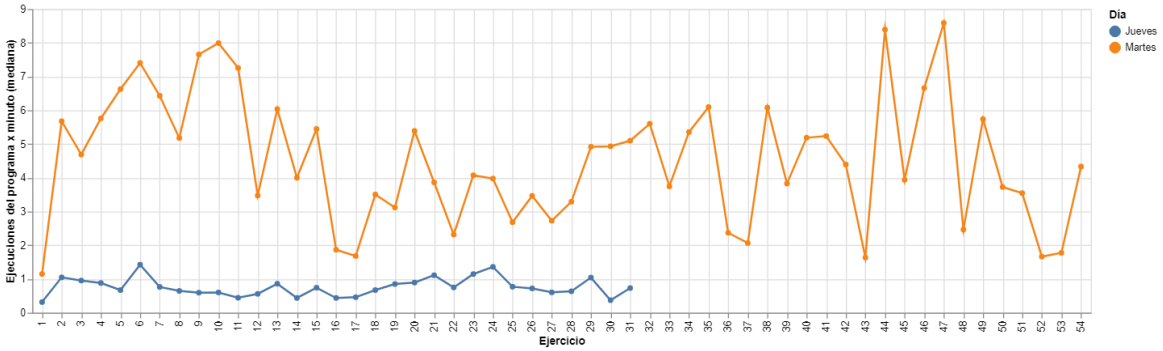
RQ3 Impacto en la duración del "feedback loop"

En lo que respecta al "feedback loop" la forma más fácil de medirlo es considerar el tiempo desde que cada alumno empieza a trabajar en un ejercicio hasta que puede ver el programa ejecutándose en el robot.

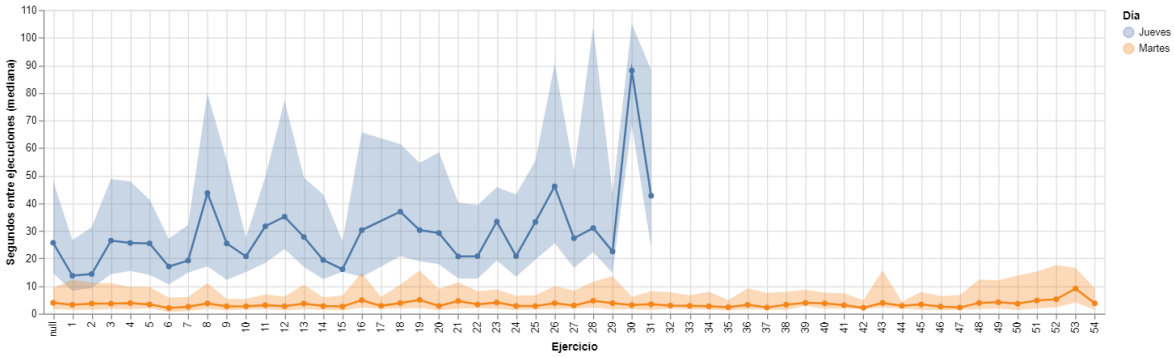
En ese sentido, se puede observar que el grupo experimental tardó, en general, mucho menos en ejecutar el programa por primera vez (nótese la escala logarítmica). Como demuestra este gráfico, el tiempo desde que se empieza a resolver un problema hasta tener un programa corriendo en el robot es en promedio un orden de magnitud menor que en el grupo de control.



Otra forma de analizar la duración del "feedback loop" es considerar las ejecuciones por minuto. En este caso también se hace evidente la diferencia entre el grupo experimental y el grupo de control.



Finalmente, se puede calcular también el tiempo entre ejecuciones del programa, lo cual se muestra en el gráfico siguiente. De forma similar a los gráficos anteriores, el grupo experimental exhibe ejecuciones mucho más frecuentes que el grupo de control.



Lo que estos gráficos sugieren es la importancia de la interactividad para resolver problemas: un entorno interactivo implica menos trabas para probar los programas, lo cual a su vez fomenta ejecuciones más frecuentes, rápidas, y en general eso redunda en un mayor progreso en la resolución de los ejercicios.

4. Conclusiones

Luego de completar el desarrollo del prototipo y habiendo llevado a cabo talleres con alumnos de secundaria se puede concluir que el proyecto es apto para ser utilizado como herramienta pedagógica en contextos educativos. Los talleres realizados, además, 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.

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

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., 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

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