



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE  
TELECOMUNICACIÓN

GRADO EN INGENIERÍA EN SISTEMAS AUDIOVISUALES  
Y MULTIMEDIA

TRABAJO FIN DE GRADO

***Gamificación de una plataforma web de robótica  
educativa***

Autora : Marta Quintana Portales

Tutor : Dr. José María Cañas Plaza

Curso Académico 2020/2021

## Agradecimientos

*A mi familia, amigas, tutor, compañeros y a mí.*

Quiero empezar este trabajo dando las gracias a todas las personas que me han acompañado a lo largo de esta aventura. Especialmente a mi familia, no puedo estar más orgullosa, si lo he conseguido ha sido gracias a ellos, sobre todo a mi madre, a mi padre y a mi hermana que siempre me apoyan y sacan lo mejor de mí.

Gracias a Jose María por las oportunidades y la confianza en mí para la realización de este TFG. También a todo el equipo de *Kibotics* en especial a Pablo, David, Roberto y Sergio y a la asociación *JdeRobot*, gracias por hacerlo un poco más fácil a pesar de la situación que estamos viviendo por la pandemia.

También quiero agradecer a mis amigas por estar ahí siempre, a ese grupo que tenemos ‘No me da la vida’ que refleja perfectamente lo que hemos vivido estos años en la universidad, gracias por apoyarme y colaborar en la realización de este trabajo y no me puedo olvidar de mis compañeros y compañeras de la carrera que han hecho que estos años sean más llevaderos y llenos de momentos para recordar.

Muchas gracias a todos por formar parte de mi vida.

*No ha sido fácil pero nada que valga la pena lo será.*

## Resumen

Gracias al avance de la tecnología, la robótica y la programación se han convertido en pilares fundamentales para el futuro de los más jóvenes, es por ello que las plataformas web cobran mucha importancia en el aprendizaje. Kibotics es una plataforma web de robótica educativa enfocada a niños y adolescentes en la que pueden aprender los fundamentos de programación de robots en lenguajes como Python y Scratch. Jugar permite desarrollar aspectos psíquicos, físicos y sociales, por estos motivos, el aprendizaje a través de juegos en el entorno educativo y profesional, más conocido como *gamificación*, es fundamental para la formación académica de las nuevas generaciones.

En este proyecto nos hemos centrado en la realización de tres nuevos ejercicios educativos en formato juego, divertidos y realistas para Kibotics. El primer ejercicio es un aspirador robótico en el que se ha hecho un nuevo robot que aspira piezas de confeti que están esparcidas por una habitación. El segundo ejercicio es el juego del pañuelo, se ha creado un robot Mbot con pinzas que es capaz de coger un lata y recorrer un circuito con ella. Estos dos ejercicios poseen evaluadores automáticos que los hacen más interesantes y competitivos. El tercer ejercicio es el teleoperador acústico. En él, el usuario tiene que guiar con la voz a un dron para que se mueva por el escenario sin chocarse. Esto se ha realizado con procesamiento de audio gracias a la herramienta *Teachable Machine*. Para una mejor experiencia de usuario, se ha añadido la posibilidad de poner bandas sonoras a los ejercicios ya existentes y efectos de sonido cuando el robot choca con algún objeto de la escena.

La implementación de los tres ejercicios ha sido principalmente con tecnologías web entre las que cabe destacar JavaScript y el simulador Websim basado en A-Frame.

El ejercicio del aspirador robótico y el juego del pañuelo ya están disponibles en la plataforma para que los usuarios puedan aprender y jugar con ellos. El teleoperador acústico está integrado como prototipo.

