Trabajo de Fin de Grado

Grado en Ingeniería Informática

Robótica Educativa mediante Programación Visual

Educational Robotics through Visual Programming

Andrea Rodríguez Rivarés

La Laguna, 4 de septiembre de 2018

D. **Eduardo Manuel Segredo González**, con N.I.F. 78.564.242-Z Profesor Asociado adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor.

D. **Rafael Arnay del Arco**, con N.I.F. 78.569.591-G Profesor ayudante Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor.

C E R T I F I C A (N)

Que la presente memoria titulada:

"Robótica Educativa mediante Programación Visual"

ha sido realizada bajo su dirección por Dña. **Andrea Rodríguez Rivarés**,

con N.I.F. 79.084.084-X.

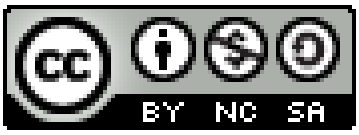
Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 4 de septiembre de 2018.

Agradecimientos

A mis padres y mi hermana pequeña por el apoyo incondicional ofrecido durante todo mi periodo académico y por esas conversaciones interminables sobre mi proyecto de futuro.

A mis tutores, Eduardo y Rafa, por la honestidad con la que me han guiado durante el desarrollo del proyecto y por los conocimientos didácticos que me han aportado.

Licencia



Resumen

Este Trabajo de Fin de Grado consiste en la creación de un entorno en donde poder utilizar un lenguaje de programación visual basado en bloques, con el que los usuarios podrán desarrollar sus capacidades de programación y depuración y fomentar la creatividad y la imaginación, mejorando así las habilidades promovidas por el pensamiento computacional desde edades tempranas, el cual implica resolver problemas y diseñar sistemas haciendo uso de los conceptos fundamentales de la informática.

Este proyecto pertenece a una línea de trabajo aun mayor que se divide en tres proyectos: creación del robot, programación visual del robot y simulador web del entorno del robot.

Con esta aplicación, el usuario podrá diseñar el robot mediante unas plantillas previamente definidas, modificando tanto el tamaño del robot como el tipo de sensores que poseerá. Una vez definidas las características del robot, el usuario podrá programar mediante bloques de código las acciones que desea que realice el robot en el entorno virtual, utilizando bloques de movimiento, bucles, comprobando los resultados que devuelven los sensores, etc. Las instrucciones generadas en este módulo se pasarán al simulador web para poder comprobar el funcionamiento de las mismas y simular el comportamiento del robot.

Palabras clave: Robótica Pedagógica, Robótica Educativa, Programación Visual, Blockly, Pensamiento Computacional

Abstract

This Final Degree Project consists of creating an environment in which to use a visual programming language based on blocks, with which users will be able to develop their programming and debugging skills and encourage creativity and imagination, thus improving skills promoted by computer thinking froam an early age, which involves solving problems and designing systems using the fundamental concepts of computer science.

This project belongs to an even greater line of work that is divided into three projects: creation of the robot, visual programming of the robot and web simulator of the environment of the robot.

With this application, users can design a robot using predefined templates, modifying the size of the base or the type of sensors of the robot. Once the robot's characteristics have been defined, the user will be able to program through code blocks the actions that the robot wishes to perform in the virtual environment, using movement blocks, loops, checking the results returned by the sensors, etc. The instructions generated in this module will be passed to the web simulator to be able to check their operation and simulate the behavior of the robot.

Keywords: Pedagogical Robotics, Educational Robotics, Visual Programming, Blockly, Computational Thinking.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

1.1 Motivación

Unos ejemplos de las preguntas que más resuenan hoy en día son: ¿Qué es la programación? ¿Para qué sirve? ¿Es muy difícil? Actualmente, la mayoría de las cosas se gestionan por medios tecnológicos, y la situación no parece que vaya a cambiar con el paso del tiempo. Al contrario, las nuevas tecnologías irán invadiendo nuestras vidas día a día.

La programación, al igual que los videojuegos, nos aportan multitud de beneficios, entre ellos el "ensayo y error", el cual nos permite obtener conocimiento a base de repeticiones. Este método de enseñanza nos permite mejorar nuestros resultados en pruebas matemáticas, razonamiento y resolución de problemas, desarrollando competencias como la lógica, entre otras.

Una de las motivaciones que han dado lugar al desarrollo de este proyecto ha sido fomentar ese tipo de pensamiento. Con la creación y programación de un robot virtual los niños podrán mejorar su concentración y sus habilidades creativas, fomentar el trabajo en equipo, potenciar su autoestima mientras van consiguiendo resultados favorables y, sobre todo, divertirse mientras exploran nuevas herramientas y aumentan su conocimiento, el cual repercutirá positivamente en sus resultados académicos futuros.

Esta aplicación con la que poder programar un robot virtual en un simulador web está pensada para que pueda utilizarse en cualquier escuela, sin necesidad de realizar la compra de ningún material físico. Actualmente existen aplicaciones similares a esta, con la que poder diseñar y programar un robot haciendo que este realice las acciones que el

usuario le indique, pero no existe ningún simulador web con el cual programar un robot mediante programación basada en bloques y ver los resultados de dicha programación en tiempo de ejecución. Se decidió utilizar la programación visual basada en bloques porque es una forma de programación más sencilla e intuitiva, ofreciendo una curva de aprendizaje muy rápida [1].

1.2 Objetivo del proyecto

El objetivo de este TFG (Trabajo de Fin de Grado) es proporcionar un lenguaje de programación visual que permita llevar a cabo la programación del comportamiento de un robot virtual en un entorno simulado.

Mediante esta aplicación se pretende introducir en la robótica a estudiantes de la escuela primaria y secundaria, con el propósito de mejorar sus habilidades relacionadas con el pensamiento computacional, el cual se define como la capacidad de un individuo de resolver problemas de distinta índole mediante el procesamiento de información, lo que implica una mejora en técnicas o habilidades de descomposición, reconocimiento y generalización de patrones y abstracción y diseño algorítmico [2]. Se decidió utilizar programación visual basada en bloques porque se trata de una forma de programación intuitiva y con la que es improbable que ocurran errores de compilación.

Con esta aplicación el usuario podrá programar un robot en un simulador, el cual habrá sido diseñado según su criterio mediante las plantillas predefinidas por el programa. El robot realizará los movimientos que el usuario haya especificado utilizando los bloques de código. La finalidad de la aplicación es que el usuario pueda programar y probar virtualmente el robot realizando acciones como, por ejemplo, mover libremente el robot por el escenario o detectar objetos y reaccionar en el caso de colisionar o acercarse a estos.

1.3 Esquema de los módulos del proyecto

Actualmente, esta aplicación se encuentra en una versión inicial. A

continuación, se describirá en profundidad los tres módulos que la conforman.

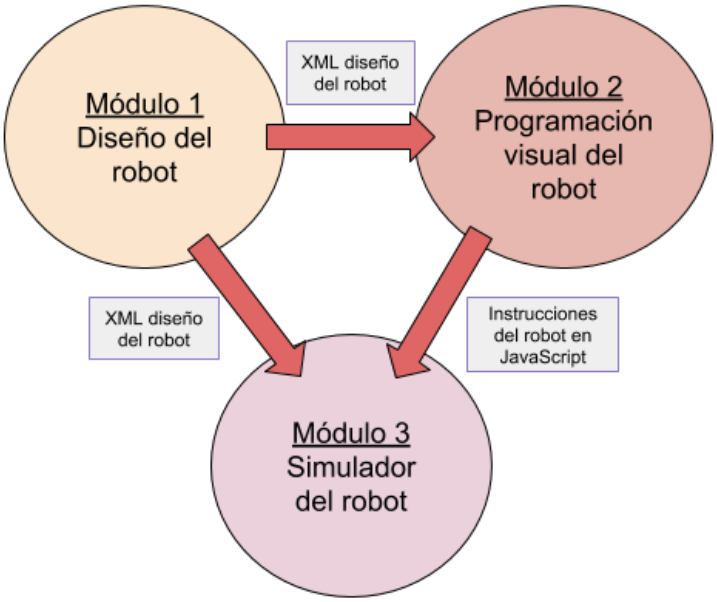


***Figura 1**: Esquema conexión módulos del proyecto*

Diseño del robot:

Este primer módulo consiste en la creación de una plataforma que permita diseñar el robot a utilizar en el simulador a partir de un entorno virtual. Las características que podrán ser modificadas del robot son: las características físicas de la base (ancho, largo y alto), el radio de las ruedas y la adición de sensores (láser, ultrasonido, infrarrojo, de contacto y telémetro láser), su ubicación en el robot y la modificación de los parámetros de los mismos (rango de medida o precisión).

Programación visual del robot:

El segundo módulo corresponde con la parte de programación del robot. Como se ha comentado anteriormente, el tipo de programación utilizada para desarrollar las acciones del robot es la programación visual mediante bloques, concretamente, desarrollada por medio de Blockly [3].

A día de hoy, se encuentran desarrollados una serie de bloques que permiten al robot moverse, utilizar los sensores, repetir acciones, etc. En la aplicación, estos bloques se encuentran divididos en categorías para

facilitar al usuario la localización de los mismos.

Una vez desarrollado el código que se quiere ejecutar en el simulador, la aplicación generará automáticamente un fichero con el código en JavaScript a partir del programa creado por el usuario a través del apilamiento de bloques. El simulador será el encargado de leer dicho fichero e interpretar las acciones que debe realizar el robot.

Simulador del robot:

Por último, el tercer módulo se encarga de recoger el fichero de configuración del robot y el fichero con las instrucciones que debe seguir el robot y se encarga de representar su comportamiento dentro de un simulador.

1.4 Estructura de la memoria

A continuación, se exponen los capítulos que componen esta memoria y una breve descripción sobre lo que trata cada uno:

- **Capítulo 2:** Contexto. Con este capítulo se pretende introducir al lector en el contexto en el que se pretende desarrollar este TFG, nombrando y explicando otros proyectos parecidos a este y las diferencias que se encuentran entre unos y otros, con el fin de que el lector pueda comprender este trabajo.
- **Capítulo 3:** Metodología. Este capítulo contendrá las fases del desarrollo que se han llevado a cabo para la realización de este proyecto y las tecnologías utilizadas para ello. También se explicará la librería principal para la realización de la aplicación, Blockly.
- **Capítulo 4:** Desarrollo del proyecto. En este capítulo de describe con detenimiento la propuesta desarrollada, sus características, funcionalidades a nivel interno y modo de uso de la aplicación.
- **Capítulo 5:** Presupuesto. Se detallará el presupuesto necesario para llevar a cabo el proyecto.
- **Capítulo 6:** Conclusiones y trabajos futuros. En este capítulo se

expondrán las conclusiones obtenidas tras la finalización del proyecto y las líneas de trabajo futuras.

- **Capítulo 7:** Conclusions and future works. Este es el capítulo de Conclusiones y trabajos futuros traducido al inglés.
- **Capítulo 8:** Códigos destacables de la aplicación.
- **Bibliografía:** Enumeración de las referencias utilizadas para la realización del proyecto.

Capítulo 2. Contexto

2.1 Conceptos básicos

2.1.1 Robótica Educativa y Robótica Pedagógica

Dentro de los conceptos de robótica y educación podemos distinguir dos áreas: la robótica educativa y la robótica pedagógica [4], las cuales tienen un objetivo común, pero se diferencian en el método que utilizan para llegar al mismo.

La robótica educativa [5] se define como un sistema de aprendizaje basado en la iniciativa y las actividades relacionadas con el estudio de las ciencias y la tecnología que potencian el desarrollo de habilidades y competencias en los estudiantes, como el trabajo en equipo, el liderazgo, el aprendizaje a partir de los errores y el emprendimiento [6].

Por otro lado, podemos definir la robótica pedagógica como una disciplina integradora de distintas áreas del conocimiento, que tiene como objetivo instituir las tecnologías en ambientes de aprendizaje mediante la adquisición de habilidades tanto científicas como tecnológicas para la resolución de problemas, siempre partiendo de la realidad, imaginando, formulando, construyendo y experimentando soluciones [7]. Enrique Ruíz Velasco Sánchez en su libro "Educatrónica: Innovación en el aprendizaje de las ciencias y la tecnología" define el objetivo de la robótica pedagógica como: "*La robótica pedagógica trata de desarrollar en el estudiante un pensamiento estructurado, que le permita encaminarse hacia el desarrollo de un pensamiento más lógico y formal*." [8].

Como se puede apreciar, ambas definiciones pretenden llegar al mismo objetivo, ya que realmente lo que diferencia a estos dos tipos de

aprendizaje es cómo llegar a dicho objetivo:

- La robótica educativa centra su campo experimental en la construcción o armado de robots con fines educativos, que no tienen que estar ligados a los ordenadores y en cierto grado interactúan con el medio ambiente por medio de sensores.
- La robótica pedagógica, en cambio, centra su campo experimental de desarrollo en el laboratorio virtual, donde la interacción y recreación de experiencias se da por medio de un ordenador.

