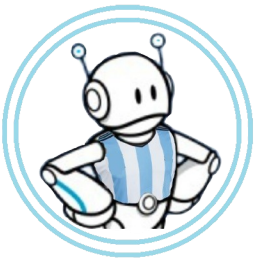
Actas de las Jornadas Argentinas de Robótica (JAR2024)

del 4 al 7 de Junio

Buenos Aires, Argentina

BUENOS AIRES
JAR
2024
del 4 al 7 de junio



Sesión 11: Robótica Educativa

El impacto de la programación interactiva en actividades de robótica educativa

Ricardo Moran
Centro de Altos Estudios en Tecnología Informática
Universidad Abierta Interamericana
Buenos Aires, Argentina
ricardo.moran@uai.edu.ar

Gonzalo Zabala
Centro de Altos Estudios en Tecnología Informática
Universidad Abierta Interamericana
Buenos Aires, Argentina
gonzalo.zabala@uai.edu.ar

***Resumen*—La programación interactiva es usualmente considerada beneficiosa para el aprendizaje debido a que reduce el costo de la experimentación, facilitando la programación exploratoria. Para medir el impacto de la programación interactiva en robótica educativa, realizamos un estudio piloto con estudiantes de secundaria. Desarrollamos dos configuraciones distintas del mismo entorno de programación: una con todas las funciones del lenguaje habilitadas y otra con las funciones de programación interactiva, monitoreo y depuración desactivadas. Utilizamos estos dos entornos para comparar el rendimiento de un grupo de estudiantes durante un taller introductorio de robótica. Los resultados encontrados son consistentes con algunos estudios similares: la programación interactiva puede ayudar a los estudiantes a progresar más, mejorar su eficiencia, reducir el tiempo requerido para resolver cada tarea y aumentar la frecuencia con la que se prueban los programas en los robots.**

Keywords—Programación interactiva, Robótica educativa

I. INTRODUCCIÓN

La programación interactiva es usualmente considerada beneficiosa para el aprendizaje debido a que reduce el costo de la experimentación, facilitando la programación exploratoria [1]. Permite un estilo de programación caracterizado por el ensayo y error, fomentando la prueba de diferentes ideas y acortando el intervalo entre la concepción de una idea y su validación. Asimismo, los beneficios de la programación interactiva en educación radican en hacer visible el comportamiento dinámico de los programas, facilitando así su comprensión e inspección [1]. Aunque la mayoría de los entornos de programación visual (como Scratch, Etoys, Alice y otros) han sido diseñados para ofrecer una experiencia de aprendizaje interactiva, donde el feedback inmediato es una característica fundamental y las modificaciones al programa se evalúan y reflejan automáticamente en la pantalla, un énfasis comparable en la inmediatez y la transparencia no ha sido tan prevalente en los entornos de programación de robótica educativa.

En las últimas décadas, ha habido un aumento constante en el interés y la posterior adopción de entornos de programación visual y kits de robótica específicamente diseñados para principiantes [2]. Y aunque la robótica naturalmente crea un bucle de retroalimentación donde los resultados del programa se traducen en acciones del mundo real por parte del robot, las herramientas para programar estos robots a menudo carecen del mismo nivel de interactividad observado en las contrapartes puramente virtuales. La mayoría de los lenguajes de programación educativos para robótica requieren que los usuarios compilen y envíen sus programas al robot antes de poder ver sus efectos en el mundo real (por ejemplo, Arduino, LEGO Mindstorms, MakeCode, entre otros) [3][4]. En otros casos, el feedback que los estudiantes pueden obtener está limitado por la falta de herramientas de monitoreo que muestren el estado interno del robot, obligando a los principiantes a "imaginar" lo que está sucediendo con sus robots. Esto puede ser especialmente difícil cuando el robot no hace lo que el usuario espera. Finalmente, ciertos entornos proporcionan un bucle de retroalimentación rápido y una experiencia interactiva, pero esto a menudo se hace a expensas de la autonomía del robot [5][6][7].

Para abordar este problema, hemos desarrollado Physical Bits [8], un entorno de programación basado en web para robótica educativa que admite programación interactiva y autonomía mediante un lenguaje de programación híbrido de bloques y texto. Con el objetivo de medir el impacto de la programación interactiva en robótica educativa, realizamos un estudio piloto con estudiantes de secundaria utilizando este entorno de programación. Al diseñar nuestro propio entorno, pudimos ajustar fácilmente las características del lenguaje, como ocultar herramientas de monitoreo o desactivar la programación interactiva. Aprovechando esta adaptabilidad, desarrollamos dos configuraciones distintas: una con todas las funciones del lenguaje habilitadas y otra con las funciones de programación interactiva, monitoreo y depuración desactivadas. Luego dividimos a los estudiantes en dos grupos (cada uno utilizando una configuración diferente) y comparamos su rendimiento durante un taller introductorio sobre robótica. Este artículo analiza los resultados de este estudio.

