



Universidad de Murcia

Facultad de Informática

Grado en Ingeniería en Informática

Trabajo Fin de Grado

Integración de un modulo de simulación de robot en el proyecto Descubre

Autor

Alberto López Sánchez

Dirigido por

Juan Antonio Sánchez Laguna

Febrero 2016

Resumen

Extended Abstract

Índice general

Resumen	3
Extended Abstract	5
Índice	8
Lista de figuras	9
Lista de tablas	11
1. Introducción	1
1.1. Situación actual	2
1.2. Electrónica y robótica como herramienta de aprendizaje	3
1.3. Proyecto Descubre la programación	4
1.4. Motivación y enfoque de este proyecto	6
2. Estado del arte	9
2.1. Logo y el robot Turtle	9
2.1.1. Turtle de Logo	10
2.2. Scratch	12
2.2.1. Funcionamiento de Scratch	13
2.3. Aprendiendo con robots	15
2.3.1. Lego Mindstorm NXT/EV3	15
2.3.2. Moway	16
2.3.3. Karel the Robot	18
2.3.4. Robomind	18
2.4. Aprendizaje Online	21
2.4.1. CodeHS	21
2.4.2. Code.org	22

3. Objetivos, metodología y herramientas	23
3.1. Análisis de objetivos	23
3.2. Metodología	24
3.2.1. Tecnologías	24
3.3. Herramientas utilizadas	25
4. Diseño y resolución del trabajo realizado	27
4.1. Creación del mundo	28
4.1.1. Construcción del robot	29
4.1.2. Construcción de circuitos	29
4.1.3. Colisiones	30
4.2. Curvas de Bezier	30
4.3. Integración en Descubre	30
4.3.1. Modificación del lenguaje iJava	30
5. Conclusiones y vías futuras	31
A. Edad escolar en diferentes Sistemas Educativos	33
B. Lenguaje Logo	35
B.1. <i>words, sentences</i> , operaciones y comandos	35
B.2. Variables, comentarios y definiendo nuevas operaciones	36
B.3. Operadores aritméticos	37
B.4. Operadores IF y REPEAT	38
Bibliografía	43
Glosario	45
Acrónimos	47

Índice de figuras

1.1.	Mapa de visualización de eventos de llevados a cabo por el proyecto <i>Hora del código</i> alrededor del mundo. Actualmente 198,473 en todo el mundo, 1,839 en España. Obtenido de [13].	3
1.2.	Sección <i>Crea</i> del proyecto Descubre la programación. Obtenido de [26].	5
1.3.	Salida del programa escrito en iJava ejecutado por el código 1.2. . .	6
2.1.	Gráfico de edad de los usuarios de Scratch. Obtenido de https://scratch.mit.edu/statistics/	13
2.2.	Ejemplo de la interfaz de usuario y el editor de código que ofrece Scratch.	14
2.3.	Entorno de programación en Robomind Academy. Obtenido de [41]. .	19
A.1.	Tabla que muestra los típicos patrones de progresión en el Sistema Educativo Americano. Fuente: U.S. Department of Education, National Center for Education Statistics, Annual Reports Program. Obtenido de http://nces.ed.gov/programs/digest/d11/figures/fig_01.asp	34

Índice de cuadros

2.1. Tabla que muestra los resultados obtenidos en el ITBS por los alumnos que aprendieron a programar (denominado <i>computer</i>) y el grupo de control. Obtenido de [21, p.251].	10
2.2. Resumen de órdenes en Turtle usando el lenguaje Logo.	12
A.1. Comparativa de edades de escolarización en diferentes Sistemas Educativos con respecto a las etapas del Sistema Educativo Español.	33
B.1. Resumen de las operaciones elementales que ofrece el lenguaje Logo.	36

Capítulo 1

Introducción

Como dice M. Lemaire[28], en una sociedad que está incrementando su dependencia a las nuevas tecnologías¹, es imprescindible que las nuevas generaciones desarrollen la habilidad de pensar de manera crítica sobre tecnología.

El *Computational Thinking* (pensamiento computacional)[48], es el arte de pensar como un ordenador. El pensamiento computacional, como se refiere A. Bundy en [11], afecta a investigaciones de casi todas las disciplinas, tanto de ciencias como de humanidades. [...] La informática no solo ha permitido que los investigadores puedan hacer nuevas preguntas, sino también ha permitido aceptar nuevos tipos de respuesta. Por ejemplo, preguntas que requieren el procesamiento de una gran cantidad de datos.

Es necesario comprender la informática y desarrollar el *pensamiento computacional*. Se necesitan desarrollar nuevas tecnologías, nuevo Hardware y Software que automatice tareas largas, complejas y con una alta cantidad de cómputo. Tareas que no siempre los humanos podemos resolver de manera directa. Para conseguir esto, muchas veces es necesario programar.

Pero aprender a programar es mencionado como uno de los 7 grandes retos de la educación informática [34] y diversos estudios [40] muestran que las principales dificultades para un alumno cuando está en el proceso de aprender a programar son (a) como empezar un programa; (b) comprensión de la sintaxis específica del lenguaje de programación; (c) comprensión de la lógica² y (d) problemas a la hora de depurar el código escrito.

Aunque a primera vista aprender a programar pueda parecer una tarea ardua y compleja, tiene sus ventajas. En el estudio realizado por J. Siegmund y otros en [45], podemos ver como los participantes mientras comprendían, analizaban y buscaban errores en pequeños trozos de código, daban claras muestras de estar desarrollando

¹Aquí, con nuevas tecnologías me refiero a productos y proyectos derivados de la Informática y las Tecnologías de la Información y la Comunicaciones (TIC) como puede ser los proyectos que detallamos en la sección 2.

²En este caso me refiero a lógica booleana, o también llamada Álgebra de Boole.

actividad cerebral en regiones del cerebro relacionadas con el procesado del lenguaje, la atención y la memoria de trabajo.

De la misma manera, estudios realizados en niños [12] de entre 6 y 8 años, muestran que estos demostraron mayor capacidad de atención, más autonomía y un mayor placer por el descubrimiento de nuevos conceptos. En la misma linea, un estudio en niños de infantil [18] que utilizaban el Lenguaje Logo demostró que los mismos obtuvieron mejores resultados en pruebas de razonamiento, matemáticas o resolución de problemas. Otro estudio más reciente [29] demuestra que aprender a programar (independientemente del lenguaje) a una corta edad, potencia la creatividad y la habilidad de aprendizaje.

1.1. Situación actual

Actualmente existen muchos proyectos con el fin de introducir la programación como asignatura obligatoria en Educación Primaria y Secundaria³. Estos proyectos vienen respaldados por importantes cambios en los planes de estudios de todo el mundo para enseñar programación[17, 33, 2, 22, 6], marcando una clara tendencia social de incluir la programación en los currículos académicos.

Uno de los proyectos más importantes y con más repercusión a nivel global es Code.org[14]. Propone una serie de herramientas y juegos diseñados especialmente para que los niños aprendan programación mientras juegan con sus personajes favoritos de películas y videojuegos. Todo ello acompañado de una infraestructura para que profesores puedan incluir estos proyectos en las aulas, y padres en sus casas. Su proyecto *Hora del código*[13] consigue reunir a niños de todas las edades una hora al día para que la inviertan en programar. Code.org cuenta con decenas de millones de estudiantes de más de 180 países, disponible en más de 30 idiomas. De manera gratuita, cualquiera puede aprender a programar en eventos que se realizan por todo el mundo (figura 1.1). Code.org está apoyado por grandes compañías y personalidades a nivel global como puede ser Microsoft, Google, el Presidente Barack Obama, Mark Zuckerberg, Bill Gates o Walt Disney Company. En la sección 2.4.2 se abordará más detalladamente las diferentes formas que tiene Code.org de enseñar a programar.

A nivel europeo, la Comisión Europea[15] ha promovido durante el año 2015 la *EU Code Week*[47] como parte de su Estrategia para la Educación y la Formación 2020. Este proyecto consistió en eventos de una semana de duración en la que se

³En este documento se hablará a menudo de alumnos de Educación Primaria, Secundaria o Bachillerato pero no siempre referido al sistema educativo español. Cada país tiene un sistema educativo diferente que varía la edad de escolarización de los alumnos. Por generalizar, se tomará la edad y cursos escolares como referencia cuando se hable de Educación Primaria, Secundaria o Bachillerato. En el Apéndice A se profundiza más en la diferencia de edad en los principales Sistemas Educativos que pueden ser referenciados, directa o indirectamente, en el presente documento.

1.2. ELECTRÓNICA Y ROBÓTICA COMO HERRAMIENTA DE APRENDIZAJE3



Figura 1.1: Mapa de visualización de eventos de llevados a cabo por el proyecto *Hora del código* alrededor del mundo. Actualmente 198,473 en todo el mundo, 1,839 en España. Obtenido de [13].

enseñaba informática y programación en lugares de toda Europa⁴.

1.2. Electrónica y robótica como herramienta de aprendizaje

Existen proyectos como Arduino[7] y Raspberry Pi[24] que proporcionan los elementos y materiales básicos para poder crear nuevos proyectos. Arduino es una placa base con un microprocesador que es capaz de analizar entradas de información (un sensor, un botón o incluso una notificación de Twitter (www.twitter.com) y convertirlo en una respuesta como el activar un robot o encender un led. Todo esto acompañado de un software facil de programar. El proyecto es completamente *Open Source*. Raspberry Pi es un ordenador del tamaño de una tarjeta de crédito. También se provee por defecto un Sistema Operativo *Open Source* pero el sistema está completamente abierto a cambios y mejoras por parte de la comunidad.

En concreto, una de las meta originales de Raspberry Pi es conseguir que los niños aprendan a programar y conozcan como funciona realmente un ordenador. Tanto Arduino como Raspberry Pi pueden adquirirse a un bajo precio, lo que facilita que las instituciones de enseñanza introduzcan estas tecnologías en las aulas como parte de su currículo.

Otro proyecto que también está muy extendido en las aulas es Lego Mindstorm EV3 y NXT[35]. Dos kits de iniciación a la robótica con Lego. Permite al estudiante crear con piezas de lego un robot y programar su comportamiento. Los robots suelen

⁴En España, se realizaron eventos dentro del marco del proyecto *EU Code Week* en Madrid, Sevilla, Murcia, Asturias, Canarias, Cantabria, Zamora, Cataluña, Ceuta, Badajoz, La Rioja, País Vasco y Valencia.

tomar forma de brazo robótico o de vehículo capaz de seguir líneas pintadas en el suelo haciendo uso de sensores. Toda este material robótico viene acompañado de un software para programar el comportamiento de los diferentes componentes del robot y un estilo de programación con bloques[36], similar al usado en Scratch.