2.1.2 Robótica como asignatura obligatoria

Muchos países como China, Reino Unido o España [9] entre otros, se encuentran cerca de incluir la robótica dentro de su plan de estudios anual para el nivel de Educación Secundaria Obligatoria (ESO). Esto se debe a la implantación de un nuevo sistema de educación denominado STEAM. STEAM engloba el estudio y aprendizaje de distintas disciplinas como ciencias, tecnologías, ingeniería, artes y matemáticas, de cuyos nombres en inglés proceden sus siglas [10][11].

Este nuevo tipo de educación tiene como objetivo promover entre los alumnos una cultura de pensamiento científico, fomentar el razonamiento basado en la evidencia para la toma de decisiones y desarrollar habilidades de análisis y pensamiento crítico, ayudando también a que el alumnado adquiera confianza y conocimientos e inspire a los alumnos a aspirar a carreras científicas o innovadoras.

2.2 Antecedentes

Desde hace un par de años, la robótica ha tenido mucha representación en las escuelas, ayudando en el proceso de construcción del conocimiento de alumnos de todas las edades. Actualmente, se imparten clases extraescolares para aquellos niños que deseen ampliar su formación introduciéndose en el extenso mundo de la tecnología y la robótica.

En 2009 se realizó un avance significativo en esta ciencia con un robot creado por la empresa Fischertechnik Education, llamado **Pro-Bot** [12], el

cual posee forma de coche. Este robot está pensado para los últimos cursos de la escuela primaria e incluso Educación Secundaria.

***Figura 2**: Robot Pro-Bot*

Pro-Bot es programable mediante Bluetooth desde el ordenador con un simulador llamado Probotix, el cual se basa en un lenguaje de programación llamado Logo, y también se puede programar mediante una serie de botones que tiene el robot en la parte superior del mismo. La ventaja de programar el robot mediante un simulador es poder programar todos los movimientos que desees que realice el robot sin llegar a probarlo físicamente, sino ver el recorrido que realizaría el robot en el simulador, lo que resulta muy útil a la hora de realizar múltiples pruebas. Esta aplicación posee una serie de funciones como: dibujar patrones con una función de "lápiz" para dibujar patrones, modificar y mejorar las rutas del robot en la pantalla, etc.

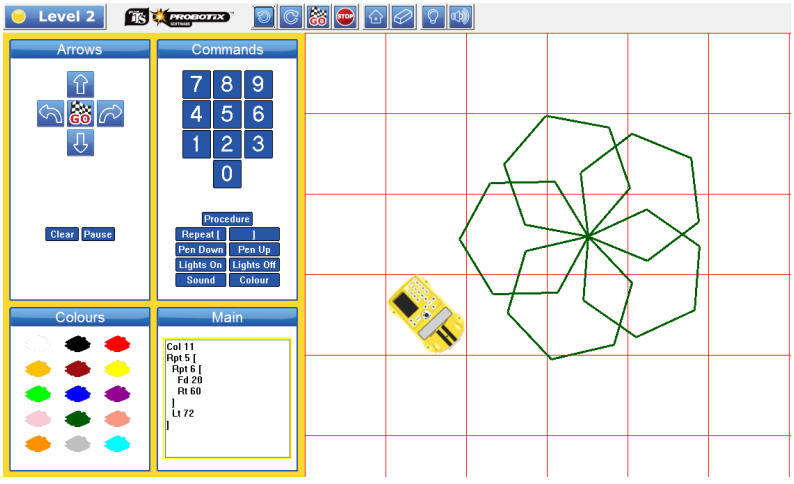


***Figura 3:** Simulador Pro-Bot*

Las características de esta aplicación se asemejan mucho a lo realizado en la aplicación de este proyecto ya que cuenta con un simulador virtual, que es el claro objetivo de este trabajo, aunque con muchas diferencias en

cuanto al lenguaje utilizado, ya que Logo es un lenguaje de programación funcional-estructurado y el lenguaje de programación elegido en este proyecto es de tipo visual por bloques. También, en cuanto al diseño del robot, Pro-Bot cuenta con un aspecto fijo mientras que el robot planteado en este proyecto podrá ser diseñado a gusto del usuario, aunque con algunas restricciones.

Con características similares a Pro-Bot, de mano de la misma empresa, en 2017 se desarrolló el robot **InO-Bot** [13].

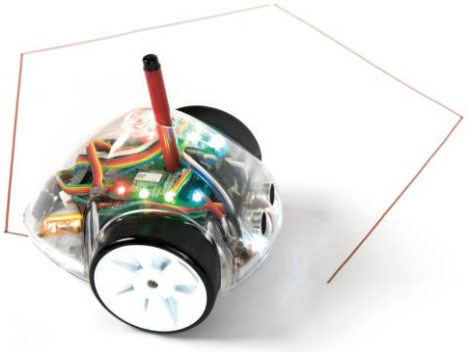


Figura 4: *Robot Ino-Bot*

Este robot es programable mediante Scratch o mediante una aplicación propia del fabricante. Este robot se puede conectar mediante Bluetooth con una conexión bidireccional, lo que permite que se pueda programar el robot en tiempo real.

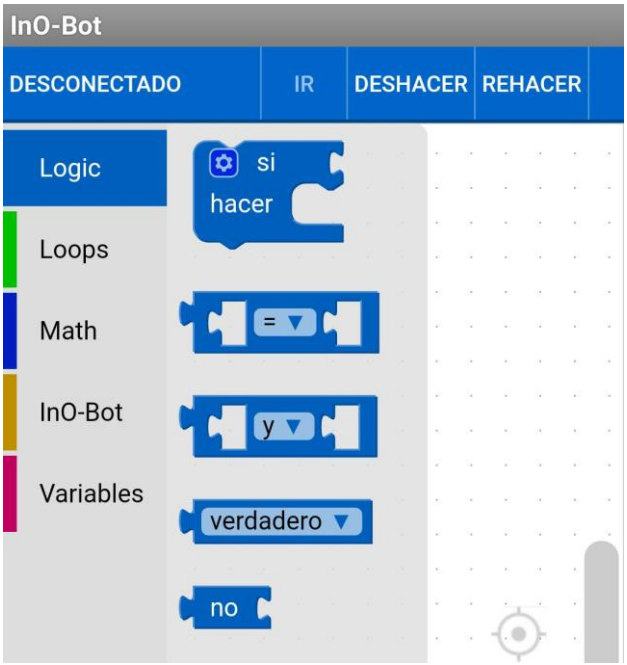


***Figura 5**: Aplicación para programar Ino-Bot*

Esta aplicación se encuentra basada en la biblioteca Blockly y se encuentra formada por una serie de categorías para separar los tipos de bloques disponibles.

Un tercer ejemplo de los avances de la robótica en la educación es el robot diseñado por la empresa LEGO Education, en el año 2013, cuya serie de robots se llaman **LEGO® MINDSTORMS® Education EV3** [14].

***Figura 6:** Robots LEGO® MINDSTORMS® Education EV3*

Estos robots son programables mediante una aplicación gratuita proporcionada por la empresa, que se basa en una programación intuitiva basada en iconos. Este software es muy similar a la programación visual por bloques en la que se basa este proyecto, con la diferencia de que no posee un simulador en el que visualizar las acciones programadas para el robot, sino que primero debe desarrollarse el código y después introducirlo en el robot para comprobar el desarrollo del programa.

En 2016, la empresa LEGO Education también lanzó al mercado un robot llamado **WeDo 2.0** [15], enfocado a niños más pequeños, de edades comprendidas entre los seis y los ocho años. Al igual que la serie de robots EV3, este robot también trabaja con un software basado en programación visual por iconos desarrollado por LEGO. La diferencia de este robot con la serie EV3 es que se puede programar también mediante Scratch [16], el cual es encuentra basado en la librería de Blockly, uno de los lenguajes de programación que nos servirá de ejemplo para crear nuestro propio lenguaje de programación visual basado en bloques.

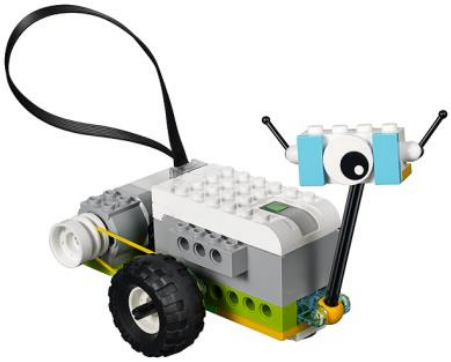


Figura 7: *Ejemplo robot WeDo 2.0*

Figura 8*: Aplicación para programar los robots de LEGO*

Por último, otro ejemplo relacionado con el tema principal de este proyecto es el robot diseñado por la empresa mOway, a finales de 2013. Este robot, llamado **mOwayduino** [17] no cuenta con un simulador, pero se puede programar con múltiples lenguajes de programación como Java, Arduino, Python o Scratch.

Figura 8: *Robot mOwayduino*

2.3 Lenguaje de Programación Visual

La Programación Visual se define comúnmente como el uso de expresiones visuales (tales como gráficos, animaciones o iconos) en el proceso de la programación. La programación visual suele ser utilizada para formar la sintaxis de nuevos lenguajes de programación visuales que conducen a nuevos paradigmas tales como programación por la demostración o, también, puede ser utilizada en las presentaciones gráficas del comportamiento o de la estructura de un programa. El objetivo de la programación visual es mejorar la comprensión de los programas y simplificar la programación en sí.

A partir de esta definición, podemos deducir que un lenguaje de programación visual es un lenguaje de programación que tiene una representación visual [18].

2.3.1 Ejemplos de Lenguajes de Programación Visual

Para entender en profundidad lo que es un lenguaje de Programación Visual, a continuación, se analizarán las aplicaciones o lenguajes más conocidos actualmente.

2.3.1.1 Scratch

Scratch [16][19] es un lenguaje de Programación Visual de código abierto el cual se puede utilizar como aplicación web o como aplicación local. Scratch tiene como principal característica su facilidad de uso. Es un lenguaje intuitivo que sirve para crear historias interactivas, juegos y animaciones, orientado a niños de todas las edades con el fin de adentrarlos en los fundamentos de la programación.

Figura 9: *Logo Scratch*

Scratch se divide en dos partes: simulador y sprites o bloques de programación. Estos sprites poseen forma de puzle y permiten programar las acciones que realizaran los objetos que tengamos activos en el simulador. Estos se dividen por categorías como: Movimiento, Sonido, Eventos, Sensores, etc., haciendo más fácil la localización de los bloques que necesitemos utilizar.

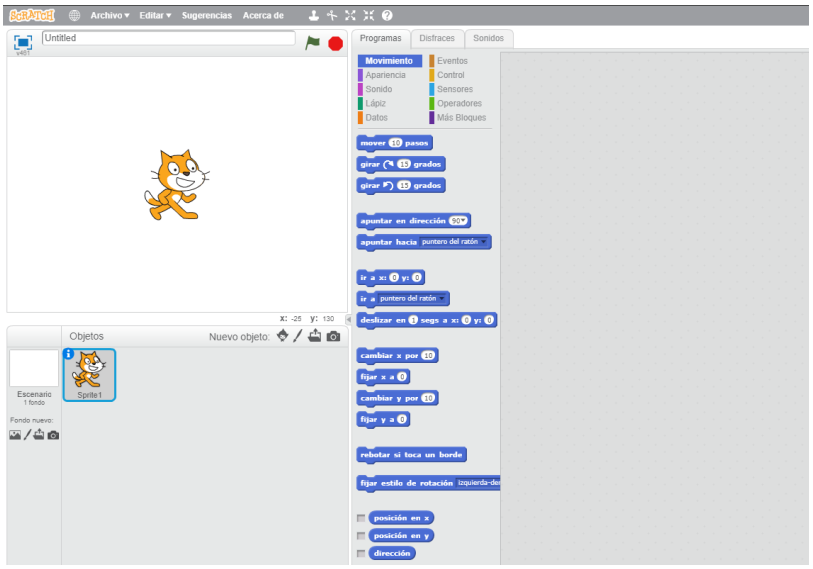


Figura 10: *Aplicación de Scratch*

Scratch es utilizado como base para otros entornos gráficos de programación, como por ejemplo mBlock. mBlock [20] es un entorno gráfico de programación gratuito basado en el editor Scratch 2.0. Con él se pueden programar mediante bluetooth robots físicos llamados Makeblock, cuya programación se basa en el lenguaje Arduino. Al igual que Scratch cuenta con un simulador y una zona de programación donde los bloques se encuentran divididos en categorías. Una característica interesante de mBlock es que permite comprobar los resultados de la programación en tiempo real, es decir, que a medida que vamos programando se puede ir comprobando los resultados instantáneamente.

