UNIVERSIDAD REY JUAN CARLOS

TRABAJO FIN DE MÁSTER

Enseñanza de la Programación y el Pensamiento Computacional con la Plataforma Kibotics

MÁSTER UNIVERSITARIO EN FORMACIÓN DEL PROFESORADO DE EDUCACIÓN SECUNDARIA OBLIGATORIA Y BACHILLERATO, FORMACIÓN PROFESIONAL Y ENSEÑANZA DE IDIOMAS.

INFORMÁTICA Y TECNOLOGÍA

2021-2022

Apellidos y Nombre del Alumno/a

Luis Castro San Martín

Director/a TFM:

Raquel Belén Hijón Neira

Agradecimientos

En primer lugar, mi más sincero agradecimiento a la dirección, profesores y alumnos del instituto en que se llevó a cabo esta intervención educativa, pues sin su colaboración este proyecto nunca hubiera sido posible. Especial agradecimiento a Mª José, Miguel Ángel y Gloria por todo su apoyo y colaboración.

En segundo lugar, agradecer al equipo de Kibotics su soporte y disponibilidad durante el desarrollo de las sesiones prácticas. Especial agradecimiento a José María y Celeste.

En tercer lugar, agradecer a Raquel Belén Hijón Neira, la directora del trabajo de fin de master, su confianza, apoyo y recursos facilitados durante el desarrollo del proyecto.

Por último, agradecer a mi familia el apoyo mantenido durante este proyecto, que transciende el presente estudio, y que ha supuesto muchas horas de trabajo que no he podido dedicarles.

Tabla de contenido

Índice de Figuras

Índice de Tablas

Resumen

En este trabajo de fin de máster se presenta una intervención educativa en alumnos de secundaria para la enseñanza de la programación y del pensamiento computacional basada en el uso de la plataforma educativa Kibotics.

Kibotics[1] es una plataforma web que permite aprender programación y robótica educativa, dirigida principalmente a la comunidad educativa de los ciclos de primaria y secundaria. En ella los alumnos aprenden conceptos STEM (Science, Technology, Engineering and Mathematics), desde un enfoque eminentemente práctico, a través de la resolución de ejercicios de programación de algoritmos de robótica con el lenguaje de bloques Scratch o Python que ejecutan robots virtuales en diferentes escenarios. Los ejercicios vienen acompañados de contenido audiovisual, atractivo y adaptado al público objetivo, sobre los conceptos STEM objetivo del aprendizaje y necesarios para la resolución del ejercicio.

El pensamiento computacional (PC) es una habilidad humana que ha ido ganando relevancia en los últimos 15 años para investigadores, profesionales y políticos del ámbito educativo. El PC se puede definir como la habilidad de resolver problemas, diseñar sistemas y comprender conceptos informáticos basados en el comportamiento humano. También puede definirse como los procesos de pensamiento involucrados en la formulación de problemas para que sus soluciones puedan representarse como pasos computacionales y algoritmos.

El objetivo del estudio es evaluar la influencia del empleo de la herramienta Kibotics en el aprendizaje de conceptos de programación y pensamiento computacional tales como direcciones, bucles, condicionales y funciones.

El estudio emplea el método de investigación experimental con un enfoque cuantitativo cuya variable dependiente es el aprendizaje de los estudiantes. El contexto del desarrollo del estudio experimental (diseño cuasi-experimental pretest-posttest con grupo control) son alumnos de 2º curso de ESO de la asignatura "Tecnología,

[1] www.kibotics.org

Programación y Robótica", realizando un muestreo no probabilístico de tipo casual en un Instituto de Enseñanza Secundaria.

Tras la intervención, con talleres de programación de algoritmos de robótica, de tres semanas de duración en total, los alumnos que han usado Kibotics obtienen mejoras significativas en los test de pensamiento computacional respecto a los mismos test realizados antes de las actividades. En cambio, el grupo de control, que ha utilizado otra plataforma de aprendizaje en las actividades, obtiene mejoras en sus puntuaciones, pero no fueron significativas.

El análisis de los resultados obtenidos permite concluir que el empleo de la herramienta Kibotics, en la que se programan robots usando un lenguaje de bloques, ha mejorado significativamente el aprendizaje de conceptos de programación y pensamiento computacional frente a la otra plataforma, que también emplea lenguaje por bloques pero no emplea robots.

Palabras clave: Educación, Enseñanza Secundaria, Algoritmos, Robótica, Scratch.

Abstract

In this master's thesis, an educational intervention in secondary school students for the teaching of programming and computational thinking based on the use of the Kibotics educational platform is presented.

Kibotics[2] is a web platform that allows learning programming and educational robotics, aimed mainly at the educational community of primary and secondary cycles. In Kibotics, students learn STEM concepts (Science, Technology, Engineering and Mathematics), from an eminently practical approach, by solving robotics algorithm programming exercises with the Scratch block language or Python that execute virtual robots in different scenarios. The exercises are accompanied by media content, attractive and adapted to the target audience, on the STEM concepts that are the objective of learning and necessary to solve the exercise.

Computational thinking (CT) is a human skill that has been gaining relevance in the last 15 years for researchers, professionals and politicians in the educational field. CT can be defined as the ability to solve problems, design systems, and understand computing concepts based on human behaviour. It can also be defined as the thought processes involved in formulating problems so that their solutions can be represented as computational steps and algorithms.

The objective of the study is to evaluate the influence of the use of the Kibotics tool in the learning of programming concepts and computational thinking such as directions, loops, conditionals and functions.

The study uses the experimental research method with a quantitative approach whose dependent variable is student learning. The context of the development of the experimental study (pretest-posttest quasi-experimental design with control group) are students of 2nd year of ESO of the subject "Technology, Programming and Robotics", carrying out a non-probabilistic random sampling in a secondary school.

After the intervention, with workshops on programming robotics algorithms, lasting three weeks in total, the students who have used Kibotics obtain significant

[2] www.kibotics.org

improvements in the computational thinking tests compared to the same tests carried out before the activities. On the other hand, the control group, which has used another learning platform in the activities, obtains improvements in their scores, but they were not significant.

The analysis of the results obtained allows us to conclude that the use of the Kibotics tool, in which robots are programmed using a block language, has significantly improved the learning of programming concepts and computational thinking compared to the other platform, which also uses a block language but does not use robots.

Keywords: Education, Secondary School, Algorithms, Robotics, Scratch.

1 Introducción

1.1 Justificación

El pensamiento computacional (PC en adelante) es una habilidad humana que ha ido ganando relevancia en los últimos 15 años para investigadores, profesionales y políticos del ámbito educativo (Bocconi et al. 2016; Caballero-González y García-Valcárcel, 2020).

El PC se puede definir como la habilidad de resolver problemas, diseñar sistemas y comprender conceptos informáticos basados en el comportamiento humano (Wing, 2006). También puede definirse como los procesos de pensamiento involucrados en la formulación de problemas para que sus soluciones puedan representarse como pasos computacionales y algoritmos (Aho, 2012). El PC es una habilidad clave para los niños en el siglo XXI (Wing, 2011, 2016). Sin embargo, no está claro cómo se puede desarrollar el PC de la manera más efectiva en los niños. Actualmente, se están investigando diferentes metodologías pedagógicas que se pueden utilizar para desarrollarlo.

En los últimos años, algunos autores han afirmado que el PC se puede adquirir y desarrollar mediante la enseñanza de la programación a los niños. Además, se ha afirmado que esto debe hacerse lo antes posible (Heintz et al., 2016; Kazakoff et al., 2013; Papadakis et al., 2016; Strawhacker et al., 2015).

Aprender a programar puede inducir cambios en la forma en que las personas piensan (Papert, 1980; Resnick, 1996). Esto es probablemente debido a la componente analítica del PC, que es bastante similar al pensamiento matemático (es decir, resolución de problemas), pensamiento de ingeniería (diseño y evaluación de procesos) y el pensamiento científico (análisis sistemático).

Es posible que el PC se pueda adquirir por otros medios como robótica educativa (Bers, 2010), storytelling (Lee et al., 2011), actividades de computación desconectadas (Brackmann et al., 2016; Montes-León et al., 2020), Scratch Jr (Papadakis et al., 2016) o incluso en lecciones de ética (Seoane-Pardo, 2018).

Un enfoque común para la enseñanza de la programación para los niños es Scratch, desarrollado por el grupo de investigación Lifelong Kindergarten del MIT Media Lab. para diseñar programas interactivos uniendo instrucciones de programación por

bloques (Resnick et al., 2009; Ouahbi et al., 2015). Otros enfoques usan robots, como los Lego WeDo o Mindstorms EV3 (Sovic et al., 2014), o generan juegos (Denner et al., 2019).

García-Valcárcel y Caballero-González (2019) concluyen en su estudio con estudiantes de educación infantil, de entre 3 y 6 años de edad, que es posible desarrollar habilidades de PC a tan tempranas edades y que el uso de actividades de robótica tiene un impacto positivo significativo en el aprendizaje.

Witherspoon et al. (2017), tras sus dos estudios con estudiantes estadounidenses de primaria y secundaria, afirmaron que la participación en un plan de estudios dirigido de programación, en el que se programan robots virtuales, desarrolla los conocimientos y habilidades de PC.

La presente investigación se centra en la programación de robots en la plataforma Kibotics utilizando Scratch para fomentar el PC. Vale la pena aprender a programar por la necesidad real de programadores en nuestra sociedad digital (Margulieux et al., 2016), así como otras ventajas como la mejora de habilidades cognitivas superiores (Pea y Kurland, 1984).

En el estudio se va a evaluar la influencia del empleo de la herramienta Kibotics en el aprendizaje de conceptos de programación y pensamiento computacional tales como direcciones, bucles, condicionales y funciones.

1.2 Estructura del TFM

Este trabajo de Fin de Máster es un trabajo experimental, para estructurarlo se toma como referencia el informe final de investigación (García Lázaro et al., 2018), explicando la intervención educativa dentro del apartado “Metodología o método”:

- Título.
- Resumen y abstract.
- Palabras claves.
- Introducción o justificación.
- Metodología o método.
- Resultados.
- Conclusiones.
- Referencias bibliográficas.
- Anexos.

2 Metodología

Se explica en este apartado la metodología seguida en cada fase del estudio (Figura 1), según las previstas para la investigación educativa (García et al, 2018) más la intervención educativa.

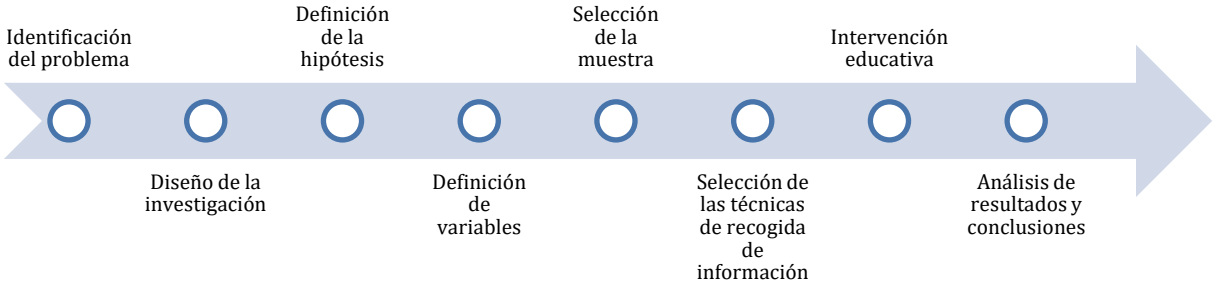


Figura 1. *Fases del estudio. Fuente: Elaboración propia*

2.1 Identificación del problema

Dada la relevancia del aprendizaje del PC y de su presencia en el currículo de educación secundaria de la Comunidad de Madrid (Consejería de Presidencia de la Comunidad de Madrid, 2015), los investigadores y profesionales de la educación necesitan herramientas eficaces para la enseñanza de esta habilidad en las aulas. Por lo tanto, existe un problema si los docentes no tienen evidencias científicas que avalen la eficacia de herramientas de aprendizaje que emplean en sus aulas para enseñar a sus estudiantes.

2.2 Diseño de la investigación

La investigación emplea el método de investigación experimental con un enfoque cuantitativo cuya variable dependiente es el aprendizaje de los estudiantes y se ha justificado en el apartado “1.1 Justificación” de este mismo documento.

El contexto del desarrollo del estudio experimental (diseño cuasiexperimental pretest-posttest con grupo control) son alumnos de 2º curso de ESO de la asignatura "Tecnología, Programación y Robótica", realizando un muestreo no probabilístico de tipo casual con una muestra de 85 estudiantes de un instituto de enseñanza secundaria.

2.3 Definición de la hipótesis u objetivos

Partiendo de que es posible aprender pensamiento computacional por medios distintos a la programación como la Robótica Educativa (Bers, 2010), el objetivo del estudio es evaluar la influencia del empleo de la herramienta Kibotics en el aprendizaje de conceptos de programación y pensamiento computacional tales como direcciones, bucles, condicionales y funciones, en comparación con otras herramientas de aprendizaje que no usan robótica, con contexto de ejecución en dos dimensiones, en lugar de tres, y cuyos personajes no requieren de programación adicional para adaptarse a un entorno con sus propias leyes físicas simuladas, como el rozamiento, u obstáculos e interacción con otros objetos en movimiento no previstos.

2.4 Definición de variables

La variable objeto de la investigación, o dependiente, es el aprendizaje, que se mide a partir de la variable independiente cuantitativa y discreta número de aciertos del cuestionario de pensamiento computacional validado, con valores comprendidos entre 0 y 28 (Román-González et al., 2017).

2.5 Selección de la muestra

En lo que respecta la selección de la muestra, se trata de un grupo de 85 estudiantes de 2º curso de la ESO al que se ha tenido acceso de forma casual, cuya asignatura "Tecnología, Programación y Robótica" contempla unidades didácticas sobre programación, pensamiento computacional y robótica, lo que hacía de ellos candidatos ideales para participar en las actividades previstas en el estudio, directamente relacionadas con contenidos y objetivos de su asignatura.

Por lo tanto, se realiza un muestreo no probabilístico de tipo casual, por facilidad de acceso, con una muestra de 85 participantes de un instituto de educación secundaria (I.E.S.) de la Comunidad de Madrid.

La muestra se divide en dos grupos de individuos, uno de control formado por 37 estudiantes, y otro experimental de 48. Los grupos se forman por grupos de clases completos asignados al azar, dos clases al grupo de control y otras dos al experimental. El grupo de control usa para completar las actividades previstas en la intervención la

plataforma web de Scratch[3], y el experimental utiliza la plataforma web de robótica educativa Kibotics[4]. Tanto Scratch como Kibotics son plataformas on-line, todos los alumnos disponen de un usuario de acceso a la plataforma del grupo que les ha correspondido y deben realizar las actividades de forma individual.

No se ha considerado imprescindible conocer la edad exacta de los estudiantes porque se trata de medir las capacidades de pensamiento computacional en alumnos cursando 2º de la ESO, no en edades concretas.

Además, no se realiza ningún tipo de selección entre el alumnado para su participación. Se ofrece la posibilidad de participación en el estudio a todos los alumnos de 2º curso de ESO del instituto.

2.6 Selección de técnicas de recogida de información

Elegida la muestra, se elige la técnica de recogida de información más adecuada, que en este caso será un cuestionario validado para medir el PC de los estudiantes.

2.6.1 Elección del test de pensamiento computacional

Se elige el cuestionario validado "Test de Pensamiento Computacional' (Román-González et al., 2017) para la recogida de información. Se realizará un test antes del comienzo de las actividades (pretest) y después de la finalización de las actividades (posttest).

El "Test de Pensamiento computacional" tiene las siguientes características:

- Objetivo: medir el nivel de desarrollo del PC en el sujeto.
- Definición operativa del constructo medido: el PC es la capacidad de formular y solucionar problemas apoyándose en los conceptos fundamentales de la computación, y usando la lógica inherente a los lenguajes informáticos de programación: secuencias o direcciones básicas, bucles, condicionales, funciones, y variables.

[3] https://scratch.mit.edu/
[4] https://kibotics.org

- Población objetivo: población escolar española de 1º y 2º de ESO (12 y 13 años)
- Tipo de instrumento: prueba objetiva de elección múltiple con 4 opciones de respuesta y con solo una correcta.
- Tiempo máximo de realización: 45 minutos.

Cada una de las 28 preguntas o ítems del test se han diseñado y caracterizado dentro de cinco ejes:

- Concepto computacional abordado: cada pregunta aborda uno o más de los 7 conceptos de computación, ordenados por dificultad creciente: Direcciones, Bucles-repetir veces, Bucles-repetir hasta, Condicional simple (if), Condicional compuesto (if-else), Mientras que (While) y Funciones.
- Entorno-Interfaz del ítem: el contexto de la pregunta es un laberinto o un lienzo.
- Estilo de las alternativas de respuesta: las respuestas, siempre en formato visual, se presentan en forma de flechas o de bloques.
- Existencia o inexistencia de anidamiento: las preguntas pueden contener conceptos computacionales anidados o no.
- Tarea requerida: la tarea cognitiva requerida puede ser de tres tipos:
    - Secuenciación: enunciar de forma ordenada una serie de comandos u órdenes.
    - Completamiento: completar un conjunto incompleto de comandos facilitado previamente.
    - Depuración: encontrar el error en un conjunto de comandos incorrecto previamente dado.

En las 28 preguntas del test se van planteando cuestiones combinando los cinco ejes descritos según recoge la Figura 2.

	Entorno - Interfaz del reactivo	Estilo de las alternativas de respuesta	Concepto computacional abordado								Existencia de anidamiento	Tarea requerida
			Direcciones	Bucles (loops)		Condicionales (conditionals)			Funciones (functions)			
				Repetir veces (repeat times)	Repetir hasta (repeat until)	Condicional simple (if)	Condicional compuesto (if/else)	Mientras que (while)	Funciones simples	Funciones con parámetros		
Item 1	Laberinto	Visual por flechas	Sí	No	No	No	No	No	No	No	No	Secuenciación
Item 2	Laberinto	Visual por flechas	Sí	No	No	No	No	No	No	No	No	Completamiento
Item 3	Laberinto	Visual por bloques	Sí	No	No	No	No	No	No	No	No	Depuración
Item 4	Lienzo	Visual por bloques	Sí	No	No	No	No	No	No	No	No	Secuenciación
Item 5	Laberinto	Visual por flechas	Sí	Sí	No	No	No	No	No	No	No	Secuenciación
Item 6	Laberinto	Visual por flechas	Sí	Sí	No	No	No	No	No	No	No	Completamiento
Item 7	Lienzo	Visual por bloques	Sí	Sí	No	No	No	No	No	No	No	Depuración
Item 8	Laberinto	Visual por bloques	Sí	Sí	No	No	No	No	No	No	Sí	Secuenciación
Item 9	Laberinto	Visual por flechas	Sí	No	Sí	No	No	No	No	No	No	Secuenciación
Item 10	Laberinto	Visual por bloques	Sí	No	Sí	No	No	No	No	No	No	Completamiento
Item 11	Laberinto	Visual por flechas	Sí	Sí	Sí	No	No	No	No	No	Sí	Depuración
Item 12	Lienzo	Visual por bloques	Sí	Sí	Sí	No	No	No	No	No	Sí	Secuenciación
Item 13	Laberinto	Visual por flechas	Sí	No	Sí	Sí	No	No	No	No	Sí	Secuenciación
Item 14	Laberinto	Visual por bloques	Sí	No	Sí	Sí	No	No	No	No	Sí	Secuenciación
Item 15	Laberinto	Visual por flechas	Sí	Sí	Sí	Sí	No	No	No	No	Sí	Completamiento
Item 16	Laberinto	Visual por bloques	Sí	No	Sí	Sí	No	No	No	No	Sí	Depuración
Item 17	Laberinto	Visual por bloques	Sí	No	Sí	No	Sí	No	No	No	Sí	Secuenciación
Item 18	Laberinto	Visual por bloques	Sí	No	Sí	No	Sí	No	No	No	Sí	Secuenciación
Item 19	Laberinto	Visual por bloques	Sí	No	Sí	No	Sí	No	No	No	Sí	Depuración
Item 20	Laberinto	Visual por bloques	Sí	No	Sí	No	Sí	No	No	No	Sí	Completamiento
Item 21	Laberinto	Visual por bloques	Sí	Sí	No	No	No	Sí	No	No	Sí	Secuenciación
Item 22	Laberinto	Visual por bloques	Sí	Sí	No	No	No	Sí	No	No	Sí	Secuenciación
Item 23	Laberinto	Visual por bloques	Sí	No	No	Sí	No	Sí	No	No	Sí	Completamiento
Item 24	Laberinto	Visual por bloques	Sí	No	No	Sí	No	Sí	No	No	Sí	Completamiento
Item 25	Lienzo	Visual por bloques	Sí	Sí	No	No	No	No	Sí	No	Sí	Secuenciación
Item 26	Lienzo	Visual por bloques	Sí	Sí	No	No	No	No	Sí	No	Sí	Completamiento
Item 27	Laberinto	Visual por bloques	Sí	Sí	No	No	No	No	Sí	No	Sí	Secuenciación
Item 28	Laberinto	Visual por bloques	Sí	Sí	No	No	No	No	Sí	No	Sí	Completamiento

***Figura 2**. Cuadro resumen de especificaciones de los 28 items del TPC (versión 2) de los 5 ejes de diseño. Fuente: Román-González (2016)*

En el anexo "6.10 Test de Pensamiento Computacional" se ha recogido una selección de preguntas, ítems de la Figura 2, que recogen las distintas tipologías utilizadas en el test. A modo ejemplo, se muestra en la Figura 3 una pregunta en el entorno de interfaz Laberinto, con estilo de alternativas de respuesta de tipo Visual por Flechas, concepto computacional Direcciones, sin anidamiento de conceptos y con tarea cognitiva requerida tipo Secuenciación.

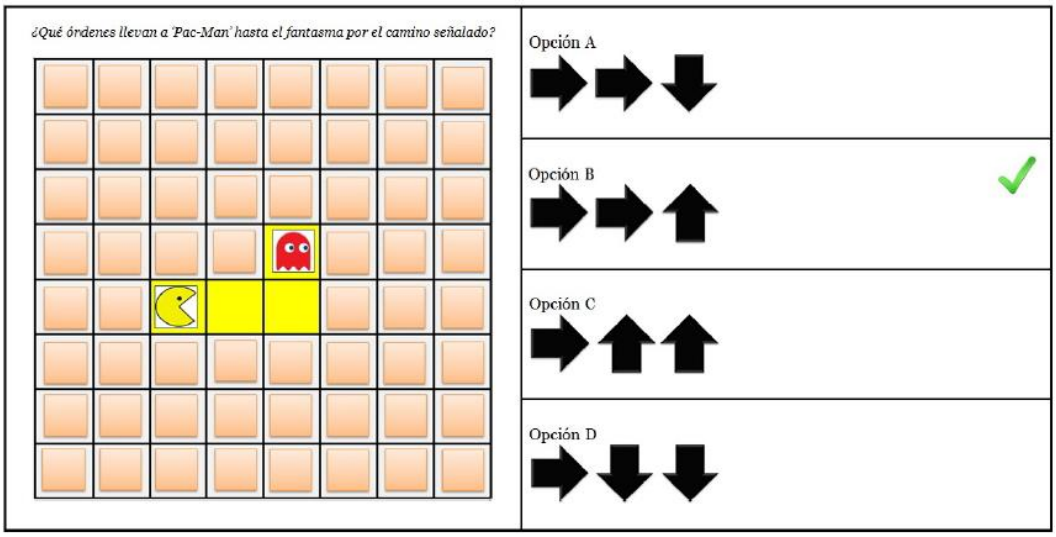


***Figura 3**. Cuestionario. Pregunta direcciones en laberinto con flechas (Secuenciación). Fuente: Román-González (2016)*

2.7 Intervención educativa

Es el momento en el que se lleva a cabo la acción educativa de los estudiantes. Durante esta fase, se realizan las actividades formativas y la recogida de información que permite después analizar los resultados de la intervención educativa (Figura 4).

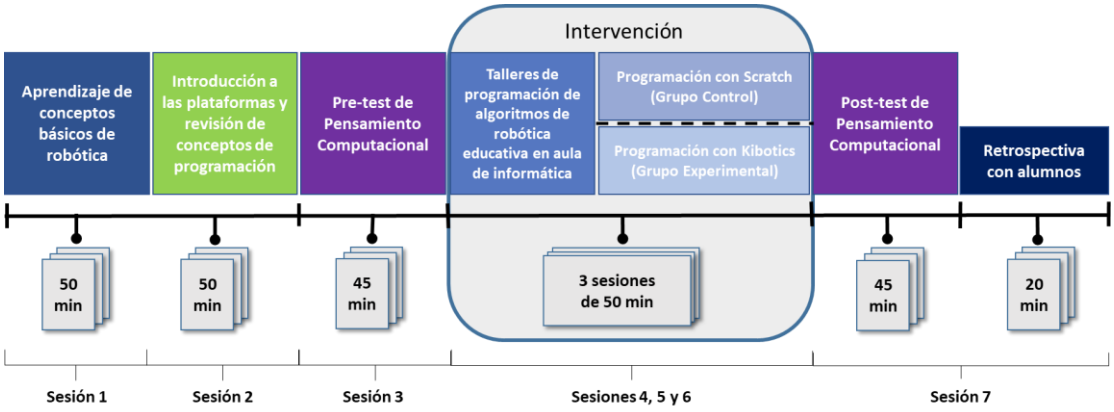


Figura 4. *Esquema de la intervención educativa. Fuente: Elaboración propia.*

Se celebran 7 sesiones en la que los alumnos aprenden conceptos de robótica y programación en las plataformas Kibotics y Scratch.

Las dos primeras sesiones son preparatorias para las actividades de programación (talleres). En la primera se introduce a los alumnos a la robótica educativa y, en la segunda, se explica cómo se programan los robots y se presentan las plataformas que utilizarán durante los talleres de programación.

En la tercera sesión se realiza el primer test (Pretest) de PC. Y en las sesiones cuarta, quinta y sexta se realizan las actividades de programación, cada grupo con la plataforma correspondiente según su pertenencia al grupo de control (Scratch) o experimental (Kibotics).

En la séptima sesión, la última, se repite el test de PC (Post-test). Tras la finalización del test, se invita a los alumnos a trasladar sus comentarios (retrospectiva) sobre las distintas plataformas.

2.7.1 Preparación

Para poder llevar al aula la intervención es necesario realizar previamente una serie de trámites y actividades preparatorias. Se han de obtener los permisos para la intervención, conseguir los recursos necesarios, así como concretar las actividades y las herramientas para la obtención de la información (Figura 5).

***Figura 5**. Preparación de la intervención. Fuente: Elaboración propia*

2.7.1.1 Obtención de la evaluación favorable del Comité de Ética

Para obtener la aprobación del Comité de Ética de Investigación la Universidad Rey Juan Carlos, en primer lugar, se elabora una memoria con el diseño de la investigación y la justificación de la intervención. Después, se solicita la colaboración y autorización para participar en el estudio al centro de estudios. En tercer lugar, se diseña el formulario en el que se recogerá la información y se elaboran las hojas informativas y de consentimiento informado. Por último, se procede a la solicitud de estudio al comité de ética y se queda a la espera de la respuesta. Se realizan las subsanaciones solicitadas y finalmente se obtiene la evaluación favorable para la intervención con número de registro interno 1901202203322.