En la sección 2.3 se hablará más del uso de la electrónica y la robótica como elementos de enseñanza.

1.3. Proyecto Descubre la programación

Descubre la programación[26] es un proyecto diseñado en la Facultad de Informática de la Universidad de Murcia y tiene como objetivo ayudar a los alumnos de Secundaria y Bachillerato a que desarrollen sus capacidades descubriendo lo que es la informática y aprendiendo a programar. Así como fomentar la inclusión del aprendizaje de la programación en secundaria y bachillerato.

Para ello, en un mismo sitio web, se integra (a) un conjunto de tutoriales de programación; (b) una herramienta que permite programar (figura 1.2) y realizar ejercicios o retos propuestos y (c) una pequeña red social que permite publicar y compartir con el resto de compañeros los programas realizados. Adicionalmente se puede consultar las estadísticas de aprendizaje y tiempo dedicado en la plataforma, tanto por el alumno como por el profesor. De esta manera se permite que los profesores puedan utilizar Descubre en las aulas y realizar un seguimiento de la dedicación y progreso del alumno.

En cuanto a la herramienta para desarrollar programas, el lenguaje utilizado es iJava[42] y ha sido desarrollado por J. A. Sánchez Laguna. iJava es un lenguaje imperativo basado en Java que comparte su sintaxis. Ofrece un enfoque procedural para programar con la Orientación a Objetos como algo opcional. También incorpora un conjunto reducido de funciones de librería clasificadas en los tres grupos siguientes: numéricas, entrada/salida y gráficas⁵.

A modo de ilustración, podemos ver el código 1.1, un programa que imprime por pantalla la cadena "Hello World". En el código 1.2 podemos ver un programa un poco más complejo que dibuja círculos de colores en la pantalla según la posición del ratón. En la figura 1.3 se puede ver la salida de este último programa.

```

1 void main() {
2     print("Hello World");
3 }
```

Código 1.1: Programa básico en iJava imprimiendo la cadena "Hello World".

⁵Si el lector está interesado en aprender más sobre este lenguaje, puede consultar [42] o [27].

```

1 void main() {
    //repetimos en bucle la función 'draw'
    animate(draw);
}

5
void draw() {
    //coloreamos el círculo según la posición del ratón
    fill(mouseX, mouseY, 0); //valores RGB
    //dibujamos una ellipse con radio 50
    ellipse(mouseX, mouseY, 50,50);
11 }

```

Código 1.2: Programa en iJava que dibuja un círculo de un color diferente según la posición en la pantalla en la que se encuentra el ratón.

La sintaxis de iJava se parece a muchos de los lenguajes modernos y más utilizados (al ser un subconjunto de la sintaxis de Java), lo cual simplifica el aprendizaje cuando se intenta aprender un nuevo lenguaje. También, gracias a la librería gráfica y matemática, los programas se simplifican mucho en cuanto a complejidad y longitud. De esta manera se consigue aligerar la carga de trabajo que tiene que realizar el alumno para conseguir hacer un programa vistoso, haciendo que la actividad de programar sea más atractiva.

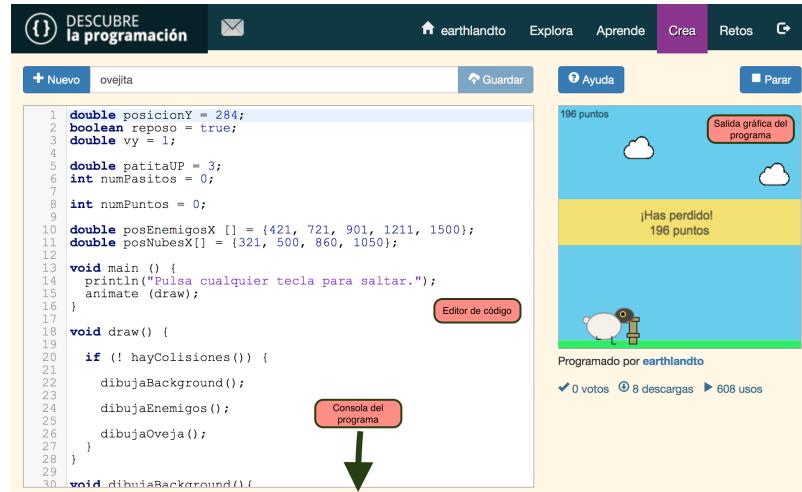


Figura 1.2: Sección *Crea* del proyecto Descubre la programación. Obtenido de [26].



Figura 1.3: Salida del programa escrito en iJava ejecutado por el código 1.2.

1.4. Motivación y enfoque de este proyecto

Las nuevas tecnologías están haciendo que cada vez más gente de todas las edades se interese por la informática, y más concretamente por la programación. Poco a poco la informática deja de ser cosa de un grupo selecto de gente que entiende su funcionamiento. Como bien argumenta J. Wing en [48] y [49], se debería extender el pensamiento computacional en ambientes pre-universitarios y exponer a los alumnos al pensamiento y a los métodos computacionales.

Como ya hemos comentado anteriormente, existe un movimiento que pretende introducir la informática y la programación en las aulas puesto que se han demostrado los beneficios de enseñar a programar en una edad escolar temprana.

Es en este momento cuando se hace imprescindible tomar decisiones acertadas. Aprender a programar en edades pre-universitaria debe ser algo accesible a cualquier estudiante, independiente de su condición o los antecedentes del mismo. Igualmente, los conceptos de programación deben ser presentados de manera incremental, empezando por los más simples para luego ampliar a conceptos más complejos, como defiende L. Fernandez y otros en [20].

Este proyecto se enfocará desarrollando un simulador de robot de dos ruedas en el que los alumnos puedan programar su comportamiento. Se ofrece una librería de funciones para simplificar la interacción con el mismo y conseguir cierta funcionalidad extra que simplifique la tarea de comprensión y usabilidad del simulador.

El simulador se integrará en la plataforma Descubre la programación, mencionada en la sección 1.3. Esto supone que el estudiante programará el comportamiento del

robot simulado en iJava. El robot estará dentro de un circuito con distintos elementos con los que podrá interactuar. Asimismo, el robot dispondrá de una serie de sensores para poder recibir información del mundo que le rodea.

Al incluir el simulador como un módulo de Descubre, se pretende reforzar el esfuerzo por parte de sus creadores de hacer llegar la programación al mayor numero de estudiantes pre-universitarios posible, proponiendo una alternativa atractiva y que añade una componente más de entretenimiento a la actividad de programar. Por otra parte, permite que el alumno asimile conceptos de robótica de manera transparente. Al utilizar una plataforma web y de libre acceso, también se busca eliminar la barrera que puede suponer realizar una inversión en material electrónico como puede ocurrir con Arduino, Raspberry Pi o Lego.

Igualmente, los estudiantes que ya están usando la plataforma Descubre, podrán trabajar los conceptos de programación (bucles, condiciones, variables y funciones, entre otros) en un entorno diferente, renovando así su interés en otra actividad y aprender jugando.

En el capítulo 2 se analizaran cuales son las alternativas que existen actualmente que trabajan en esta linea. En el capítulo 4 se verá como se ha desarrollado la idea principal y que resultados se han obtenido. Por último, en el capítulo 5 se analizarán los objetivos conseguidos y que ofrece mi propuesta en comparación a los proyectos actuales.

Capítulo 2

Estado del arte

A nivel global y desde hace varias décadas, existe una gran cantidad de proyectos con la única intención de enseñar, a alumnos de Educación Primaria, Secundaria y Bachillerato, diferentes aspectos de la informática como lo es la programación [43, 14, 4], la robótica [41, 19] e incluso electrónica (con Arduino[7], Raspberry Pi[24] o Lego Mindstorm[36]). La mayoría de estos proyectos promueven una enseñanza independiente y autodidacta bajo un entorno on-line y gratuito. De esta manera, el alumno puede aprender a su propio ritmo y desde cualquier parte del mundo.

En las siguientes secciones se hablará del Lenguaje Logo[32] y el proyecto Turtle[1], un proyecto que lleva décadas activo, con la principal finalidad de enseñar programación y matemáticas a niños. También se estudiarán los diferentes proyectos que existen actualmente y que promueven una enseñanza a niños en entornos web. Por último, se analizarán los diferentes proyectos que enseñan programación jugando y que utilizan robots y/o simuladores.

2.1. Logo y el robot Turtle

El lenguaje Logo, desarrollado a finales de la década de los 60 y basado en Lisp, fue diseñado como una herramienta de aprendizaje a niños. Todas sus características -interactividad, modularidad, extensibilidad, flexibilidad en los tipos de datos- persiguen esta meta. El lenguaje Logo fue uno de los primeros proyectos en emprender la difícil tarea de enseñar a programar a niños de Primaria.

Como se explica en la página oficial del proyecto Logo[32], durante la década de los 70, en el Massachusetts Institute of Technology (MIT) y diferentes centros de investigación europeos, se llevaron a cabo investigaciones sobre el uso del Lenguaje Logo en pequeños grupos de alumnos de Educación Primaria.

A pesar de los diversos estudios que se han realizado a lo largo de los años, no se han conseguido obtener unos resultados claros sobre la posible ventaja de enseñar

a programar a niños de Primaria y Secundaria con Logo. En [21], Feurzeig y otros consiguen una leve mejoría en la nota obtenida en el Test de Habilidades Básicas de Iowa (Iowa Test of Basic Skills, o ITBS)¹. En la tabla 2.1 se puede ver como la nota global del grupo que aprende a programar (denominado *computer*) es de 114 puntos más que el curso anterior, mientras que la obtención del grupo de control es solo de 6 puntos. Aún así, se puede ver como la nota del grupo de control sigue siendo mayor que la del grupo *computer*. Ante éste hecho, Feurzeig concluye que el grupo de control no representaba muy bien al grupo *computer* y que, por tanto, la comparación perdía validez. Más tarde, Pea y Kurland en [39] concluyen que, tras el estudio en dos cursos separados de alumnos, aprender a programar no conseguía mostrar ningún beneficio claro en el rendimiento de los alumnos en comparación al grupo de control. Poco más tarde, Moss [37] obtiene evidencias de la relación en el aprendizaje de Logo con el desarrollo de conceptos primitivos que más adelante se enlazarían con la álgebra básica.