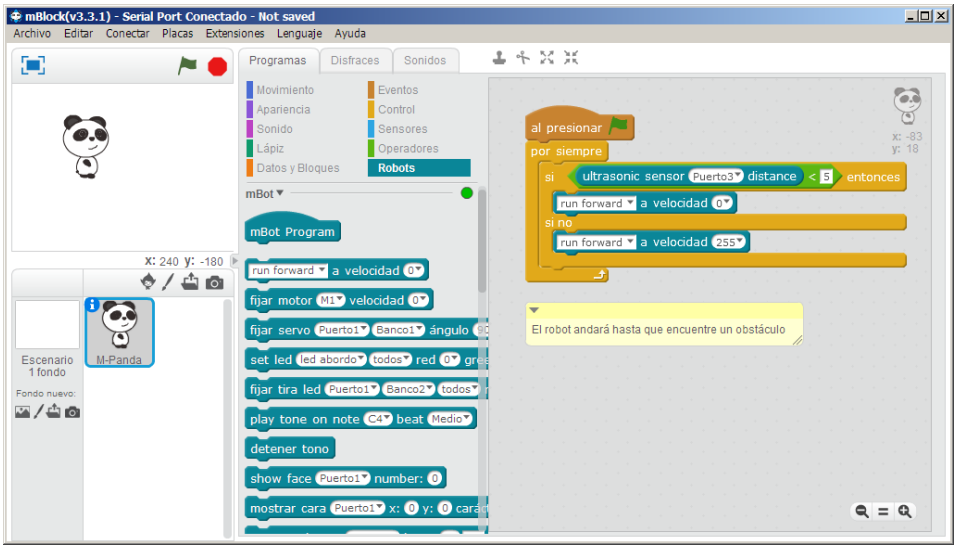


Figura 11: *Aplicación de mBlock*

2.3.1.2 Kodu

Kodu [21] es un lenguaje de programación visual desarrollado por Microsoft que permite a niños de todas las edades diseñar juegos de forma sencilla y rápida, tanto en el ordenador como en la Xbox, y también permite utilizar los juegos ya desarrollados en la propia aplicación.

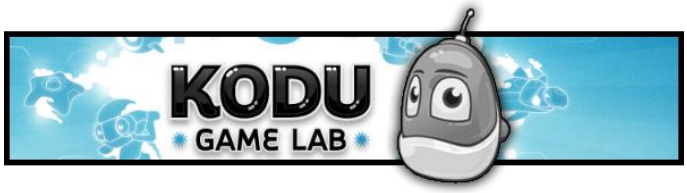


Figura 12: *Logo Kodu*

Al igual que Scratch, la programación visual de Kodu se basa en bloques de código, los cuáles se juntan para crear un programa, que puede ser ejecutado al momento.

Figura 13: *Aplicación de Kodu*

2.3.1.3 APP EV3 PROGRAMMER

Este programa, desarrollado por LEGO Mindstorms Education, consiste en un lenguaje de programación visual basado en iconos que permite la programación en tiempo real vía bluetooth de los robots creados por la empresa. Este software no consta de un simulador, el programa debe ejecutarse directamente sobre el robot para comprobar los resultados [22].

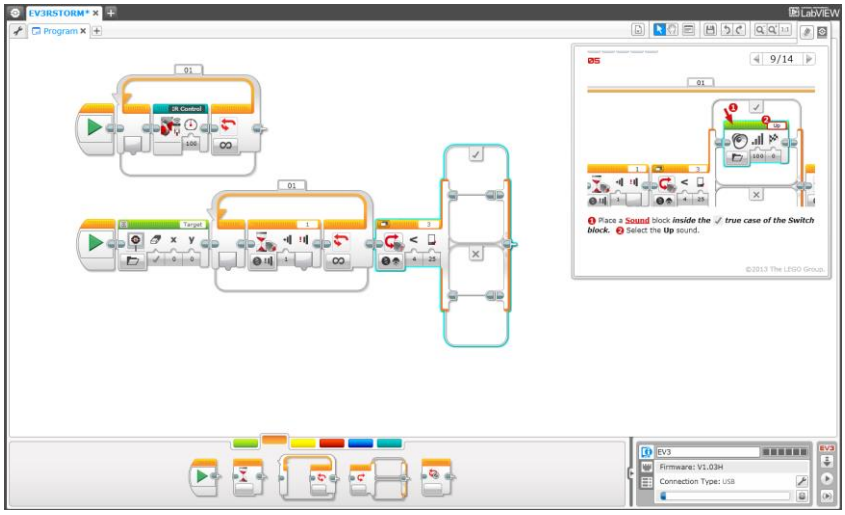


Figura 14: *Aplicación LEGO® MINDSTORMS® Education EV3*

La empresa LEGO Mindstorms Education cuenta con otros programas específicos para el ordenador, como LEGO Mindstorms EV3, con el que desarrollar los programas mediante la programación visual y comprobar los resultados una vez cargado el programa en el robot.

Capítulo 3. Metodología

3.1 Fases del desarrollo

Para realizar el desarrollo del presente proyecto se han llevado a cabo las siguientes fases:

- ***Primera fase. Búsqueda de documentación relacionada con robótica educativa***: En primer lugar, se realizó una investigación acerca de la robótica educativa actual, para conocer qué es lo que interesa a los estudiantes de la escuela primera hoy en día, cómo se ha introducido la robótica educativa en su educación, las distintas empresas que han participado en este tema, etc. Después de buscar información relacionada con esta materia, nos dimos cuenta de que la robótica pedagógica describe mejor nuestro objetivo a alcanzar con esta aplicación.
- ***Segunda fase. Búsqueda de documentación relacionada con programación visual***: En segundo lugar, se realizó una búsqueda acerca de los lenguajes de programación visual utilizados actualmente, más concretamente, sobre los lenguajes de programación visual basados en bloques, como Scratch o los lenguajes utilizados en las aplicaciones desarrolladas por la empresa Lego Education. En este punto, se decidió utilizar Blockly como herramienta principal para el desarrollo del trabajo, ya que es la librería en la que se basan muchos lenguajes de programación visual ampliamente utilizados en la actualidad.
- ***Tercera fase. Estudio de la herramienta Blockly***: A continuación, se llevó a cabo una investigación exhaustiva acerca de Blockly, leyendo la documentación ofrecida por Google, buscando

ejemplos para ayudar a su comprensión, etc.

- ***Cuarta fase. Desarrollo de un entorno web para trabajar con Blockly y creación de los primeros bloques de prueba***: Se procedió a crear un espacio de trabajo para utilizar los bloques elaborados en el entorno web que ofrece Blockly, Blockly Developer Tools [23], desarrollando los primeros bloques de prueba, como son los bloques de avanzar o retroceder el robot, girar cierto número de grados, bucles, condicionales, etc.
- ***Quinta fase. Procesamiento del fichero de configuración del robot*:** Esta tarea se llevó a cabo para habilitar los bloques necesarios para la programación del robot, haciendo una selección del conjunto de bloques que dispone el programa y mostrando aquellos que sean de utilidad para el robot diseñado. Por ejemplo, en el caso de que el robot no disponga de un sensor de ultrasonido, este bloque no le aparecerá a usuario. Para realizar esto, la aplicación deberá leer el fichero de configuración del robot y seleccionar los bloques a mostrar.
- ***Sexta fase. Desarrollo específico de los bloques del robot***: Por último, se procedió a desarrollar los bloques específicos del robot, como son: los bloques de los sensores, ajustes en los bloques de movimiento y rotación del robot, ya que los primeros bloques desarrollados solo estaban en fase de prueba.

3.2 Tecnologías utilizadas

Para el desarrollo del presente proyecto se han utilizado las siguientes tecnologías:

- HTML, CSS: Estos lenguajes de programación web se han utilizado para el desarrollo de la aplicación ya que este segundo módulo debe ejecutarse en un navegador para que el usuario pueda acceder e interactuar con la misma.
- JavaScript: Este ha sido el lenguaje principal seleccionado para el desarrollo del módulo de programación visual. Esto se debe a que

este lenguaje posee algunas características como:

- Su ubicuidad, ya que sus diferentes versiones funcionan en prácticamente todos los navegadores usados actualmente.
- Puede ser utilizado tanto para la parte del cliente como para la parte servidora de la aplicación, debido a sus estándares de codificación. Aunque en nuestra aplicación, de momento, no contemos con una base de datos, esta podría desarrollarse en versiones futuras.
- Permite corregir errores de compilación "fácilmente", ya que los navegadores cuentan con un depurador con el que recorrer el código JavaScript.

- XML: Este lenguaje se utiliza en el fichero de diseño del robot que el primer módulo, creación del robot, debe pasarle al segundo módulo, programación visual. Se decidió utilizar este lenguaje debido a que permite estructurar los datos de una forma clara y concisa, la cual nos facilita su interpretación en desarrollos posteriores. Debido a que se debe interpretar un fichero XML y obtener sus datos para poder utilizarlos en la programación visual, fue necesario desarrollar un código que leyera las etiquetas del XML [24][25]. Actualmente en el navegador Google Chrome solo se pueden obtener los datos de un XML si la aplicación y el fichero XML se encuentran en un servidor por lo que, en esta primera versión, la aplicación se ha desarrollado únicamente para el navegador Mozilla Firefox.

3.3 Blockly

Blockly [26] es una librería desarrollada por Google que proporciona un editor con el que poder crear tu propio lenguaje de programación visual para aplicaciones web y Android. Este editor utiliza bloques gráficos con forma de puzle con los que representar trozos de código como variables, bucles, condicionales, etc.

Blockly ha sido el editor elegido para llevar a cabo este proyecto debido

a su extensa documentación, facilidad de uso y su capacidad de adaptación a múltiples lenguajes de programación, ya que permite programar los bloques de código diferentes lenguajes.

Las herramientas o aplicaciones que se pueden desarrollar a partir de la librería de Blockly pueden llegar a resultar muy útiles para aquellos que se inician en la programación, ya que no contiene estructuras de código complejas, no utiliza signos de puntuación y se puede restringir los bloques que se pueden unir para, por ejemplo, evitar código que no tenga sentido.

3.3.1 Creando una aplicación utilizando Blockly

En primer lugar, para crear una aplicación basada en Blockly [27], debemos incorporar el editor de Blockly en nuestra aplicación, la cual se compone de un espacio de trabajo y una caja donde se almacenan los tipos de bloques.

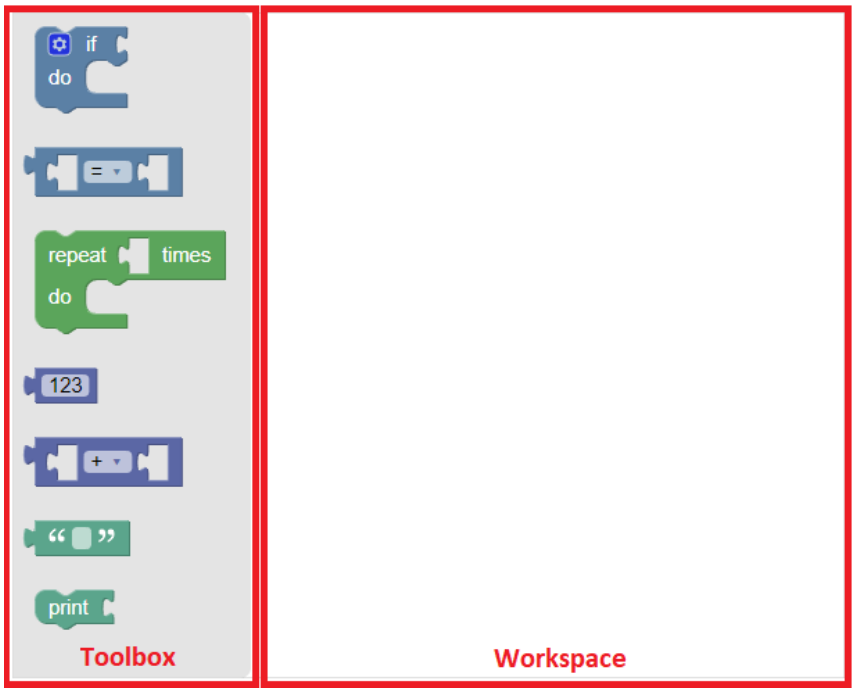


Figura 15: *Versión inicial de una aplicación basada en Blockly*

Para ello, debemos incluir los scripts necesarios para que Blockly funcione. Estos son: scripts de configuración del entorno, del idioma, del estilo, etc.