2.7.1.2 Organización con el centro educativo

Dado que la intervención se realizará dentro de un instituto, es necesario coordinarse con los profesores de los estudiantes para preparar todo lo necesario para llevar a cabo la intervención.

- Coordinación y temporalización con los profesores de la asignatura para identificar las sesiones disponibles para la celebración de las actividades.

Se habla con los profesores de los alumnos para evaluar las posibilidades de participación de los estudiantes sin afectar a la programación de la asignatura. Determinan que es posible dedicar siete sesiones de clase a la actividad. Con su ayuda, se identifican las clases en las que celebrar las sesiones de los cuatro grupos de alumnos de 2º que participarán en las actividades. Los grupos tienen diferentes horarios de clase de la asignatura "Tecnología, Programación y Robótica" en la que se celebraran las sesiones, lo que permite realizarlas en su clases habituales. Se van tomando secuencialmente las dos sesiones por semana de clase de 50 minutos, de cada grupo, hasta completar las siete sesiones previstas, distribuidas en un total de cuatro semanas.

- Entrega y recogida de hojas informativas y consentimientos informados para obtener los consentimientos de participación de los estudiantes que así lo deseen en el estudio.
- Preparación de los recursos materiales para la celebración de las sesiones.
    - Reserva de aulas de informática y dotación con equipos suficientes.

    Se consiguen las mesas, sillas y los equipos que faltaban en el aula de ordenadores facilitada por el centro para la celebración de las sesiones, para que todos los alumnos puedan hacer las actividades previstas de forma individual.

    Se realiza la prueba de ejecución de la plataforma Kibotics en todos los equipos del aula de informática como sugerencia del equipo de soporte de la plataforma, dado que la aplicación utiliza los recursos de memoria y procesador para su ejecución desde el navegador.

    - Facilitar el acceso a las plataformas de programación a los alumnos

    En las primeras sesiones se reservan los últimos 15 minutos de clase para registrar a todos los alumnos en las plataformas Kibotics y Scratch.

    Se contacta con el equipo de soporte de Kibotics para gestionar las licencias de acceso al curso "Aprende robótica con Scratch", elegido para las actividades, y las incidencias en los registros.

2.7.1.3 Concreción de las actividades

Cuando se conoce el número de sesiones disponibles se eligen las actividades de programación más apropiadas, para los conocimientos previos de los alumnos y para el tiempo que podrá dedicarse a las mismas. Aunque se van a dedicar siete sesiones en total, las dos primeras serán de conceptos teóricos y otras dos se dedicarán a realizar los test, por lo que realmente serán tres las dedicadas a los talleres de programación. En este apartado se introducen las actividades que se llevarán a cabo en los talleres de programación.

- Selección de las actividades de la plataforma Kibotics.

  Las actividades elegidas son las que se enumeran a continuación:

  1. Cuadrado con GoPiGo en Scratch: actividad de introducción en la plataforma con la menor complejidad, tanto desde el punto de vista computacional como de robótica. Consiste en programar el robot para que pase por las esquinas que forman un cuadrado.
  2. Choca gira ultrasonidos con GoPiGo en Scratch: actividad que introduce conceptos computacionales más avanzados y el uso de un sensor del robot, pero con un algoritmo que no es complejo. Consiste en programar el robot para que haciendo uso de la lectura de un sensor se mueva libremente por una habitación sorteando los obstáculos (paredes, muebles, etc.).
  3. Siguelíneas infrarrojos con GoPiGo en Scratch: actividad que incorpora nuevos conceptos computacionales respecto a las anteriores, a la vez que aumenta de dificultad el algoritmo. Consiste en programar un robot para que recorra un circuito usando un sensor que permite detectar el camino.
  4. Recoge confeti con Roomba en Scratch: actividad similar a la segunda, pero que requiere el desarrollo de un algoritmo más avanzado. Consiste en programar un robot aspirador para que además de evitar obstáculos, recorra la mayor superficie posible para recoger el mayor número de confeti.

- Diseño de las actividades de Scratch, equivalentes a las elegidas de la plataforma Kibotics, para el grupo de control.

  Elegidas las actividades de la plataforma Kibotics, se diseñan las actividades para las sesiones de trabajo del grupo de control con Scratch, que requieren la programación de algoritmos equivalentes y el uso de los mismos conceptos computacionales. Son las siguientes:

  1. Cuadrado con Scratch: actividad que consiste en programar los movimientos de un personaje con apariencia de robot para que pase por las cuatro esquinas de un escenario formando un cuadrado. Para simular el uso de los actuadores de un robot se deben emplear las instrucciones de movimiento y giro, no las de desplazamiento por coordenadas x-y que también ofrece Scratch.
  2. Chocagira con Scratch: actividad que consiste en programar el movimiento de un personaje con apariencia de robot para que recorra un escenario "sin chocarse" con los obstáculos simulados del escenario (paredes). Para detectar las paredes del escenario se usa el sensor de colores de Scratch.
  3. Siguelíneas con Scratch: actividad que consiste en programar los movimientos de un personaje con apariencia de robot para que recorra un circuito, el escenario, sin salirse. Para detectar si se sale del circuito se ha de usar el sensor de colores junto con dos elementos incorporados en el personaje facilitado, que se explican en la actividad.
  4. Recoge confeti con Scratch: actividad que consiste en mejorar el movimiento del personaje utilizado en el ejercicio 2 para que recorra la mayor superficie posible del escenario y recoja el mayor número posible de confeti.

El detalle de las actividades anteriores se explicará en el apartado "2.7.5 Actividades prácticas de programación". Además, en los anexos "6.2 Cuadrado con Scratch", "6.3 Cuadrado con Kibotics", "6.4 Chocagira con Scratch", "6.5 Chocagira con

Kibotics", "6.6 Siguelíneas con Scratch", "6.7 Siguelíneas con Kibotics", "6.8 Recoge confeti con Scratch" y "6.9 Recoge confeti con Kibotics" pueden consultarse los enunciados completos de las actividades.

2.7.1.4 Preparación del cuestionario on-line

Para la confección del cuestionario se opta por la herramienta web Google Forms, accesible a través de Internet, que facilita la distribución del cuestionario, su autocorrección y el recopilado y tratamiento de la información.

El cuestionario consta de un apartado al inicio en el que el estudiante tiene que introducir su código unívoco de participación. Después, se presentan las instrucciones y se muestran ejemplos de los tipos de preguntas y respuestas, con el objetivo de que los participantes tengan claro la dinámica de las preguntas.

Tras las instrucciones, empieza el test de PC, formado por 28 preguntas con cuatro opciones de respuesta y una sola correcta. Cada pregunta acertada suma 1 punto a la puntuación total del test, no restando sobre la puntuación total las no acertadas o no contestadas. Después de las 28 preguntas, se incluyen 2 de autoevaluación, que no puntúan en el resultado del test de PC, en las que se pregunta al encuestado cómo considera que le ha salido el test y cómo se le dan los ordenadores y la informática, las cuales, son contestadas en una escala Likert de 11 puntos (de 0 a 10), que permite mayor sensibilidad que la habitualmente utilizada de 5 puntos (Bisquerra, R. y Pérez-Escoda, N., 2015).

En el anexo "6.10 Test de Pensamiento Computacional" se recoge extracto del formulario de Google Forms en el que se pueden ver los contenidos expuestos en este apartado. También puede consultarse este mismo extracto en formato electrónico desde el enlace la Figura 6.

***Figura 6**. Código QR de enlace al extracto del cuestionario utilizado. Fuente: Adaptado de Román-González (2016)*

2.7.2 Contenidos

Las sesiones se han programado dentro de la unidad didáctica "Robótica" de la programación de la asignatura "Tecnología, programación y robótica", que recoge los contenidos del currículo de educación secundaria de la Comunidad de Madrid "10. Programación de sistemas electrónicos (robótica)" (Consejería de Presidencia de la Comunidad de Madrid, 2015).

Antes de las sesiones prácticas objeto del estudio, se celebran dos sesiones teóricas sobre conceptos de robótica y de programación de robots para trabajar los conceptos teóricos de la unidad didáctica:

- Automatismos y robots.
- Sistemas de control.
- Elementos de un sistema de control.
- Las tarjetas controladoras Arduino y su programación.
- La tarjeta controladora ZUM de BQ.
- Los robots y su programación.
- Partes de la tarjeta Arduino.
- Cómo conectar la tarjeta Arduino al ordenador.

Esta unidad didáctica sobre robótica y programación de robots se ejecuta después de que en el primer trimestre se completara la unidad sobre algoritmos de programación, lo que debería contribuir positivamente al abordaje de las nuevas actividades.

La Figura 7 recoge una de las páginas del material elaborado para las sesiones teóricas sobre conceptos de robótica. En ella se explican los sensores, uno de los componentes de los sistemas de control, y los tipos de sensores más utilizados en robótica educativa. Algunos de los mostrados en la figura, como el sensor de ultrasonidos y el de infrarrojos serán utilizados en las actividades previstas de programación de robots con Kibotics. El contenido completo de la unidad se encuentra en el apartado "6.1 Conceptos de robótica" del anexo.

***Figura 7**. Unidad didáctica sobre robótica. Fuente: Elaboración propia*

2.7.3 Metodología didáctica

Desde el punto de vista metodológico, las sesiones consisten en la exposición por parte del profesor de conceptos teóricos y prácticos, a la vez que recibe retroalimentación sobre los conocimientos previos de los alumnos y, una vez expuesta la información necesaria, son los alumnos los que toman la iniciativa para resolver las actividades prácticas de programación propuestas en las plataformas Kibotics y Scratch, con el soporte en todo momento a demanda de los alumnos. Además, se prevé durante el desarrollo de las sesiones la llamada de atención sobre información relevante contenida en el propio material entregado a los alumnos si se percibiera excesiva dificultad en la ejecución de las actividades.

2.7.4 Las plataformas

2.7.4.1 Kibotics

Kibotics es una plataforma web para la enseñanza de programación y robótica educativa accesible a través de la dirección www.kibotics.org de Internet. Está dirigida principalmente a la comunidad educativa de los ciclos de primaria y secundaria. En ella los alumnos aprenden conceptos básicos de tecnología, robótica y programación de robots desde un enfoque eminentemente práctico a través de la resolución de ejercicios de programación con Scratch o Python. Los ejercicios se plantean acompañados de material explicativo audiovisual suficiente para que los estudiantes puedan resolver los ejercicios de forma autónoma y aprender haciendo. No obstante, el acompañamiento de un docente que proporcione la base previa necesaria a los estudiantes y que resuelva las cuestiones que puedan surgir durante la realización de las actividades, reduce considerablemente la curva de adaptación a la plataforma y acelera el aprendizaje.

Para el uso de la plataforma solo es necesario disponer de una conexión a Internet y de un navegador web actualizado, es decir, no requiere instalación de programas adicionales en el ordenador.

La plataforma permite programar tanto robots reales como simulados (Mbot, drones…). Esto permite probar los algoritmos de programación en un entorno virtual antes de realizar las pruebas en un entorno real. Una de las ventajas de usar entornos virtuales es la facilidad de acceso a la actividad, pues no es necesario disponer de componentes electrónicos para su realización. Y en caso de disponer de los robots y querer ejecutar los algoritmos programados en el entorno virtual en el real (Figura 8), es posible depurar los programas antes de llevarlos al entorno real, aislándolos de la parte electrónica del robot, lo que evita las interferencias de posibles, y no poco frecuentes, errores electrónicos durante la programación.

***Figura 8**. Kibotics. Robots Reales. Recuperado el 3 de mayo de 2022 de https://kibotics.org*

El aprendizaje está organizado en cursos de programación de robots con los lenguajes Scratch (bloques) o Python (Figura 9).

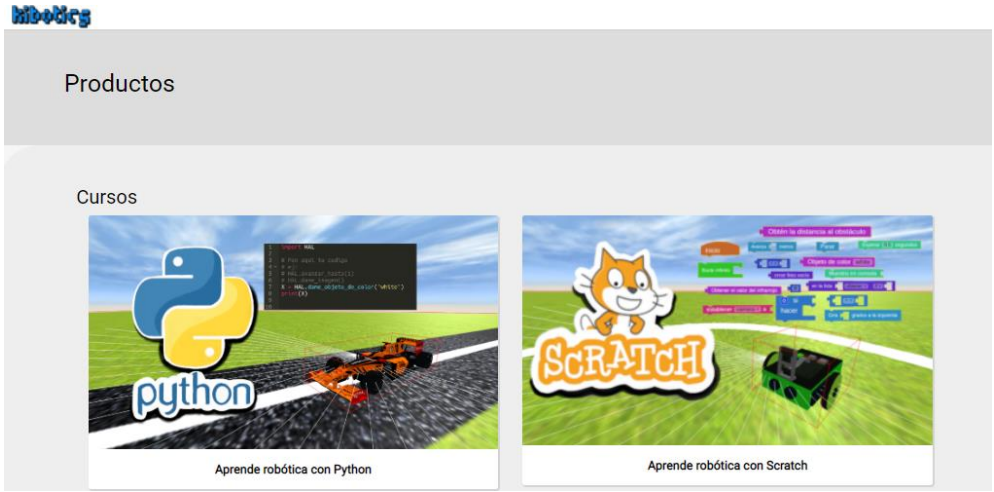


***Figura 9**. Kibotics.org.Cursos. Recuperado el 3 de mayo de 2022 de https://kibotics.org*

Además de los dos cursos de la Figura 9, la plataforma organiza torneos de forma periódica en la que los estudiantes puedan poner a prueba las habilidades de programación desarrolladas durante la realización de los cursos (Figura 10).

Figura 10. *kibotics.org. Juegos Robóticos 2022. Recuperado el 3 de mayo de 2022 de https://kibotics.org*

Kibotics cuenta además con un foro en el que los usuarios comparten su experiencia y dificultades en las distintas pruebas de programación, recibiendo soporte tanto de la propia comunidad de usuarios como del equipo de programadores de la plataforma (Figura 11).

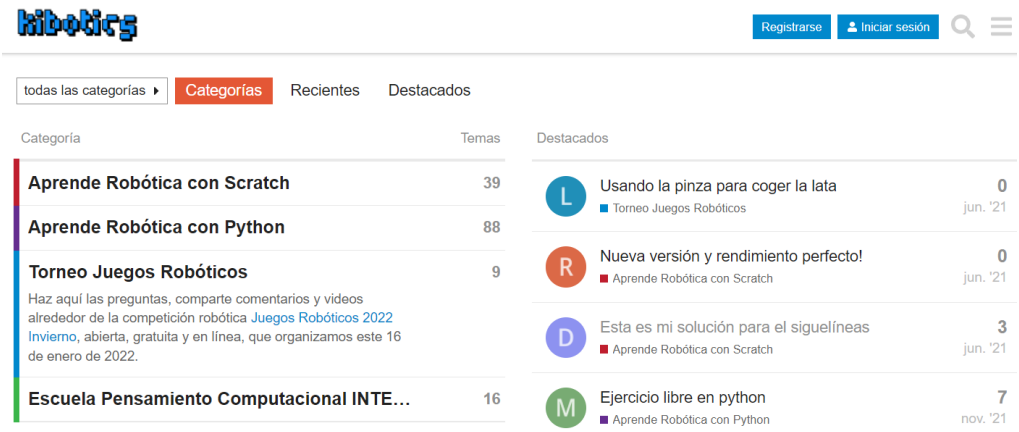


Figura 11. *Kibotics.org. Foro. Recuperado el 3 de mayo de 2022 de https://foro.kibotics.org/*

2.7.4.2 Cursos y actividades con Kibotics

En este apartado se explica en qué consiste el curso "Aprende robótica con Scratch" de Kibotics, elegido para las actividades realizadas con los alumnos, por haberse considerado el más adecuado para iniciación en la programación de robots, por la complejidad de sus actividades y por usar el lenguaje de programación Scratch, desarrollado explícitamente para estudiantes en las edades en las que se encuentran los alumnos de 2º de ESO y que están iniciándose en la programación.

El curso "Aprende robótica con Scratch" está enfocado para que los estudiantes "aprendan haciendo" a la vez que reciben contenidos sobre conceptos necesarios para el aprendizaje en forma de texto, imágenes y videos.

Para asegurarse de que los alumnos afrontan el curso con una base de conocimientos sobre robótica, el lenguaje Scratch y la plataforma, el curso de Kibotics incluye cuatro sesiones al inicio sobre estos contenidos.

Tras estas primeras lecciones introductorias, se plantean distintos retos a los estudiantes de programación de los robots GoPiGo, MBot, Roomba y de un coche de fórmula 1. Los retos consisten en que los robots ejecuten algoritmos muy conocidos en robótica educativa: sigue-línea, choca-gira, sigue la pelota, reto del pañuelo, atravesar un bosque de obstáculos o recoger confeti con el robot aspiradora Roomba. En la Figura 12 pueden verse algunas de las actividades disponibles en la plataforma dentro del curso "Aprende robótica con Scratch".

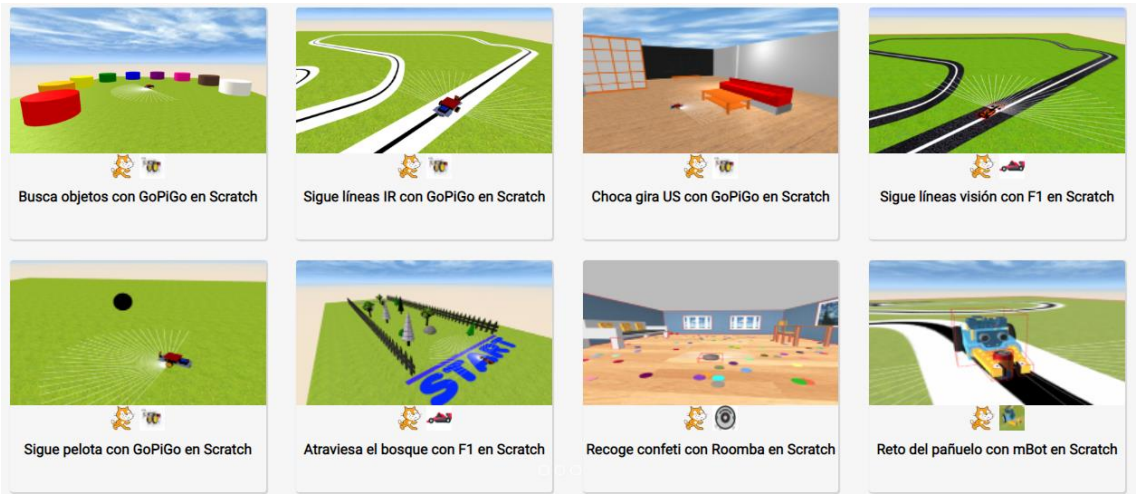


***Figura 12**. Actividades Aprende robótica con Scratch. Recuperado el 5 de mayo de 2022 de https://kibotics.org*

Las actividades están clasificadas con distinto nivel de dificultad (1 a 3 engranajes) y duración (2 a 4 horas), lo cual, permite al docente, o al estudiante, valorar la conveniencia de su realización en un función de los conocimientos previos y del tiempo disponible para su realización.

2.7.4.3 Entorno de programación de Kibotics

La plataforma dota al usuario, durante la realización de las actividades, de múltiples herramientas para la programación de los movimientos de los robots en el entorno virtual 3D:

- Instrucciones: Descripción detallada del ejercicio a realizar en la plataforma. Comienza con una introducción del ejercicio (Figura 13), después se explican los conceptos de robótica y programación que se trabajan en el ejercicio junto con consejos para su realización.

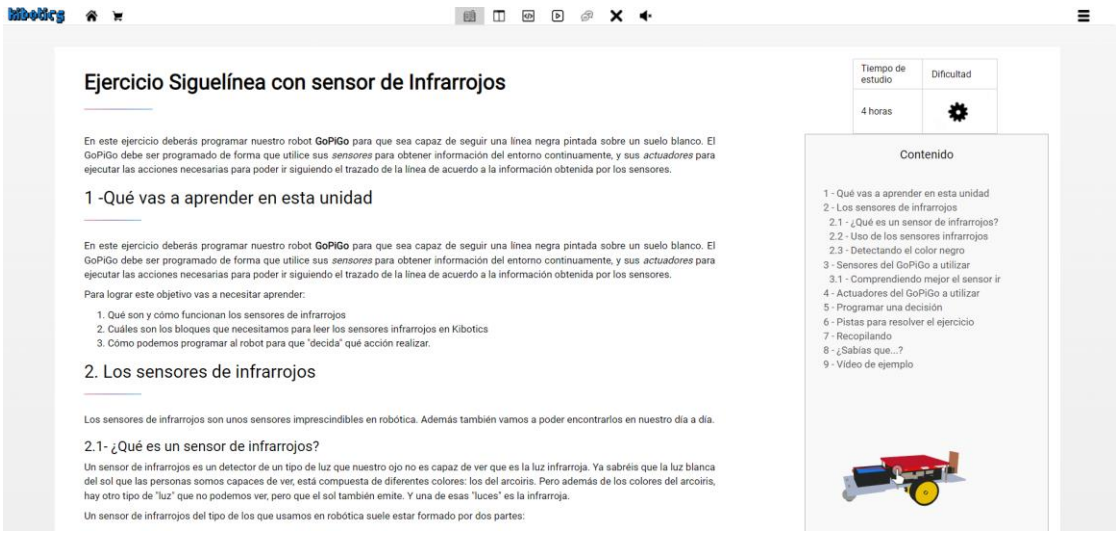


***Figura 13**. Ejercicio Siguelínea con Scratch. Recuperado el 5 de mayo de 2022 de https://kibotics.org*

- Panel de programación con bloques Scratch: en el panel de programación el usuario va construyendo el programa conforme va añadiendo bloques de instrucciones que arrastra de la paleta de instrucciones al panel de código (Figura 14).

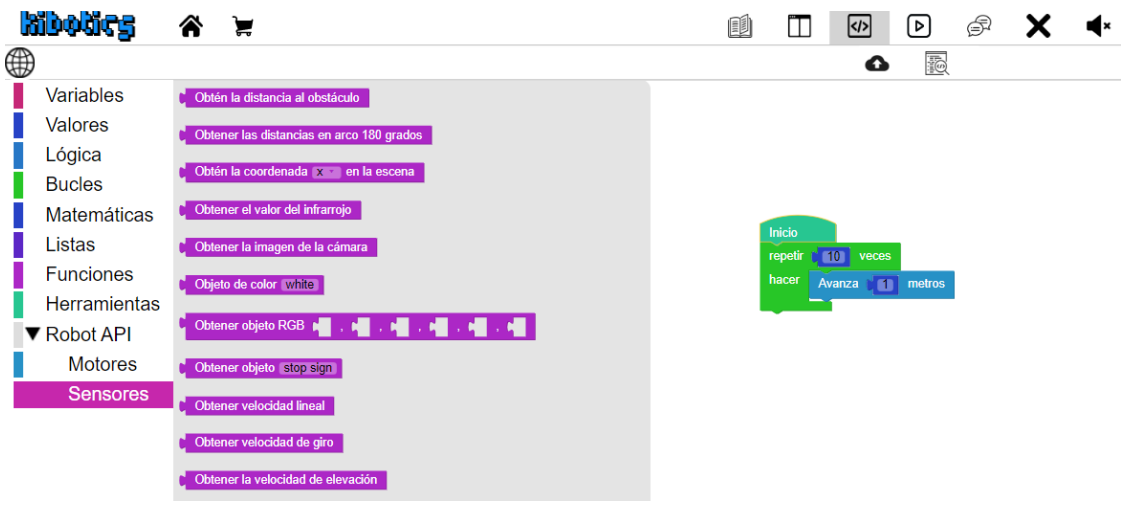


***Figura 14**. Panel de programación con Scratch. Recuperado el 5 de mayo de 2022 de https://kibotics.org*

- Entorno 3D de ejecución: permite seguir el movimiento del robot desde diferentes perspectivas: frontal, atalaya y vista de pájaro, y en modo VR o pantalla completo (Figura 15).
- Depuración: muestra en pantalla herramientas para la depuración de los programas en tiempo real del robot (Figura 15):
    - Consola: permite mostrar valores de las funciones y variables que se usan en los bloques de código.
    - Panel de sensores: muestra visualmente lo que están captando los sensores del robot en tiempo real: la cámara frontal, los dos sensores de infrarrojos y el sensor de distancia.
    - Mapa: muestra un mapa de situación del robot en el escenario de la actividad.
- Operador: permite al desarrollador mover libremente al robot para llevarlo hasta la ubicación del escenario deseada para la ejecución del código o para la comprobación de los sensores, por ejemplo. (Figura 15).

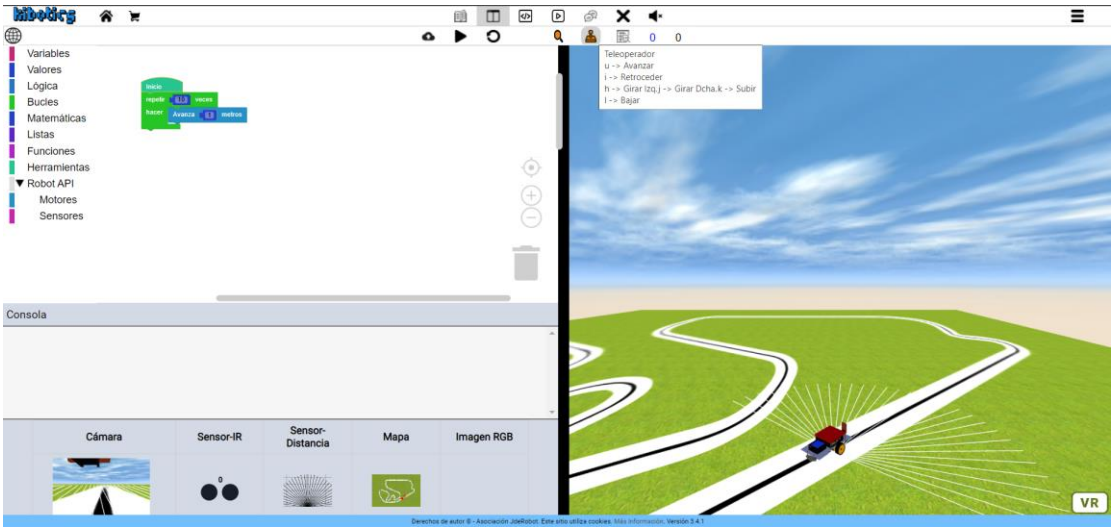


Figura 15. *Entorno 3D, Depuración y Operador. Recuperado el 5 de mayo de 2022 de https://kibotics.org*

2.7.4.4 Scratch

Scratch es un lenguaje de programación desarrollado y publicado en 2003 por el MIT Media Lab y la compañía Playful Invention Company con el propósito de ayudar a los jóvenes, con edades a partir de los 8 años principalmente, a aprender a programar. La primera versión solo estaba disponible en versión de aplicación de escritorio pero a partir de la versión 2.0, que se publicó en mayo de 2013, también estuvo disponible la versión web. En 2018 se publicó la versión 3.0, que es la actualmente disponible, tanto en versión web como de escritorio. Esta última versión incorporó elementos interactivos que, a modo de asistente, facilitan el aprendizaje del uso de la plataforma y del lenguaje de programación a sus usuarios. Además, existe una versión denominada Scratch Junior para tabletas, especialmente diseñada para edades entre 5 y 7 años (Unahalekhaka y Bers, 2021).