Number of Correct Answers in ITBS			
		Range for <u>Individual Students</u>	<u>Grand Total</u>
7th Grade (1968)	(Computer)	166 - 298	2896
	(Control)	214 - 371	3174
8th Grade (1969)	(Computer)	144 - 305	3010
	(Control)	209 - 382	3180

Tabla 2.1: Tabla que muestra los resultados obtenidos en el ITBS por los alumnos que aprendieron a programar (denominado *computer*) y el grupo de control. Obtenido de [21, p.251].

[En el Apéndice B se hace un breve estudio del lenguaje Logo y las principales características que posee el lenguaje.](#)

2.1.1. Turtle de Logo

El proyecto *Turtle* es el proyecto más popular del lenguaje Logo. Nació como una criatura robótica, arreglada para que pareciera una tortuga, que se movía por el suelo con un rotulador atado que pintaba el suelo. Más tarde evolucionó a una

¹El Iowa Test of Basic Skills (ITBS), es un test que se realiza anualmente siguiendo una serie de estándares a nivel de estado para medir el rendimiento académico de los alumnos en materias como: vocabulario, ortografía, álgebra, conceptos aritméticos y comprensión de tablas y gráficos, entre otros.

imagen de una tortuga que se movía por la pantalla pintando según se desplazaba. Turtle se utiliza ampliamente desde hace décadas para enseñar a programar a niños de Primaria y Secundaria conceptos de programación, pero también de matemáticas y geometría[3, 10].

A pesar de su larga vida, el proyecto Turtle ha sabido mantenerse vivo y se ha modernizado en proyectos más recientes como Turtle Academy [5] o Curly Logo [30].

Programando el movimiento de Turtle en el lenguaje Logo

A parte de toda la funcionalidad que ofrece Logo como lenguaje, Turtle aporta una librería para poder controlar a nuestra criatura que pintará la pantalla. Ésta suele ser tradicionalmente representada con una tortuga o una flecha. Principalmente, se podrá controlar el movimiento de nuestra tortuga y el giro de ésta. También se le podrá ordenar que deje de pintar para permitir movimiento por la pantalla sin rastro. La mayoría de estos comandos pueden abreviarse para simplificar la escritura.

El resumen sobre la funcionalidad de Turtle que se muestra en las secciones siguientes es un compendio formado a partir de [46], [5] y [3].

Moviendo a Turtle

La tortuga siempre se moverá según el eje de coordenadas. El punto (0, 0) se encontrará normalmente en medio de la pantalla y es la posición inicial de la tortuga. El eje de coordenadas y crece positivamente en dirección Norte (hacia arriba).

Los comandos `forward` y `backward` mueven la tortuga hacia delante y atrás, respectivamente, según la posición en la que esté mirando. Los comandos `setx`, `sety` y `setxy` se utilizan para establecer la tortuga en el eje de coordenadas `x`, `y` o en ambos, respectivamente.

La orden `home` devuelve a la tortuga a la posición (0, 0) de la pantalla. Adicionalmente, se puede ocultar o mostrar (si ya está oculta) la tortuga con las órdenes `showturtle` y `hideturtle`.

Otra función muy importante de Turtle, es la opción de girarla. Podemos decidir el giro con las instrucciones `right` y `left` y el ángulo de giro (el valor tiene que ser un número positivo). Un ángulo de giro de 380° sería equivalente a girar la tortuga 20°.

Pintando con Turtle

Hasta ahora, cualquier desplazamiento de Turtle por la pantalla dejaba una linea o rastro. Con la orden `penup`, la tortuga dejará de pintar hasta que se ejecute la orden `pendown`. También se puede ejecutar la instrucción `clearscreen` que limpiará la pantalla de cualquier rastro dejado por la tortuga.

La instrucción `label` acepta un argumento y mostrará por pantalla su contenido. El argumento debe ser de tipo *word* o *sentence*².

Resumen de instrucciones de Turtle

En la tabla 2.2 se muestra un resumen de las órdenes que puede recibir Turtle.

Orden	Abreviatura	Argumentos
forward	fd	Longitud del movimiento
backward	bk	Longitud del movimiento
home	-	-
setx	-	Nueva coodenada X
sety	-	Nueva coodenada Y
setxy	-	Nueva coodenada X e Y
right	rt	Ángulo en el sentido de las agujas del reloj
left	lt	Ángulo en el sentido contrario a las agujas del reloj
penup	pu	-
pendown	pd	-
clearscreen	cs	-
label	-	Una <i>word</i> o <i>sentence</i>
showturtle	st	-
hideturtle	ht	-

Tabla 2.2: Resumen de órdenes en Turtle usando el lenguaje Logo.

2.2. Scratch

Scratch[44] es un proyecto creado en el seno del Lifelong Kindergarten Group en el MIT Media Lab, Massachusetts Institute of Technology (MIT). Permite programar juegos interactivos, animaciones y programas de todo tipo en un entorno web. Todo esto apoyado por una gran comunidad de usuarios y profesores que forman una enorme red social de más de 9.5 millones de usuarios registrados y casi 13 millones de proyectos compartidos actualmente³.

Scratch está orientado a enseñar a programar a niños de Primaria y Secundaria y como se puede apreciar en la figura 2.1 la mayoría de usuarios de Scratch cumplen dicho perfil, la edad está en torno a los 12 y 15 años.

²Para más información sobre los tipos de datos *words* y *sentences*, como se forman y cual es su funcionamiento, se puede consultar la sección B.2.

³En la página oficial de Scratch se pueden obtener las estadísticas de uso y participación (<https://scratch.mit.edu/statistics/>).

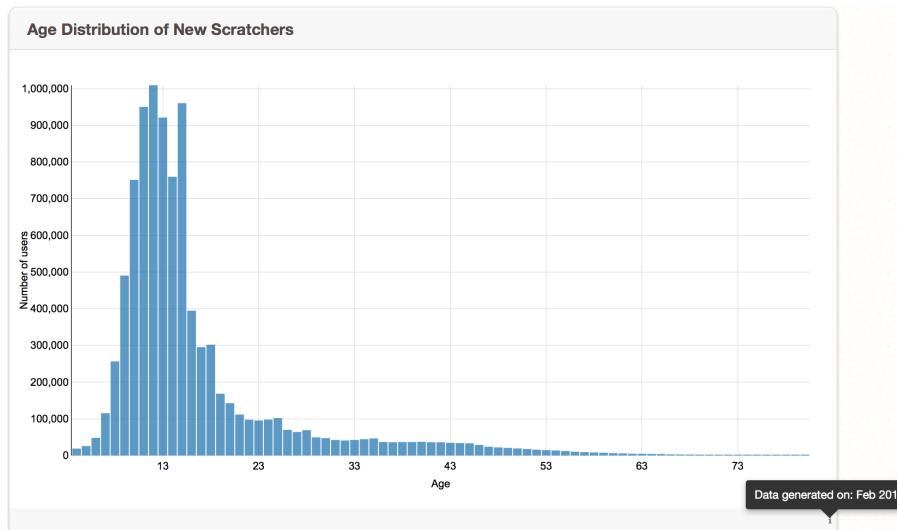


Figura 2.1: Gráfico de edad de los usuarios de Scratch. Obtenido de <https://scratch.mit.edu/statistics/>.

2.2.1. Funcionamiento de Scratch

La idea básica para programar con Scratch es la de crear el flujo del programa mediante la unión de bloques que representan las diferentes funcionalidades del lenguaje (crear o modificar variables, bucles, condiciones, funciones de librería, etc). Existen dependencias entre la forma de los bloques, de esta manera, solo unos bloques *encajarán* con los que tengan una forma compatible. De esta manera, se consigue una relación visual y directa por parte del programador entre los bloques que puedes unir, y sobretodo, **de lo que no puedes hacer**.

En general, la plataforma Scratch tiene tres grandes componentes:

1. **Un editor visual** que presenta un conjunto de bloques cuya forma ofrece pistas de como pueden encajar los diferentes bloques.
2. **Un interprete** que puede traducir y ejecutar el flujo de código que forman los diferentes bloques unidos entre sí.
3. **Una interfaz** que muestra la salida (tanto de texto como gráfica) del código ejecutado por el interprete. También permite controlar la entrada del ratón, de las teclas, etc.

En la figura 2.2 podemos ver un ejemplo del editor y la interfaz así como de un programa de ejemplo que mueve al personaje seleccionado por la pantalla, cambiado su color.

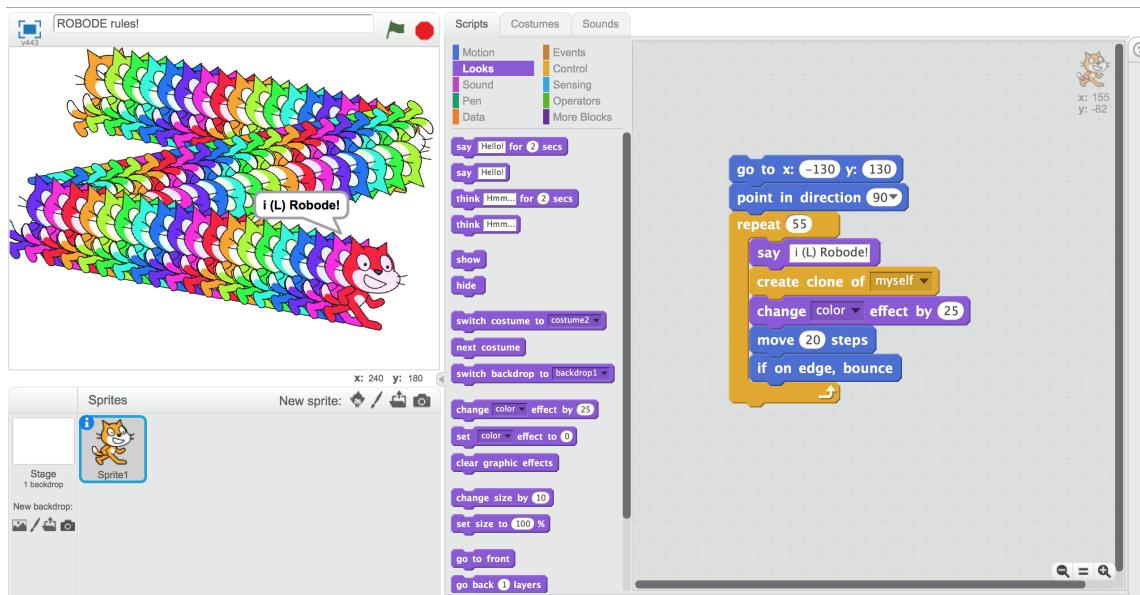


Figura 2.2: Ejemplo de la interfaz de usuario y el editor de código que ofrece Scratch.