II. TRABAJO RELACIONADO

Varios trabajos previamente han analizado la importancia de la programación interactiva y del feedback continuo, con resultados diversos.

Un estudio que compara la efectividad de desarrolladores de JavaScript trabajando con un entorno interactivo con aquellos que trabajan sin él encontró que aquellos participantes que utilizaron programación interactiva detectaron y corrigieron errores más rápido, realizaron modificaciones al código uniformemente distribuidas en el tiempo y verificaron el código con mayor frecuencia sin una disminución en el tiempo de finalización de la tarea [9].

Otro estudio intentó verificar la efectividad de proporcionar feedback en tiempo de ejecución automáticamente después de cada modificación del programa [10]. Los resultados sugieren que presentar feedback al usuario (ya sea automática o manualmente) ayuda a desarrollar programas con menos errores sintácticos y semánticos. Sin embargo, la visualización automática del feedback no muestra beneficios significativos en comparación con el feedback manual, siempre y cuando éste se presente sin demora.

Un estudio independiente intentó evaluar el impacto del feedback continuo en la tarea de depuración. Sus hallazgos indican que si bien el feedback visual continuo no muestra una mejora significativa en la depuración en general, sí demuestra un efecto positivo significativo en circunstancias específicas[11].

Estudios adicionales han destacado la importancia de mantener un bucle de retroalimentación corto durante el aprendizaje de programación, postulando que ver inmediatamente los efectos de los cambios en el programa anima a los usuarios a experimentar con el código [12][13].

Finalmente, se llevó a cabo un estudio piloto con el objetivo de comprender el impacto de la programación interactiva en las interacciones con dispositivos de computación física, con niños de entre 11 y 15 años [14]. En este estudio utilizaron dos entornos de programación comparables: uno compatible con la programación interactiva y otro no. Los hallazgos indican que la interactividad realmente influye en las interacciones de los niños tanto en el juego libre como en las actividades estructuradas. Sin embargo, los autores señalan una limitación en su estudio: el uso de entornos de programación distintos, lo que podría haber impactado en los resultados, especialmente en términos de las tasas de éxito dentro de cada grupo.

Hasta la fecha, no ha habido ningún estudio que evalúe la influencia de la programación interactiva en el rendimiento de un grupo de estudiantes utilizando el mismo entorno de programación para robótica educativa.

III. Metodología

Se decidió realizar un taller extracurricular para estudiantes de secundaria. En el mismo participaron 19 alumnos de entre 17 y 18 años, todos provenientes del mismo curso. Los alumnos expresaron haber tenido poca o nula experiencia en programación.

El taller consistió de 5 encuentros de 4 horas cada uno, y se realizó un encuentro por semana. Durante la primera sesión, se presentó la actividad y se llevó a cabo un pretest para estimar el conocimiento previo de cada estudiante. El pretest consistió principalmente en ejercicios simples de lógica y programación. Los resultados del pretest se utilizaron para dividir a los estudiantes en 2 grupos, procurando mantener puntajes similares en ambos grupos: el grupo experimental (E), que recibió la versión completa del entorno; y el grupo de control (C), que recibió el entorno configurado con algunas características deshabilitadas (la interactividad, las herramientas de monitoreo, y las herramientas de depuración). Aunque el género no fue un factor a considerar en la separación de los estudiantes, en el grupo experimental la proporción fue equitativa, con una distribución de 5 varones y 5 mujeres, mientras que en el grupo de control participaron 6 varones y 3 mujeres.

Para simplificar el problema y facilitar que se concentren sólo en la programación se dió a cada alumno una placa de pruebas que incluía ya resuelta toda la electrónica necesaria para resolver los ejercicios. La placa estaba compuesta por un Arduino Nano, 3 LEDs, 2 botones, 1 potenciómetro, 1 LDR, y pines para conectar servomotores y motores de corriente continua. El haber resuelto la electrónica previamente permitió una mejor utilización del tiempo de actividad, lo que llevó a la definición de una serie de 70 ejercicios. Estos ejercicios fueron diseñados con la intención de suavizar la curva de dificultad y minimizar el impacto de la frustración en los alumnos.