Scratch es un lenguaje de programación visual, de bloques, que permite desarrollar el pensamiento computacional (Zhang y Nouri, 2019) a la vez que se inicia a sus usuarios en la programación sin necesidad de tener conocimientos previos, desde cero, y cuyo aprendizaje puede aplicarse después en lenguajes de programación de alto nivel como Java, Python o C#. (Figura 16)

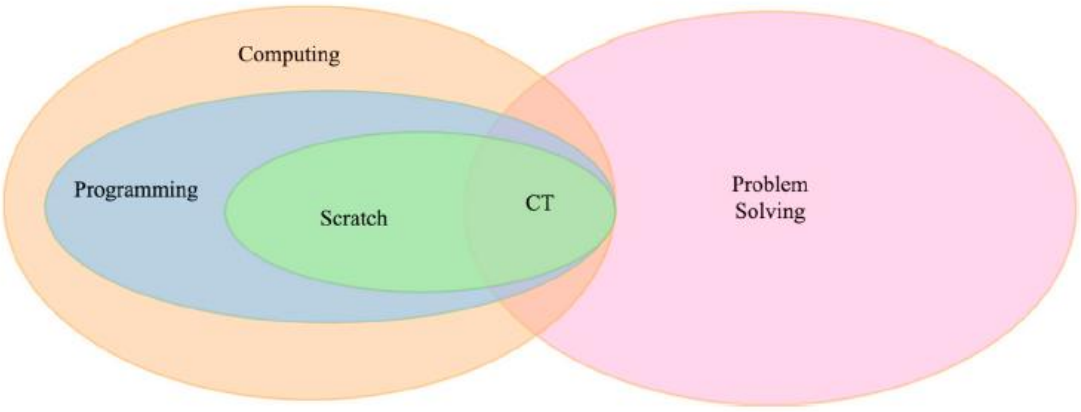


Figura 16. *Relación entre conceptos coding, programming, computing and Computational Thinking. Fuente: Zhang y Nouri (2019)*

Scratch dispone de una comunidad de usuarios que no ha parado de crecer en todo el mundo desde su creación. Sus usuarios pueden contribuir a la plataforma publicando sus proyectos para que otros usuarios los vean y puedan usarlos como base para sus propios proyectos. En abril de 2020 Scratch entró en el top 20[5] del índice TIOBE[6], que mide la popularidad de los lenguajes de programación, convirtiéndose en el primer lenguaje de programación creado específicamente para niños que entraba en la lista. Actualmente ocupa el puesto 24 de este mismo índice (Figura 17)

Figura 17*. Índice TIOBE mayo 2022. Recuperado el 1 de junio de 2022 de https://www.tiobe.com/tiobe-index/*

Demostrada la utilidad de Scratch como lenguaje de iniciación a la programación en jóvenes, son muchas las plataformas y herramientas que lo han adoptado como lenguaje de programación. Kibotics es una de ellas, pero hay muchas otras como Lego

[5]https://www.i-programmer.info/news/98-languages/13626-scratch-makes-its-debut-in-tiobe-index.html
[6] https://www.tiobe.com/tiobe-index/

Mindstorms o Arduino, que gracias al protocolo de extensión lo utilizan como interfaz de programación para programar sus placas controladoras de robótica educativa.

2.7.4.5 Recursos de aprendizaje con Scratch

Dentro de la propia web de Scratch hay un apartado con recursos para educadores y estudiantes que facilitan la preparación de las clases de los docentes y el autoaprendizaje y estudio de los estudiantes (Figura 18). Incluso invita a los educadores a unirse a grupos de redes sociales con otros docentes en los que compartir experiencias y colaborar.

Figura 18. *Recursos educativos de Scratch. Recuperado el 5 de mayo de 2022 de https://scratch.mit.edu/educators*

Dentro de los recursos para estudiantes, la plataforma incluye tutoriales agrupados por los distintos tipos de proyectos que permite desarrollar: Animaciones, Arte, Música, Juegos e Historias (Figura 19). Cada uno de estos grupos tiene tutoriales de diferentes niveles con los que aprender los conceptos básicos y el uso de las herramientas de Scratch para la creación de proyectos.

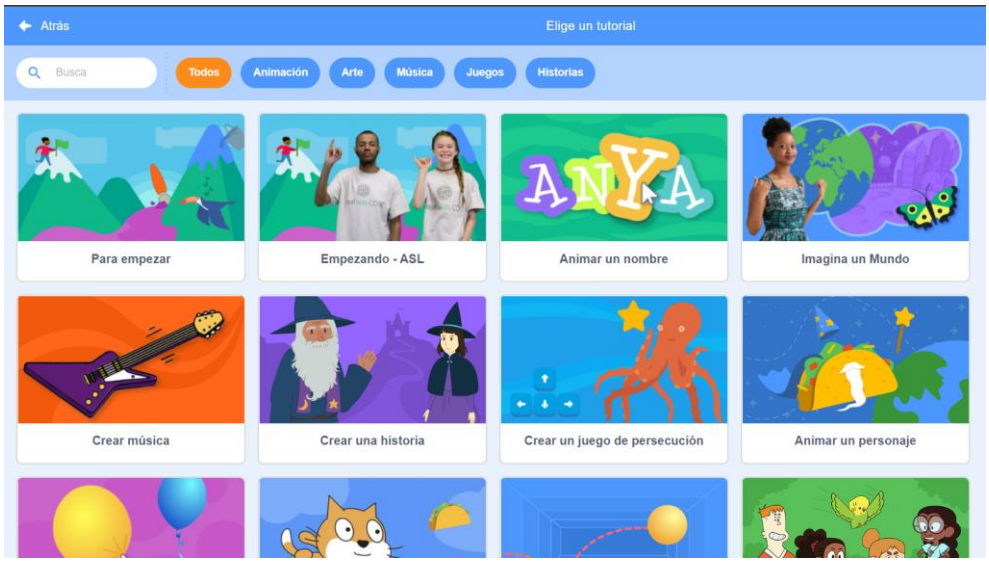


Figura 19. *Tutoriales de Scratch. Recuperado el 5 de mayo de 2022 de https://scratch.mit.edu/projects/editor/?tutorial=all*

2.7.4.6 Entorno de programación

El entorno de programación de Scratch es completamente gráfico, multi-idioma y muy intuitivo para facilitar el uso y aprendizaje de sus usuarios objetivo, jóvenes a partir de 8 años de edad.

La interfaz de usuario de Scratch está dividida en 5 grandes bloques (Figura 20):

- Barra de menú: desde sus opciones se permite a los usuarios realizan acciones relacionadas con sus proyectos y su usuario como cargar proyectos desde sus equipos, guardar proyectos en sus equipos, iniciar sesión o elegir el idioma. Aunque no es necesario estar registrado en la plataforma para crear proyectos, es recomendable porque si no se está registrado no se guardan los proyectos en la plataforma y para no perder el trabajo hay que exportar los proyectos al ordenador antes de cerrarlos.
- Paleta de bloques, disfraces, fondos y sonidos: desde esta sección de la pantalla se tiene acceso a los elementos con los que se creará el código, los bloques, a los sonidos incorporados al proyecto de la galería de Scratch o externos, a los disfraces o modos de visualización de los personajes y a la edición de los fondos del proyecto.

- Área de programación: es la zona en la que se van apilando los bloques para crear los programas en lenguaje Scratch. En Scratch se programa añadiendo funcionalidad a sus personajes y fondos, es decir, cada objeto tiene asociado su fragmento de código.
- Escenario: es el espacio en el que se ejecuta el programa. Es la interfaz de usuario del programa en la que interactúan personajes, fondos y usuario.
- Zona de objetos y escenarios: se trata del apartado en el que pueden verse y modificarse las propiedades de los personajes (nombre, posición en el escenario, tamaño, dirección y visualización). Esta zona también permite la incorporación de escenarios desde la galería de Scratch o importados.

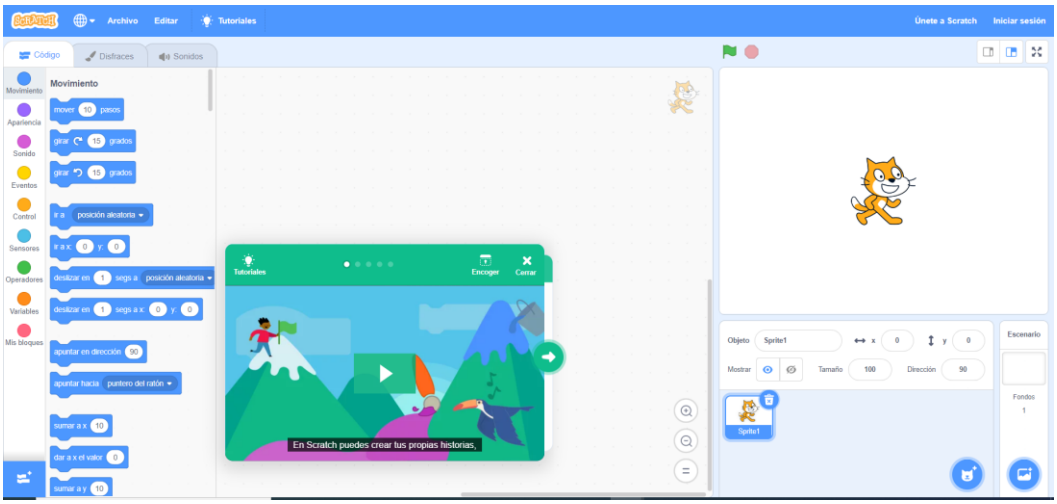


Figura 20. *Entorno de programación de Scratch. Recuperado el 5 de mayo de 2022 de https://scratch.mit.edu/projects/editor/?tutorial=all*

2.7.5 Actividades prácticas de programación

Las actividades de programación elegidas en la plataforma Kibotics para las sesiones fueron seleccionadas con el objetivo de iniciar a los estudiantes en la programación de robots, teniendo en cuenta que no tenían ninguna experiencia previa en este tipo de programación:

1. Cuadrado con GoPiGo en Scratch: actividad para empezar a utilizar la plataforma en la que el usuario utiliza únicamente los actuadores del robot para introducir y trabajar el concepto de lazo abierto o programación del robot "a ciegas" a la vez que se familiariza al usuario con los bloques de control de los actuadores del robot. Consiste en programar el robot para que pase por las esquinas que forman un cuadrado (Figura 21).

Figura 21. *Actividad Cuadrado con GoPiGo en Scratch. Fuente: www.kibotics.org. Recuperado el 5 de mayo de 2022 de https://kibotics.org*

   Los conceptos previstos que se han de usar para resolver la actividad son:
   a. Dirección
   b. Bucles repetir-veces
2. Choca gira ultrasonidos con GoPiGo en Scratch: actividad en la que se introduce el concepto de lazo cerrado con el uso del sensor de ultrasonidos, que se utiliza para evitar colisiones con los obstáculos del escenario en el que se mueve el robot. Consiste en programar el robot para que, haciendo uso de la lectura de un sensor de ultrasonidos, se mueva libremente por una habitación sorteando obstáculos (paredes, muebles, etc.) (Figura 22).

***Figura 22**. Actividad Choca gira ultrasonidos con GoPiGo en Scratch. Fuente: www.kibotics.org. Recuperado el 5 de mayo de 2022 de https://kibotics.org*

Los conceptos previstos que se han de usar para resolver la actividad son:

a. Dirección
b. Bucles repetir-hasta
c. Condicionales simples (if)
d. Funciones

3. Siguelíneas infrarrojos con GoPiGo en Scratch: actividad en la que se introduce un nuevo sensor, el de infrarrojos, que requiere un algoritmo más sofisticado para recorrer el circuito sin salirse de la línea. Consiste en programar un robot para que recorra un circuito usando un sensor que permite detectar el camino (Figura 23).

***Figura 23**. Actividad Siguelíneas infrarrojos con GoPiGo en Scratch. Fuente: www.kibotics.org. Recuperado el 5 de mayo de 2022 de https://kibotics.org*

Los conceptos previstos que se han de usar para resolver la actividad son:

a. Dirección
b. Bucles repetir-hasta
c. Condicionales compuestas (if-else)
d. Funciones

4. Recoge confeti con Roomba en Scratch: actividad que consistente en mejorar el algoritmo desarrollado en la actividad "Choca gira ultrasonidos con GoPiGo en Scratch" para que el robot aspirador, en lugar del GoPiGo, además de esquivar los obstáculos de la habitación, recorra la mayor superficie y recoja el mayor número de confeti posible (Figura 24).

Figura 24. *Actividad Recoge confeti con Roomba en Scratch. Fuente: www.kibotics.org. Recuperado el 5 de mayo de 2022 de https://kibotics.org*

Los conceptos previstos para resolver la actividad son:

a. Dirección
b. Bucles repetir-hasta
c. Condicionales simples (if)
d. Funciones

Una vez seleccionadas las actividades en Kibotics, se diseñaron las actividades equivalentes en Scratch que permitieran a los alumnos de ambos grupos, experimental y de control, trabajar los mismos conceptos y algoritmos presentes en la programación de robots. Para la confección de las actividades de Scratch se siguió la misma estructura utilizada en la plataforma Kibotics con el objetivo de que los alumnos tuvieran la misma información teórica para afrontar las actividades y que trabajaran los mismos conceptos.

2.7.6 Temporalización

Las sesiones previstas siguen una secuencia de aprendizaje que permite a los alumnos ir asimilando los conceptos en un orden de complejidad creciente y abordar los nuevos conceptos y actividades con la información previa y experiencias prácticas necesarias.

En la Tabla 1 se recogen, en orden de ejecución, las 7 sesiones de 50 minutos de duración celebradas.

***Tabla 1**. Sesiones: Temporalización. **Fuente:** Elaboración propia.*

Sesión	Nombre	Contenido
1	Robótica	Exposición de conceptos clave de robótica, de los elementos que forman los sistemas de control y de la tarjeta Arduino.
2	Programación de robots	Exposición sobre las herramientas y lenguajes de programación utilizados para la programación de robots educativos, introducción a las plataformas a usar en las actividades prácticas, revisión de conceptos sobre pensamiento computacional vistos en el primer trimestre y de programación con bloques necesarios para abordar las actividades prácticas de programación de robots.
3	Cumplimentación de Test de PC (pretest)	Sesión en la que los estudiantes realizan el test sobre pensamiento computacional.
4	Cuadrado con Scratch 	Programación con Scratch del movimiento para que el robot de la plataforma realice los movimientos necesarios para formar un cuadrado que pase por cuatro esquinas definidas en el escenario, usando idealmente las estructuras de programación más adecuadas y, teniendo presente el concepto de actuador expuesto en las sesiones 1 y 2.
5	Chocagira con Scratch 	Una vez familiarizado con la plataforma y con las instrucciones de movimiento con la sesión 2, se plantea la actividad de programación del robot para que recorra el escenario salvando los obstáculos que se encuentre. Para ello, además de las estructuras de programación necesarias el alumno tendrá que poner en práctica los conceptos de robótica sobre actuadores y sensores de obstáculos expuestos en las sesiones 1 y 2.
6	Siguelíneas con Scratch	En las sesiones previas el alumno ya debe controlar los movimientos de avance, retroceso y giro del robot así como las

Sesión	Nombre	Contenido
		instrucciones de toma de decisión en función de la lectura del sensor de obstáculos, por lo tanto, se considera que tiene la suficiente pericia para incorporar el uso de un nuevo sensor y del desarrollo de un nuevo algoritmo de movimiento. En lugar de evitar obstáculos deberá usar los sensores para seguir una línea.
6	Roomba con Scratch 	Este ejercicio, similar al realizado en la sesión 5, donde el alumno debe haber programado un algoritmo que permita al robot recoger el escenario salvando los obstáculos, se incluye como actividad ampliación para aquellos alumnos que avancen a mayor ritmo. El ejercicio consiste en mejorar el algoritmo de la sesión 5, para recorrer la mayor superficie posible del escenario de forma que recoja el mayor número de confeti en el menor tiempo posible.
7	Cumplimentación de Test de PC (posttest)	Sesión en la que los estudiantes repiten el test sobre pensamiento computacional.

En el apartado “6 Anexos” se han incluido los enunciados de las actividades utilizadas por los alumnos en las distintas plataformas, Kibotics y Scratch, con el objetivo de que pueda apreciarse el tipo de actividades realizadas y pueda reproducirse la experiencia por otro docente en el aula fácilmente.

2.8 Análisis de resultados y conclusiones

Realizada la intervención educativa y la recogida de la información, se procede al análisis de los resultados para contrastarlos con la hipótesis u objetivo de la investigación y sacar conclusiones.

A continuación, en los apartados “3 Resultados” y “4 Conclusiones”, se exponen los resultados y las conclusiones de la intervención.

3 Resultados

Durante el estudio se han recogido datos cuantitativos, a través del cuestionario de PC, y cualitativos, mediante la observación directa durante el desarrollo de la intervención. Sobre los primeros, se realizará un análisis estadístico y, sobre los segundos, se realizará una descripción de los hechos observados por el investigador.

3.1 Cuantitativos

Los datos obtenidos en la intervención pertenecen a una muestra no probabilística casual (por facilidad de acceso) dividida en dos grupos, el de control que consta de 37 individuos y el experimental de 48.

Para analizar los datos, se puntúa sobre 10 los aciertos a las 28 preguntas del cuestionario utilizado en los pretest y posttest, y se recogen en las variables Pre_total10 y Post_total10 con presencia en la Tabla 2 y la Figura 25.

En la Tabla 2 puede observarse como ambos grupos, control y experimental, muestran un valor mayor en el posttest (Post_total10) con respecto al pretest (Pre_total10). Aumentando también la dispersión en ambos grupos.

***Tabla 2**. Medidas estadísticas. **Fuente**: Elaboración propia.*

Variable	Grupo		Estadístico
Pre_total10	Control	Media	5,3958
		Mediana	5,3571
		Varianza	1,111
		Mínimo	3,21
		Máximo	7,50
	Experimental	Media	6,2277
		Mediana	6,6071
		Varianza	2,209
		Mínimo	3,21
		Máximo	8,57
Post_total10	Control	Media	5,4633
		Mediana	5,3571
		Varianza	2,118
		Mínimo	2,86
		Máximo	8,21

	Experimental	Media	6,7634
		Mediana	6,7857
		Varianza	2,640
		Mínimo	3,21
		Máximo	9,64

Por otro lado, si analizamos los datos de la Figura 25, se aprecia, además, una falta de homogeneidad entre el grupo de control y el grupo experimental.

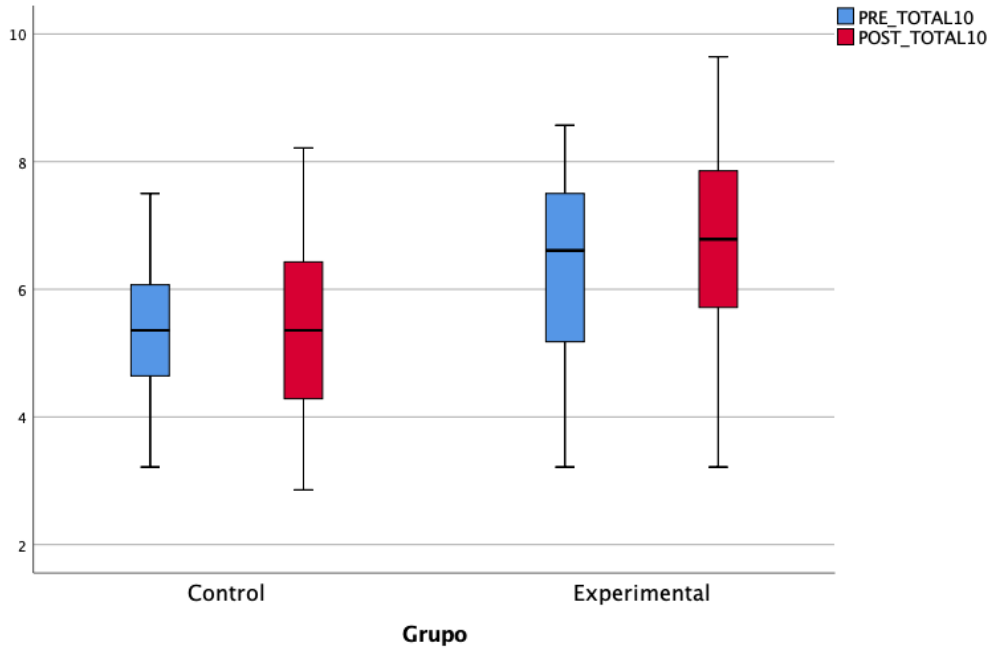


***Figura 25**. Diagrama de cajas y bigotes. Puntuaciones Pretest y Posttest. Fuente: Elaboración propia*

La diferencia entre ambos grupos apreciada en la Figura 25 se verifica estadísticamente mediante un test t para muestras independientes (ambos grupos presentan una distribución normal) recogido en la Tabla 3**.** Esta falta de homogeneidad en el pretest, significación menor que 0,05 (0,012) en Pre_total10 en la Tabla 3, hace que no sea posible comparar directamente los resultados del posttest, pues los grupos parten de diferentes conocimientos previos.

***Tabla 3**.Test t para muestras independientes. **Fuente**: Elaboración propia.*

		Prueba de Levene de igualdad de varianzas				
		F	**Sig.**	**t**	**gl**	**Sig. (bilateral)**
Pre_total10	Se asumen varianzas iguales	6,630	**0,012**	-2,889	83	0,005
	No se asumen varianzas iguales			-3,017	82,493	0,003

Puesto que no es posible analizar los grupos juntos, se procede al análisis de la posible mejora obtenida por separado. En la Tabla 4 se muestran los resultados de un test para muestras emparejadas basado en la t de Student. En el grupo de control el aumento de la puntuación entre el pretest y el posttest no es estadísticamente significativo (p=0,763). Sin embargo, en el grupo experimental este aumento sí lo es (p=0,001).

***Tabla 4**. Prueba de muestras emparejadas. **Fuente**: Elaboración propia*

		Diferencias emparejadas					t	gl	Sig. (bilateral)
		Media	Desv. Desviación	Desv. Error promedio	95% de intervalo de confianza de la diferencia				
					Inferior	Superior			
Control	**Pre-post**	-0,06757	1,35170	0,22222	-0,51825	0,38311	-0,304	36	**0,763**
Experimental	**Pre-post**	-0,53571	1,05741	0,15262	-0,84275	-0,22868	-3,510	47	**0,001**

Para tener en cuenta las peculiaridades que se han presentado por separado (en cada grupo), se presenta un modelo matemático más avanzado, ANOVA para medidas repetidas para dos factores. En este modelo se analizan dos variables dependientes relacionadas, Pretest y Posttest, en las que aparece el factor Grupo que interviene en el modelo.

Antes de aplicar el modelo ANOVA se ha comprobado que se cumplían las condiciones para ello, a saber, normalidad de los datos (test de Shapiro Wilk, p-valor>0,05), esfericidad (pruebas de espericidad de Mauchy, p-valor>0,05) y test de igualdad en la matriz de covarianzas (test M de Box, p-valor>0,05). Se descartan el factor sexo y clase al no resultar significativo el modelo. Así, se presenta en la Tabla 5 el modelo

final. Se observa que el efecto del modelo general "Pre-post" entre pretest y posttest es significativo (p=0,023), con un valor Eta parcial al cuadrado de 0,060, lo que significa un alto efecto entre ambas variables. Además, si tenemos en cuenta la intervención de cada nivel de la variable Grupo (control y experimental), en la segunda fila "Pre_post * Grupo", también se aprecia un valor significativo (p=0,017), por lo que se puede afirmar que el factor Grupo influye en el modelo, es decir, que las puntuaciones no evolucionan de la misma forma en el grupo de control y en el grupo experimental, algo que también se observa en la Figura 26. En el caso de la variable Grupo, el valor de Eta parcial al cuadrado (p=0,037) denota en este caso una influencia moderada.

***Tabla 5**. Modelo ANOVA. **Fuente**: Elaboración propia*

Origen	Pre_post	Tipo III de suma de cuadrados	gl	Media cuadrática	F	Sig.	Eta parcial al cuadrado	Potencia observada
Pre-post	**Lineal**	3,802	1	3,802	5,334	**0,023**	**0,060**	0,627
Pre_post * Grupo	**Lineal**	2,290	1	2,290	3,212	**0,017**	**0,037**	0,425
Error(Pre_post)	**Lineal**	59,163	83	0,713				

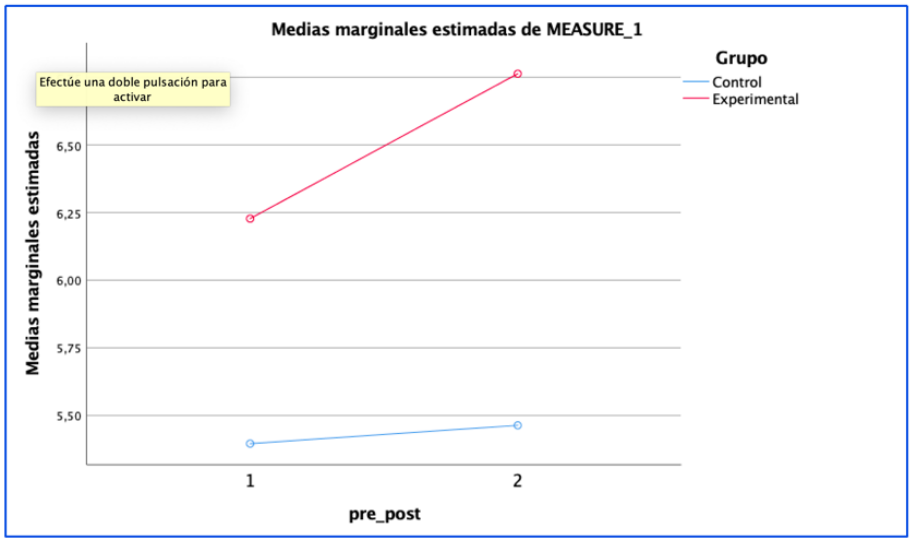


***Figura 26**. Medias marginales estimadas Pre_Post Grupo. Fuente: Elaboración propia*

Por otro lado, se realiza un análisis descriptivo de los datos relativos al concepto computacional reflejados en la Figura 27, la cual, nos muestra las diferencias, en media, de las puntuaciones obtenidas para cada uno de los conceptos, según la leyenda: Direcciones (Direction), Bucles-Repetir-Veces (Loops_t), Bucles-Repetir-Hasta (Loops_u), Condicionales-Simple (Condit_s), Condicionales-Compuesto (Condit_c),

Condicionales-Mientras (Condit_w), Funciones-Simples (Funtion). En este caso las puntuaciones son evaluadas entre 0 y 4.

En lo que respecta al grupo de control, en la Figura 27 puede verse que no solamente no ha mejorado en diversos conceptos, sino que ha habido empeoramiento en el caso de Direcciones (Direction), Bucles-Repetir-Veces (Loops_t), Bucles-Repetir-Hasta (Loops_u). En Condicionales-Simples (Condit_s), la nota media se ha mantenido y, para el resto, Condicionales-Compuestos (Condit_c), Condicionales-mientras (Condit_w) y Funciones (Function), ha habido mejora, mínima en los dos últimos y mayor en el caso de Condicionales-Compuestos (Condit_c).

En cuanto al grupo experimental, la Figura 27 muestra que ha habido mejora en todos los conceptos, salvo en el concepto Funciones (Function), donde la puntuación se ha mantenido.