A continuación, debemos crear nuestros propios bloques para incorporarlos a la aplicación. Blockly proporciona un editor alojado en una página web donde crear bloques personalizados, denominado Blockly Developer Tools. Este editor nos proporciona todas las herramientas para crear de forma gráfica nuestros bloques y, una vez terminados, exportarlos e incluirlos en nuestra aplicación. El código de generación de los bloques (Generator Stub) se puede exportar en los lenguajes de programación JavaScript, Lua, Python, Dart y PHP, pero también se pueden encontrar versiones modificadas del editor de Blockly para generar código en otros lenguajes, como puede ser C# [28]. El código de definición de los bloques (Block Definition) solo se puede exportar en los lenguajes JSON y JavaScript.

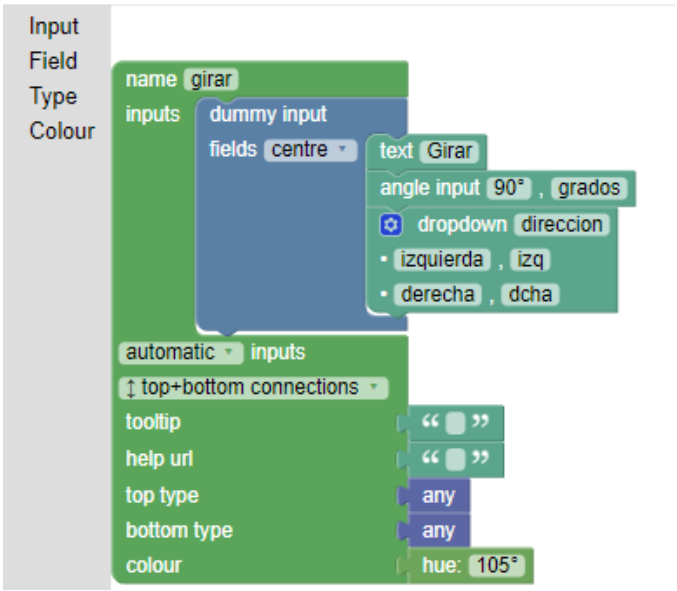


Figura 16: *Zona de desarrollo de bloques en Blockly Developer Tools*

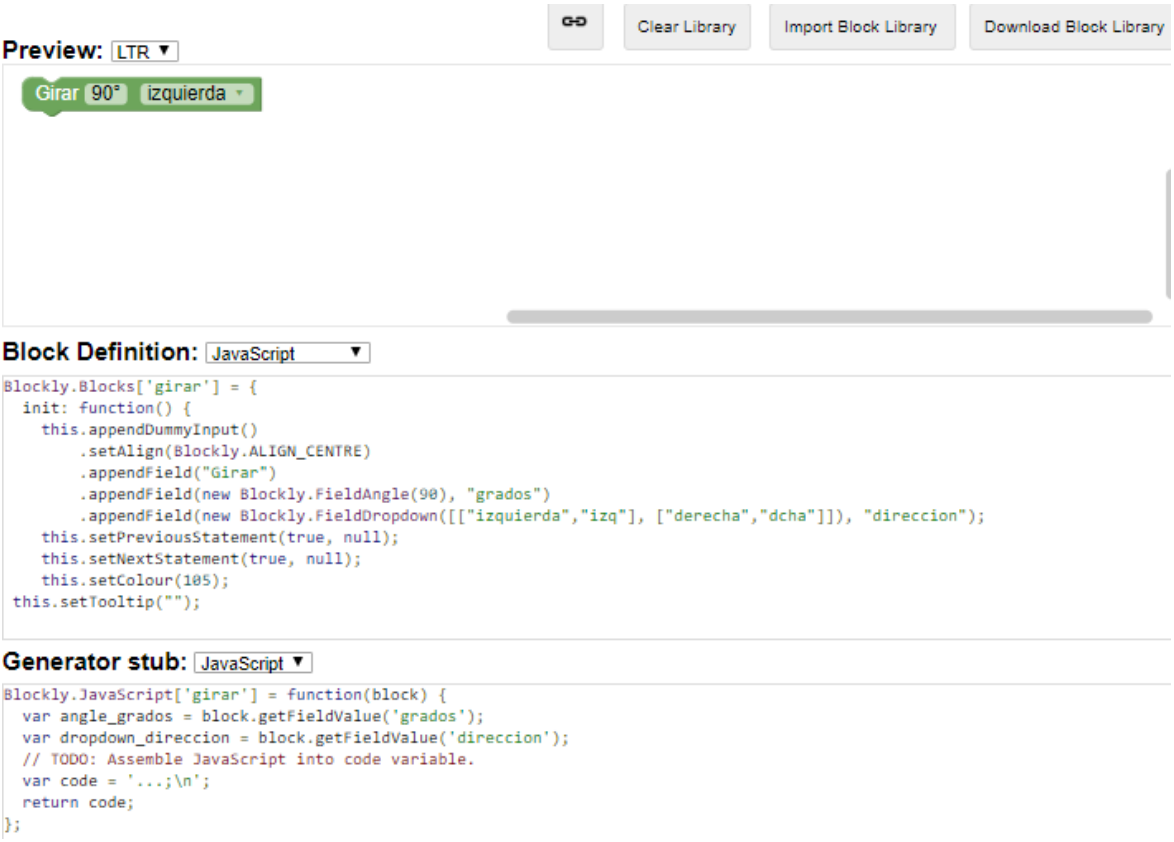


Figura 17: *Zona de generación de código de los bloques en Blockly Developer Tools*

Más concretamente, en cada área se puede realizar lo siguiente:

- **Zona de desarrollo:** De las tres áreas que contiene Blockly Developer Tools, esta es la única que podemos modificar. En ella debemos crear el bloque que queramos definir para nuestra aplicación haciendo uso de los bloques que nos facilita Blockly para tal tarea, como son los bloques "dummy input" en los que no se les puede añadir bloques en sus laterales ya que no posee ranuras para tal fin; los bloques "statement input" que poseen un hueco o ranura en la que introducir bloques que no posean conexiones de tipo puzle sino con conexión redonda"; o los bloques "value input" que incorporan una ranura de tipo puzle donde introducir bloques que posean dicha conexión.

Figura 18: *Bloque de prueba de tipo "dummy input" con conexión en puzle*

***Figura 19**: Bloque de prueba de tipo "dummy input" con conexión redonda*

***Figura 20:** Bloque de prueba de tipo "value input" con conexión por arriba y por debajo*

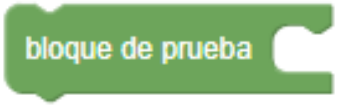


***Figura 21:** Bloque de prueba de tipo "statement input" con conexión por arriba y por debajo*

- **Block Definition:** En esta área se mostrará el código de definición del bloque que estemos creando en la zona de desarrollo. Este código es el encargado de definir características como:
    - el tipo de conexiones que tiene (por arriba o por debajo del bloque, a los lados, etc.) y tipos bloques que se pueden unir al mismo (bloques que devuelvan un booleano o un entero, por ejemplo).
    - los inputs que contiene para escribir campos.
    - campos desplegables con varias opciones.
    - el color del bloque.

  Este código debe introducirse en nuestra aplicación y debe llamarse antes de utilizar las funciones que se encargan de definir el código que generarán los bloques.

- **Generator Stub**: Esta área es la encargada de definir el código JavaScript que generará cada bloque al ser utilizado por el usuario. Se deben definir como variables los campos del bloque con los que el usuario puede interactuar, como son los inputs, desplegables, etc. También se debe introducir en una variable (en nuestro caso, llamada "code") el código de salida del bloque. Esta salida debemos definirla mediante cadenas de texto y llamadas a

las variables declaradas en la definición del bloque, por ejemplo: “var code = 'Avanzar(' + metros_seg + ',' + segundos + ');';”, donde “metros_seg” y “segundos” se corresponden con variables de tipo input que se han declarado en la definición del bloque y el resto se corresponde con cadenas de texto.

Por último, se debe modificar el código de los bloques para que realicen las acciones que necesitemos. Blockly nos proporciona las herramientas para generar el código de los bloques y nosotros debemos utilizarlo para que estas hagan funcionar nuestra aplicación.

Capítulo 4. Desarrollo del proyecto

4.1 Componentes

La aplicación web está compuesta por tres partes bien diferenciadas:

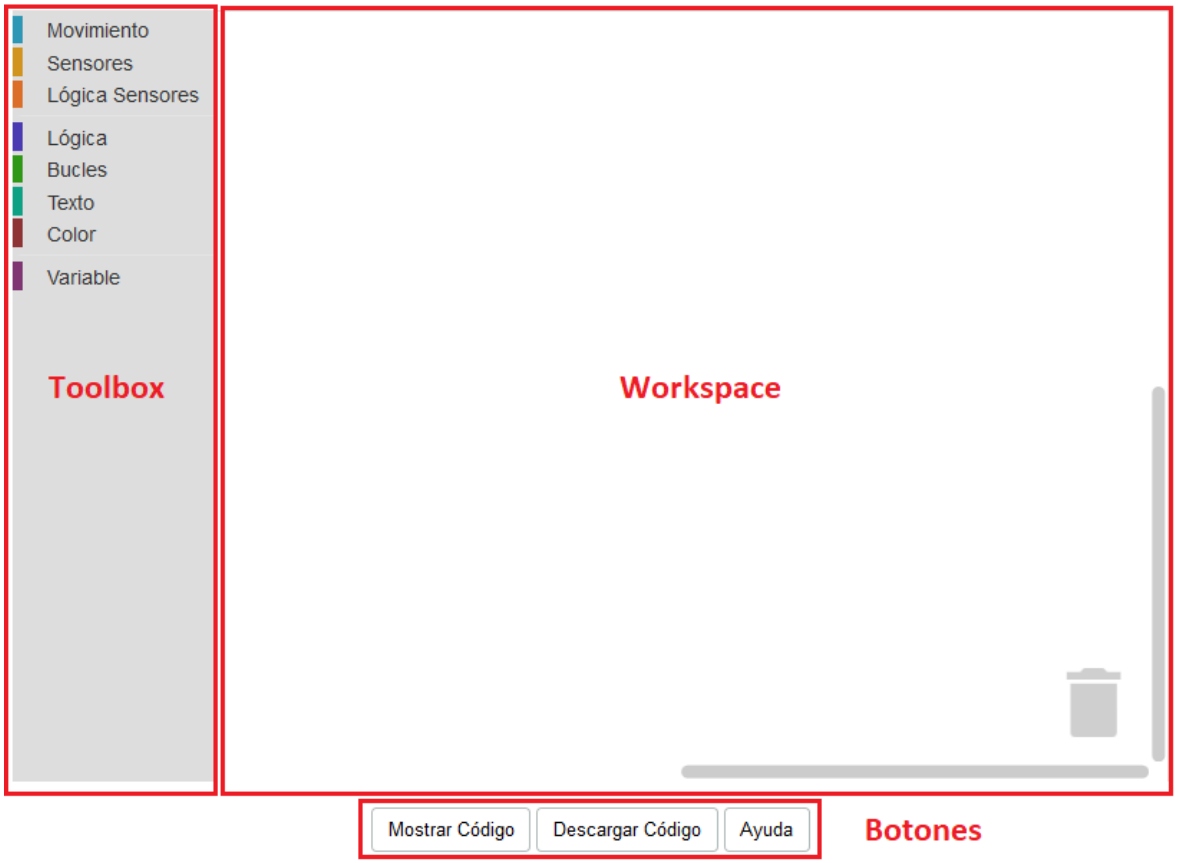


***Figura 22**: Aplicación diferenciando las tres áreas que la componen*

- **Toolbox o caja de herramientas**: Área donde se encuentran los bloques disponibles para la programación del robot.
- **Workspace o espacio de trabajo**: Área de trabajo donde se insertan los bloques cuyo código asociado será el que se le pase al simulador para su posterior interpretación.

- **Botones**: Se disponen de tres botones que realizan las acciones de: visualizar el código que se genera a partir de los bloques seleccionados en el workspace, generar un fichero JavaScript con dicho código y consultar una ayuda sobre cómo utilizar los bloques disponibles en el toolbox.

A continuación, se describirán estas tres áreas de la aplicación, entrando en detalle sobre su contenido y sobre cómo funciona dicha área.

4.1.1 Toolbox o caja de herramientas

Como se ha mencionado anteriormente, el toolbox [29] es el área donde se encuentran los bloques que el usuario puede utilizar para programar el robot. Esta área solo contendrá bloques con los que programar el robot según el diseño seleccionado previamente. En otras palabras, solo estarán disponibles aquellos bloques que le sean de utilidad al usuario en función del robot diseñado. La finalidad de este resultado es que, a la hora de ejecutar el código, el simulador sepa interpretar correctamente todos los bloques de código, minimizando el porcentaje de error, ya que la aplicación ha sido diseñada principalmente para estudiantes de la escuela primaria y secundaria.