El diseño inicial de la actividad contemplaba comenzar el taller explicando los conceptos elementales de programación y robótica usando solo la placa de pruebas conectada por USB a la computadora, y luego, ya en los últimos encuentros, anexar una estructura de ruedas que permitiera convertir a la placa de pruebas en un robót móvil. Sin embargo, la limitación de tiempo y el progreso de los estudiantes en la resolución de los ejercicios no permitió alcanzar el objetivo.

Todas las sesiones comenzaron con una breve presentación por parte del instructor explicando los temas a abordar en los ejercicios del día. Después de la presentación, cada estudiante se sentó en su computadora y comenzó a trabajar en los ejercicios de forma individual. Para facilitar la tarea del instructor, se desarrolló una aplicación que presenta los ejercicios y permite a los estudiantes "enviar" sus soluciones [15]. Una vez completada la actividad, la tarea del instructor consistió en recopilar los archivos generados por la aplicación. Se grabaron videos de ejemplo para explicar las consignas, asegurando que ningún estudiante tuviera que depender únicamente de su propia interpretación; en su lugar, tenían un ejemplo funcional para ver y comparar con su implementación. La aplicación, además, permitió garantizar una resolución estrictamente secuencial de los ejercicios, y se encargaba de registrar la hora en que se enviaba la solución de cada ejercicio, facilitando la tarea posterior de segmentar los eventos correspondientes a cada ejercicio. Sin embargo, la verificación de soluciones no pudo automatizarse con la aplicación, por lo que cada estudiante tuvo que esperar la verificación del instructor antes de pasar al siguiente ejercicio.

IV. Preguntas de investigación

El objetivo de esta experiencia es medir el impacto de la interactividad en la resolución de ejercicios de programación y robótica. Para ello, se buscan responder las siguientes preguntas de investigación:

- RQ1: ¿Cómo impacta la interactividad en el progreso de los alumnos?
- RQ2: ¿Cómo impacta la interactividad en el tiempo de resolución de cada ejercicio?
- RQ3: ¿Cómo impacta la interactividad en la forma de trabajar de los estudiantes?
- RQ4: ¿Cómo impacta la interactividad en la duración del bucle de retroalimentación?

V. Ejercicios

Se planificaron 70 ejercicios a completarse durante el transcurso de 4 sesiones. Si bien esta es una lista de ejercicios ambiciosa para un período de tiempo tan corto, experiencias previas con estudiantes sugieren que una curva de dificultad gradual sin saltos abruptos en complejidad es preferible. El escaso tiempo disponible para la actividad implicó una limitación al diseño del taller, imposibilitando trabajar en problemas más libres o proyectos de mayor duración. Como se puede observar en la tabla 1, los conceptos a enseñar durante el taller incluyeron temas relacionados con la robótica (entradas y salidas, motores, sensores, etc.) y la programación (bucles, condicionales, variables, etc.).

Se decidió organizar estos conceptos en una jerarquía de "subconjuntos de lenguaje" donde cada subconjunto es

completo e incluye los subconjuntos anteriores [16]. La lista completa de ejercicios puede verse online en el siguiente link: https://richom.github.io/pbits-exercises/

TABLE I. CONCEPTOS A EXPLICAR Y RANGO DE EJERCICIOS

Conceptos	Ejercicios
Salidas digitales Delays con valores constantes Repetición	1 a 10
Entradas analógicas Multitasking Delays con valores variables	11 a 20
Entradas digitales Condicionales Operaciones lógicas y comparación	21 a 40
Manejo de botones Generación de audio Operaciones aritméticas Uso de variables	41 a 60
Servomotores Motores de corriente continua	61 a 70

VI. ENTORNO DE PROGRAMACIÓN

El entorno de programación que se utilizó en el taller, denominado Physical Bits, es un desarrollo propio que incluye una serie de características que lo distinguen de otras herramientas similares.

En primer lugar, el entorno soporta un modelo de programación híbrido basado en editores de bloques y de código que se pueden utilizar simultáneamente. Este modelo está diseñado especialmente para facilitar la transición gradual de los alumnos a lenguajes de programación más sofisticados.

En segundo lugar, el entorno usa un lenguaje de programación simplificado diseñado específicamente para robótica educativa y basado en la sintaxis de C.

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. La comunicación entre el robot y el entorno de programación sólo es necesaria para la programación, y puede realizarse por USB o por Bluetooth.

Finalmente, el entorno incluye además herramientas de monitoreo y depuración, algunas de las cuales son prácticamente inexistentes en otras herramientas similares.