# Índice general

|   |           |
|---|-----------|
| <b>Índice de figuras</b>  | <b>5</b>  |
| <b>1. Introducción</b>  | <b>1</b>  |
| 1.1. Robótica . . . . .   | 1         |
| 1.2. Tecnologías Web . . . . .  | 5         |
| 1.2.1. Tecnologías Web lado cliente . . . . .                             | 7         |
| 1.2.2. Tecnologías Web lado servidor . . . . .                            | 8         |
| 1.3. Robótica educativa . . . . .   | 9         |
| 1.3.1. Importancia de los juegos y multimedia en el aprendizaje . . . . . | 12        |
| 1.3.2. Campeonatos de robótica educativa existentes . . . . .             | 13        |
| 1.4. Estructura del documento . . . . .                                   | 15        |
| <b>2. Objetivos</b>   | <b>16</b> |
| 2.1. Objetivos . . . . .  | 16        |
| 2.2. Requisitos . . . . .   | 17        |
| 2.3. Metodología . . . . .  | 17        |
| 2.4. Plan de Trabajo . . . . .  | 19        |
| <b>3. Infraestructura utilizada</b>                                       | <b>20</b> |
| 3.1. Lenguajes de programación y de documentos . . . . .                  | 20        |
| 3.1.1. HTML5 . . . . .  | 20        |
| 3.1.2. JavaScript . . . . .   | 21        |
| 3.1.3. Python y Scratch . . . . .   | 22        |
| 3.1.4. JSON . . . . .   | 24        |
| 3.2. Herramientas . . . . .   | 25        |

|           |   |           |
|-----------|---|-----------|
| 3.2.1.    | TensorFlowJS, Web Audio API y Teachable Machine . . . . .     | 25        |
| 3.2.2.    | A-Frame . . . . .   | 26        |
| 3.2.3.    | Blender . . . . .   | 28        |
| 3.2.4.    | Plataforma Kibotics . . . . .                                 | 29        |
| 3.2.5.    | Navegadores web utilizados . . . . .                          | 31        |
| <b>4.</b> | <b>Ejercicio aspiradora robótica atrapa confeti</b>           | <b>32</b> |
| 4.1.      | Enunciado . . . . .   | 32        |
| 4.2.      | Modelos del mundo . . . . .                                   | 35        |
| 4.3.      | Modelo aspiradora . . . . .                                   | 39        |
| 4.4.      | Evaluador automático . . . . .                                | 42        |
| 4.5.      | Solución de referencia . . . . .                              | 43        |
| <b>5.</b> | <b>Ejercicio juego del pañuelo</b>                            | <b>45</b> |
| 5.1.      | Enunciado . . . . .   | 45        |
| 5.2.      | Modelos del mundo . . . . .                                   | 51        |
| 5.3.      | Modelo robot con pinza . . . . .                              | 53        |
| 5.3.1.    | Antecedentes . . . . .  | 54        |
| 5.3.2.    | Modelo Mbot con pinzas . . . . .                              | 55        |
| 5.4.      | Evaluador automático . . . . .                                | 60        |
| 5.5.      | Solución de referencia . . . . .                              | 62        |
| <b>6.</b> | <b>Ejercicio teleoperador acústico y banda sonora</b>         | <b>64</b> |
| 6.1.      | Enunciado . . . . .   | 64        |
| 6.2.      | Modelos del mundo . . . . .                                   | 65        |
| 6.3.      | Procesamiento de audio con <i>Teachable Machine</i> . . . . . | 67        |
| 6.4.      | Solución de referencia . . . . .                              | 78        |
| 6.5.      | Alternativas probadas . . . . .                               | 78        |
| 6.6.      | Banda Sonora . . . . .  | 80        |
| 6.6.1.    | Efecto de sonido por colisión . . . . .                       | 82        |
| <b>7.</b> | <b>Conclusiones y trabajos futuros</b>                        | <b>83</b> |
| 7.1.      | Conclusiones . . . . .  | 83        |

|                                 |           |
|---------------------------------|-----------|
| 7.2. Trabajos futuros . . . . . | 85        |
| <b>Bibliografía</b>             | <b>86</b> |