La estructura del toolbox se programa en lenguaje XML. Esta se puede crear de la siguiente forma:

```
<xml id="toolbox" style="display: none;">
        <block type="controls_if"></block>
        <block type="controls_whileUntil"></block>
</xml>
```

Lo que generará los siguientes bloques:

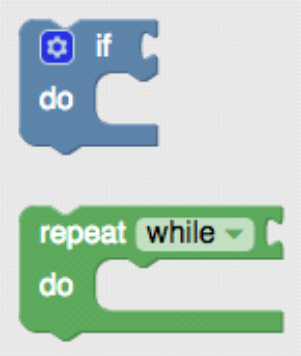


Figura 23: *Bloques de ejemplo*

Los bloques disponibles en el toolbox se encuentran organizados en categorías [30]. Estas categorías se utilizan para facilitar al usuario la búsqueda de los bloques cuando se dispone de un número elevado de bloques. En el caso de nuestra aplicación se han incorporado también categorías dinámicas [31]. Lo que diferencia a este tipo de categorías de las categorías normales es que estas varían su contenido en función de los bloques que queramos mostrar.

En la aplicación se dispone de las siguientes categorías:

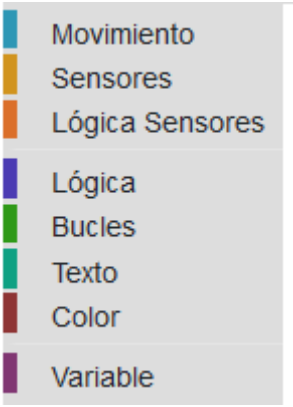


Figura 24: *Categorías del toolbox aplicación*

Las categorías Movimiento, Sensores y Lógica Sensores son del tipo dinámicas. Estas categorías contienen bloques dependiendo del diseño del robot. El diseño del robot se encuentra especificado en un fichero XML el cual, en esta fase de la aplicación, se debe interpretar y mostrar los bloques específicos para dicho robot.

La creación de estas categorías se realizó de la siguiente forma:

- Creación de la categoría con una propiedad personalizada:

```
<category name="Movimiento" custom="RobotBlocks"></category>
```

- Definición de la devolución de la llamada para proporcionar los contenidos de la categoría. Esta función debe devolver un array de bloques en lenguaje XML:

```
Blockly.defineRobotBlocks = function(workspace) {…};
```

- Registro de la devolución de la llamada en el workspace:

```
demoWorkspace.registerToolboxCategoryCallback('RobotBlocks
', Blockly.defineRobotBlocks);
```

Una vez creadas las categorías dinámicas, se procedió a la inserción de los bloques en su interior. A continuación, se describirá cada categoría con sus correspondientes bloques y las funciones de cada uno de estos:

- **MOVIMIENTO**:

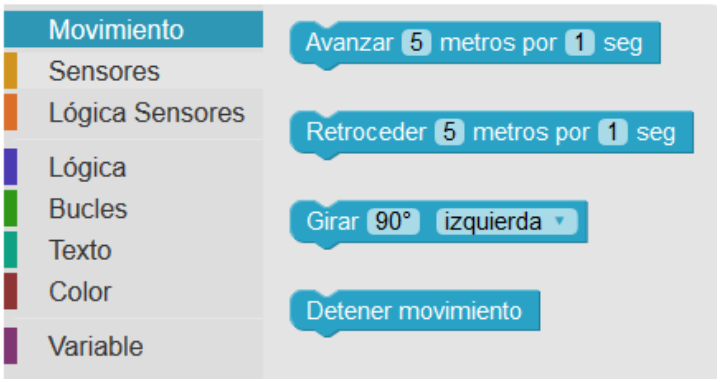


Figura 25: *Bloques categoría Movimiento*

Esta categoría contendrá los bloques que se utilizan para proporcionar acciones al robot. Estos bloques son:

- **Avanzar**: Este bloque devuelve la siguiente función: "Avanzar (X, Y);". Esta función le otorga movimiento hacia delante al robot. La variable X corresponde con el número

de metros que avanza el robot y la variable Y los segundos que tardará en recorrer dichos metros.

- **Retroceder**: Este bloque devuelve la siguiente función: “Retroceder (X, Y);”. Esta función le otorga movimiento hacia atrás al robot. La variable X corresponde con el número de metros que retrocede el robot y la variable Y los segundos que tardará en recorrer dichos metros.

- **Girar**: Este bloque devuelve la siguiente función: “Girar (grados, dirección);”. Esta función permite girar al robot el número de grados que le indique el usuario en dirección hacia la izquierda o hacia la derecha. La variable grados corresponde con el número de grados que girará el robot y la variable dirección, la dirección a la que debe girar: izquierda o derecha.

- **Detener**: Este bloque devuelve la siguiente función: “Stop ();”. Esta función hace que el robot detenga su movimiento. Puede ser útil en situaciones en las que, por ejemplo, se desee que el robot avance una serie de metros hasta que se encuentre con un obstáculo y deba detenerse.

- **SENSORES**:

Figura 26: *Bloques categoría Sensores*

Esta categoría contiene los sensores que se ha especificado en el diseño del robot. Estos bloques se utilizarán para comprobar si el robot

está detectando un objeto con el sensor especificado. Los sensores disponibles en el diseño del robot son: sensor láser, sensor de contacto, sensor de ultrasonido, sensor infrarrojo y sensor telémetro láser con barrido 2D (sensor láser 2D). Estos bloques son variables, las cuales se pueden comparar a 0 o a una variable booleana para confirmar si el sensor está en contacto o detecta algún objeto. El diseño de estos bloques se compone de un input en el cual introducir el número del sensor al cuál queramos referirnos. Estos números corresponderán con el orden en el que fueron insertados los sensores en la fase de diseño del robot. En el caso de introducir un número mayor al número total de sensores que haya de ese tipo, la aplicación lanzará un error con el cometido de informar al usuario de que no existe ningún sensor que corresponda con ese número y le indicará el número de sensores totales de ese tipo que haya disponibles.

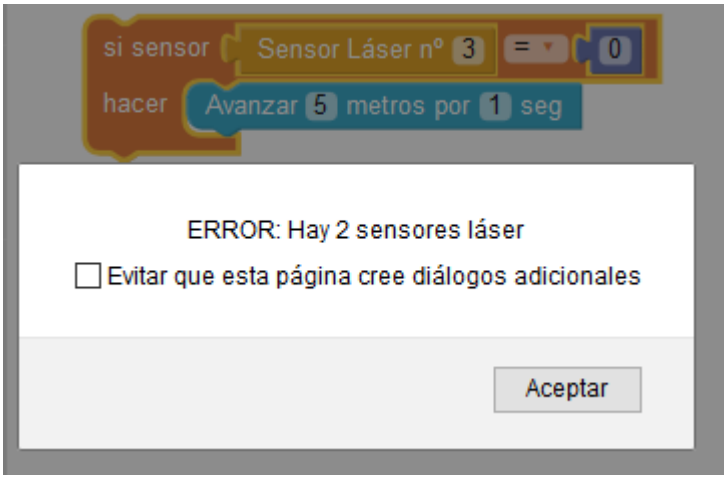


Figura 27: *Ejemplo error al introducir el número de sensor a utilizar*

- **LÓGICA SENSORES**:

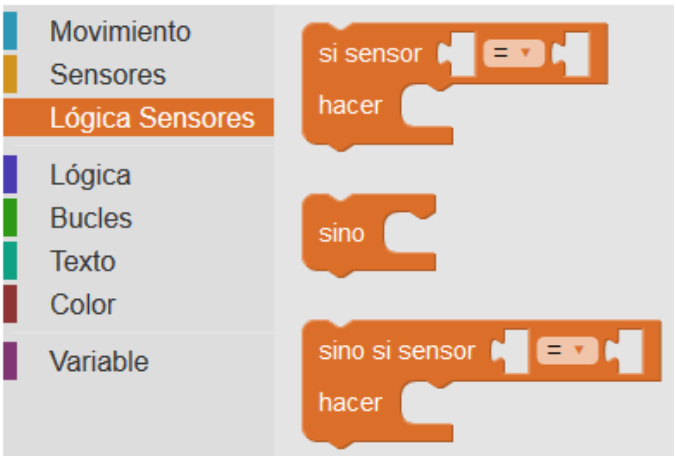


Figura 28: *Bloques categoría Lógica Sensores*

En esta categoría se encuentran las sentencias condicionales para trabajar con los sensores. Blockly ofrece un bloque específico para realizar este tipo de consultas, pero el campo que se le puede introducir a la sentencia "if" no reconoce tipo de datos String, que es el tipo de dato de los sensores, por lo que se decidió crear bloques específicos para trabajar con los sensores.

Por otro lado, en el toolbox de la aplicación también se disponen de las categorías Lógica, Bucles, Texto y Color. Estas categorías se encuentran definidas por Blockly en un script el cual incluimos en nuestro fichero principal, el HTML donde se encuentra la aplicación. Una vez hecha la referencia a dicho script, debemos insertar los bloques que queremos utilizar en el toolbox de la aplicación mediante su código XML. Una vez hecho esto, dispondremos de los siguientes bloques:

Figura 29: *Bloques categorías Lógica, Bucles, Texto y Color*

Dentro de los bloques desarrollados por Blockly, hay una categoría dinámica denominada Variables. Esta categoría contiene un botón que al pulsarlo aparece un cuadro de diálogo con el que poder crear una nueva variable. Una vez introducimos el nombre de dicha variable, nos aparecerán una serie de bloques con lo que poder interactuar con la misma.

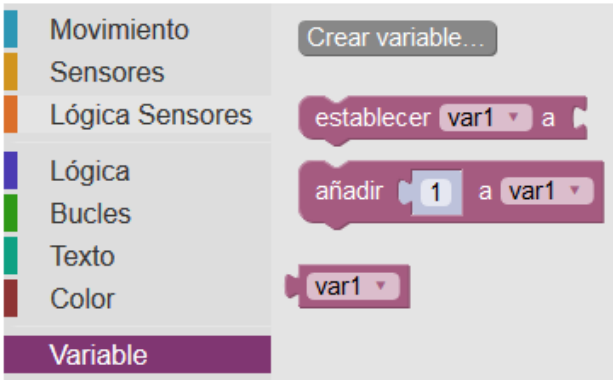


Figura 30: *Bloques categoría Variable*

Por último, en cuanto al estilo del toolbox, las categorías se encuentran separadas entre sí por un componente denominado <sep></sep>, que crea una línea divisoria. En nuestro caso la hemos utilizado para diferenciar los bloques de uso específico para el robot de los bloques implementados por Blockly. A las categorías también se les puede añadir una pequeña línea de color en la parte izquierda del nombre, el cuál coincide con el color asignado a cada bloque dentro de su categoría.

4.1.2 Workspace o espacio de trabajo

El workspace es el área donde se muestran los bloques que se desean utilizar en la programación del robot. Esto incluye también al toolbox de la aplicación, el cual debe insertarse mediante un Script a dicho espacio de trabajo:

```
var demoWorkspace = Blockly.inject('blocklyDiv', {
        toolbox: document.getElementById('toolbox')
});
```

Blockly ofrece múltiples características que se le pueden añadir al workspace para modificar su diseño y añadirle funcionalidades que faciliten la tarea de programación al usuario. A continuación, se explicarán aquellas características que se han añadido a nuestra aplicación:

- Añadir una cuadrícula: Con esto se puede hacer que los bloques se ajusten a la cuadrícula, resultando en un diseño más limpio. Esto podría ser útil cuando sea necesario agrupar un gran número de bloques.
- Añadir una opción de zoom: Esta funcionalidad permite que el tamaño del workspace sea dinámico para el usuario. Se puede configurar para que el zoom pueda hacerse mediante botones o mediante la rueda del ratón.
- Papelera: Esta característica se puede añadir al workspace para definir un lugar específico donde el usuario puede desechar los bloques que haya seleccionado y finalmente no desee utilizar. Esta acción también se puede realizar arrastrando de nuevo los bloques al toolbox de la aplicación.

Actualmente, el workspace posee un tamaño fijo, por lo que se le han añadido barras de desplazamiento en la parte inferior y en el lateral de la aplicación.

Figura 31: *Imagen workspace de la aplicación*

4.1.3 Botones

En primer lugar, para el funcionamiento de la aplicación, es necesario tener un botón que genere un fichero con el código resultante de los bloques seleccionados en el workspace.

Figura 32: *Botón Descargar Código*

El fichero generado mediante el botón se trata de un fichero con extensión JavaScript, ya que es el lenguaje del código que devuelven los bloques. Una vez pulsemos el botón se descargará en nuestro navegador el fichero con nombre “código.js”. Este fichero deberemos guardarlo en el directorio desde donde el simulador deberá interpretarlo y ejecutarlo.