VII. LECCIONES APRENDIDAS

Haber preparado la electrónica con anticipación permitió establecer un número sustancial de ejercicios, aproximadamente 20 por día. Al final, aunque los estudiantes no pudieron completar toda la actividad, el alto número de ejercicios ayudó a suavizar la curva de dificultad, introduciendo conceptos con un ejercicio simple y luego complicándolo gradualmente con problemas más sofisticados, combinándolo con otros conceptos y requiriendo una lógica más compleja. Sin embargo, el inconveniente de este enfoque es que algunos estudiantes se quejaron de la monotonía de los ejercicios, y el entusiasmo que caracteriza a los talleres de robótica no fue tan evidente al finalizar el taller.

A pesar de diseñar los ejercicios para tener un aumento gradual en la dificultad, ninguno de los estudiantes logró completar los ejercicios más complejos, que involucraban motores y una programación más sofisticada. Quizás hubo demasiados ejercicios para tan poco tiempo, y podría haber sido preferible tener menos ejercicios pero más desafiantes.

Por otro lado, la automatización de las entregas utilizando una aplicación resultó ser un éxito. Aunque llevó tiempo desarrollarla, posteriormente simplificó el trabajo en las clases, permitiendo que el instructor se enfocara sólo en responder preguntas y verificar la corrección de las soluciones. Desafortunadamente, el proceso de verificación no pudo automatizarse, por lo que dependía del instructor informar a los estudiantes si podían proceder al siguiente ejercicio o si había un error en el ejercicio actual que necesitaba corrección. Esto normalmente solo tomaba unos segundos, por lo que la intervención tuvo poco impacto en el tiempo de trabajo de los estudiantes

Al discutir diferentes formas de programar Arduino, algunos estudiantes comentaron que los lenguajes basados en texto parecían más “serios/profesionales”, mientras que los lenguajes basados en bloques eran más “de juguete” (esta afirmación es consistente con los hallazgos en [17]). En respuesta a este comentario, se les mostró a los estudiantes un el editor de código del entorno de programación. Un estudiante se quejó de que el idioma estaba en inglés, lo cual es algo que a menudo se pasa por alto pero que representa un desafío adicional en el aprendizaje. Esta anécdota sirve para resaltar una ventaja de los lenguajes basados en bloques, ya que son fáciles de traducir a otros idiomas.

Finalmente, cuando se trató de ejercicios involucrando condicionales, el depurador de Physical Bits resultó ser muy útil para explicar la diferencia entre condicionales anidados y secuenciales. Ejecutar el programa paso a paso utilizando el depurador hizo que fuera muy fácil visualizar el flujo de ejecución. Esto solo fue posible en el grupo experimental, ya que el grupo de control tenía esta herramienta desactivada. Esta ventaja puede haber contribuido a la diferencia en el progreso de los estudiantes, como se verá a continuación.

VIII. RESULTADOS

Después de completar la actividad, se recopilaron los archivos de solución generados por los estudiantes y los archivos de log generados por la aplicación. A partir del procesamiento de estos archivos, se obtuvieron los siguientes conjuntos de datos:

- estudiantes: Información relacionada con cada uno de los estudiantes (19 registros)
- ejercicios: Información sobre la lista de ejercicios (70 registros)
- soluciones: Información sobre las soluciones a los ejercicios (491 registros)
- eventos: Información sobre los eventos registrados por la aplicación (332851 registros)

Dado el número limitado de participantes en el estudio y la duración de la actividad, los datos recopilados no son suficientes para obtener resultados estadísticamente significativos. Sin embargo, existen tendencias interesantes que vale la pena explorar en un estudio más grande.

RQ1 - Impacto en el progreso de los estudiantes

Para calcular el progreso de cada estudiante, se consideró el porcentaje de ejercicios resueltos sobre el total (70 ejercicios). Tanto para el grupo de control como para el grupo experimental, se observa un progreso similar, siendo el promedio más alto para el grupo experimental. Es notable también que el estudiante más avanzado en el grupo experimental logró resolver el 77.14% de los ejercicios, mientras que el estudiante más avanzado en el grupo de control solo alcanzó el 44.28%.

Dado que los ejercicios están ordenados siguiendo una curva de dificultad ascendente tiene sentido analizar el progreso de los alumnos encuentro por encuentro. De esta forma, se puede observar en la figura 1 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.

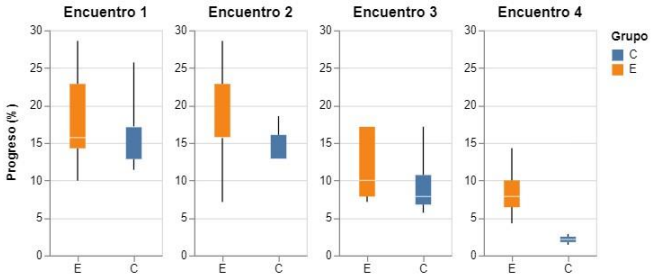


Fig. 1. Progreso de los estudiantes en cada encuentro

RQ2 - Impacto en el tiempo de resolución de cada ejercicio

Como puede observarse en la figura 2, no encontramos una diferencia significativa en el tiempo de resolución para los ejercicios iniciales, pero a medida que aumenta la dificultad, comienzan a surgir diferencias entre los grupos.