Cuando se programa con Scratch, la meta final es la de animar uno o más elementos por la pantalla. Estos elementos son los *sprites* (imágenes), los cuales vienen predefinidos en una galería o pueden ser proporcionados por el propio usuario. Por tanto, los bloques irán enfocados a otorgar movimiento al *sprite* en cuestión.

En cuanto a los tipos de bloques, actualmente están divididos en 11 categorías distintas:

- **Movimiento:** Incluye funciones para mover, girar o establecer la posición y ángulo de giro del *sprite*.
- **Apariencia:** Esta sección contiene funciones tanto para modificar la apariencia del *sprite* como para mostrar mensajes en la pantalla a modo de diálogos de los *sprites*.
- **Sonido:** Funciones para controlar el sonido de la aplicación que se está creando con Scratch. Permite reproducir sonidos precargados en la plataforma o subidos por el programador, diferentes instrumentos o establecer el volumen de la aplicación.
- **Pincel:** Permite pintar el escenario de forma similar a Turlte, dejando un rastro por donde se va desplazando (para más información sobre Turtle, consultar sección 2.1.1).

- **Datos:** Creación de variables y listas y modificación de los valores de las mismas.
- **Eventos:** Bloques para detectar y reaccionar a ciertos eventos como el click de un ratón sobre un *sprite* o el teclado.
- **Control:** Incluye bloques con las condiciones y bucles clásicos.
- **Percepción:** Esta sección contiene funciones para detectar eventos de teclado y ratón. También provee funciones para establecer lapsos de tiempo.
- **Operadores:** Bloques para utilizar operadores matemáticos.
- **Bloques propios y Extensiones:** Scratch permite crear tus propios bloques y usar extensiones integradas con la aplicación como *Lego WeDo* y *PicoBoard*.

Como conclusión, Scratch es un entorno completo y **autosuficiente** con una gran y compleja funcionalidad que permite a niños y mayores realizar aplicaciones sin necesidad de saber programar pero aprendiendo conceptos básicos como la repetición, el control de instrucciones y como llevar a cabo una animación.

2.3. Aprendiendo con robots

En secciones anteriores se han analizado dos formas de aprender a programar. Logo, junto a Turtle, fue el primer lenguaje de programación que se creó para enseñar a programar a niños de Primaria y Secundaria. Por otra parte, Scratch es actualmente un gran promotor del *Computational Thinking* con una cantidad considerable de usuarios activos.

No obstante, existen otros enfoques para enseñar programación y desarrollar el pensamiento computacional. La robótica es una opción muy extendida y que atrae a partes iguales a alumnos y docentes. En las siguientes secciones se estudiaran diferentes alternativas que utilizar tanto simuladores como robots reales.

2.3.1. Lego Mindstorm NXT/EV3

Como se ha mencionado en la sección 1.2, el proyecto Lego Mindstorm[35] está orientado a enseñar robótica a estudiantes de Primaria y Secundaria. Lego proporciona dos kits para crear tu propio robot: NXT y EV3. Lego Mindstorm NXT⁴ fue lanzado en 2006, y su siguiente versión, Lego Mindstorm EV3, salió al público en 2013. Tanto la versión NXT como EV3 son ampliamente utilizadas y existen una gran cantidad de competiciones que fomentan su uso.

⁴Actualmente, Lego Mindstorm NXT tiene soporte por parte de Lego hasta finales del año 2015.

Cada kit está compuesto por elementos Software para poder programar el comportamiento del robot y elementos Hardware como sensores, accesorios, piezas de lego y el *Brick* (ladrillo). El *Brick* es un aparato con el que se pueden programar comportamientos básicos a las piezas conectadas al mismo, pero principalmente es el cerebro de todo el robot. Las piezas son mayormente compatibles entre los dos sistemas, la diferencia radica en el *Brick*.

En concreto, EV3 viene acompañado del *Lego Mindstorm EV3 Home Edition Software* y su *brick*, el *Brick P EV3*. De esta manera se puede programar instrucciones al robot con bloques (o iconos) que recuerdan a Scratch. Los bloques representan diferentes funciones del lenguaje y el programador debe conectar dichos bloques para crear el flujo de ejecución del programa. Lego también ofrece *EV3 Programmer*, una versión gratuita y simplificada (con menos funcionalidad y bloques) para programar los robots de Lego.

Los bloques se dividen en: bloques de acción, bloques de flujo, bloques de sensores, bloques de operación de datos y bloques avanzados. Los bloques de acción permiten controlar la rotación de los motores y las luces del *Brick P EV3*. Los bloques de flujo definen cual será el flujo del programa, cuyo inicio se define con el *bloque de inicio*. Los bloques de sensores, entre otras funciones, controlan los sensores táctiles y de color conectados al *brick*. Los sensores de operación de datos permiten definir variables, realizar cálculos matemáticos u obtener valores aleatorios. Por último, los bloques avanzados permiten administrar ficheros o establecer conexiones Bluetooth.

Por otra parte, existen muchas **aplicaciones de terceros no oficiales** que permiten utilizar lenguajes de programación de alto nivel para controlar el robot, como pueden ser Python, Lua, .Net, C/C++, Java o Robomind (del cual hablaremos en la sección 2.3.4).

2.3.2. Moway

El robot Moway[19] es un robot desarrollado como herramienta educativa. Moway se compone por un conjunto de sensores y actuadores que le permiten recibir información del mundo real y reaccionar a la misma según las instrucciones que se le hayan proporcionado.

Los actuadores de los que dispone el robot Moway son: dos ruedas, luces LEDs y un altavoz. Por otra parte, los sensores de los que dispone Moway son:

- Un sensor de luz que le permite detectar el nivel de luz ambiente.
- 2 sensores de linea, mirando hacia el suelo, que detectan diferencias de color.
- 4 sensores de obstáculos en la parte frontal del robot que detectan la distancia a los mismos.

- Un micrófono para detectar el nivel de ruido.
- Un sensor de temperatura.
- Un acelerómetro que detecta inclinación y fuerzas.

La aplicación gratuita Moway World permite crear diagramas de flujo para programar el comportamiento del robot, con funciones para activar y controlar los diferentes actuadores del robot Moway en función de la información obtenida por los sensores. También permite programar su funcionamiento con Scratch, el cual se ha analizado en la sección 2.2.

Moway tiene varios módulos que pueden ser añadidos para ampliar su funcionalidad o manejo, como son: módulos de radiofrecuencia, placas Raspberry Pi, tarjetas wifi o una cámara para captar imágenes en tiempo real. También se ofrece una alternativa a Moway de código abierto, llamada Mowayduino. Mowayduino está basado en Arduino y tiene toda la funcionalidad y potencia que ofrece Arduino. Los kits que se disponen inicialmente para Mowayduino son los mismos que para Moway. Para programar un Mowayduino se proporciona el entorno de programación oficial de Arduino.

Un uso común de Moway es como robot *sigue-lineas*. Los sensores de linea de los que dispone proporcionan información para poder conocer el estado del robot con respecto a una linea, es decir, si está sobre una linea o no. Al tener 2 sensores, puede conocer si se está saliendo de la linea (quizás porque esta realiza una curva) y en la dirección en la que esto ocurre. Por ejemplo, si el sensor derecho deja de detectar la linea, quiere decir que: o la linea está realizando una curva hacia la izquierda o que el robot Moway está moviéndose hacia la derecha fuera de la linea (lo cual ocurre en ambos casos, si se analiza la situación desde el punto de vista de la linea). Por tanto, al estar saliéndose de la linea por la derecha, el robot Moway tendrá que moverse hacia la izquierda para mantenerse dentro de la misma.

En general, Moway ofrece un robot para programarlo y configurarlo junto a un entorno de programación, como puede ser Scratch. Es un buen inicio en la robótica pero su software privativo deja las opciones de ampliación y de creación de complementos y terceras partes un poco limitadas.

Moway, junto a Robomind, será uno de los pilares en los que se basará el simulador que se desarrollará, en especial la parte que tiene que ver con las características del simulador.

2.3.3. Karel the Robot

Karel⁵ es un lenguaje de programación creado por R. Pattis en su libro *Karel the robot: A gentle introduction to the Art of Programming* (El robot Karel: Una introducción gentil al arte de programar) en 1981[38]. En él, Pattis enseña conceptos de programación siguiendo una metodología *Early bird*[8], que consiste en enseñar primero los conceptos más importantes.

Karel se mueve en un escenario discreto con celdas y con el que se puede interactuar. Las órdenes básicas del robot Karel son: move para avanzar una posición en la dirección que está mirando; turnLeft para girar 90° en sentido contrario a las agujas del reloj; putBeeper y pickBeeper para depositar o recoger, respectivamente, un *beeper* (localizador) en la posición en la que se encuentra Karel; y por último, turnOff, para apagarse a sí mismo y finalizar la ejecución del programa.

Adicionalmente, Karel también proporciona instrucciones para comprobar si hay obstáculos o un *beeper* en la posición que se encuentra. Toda esta funcionalidad viene acompañada de instrucciones de control, bucles y operaciones aritméticas.

2.3.4. Robomind

Robomind Academy[41] es una plataforma que busca potenciar y extender el *Computational Thinking* entre alumnos de Educación Primaria y Secundaria. Sus creadores opinan⁶ que la informática debería tener el mismo estatus que las matemáticas o la lengua, algo esencial en la formación del estudiante. Argumentan que, a través del *Computational Thinking* se podrán realizar avances en el cuidado de la salud o la industria, algo totalmente necesario en pleno Siglo XXI.

Actualmente, Robomind ofrece un entorno de programación multiplataforma: aplicación de escritorio, entorno web y aplicación para Tablet. En la figura 2.3 podemos ver un ejemplo de como es el entorno de programación que propone Robomind.

El proyecto en sí está orientado a que profesores y alumnos trabajen conjuntamente en desarrollar las actividades, permitiendo que los profesores puedan realizar un seguimiento del avance y trabajo de los alumnos a cargo. También permite que los alumnos puedan registrarse independientemente y aprender a su propio ritmo. Los cursos están divididos entre alumnos de Educación Primaria y Secundaria.