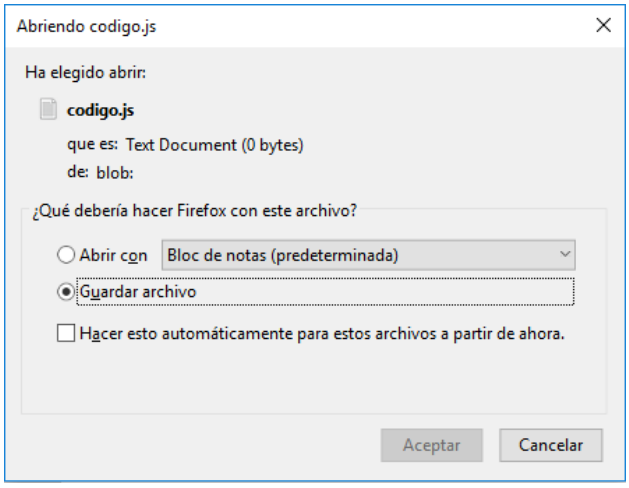


Figura 33: *Ejemplo de funcionamiento del botón "Descargar Código"*

En segundo lugar, la aplicación dispone de un botón para visualizar en un modal el código resultante de los bloques, sin tener que llegar a descargar el fichero. Esto nos permite ir comprobando de forma más rápida el código que se va generando a medida que vamos programando.

Figura 34: *Botón Mostrar Código*

Figura 35: *Ejemplo de funcionamiento del botón "Mostrar Código"*

Por último, se ha incorporado a la aplicación un botón que muestra una ayuda o información de interés para el usuario. Al ejecutar el botón, se nos abrirá un modal que contendrá imágenes de ejemplo del uso de los

distintos bloques de la aplicación y una guía sobre cómo utilizar los bloques de los sensores.

Figura 36: *Botón Ayuda*

Figura 37: *Modal botón Ayuda*

4.2 Funcionamiento a nivel interno

En este apartado se explicará en detalle qué información debe recibir el módulo de programación visual del robot y qué información debe devolver el mismo.

Para poder comenzar con la programación del robot mediante la programación visual basada en bloques, es necesario recoger, en primer lugar, la información de diseño acerca del robot. Para ello, el módulo de diseño del robot proporciona un fichero XML con toda la información necesaria para la selección de los bloques.

A continuación, se carga el fichero de definición de los bloques. Este es el fichero encargado del diseño de los bloques, donde se especifica el tipo de bloque, el tipo de datos que puede recoger dicho bloque, el color del bloque, etc. El código se puede generar en los lenguajes JSON o JavaScript.

Luego, se carga el fichero de generación de los bloques. Este fichero contiene el código de salida de cada uno de los bloques. El código de salida se encuentra desarrollado en JavaScript, aunque también se puede generar en los lenguajes Lua, Python, Dart y PHP.

A continuación, con los bloques desarrollados y con su salida definida, es el momento de seleccionar aquellos bloques que sean de utilidad para el usuario. Actualmente, los bloques que varían dependiendo del diseño del robot son los bloques de los sensores. Cada sensor dispone de su propio bloque, por lo que, si el usuario no le añade un tipo de sensor al robot, el bloque correspondiente a ese sensor no le aparecerá al usuario. Con esto conseguimos un diseño simple e intuitivo para el usuario.

Una vez cargados todos estos ficheros, es el momento de comenzar a programar el robot, utilizando los bloques disponibles en el Toolbox de la aplicación.

Por último, se debe generar el fichero que contendrá el código de programación que debemos pasarle al simulador para que el robot pueda realizar las acciones indicadas.

4.3 Modo de uso

En primer lugar, debemos guardar el fichero de configuración del robot (fichero XML con la especificación de diseño del robot) en la carpeta principal de nuestro proyecto. En esta carpeta se encuentran todos los ficheros fundamentales para que nuestra aplicación funcione, es decir, los ficheros JavaScript, CSS y HTML.

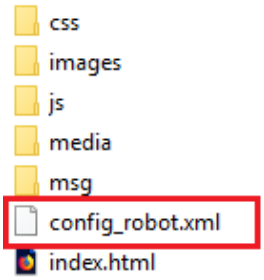


Figura 38: *Estructura directorio principal de la aplicación*

Una vez realizado el primer paso, debemos abrir la aplicación web con el navegador Mozilla Firefox. Como se ha comentado en capítulos anteriores, actualmente, la aplicación solo está disponible para este navegador.

A continuación, podemos comprobar que en nuestra aplicación aparecen todos los bloques necesarios para la programación del robot. Si abrimos la categoría "Sensores" podremos observar que se encuentran disponibles los bloques de los sensores que hayamos indicado en el diseño del robot. Para hacer uso de los bloques de los sensores de manera correcta, es necesario tener en cuenta lo siguiente:

- Los sensores láser, ultrasonido y láser 2D devuelven una medida con la distancia en metros a la cual se sitúa el objeto con el que se encuentra colisionando el sensor, por lo que para comprobar si dichos sensores se encuentran en contacto con algún objeto, debemos verificar si dicho bloque es mayor a cero.

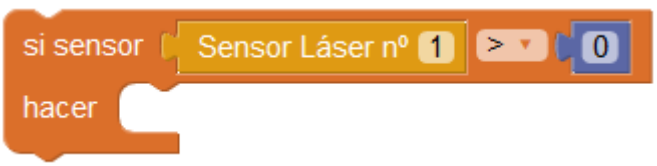


Figura 39: *Ejemplo uso de sensor que devuelve una distancia en metros*

- Los sensores de contacto e infrarrojo devuelven un booleano con la información que retorna el sensor, devolviendo verdadero (true) en caso de que haya detectado un objeto o falso (false) en caso contrario.

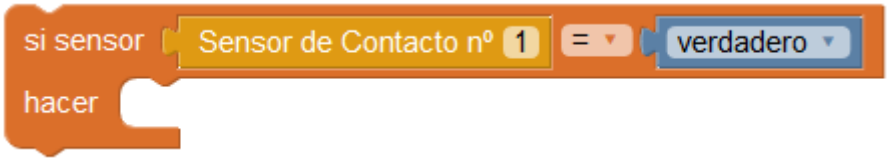


Figura 40: *Ejemplo uso de sensor que devuelve un booleano*

El usuario puede encontrar esta información haciendo uso del botón "Ayuda" localizado en la parte inferior de la aplicación. Junto con esta información sobre de los sensores, también podrá encontrar ejemplos de uso de los bloques de la aplicación.

Para finalizar, una vez desarrollado el código deseado, se debe pulsar el botón "Descargar Código" que generará el fichero con el código de los bloques, el cual deberá ser leído por el tercer módulo que corresponde con el simulador del robot.

4.5 Caso de uso

En este capítulo se expondrá un caso de uso del módulo de programación visual. Para ello, se generará a partir del módulo de diseño del robot, una configuración de ejemplo que se le pasará al segundo módulo para que este pueda generar los boques necesarios para la programación del robot. Por último, se generará un fichero con las instrucciones que debe seguir el robot para realizar las acciones pertinentes en el simulador.