Los datos indican que, para los ejercicios más simples, no hay diferencias notables en el tiempo de resolución. Sin embargo, a medida que aumenta la complejidad de los ejercicios, también se observa que el grupo de control tarda más en resolverlos, lo que sugiere que los beneficios de la programación interactiva son más pronunciados a medida que aumenta la complejidad del problema a resolver.

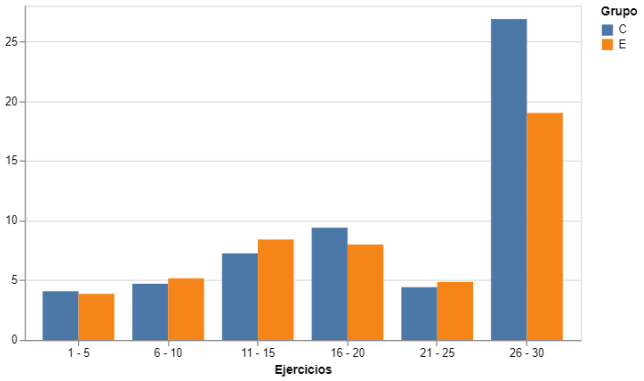


Fig. 2. Tiempo de resolución por ejercicio (mediana en minutos)

RQ3 - Impacto en la forma de trabajar de los estudiantes

Para analizar el trabajo de los estudiantes se utilizarán principalmente 2 métricas: el ritmo de trabajo y la eficiencia.

Physical Bits compila automáticamente el programa ante cada cambio, por lo que el tiempo entre compilaciones es una estimación razonable del ritmo de trabajo de cada estudiante. Como se observa en la figura 3, los estudiantes de ambos grupos trabajaron a un ritmo aproximadamente similar.

Por otro lado, la estimación de la eficiencia de cada estudiante se calculó contando cuántos ejercicios pudieron resolver por hora. En este caso, si observamos el resumen de todas las sesiones, vemos que, en promedio, la eficiencia fue muy similar entre ambos grupos, resultando en un rápido descenso a medida que avanzaban las sesiones. Coherente con los gráficos anteriores, esto tiene sentido considerando la creciente dificultad de los ejercicios.

Sin embargo, al analizar cada encuentro por separado, se puede observar que aunque el grupo de control muestra una eficiencia más alta en el primer encuentro, en las sesiones posteriores, su eficiencia promedio siempre está ligeramente por debajo de la del grupo experimental. La brecha aumenta, especialmente en el último encuentro, donde la eficiencia del grupo experimental triplicó la del grupo de control (figura 4).

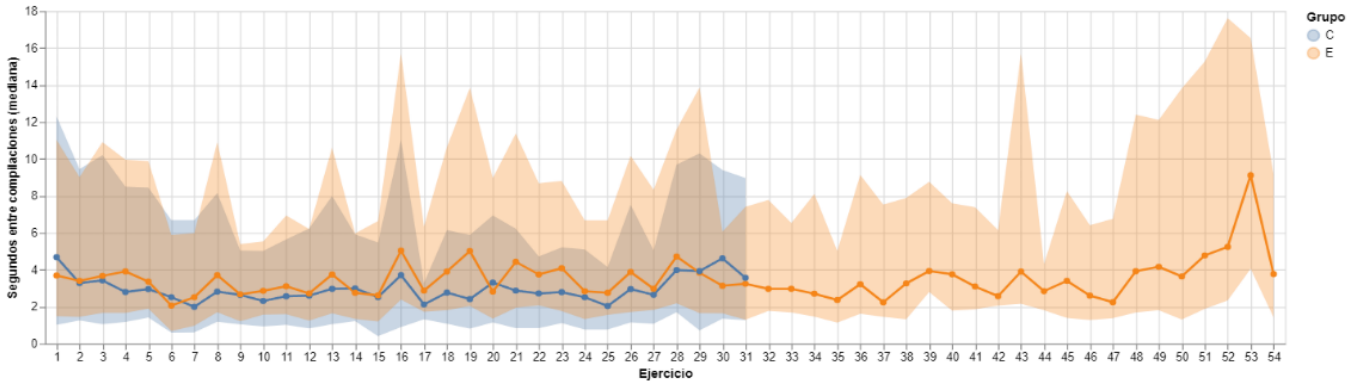


Fig. 3. Tiempo entre compilaciones por ejercicio (mediana en segundos)