Las soluciones antes mencionadas son **de pago**. Dependiendo de si eres profesor o estudiante y de si quieres (o no) utilizar la versión de escritorio, se ofrecen distintos planes de compra. A parte, Robomind ofrece planes gratuitos para alumnos con

⁵El nombre de Karel the robot es nombrado en honor a Karel Čapek, un escritor checo que introdujo el término *robot* por primera vez.

⁶En la sección *Visión* de la página web oficial de Robomind Academy se puede ver cuales son los principios y metas detrás del proyecto (<https://www.robomindacademy.com/go/robomind/about#vision>). Consultado el 2 de febrero de 2016.

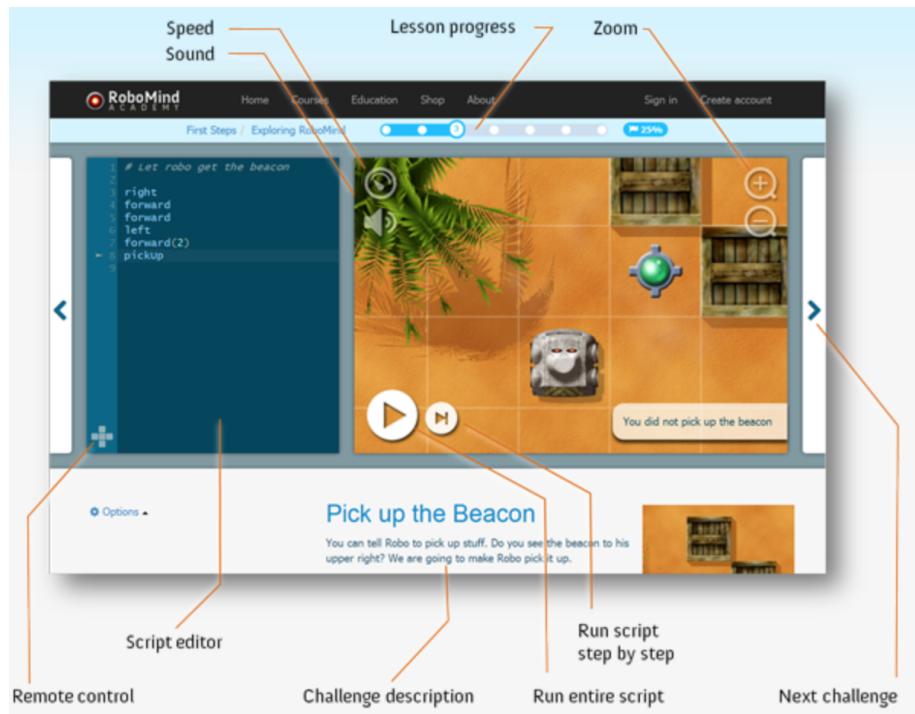


Figura 2.3: Entorno de programación en Robomind Academy. Obtenido de [41].

características más limitadas o una cantidad inferior de cursos. Adicionalmente, Robomind participa en el proyecto *La Hora del Código*[13], ofreciendo una introducción a Robomind completamente gratuita. Una de sus funciones más interesantes es la de poder traducir el código escrito en Robomind para poder utilizarlo con tu propio Lego Mindstorm⁷.

Por último, tanto la interfaz, la página web, la aplicación o los tutoriales está completamente en inglés, ofreciendo una cierta barrera de entrada a todo aquel interesado que no sea angloparlante.

Programa con Robomind Academy

En Robomind se plantea la programación de un robot simulado que está dentro de un escenario. Este escenario es una cuadrícula por la que se mueve el robot. El robot puede interactuar con el escenario pintando o limpiando el suelo, recogiendo y dejando objetos, o comprobando el estado de su casillas adyacentes (si hay algún objeto, obstáculo o está pintado).

⁷Ya se ha hablado de Lego Mindstorm en la sección 2.3.1. Para más información sobre Lego Mindstorm NXT se puede acudir a [35].

A continuación vamos a enumerar las funciones básicas para controlar el movimiento o acciones del robot de Robomind⁸. Adicionalmente, Robomind también permite usar definición de variables y nuevas funciones, estructuras de control y bucles, típicos de cualquier lenguaje de programación moderno.

- Moviendo el robot:
 - `forward` o `forward(n)`: Mover el robot 1 o n pasos hacia el frente.
 - `backward` o `backward(n)`: Mover el robot 1 o n pasos hacia el atrás.
 - `left` o `left(n)`: Girar el robot 90° hacia la derecha, 1 o n veces.
 - `right` o `right(n)`: Girar el robot 90° hacia la derecha, 1 o n veces.
- Interactuando con el escenario:
 - `pickUp`: Coger el objeto que se encuentra delante del robot.
 - `putDown`: Dejar el objeto justo delante del robot.
 - `eatUp`: *Comer* el objeto en frente del robot. Implica destruirlo y no poder volver a interactuar con él.
 - `paintWhite`: Pinta un rastro blanco por las cuadrículas por las que se desplazará el robot.
 - `paintBlack`: Pinta un rastro negro por las cuadrículas por las que se desplazará el robot.
 - `stopPainting`: Deja de pintar.
- El robot puede comprobar, de manera independiente, el estado de las casillas adyacentes a sí mismo, excepto hacia atrás. Para representar cada dirección (*front*, *left* y *right*), se usará una *X*. Todas estas instrucciones devuelven un valor *booleano*.
 - `XIsClear`: Comprueba si la casilla *X* está vacía.
 - `XIsObstacle`: Comprueba si en la casilla *X* hay un obstáculo, generalmente una pared o un objeto que impide su avance.
 - `XIsBeacon`: Comprueba si en la casilla *X* hay un objeto (con el que podrá interactuar).
 - `XIsWhite`: Comprueba si la casilla *X* está pintada de blanco.
 - `XIsBlack`: Comprueba si la casilla *X* está pintada de negro.

⁸Para una información más detallada se puede consultar <https://www.robomindacademy.com/go/robomind/help>.

Como se puede ver, Robomind está formado por elementos de *Karel the robot*, como lo es el movimiento en un escenario discreto y la forma de interactuar con el entorno, y por *Turtle* de Logo y su capacidad de dejar un rastro por donde va pasando.

2.4. Aprendizaje Online

En la sección anterior se han analizado diferentes alternativas para aprender a través de la robótica. Algunas de estas alternativas tienen una gran componente Hardware, como puede ser Lego Minstorm. Por otra parte, Robomind, como simulador de un robot, permite al alumno iniciarse en la robótica sin necesidad de un robot.

Aunque es una práctica bastante utilizada, usar un robot no es la única alternativa para aprender a programar. Usar un entorno online para enseñar a programar es una alternativa que ha obtenido notable popularidad últimamente. Como se vio en la sección 2.2, Scratch es un gran exponente del *Computational Thinking* que utiliza un entorno principalmente online para que niños de todo el mundo aprenda a programar. Aunque éste no es el único proyecto que existe en esta línea. CodeHS y CodeSchool son algunos de los proyectos con más repercusión que buscan extender la cultura del *Computational Thinking*, especialmente entre niños. En las siguientes secciones se estudiarán estas alternativas.

2.4.1. CodeHS

Otro proyecto muy extendido que promueve el *Computational Thinking* es CodeHS. CodeHS es una plataforma online con la que personas de todas las edades pueden aprender informática. Tienen cursos de introducción a la informática, programar con diferentes lenguajes, bases de datos, e incluso una versión modernizada de Karel the robot, el cual se ha analizado en la sección 2.3.3.

CodeHS ofrece muchas opciones de aprendizaje independiente, incluso con tutores personalizados para resolver dudas de los usuarios, está especialmente dirigido a enseñar programación en colegios e institutos. Ofrece planes de pago para implantar la plataforma en las clases o herramientas de seguimiento y evaluación para que el profesor pueda supervisar el rendimiento de los alumnos. Aunque los planes gratuitos ofrecen un reducido número de cursos, permite al usuario programar libremente en una gran selección de lenguajes que tiene disponible, como pueden ser Javascript, Ruby o Python.

2.4.2. Code.org

Code.org[14] es una plataforma on-line cuyo objetivo es extender el desarrollo de las habilidades relacionadas con el pensamiento computacional entre niños de todas las edades. Actualmente cuenta con más de 8 millones de usuarios y está respaldada por una gran comunidad de educadores y gente influyente de todos los sectores, no solamente el de la informática.

Uno de los proyectos con más repercusión es el *Hour of Code*[13] (Hora del Código) que intenta que todos los niños dediquen una hora diaria a programar. Esta idea está apoyada por muchas empresas que ponen sus recursos y productos a disposición de Code.org para que puedan hacer uso de estos. Uno de ellos, por ejemplo, es Robomind, que como ya se comentó en la sección 2.3.4, tiene una modalidad gratuita con pruebas para aprender a utilizar el simulador del robot Robomind. Marcas como Disney, Flappy bird o Rovio⁹ ponen a disposición de Code.org y los niños el uso de sus personajes, intentando captar la atención de los niños y haciendo la actividad altamente atractiva. También, personas influyentes del panorama internacional se ofrecen a explicar conceptos de programación durante la Hora del Código a los niños.

Para programar, en Code.org se utiliza una versión simplificada de Scratch (el cual se ha estudiado en la sección 2.2) con la misma mecánica. Code.org propone muchos cursos independientes y decenas de horas de aprendizaje, con actividades nuevas para captar la atención del niño.

⁹Rovio es una empresa conocida por desarrollar los populares juegos de smartphone Angry Birds.

Capítulo 3

Objetivos, metodología y herramientas

3.1. Análisis de objetivos

Este Trabajo Fin de Grado consiste en desarrollar un módulo de simulación de un robot para integrarlo en la plataforma Descubre la programación(descubre.inf.um.es).

El simulador se ha denominado *Robode* y éste simulará un robot real con **funciones realistas** y en un entorno continuo (**y no discreto**). El simulador dispondrá de una serie de características básicas:

- Robode será un robot de dos ruedas.
- Cada rueda de Robode se moverá de manera independiente, simulando un motor para cada rueda.
- Robode se encontrará en un circuito finito.
- El circuito tendrá *lineas* pintadas en el suelo, *obstáculos* y *fronteras*.
- Las *lineas* pintadas el suelo no colisionarán con Robode.
- Los *obstáculos* podrán colisionar con Robode, produciendo un efecto en el contacto de ambos cuerpos, desplazándolos según la velocidad y características físicas de cada uno.
- Las *fronteras* serán elementos estáticos del circuito, impidiendo su movimiento o el de Robode a través de las mismas. El circuito tendrá fronteras limitando su superficie.