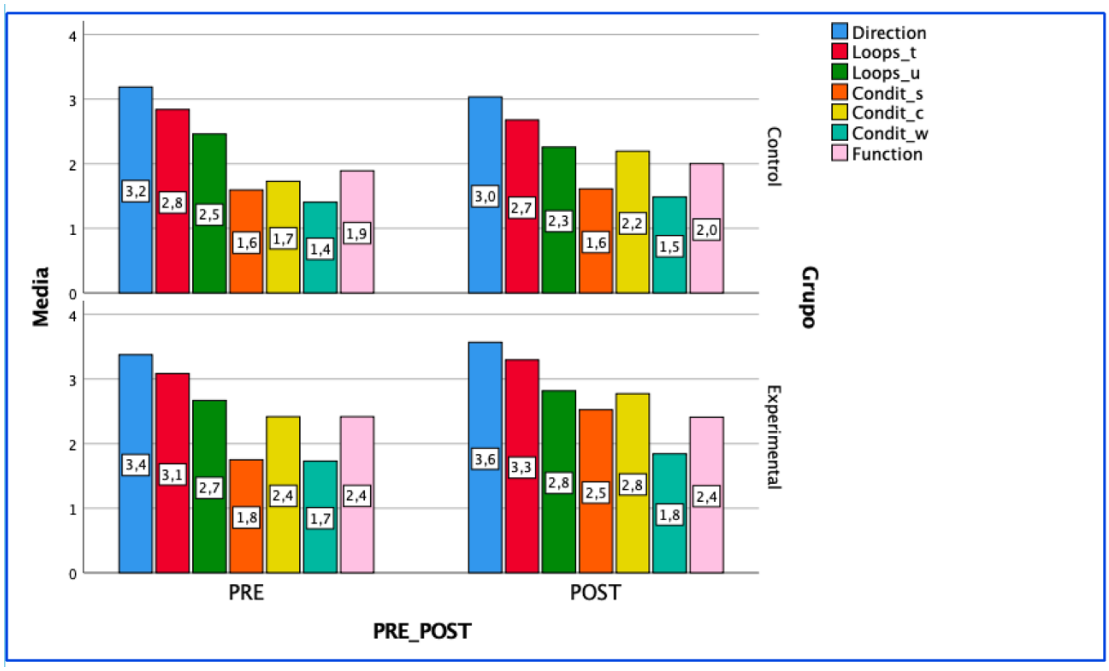


***Figura 27**. Diagrama de barras de la media pretest y posttest por grupo y concepto. Fuente: Elaboración propia*

Por último, se analiza a partir de la Figura 28 los resultados de las preguntas realizadas a los estudiantes, sobre su valoración personal y objetiva de cómo les había salido el test (variable RESULTADOTEST) y "cómo se le daban los ordenadores" (variable ORDENADORES), utilizando escala Likert con valores comprendidos entre 0 y 10. Observando el gráfico, que recoge la media de las variables para los dos grupos en

el pretest y posttest, puede afirmarse que no se aprecian diferencias significativas en los resultados obtenidos entre el pretest y el posttest en ninguno de los dos resultados estudiados.

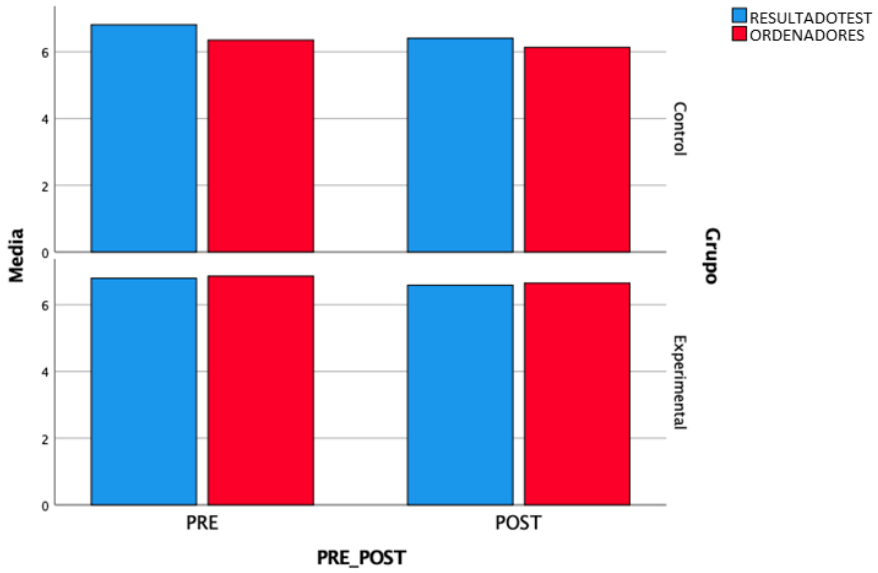


Figura 28. *Diagrama de barras de la encuesta sobre el resultado del test y conocimiento de ordenadores. Fuente: Elaboración propia*

3.2 Cualitativos

Durante las sesiones con los estudiantes se recoge, mediante la observación e interacción directa, información adicional a la prevista en los cuestionarios pretest y posttest.

En general, se observa que los participantes tenían los conocimientos previos necesarios sobre pensamiento computacional y Scratch para abordar las actividades.

Cuando se realiza la sesión introductoria de las sesiones y se informa a los estudiantes que van a realizar actividades relacionadas con robots, y su programación, se percibe un alto interés por parte de los participantes.

Las dos primeras sesiones, que tienen como objetivo dotar a los alumnos de los conceptos necesarios sobre robótica educativa y algoritmos de movimiento de este tipo de robots, se desarrollan con mucho interés en términos generales, aunque también se percibe impaciencia y ganas de empezar con las actividades prácticas.

En la sesión de realización del pretest de pensamiento computacional algunos alumnos hacen preguntas sobre el cuestionario, pero tras las primeras cuestiones todos entienden la dinámica de las preguntas del ejercicio y lo cumplimentan como mejor saben, a pesar de no haber realizado ninguno similar previamente.

En la primera sesión de ejercicios prácticos con las plataformas, ambos grupos tienen que programar el movimiento de un robot para que pase por cuatro esquinas que forman un cuadrado. El grupo experimental, que usa por primera vez la herramienta Kibotics, tiene mayores dificultades para iniciar la actividad que el grupo que usa la plataforma Scratch, que ya conoce la herramienta. No obstante, la mayor dificultad que se aprecia no es la herramienta en sí misma, es la forma en la que abordar la actividad, pues los estudiantes manifiestan que ya han programado algoritmos que hacen ese movimiento pero no utilizando los actuadores o movimientos propios de un robot, lo que les supone pensar de nuevo el algoritmo y definir los pasos uno a uno sin una pauta que seguir previa.

Otra de las cuestiones que desconcierta bastante a los estudiantes es el comportamiento inesperado de los robots en la plataforma Kibotics, que a diferencia de la plataforma Scratch, incorpora variables en el movimiento como la fricción, que requieren añadir condiciones adicionales al algoritmo, usando los sensores, para mantener el control del movimiento del robot.

Al finalizar las sesiones se percibe que a pesar de haber dispuesto del mismo tiempo, mientras que unos alumnos han realizado con bastante agilidad las actividades planteadas, otros han mostrado ciertas dificultades, y otra parte, la menos numerosa, ha mostrado muchas dificultades.

En la última sesión hubo oportunidad de recibir retroalimentación sobre la plataforma de los grupos que usaron Kibotics. Se invitó a los estudiantes a expresar su opinión y, aquellos que participaron, indicaron que a pesar de que Kibotics estuviera basado en el lenguaje Scratch, les había parecido menos intuitivo, lo que relacionaban con la disposición de los bloques de programación y con movimientos "no esperados", poniendo como ejemplo el ejercicio del cuadrado, en el que el robot se mueve de forma diferente a la que esperaban debido al rozamiento con la superficie, que les supuso una dificultad añadida en los primeros pasos en la plataforma. Además, sugirieron mejoras en la plataforma como que la disposición de los bloques de programación fuera en una lista única, no dentro de secciones, e incorporar un buscador de bloques. Otra mejora que propusieron fue la de incorporar la posibilidad de tener más de una versión de una actividad, de forma que se pudiera trabajar en diferentes enfoques de solución a la vez y hacer múltiples pruebas sin perder cambios, o bien, poder exportar el código a un archivo para versionado o compartir el código con terceras personas.

4 Conclusiones

El objetivo del estudio era comprobar la influencia del uso de la herramienta Kibotics en el aprendizaje de la programación y del pensamiento computacional en alumnos de segundo de la ESO en la asignatura "Tecnología, Programación y Robótica". En los resultados se observa que los grupos de control y experimental parten de conocimientos diferentes en pensamiento computacional, por lo tanto, no es posible sacar conclusiones comparando los resultados de un grupo con el otro, pero sí es posible comparar el aprendizaje obtenido en cada grupo por separado. La primera conclusión que se obtiene, a la vista de los resultados y teniendo en cuenta la premisa indicada, es que los dos grupos han mejorado sus conocimientos. No obstante, ambos no han mejorado de la misma forma, el grupo que usó la herramienta Kibotics ha mejorado de forma significativa en los conceptos de programación y pensamiento computacional Direcciones, Bucles-Repetir-Veces, Bucles-Repetir-Hasta, Condicionales-Simples, Condicionales-Compuestos y Condicionales-mientras, es decir, en todos menos en Funciones, que no ha variado. En cambio, para el grupo de control las mejoras no fueron significativas.

El número de sesiones fue exactamente el mismo para ambos grupos, por lo tanto, aunque los estudiantes partieran de conocimientos diferentes, el hecho de que el grupo que usó Kibotics consiguiera progresar en mayor medida que el grupo que usó la otra plataforma, permite afirmar que el empleo de la robótica educativa de Kibotics ha influenciado de forma positiva en la aceleración del aprendizaje de los estudiantes en conceptos de PC.

En cuanto a los resultados sobre la autoevaluación de los alumnos, no se obtienen conclusiones significativas de su análisis, pues apenas cambian entre el pretest y el posttest. Lo que permite concluir que el uso de una herramienta u otra no ha influido en la autoevaluación sobre el resultado del test de pensamiento computacional o sobre los conocimientos sobre ordenadores en general.

Respecto al desarrollo de las sesiones de aprendizaje, destacar que la mayor dificultad fue atender las cuestiones e incidencias técnicas que surgían durante su trascurso por el alto número de alumnos involucrados en las actividades y la corta

duración de cada sesión, 50 minutos. No obstante, a pesar de las dificultades, para los alumnos fue motivador el hecho de acudir a los talleres de informática y llevar a la práctica los conceptos de robótica vistos en clase. Por otro lado, a pesar de que para ambos grupos resultaba más fácil a priori el uso de la plataforma Scratch, por conocerlo previamente y parecerles más intuitivo, la sensación fue que los alumnos que utilizaron Kibotics mantuvieron hasta la última sesión la motivación más alta hacia las actividades.

Sobre el impacto de la intervención, indicar que la retroalimentación recibida por los alumnos, y trasladada a los responsables de la plataforma Kibotics, ha propiciado el desarrollo y publicación en Internet de una nueva versión de la plataforma, que ya incorpora la funcionalidad que permite múltiples versiones de una misma actividad.

Destacar también el esfuerzo realizado para la obtención de la evaluación favorable del Comité de Ética de la Universidad Rey Juan Carlos. En estudios en los que participan menores, como es el caso, los requisitos están muy marcados y su cumplimiento requiere un esfuerzo y tiempo adicional durante la preparación de la intervención, que si no se tiene en cuenta puede retrasar el inicio de la investigación.

En relación a líneas futuras de trabajo, en primer lugar, se propone repetir la intervención ampliando el número de sesiones y en un periodo de tiempo mayor. El disponer de mayor tiempo permitiría tratar con mayor detalle los conceptos de pensamiento computacional incluidos en cada actividad, daría la oportunidad a los alumnos a progresar a un ritmo más adaptado a su punto de partida y necesidades, y además, daría tiempo a que se manifestara en mayor medida el progreso en el aprendizaje realizado al asentar los conocimientos. En segundo lugar, se propone la incorporación de un test de valoración subjetiva de la actividad, pues aunque se recibió retroalimentación de forma verbal por parte de algunos alumnos, esta información no ha podido traducirse a datos cuantitativos y puede no recoger las opiniones de todos los participantes. Y por último, se propone estudiar la influencia de usar Kibotics para programar robots reales en el aprendizaje de la programación y del desarrollo del PC en comparación con su uso para la programación de robots virtuales.

5 Referencias bibliográficas

- Aho, A. V. (2012). Computation and computational thinking. The Computer Journal, 55(7), 832-835. DOI:10.1093/comjnl/bxs074 https://people.cs.vt.edu/~kafura/CS6604/Papers/Computation-And-CT.pdf
- Bers, M. U. (2010). The TangibleK Robotics program: Applied computational thinking for young children. *Early Childhood Research & Practice,* 12(2), 2. https://eric.ed.gov/?id=EJ910910
- Bisquerra, R. y Pérez-Escoda, N. (2015). ¿Pueden las escalas Likert aumentar en sensibilidad? *REIRE, Revista d'Innovació i Recerca en Educació,* 8 (2), 129-147. DOI: 10.1344/reire2015.8.2.828. https://www.researchgate.net/publication/280310402_Pueden_las_escalas_Likert_aumentar_en_sensibilidad
- Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A. y Engelhardt, K. (2016). *Developing computational thinking in compulsory education – Implications for policy and practice*; EUR 28295 EN; doi: 10.2791/792158. DOI: 10.2791/792158. https://publications.jrc.ec.europa.eu/repository/handle/JRC104188.
- Brackmann, C., Barone, D., Casali, A., Boucinha, R. y Muñoz-Hernández, S. (2016). Computational thinking: Panorama of the Americas. *Computers in education (SIIE), international symposium on IEE* (pp. 1–6). DOI: 10.1109/SIIE.2016.7751839. https://www.researchgate.net/publication/310807767_Computational_thinking_Panorama_of_the_Americas
- Caballero-González, Y. A. y García-Valcárcel, A. (2020). ¿Aprender con robótica en Educación Primaria? Un medio de estimular el pensamiento computacional. *Education in the Knowledge Society (EKS),* 21, 15. https://doi.org/10.14201/eks.22957 https://revistas.usal.es/index.php/eks/article/view/eks20202110
- Consejería de Presidencia de la Comunidad de Madrid. (Mayo 2015). Decreto 48/2015, de 14 de mayo, del Consejo de Gobierno, por el que se establece para

la Comunidad de Madrid el currículo de la Educación Secundaria Obligatoria. http://www.madrid.org/wleg_pub/servlet/Servidor?opcion=VerHtml&nmnorma=8934

- Denner J., Campe S. y Werner L. (2019). Does Computer Game Design and Programming Benefit Children? A Meta-Synthesis of Research. *ACM Trans. Comput. Educ.* 19, 3, Article 19 (January 2019), 1-35. https://doi.org/10.1145/3277565
- García Lázaro, D., Versteegen G. y García Muiña, F. (2018). *Investigación y gestión del conocimiento*. (1ª edición). OMMPRESS.
- García-Valcárcel, Y. A. y Caballero-González, Y. (2019). Robotics to develop computational thinking in early Childhood Education. [Robótica para desarrollar el pensamiento computacional en Educación Infantil]. *Comunicar*, 59, 63-72. https://doi.org/10.3916/C59-2019-06
- Heintz, F., Mannila, L. y Färnqvist, T. (2016). A review of models for introducing computational thinking, computer science and computing in K-12 education. *Frontiers in education Conference (FIE), 2016* (pp. 1–9). IEEE. DOI: 10.1109/FIE.2016.7757410. https://ieeexplore.ieee.org/document/7757410
- Kazakoff, E. R., Sullivan, A. y Bers, M. U. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education*, 41, 245–255. https://doi.org/10.1007/s10643-012-0554-5
- Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J. y Werner, L. (2011). Computational thinking for youth in practice. *Acm Inroads*, 2(1), 32–37. https://doi.org/10.1145/1929887.1929902 https://users.soe.ucsc.edu/~linda/pubs/ACMInroads.pdf
- Margulieux, L. E., Catrambone, R. y Guzdial, M. (2016). Employing subgoals in computer programming education. *Computer Science Education*, 26(1), 44–67. https://doi.org/10.1080/08993408.2016.1144429.
- Montes-León H., Hijón-Neira R., Pérez-Marín D. y Montes-León R. (2020). "Mejora del Pensamiento Computacional en Estudiantes de Secundaria con

Tareas Unplugged," *Education in the Knowledge Society*, vol. 21, 2020, Art no. 24, doi: 10.14201/eks.23002.

- Ouahbi, I., Kaddari, F., Darhmaoui, H., Elachqar, A. y Lahmine, S. (2015). Learning basic programming concepts by creating games with scratch programming environment. Procedia-Social and Behavioral Sciences, 191, 1479–1482. https://doi.org/10.1016/j.sbspro.2015.04.224
- Papadakis, S., Kalogiannakis, M. y Zaranis, N. (2016). Developing fundamental programming concepts and computational thinking with ScratchJr in preschool education: A case study. *International Journal of Mobile Learning and Organisation*, 10(3), 187–202. DOI:10.1504/IJMLO.2016.077867. https://www.researchgate.net/publication/305390965_Developing_fundamental_programming_concepts_and_computational_thinking_with_ScratchJr_in_preschool_education_A_case_study.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books.
- Pea, R. D. y Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology*, 2(2), 137–168. DOI:10.1016/0732-118X(84)90018-7
- Resnick, M. (1996). *Constructionism in practice: Designing, thinking, and learning in a digital world*. New paradigms for computing, new paradigms for thinking. In Y. Kafai, & M. Resnick (Eds.). Mahwah, NJ: Erlbaum.
- Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K. (2009). Scratch: Programming for all. *Communications of the ACM*, 52(11), 60–67. https://doi.org/10.1145/1592761.1592779
- Román González, M. (2016). *Codigoalfabetización y pensamiento computacional en educación primaria y secundaria: validación de un instrumento y evaluación de programas.* (Tesis doctoral) Universidad Nacional de Educación a Distancia (España). Escuela Internacional de Doctorado. Programa de Doctorado en Educación. http://e-spacio.uned.es/fez/view/tesisuned:Educacion-Mroman
- Román-González, M., Pérez-González, J. C. y Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of

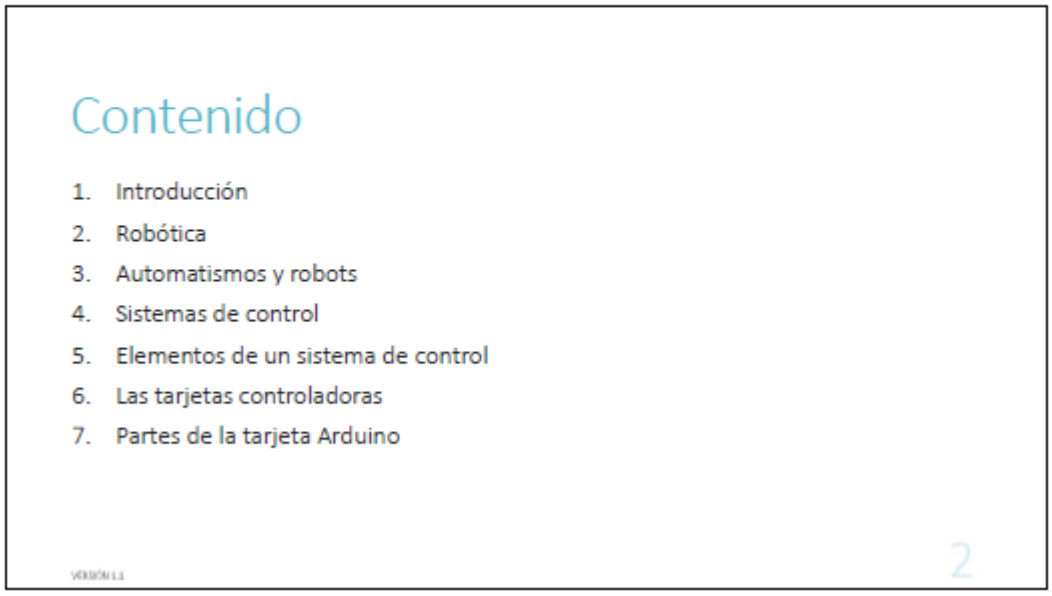
the computational thinking test. *Computers in Human Behavior*, 72, 678–691. https://doi.org/10.1016/j.chb.2016.08.047. https://www.sciencedirect.com/science/article/pii/S0747563216306185

- Seoane-Pardo, A. M. (2018). Computational thinking between philosophy and STEM. Programming decision making applied to the behaviour of "moral machines" in ethical values classroom. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, 13(1), 20-29. https://doi.org/10.1109/RITA.2018.2809940.
- Sovic, A., Jagušt, T., y Seršic, D. (2014). How to teach basic university-level programming concepts to first graders? *Integrated STEM education conference (ISEC)*, 2014IEEE (pp. 1–6). IEEE. DOI: 10.1109/ISECon.2014.6891050
- Strawhacker, A., Portelance, D., Lee, M., y Bers, M. (2015). Designing tools for developing minds: The role of child development in educational technology. *IDC 2015 workshop*. https://sites.tufts.edu/devtech/files/2018/02/devtech-position-idc2015.pdf
- Unahalekhaka, A. y Bers, M.U. (2021). Taking coding home: analysis of ScratchJr usage in home and school settings. Education Tech Research Dev 69, 1579–1598. https://doi.org/10.1007/s11423-021-10011-w
- Wing, J. M. (2006). Computational thinking. *Commun. ACM* 49, 3 (March 2006), 33–35. https://doi.org/10.1145/1118178.1118215 http://www.cs.cmu.edu/afs/cs/usr/wing/www/publications/Wing06.pdf
- Wing, J. M. (2011). Research notebook: Computational thinking— what and why. *The magazine of the Carnegie Mellon University School of Computer Science*. https://people.cs.vt.edu/~kafura/CS6604/Papers/CT-What-And-Why.pdf
- Wing, J. M. (2016). Computational thinking, 10 years later. *Microsoft Research Blog*. https://www.microsoft.com/en-us/research/blog/computational-thinking-10-years-later
- Witherspoon E., Higashi R., Schunn C., Baehr E. y Shoop R. (2017). Developing Computational Thinking through a Virtual Robotics Programming Curriculum. *ACM Trans. Comput. Educ.* 18, 1, Article 4 (March 2018), 20 pages. https://doi.org/10.1145/3104982

- Zhang L. y Nouri J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education*, Volume 141, https://doi.org/10.1016/j.compedu.2019.103607. https://www.sciencedirect.com/science/article/pii/S0360131519301605

6 Anexos

6.1 Conceptos de robótica



1. Introducción

En las próximas clases aprenderemos lo **que es un robot, sus partes, para qué sirve, qué es un programa** y cómo **programar robots**.

Para ello veremos primero un poco de teoría, después pondremos a prueba lo que sabemos de programación, realizaremos actividades de programación en robots simulados de algoritmos muy habituales en robótica y, para finalizar, volveremos a poner a prueba nuestras habilidades de programación.

1. Un poco de teoría → 2. Ejercicio de pensamiento computacional → 3. Programar robots → 4. Ejercicio de pensamiento computacional

VERSIÓN 1.1

3

2. Robótica I

La robótica es una rama de la tecnología que combina los conocimientos de diversas ingenierías: **mecánica, eléctrica, electrónica e informática** que se ocupa del **diseño, construcción, operación, estructura, manufactura y aplicación** de los robots.

Aplicaciones de la robótica: **Tareas de las tres Ds**

Tareas peligrosas (dangerous): tareas peligrosas para las personas. Por ejemplo, robots que desactivan bombas, viajan a marte o limpian centrales nucleares.

Tareas aburridas (dumb): tareas aburridas, repetitivas y tediosas como limpiar el suelo o almacenes automatizados.

Tareas sucias (dirty): tareas sucias, en entornos ruidosos y hostiles. Por ejemplo, ordeñar vacas.

VERSIÓN 1.1

4

2. Robótica II

En la actualidad se usan para otras muchas tareas, por ejemplo:

- **Medicina:** Exoesqueletos médicos que ayudan a andar a personas con dificultades.

- **Agricultura:** En agricultura se usan drones en los campos de cultivo para tomar fotos aéreas con un sensor infrarrojo que permite saber el grado de estrés vegetativo del terreno y aplicar a cada zona un tipo personalizado de variedad de cultivo, riego y fertilizante.

- **Urbanismo:** Ciudades inteligentes como el proyecto Smart City de Pozuelo de Alarcón que cuenta con sistemas de riego, aparcamiento y de iluminación telegestionados.

- **Doméstico:** Astro es un nuevo robot desarrollado por la empresa Amazon que combina las funcionalidades de sus asistentes virtuales Alexa (audio), Echo Show (video) y videovigilancia.

5

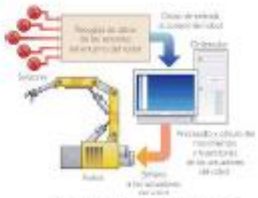
3. Automatismos y robots I

Automatismo: es un mecanismo o máquina que realiza una **tarea concreta**, pero cuyo funcionamiento **no se puede modificar**. Ejemplos: limpiaparabrisas de un coche, cisterna de un váter, un semáforo..

El semáforo es un ejemplo de automatismo cíclico, pues repite la tarea indefinidamente.

Robot: es una máquina automática **programable** capaz de **captar** la información de su entorno, **procesarla** y **actuar** en consecuencia.

- **Percibe su entorno**, es decir, sabe lo que pasa a su alrededor.
- **Procesa la información**. Realiza cálculos y toma decisiones a partir de la información recibida.
- **Actúa** a partir del resultado del proceso de la información.
- **Se comunica** con otros robots, sistemas o personas.



Este brazo robótico programable es un ejemplo de robot que actúa en función del entorno percibido a través de sus sensores

Un **robot** suele desempeñar **tareas que reemplazan o extienden el trabajo humano**: manipulación de objetos pesados, desactivación de explosivos, tareas de gran precisión, trabajo en el espacio..

6

4. Sistemas de control

Sistema de control: Un sistema de control es un conjunto de elementos o dispositivos electrónicos que, **al recibir información** del exterior (sensores), generan una **respuesta a la salida**.

Los **robots utilizan sistemas automáticos de lazo cerrado**. En ellos, la señal de salida se compara con la de entrada para ajustar la acción al valor deseado a través de actuador.

Sistemas automáticos: Los sistemas automáticos son aquellos que **solo precisan la intervención humana para su puesta en marcha y en caso de bloqueo por alguna incidencia**. Ejemplos: calentador eléctrico de agua, la cisterna de un inodoro, tostadora, lavadora, brazo mecánico con herramienta de soldadura...

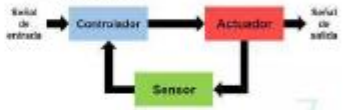
Los **sistemas automáticos** pueden ser de dos tipos:

- **De lazo abierto**: El ciclo que se realiza está **prefijado y no se modifica en función del resultado del proceso**, tanto si es correcto como si no lo es. Ejemplo: El microondas una vez puesto en funcionamiento calienta hasta que se completa el tiempo programado independientemente de si la comida está caliente o fría.

- **De lazo cerrado**: En estos sistemas **hay un sensor que permite ajustar el mecanismo de control** (Comparador y Controlador) **en función de la respuesta del sistema**. Ejemplo: El calentador eléctrico de agua calienta el agua con una resistencia y controla su temperatura con un sensor denominado termostato que cuando detecta que se ha alcanzado la temperatura deseada apaga la resistencia que calienta el agua.