4.5.1 Ejemplo de uso

EL fichero XML de configuración del robot que generaremos a modo de ejemplo es el siguiente:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ROBOT>
    <Base>
        <Tipo>rectangular</Tipo>
        <Largo>2.467716</Largo>
        <Anchura>1.341158</Anchura>
        <Altura>0.7882345</Altura>
    </Base>
    <Ruedas>
        <Numero>3</Numero>
        <RadioRuedasAtras>0.6749998</RadioRuedasAtras>
        <RadioRuedaCentral>0.5749999</RadioRuedaCentral>
    </Ruedas>
    <Sensores>
        <Laser>
            <Laser1>
                <Activo>true</Activo>
                <Rango>20</Rango>
                <Precision>100</Precision>
            </Laser1>
            <Laser2>
                <Activo>true</Activo>
                <Rango>30</Rango>
                <Precision>80</Precision>
            </Laser2>
        </Laser>
        <Ultrasonido>
            <Activo>true</Activo>
            <Rango>9.820101</Rango>
            <Precision>57.13429</Precision>
        </Ultrasonido>
        <Infrarrojo>
            <Activo>true</Activo>
            <Rango>20</Rango>
            <Precision>100</Precision>
        </Infrarrojo>
        <Contacto>
            <Activo>true</Activo>
            <Rango>20</Rango>
            <Precision>100</Precision>
        </Contacto>
        <Laser2D>
            <Activo>true</Activo>
            <Rango>20</Rango>
            <Precision>100</Precision>
        </Laser2D>
    </Sensores>
</ROBOT>
```

Figura 41:*Ejemplo fichero de configuración del robot*

Este fichero también debe pasarse al módulo del entorno de simulación del robot, por lo que contiene información innecesaria para el módulo de programación visual. La información importante para dicho módulo es la referente a los sensores. Dentro de la etiqueta sensores podemos observar que el robot posee dos sensores de tipo láser, un sensor ultrasonido, un sensor infrarrojo, un sensor de contacto y un sensor láser 2D.

Para esta configuración los bloques generados serán los siguientes:

Figura 42: *Bloques generados a partir del fichero de configuración del robot*

El usuario podrá utilizar todos los sensores disponibles en el fichero de configuración del robot. En el campo de introducción de texto del sensor se debe indicar el número del sensor al que se quiere hacer referencia. Este número corresponderá con el orden en el que haya seleccionado los sensores al diseñar el robot. En el caso de seleccionar un número mayor al número de sensores disponibles para ese tipo de sensor, aparecerá un mensaje de error en el navegador.

Una vez se encuentre cargado el entorno de programación visual, debemos arrancar el entorno de simulación, para conocer el escenario donde se moverá el robot. A modo de ejemplo, se cargará un escenario en el simulador en el que podemos encontrar un robot que tendrá el diseño creado en el primer módulo y una serie de objetos con lo que poder interactuar.

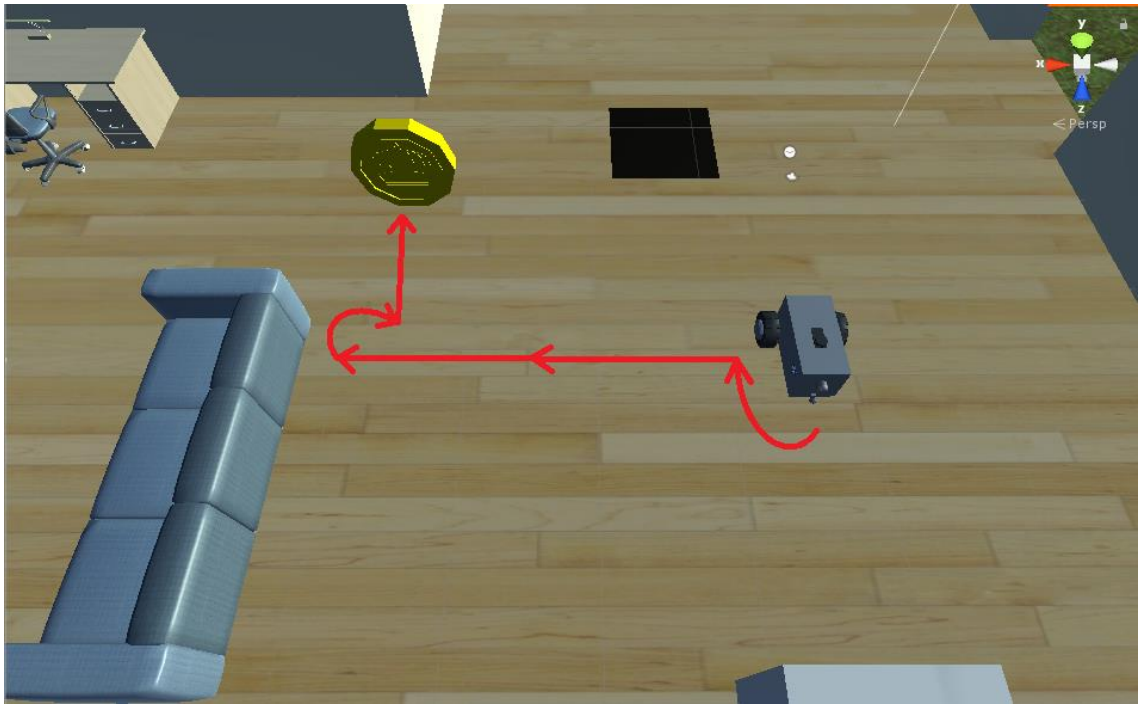


Figura 43: *Ejemplo de ruta hacia un objetivo en el simulador web*

La línea roja representa el camino que debe seguir el robot hasta el objetivo, en este caso una moneda. Mediante la programación visual basada en bloques se debe resolver este circuito, utilizando para ello los bloques de movimiento y de sensores. Una posible solución para llegar al objetivo podría ser la siguiente:

1. Girar 90 grados a la derecha.
2. Avanzar hasta que encontremos un obstáculo (el sillón) con el sensor de contacto, ya que necesitamos llegar hasta ese obstáculo para realizar la siguiente acción, no solo encontrarlo.
3. Girar 90 grados a la derecha.
4. Avanzar hasta encontrar un obstáculo (la moneda) con el sensor de contacto, para situarnos lo más cerca del objeto posible.

Figura 44: *Solución con la programación visual al ejemplo de uso*

Ahora que tenemos el código terminado, debemos descargarlo mediante el botón “Descargar Código” y, posteriormente, cargarlo en el simulador.

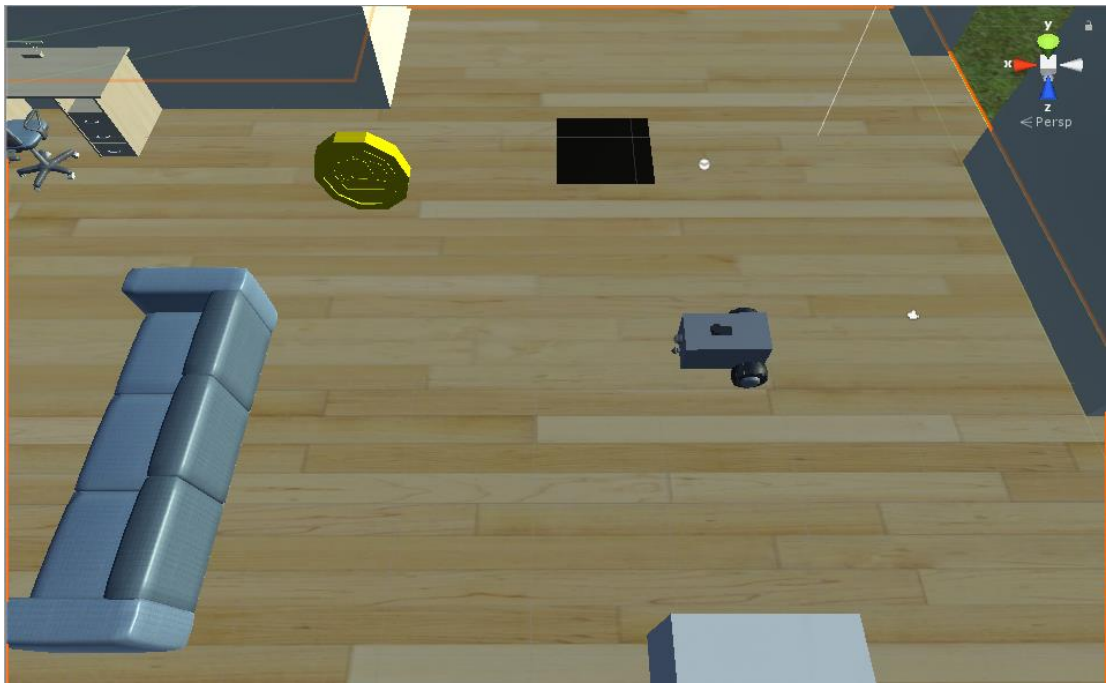


Figura 45: *Visualización de ejecución del código (Paso 1)*

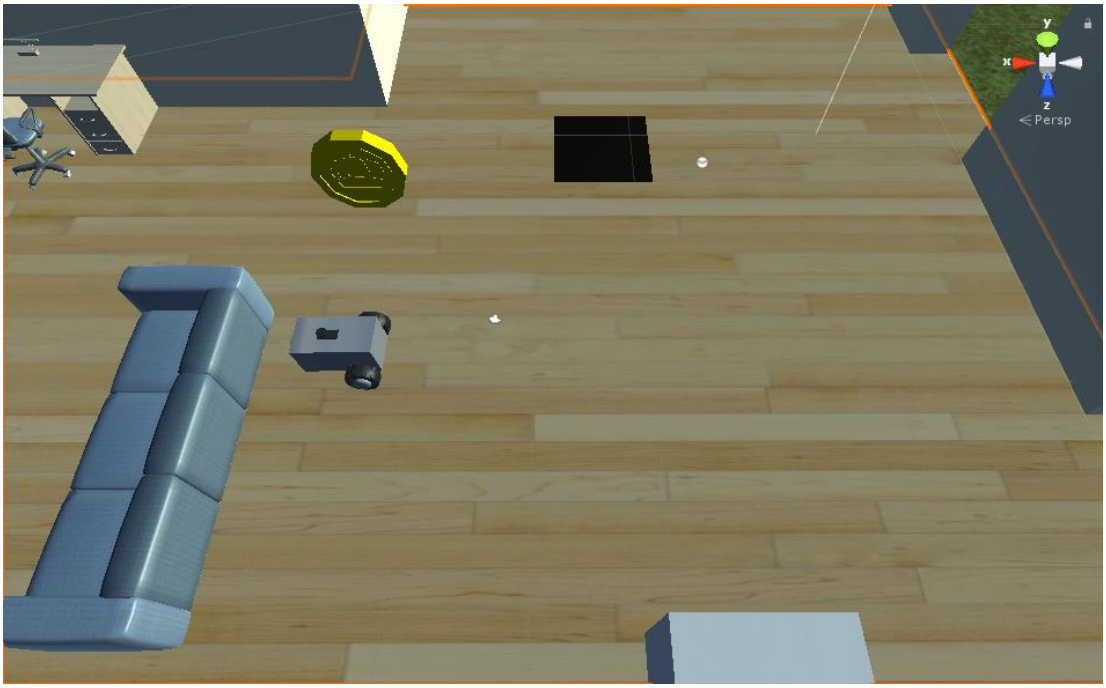


Figura 46: *Visualización de ejecución del código (Paso 2)*

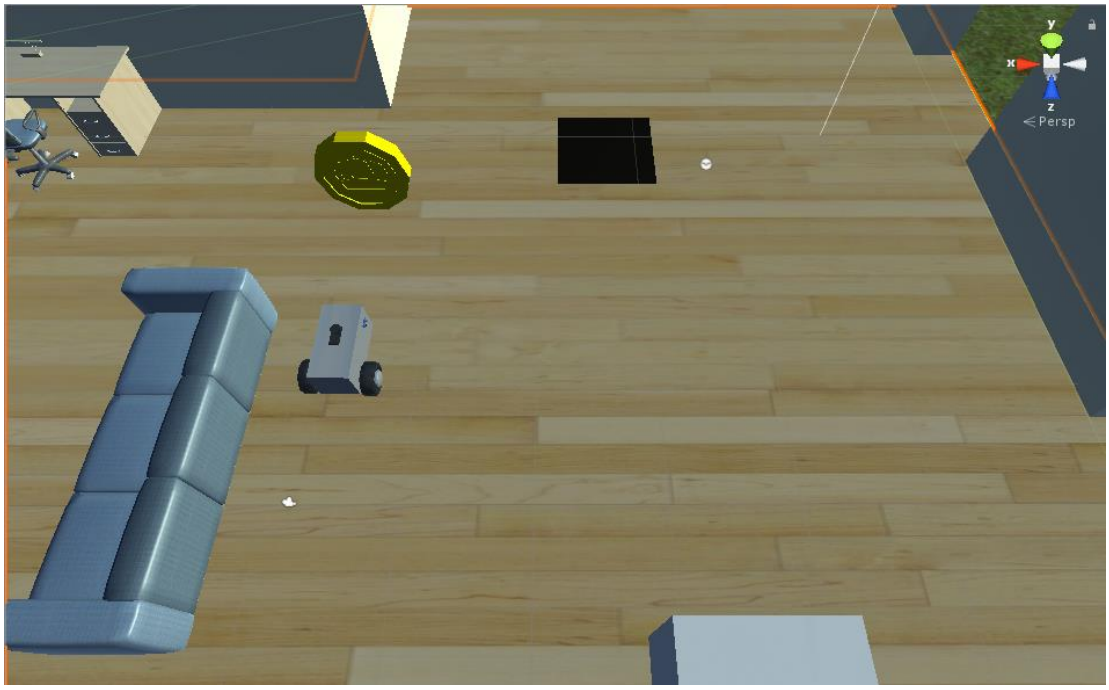


Figura 47: *Visualización de ejecución del código (Paso 3)*

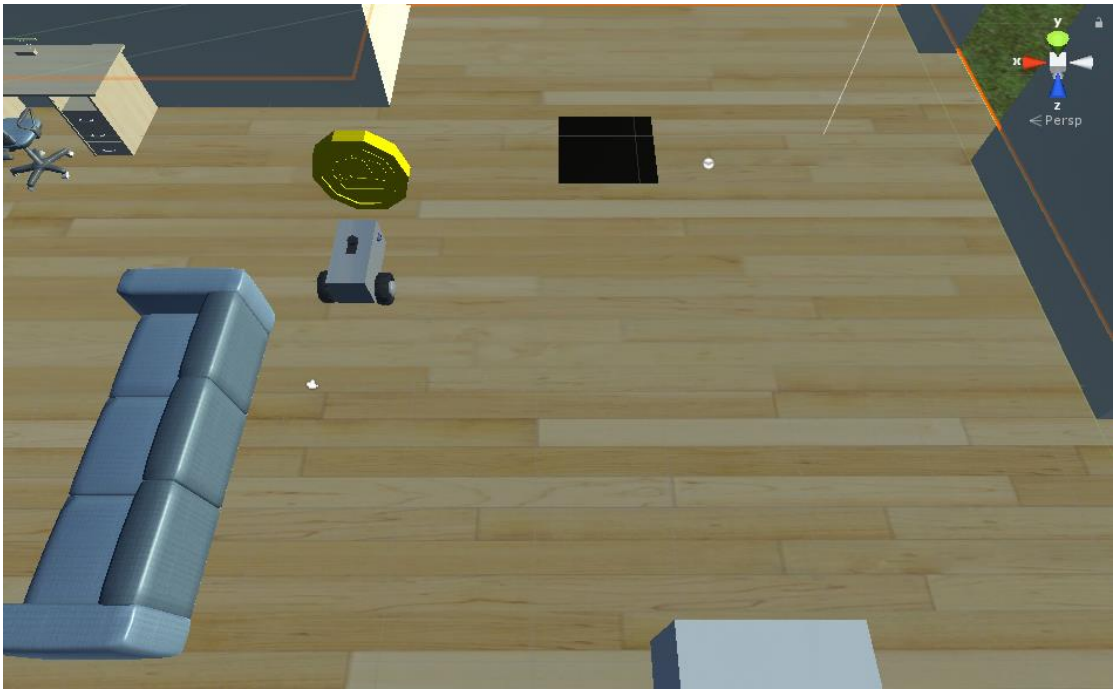


Figura 48: *Visualización de ejecución del código (Paso 4)*

Capítulo 5. Presupuesto

En este capítulo se detallará el presupuesto correspondiente al desarrollo de este proyecto:

5.1 Presupuesto

Actividad	***Horas***	***Precio(€)/Hora***	***Total***
Búsqueda de documentación	*8 h*	*10 €/h*	*80 €*
Estudio de la herramienta Blockly	*25 h*	*10 €/h*	*250 €*
Desarrollo de la aplicación	*120 h*	*18 €/h*	*2.160 €*
Desarrollo del paso de ficheros entre los módulos de la aplicación	*10 h*	*18 €/h*	*180 €*
Desarrollo de la memoria	*50 h*	*15 €/h*	*750 €*
Total	*213 h*		*3.420 €*

Tabla 1: *Tabla de presupuesto*

Capítulo 6. Conclusiones y trabajos futuros

A modo de conclusión, con este Trabajo de Fin de Grado se ha conseguido crear una aplicación con la que crear un lenguaje de programación visual basado en bloques que ha logrado alcanzar los objetivos propuestos inicialmente. La aplicación, destinada a niños de la escuela primaria y secundaria, tiene como función principal mejorar las habilidades promovidas por el pensamiento computacional, lo que conlleva una mejora en habilidades de razonamiento lógico, resolución de problemas, diseño de sistemas, etc. Con ella, incluso los usuarios que no posean conocimientos sobre programación podrán aprender fácil y rápidamente, de una forma práctica, a utilizar este tipo de lenguajes.

Como experiencia personal, el desarrollo de este proyecto me ha ayudado a conocer nuevas herramientas, aplicaciones y lenguajes de programación y a adentrarme un poco más en el campo de la robótica educativa. Anteriormente ya había trabajado con aplicaciones parecidas a lo desarrollado en este proyecto, como las aplicaciones desarrolladas por la empresa LEGO, pero sin entrar en detalles funcionales, a nivel interno de la aplicación, ni en conceptos pedagógicos.

En cuanto a las líneas futuras, actualmente, este proyecto es solo una primera versión de lo que realmente se pretende conseguir. Más adelante, en el módulo de programación visual, se podrían incluir nuevas o mejoradas funcionalidades, por ejemplo:

- Mejora en los bloques desarrollados o creación de nuevos bloques para que el robot realice más acciones. De momento solo hay desarrollados cuatro bloques de movimiento, pero más

adelante, se podrán ir desarrollando más bloques según las necesidades que vayan surgiendo o las funcionalidades que se quieran implementar.

- Mejora del paso de ficheros entre los distintos módulos. Ahora mismo, el paso del fichero del diseño del robot al módulo de programación visual debe hacerlo el usuario de forma manual, por lo que lo mejor sería que existiera un botón en el módulo de programación visual que permitiera cargar automáticamente dicho fichero, sin necesidad de que el usuario tenga que guardarlo en una ubicación determinada cuando haya terminado de configurar el robot. Otra opción podría ser integrar el módulo de programación visual en la plataforma Unity [32], ya que es donde se encuentran desarrollados los otros dos módulos que compondrían la aplicación final.
- Desarrollo de una aplicación web que unifique los tres módulos, para que cumpla con uno de los objetivos planteados, que es la accesibilidad de este proyecto para cualquier escuela que desee utilizarlo sin necesidad de descargar ni instalar ningún software, sino acceder directamente desde una dirección URL.

Capítulo 7. Conclusions and future works

In conclusion, with this Final Degree Project I have managed to create an application with which to create a visual programming language based on blocks that has achieved the initially proposed objetives. The application, aimed at primary and secondary school children, has as main function to improve the skills promoted by computer thinking, which leads to an improvement in logical reasoning skills, problem solving, system design, etc. With it, even users who don't have programming knowledge can easily and quickly learn, in a practical way, to use this type of language.

As a personal experience, the development of this project has helped me to learn new tools, applications and programming languajes and to delve a bit more into the field of educational robotics. Previously I had already worked with applications developed by the company LEGO, but without going into functional details, internally of the application, or in pedagogical concepts.

About the future lines, currently, this Project is only a first versión of what is really intented to be achieved. Later, in the visual programming module, new or improved functionalities could be included, for example:

- Improvement in the developed blocks or creation of new blocks for the robot to perform more actions. At the moment there are developed only four movement blocks, but later, more blocks could be developed according to the needs that raise or the functionalities that want to implement.
- Improvement of the passage of files between the different modules. Right now, the passage of the robot design file to the

visual programming module must be done by the user manually, so it would be better if there were a button in the visual programming module that allow the file to be loaded automatically, without the need for that the user has to save it in a certain location when he has finished configuring the robot. Another option could be to integrate the visual programming module in the Unity platform [32], which is where the other two modules that would compose the final application are developed.

- Development of a web application that unifies the three modules, so that it meets one of the proposed objetives, which is the accessibility of this project for any school that wished to use it without having to download or install any software, but access directly from an URL.

Capítulo 8. Códigos destacables de la aplicación