# Índice de figuras

|       |  |    |
|-------|--|----|
| 1.1.  | Sensor ultrasonidos . . . . .  | 2  |
| 1.2.  | Procesador . . . . .   | 2  |
| 1.3.  | Servomotores . . . . .   | 2  |
| 1.4.  | Pantalla . . . . .   | 2  |
| 1.5.  | Ejemplos de Robots . . . . .   | 3  |
| 1.6.  | Ejemplos de robots educativos . . . . .                                  | 4  |
| 1.7.  | Ejemplos aplicaciones web . . . . .                                      | 5  |
| 1.8.  | Comunicación cliente/servidor a través de HTTP . . . . .                 | 6  |
| 1.9.  | Ejemplo petición HTTP Cliente a Servidor . . . . .                       | 6  |
| 1.10. | Ejemplo respuesta HTTP Servidor a Cliente . . . . .                      | 6  |
| 1.11. | Bases de Datos . . . . .   | 8  |
| 1.12. | Scratch . . . . .  | 9  |
| 1.13. | Open Roberta . . . . .   | 10 |
| 1.14. | LEGO EDUCATION . . . . .   | 11 |
| 1.15. | MBlock IDE . . . . .   | 11 |
| 1.16. | Plataforma Kibotics. . . . .   | 12 |
| 1.17. | Campeonatos de robótica. . . . .   | 14 |
| 2.1.  | Modelo iterativo . . . . .   | 17 |
| 2.2.  | Página web de este TFG . . . . .   | 18 |
| 3.1.  | Ejemplo de página web con HTML5 . . . . .                                | 21 |
| 3.2.  | Ejemplo de página web interactiva con HTML5, CSS3 y JavaScript . . . . . | 22 |
| 3.3.  | Ejemplo de código en Kibotics . . . . .                                  | 23 |
| 3.4.  | Parte de un fichero config.json de un ejercicio en Kibotics . . . . .    | 25 |

|       |  |    |
|-------|--|----|
| 3.5.  | Herramientas de reconocimiento de audio . . . . .                    | 26 |
| 3.6.  | Ejemplo escena en A-Frame . . . . .                                  | 27 |
| 3.7.  | Inspector en A-Frame . . . . .                                       | 28 |
| 3.8.  | Blender . . . . .  | 29 |
| 3.9.  | Ejercicios Kibotics . . . . .  | 30 |
| 3.10. | Arquitectura de la aplicación web Kibotics . . . . .                 | 31 |
| 4.1.  | Página de teoría objetivos . . . . .                                 | 33 |
| 4.2.  | Página de teoría requisitos . . . . .                                | 34 |
| 4.3.  | Teoría . . . . .   | 34 |
| 4.4.  | Pistas . . . . .   | 34 |
| 4.5.  | ¿Sabías que ...? . . . . .   | 35 |
| 4.6.  | Modelos en Blender . . . . .   | 35 |
| 4.7.  | Modelo sofá en Blender . . . . .                                     | 36 |
| 4.8.  | Habitación amueblada . . . . .                                       | 36 |
| 4.9.  | Modelo de la aspiradora robótica realizado en Blender. . . . .       | 39 |
| 4.10. | Absorción por posición . . . . .                                     | 41 |
| 4.11. | Evaluador automático para el ejercicio aspiradora robótica . . . . . | 43 |
| 4.12. | Solución en Scratch . . . . .  | 43 |
| 4.13. | Solución en Python . . . . .   | 43 |
| 5.1.  | Juego del Pañuelo en competiciones robóticas . . . . .               | 46 |
| 5.2.  | Página de teoría enunciado y requisitos . . . . .                    | 47 |
| 5.3.  | Qué vas a aprender y teoría infrarrojos . . . . .                    | 47 |
| 5.4.  | Teoría infrarrojos . . . . .   | 48 |
| 5.5.  | Teroría sensor infrarrojos en Python . . . . .                       | 48 |
| 5.6.  | Teroría sensor infrarrojos en Scratch . . . . .                      | 49 |
| 5.7.  | Sensor dame objeto en pinza en Python . . . . .                      | 49 |
| 5.8.  | Sensor dame objeto en pinza en Scratch . . . . .                     | 50 |
| 5.9.  | Teroría actuadores en Python . . . . .                               | 50 |
| 5.10. | Teroría actuadores en Scratch . . . . .                              | 51 |
| 5.11. | ¿Sabías que.. ? . . . . .  | 51 |

|  |    |
|--|----|
| 5.12. Modelos en Blender . . . . .                             | 52 |
| 5.13. Prototipo robot con pinza en A-Frame nativo. . . . .     | 54 |
| 5.14. Modelo Mbot cambiado de color . . . . .                  | 55 |
| 5.15. Modelo Pinza . . . . .                                   | 56 |
| 5.16. Mallas de colisión del robot . . . . .                   | 58 |
| 5.17. Robot Mbot con pinzas . . . . .                          | 60 |
| 5.18. Evaluador automático del juego del pañuelo . . . . .     | 62 |
| 5.19. Soluciones de referencia . . . . .                       | 63 |
| <br>   |    |
| 6.1. Modelos portería en Blender y en Websim . . . . .         | 66 |
| 6.2. Pared de piedra en Websim . . . . .                       | 66 |
| 6.3. Modelo realizado en Teachable Machine . . . . .           | 68 |
| 6.4. Vista previa del modelo . . . . .                         | 69 |
| 6.5. Página web de prueba de reconocimiento de audio . . . . . | 72 |
| 6.6. Análisis audio de entrada con Teachable Machine . . . . . | 72 |
| 6.7. Ejercicio teleoperador acústico . . . . .                 | 77 |
| 6.8. Optimización del modelo . . . . .                         | 77 |
| 6.9. Ejemplo solución teleoperador acústico . . . . .          | 78 |
| 6.10. TensorFlow JS ejemplo detección de objetos. . . . .      | 79 |
| 6.11. Diagrama audio de salida . . . . .                       | 82 |