En un robot que suelda, cuando el sensor detecta una pieza, acciona el brazo del robot para situarlo en el punto que debe soldar y luego realiza la soldadura:

Sensor: detecta la pieza y su colocación.
Comparador: comprueba que está en el lugar correcto
Controlador: en función de la señal que le llega del comparador envía una señal al actuador para que mueva los motores.
Actuador: al recibir la señal del controlador mueve los motores y acciona la soldadura.



7

5. Elementos de un sistema de control

En robótica, la **tarjeta controladora** junto con los **sensores** y **actuadores** de los robots forman el sistema de control.

- Sensores

- Actuadores

- Tarjeta controladora

8

5. Elementos de un sistema de control I

Sensores: un sensor es un dispositivo **capaz de detectar** magnitudes físicas o químicas (**temperatura, humedad, intensidad luminosa, distancia, aceleración, presión, humedad, fuerza, etc.**) y **transformarlas en variables eléctricas** que pueden ser interpretadas por la tarjeta controladora de un sistema de control.

Los sensores más utilizados en robótica son:

Sensor de infrarrojos
Detecta la luz reflejada y, por tanto, es capaz de diferenciar entre blanco y negro o claro y oscuro

Intensidad luminosa o LDR (Light dependent resistor)

Sensor de temperatura o NTC (negative temperatura coefficient)

Sensor de final de carrera o final de recorrido
Se trata de un pulsador usado por ejemplo en las puertas de garajes para detectar cuando se ha cerrado o abierto la puerta completamente

Sensor de Ultrasonidos
Utiliza ondas sonoras de ultrasonidos para saber a qué distancia se encuentra un objeto

9

5. Elementos de un sistema de control II

Actuadores: un actuador es un dispositivo **capaz de transformar energía eléctrica en otro tipo de energía** (mecánica, lumínica, etc.) y así activar procesos como hacer girar un motor, encender una bombilla, un led o hacer sonar un timbre.

Principales actuadores que usaremos en robótica:

Motores eléctricos C.C.
Los **motores de corriente continua** normalmente tienen sólo **dos cables, uno positivo y otro negativo**. Es muy importante **no accionar el motor directamente desde los pines de la placa Arduino** pues, además de no darle suficiente potencia, esto puede dañar la placa. **Debe utilizarse un circuito controlador de motores** que a su vez recibe corriente directamente de la pila.

Servomotor
Son **motores CC de precisión** que tienen en su interior un motor con reductora de velocidad y un pequeño circuito de control que solo permite el giro desde los 0 a los 180º. Algunos permiten rotación continua. Tienen tres cables: uno para +5V, otro a GND y otro a un pin digital.

Diodos led
Tienen polaridad, por lo tanto, deben conectarse correctamente el ánodo y cátodo. Necesitan una resistencia en serie de un valor aproximado de 220 ohmios para poder soportar la corriente.

10

5. Elementos de un sistema de control III

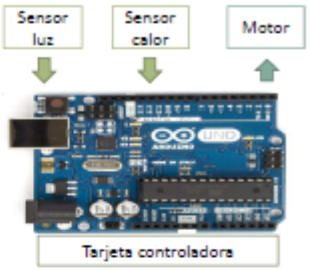
Tarjeta controladora: La tarjeta controladora (o placa controladora) es un circuito electrónico que permite **controlar los procesos** que realiza un robot.

Una tarjeta controladora está **compuesta por los componentes básicos de un ordenador**:

- Unidad de proceso
- Memoria
- Adaptadores de señales de
  - *entrada*
  - *Salida*

La tarjeta controladora procesa las señales de entrada que recibe de los sensores y calcula las órdenes que hay que darle a los actuadores del robot, como motores, luces, displays, etc. Por otro lado, para programar la tarjeta se necesita un ordenador.

Las más extendidas son Arduino y ZUM de BQ.



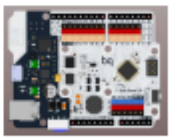
11

6. Las tarjetas controladoras I

Existen distintos **modelos de tarjetas controladoras**. Actualmente dos de las más utilizadas son las tarjetas **ZUM de BQ** y **Arduino UNO**.

ZUM de BQ

- **Software de programación Bitbloq 2.**
- **Bitbloq 2 es una plataforma de programación online con bloques similares a los de Scratch que requiere conexión a internet.** http://bitbloq.bq.com
- **Compatible con Arduino, lo que quiere decir que también funciona con Arduino IDE, plataforma de programación de Arduino.**



12

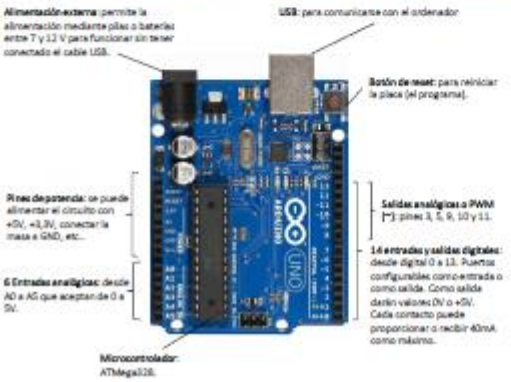
6. Las tarjetas controladoras II

Arduino

- Arduino es una plataforma de electrónica open hardware para la creación de prototipos.
- Tarjeta basada en un microcontrolador que permite conectar sensores, actuadores y otros elementos mediante sus entradas y salidas, analógicas y digitales.
- Para su programación dispone de la plataforma de programación Arduino IDE, con versión online y offline. https://www.arduino.cc/en/software
- La plataforma usa un lenguaje de programación propio de alto nivel, no de bloques, basado en el lenguaje de alto nivel Processing, similar a C++.

13

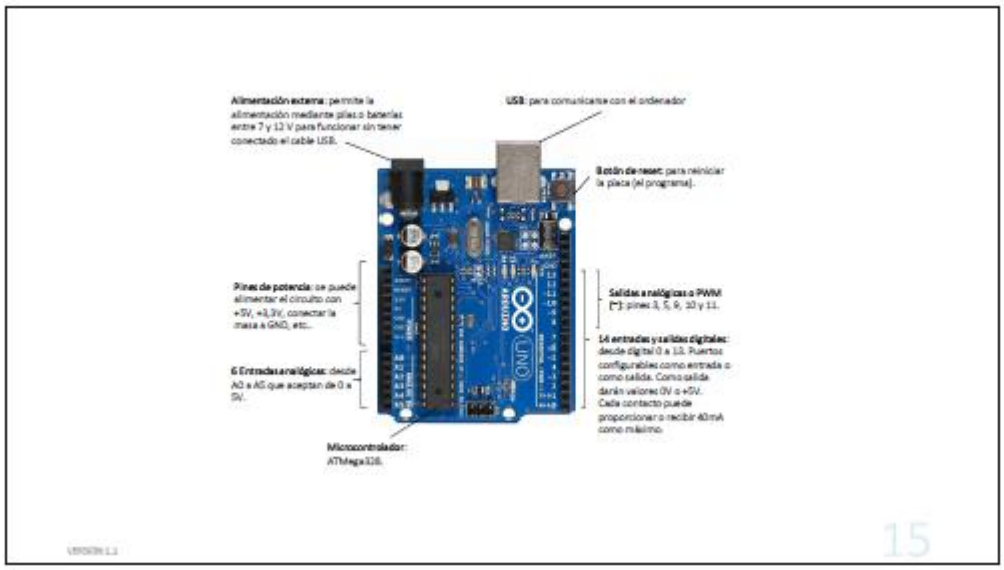
7. Partes de la tarjeta Arduino



Sensores analógicos: proporcionan una variación de voltaje dependiendo de lo que varíe la magnitud física medida. **Se deben conectar a las entradas analógicas, que traducirán los valores de tensión continua que les llegue entre 0 y 5V a valores digitales entre 0-1023.** Un ejemplo de sensor analógico es el **sensor de temperatura** LM35.

Sensores digitales: proporcionan valores de tensión que pueden ser interpretados por la tarjeta controladora como un **valor alto (high, 5V) o un valor bajo (low, 0V)**. Por ejemplo, al pulsar o soltar. Lo mismo ocurre con los **actuadores digitales**, por ejemplo, un led, que puede estar encendido o apagado.

14


Alimentación externa: permite la alimentación mediante pilas o baterías entre 7 y 12 V para funcionar sin tener conectado el cable USB.
USB: para comunicarse con el ordenador
Botón de reset: para reiniciar la placa (el programa).
Pines de potencia: se puede alimentar el circuito con +5V, +3,3V, conectar la masa a GND, etc...
Salidas analógicas o PWM (~): pines 3, 5, 6, 10 y 11.
14 entradas y salidas digitales: desde digital 0 a 13. Pueden configurarse como entrada o como salida. Como salida darán valores 0V o +5V. Cada contacto puede proporcionar o recibir 40mA como máximo.
6 Entradas analógicas: desde A0 a A5 que aceptan de 0 a 5V.
Microcontrolador: ATMega328.
15

6.2 Cuadrado con Scratch

Actividad 1: Cuadrado con Scratch

1. Introducción

Este proyecto consiste en conseguir que el personaje, un robot, se desplace formando un cuadrado por las esquinas de uno de los cuadrados del escenario.

Para completar el ejercicio no está permitido emplear bloques de sensores, es lo que se denomina programar el robot "a ciegas".

No hay restricciones para realizar la trayectoria, de modo que el cuadrado puede ser del tamaño y empezando desde la esquina que quieras.

2. Qué vas a aprender en esta unidad

En esta unidad aprenderás lo siguiente:

- Programarás el personaje para que se mueva haciendo un cuadrado.
- Qué quiere decir cuando hablamos de control de un robot en lazo abierto.

Se pide que implementes un algoritmo, en el área indicada, que permita al personaje seguir una trayectoria cuadrada, utilizando los bloques de código disponibles en el área de trabajo. Ten en cuenta que el robot debe moverse lo más parecido a como lo haría realmente.

3. Controlando al personaje

Para lograr que nuestro personaje se mueva tenemos que utilizar las instrucciones de movimiento para que avance, gire o retroceda según lo que queramos. En esta práctica vamos a hacerlo sin tener en cuenta la información de los sensores y únicamente podrá utilizarse el bloque "Ir a x: y:" en una ocasión para empezar las acciones desde un punto. Es decir, nuestro personaje se moverá "a ciegas", no pudiendo detectar si está tocando algún color en la que se encuentre, por lo tanto, no podrá tampoco detectar las marcas dibujadas en el escenario. Se va a mover tal y como lo haríamos nosotros si nos vendasen los ojos, nos tapasen los oídos y no fuéramos capaces de enterarnos de nada de lo que pase a nuestro alrededor.

En robótica se dice que esta manera de controlar a un robot sin tener en cuenta la información de los sensores (como si fuera "a ciegas") se llama **control en lazo abierto**.

4. Funciones útiles y comportamientos

Este ejercicio requerirá principalmente que utilices los bloques del área de trabajo que te permiten enviar instrucciones de movimiento al personaje. Estos bloques puedes encontrarlos en la paleta de bloques, accediendo al apartado Movimiento.

5. Algunas pistas

Antes de comenzar el algoritmo piensa lo siguiente:

¿Qué bloques de movimiento te serán más útiles para programar el robot para que haga un cuadrado?

¿Cuántos grados tiene que girar el robot para lograr que su trayectoria sea un cuadrado perfecto?

¿Qué tipo de giro hace más natural el movimiento del robot? ¿Recuerdas la instrucción de Scratch para establecerlo?

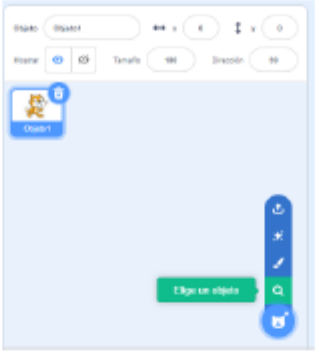
6. ¿Cómo utilizar las esquinas para conseguir una trayectoria cuadrada?

Ayúdate de las esquinas como "forma de medir" la distancia, es decir, envía órdenes al personaje que le permitan alcanzar cada una de las esquinas en función de la distancia a la que están.

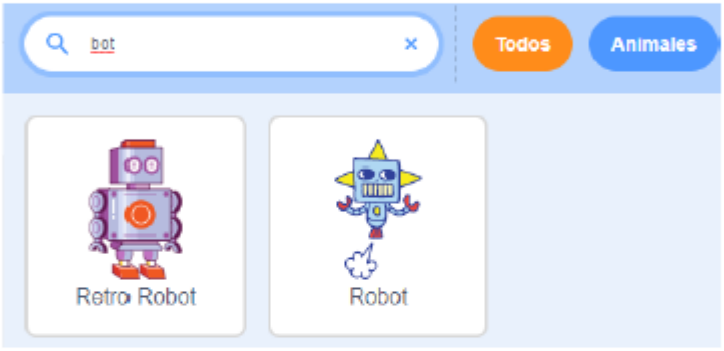
7. Para empezar

1. Accede a la web https://scratch.mit.edu/
2. Abre sesión con tu usuario seleccionando en el menú **Iniciar sesión**
3. Selecciona la opción "Crear" para crear un nuevo proyecto
4. Estable el nombre de tu proyecto con el nombre Actividad1 y a continuación tu nombre. Ejemplo: "Actividad1 Nombre Apellido1 Apellido2"

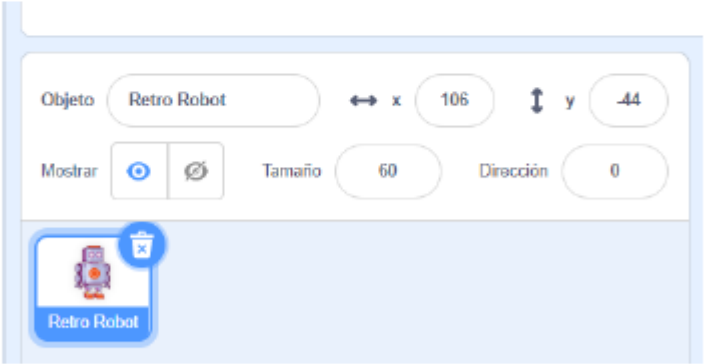
5. Carga el escenario
6. Elimina el personaje por defecto y añade el de un robot.



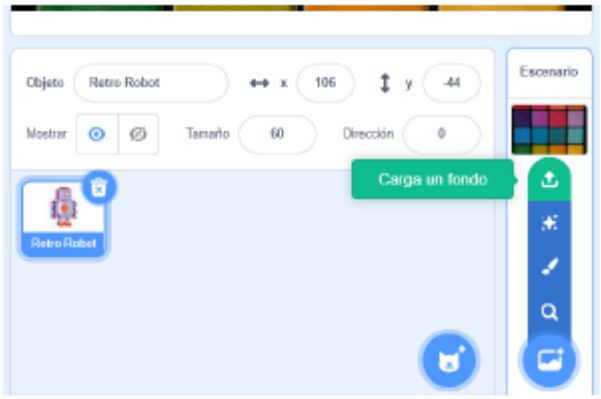
Elige el que más te guste de los disponibles:



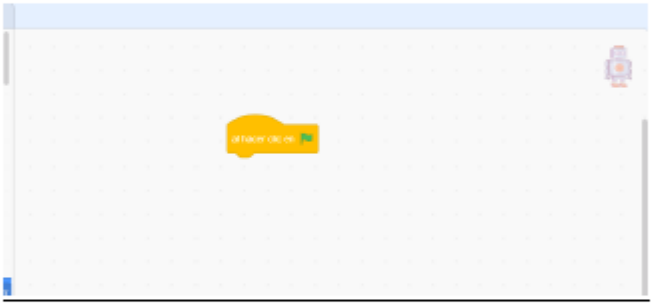
Recuerda cambiar su tamaño a 60%



7. Ahora añade el fondo con el archivo con el nombre "FondoCuadrados.jpeg" facilitado con la actividad:



8. Ya tienes todo listo para empezar a programar tu robot para que forme un cuadrado.
9. Cuando termines el ejercicio recuerda:
   a. Guardar el proyecto en el ordenador
   b. Subir el proyecto a Google Classroom junto con una captura de pantalla de Scrath en la que se vean los bloques de código.



8. Antes de terminar... ¿Sabías qué?

Aunque Scratch no tiene en cuenta cuestiones como la fricción o el entorno en los desplazamientos de sus personajes, cuando se programan robots que interacción con el entorno real sí hay que considerar la fricción del suelo y el entorno en general con el que interactuará el robot.

La fricción en robótica

La fuerza de rozamiento es una fuerza que aparece cuando dos cuerpos que se encuentran en contacto, en nuestro caso el suelo y el robot. Esta fuerza se opone al movimiento.

Veamos algunos ejemplos: Cuando andamos sobre hielo no nos cuesta nada andar, en este caso la fuerza de rozamiento es pequeña y nos podemos deslizar. Sin embargo cuando andamos en un suelo rugoso como el cemento o el hormigón sin pulir, nos cuesta mucho más andar. Esto mismo ocurre en el simulador.

Lazo abierto en robótica

Los comportamientos de robots en lazo abierto son complicados y los usamos poco. ¿Por qué? Piensa en el siguiente juego. Pintamos una línea recta en el suelo. Colocamos al jugador al comienzo de la línea y le vendamos los ojos. Tiene que ir caminando y pararse cuando crea que ha llegado al final de la línea. ¡Piensa que aunque la ha visto camina con los ojos cerrados!

Aunque en su cabeza tenga una idea de lo que mide esa línea mientras camina con los ojos cerrados pueden pasar muchas cosas: los pasos que da son más largos o más cortos de lo que piensa, por el camino hay un objeto que antes no estaba, se está torciendo... Y ¡claro! como tiene los ojos cerrados no se da cuenta y no puede corregir su trayectoria y su decisión de pararse.

Pues lo mismo le ocurre al robot. Mientras hace el cuadrado puede que sus ruedas patinen un poco y no avance lo que habíamos calculado, puede que un pequeño obstáculo en el camino que antes no estaba haga que se desvíe ligeramente, a lo mejor se cruza un objeto imprevisto con el que se choca... Todas estas situaciones imprevistas permanecen ocultas para el robot ¡va "a ciegas"!

Para resolverlo tendremos que usar la información de sensores, de manera que el robot pueda "percibir" los cambios que vayan ocurriendo a su alrededor y reaccionar adecuadamente. Pero eso será en próximas unidades.

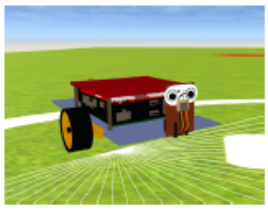
6.3 Cuadrado con Kibotics

Cuadrado con el GoPiGo

Tiempo de estudio: 2 horas

Dificultad:

El objetivo de este ejercicio es lograr que el GoPiGo realice un cuadrado.

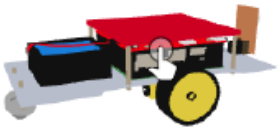


Pero...¿Cómo podremos estar seguros de que la trayectoria del robot es un cuadrado perfecto?

El GoPiGo dispondrá de ayuda "visual". Utiliza las marcas rojas del terreno para guiar al robot en todo momento.

No hay restricciones para realizar la trayectoria, de modo que el cuadrado pueder ser del lado que tú quieras, empezando por la marca que más te guste.

Contenido



1 - Que vas a aprender en esta unidad

Esta unidad es uno probablemente uno de los primeros contactos que vas a tener con la programación de un robot. En ella aprenderás lo siguiente:

- Qué quiere decir cuando hablamos de control de un robot en lazo abierto.
- Practicarás el uso de los comandos de movimiento del robot.
- Programarás el robot para que se mueva haciendo un cuadrado.

Se pide que implementes un algoritmo, en el área indicada, que permita al robot seguir la trayectoria cuadrada delimitada por las cruces rojas del suelo de forma autónoma, utilizando los bloques de código disponibles en el área de trabajo.

2 - Controlando un robot

Para lograr que nuestro robot se mueva tenemos que activar los motores de las ruedas para conseguir que el robot avance, gire o retroceda según lo que queramos. En esta práctica vamos a hacerlo sin tener en cuenta la información de los sensores. Es decir, nuestro robot se moverá "a ciegas", si hay obstáculos no los va a detectar, no podrá tampoco detectar las marcas. Se va a mover tal y como lo haríamos nosotros si nos vendasen los ojos, nos tapasen los oídos y no fuéramos capaces de enterarnos de nada de lo que pase a nuestro alrededor.

En robótica decimos que esta manera de controlar al robot sin tener en cuenta la información de los sensores (como si fuera "a ciegas") se llama **control en lazo abierto**.

3 - Funciones útiles y comportamientos

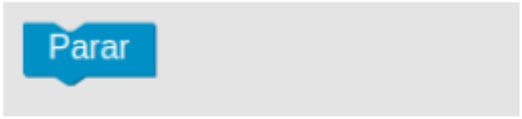
Este ejercicio requerirá principalmente que utilices los bloques del área de trabajo que te permiten enviar instrucciones a los motores del GoPiGo. Estos bloques puedes encontrarlos en el menú de trabajo, accediendo al apartado *RobotAPI* y luego haciendo click sobre el subapartado *Motors*.

Recuerda que dentro de los bloques que podemos usar para controlar el robot tenemos 3 grupos:

1. Bloques a los que tenemos que darles la velocidad de avance o retroceso o la de giro.

2. Bloques a los que tenemos que darles la distancia a recorrer en el caso de avance/retroceso o el ángulo de giro si se trata de uno de los bloques de girar.

3. Bloques a los que no tenemos que darle ningún parámetro, como el bloque *Parar*



Más información: Píldora fricción inicial

Un par de pistas

Antes de comenzar el algoritmo piensa lo siguiente:

- ¿Qué bloques de los tres grupos anteriores te serán más útiles para programar el robot para que haga un cuadrado?
- ¿Cuántos grados tiene que girar el robot para lograr que su trayectoria sea un cuadrado perfecto?

4 - ¿Cómo utilizar las marcas para conseguir una trayectoria cuadrada?

El escenario contiene una serie de marcas rojas que hay sobre el suelo en forma de cruz, como las que ves en la imagen:

Ayúdate de estas marcas como "forma de medir" la distancia, es decir, envía órdenes al robot que le permitan alcanzar cada una de las marcas en función de la distancia a la que están.

5 - Recapitulando.

En esta práctica habrás adquirido soltura en el uso de los bloques que controlan el movimiento del robot.

Habrás programado tu primer comportamiento robótico: una trayectoria cuadrada.

También habrás aprendido qué queremos decir cuando hablamos de un comportamiento en lazo abierto.

6 - ¿Sabías que...?

Los comportamientos de robots en lazo abierto son complicados y los usamos poco. ¿Por qué? Piensa en el siguiente juego. Pintamos una línea recta en el suelo. Colocamos al jugador al comienzo de la línea y le vendamos los ojos. Tiene que ir caminando y pararse cuando crea que ha llegado al final de la línea. ¡Piensa que aunque la ha visto camina con los ojos cerrados! Aunque en su cabeza tenga una idea de lo que mide esa línea mientras camina con los ojos cerrados pueden pasar muchas cosas: los pasos que da son más largos o más cortos de lo que piensa, por el camino hay un objeto que antes no estaba, se está torciendo... Y ¡claro! como tiene los ojos cerrados no se da cuenta y no puede corregir su trayectoria y su decisiónde pararse.

Pues lo mismo le ocurre al robot. Mientras hace el cuadrado puede que sus ruedas patinen un poco y no avance lo que habíamos calculado, puede que un pequeño obstáculo en el camino que antes no estaba haga que se desvíe ligeramente, a lo mejor se cruza un objeto imprevisto con el que se choca... Todas estas situaciones imprevistas permanecen ocultas para el robot ¡va "a ciegas"!

Para resolverlo tendremos que usar la información de sensores, de manera que elrobot pueda "percibir" los cambios que vayan ocurriendo a su alrededor y reaccionar adecuadamente. Pero eso será en próximas unidades.

6.4 Chocagira con Scratch

Actividad 2: Chocagira con Scratch

1. Introducción

Este proyecto consiste en conseguir que el personaje, un robot, se desplace por la habitación sorteando los obstáculos.

Para completar el ejercicio es necesario usar bloques de sensores que permitan al robot identificar que hay un obstáculo y actuar cambiando de dirección.

El robot debe moverse en línea recta por la habitación libremente sin detenerse hasta que identifique un obstáculo y tenga que cambiar su dirección para continuar moviéndose por la habitación.

2. Qué vas a aprender en esta unidad

El objetivo de esta unidad es simular con Scratch la funcionalidad del sensor de ultrasonidos muy usado en robótica para medir la distancia a objetos. ¿Se te ocurre cuál puede ser el bloque de Scratch que tiene el personaje de la actividad que pueda ayudarte a identificar un obstáculo?

El movimiento choca-gira es uno de los más comunes en robótica. Para programarlo a partir de la información del sensor deberás tomar decisiones de cambio de dirección.

3. Los sensores de ultrasonidos

Los ultrasonidos son sonidos muy agudos que el oído humano no es capaz percibir, sin embargo, hay animales que sin capaces de oírlos.

Los murciélagos, por ejemplo, además de oírlos son capaces de emitir ultrasonidos y aprovechan estas capacidades para orientarse cuando vuelan por la noche. Su cerebro es capaz de determinar la ubicación de los objetos al recibir el eco de los sonidos que el propio murciélago ha emitido.

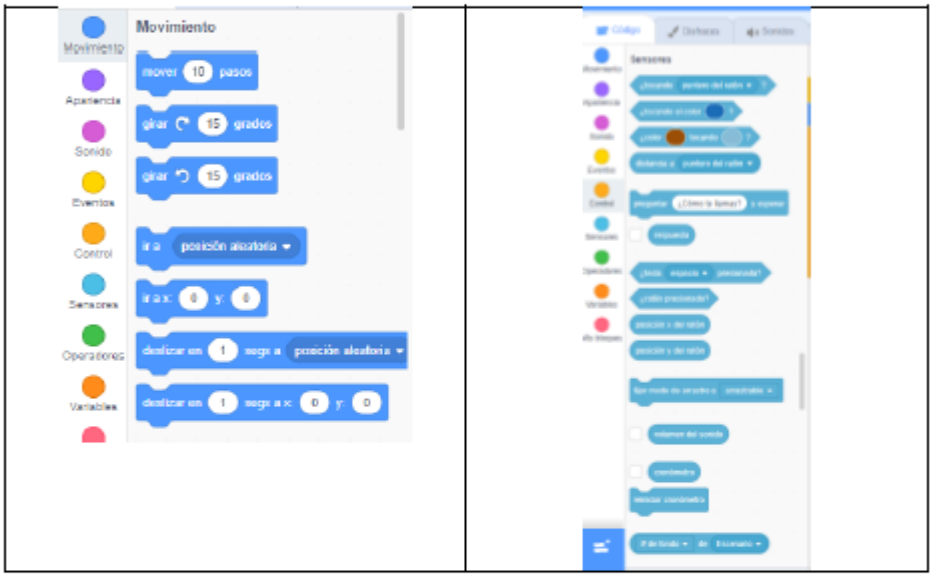
Ilustración 1Fuente YouTube: ORIENTACÒN DE LOS MURCIELAGOS

Actividades con Scratch Actividad 2: Chocagira con Scratch

En robótica el sensor ultrasonido simula la capacidad de los murciélagos y permite a los robots conocer la distancia de los objetos de delante de los ultrasonidos para actuar en consecuencia y evitar chocarse.