```
Blockly.Blocks['avanzar'] = {
  init: function() {
    this.appendDummyInput()
        .appendField("Avanzar")
        .appendField(new Blockly.FieldNumber(5, 1), "metros/seg")
        .appendField("metros por")
        .appendField(new Blockly.FieldNumber(5, 1), "segundos")
        .appendField("seg");
    this.setPreviousStatement(true, null);
    this.setNextStatement(true, null);
    this.setColour("#29A2C5");
 this.setTooltip("");
 this.setHelpUrl("");
  }
};
```

Figura 49: *Ejemplo de generación del bloque Avanzar*

```
Blockly.Blocks['sensor_laser'] = {
  init: function() {
    this.appendDummyInput()
        .setAlign(Blockly.ALIGN_CENTRE)
        .appendField("Sensor Láser nº")
        .appendField(new Blockly.FieldTextInput("1"), "num_sensor");
  this.setOutput(true, null);
  this.setColour("#DF9A13");
  this.setTooltip("");
  this.setHelpUrl("");
  }
};
```

Figura 50: *Ejemplo de generación del bloque del sensor láser*

```
Blockly.JavaScript['avanzar'] = function(block) {
  var metros_seg = block.getFieldValue('metros/seg');
  var segundos = block.getFieldValue('segundos');

  var code = 'Avanzar(' + metros_seg + ',' + segundos + ');\n';
  return code;
};
```

Figura 51: *Ejemplo de definición del bloque Avanzar*

```
Blockly.JavaScript['sensor_laser'] = function(block) {
  var num_sensor = block.getFieldValue('num_sensor');

  var num_sensores_laser = Laser[0].children.length;

  // Si hay más de un sensor...
  if (Laser[0].children[0].tagName != "Activo") {

    // Si el usuario introduce un nº mayor al nº de sensores que hay...
    if (num_sensor > num_sensores_laser) {
    window.alert("ERROR: Hay " + num_sensores_laser + " sensores láser");
    } else {
      debugger;
      var code = "sensor_laser_" + num_sensor;
      if (typeof code == 'string') console.log('ES UN STRING');
      else console.log('ES UN NUMERO');
      return code;
    }

  } else {
    if (num_sensor != 1) {
      window.alert("ERROR: Solo hay 1 sensor láser");
    } else {
      var code = "sensor_laser_1";
      return code;
    }
  }
};
```

Figura 52: *Ejemplo de definición del bloque del sensor láser*

```
var xmlhttp = new window.XMLHttpRequest();

xmlhttp.open(null, 'config_robot.xml', false);
xmlhttp.send(null);

var parser = new DOMParser();
var xmlDoc = parser.parseFromString(xmlhttp.responseText, "application/xml");

xmlDoc = xmlhttp.responseXML.documentElement;

var sensoresXML = xmlDoc.getElementsByTagName('Sensores');

for (var i = 0; i < sensoresXML.length; i++) {
    sensores = sensoresXML[i];
    Laser = sensores.getElementsByTagName("Laser");
    ...
}

if (Laser[0].children[0].tagName != "Activo") {
    console.log('Hay ' + Laser[0].children.length + ' sensores láser');
} else {
    console.log('Hay 1 sensor láser');
}

if ((Laser[0].children.length > 1) && (Laser[0].children[0].tagName != "Activo")) {

    var num = 1;
    for (var i = 0; i < Laser[0].children.length; i++) {
        window["Laser" + num + "_Activo"] = Laser[0].children[i].children[0].textContent;
        window["Laser" + num + "_Rango"] = Laser[0].children[i].children[1].textContent;
        window["Laser" + num + "_Precision"] = Laser[0].children[i].children[2].textContent;
        num++;
    }
}
```

Figura 53: *Código para extraer los datos del fichero de configuración del robot (ejemplo para el sensor láser)*

Bibliografía

[1] Carmen San Martín. (2016). 10 beneficios de aprender a programar en edad infantil. Accedido 02/08/2018. Sitio web: https://camptecnologico.com/10-beneficios-programar-edad-infantil/

[2] CodeLearn (2018). Beneficios del pensamiento computacional. Accedido 05/06/2018. Sitio web: https://codelearn.es/beneficios-del-pensamiento-computacional/

[3] Google. Manual Blockly. Accedido 20/03/2018. Sitio web: https://developers.google.com/blockly/

[4] Jorge Roberto Trujillo Cabrera. (2007). Robótica Pedagógica VS Robótica Educativa. 11/06/2018. Sitio web: http://nticsaeiuaemex.blogspot.com/2007/04/robtica-pedaggica-vs-robtica-educativa.html

[5] Robótica Educativa Wikipedia. Accedido 11/06/2018. Sitio web: https://es.wikipedia.org/wiki/Rob%C3%B3tica_educativa

[6] Edukative. Robótica Educativa. Accedido 11/06/2018. Sitio web: https://edukative.es/que-es-la-robotica-educativa/

[7] Robótica en Arequipa - Grupo Educativa. (2011). Robótica Pedagógica y Robótica Educativa: características y diferencias. Accedido 11/06/2016, de Robótica en Arequipa - Grupo Educativa. Sitio web: https://grupoeducativa.blogspot.com/2011/02/robotica-pedagogica-y-robotica.html

[8] Enrique RUIZ-VELASCO SÁNCHEZ. (2008). La robótica pedagógica como disciplina integradora de distintas áreas del conocimiento. En ROBÓTICA PEDAGÓGICA VIRTUAL PARA LA INTELIGENCIA COLECTIVA (17). México:

Universidad Nacional Autónoma de México.

[9] EFE. (2018). China encabezará el estudio obligatorio de robótica en los colegios. Accedido 11/06/2018, de El Economista. Sitio web: http://www.eleconomista.es/tecnologia/noticias/9087563/04/18/China-encabezara-el-estudio-obligatorio-de-robotica-en-los-colegios-.html

[10] EDUCACIÓN 3.0. (2016). Los ciudadanos del futuro y la educación STEAM. Accedido 11/06/2018, de EDUCACIÓN 3.0 Sitio web: https://www.educaciontrespuntocero.com/opinion/los-ciudadanos-del-futuro-la-educacion-steam/33941.html

[11] Gobierno de Canarias. Programa STEAM: Fomento de Vocaciones Científicas y Creatividad. Accedido 11/06/2018, de Gobierno de Canarias Sitio web: http://www.gobiernodecanarias.org/educacion/web/programas-redes-educativas/programas-educativos/steam/

[12] RO-BOTICA. (2017). PRO-BOT Robot infantil programable. Accedido 08/06/2018. Sitio web: http://ro-botica.com/Producto/PRO-BOT/

[13] RO-BOTICA. (2017). InO-Bot Robot infantil programable. Accedido 08/06/2018. Sitio web: https://www.ro-botica.com/Producto/INO-BOT/

[14] RO-BOTICA. (2017). Set LEGO® MINDSTORMS® Education EV3. Accedido 08/06/2018. Sitio web: http://ro-botica.com/Producto/Set-basico-LEGO-MINDSTORMS-Education-EV3/

[15] RO-BOTICA. (2017). Set básico LEGO® Education WeDo 2.0 Bluetooth. 08/06/2018. Sitio web: http://ro-botica.com/Producto/Set-basico-LEGO-Education-WeDo-2/

[16] Scratch. Accedido 12/06/2018. Sitio web: https://scratch.mit.edu/

[17] RO-BOTICA. (2017). mOwayduino. Accedido 08/06/2018. Sitio web: http://ro-botica.com/Producto/robot-mOwayduino-basado-en-ARDUINO/

[18] Eugenio Jacobo Hernández Valdelamar & Humberto Manuel Uribe León. (2002). ¿Qué es la programación visual? En El paradigma de la programación visual (10). México.

[19] Garaje Imagina. (2017). ¿Qué es Scratch? y ¿Para qué sirve?. Accedido 12/06/2018, de Garaje Imagina. Sitio web: https://garajeimagina.com/es/blog/2016/08/22/que-es-scratch-y-para-que-sirve/

[20] Make Block. (2018). mBlock=Scratch+Arduino. Accedido 12/06/2018. Sitio web: https://www.makeblock.es/soporte/mblock/

[21] Kodu. Accedido 11/06/2018. Sitio web: https://www.kodugamelab.com/about/

[22] LEGO. (2018). APP EV3 PROGRAMMER. Accedido 12/06/2018, de LEGO. Sitio web: https://www.lego.com/es-es/mindstorms/apps/ev3-programmer-app

[23] Google. Accedido 20/03/2018. Sitio web: https://blockly-demo.appspot.com/static/demos/blockfactory/index.html

[24] (2013). Como leer xml con javascript. Accedido 10/07/2018. Sitio web: https://social.msdn.microsoft.com/Forums/es-ES/040c7f0f-964c-4c0e-8317-3eafdf585c0e/como-leer-xml-con-javascript?forum=netfxwebes

[25] (2014). Como leer un archivo xml del servidor con JavaScript. Accedido 10/07/2018. Sitio web: https://www.lawebdelprogramador.com/codigo/JavaScript/2762-Como-leer-un-archivo-xml-del-servidor-con-JavaScript.html

[26] Google. Manual Blockly. Accedido 20/03/2018. Sitio web: https://developers.google.com/blockly/guides/overview

[27] Google. Manual Blockly. Accedido 20/03/2018. Sitio web:

https://developers.google.com/blockly/guides/get-started/web

[28] Repositorio GitHub. Accedido 15/04/2018. Sitio web: https://github.com/cityindex/blockly-generator-csharp

[29] Google. Manual Blockly. Accedido 20/03/2018. Sitio web: https://developers.google.com/blockly/guides/configure/web/toolbox

[30] Google. Manual Blockly. Accedido 20/03/2018. Sitio web: https://developers.google.com/blockly/guides/configure/web/toolbox#categories

[31] Google. Manual Blockly. Accedido 20/03/2018. Sitio web: https://developers.google.com/blockly/guides/configure/web/toolbox#dynamic_categories

[32] Unity. Accedido 14/08/2018. Sitio web: https://unity3d.com/es