- Dispondrá de 2 sensores que, mirando al suelo, detectarán líneas pintadas y, por tanto, si Robode se encuentra sobre una línea.
- Dispondrá de sensores de proximidad delanteros y traseros, que detectarán si se va a producir una colisión con un obstáculo o una frontera.

El simulador se integrará con Descubre, una plataforma para aprender programación utilizando el lenguaje iJava¹. Por tanto, será necesario entender el funcionamiento interno de la plataforma y modificarla apropiadamente para poder integrar Robode con Descubre. De esta manera, se añadirán las funciones necesarias para poder crear programas en iJava y controlar el robot.

3.2. Metodología

En serio, ni idea de que poner/inventarme aquí.

3.2.1. Tecnologías

Robode utiliza una serie de tecnologías web sobre las que trabaja para poder integrarse en el entorno de Descubre y crear un simulador en un entorno online. A continuación se describirán brevemente estas tecnologías.

- **HTML** o *HyperText Markup Language* (lenguaje de marcas de hipertexto) es un estándar que proporciona la estructura de las páginas web. En concreto, se ha utilizado la última versión de HTML, HTML5, que proporciona elementos como *canvas*, el cual usaremos para dibujar nuestro robot.
- **CSS** son las siglas para *Cascading Style Sheets*. Es un lenguaje para definir el estilo (o presentación) de la página web sobre HTML.
- **Javascript**, como se le conoce comúnmente, es un lenguaje para programar páginas web que trabaja junto a HTML. Es un lenguaje interpretado por los navegadores, con orientación a objetos basada en prototipos, imperativo, con tipado débil y dinámico. Javascript está estandarizado por ECMA International² y su nombre oficial es ECMAScript. La versión más extendida de Javascript actualmente es ECMAScript 5. Aunque una nueva versión se hizo pública en junio de 2015[25].

¹Para más información sobre la plataforma Descubre y el lenguaje iJava se puede consultar la sección 1.3.

²Página oficial de ECMA International (<http://www.ecma-international.org>).

3.3. Herramientas utilizadas

Las herramientas que se han usado para desarrollar este proyecto se describen a continuación:

- **Atom**³ como editor de código. Acompañado de paquetes *linter*⁴ para JavaScript, HTML y CSS que detectan la mayoría de los errores sintácticos que se producen a la hora de programar. También se han utilizado diferentes paquetes para tabular y organizar el código automáticamente.
- Como herramienta de control de versiones se ha utilizado **Git**⁵ y como repositorio **GitHub**⁶.
- Para las pruebas, ejecución y depuración de la aplicación se ha utilizado tanto **Google Chrome**⁷ versión 48, como **Safari**⁸ versión 9.0.3.
- Para el desarrollo de esta documentación se ha utilizado **MacTex** como motor de *Latex*⁹ y **TexMaker**¹⁰ para la redacción del documento. También se ha utilizado **BibTex**¹¹ como gestor bibliográfico.

³Página oficial de Atom (<https://atom.io>).

⁴Un *linter* analiza el código escrito en busca de errores tipográficos. Para más información sobre el *linter* de Atom, se puede consultar la página web de la herramienta: <https://atom.io/packages/linter>.

⁵Página oficial del proyecto Git (<https://git-scm.com>).

⁶Página oficial de GitHub (<https://github.com>)

⁷Página oficial de Google Chrome (<https://www.google.es/chrome/browser/desktop/>).

⁸Página oficial de Safari (<http://www.apple.com/es/safari/>).

⁹Página oficial del proyecto Latex (<https://www.latex-project.org>).

¹⁰Página oficial de TexMaker (<http://www.xmlmath.net/texmaker/>).

¹¹Página oficial de BibTex (<http://www.bibtex.org>).

Capítulo 4

Diseño y resolución del trabajo realizado

Después de estudiar cual es la situación actual de la enseñanza de la programación a un nivel global y los diferentes proyectos que promueven el *pensamiento computacional*, en este capítulo se analizarán cuales son las necesidades del proyecto Robode y como se han resuelto.

Como ya se ha mencionado anteriormente, se creará un simulador de un robot, al que hemos denominado *Robode*, y se integrará en la plataforma Descubre la programación. Robode estará dentro de un mundo del que podrá moverse con libertad y con el que podrá interactuar. El robot tendrá dos ruedas que se mueven independientemente (con un motor para cada una de ellas) y dispondrá de sensores que informarán de posibles colisiones con el robot. También, el robot tendrá la capacidad de detectar líneas o caminos pintados en el suelo, sabiendo también en qué dirección detecta (o no) la línea.

Adicionalmente, el robot y el mundo en el que este se encuentra será dibujado en un elemento *canvas* de HTML5, al igual que ocurre en la plataforma Descubre.

Como se puede ver, Robode es la unión de varias de los proyectos que se han analizado en el capítulo 2. Por una parte, la idea principal de crear un simulador es igual a la que se ha realizado en Robomind (el cual se analiza en la sección 2.3.4) pero simulando un robot en un mundo continuo y con características similares a las de Moway (analizado en la sección 2.3.2).

En las secciones siguientes se analizarán las distintas alternativas que se han manejado para poder cumplir esta meta y

4.1. Creación del mundo

Para la creación del mundo y de las físicas que gestionen las colisiones y la interacción entre todos los elementos del mundo se podría tanto haber creado una librería de físicas desde 0 como utilizar una ya existente, y en el caso de que fuera necesario, adaptarla a las necesidades de Robode.

Por supuesto, crear una librería de físicas desde 0 y dotarlo de una parte gráfica que dibujara los elementos sobre el *canvas* de HTML5 podría conformar, eso únicamente, un Trabajo Fin de Grado completo. La idea de utilizar alguna de las librerías que ya existen y que son de software libre para adaptarla a las necesidades actuales cobra mucha fuerza.

Entre las opciones de librería que existen actualmente para Javascript, y habiendo descartado las opciones de pago, restaban dos alternativas que podían cumplir los requisitos que se planteaban. Estas son PhysicsJS¹ y Box2d[9]. Ambas permiten multitud de implementaciones para realizar casi cualquier aplicación o juego, pero lo que decanta la balanza es la falta de elementos constrictores² en la librería PhysicsJS. PhysicsJS está aún en fase beta y, aunque promete ser una gran librería de físicas y es muy completa, aún necesita ser mejorada. La necesidad de estos elementos constrictores se verá en la sección 4.1.1, cuando se construya el robot.

Por tanto, se ha elegido la librería Box2dweb³, la cual es un port de la librería Box2D desarrollada para Flash⁴ y tiene una gran y activa comunidad usando ambas versiones de Box2d, lo que facilita el trabajo de resolución de dudas y problemas. También existe otra alternativa en Javascript de la librería Box2D, llamada Box2DJS, pero ésta está basada en una versión anterior de Box2D y no se ha seguido actualizando.

Box2D tiene una serie de elementos básicos que afectan al diseño y uso de nuestro simulador. Estos son: el Mundo (World), los Cuerpos (Bodies), los Accesorios (Fixtures) y las Articulaciones o elementos constrictores (Joints).

El núcleo del motor de físicas en Box2D es *World*. *World* define los parámetros básicos que regirán la simulación más tarde, como puede ser el ratio de actualización o la gravedad. También contiene y controla el resto de elementos: *Bodies*, *Fixtures* y *Joints*. La creación de estos elementos siguen un *patrón factoría*, teniendo que definirlo primero en un objeto para poder crear el elemento real.

En el código 4.1 se puede ver como se crea el cuerpo principal del robot utilizando

¹Página oficial de la librería PhysicsJS (<http://wellcaffeinated.net/PhysicsJS>).

²En este caso, un elemento constrictor es un objeto que limita de alguna manera los movimientos de otro. Un ejemplo en la vida real puede ser una articulación en el cuerpo humano: el movimiento del brazo está limitado por el ángulo de movimiento que esa articulación le permite hacer.

³El repositorio de Box2Dweb está alojado en GitHub (<https://github.com/hecht-software/box2dweb>).

⁴Página oficial de la librería Box2D para Flash (<http://www.box2dflash.org>).

una factoría tanto para definir el objeto *Body* como la *Fixture* a partir de *World*, que realizará la creación del mismo.

```

1 var bodyDef = new b2BodyDef();
bodyDef.type = b2Body.b2_dynamicBody;
3 bodyDef.position.Set(posX, posY);
bodyDef.linearDamping = 8;
5 bodyDef.angularDamping = 8;

7 var fixDef = new b2FixtureDef();
fixDef.density = 40;
9 fixDef.friction = 1;
fixDef.restitution = 0;
11 fixDef.shape = new b2PolygonShape();
fixDef.shape.SetAsBox(width, height);

13 //robot BODY
15 var robot = Simulator.World.CreateBody(bodyDef);
robot.setName("robot");

17 robot.CreateFixture(fixDef);

```

Código 4.1: Creación del cuerpo principal del robot utilizando la librería Box2dweb.

El elemento *Body* será el que contenga la información básica del objeto, como es su posición, velocidad o ángulo de giro. Será el elemento *Fixture* el que defina la forma (si es un polígono o círculo y de qué forma) y características (fricción que produce, densidad, etc) del *Body* al que está asociado. El mismo método se sigue para crear el resto de elementos como un *Joint* en Box2D.

4.1.1. Construcción del robot

4.1.2. Construcción de circuitos

El entorno o circuito por el que se moverá el robot tiene los siguientes elementos: (a) obstáculos, con los que podrá chocar y que se desplazarán por la colisión; (b) fronteras, que no podrán ser desplazadas o atravesadas por el robot y (c) líneas, que no producirán una colisión pero que podrán ser detectadas por el robot.

4.1.3. Colisiones

4.2. Curvas de Bezier

Uno de los aspectos más importantes en Robode es la capacidad del mismo de poder detectar lineas. La finalidad de esto es que se pueda programar un comportamiento *sigue-lineas*. Para ello, era necesario responder a una serie de preguntas:

1. ¿Cómo se representan las lineas pintadas en el suelo?
2. ¿Cómo se modelan dichas lineas?