4. Bloques a utilizar

Además del sensor de Scratch que te permita identificar la proximidad de un objeto al personaje en el escenario, este ejercicio requerirá que utilices principalmente los bloques de control y del área de trabajo que te permiten enviar instrucciones de movimiento al personaje para avanzar, retroceder y/o girar. Estos bloques puedes encontrarlos en la paleta de bloques, accediendo los apartados: Sensores, Control y Movimiento.



5. Algunas pistas

Antes de comenzar el algoritmo piensa lo siguiente:

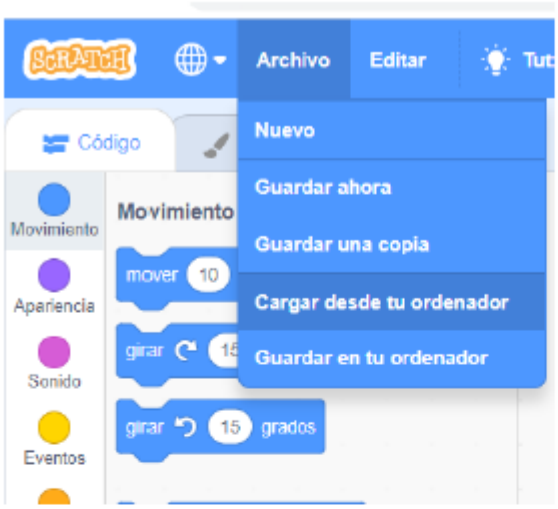
¿Qué bloques de movimiento te serán más útiles para programar el robot?

¿Qué bloques se utilizan para que el robot repita instrucciones de forma continua?

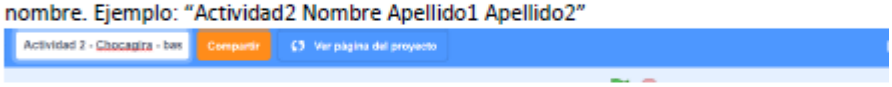
¿Qué bloques puedes usar para tomar la decisión de hacia dónde mover el robot para evitar que choque contra un objeto? ¿Recuerdas los bloques de tipo if-then?

6. Para empezar

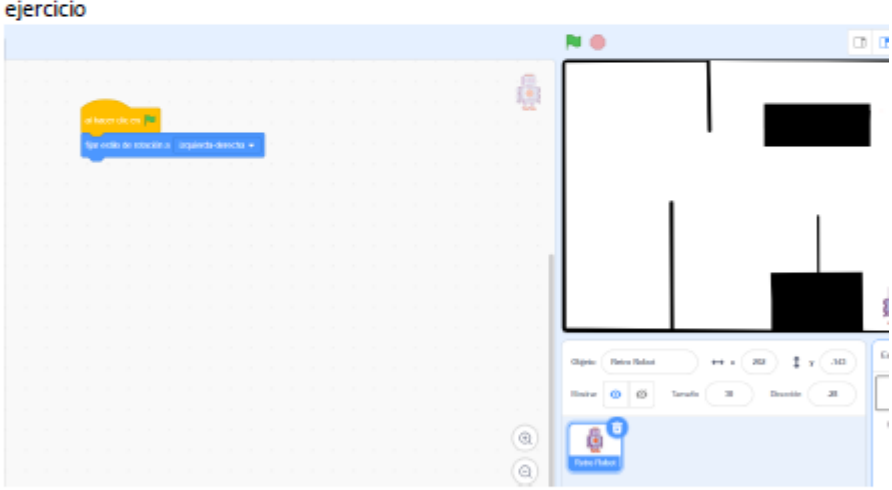
1. Accede a la web https://scratch.mit.edu/
2. Abre sesión con tu usuario seleccionando en el menú **Iniciar sesión**
3. Selecciona la opción "Cargar desde tu ordenador"



4. Establece el nombre de tu proyecto con el nombre Actividad2 y a continuación tu nombre. Ejemplo: "Actividad2 Nombre Apellido1 Apellido2"

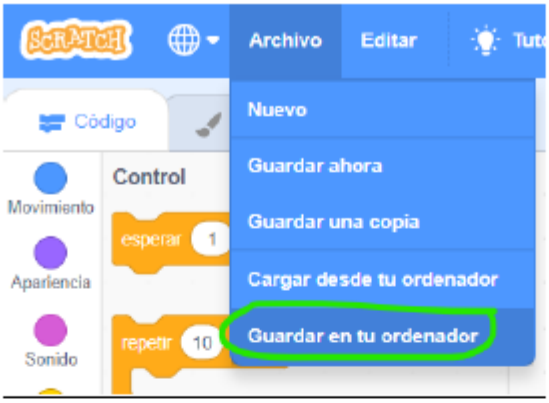


5. Ahora ya tienes cargado el proyecto con el personaje y el escenario a utilizar en el ejercicio



6. Ya tienes todo listo para empezar a programar tu robot con el movimiento choca-gira. ¡A por ello!
7. <u>Cuando termines el ejercicio recuerda:</u>
   a. <u>Guardar el proyecto en el ordenador</u>
   b. <u>Subir el proyecto a la tarea de Google Classroom</u>

   Pulsando la opción "Guardar en tu ordenador" se descarga tu proyecto de Scratch en el ordenador, normalmente dentro de la carpeta descargas, para que después puedas seleccionarlo y entregarlo en la actividad de Google Classroom.



c. <u>Subir una captura de pantalla de Scrath en la que se vean los bloques a la tarea de Google Classroom</u>
Para hacer la captura puedes usar la tecla "Imp pant" (Imprimir pantalla), luego pegar la captura de pantalla en el programa Paint, guardar la imagen con el nombre de tu proyecto "Actividad 2 – Nombre Apellido Apellido" y, por último, entregar la imagen en la tarea de Google Classroom.

7. Antes de terminar... ¿Sabías qué?

Además de para calcular la distancia a objetos los ultrasonidos son utilizados para otras muchas cosas:

- En las ecografías para generar imágenes que permiten "ver" dentro del cuerpo humano.
- Para detectar defectos internos en piezas, por ejemplo de alas de avión o vigas.
- Para limpiar instrumentos técnicos de bacterias y otros microorganismos.
- De forma similar a la empleada en los robots choca-gira, en los sensores de aparcamiento que tienen los coches.

6.5 Chocagira con Kibotics

Ejercicio ChocaGira Ultrasonidos

Tiempo de estudio: 2.5 horas

Dificultad:

En este ejercicio deberás programar nuestro robot **GoPiGo** para que sea capaz de deambular por una habitación sin chocarse con ningún obstáculo. El GoPiGo debe ser programado de forma que utilice sus *sensores* para obtener información del entorno continuamente, y sus *actuadores* para ejecutar las acciones necesarias para poder ir moviéndose por la habitación evitando los diferentes obstáculos, de acuerdo a la información obtenida por los sensores.

Concretamente, el programa debe hacer que el GoPiGo avance en línea recta hasta que se encuentre a poca distancia de chocar contra un obstáculo frontal. En ese momento, el programa debe hacer que el GoPiGo se detenga, retroceda durante unos instantes, gire a la izquierda una cantidad de grados *aleatoria*, y continúe avanzando en línea recta como al principio.

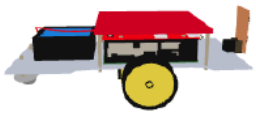
Contenido

1 - Qué vas a aprender

En esta unidad vas a aprender las bases del funcionamiento y el modo de uso de un sensor muy utilizado en robótica: el sensor de ultrasonidos.

También profundizaremos en los bloques de programación *if-else*. Usaremos este bloque para tomar un abanico de decisiones en función de las lecturas del sensor de ultrasonidos.

Por último programarás uno de los comportamientos más famosos de los robots, el comportamiento **choca-gira**.



2- Los sensores de ultrasonidos

Los sensores de ultrasonidos son unos dispositivos electrónicos que tienen dos componentes:

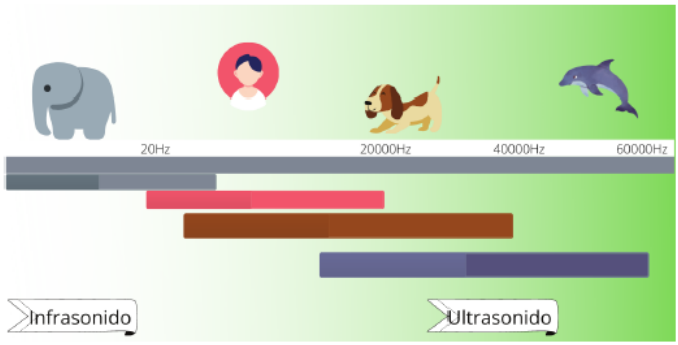
1. Un emisor de ultrasonidos, que se encargará de emitir un sonido que los humanos no somos capaces de oír (aunque sí otros animales).
2. Un receptor de ultrasonidos sensible a los ultrasonidos. Una "oreja" electrónica capaz de transformar ese sonido en una señal eléctrica que puede ser leída por un procesador.

2.1- Los ultrasonidos

El sonido se produce cuando una vibración que llega al oído. Por ejemplo la vibración que nuestas cuerdas vocales producen al hablar.

Todas las vibraciones producen esa perturbación pero nuestro oído sólo puede oír algunas de ellas. Podríamos decir que hay sonidos que los humanos no podemos oir.

Esos sonidos que son demasiado agudos para nuestros oídos son los **ultrasonidos**. Y aunque los humanos no podemos oirlos hay animales que sí pueden, como los perros o los delfines entre otros.



2.2- Usando los ultrasonidos como sensores

Ya sabemos qué son los ultrasonidos, ahora nos toca aprender cómo podemos usarlos en robótica.

Fijémonos en los murciélagos.

Los murciélagos para poder volar sin chocarse por la noche emiten ultrasonidos con su laringe. Nosotros no podemos oirlo pero ellos sí. Esos ultrasonidos que el murciélago emite por la boca abierta o la nariz rebota en los objetos que haya alrededor y el murciélago oye el eco de este ultrasonido. Ese eco le ayuda a saber donde están los obstáculos a su alrededor y así orientarse

Nuestros sensores de ultrasonidos funcionan de una forma muy parecida. Emiten un ultrasonido y reciben el eco. Sabiendo lo que ha tardado en regresar el eco se puede saber lo lejos que está el obstáculo en el que ha "rebotado" el ultrasonido.

0:00 / 0:09

3 - Sensores del GoPiGo a utilizar

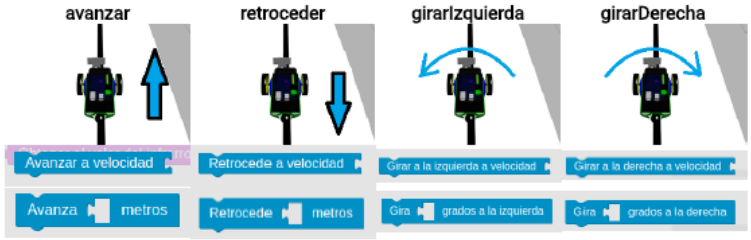
Vamos a usar el sensor de Ultrasonidos (abreviado US) que tiene el GoPiGo en su parte frontal. Este sensor es capaz de detectar la presencia de un obstáculo que se encuentre enfrente del GoPiGo por medio de la emisión/detección de ondas de ultrasonidos. Para acceder a los datos de este sensor, dirígete en el menú de trabajo al apartado **RobotAPI**, y dentro de este haz click sobre el subapartado **Sensors**, donde encontrarás bloques funcionales que te permitirán determinar la distancia a los obstáculos situados delante del robot.

Más información: Píldora sensor US

4 - Actuadores del GoPiGo a utilizar

Vamos a usar los motores que incorpora el GoPiGo (uno en cada rueda motriz) para hacer que se mueva según nuestras necesidades. Para ello, en el el menú de trabajo vas a encontrar los bloques necesarios accediendo a **RobotAPI → Motors**.

Los comportamientos que puedes conseguir en el GoPiGo mediante los bloques de código de este apartado son los que se muestran en la siguiente tabla:



5 - Pistas para resolver el ejercicio

- Debes hacer que tu programa capture la información del sensor de ultrasonidos y, en función de esa información, realizar acciones utilizando los actuadores para que el GoPiGo se comporte de la forma deseada
- El punto anterior no debe hacerse una sola vez, sino de forma continua (varias veces por segundo) para que el GoPiGo pueda ir moviéndose por la habitación sin chocar con ningún obstáculo.
- Cuando el GoPiGo reciba una orden de avanzar, retroceder o girar, continuará ejecutando dicha orden hasta que reciba otra orden (si la orden es la misma, continuará haciendo lo mismo que estaba haciendo). Ayúdate de los bloques de ejecución temporal o los bucles de código si lo ves necesario.

6 - Toma de decisiones en Scratch

Para poder decidir en cada momento lo que el robot tiene que hacer en función de la lectura del sensor de ultrasonidos vas a necesitar unos bloques de programación básicos los bloques *if-then*.

En el ejemplo anterior tras leer el sensor de ultrasonidos si la lectura es menor de 0.5m el robot parará, en caso contrario el robot avanza.

Sin embargo es posible que necesites tomar decisiones algo más elaboradas, donde haya varias opciones posibles. En ese caso necesitamos ampliar el bloque `if-else`.

0:00 / 2:11

¿Puedes conseguir que el GoPiGo se mueva por la habitación sin chocarse con los obstáculos? Inténtalo, ¡es divertido!

7 - Recopilando

¡Enhorabuena! Has programado uno de los comportamientos más famosos en el mundo de la robótica el choca-gira. Y no sólo es famoso sino también importante porque permite que nuestro robot se mueva de manera segura, sin chocarse. Para conseguirlo has tenido que aprender:

- Qué es un sensor de ultrasonidos y cuál es el principio físico que hay detrás.
- El bloque para leer el sensor de ultrasonidos en Kibotics.
- Qué bloques tiene Kibotics para controlar el movimiento del robot.
- Cómo usar los bloques dedecisiones.

8 - ¿Sabías que...?

Los sensores de ultrasonidos no se usan únicamente para detectar distancias a objetos. También se usan para:

- En maquinaria agrícola para controlar la posición de las herramientas del tractor.
- Los sensores de ayuda al aparcamiento en los coches son sensores de ultrasonidos (en este caso su uso es similar al que habéis programado en esta unidad).
- Sensores de ultrasonidos muy sensibles permiten detectar defectos internos (grietas y otros fallos) en piezas, por ejemplo alas de avión o vigas.
- Gracias a los ultrasonidos también podemos "ver" dentro del cuerpo humano mediante las ecografías.

9- Vídeo de ejemplo

Derechos de autor © - Asociación JdeRobot. Este sitio utiliza cookies. Más información (/terms#cookies). Versión 3.1.0

6.6 Siguelíneas con Scratch

Actividad 3: Siguelíneas con Scratch

1. Introducción

Este proyecto consiste en conseguir que el personaje, se desplace por el circuito sin salirse de la carretera.

Para completar el ejercicio es necesario usar bloques de sensores que permitan al personaje identificar que sigue dentro de la carretera.

El robot debe moverse en línea recta sin detenerse hasta que identifique que se está saliendo de la carretera porque toca el arcén y tenga que cambiar su dirección para continuar circulando por la carretera.

2. Qué vas a aprender en esta unidad

El objetivo de esta unidad es simular con Scratch la funcionalidad del sensor de infrarrojos IR muy usado en robótica para la programación de robots siguelíneas como el GoPiGo. ¿Se te ocurre qué sensor de Scratch puede ayudarte a reconocer que el robot sigue la línea negra igual que lo haría uno de infrarrojos? Para programarlo a partir de la información del sensor deberás tomar decisiones de cambio de dirección.

La construcción y programación de robots siguelíneas es tan común en el aprendizaje de robótica que hay gran número de competiciones para estudiantes en las que participar.

3. Los sensores de infrarrojos

Además de ser uno de los más usados en robótica para detectar obstáculos cercanos o seguir líneas, también es muy habitual encontrarlo en nuestra vida cotidiana. Los siguientes son algunos de los usos en nuestro día a día:

- En las puertas de los ascensores o de los garajes, para que no se cierren si alguien o un coche está pasando.
- En las puertas automáticas, que se abren si alguien se acerca.
- En el mando de la televisión, que nos permite cambiar de canal.

¿Cómo funciona el sensor de infrarrojos?

El sensor de infrarrojos es un dispositivo capaz de detectar un tipo de luz, la luz infrarroja, que el ojo humano no es capaz de ver.

Un sensor de infrarrojos suele estar formado por dos partes:

- Emisor: Un led que emite la luz infrarroja.
- Receptor: Un sensor de luz que al detectar la luz infrarroja reacciona y genera corriente eléctrica.

El emisor está constantemente emitiendo luz infrarroja. Si esa luz "rebota" contra alguna superficie el detector la recibe y envía una señal eléctrica. Así es cómo podemos detectar obstáculos.

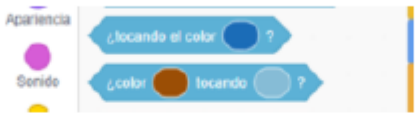
Detección del color negro

Uno de los usos más habituales de los sensores infrarrojos es la detección del color negro.

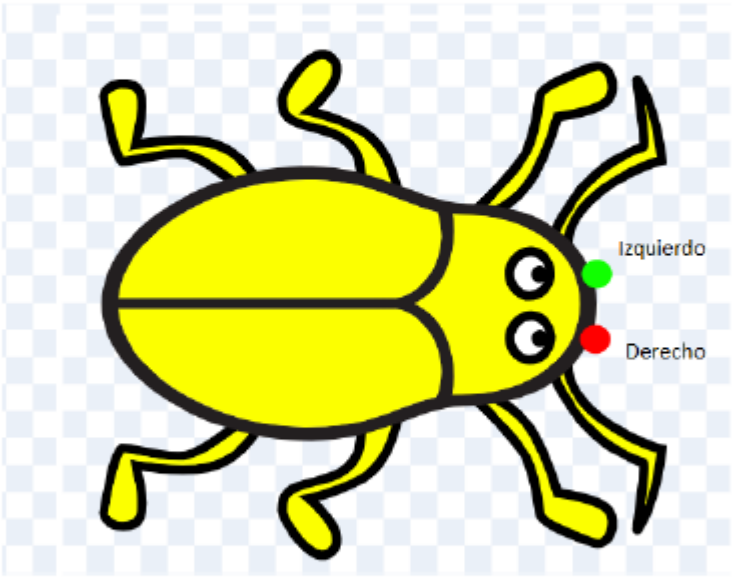
El motivo es que cuando la luz infrarroja impacta contra objetos de color negro, igual que ocurre con otros tipos de luz, esta es absorbida por completamente por el objeto y no rebota nada y, por lo tanto, el receptor del sensor no recibe luz alguna. Si se considera que los objetos serán de color distinto al negro, puede interpretarse que si no se recibe luz no hay objeto y a la inversa.

4. Sensores y bloques a utilizar

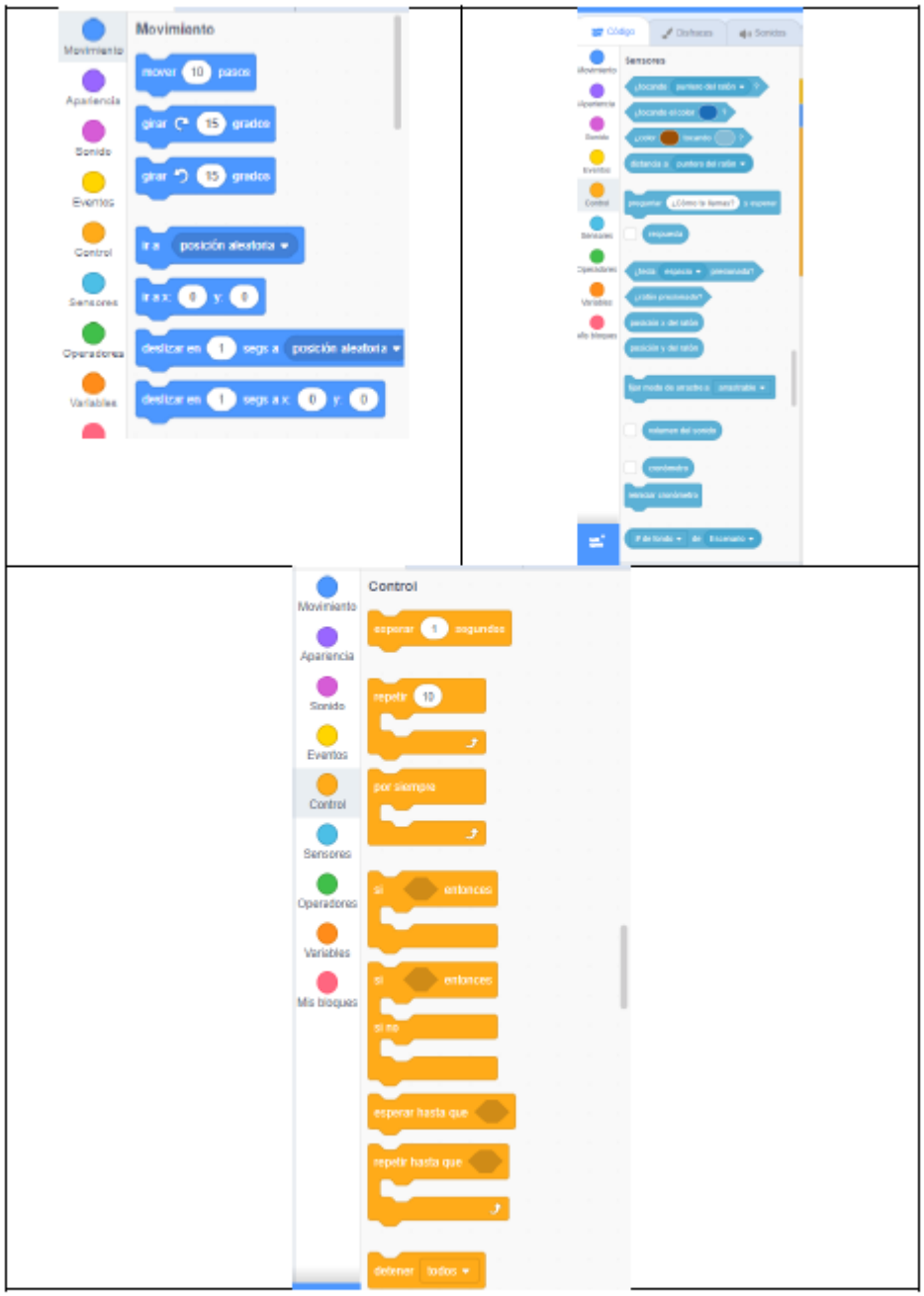
Los personajes de Scratch disponen de sensores que permite identificar el contacto de nuestro personaje con otros otros colores. Estos sensores son la herramienta ideal para simular el comportamiento del sensor de infrarrojos.



El personaje a usar en el ejercicio dispone de dos "sensores" (en la imagen "izquierdo" y "derecho") de este tipo a usar en su parte delantera.



Este ejercicio requerirá que utilices principalmente los bloques de sensores, de control y de movimiento para mover el personaje sobre la carretera: mover, girar, "ir a x: y:", "apuntar en dirección"... Estos bloques puedes encontrarlos en la paleta de bloques, accediendo los apartados: Sensores, Control y Movimiento.



5. Algunas pistas

- Estable la posición inicial nada más empezar el programa.
- Además de la posición, debes tener en cuenta que el personaje apunte en la dirección correcta para que al moverse avance por la carretera.
- ¿Qué bloques se utilizan para que el robot repita instrucciones de forma continua?

- En función de los valores que se obtengan en los sensores se deberá girar para un lado u otro con el objetivo de no salirse del camino. Leyendo la siguiente tabla sabrás la lógica a emplear en función de la lectura que reciba el personaje en los sensores.

lectura sensor izquierdo	lectura sensor derecho	Acción (Izquierda / Derecha)
Verde tocando negro	Rojo tocando negro	Mover
Verde tocando negro	Rojo tocando otro	Izquierda
Verde tocando otro	Rojo tocando negro	Derecha
Verde tocando otro	Rojo tocando otro	Detener todos

6. Algunas pistas

Antes de comenzar el algoritmo piensa lo siguiente:

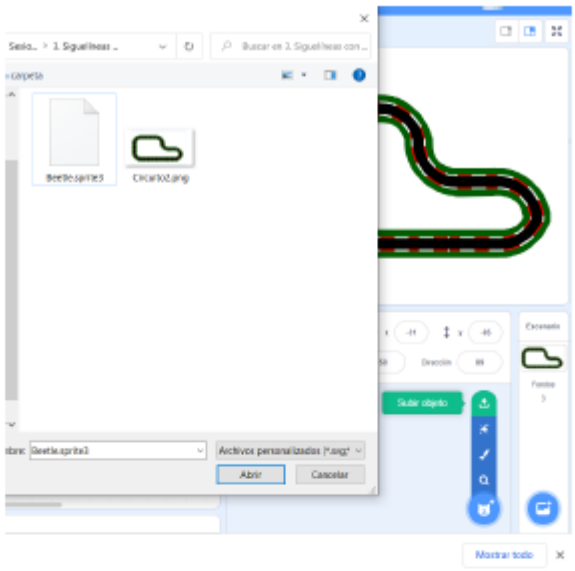
¿Qué bloques de movimiento te serán más útiles para programar el robot?

¿Qué bloques se utilizan para que el robot repita instrucciones de forma continua?

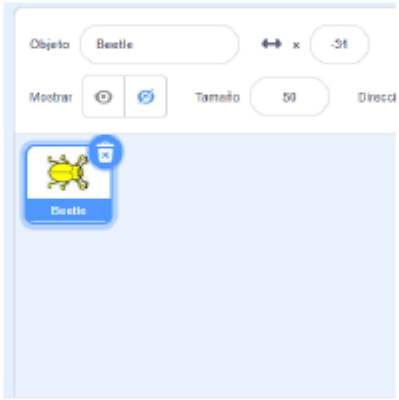
¿Qué bloques puedes usar para tomar la decisión de hacia dónde mover el robot para evitar que choque contra un objeto? ¿Recuerdas los bloques de tipo if-then?

7. Para empezar

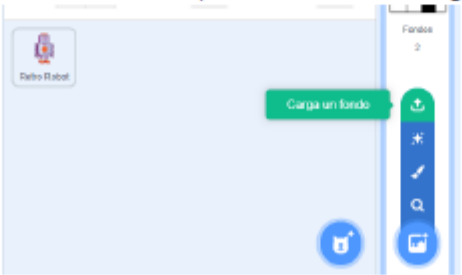
1. Accede a la web https://scratch.mit.edu/
2. Abre sesión con tu usuario seleccionando en el menú **Iniciar sesión**
3. Selecciona la opción "Crear" para crear un nuevo proyecto
4. Estable el nombre de tu proyecto con el nombre Actividad3 y a continuación tu nombre. Ejemplo: "Actividad3 Nombre Apellido1 Apellido2"
5. Elimina el personaje por defecto y añade el del archivo "Beetle.sprite3"



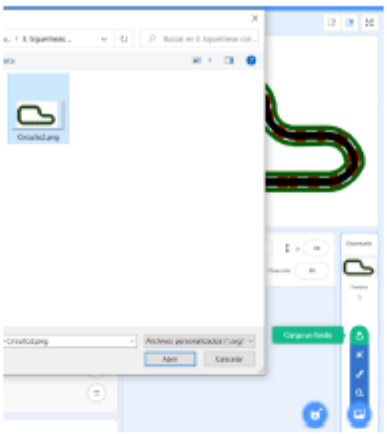
Recuerda cambiar su tamaño a 50%



6. Ahora añade el fondo de la habitación "Circuito2.png":
   a. En la zona Escenario, selecciona la acción "Cargar un fondo"



   b. En el menú que permite elegir un archivo, selecciona el archivo "Circuito.png" incluido en la actividad y pulsa Abrir.