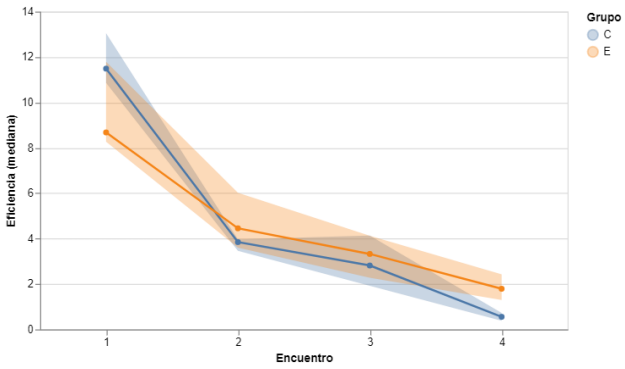


Fig. 4. Eficiencia de los estudiantes en cada encuentro

RQ4 - Impacto en el bucle de retroalimentación

La forma más sencilla de medir la duración del bucle de retroalimentación es considerar el tiempo desde que cada estudiante comienza a trabajar en un ejercicio hasta que puede ver el programa ejecutándose en el robot. En este sentido, se puede observar que el grupo experimental generalmente tardó mucho menos tiempo en ejecutar el programa por primera vez. Los datos indican que el tiempo hasta tener un programa en funcionamiento en el robot es, en promedio, un orden de magnitud menor para el grupo experimental que para el grupo de control (figura 5).

Otra forma de analizar la duración del bucle de retroalimentación es considerar las ejecuciones por minuto. En este caso, la diferencia entre el grupo experimental y el grupo de control también es evidente (figura 6).

Finalmente, también puede calcularse el tiempo entre ejecuciones. Al igual que en los gráficos anteriores, el grupo experimental exhibe ejecuciones mucho más frecuentes que el grupo de control (figura 7).

Estos datos sostienen la importancia de la programación interactiva en la resolución de problemas: un entorno interactivo implica menos obstáculos para probar programas, lo que a su vez fomenta ejecuciones más frecuentes y rápidas. En general, esto redunda en un mayor progreso en la resolución de ejercicios.

IX. Conclusiones

Este estudio piloto arroja luz sobre el impacto positivo de la programación interactiva en el rendimiento de los estudiantes en un taller de robótica. Los hallazgos enfatizan las ventajas de facilitar un feedback inmediato a las acciones de los estudiantes, lo que resulta en mejores oportunidades para la experimentación y un progreso más rápido en la resolución de problemas. Aunque nuestro estudio se centró en resultados de rendimiento en lugar de aprendizaje, sienta las bases para investigaciones futuras.

Es crucial reconocer, sin embargo, las limitaciones del estudio y la necesidad de una investigación más integral con un mayor número de participantes para establecer evidencia definitiva sobre las ventajas de la programación interactiva para actividades de robótica educativa.

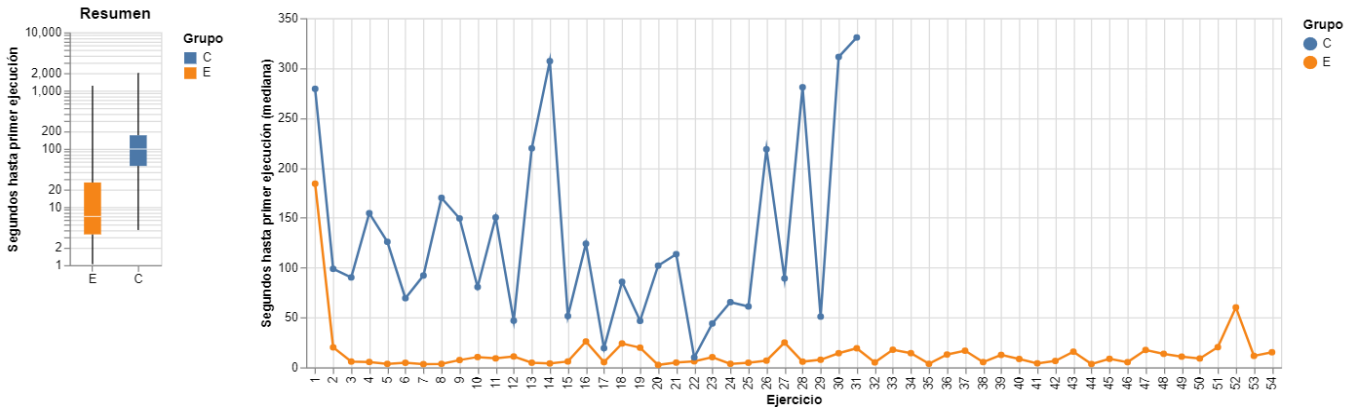


Fig. 5. Segundos hasta primer ejecución (izquierda: resumen de todos los encuentros; derecha: mediana por ejercicio)