4.3. Integración en Descubre

4.3.1. Modificación del lenguaje iJava

Capítulo 5

Conclusiones y vías futuras

Apéndice A

Edad escolar en diferentes Sistemas Educativos

Incluso en la Unión Europea, los Sistemas Educativos difieren en la edad de escolarización de los estudiantes. Por ello, se ha confeccionado una tabla que intenta resumir la edad estándar en la que un niño está escolarizado durante las distintas etapas educativas¹. La información que se muestra a continuación está basado en [50] y [16].

Se tomarán como referencia los nombres de las etapas del Sistema Educativo Español (Educación Infantil, Primaria, Secundaria y Bachillerato) para mostrar los años que pasan los estudiantes. Con respecto a la selección de países para realizar la comparación, se escogen los más representativos en los que se han realizado los estudios sobre el aprendizaje de la programación y donde más extendido están las plataformas que se mencionan en el presente documento.

País	Infantil	Primaria	Secundaria	Bachillerato
España	0-6	6-12	12-16	16-18
Estados Unidos	3-6	6-10	10-14	14-18
Reino Unido	2-5	5-11	11-16	16-18
Alemania	0-6	6-10	10-16	16-19
Francia	2-6	6-11	11-16	16-18
Bélgica	0-2.5/3	2.5/3-6	6-12	12-18
Irlanda	4-6	6-12	12-15	15-19

Tabla A.1: Comparativa de edades de escolarización en diferentes Sistemas Educativos con respecto a las etapas del Sistema Educativo Español.

¹Se hablará de las etapas en las que el Sistema Educativo referido está bajo la responsabilidad del Ministerio de Educación (o equivalente) del propio país. Esto no excluye a la educación en centros privados.

34 APÉNDICE A. EDAD ESCOLAR EN DIFERENTES SISTEMAS EDUCATIVOS

En el caso del Sistema Educativo Americano, existen muchas vías en la formación de un niño, como bien detalla A. Corsi-Bunker en [16]. Dependiendo de si se escoge una vía privada o dependiendo del estado, los años pueden variar. Aún así, en la figura A.1 se muestra el modelo estándar (sistema K-12).².

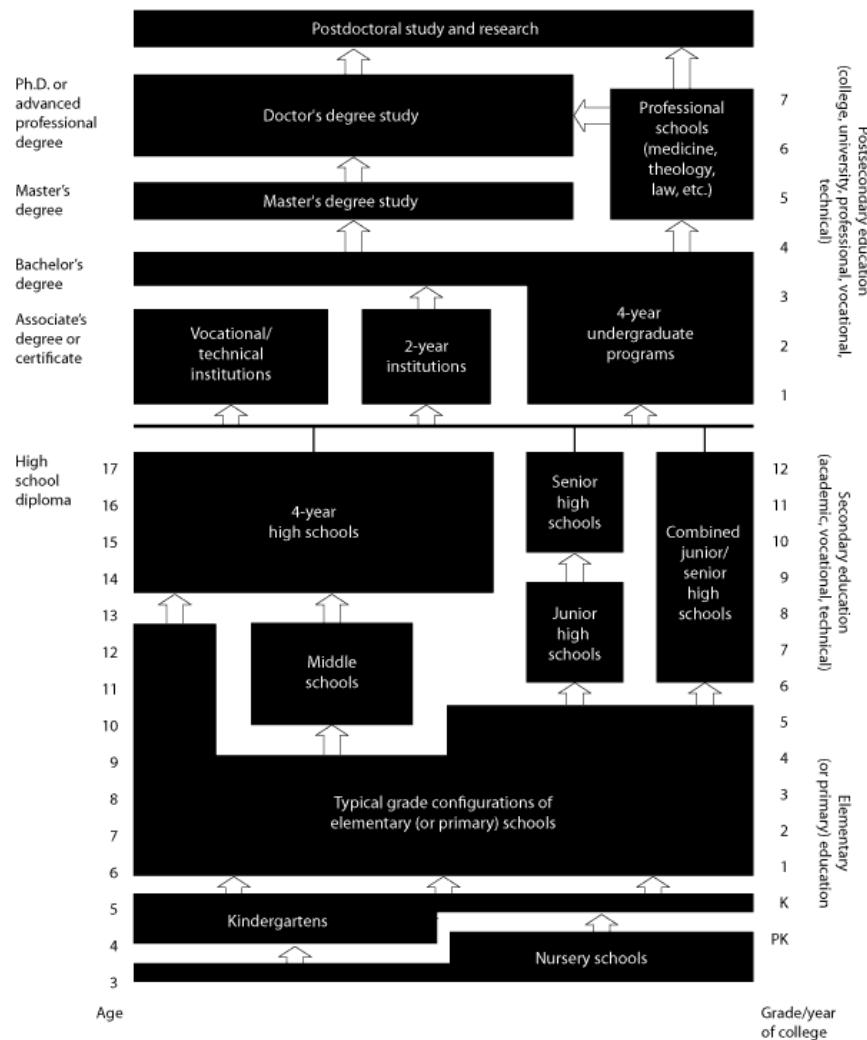


Figura A.1: Tabla que muestra los típicos patrones de progresión en el Sistema Educativo Americano. Fuente: U.S. Department of Education, National Center for Education Statistics, Annual Reports Program. Obtenido de http://nces.ed.gov/programs/digest/d11/figures/fig_01.asp

²Para más información sobre el Sistema Educativo Americano recomiendo que se acuda directamente a la fuente: U.S. Department of Education, National Center for Education Statistics (<http://nces.ed.gov>)

Apéndice B

Lenguaje Logo

A continuación se realizará una revisión de las principales características del lenguaje Logo para tener una visión general de su funcionamiento y sintaxis. Este análisis se basa en el trabajo de [21, p.274-305]. Si el lector quiere ampliar información sobre el lenguaje Logo, puede consultar [23] o [31].

B.1. *words, sentences, operaciones y comandos*

Hay dos tipos de datos principales: *words* y *sentences*. *words* está formado por 0 o más caracteres, sin espacios. Algunos ejemplos son: "SUN", "CAT", "5", "97!", "" (cadena vacía). El tipo de datos *sentences* se forma por un conjunto de *words* separado por espacios, como por ejemplo "HELLO WORLD" o "3 + 2 = 5". Un numero es una *word* pero compuesto únicamente por dígitos.

En el lenguaje Logo existen operadores para tratar con los tipos *words* y *sentences*. También se pueden encontrar una serie de operaciones predefinidas en el lenguaje que ofrecen funcionalidad extra, como controlar la entrada/salida de un programa, saber la fecha o manejo de *words* y *sentences*. En la tabla B.1 vemos un resumen de las operaciones elementales del lenguaje Logo, el número de argumentos que requiere y un ejemplo de entrada con su salida correspondiente.

Si se intentara ejecutar la operación WORD "CAT" "DOG", se produciría un error al ser uno de sus parámetros un tipo *sentence* y no *word* (el primer argumento contiene un espacio). De igual manera, las operaciones SUM, DIFFERENCE, MAXIMUN y MINIMUN deben recibir argumentos numéricos o se produciría un error.

También es posible encadenar varios operadores. De esta manera, la cadena de instrucciones FIRST OF BUTFIRST OF "CAT" devolverá el *word* "A".

<i>Operador</i>	N. argumentos	Entrada	Salida
FIRST	1	”CAT”	”C”
-	-	”CAT AND DOG”	”CAT”
LAST	1	”CAT”	”T”
-	-	”CAT AND DOG”	”DOG”
BUTFIRST	1	”CAT”	”AT”
-	-	”CAT AND DOG”	”AND DOG”
BUTLAST	1	”CAT”	”CA”
-	-	”CAT AND DOG”	”CAT AND”
COUNT	1	”CAT”	”3”
-	-	”CAT DOG”	”2”
WORD	2	”CAT” ”DOG”	”CATDOG”
SENTENCE	2	”CAT” ”DOC”	”CAT DOC”
-	-	”CAT” ”DOG”	”CAT DOC”
PRINT	1	”CAT”	CAT
TIME	0	-	”5:42 PM”
DATE	0	-	”1/31/2016”
SUM	2	”-5” ”5”	”0”
DIFERENCE	2	”5” ”5”	”0”
MAXIMUN	2	”-5” ”5”	”5”
MINIMUN	2	”3” ”5”	”3”

Tabla B.1: Resumen de las operaciones elementales que ofrece el lenguaje Logo.

B.2. Variables, comentarios y definiendo nuevas operaciones

Para definir variables basta con usar el operador **MAKE** seguido de ”(comilla doble), el nombre de la variable y el valor que se quiere asignar. Para hacer uso de la variable, utilizamos el símbolo :(dos puntos) seguido del nombre de la variable. Para hacer comentarios se utiliza el símbolo ;(punto y coma). Podemos ver un ejemplo en el código B.1.

```

1 MAKE "ADULTO 18
2
PRINT :ADULTO ; salida -> 18

```

Código B.1: Ejemplo de definición de nuevas variables en el lenguaje Logo.

También es posible crear nuevas operaciones. Se utilizan las palabras reservadas TO y END para marcar el inicio y final de la definición, respectivamente. A continuación es necesario declarar el nombre con el que se definirá la operación que se está

creando (el cual no puede coincidir con ninguna palabra reservada en el lenguaje, como es común en la mayoría de lenguajes de programación). Es opcional la declaración de parámetros que se pasaran a la operación. Por último, y antes de la palabra END, se incluirán las instrucciones que realizará la operación.

En el código B.2 se puede ver un ejemplo de definición de nuevas operación.

```

1  TO SALUDAR
2    MAKE "SALUDO "HELLO WORLD"
3    OUTPUT SALUDO
4  END
5
6  PRINT SALUDAR ; salida -> HELLO WORLD
7
8  TO PENULTIMO :algo
9    OUTPUT FIRST OF LAST :algo
10 END
11
12 PRINT PENULTIMO OF "CAT BIRD DOG" ; salida -> BIRD
13
14 PRINT PENULTIMO OF "CAT" ; salida -> A

```

Código B.2: Definición de nuevas operaciones en el lenguaje Logo.

B.3. Operadores aritméticos

A parte de los operadores SUM y DIFFERENCE antes mencionados, Logo también otorga la posibilidad de utilizar los operadores aritméticos clásicos. La única condición es que la salida generada debe ser tratada (PRINT) o almacenada (en variables, como veremos en la sección siguiente) de alguna manera. Los operadores más importantes son +, -, *, / y sqrt.

```

1 PRINT 2*3 ; salida -> 6
2
3 MAKE "RUEDAS 4
4
5 PRINT :RUEDAS ; salida -> 4

```

Código B.3: Ejemplo de uso de operadores aritméticos en el lenguaje Logo.