# Capítulo 1

## Introducción

Este Trabajo Fin de Grado se enmarca en el ámbito educativo, en concreto en la *gamificación* de una plataforma web de robótica educativa. Con el avance de la ciencia y la tecnología la sociedad ha cambiado y las formas de enseñar también. Este proyecto tiene como objetivo crear nuevos juegos y ejercicios que permitan enseñar a programar a los más jóvenes de una forma más divertida para fomentar el aprendizaje y la motivación, así como mejorar sus habilidades. Este proyecto engloba numerosas tecnologías web y herramientas que han servido para crear nuevos ejercicios para la plataforma de robótica educativa *Kibotics* [1].

En este primer capítulo se hace una breve introducción de la Robótica y de las tecnologías web, conceptos clave para ponernos en contexto y centrarnos en el tema principal de este trabajo: la robótica educativa y la “*gamificación*”.

### 1.1. Robótica

La Robótica es una rama de la ingeniería que combina las matemáticas, la electrónica, la mecánica, la informática y la física. Gracias a la unión de estas ramas de conocimiento se han podido desarrollar sistemas electromecánicos llamados robots.

Un robot es una máquina autónoma o semi-autónoma que es capaz de percibir su entorno, realizar cálculos para tomar decisiones, actuar en el mundo real de acuerdo con esas decisiones y comunicarse con otras máquinas o con humanos. Un robot se compone principalmente de 4 elementos:

- Sensores: para recibir información de su entorno (láser, lídar, ultrasonidos (Figura 1.1),

## 1.1. ROBÓTICA

---

infrarrojos, cámaras, radio frecuencia, sensores térmicos...).

- Actuadores: permiten la locomoción del robot por su entorno y manipulación de objetos ( motores eléctricos, de combustión, amortiguadores, hélices, servomotores (Figura 1.3), patas, ruedas, cadenas...).
- Controladores: Procesador (Figura 1.2) y algoritmos, para el cálculo y toma de decisiones.
- Leds, pantallas (Figura 1.4), antenas, altavoces...: para comunicarse con otros robots o con humanos.



Figura 1.1: Sensor ultrasonidos



Figura 1.2: Procesador



Figura 1.3: Servomotores



Figura 1.4: Pantalla

Los robots se clasifican según su entorno de aplicación en robots industriales y robots de servicio. También se pueden diferenciar según su forma en androides y zoomórficos, según su capacidad de movimiento en fijos o móviles, o según el medio en el que trabajan en terrestres, acuáticos o aéreos. Éstas son algunas de sus aplicaciones:

- Robots industriales: brazos y pinzas robóticas para ensamblado de piezas, envasado de alimentos, industria automovilística y gestión de almacenes.

## 1.1. ROBÓTICA

---

- Robots de servicio: destinados a limpieza del hogar, asistencia, coches autónomos, entretenimiento, usos militares, limpieza de centrales nucleares, investigación en terrenos hostiles o fines médicos.

En la Figura 1.5 se muestran algunos ejemplos de robots que son de gran utilidad en la sociedad actual.



(a) Dron



(b) Perseverance Mars



(c) Nao



(d) Brazo biónico



(e) Da Vinci [3]



(f) Roomba

Figura 1.5: Ejemplos de Robots

La Figura 1.5(a) muestra un dron, estos pequeños vehículos no tripulados han sido toda una revolución en la grabación de eventos y películas. El entretenimiento y el mundo audiovisual no es su única aplicación, también se utilizan para la búsqueda de personas y vigilancia.