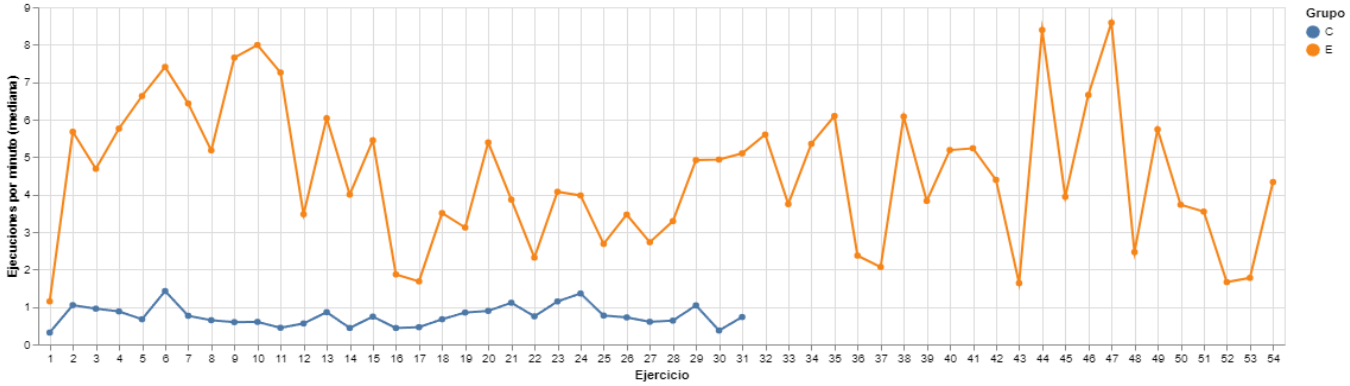


Fig. 6. Ejecuciones por minuto para cada ejercicio (mediana)

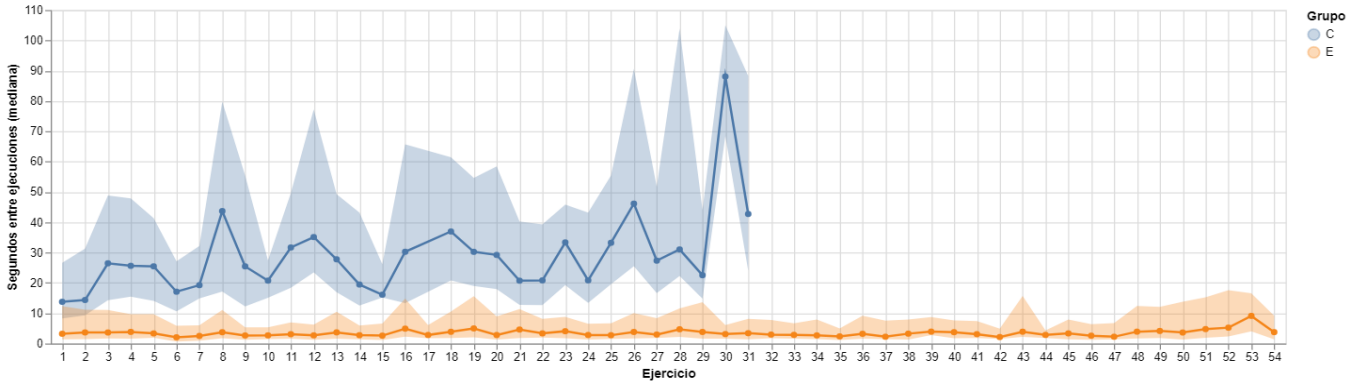


Fig. 7. Tiempo entre ejecuciones del programa para cada ejercicio (mediana en segundos)

REFERENCIAS

[1] P. Rein, S. Ramson, J. Lincke, R. Hirschfeld, and T. Pape, "Exploratory and Live, Programming and Coding: A Literature Study Comparing Perspectives on Liveness," *Programming*, vol. 3, no. 1, p. 1, Jul. 2018, doi: 10.22152/programming-journal.org/2019/3/1.

[2] S. Anwar, N. A. Bascou, M. Menekse, and A. Kardgar, "A Systematic Review of Studies on Educational Robotics," *Journal of Pre-College Engineering Education Research (J-PEER)*, vol. 9, no. 2, Jul. 2019, doi: 10.7771/2157-9288.1223.