7. Ya tienes todo listo para empezar a programar tu robot escarabajo. ¡A por ello!
8. Cuando termines el ejercicio recuerda:
    a. Guardar el proyecto en el ordenador
    b. Subir el proyecto a Google Classroom junto con una captura de pantalla de Scrath en la que se vean los bloques

6.7 Siguelíneas con Kibotics

Ejercicio Siguelínea con sensor de Infrarrojos

Tiempo de estudio | Dificultad

4 horas

En este ejercicio deberás programar nuestro robot **GoPiGo** para que sea capaz de seguir una línea negra pintada sobre un suelo blanco. El GoPiGo debe ser programado de forma que utilice sus *sensores* para obtener información del entorno continuamente, y sus *actuadores* para ejecutar las acciones necesarias para poder ir siguiendo el trazado de la línea de acuerdo a la información obtenida por los sensores.

Contenido

1 -Qué vas a aprender en esta unidad

En este ejercicio deberás programar nuestro robot **GoPiGo** para que sea capaz de seguir una línea negra pintada sobre un suelo blanco. El GoPiGo debe ser programado de forma que utilice sus *sensores* para obtener información del entorno continuamente, y sus *actuadores* para ejecutar las acciones necesarias para poder ir siguiendo el trazado de la línea de acuerdo a la información obtenida por los sensores.

Para lograr este objetivo vas a necesitar aprender:

1. Qué son y cómo funcionan los sensores de infrarrojos

2. Cuáles son los bloques que necesitamos para leer los sensores infrarrojos en Kibotics
3. Cómo podemos programar al robot para que "decida" qué acción realizar.

2. Los sensores de infrarrojos

Los sensores de infrarrojos son unos sensores imprescindibles en robótica. Además también vamos a poder encontrarlos en nuestro día a día.

2.1- ¿Qué es un sensor de infrarrojos?

Un sensor de infrarrojos es un detector de un tipo de luz que nuestro ojo no es capaz de ver que es la luz infrarroja. Ya sabréis que la luz blanca del sol que las personas somos capaces de ver, está compuesta de diferentes colores: los del arcoiris. Pero además de los colores del arcoiris, hay otro tipo de "luz" que no podemos ver, pero que el sol también emite. Y una de esas "luces" es la infrarroja.

Un sensor de infrarrojos del tipo de los que usamos en robótica suele estar formado por dos partes:

1. Un LED que emite una luz infrarroja. Vamos a llamarlo emisor.
2. El sensor propiamente dicho, que al detectar luz infrarroja genera una corriente eléctrica. Lo llamaremos receptor.

El sensor funciona de la siguiente manera. El emisor está constamente emitiendo luz infrarroja. Si esa luz "rebota" contra alguna superfície el detector la recibe y envía una señal eléctrica. Así es cómo podemos detectar obstáculos.

0:00 / 0:12

2.2- Uso de los sensores infrarrojos

Los sensores de infrarrojos son muy baratos y sencillos de usar. ¡Están por todas partes!:

- En las puertas de los ascensores, para que no se cierren si alguien está pasando.
- En las puertas automáticas, que se abren si alguien se acerca.
- En el mando de la televisión, que nos permite cambiar de canal.
- En los robots para detectar obstáculos muy próximos.
- En la base de los robots para poder seguir una línea negra.

2.3- Detectando el color negro

Uno de los usos más típicos de los sensores infrarrojos en robótica es la detección del color negro. Vamos a entender por qué.

Como decíamos anteriormente el detector de luz infrarroja detecta la luz infrarroja que emitió el emisor y que ha rebotado en un objeto. Cuando ese objeto es de color negro, esa luz infrarroja emitida es absorbida completamente por el objeto negro y por lo tanto no rebota nada y nuestro receptor no recibe luz. Para el sensor infrarrojo un objeto negro es equivalente a que no haya objeto.

Puede parecernos un fenómeno extraño, sin embargo estamos acostumbrados a ello. Pensad en qué ocurre cuando en verano bajo el sol abrasador nos vestimos de negro, nuestra ropa absorbe todo el calor. No es una idea refrescante vestirse de negro en el calor del verano.

Pues de la misma manera que nuestra ropa negra absorbe el calor del sol, la línea negra absorbe la radiación infrarroja y hace que el receptor de nuestro sensor no detecte nada.

3 - Sensores del GoPiGo a utilizar

Vamos a usar los 2 sensores de Infrarrojos (abreviado IR) que tiene el GoPiGo en su parte frontal apuntando al suelo, uno en la parte izquierda y otro en la parte derecha. Estos sensores son capaces de detectar si el suelo es oscuro (negro) o claro (blanco). El bloque de Scratch que te permite "leer" el valor de los sensores de IR es el siguiente:

Este bloque te va a dar 4 valores posibles:

Valor	Significado	Situación
0	Los dos sensores detectan negro. El robot está sobre la línea negra.	

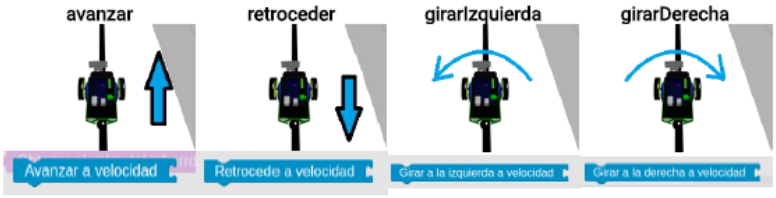
1	La línea negra se encuentra a la izquierda del robot
2	La línea negra se encuentra a la derecha del robot
3	El robot está completamente fuera de la línea negra

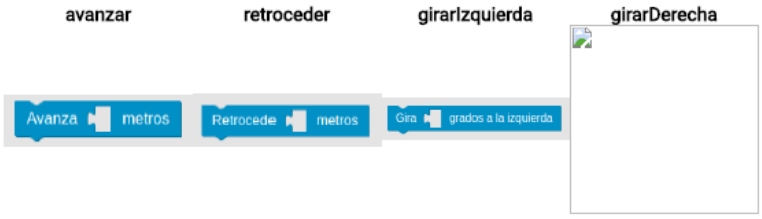
Más información: Píldora sensor IR

4 - Actuadores del GoPiGo a utilizar

Vamos a usar los motores que incorpora el GoPiGo (uno en cada rueda motriz) para hacer que se mueva según nuestras necesidades. Para ello, en el el menú de trabajo vas a encontrar los bloques necesarios accediendo a **RobotAPI → Motors**.

Los comportamientos que puedes conseguir en el GoPiGo mediante los bloques de código de este apartado son los que se muestran en la siguiente tabla:





5 - Programar una decisión

Uno de los elementos básicos de cualquier lenguaje de programación es la posibilidad de tomar una decisión en función de un valor que se calcula. Para ello tenemos los siguientes bloques:

0:00 / 0:45

6 - Pistas para resolver el ejercicio

- Debes hacer que tu programa capture la información de los sensores y, en función de esa información, realizar acciones utilizando los actuadores para que el GoPiGo se comporte de la forma deseada

- Cuando el GoPiGo reciba una orden de avanzar, retroceder o girar, continuará ejecutando dicha orden hasta que reciba otra orden (si la orden es la misma, continuará haciendo lo mismo que estaba haciendo)

¿Puedes conseguir que el GoPiGo siga la línea negra? Inténtalo, ¡es divertido!

7 - Recopilando

Al terminar esta unidad ¡¡habrás programado uno de los comportamientos básicos de un robot: un siguelíneas!! Y habrás aprendido a trabajar con los sensores de infrarrojos y a programar tomas de decisiones en scratch.

8 - ¿Sabías que...?

Nuestro ojo no puede percibir la luz infrarroja. Sin embargo las cámaras de los móviles sí suelen ser sensibles al infrarrojo. Si enfocamos con la cámara del móvil un sensor de infrarrojo (que tenga emisor, claro) podemos ver en la imagen lo que ve el emisor y la luz infrarroja, que la cámara del móvil *pinta* en color morado habitualmente.

Haz la prueba y verás. Es un truco infalible para saber si el mando a distancia de la televisión tiene pilas o no.

9- Vídeo de ejemplo

6.8 Recoge confeti con Scratch

Actividades con Scratch — Actividad 4: Recoge confeti con Scratch

Actividad 4: Recoge confeti con Scratch

1. Introducción

Este proyecto consiste en conseguir que el personaje, un robot, se desplace por la habitación sorteando los obstáculos y recogiendo el mayor número de confeti que pueda.

Para completar el ejercicio es necesario usar bloques de sensores que permitan al robot identificar que hay un obstáculo y actuar cambiando de dirección.

El robot debe moverse en línea recta por la habitación libremente sin detenerse hasta que identifique un obstáculo y tenga que cambiar su dirección para continuar moviéndose por la habitación.

2. Qué vas a aprender en esta unidad

En esta unidad vas a programar la lógica de un algoritmo de navegación del estilo al que emplea una aspiradora autónoma.

3. Los algoritmos de cobertura

Los algoritmos de cobertura son una importante línea de investigación en la planificación de rutas para robótica. El objetivo de estos algoritmos es pasar por todas las posiciones accesibles para el robot de la forma más eficiente posible.

En este ejercicio utilizaremos un algoritmo de cobertura muy sencillo llamado Exploración aleatoria.

Los algoritmos de cobertura pueden dividirse en dos categorías:

1. Cobertura sin conexión
   Se utiliza información fija y se conoce el entorno previamente. Ejemplos: algoritmos genéticos, redes neuronales...
2. Cobertura en línea
   Utiliza información que se obtiene en tiempo real. Para ello hace uso de sensores.

En nuestro caso haremos uso del algoritmo de cobertura en línea, pues haremos uso de los sensores del robot para cubrir la mayor superficie accesible por el robot posible.

Para el algoritmo se pueden usar movimientos base como los siguientes:

- Movimiento en espiral: El robot realiza círculos o cuadrados crecientes.
- Movimiento Boustrophedon: El robot se mueve en forma de S.
- Descomposición ambiental: Se divide la superficie en partes más pequeñas para luego irlas recorriendo.
- Dirección de barrido: Consiste en optimizar las rutas generadas para cada subregión ajustando la duración, la velocidad y la dirección de cada barrido.
- Retroceso óptimo: Se planifica cuando pasar de una subregión a otra hasta que no queda ningún punto para dar marcha atrás.

4. Pistas

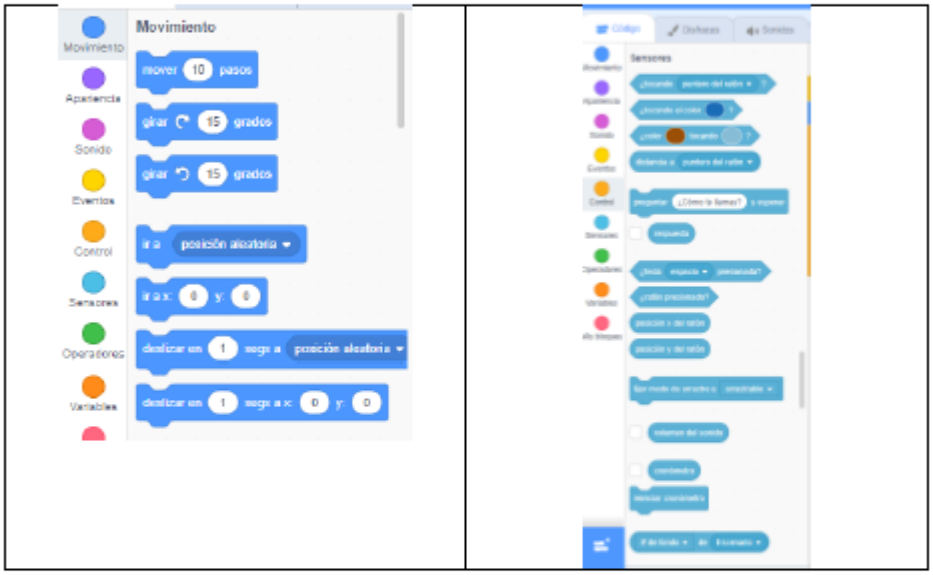
Tomar como base el algoritmo que utilizaste en la actividad 2 chocagira con Scratch y añadir mejoras al algoritmo para que recoja el mayor confeti posible en el menor número de tiempo.

Página 1 | 5

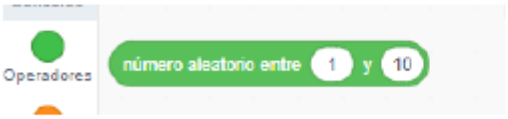
Generación de ángulos aleatorios: generación de ángulos aleatorios que nos ayuden a cubrir la mayor superficie posible puede ser una primera aproximación.

5. Bloques a utilizar

Además del sensor de Scratch que te permita identificar la proximidad de un objeto al personaje en el escenario, este ejercicio requerirá que utilices principalmente los bloques de control y del área de trabajo que te permiten enviar instrucciones de movimiento al personaje para avanzar, retroceder y/o girar. Estos bloques puedes encontrarlos en la paleta de bloques, accediendo los apartados: Sensores, Control y Movimiento.



Para la generación de ángulos aleatorios puede resultarte útil usar el bloque que genera valores aleatorios contenido en el grupo Operadores.



6. ¿Cómo lo hacen los robots como Roomba?

En robots con motores que permiten definir la velocidad lineal y angular (velocidad de giro) se aplican algoritmos con cálculos algo más complejos.

Generación de ángulos aleatorios: Se generan de dos formas:

- Duración aleatoria: al mantener la velocidad angular fija W, la duración del turno puede ser aleatoria para apuntar el robot hacia una dirección aleatoria.
- Ángulo aleatorio: Se genera un ángulo aleatorio y luego se gira hacia él.

Movimiento Dash:

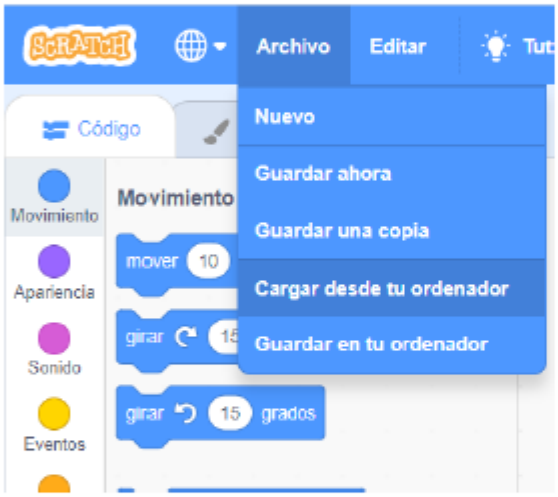
Una vez decidida la dirección a partir del ángulo calculado, se mueve el robot en esa dirección hasta que se detecte una colisión.

Movimiento en espiral:

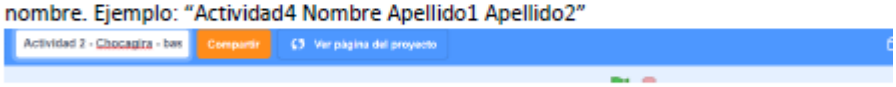
Usando la fórmula física v=r·ω. (velocidad lineal = radio * velocidad angular)

7. Para empezar

1. Accede a la tarea en Google Classroom y descarga en tu ordenar el archivo "Actividad 4 - Recoge confeti base.sb3"
2. Ahora, accede a la web https://scratch.mit.edu/
3. Abre sesión con tu usuario seleccionando en el menú **Iniciar sesión**
4. Selecciona la opción "**Cargar desde tu ordenador**" y selecciona el proyecto "Actividad 4 - Recoge confeti base.sb3" guardado previamente en tu ordenador.

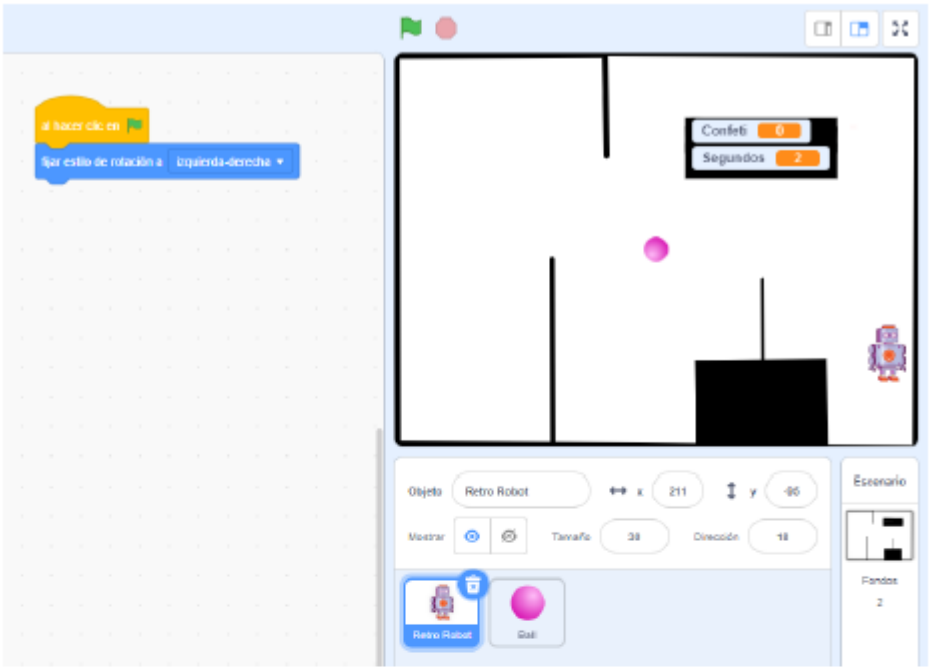


5. Establece el nombre de tu proyecto con el nombre Actividad4 y, a continuación, tu nombre. Ejemplo: "Actividad4 Nombre Apellido1 Apellido2"



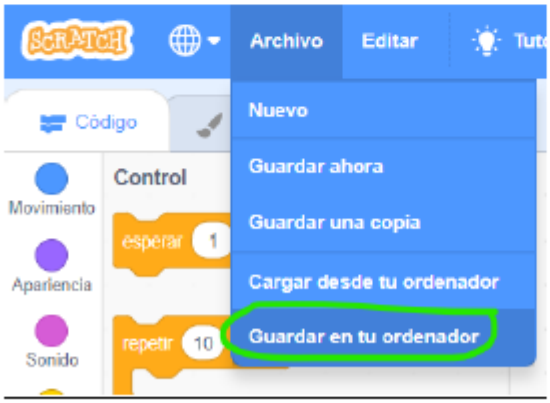
6. Ahora ya tienes cargado el proyecto con el personaje y el escenario a utilizar en el ejercicio.

Actividades con Scratch — Actividad 4: Recoge confeti con Scratch



7. Ya tienes todo listo para empezar a programar tu robot con el movimiento choca-gira. ¡A por ello!
8. Cuando termines el ejercicio recuerda:
   a. Guardar el proyecto en el ordenador
   b. Subir el proyecto a la tarea de Google Classroom

   Pulsando la opción "Guardar en tu ordenador" se descarga tu proyecto de Scratch en el ordenador, normalmente dentro de la carpeta descargas, para que después puedas seleccionarlo y entregarlo en la actividad de Google Classroom.



   c. Subir una captura de pantalla de Scrath en la que se vean los bloques a la tarea de Google Classroom

   Para hacer la captura puedes usar la tecla "Imp pant" (Imprimir pantalla), luego pegar la captura de pantalla en el programa Paint, guardar la imagen con

el nombre de tu proyecto "Actividad 2 – Nombre Apellido Apellido" y, por último, entregar la imagen en la tarea de Google Classroom.

8. Antes de terminar... ¿Sabías qué?

Además de para calcular la distancia a objetos los ultrasonidos son utilizados para otras muchas cosas:

- En las ecografías para generar imágenes que permiten "ver" dentro del cuerpo humano.
- Para detectar defectos internos en piezas, por ejemplo de alas de avión o vigas.
- Para limpiar instrumentos técnicos de bacterias y otros microorganismos.
- De forma similar a la empleada en los robots choca-gira, en los sensores de aparcamiento que tienen los coches.

6.9 Recoge confeti con Kibotics

Aspirador Robótico 'Roomba' en Scratch

Tiempo de estudio | Dificultad

3 horas

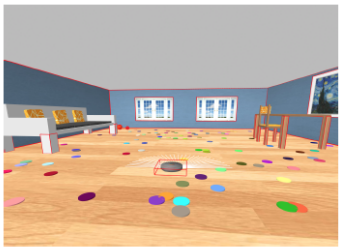
En este ejercicio tendrás que programar un robot aspiradora, más conocido por todos como 'Roomba'.

Hacieno uso del HAL API utilizando sensores y actuadores, el objetivo de este ejercicio es recoger el mayor número de confeti en 5 minutos. ¿Podrás recogerlos todos?

Roomba atrapa confeti

Para completar este ejercicio, deberás conseguir que tu robot aspire el mayor número de confeti posible, muevete por la habitación recolectando todos los círculos de colores, ¡cuidado con los obstáculos de la habitación!

Contenido



1 - Qué vas a aprender en esta unidad

En esta unidad vas a aprender a programar la lógica de un algoritmo de navegación para una aspiradora autónoma.

2 - Requisitos de la práctica.

Se pide que implementes un algoritmo, que permita al robot aspirador recoger el confeti sin chocarse con los obstaculos utilizando HAL API (sensores y actuadores)

3- Teoria

El objetivo principal es recoger el mayor numero de confeti cubriendo la mayor superficie de la habitación. Primero, entendamos qué son los algoritmos de cobertura.

3.1- Algoritmos de cobertura

La planificación de rutas de cobertura es un área importante de investigación en la planificación de rutas para robótica, que implica encontrar una ruta que pase por cada posición accesible en su entorno. En este ejercicio, utilizamos un algoritmo de cobertura muy básico llamado Exploración aleatoria.

Clasificación

Los algoritmos de cobertura se dividen en dos categorías:

La cobertura ***sin conexión*** : utiliza información fija y el entorno se conoce de antemano. Algoritmos genéticos, redes neuronales, descomposición celular, árboles de expansión son algunos ejemplos.

La cobertura ***en línea*** : utiliza medidas y decisiones en tiempo real para cubrir toda el área. El enfoque basado en sensores (nuestro caso).

Para implementar un algoritmo de cobertura podemos usar un movimiento base como los siguientes:

Movimiento en espiral: El robot sigue un patrón de círculo / cuadrado creciente.

Movimiento Boustrophedon: El robot sigue un patrón en forma de S.

Descomposición ambiental: Esto implica dividir el área en partes más pequeñas.

Dirección de barrido: Esto influye en la optimización de las rutas generadas para cada subregión ajustando la duración, la velocidad y la dirección de cada barrido.

Retroceso óptimo: Esto implica el plan para pasar de una pequeña subregión a otra. Se dice que la cobertura está completa cuando no queda ningún punto para dar marcha atrás.

4- Pistas

Generación de ángulos aleatorios: La tarea más importante es la generación de un ángulo aleatorio. Hay 2 formas de lograrlo.

Duración aleatoria : al mantener la velocidad angular fija W , la duración del turno puede ser aleatoria para apuntar el robot hacia una dirección aleatoria.

Ángulo aleatorio : este método requiere cálculo. Generamos un ángulo aleatorio y luego giramos hacia él. Aproximadamente una velocidad angular de 3 hace girar el robot 90 grados.

Movimiento Dash:

Una vez decidida la dirección, nos movemos en esa dirección. Esta es la parte más simple, tenemos que enviar un comando de velocidad al robot, hasta que se detecte una colisión.

Movimiento en espiral:

Usando la fórmula física $v=r\cdot\omega$.

No olvides usar time.sleep() y HAL

4- Pistas

Generación de ángulos aleatorios La tarea más importante es la generación de un ángulo aleatorio. Hay 2 formas de lograrlo.

Duración aleatoria: al mantener la velocidad angular fija W , la duración del turno puede ser aleatoria para apuntar el robot hacia una dirección aleatoria.

Ángulo aleatorio: este método requiere cálculo. Generamos un ángulo aleatorio y luego giramos hacia él. Aproximadamente una velocidad angular de 3 hace girar el robot 90 grados.

Movimiento Dash: Una vez decidida la dirección, nos movemos en esa dirección. Esta es la parte más simple, tenemos que enviar un comando de velocidad al robot, hasta que se detecte una colisión.

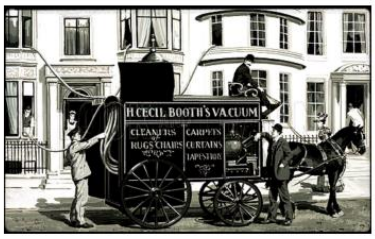
Movimiento en espiral: Usando la fórmula física $v=r\cdot\omega$. No olvides usar time.sleep() y HAL

Ayudaté también del sensor de ultrasonidos para detectar los obstáculos.

Más información: Píldora sensor US

5-¿Sabías que...?

El primer aspirador mecánico fue creado por Huber Booth en 1901 y era tan grande que para trasladarlo de una casa a otra, tenía que ser arrastrado en un carro por caballos. Los tubos de succión se introducían en las viviendas por las puertas y las ventanas. Mucho más tarde aparecieron las aspiradoras eléctricas con y sin filtros para el uso doméstico. En estos últimos años han aparecido las aspiradoras autónomas para ahorrarnos mucho tiempo. iRobot lanzó el primer Roomba en 2002, estos robots usan sensores táctiles, ópticos y acústicos, actualmente no solo aspiran la suciedad sino que algunos también friegan y purifican el suelo. Sin duda la aspiradora siempre ha sido un gran invento para cuidar la salud de las personas.

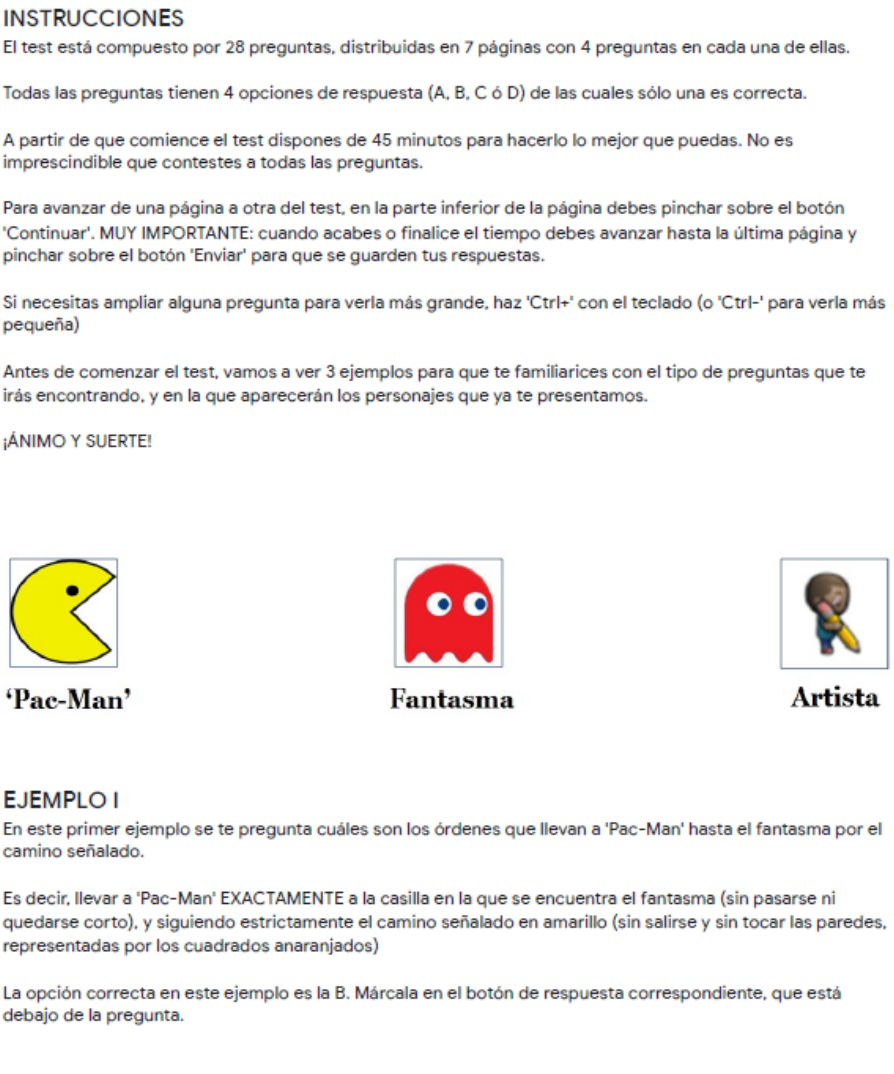


6- Vídeo de ejemplo

6.10 Test de Pensamiento Computacional

El presente anexo recoge un extracto de las preguntas del "Test de Pensamiento Computacional' con el objetivo de ilustrar sus diferentes tipologías y conceptos medidos con las mismas.