El Perseverance Mars que vemos en la Figura 1.5(b) actualmente está en Marte haciendo investigaciones del terreno. Este robot busca signos de vida y guarda muestras para un futuro regreso a la Tierra.

Los robots de asistencia como el Nao, Figura 1.5 (c), son muy importantes para que los más pequeños y las personas mayores puedan socializar de una forma divertida y a su vez son una

## 1.1. ROBÓTICA

---

herramienta de apoyo en procesos de rehabilitación, post-operatorios y terapias ocupacionales.

Los brazos biónicos como el de la Figura 1.5(d) permiten recuperar las funciones y el tacto a amputados. El robot Da Vinci Figura 1.5(e) ayuda a los cirujanos a operar con mayor precisión y seguridad. Estos robots han sido un gran avance en el campo de la medicina.

El robot aspirador, uno de los más conocidos es *Roomba* (Figura 1.5(f)), es capaz de detectar obstáculos y residuos en el suelo. Es una gran ayuda para que la casa esté limpia y ahorra mucho tiempo.

Los robots que se han mencionado son solo algunos ejemplos. La robótica es una tecnología en auge. En los últimos años, con el avance de la tecnología, la realidad ha superado a la ficción, estamos rodeados de robots. Han salido de los laboratorios y han surgido miles de aplicaciones que nos hacen la vida un poco mejor.

Recientemente la robótica se está incorporando a las aulas para que los más pequeños adquieran conocimientos y habilidades importantes para su futuro. En la siguiente Figura 1.6 se muestran unos ejemplos de robots diseñados para el ámbito educativo [4].



(a) Bee-bot



(b) Makeblock Mbot



(c) LEGO MINDSTORMS

Figura 1.6: Ejemplos de robots educativos

## 1.2. Tecnologías Web

Las tecnologías web juegan un papel muy importante en el mundo moderno gracias a Internet. Esta plataforma WWW<sup>1</sup> [6] ha ido evolucionando y ha posibilitado potentes aplicaciones con un modelo cliente/servidor. En la Figura 1.7 podemos ver algunos ejemplos de aplicaciones web.