[3] J. Devine, J. Finney, P. de Halleux, M. Moskal, T. Ball, and S. Hodges, "MakeCode and CODAL: Intuitive and efficient embedded systems programming for education," *Journal of Systems Architecture*, vol. 98, pp. 468–483, Sep. 2019, doi: 10.1016/j.sysarc.2019.05.005.

[4] "Electronics | Free Full-Text | HYDRA: Introducing a Low-Cost Framework for STEM Education Using Open Tools." Accessed: Jan. 30, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/10/24/3056

[5] P. Molins-Ruano, C. Gonzalez-Sacristan, and C. Garcia-Saura, "Phogo: A low cost, free and 'maker' revisit to Logo," *Computers in Human Behavior*, vol. 80, pp. 428–440, Mar. 2018, doi: 10.1016/j.chb.2017.09.029.

[6] A. Pina and I. Ciriza, "Primary Level Young Makers Programming & Making Electronics with Snap4Arduino," in *Educational Robotics in the Makers Era*, D. Alimisis, M. Moro, and E. Menegatti, Eds., in Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, 2017, pp. 20–33. doi: 10.1007/978-3-319-55553-9_2.

[7] P. Plaza *et al.*, "Scratch as Driver to Foster Interests for STEM and Educational Robotics," *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, vol. 14, no. 4, pp. 117–126, Nov. 2019, doi: 10.1109/RITA.2019.2950130.

[8] R. Moran, M. Teragni, and G. Zabala, "Physical Bits: A Live Programming Environment for Educational Robotics," in *Robotics in Education*, W. Lepuschitz, M. Merdan, G. Koppensteiner, R. Balogh, and D. Obdržálek, Eds., in Advances in Intelligent Systems and Computing. Cham: Springer International Publishing, 2021, pp. 291–303. doi: 10.1007/978-3-030-67411-3_26.

[9] J.-P. Kramer, J. Kurz, T. Karrer, and J. Borchers, "How live coding affects developers' coding behavior," in *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Jul. 2014, pp. 5–8. doi: 10.1109/VLHCC.2014.6883013.

[10] C. D. Hundhausen and J. L. Brown, "An experimental study of the impact of visual semantic feedback on novice programming," *Journal of Visual Languages & Computing*, vol. 18, no. 6, pp. 537–559, Dec. 2007, doi: 10.1016/j.jvlc.2006.09.001.

[11] E. M. Wilcox, J. W. Atwood, M. M. Burnett, J. J. Cadiz, and C. R. Cook, "Does continuous visual feedback aid debugging in direct-manipulation programming systems?," in *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, in CHI '97. New York, NY, USA: Association for Computing Machinery, Mar. 1997, pp. 258–265. doi: 10.1145/258549.258721.

[12] M. Lodi, D. Malchiodi, M. Monga, A. Morpurgo, and B. Spieler, "Constructionist Attempts at Supporting the Learning of Computer Programming: A Survey," *Olympiads in Informatics*, vol. 13, pp. 99–121, Jul. 2019, doi: 10.15388/ioi.2019.07.

[13] D. Weintrop and U. Wilensky, "To block or not to block, that is the question: students' perceptions of blocks-based programming," in *Proceedings of the 14th International Conference on Interaction Design and Children*, Boston Massachusetts: ACM, Jun. 2015, pp. 199–208. doi: 10.1145/2771839.2771860.

[14] L. Cabrera, J. H. Maloney, and D. Weintrop, "Programs in the Palm of your Hand: How Live Programming Shapes Children's Interactions with Physical Computing Devices," in *Proceedings of the 18th ACM International Conference on Interaction Design and Children*, in IDC '19. New York, NY, USA: Association for Computing Machinery, Jun. 2019, pp. 227–236. doi: 10.1145/3311927.3323138.

[15] R. Moran, "RichoM/pbits-exercises." Jan. 05, 2023. Accessed: Apr. 04, 2024. [Online]. Available: https://github.com/RichoM/pbits-exercises

[16] P. Brusilovsky, "Teaching Programming to Novices: A Review of Approaches and Tools.," 1994. Accessed: Dec. 16, 2022. [Online]. Available: https://www.semanticscholar.org/paper/Teaching-Programming-to-Novices%3A-A-Review-of-and-Brusilovsky/bf44088aaa3b47fe2081a9641fe34a8d715dcfac

[17] J. Rodríguez, G. Parra, D. Dolz, and R. Ramírez, "Transición desde programación basada en bloques a basada en texto: una revisión del campo," 2019. Accessed: Jul. 22, 2022. [Online]. Available: http://sedici.unlp.edu.ar/handle/10915/91318