El test viene acompañado de instrucciones antes del test en las que se muestran ejemplos de los tipos de preguntas y respuestas, con el objetivo de que los participantes tengan claro la dinámica de las preguntas (Figura 29).



INSTRUCCIONES

El test está compuesto por 28 preguntas, distribuidas en 7 páginas con 4 preguntas en cada una de ellas.

Todas las preguntas tienen 4 opciones de respuesta (A, B, C ó D) de las cuales sólo una es correcta.

A partir de que comience el test dispones de 45 minutos para hacerlo lo mejor que puedas. No es imprescindible que contestes a todas las preguntas.

Para avanzar de una página a otra del test, en la parte inferior de la página debes pinchar sobre el botón 'Continuar'. MUY IMPORTANTE: cuando acabes o finalice el tiempo debes avanzar hasta la última página y pinchar sobre el botón 'Enviar' para que se guarden tus respuestas.

Si necesitas ampliar alguna pregunta para verla más grande, haz 'Ctrl+' con el teclado (o 'Ctrl-' para verla más pequeña)

Antes de comenzar el test, vamos a ver 3 ejemplos para que te familiarices con el tipo de preguntas que te irás encontrando, y en la que aparecerán los personajes que ya te presentamos.

¡ÁNIMO Y SUERTE!

'Pac-Man'

Fantasma

Artista

EJEMPLO I

En este primer ejemplo se te pregunta cuáles son los órdenes que llevan a 'Pac-Man' hasta el fantasma por el camino señalado.

Es decir, llevar a 'Pac-Man' EXACTAMENTE a la casilla en la que se encuentra el fantasma (sin pasarse ni quedarse corto), y siguiendo estrictamente el camino señalado en amarillo (sin salirse y sin tocar las paredes, representadas por los cuadrados anaranjados)

La opción correcta en este ejemplo es la B. Márcala en el botón de respuesta correspondiente, que está debajo de la pregunta.

***Figura 29**. Cuestionario. Instrucciones. Fuente: Román-González (2016)*

La Figura 30 muestra un ejemplo de pregunta en el entorno de interfaz Laberinto, con estilo de alternativas de respuesta de tipo Visual por Flechas, concepto computacional Direcciones, sin anidamiento de conceptos y con tarea cognitiva requerida tipo Secuenciación.

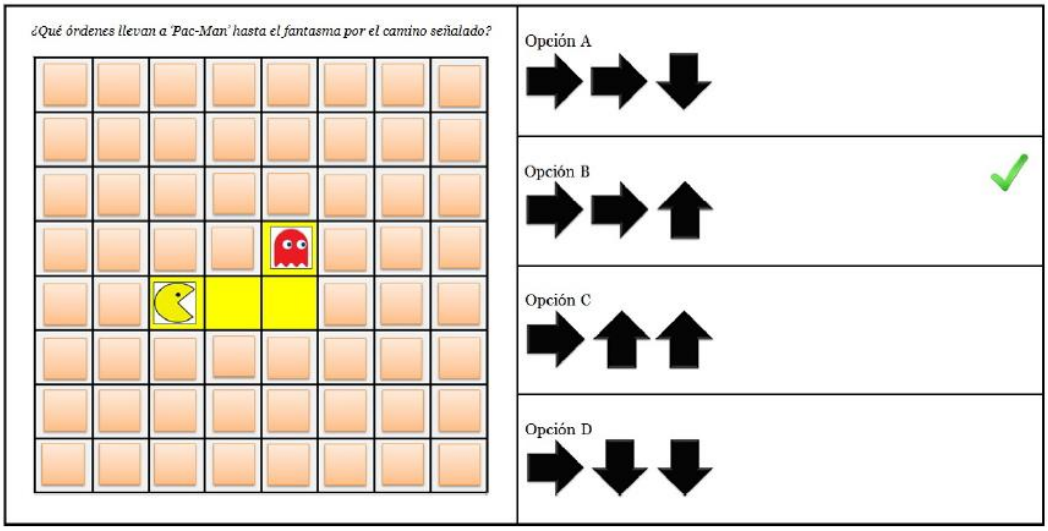


Figura 30. *Cuestionario. Pregunta direcciones en laberinto con flechas (Secuenciación).*
Fuente: Román-González (2016)

La Figura 31 muestra un ejemplo de pregunta también en el entorno de interfaz Laberinto, con estilo de alternativas de respuesta de tipo Visual por Bloques, concepto computacional Direcciones, sin anidamiento de conceptos y con tarea cognitiva requerida tipo Secuenciación.

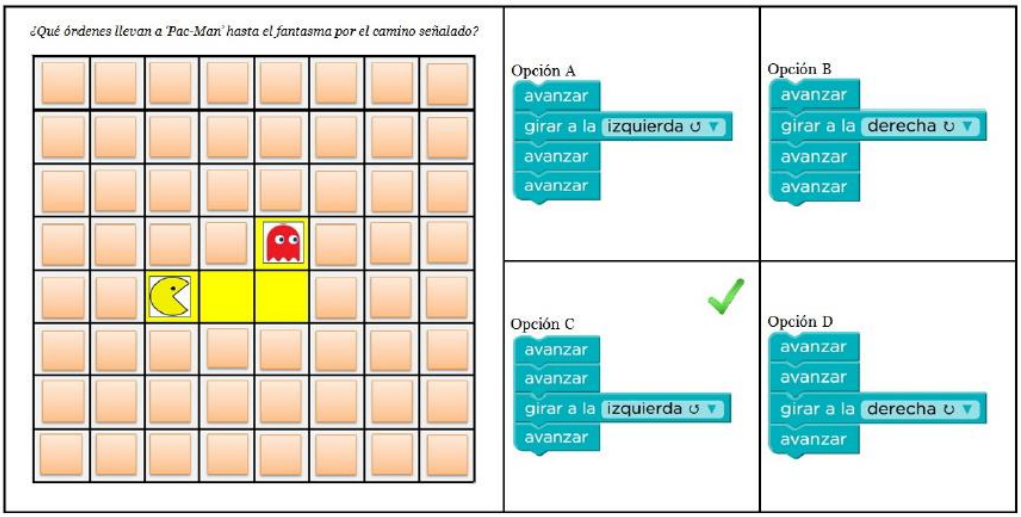


Figura 31. *Cuestionario. Pregunta direcciones en laberinto con bloques (Secuenciación).*
Fuente: Román-González (2016)

La Figura 32 muestra un ejemplo de pregunta en el entorno de interfaz Lienzo, con estilo de alternativas de respuesta de tipo Visual por Bloques, concepto computacional Direcciones, sin anidamiento de conceptos y con tarea cognitiva requerida tipo Secuenciación.

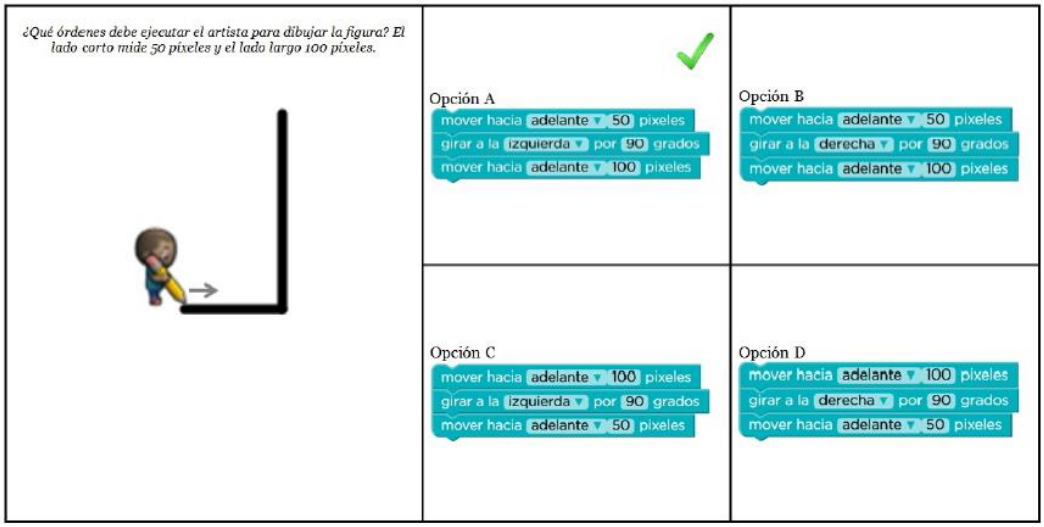


***Figura 32**. Cuestionario. Pregunta direcciones en lienzo con bloques (Secuenciación). Fuente: Román-González (2016)*

La Figura 33 muestra un ejemplo de pregunta en el entorno de interfaz Lienzo, con estilo de alternativas de respuesta de tipo Visual por Bloques, conceptos computacionales Direcciones, Bucles-repetir veces sin anidamiento de conceptos y con tarea cognitiva requerida tipo Depuración.

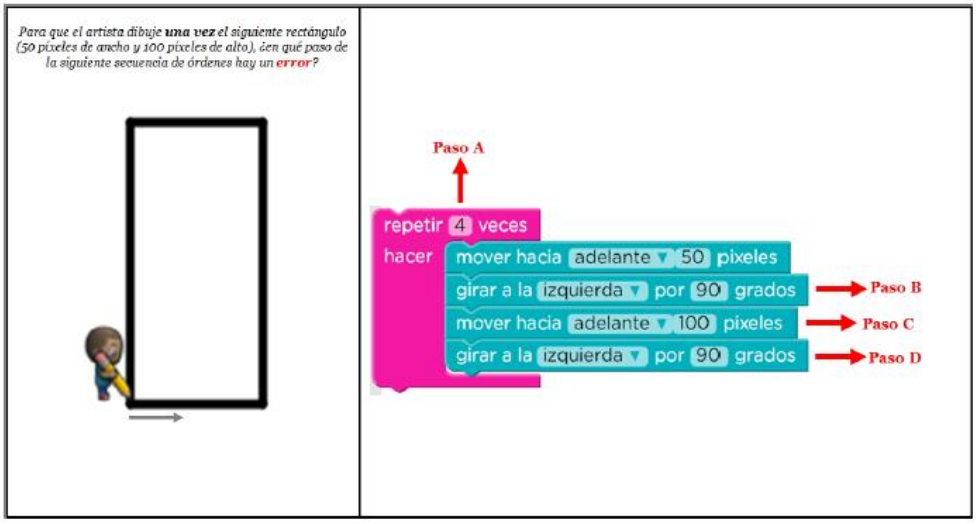


***Figura 33**. Cuestionario. Pregunta direcciones en lienzo con bloques (Depuración). Fuente: Román-González (2016)*

La Figura 34 muestra un ejemplo de pregunta en el entorno de interfaz Lienzo, con estilo de alternativas de respuesta de tipo Visual por Flechas, conceptos computacionales Direcciones, sin anidamiento de conceptos y con tarea cognitiva requerida tipo Completamiento.

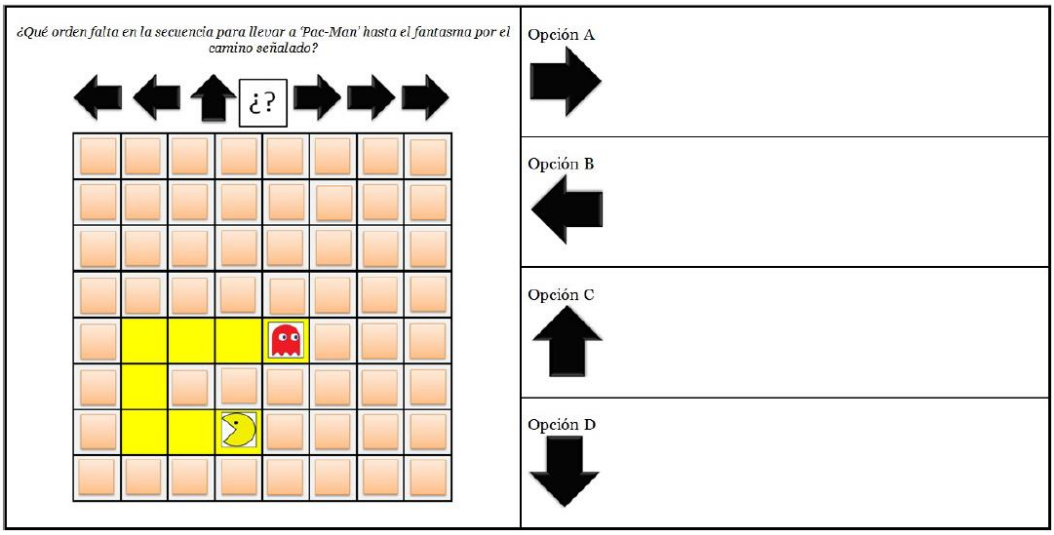


Figura 34. *Cuestionario. Pregunta direcciones en laberinto con flechas (Completamiento). Fuente: Román-González (2016)*

La Figura 35 muestra un ejemplo de pregunta en el entorno de interfaz Laberinto, con estilo de alternativas de respuesta de tipo Visual por Flechas, conceptos computacionales Direcciones, Bucles-repetir hasta sin anidamiento de conceptos y con tarea cognitiva requerida tipo Secuenciación.

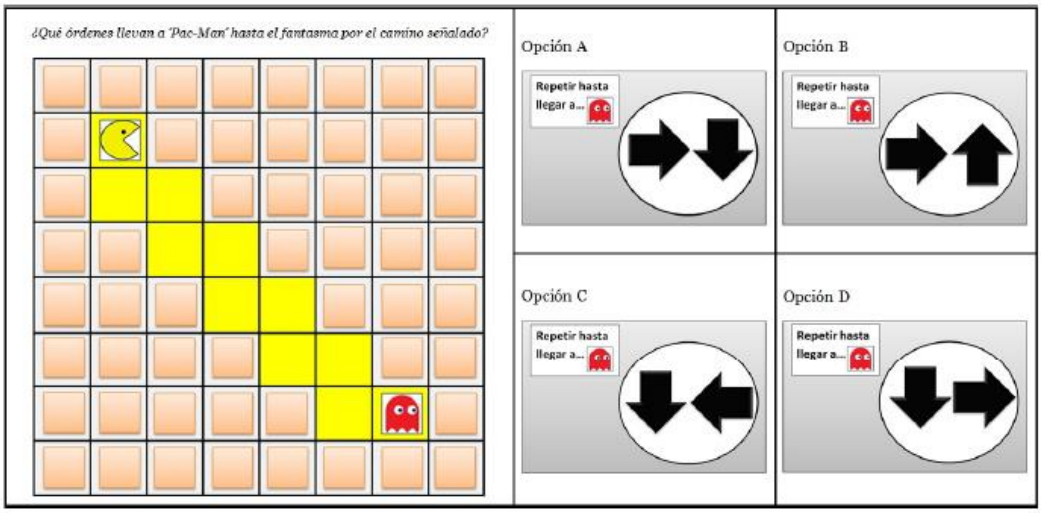


Figura 35. *Cuestionario. Pregunta direcciones en laberinto con flechas (Bucles-repetir hasta). Fuente: Román-González (2016)*

La Figura 36 muestra un ejemplo de pregunta en el entorno de interfaz Laberinto, con estilo de alternativas de respuesta de tipo Visual por Flechas, conceptos computacionales Direcciones, Bucles-repetir hasta, Condicional simple (if) con anidamiento de conceptos y con tarea cognitiva requerida tipo Secuenciación.

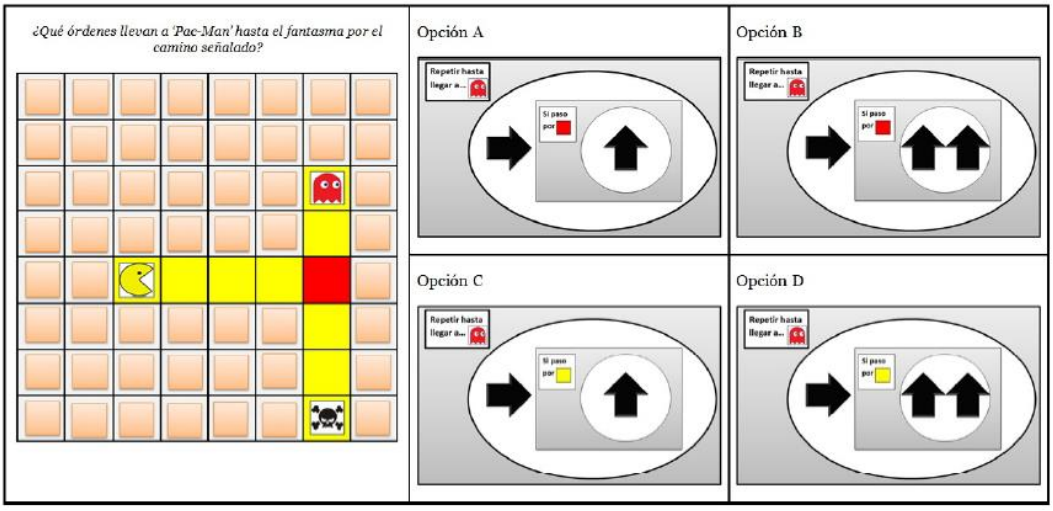


***Figura 36**. Cuestionario. Pregunta direcciones en laberinto con flechas (Condicional simple (if)). Fuente: Román-González (2016)*

La Figura 37 muestra un ejemplo de pregunta en el entorno de interfaz Laberinto, con estilo de alternativas de respuesta de tipo Visual por Bloques, conceptos computacionales Direcciones, Bucles-repetir hasta, Condicional compuesto (if-else) con anidamiento de conceptos y con tarea cognitiva requerida tipo Secuenciación.

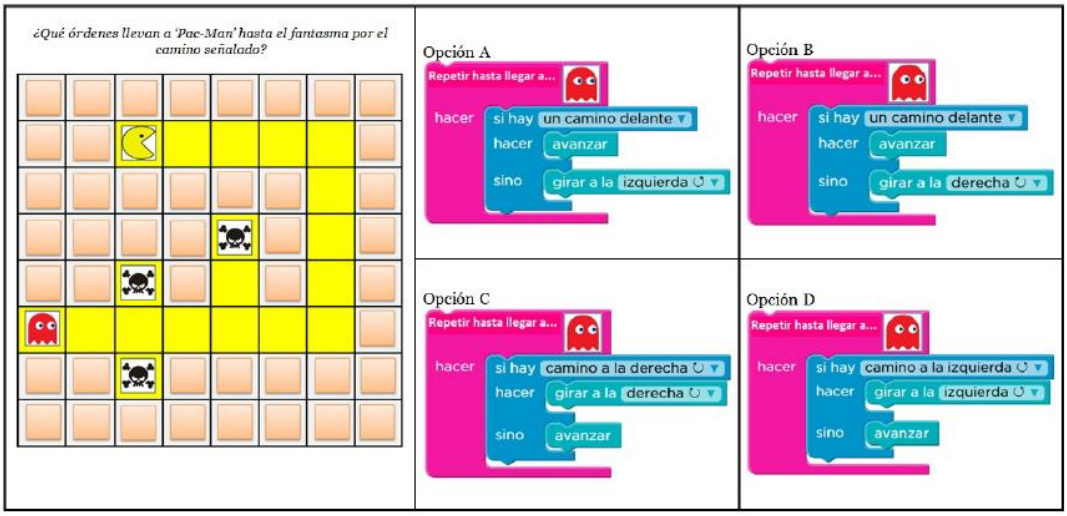


***Figura 37**. Cuestionario. Pregunta direcciones en laberinto con bloques (Condicional compuesto (if-else)). Fuente: Román-González (2016)*

La Figura 38 muestra un ejemplo de pregunta en el entorno de interfaz Laberinto, con estilo de alternativas de respuesta de tipo Visual por Bloques, conceptos computacionales Direcciones, Condicional simple (if), Mientras que (While), con anidamiento de conceptos y con tarea cognitiva requerida tipo Completamiento.

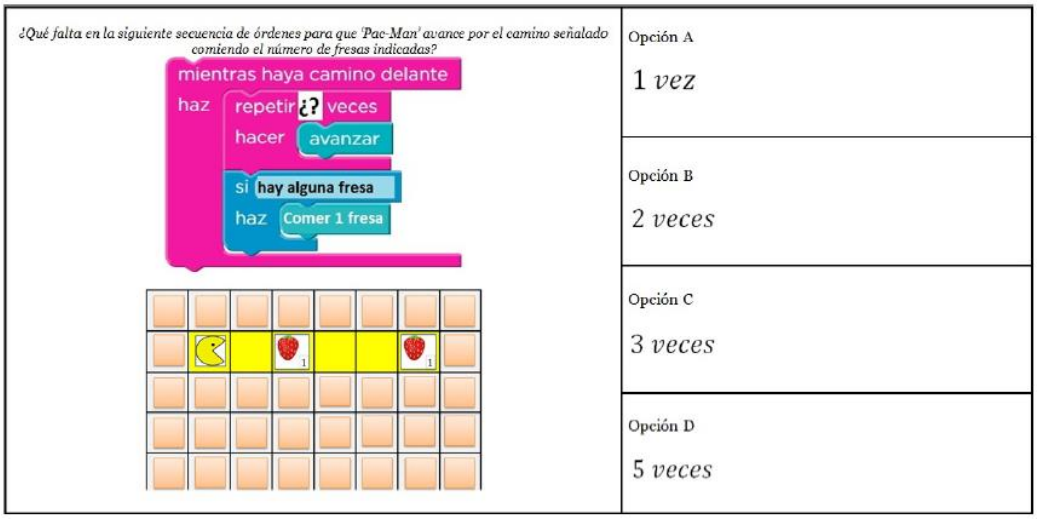


***Figura 38**. Cuestionario. Pregunta direcciones en laberinto con bloques (Mientras que (while)). Fuente: Román-González (2016)*

La Figura 39 muestra un ejemplo de pregunta en el entorno de interfaz Lienzo, con estilo de alternativas de respuesta de tipo Visual por Bloques, conceptos computacionales Direcciones, Bucles-repetir veces, Funciones simples, con anidamiento de conceptos y con tarea cognitiva requerida tipo Completamiento.

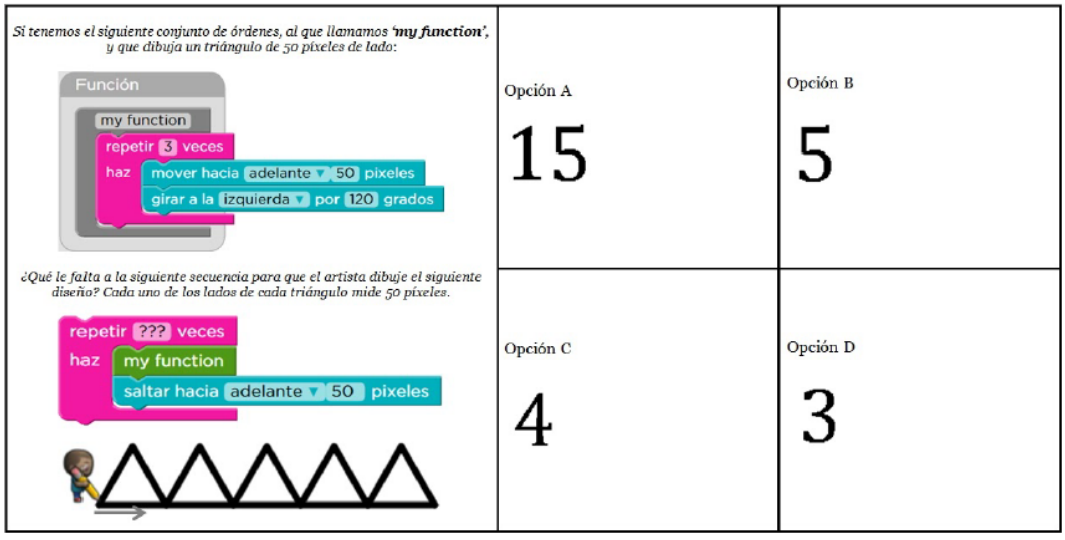


***Figura 39**. Cuestionario. Pregunta direcciones en Lienzo con bloques (Funciones). Fuente: Román-González (2016)*

Después del test de pensamiento computacional, se plantean al alumno dos preguntas de autoevaluación, las mostradas en la Figura 40.

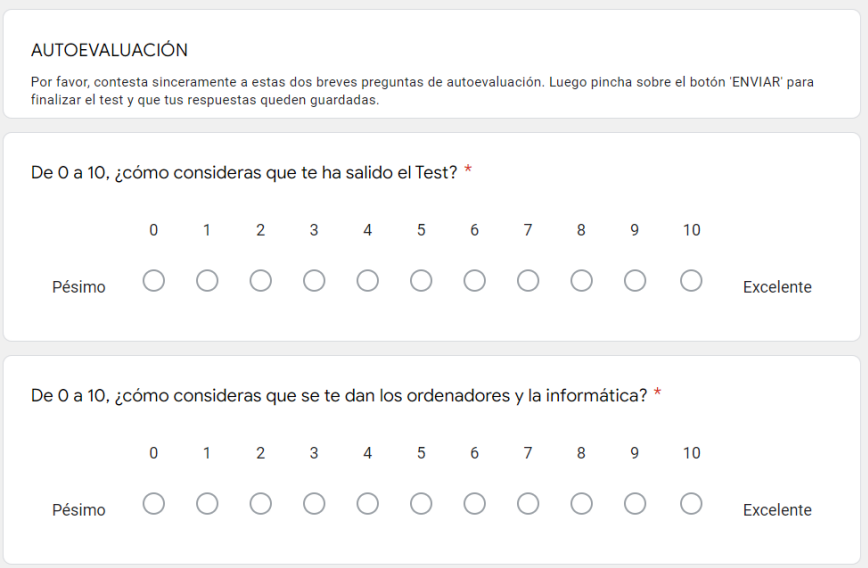


***Figura 40**. Cuestionario. Preguntas de autoevaluación. Fuente: Elaboración propia*