B.4. Operadores IF y REPEAT

Se pueden crear operaciones condicionales utilizando la palabra clave **IF** seguida de una sentencia y una lista de órdenes a ejecutar si se evalúa dicha sentencia a verdadero. La lista de órdenes se declara entre corchetes `()` y para la sentencia se pueden utilizar los operadores infijos `=`, `>` y `<`. En el código B.4 podemos ver un ejemplo de uso.

```
1 MAKE "PETALOS 4
3 IF :PETALOS > 3 [ PRINT "TREBOL DE LA BUENA SUERTE" ]
```

Código B.4: Condiciones en el lenguaje logo con el operador **IF**.

Para repetir una serie de instrucciones, Logo ofrece el comando **REPEAT**, que va seguido de un valor numérico y la lista de instrucciones. El valor numérico (que puede ser parametrizado) es el número de veces que se repetirá en bucle las instrucciones. En el código B.5 vemos un ejemplo de este comando.

```
1 REPEAT 10 [ PRINT "HOLA MUNDO" ]
```

Código B.5: Condiciones en el lenguaje logo con el operador **IF**.

Bibliografía

- [1] A logo primer. http://el.media.mit.edu/logo-foundation/what_is_logo/logo_primer.html, Último acceso en enero 2016.
- [2] Proyecto Código 21. Página oficial del proyecto código 21 promovido por el departamento de educación de gobierno de navarra. <http://codigo21.educacion.navarra.es>, Consultado el 9 de enero de 2016.
- [3] Harold Abelson. A. disessa. turtle geometry. *Cambridge (Mass.)*, 1980.
- [4] Code Academy. Página oficial de code academy. <https://www.codecademy.com>, Consultado el 12 de diciembre de 2015.
- [5] Turtle Academy. Página oficial del proyecto turtle academy. <https://turtleacademy.com>, Consultado el 25 de enero de 2016.
- [6] Noreen S. Ahmed-Ullah. Cps to add computer science as core subject. *Chicago Tribune*, Diciembre 2013.
- [7] Arduino. Página oficial del proyecto arduino. <https://www.arduino.cc/>, Consultado el 9 de enero de 2016.
- [8] Joseph Bergin, J Eckstein, M Manns, H Sharp, M Voelter, E Wallinfgord, K Marquardt, J Chandler, and A Fricke. Pedagogical patterns. early bird. <http://csis.pace.edu/~bergin/PedPat1.2.html#earlybird>, 2005. Consultado el 3 de febrero de 2016.
- [9] Box2d. Página oficial del proyecto box2d. <http://box2d.org>, Consultado el 1 de abril de 2015.
- [10] Judy Ann Brown, Dorothy M Fitch, and Scott W Earle. 101 ideas for logo, gr. 3-10, 1995.
- [11] Alan Bundy. Computational thinking is pervasive. *Journal of Scientific and Practical Computing*, 1(2):67–69, 2007.

- [12] Douglas H Clements. Effects of logo and cai environments on cognition and creativity. *Journal of Educational Psychology*, 78(4):309, 1986.
- [13] Code.org. Página oficial del proyecto hora del código de code.org. <https://hourofcode.com/>, Consultado el 30 enero de 2016.
- [14] Code.org. Página oficial de code.org. <https://code.org>, Consultado el 9 de enero de 2016.
- [15] Social Committee Commission to the European Parliament, the European Economic and the Committee of the Regions. Draft 2015 joint report of the council and the commission on the implementation of the strategic framework for european cooperation in education and training (et 2020). new priorities for european cooperation in education and training. http://ec.europa.eu/education/documents/et-2020-swd-161-2015_en.pdf, Agosto 2015.
- [16] Antonella Corsi-Bunker. *Guide to the Educational System in the United State*. University of Minnesota. Internal Student and Scholar Service.
- [17] Sean Coughlan. Computer science part of english baccalaureate. *BBC News*, Febrero 2013.
- [18] Julie Sarama Douglas H. Clements, Michael T. Battista. Logo and geometry. *Journal for Research in Mathematics Education. Monograph*, 10:i–177, 2001.
- [19] Moway Education. Página oficial de moway education. <http://moway-robot.com>, Consultado el 10 octubre de 2015.
- [20] L Fernández Muñoz, R Peña, F Nava, and A Velázquez Iturbide. Análisis de las propuestas de la enseñanza de la programación orientada a objetos en los primeros cursos. *Actas de las VIII Jornadas de Enseñanza Universitaria de la Informática (JENUI'02)*, pages 433–440, 2002.
- [21] Wallace Feurzeig et al. Programming-languages as a conceptual framework for teaching mathematics. final report on the first fifteen months of the logo project. 1969.
- [22] Department for Education UK Government. Statutory guidance. national curriculum in england: computing programmes of study. <https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study>, Septiembre 2013.

- [23] Michael Friendly. *Advanced LOGO: A language for learning*. Psychology Press, 2014.
- [24] Raspberry Pi Fundation. Página oficial de raspberry pi. <https://www.raspberrypi.org>, Consultado el 15 enero de 2016.
- [25] ECMAScript International. Ecma-262, escript 2015 language specification. <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf>, Junio 2015.
- [26] Descubre la programación. Página oficial del proyecto descubre la programación. <http://descubre.inf.um.es>, Consultado el 9 de enero de 2016.
- [27] Descubre la programación. Página oficial del proyecto descubre la programación. guía del lenguaje. <http://descubre.inf.um.es/curso.php>, Consultado el 9 de enero de 2016.
- [28] Maude Lemaire. Incorporating computer science into an elementary school curriculum. 2014.
- [29] Yuen-Kuang Cliff Liao and George W Bright. Effects of computer programming on cognitive outcomes: A meta-analysis. *Journal of Educational Computing Research*, 7(3):251–268, 1991.
- [30] Curly Logo. Página oficial del proyecto curly logo. una implementación web de la famosa turtle de logo. <http://drj11.github.io/curlylogo/>, Consultado el 31 de enero de 2016.
- [31] Fundación Logo. Página oficial de la fundación logo. recursos para aprender y enseñar logo. <http://el.media.mit.edu/logo-foundation/resources/books.html#language>, Consultado el 31 de enero de 2016.
- [32] Fundación Logo. Página oficial de la fundación logo. <http://el.media.mit.edu/logo-foundation>, Consultado el 9 de enero de 2016.
- [33] José Marcos. Los colegios de madrid impartirán clases de programación. *El País*, Septiembre 2014.
- [34] Andrew Mcgettrick, Roger Boyle, Roland Ibbett, John Lloyd, Gillian Lovegrove, and Keith Mander. Grand challenges in computing: Education. a summary. *The Computer Journal*, 48(1):42–48, 2005.
- [35] Lego Mindstorm. Página oficial de lego minsturm. <http://www.lego.com/es-es/mindstorms/>, Consultado el 20 de enero de 2016.

- [36] Lego Mindstorm. Página oficial de lego minsturm. sección aprende a programar. <http://www.lego.com/es-es/mindstorms/learn-to-program>, Consultado el 20 de enero de 2016.
- [37] Richard Moss. *Creating a mathematical environment through programming: A study of young children learning Logo*. PhD thesis, University of London, 1985.
- [38] Richard Pattis, J Roberts, and M Stehlik. Karel the robot. *A gentle introduction to the Art of Programming*, 1981.
- [39] Roy D Pea and D Midian Kurland. Logo programming and the development of planning skills. technical report no. 16. 1984.
- [40] V Renumol, S Jayaprakash, and D Janakiram. Classification of cognitive difficulties of students to learn computer programming. *Indian Institute of Technology, India*, 2009.
- [41] RoboMind. Página oficial de robomind. <https://www.robomindacademy.com>, Consultado el 10 de octubre de 2015.
- [42] Juan Antonio Sánchez Laguna. ijava: un nuevo lenguaje para facilitar el paso del paradigma imperativo al orientado a objetos. *Jornadas de Enseñanza Universitaria de la Informática (15es: 2009: Barcelona)*, 2009.
- [43] Code School. Página oficial de code school. <https://www.codeschool.com>, Consultado el 9 de enero de 2016.
- [44] Scratch. Página oficial del proyecto scratch. <https://scratch.mit.edu>, Consultado el 30 enero de 2016.
- [45] Janet Siegmund, Christian Kästner, Sven Apel, Chris Parnin, Anja Bethmann, Thomas Leich, Gunter Saake, and André Brechmann. Understanding understanding source code with functional magnetic resonance imaging. In *Proceedings of the 36th International Conference on Software Engineering*, pages 378–389. ACM, 2014.
- [46] Brown University. Logo tutorial. un tutorial de turtle y el lenguaje logo dentro de la brown university. <http://cs.brown.edu/courses/bridge/1997/Resources/LogoTutorial.html>, Consultado el 30 de enero de 2016.
- [47] EU Code Week. Página oficial del proyecto eu code week. <http://codeweek.eu>, Consultado el 9 de enero de 2016.
- [48] Jeannette M Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.

- [49] Jeannette M Wing. Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 366(1881):3717–3725, 2008.
- [50] Eurydice Datos y Cifras. Educación y formación. La estructura de los sistemas educativos europeos 2014/15: diagramas. http://eacea.ec.europa.eu/education/eurydice/documents/facts_and_figures/education_structures_es.pdf, Noviembre 2014.

Glosario de términos

Arduino . 9

Comisión Europea Órgano ejecutivo y legislativo de la Unión Europea. Se encarga de proponer la legislación, la aplicación de las decisiones, la defensa de los tratados de la Unión y del día a día de la Union Europe. 2

Descubre la programación . 4, 5, 23

Hardware . 1

iJava . 4

Java . 4

Lenguaje Logo El lenguaje Logo, basado en el lenguaje Lisp, fue diseñado como una herramienta para aprendizaje. Todas sus características -interactividad, modularidad, extensibilidad, flexibilidad en los tipos de datos- persiguen esta meta. 2, 9

Lisp . 9

Open Source . 3

Software . 1

Turtle El proyecto más popular del Lenguaje Logo ha evolucionado en la Tortuga, originalmente una criatura robótica que se movía por el suelo siguiendo una serie de instrucciones programadas previamente. 9

Acrónimos

MIT Massachusetts Institute of Technology. 9, 12

TIC Tecnologías de la Información y la Comunicaciones. 1