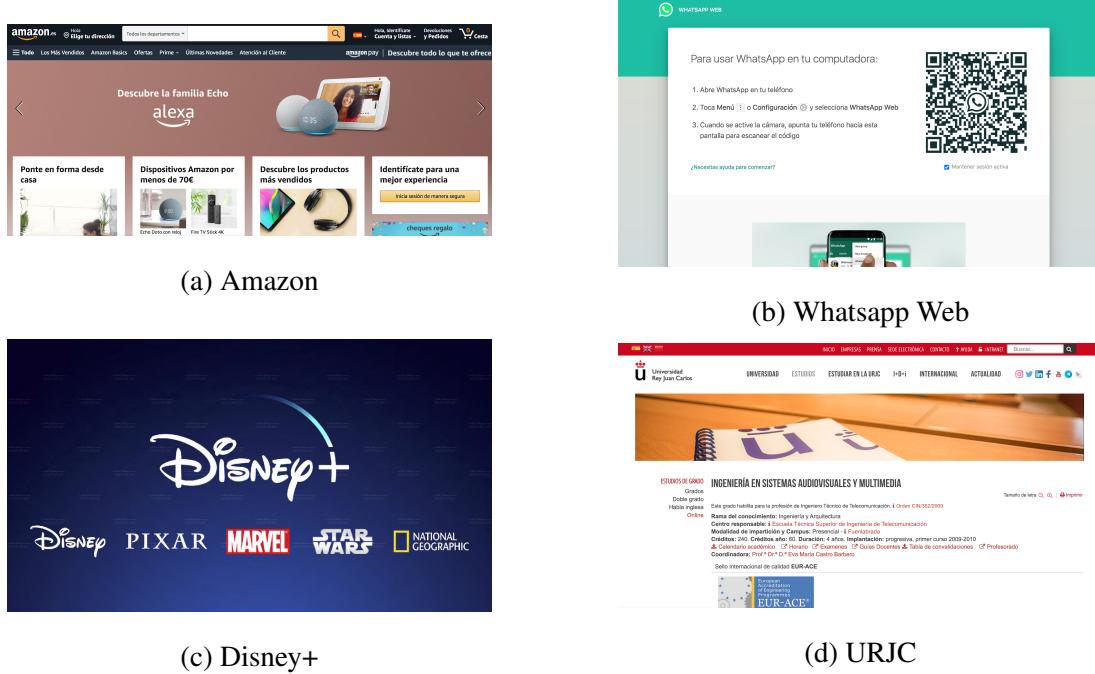


Figura 1.7: Ejemplos aplicaciones web

La web es una colección de documentos enlazados a través de hiperenlaces, cada recurso queda definido por su URL<sup>2</sup>. Cuando accedemos a la web a través del navegador, tenemos que introducir la dirección URL del sitio web al que nos queremos dirigir. El navegador enviará una solicitud al servidor con el protocolo HTTP<sup>3</sup>. El servidor le enviará a nuestro navegador un fichero HTML que quedará almacenado en nuestra máquina. Una vez el navegador obtiene el fichero HTML mostrará al usuario la página web principal de la URL que ha introducido. Si el navegador detecta que hay imágenes, vídeos u otros ficheros, volverá a mandar peticiones HTTP al servidor para que éste le envíe toda la información necesaria. En la Figura 1.8 podemos ver una representación de cómo es la comunicación entre cliente y servidor.

<sup>1</sup>World Wide Web

<sup>2</sup>Uniform Resource Locator

<sup>3</sup>Hyper Text Transfer Protocol

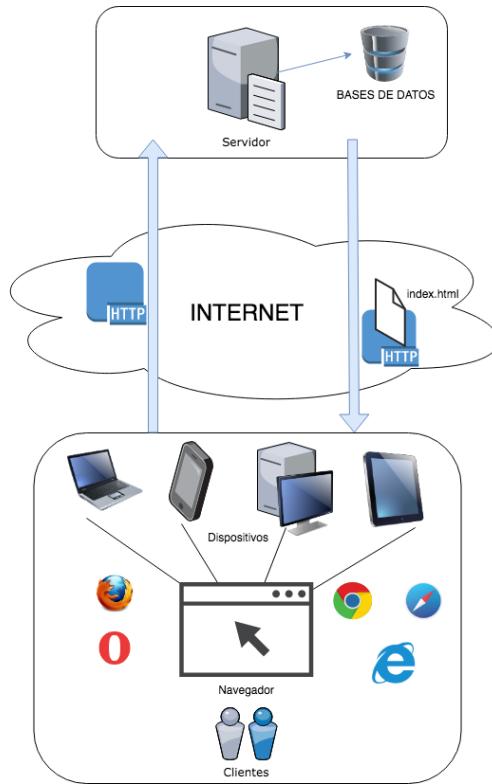


Figura 1.8: Comunicación cliente/servidor a través de HTTP

HTTP es un protocolo entre navegadores y servidores web para transferir documentos de hipertexto. El cliente envía mensajes de solicitud y el servidor manda mensajes de respuesta, ambos mensajes son del mismo formato (ver Figura 1.9 y Figura 1.10). Los tipos de mensaje más comunes de este protocolo son GET, POST, PUT, DELETE y HEAD. Este protocolo utiliza códigos de estado, los más conocidos son: 200 OK, que significa resultado exitoso, 500 Server Error, cuando hay un error en el lado servidor y el más conocido 404 Not Found, cuando hay un error en la parte cliente [7].

Línea inicial de petición  
GET /dir/pagina.html HTTP/1.1  
Host: www.uni.edu  
User-agent: Mozilla/4.0  
Connection: close  
Accept-language: es

Línea inicial de respuesta  
HTTP/1.1 200 OK  
Connection: close  
Date: Thu, 06 Aug 1998 12:00:15 GMT  
Server: Apache/1.3.0 (Unix)  
Last-Modified: Mon, 22 Jun 1998 ...  
Content-Length: 6821  
Content-Type: text/html  
Cuerpo del mensaje  
datos datos datos datos ...  
datos datos datos datos datos ...  
datos datos datos datos ...

Figura 1.9: Ejemplo petición HTTP  
Cliente a Servidor

Figura 1.10: Ejemplo respuesta HTTP  
Servidor a Cliente

## 1.2. TECNOLOGÍAS WEB

---

Una aplicación web está formada por dos partes independientes entre sí, las tecnologías del lado cliente y las del lado servidor. Las tecnologías del lado cliente (*frontend*) se encargan de interactuar con el usuario, visualizar el contenido y establecer la comunicación con el servidor. Se ejecutan en el navegador, que actúa como intérprete. Por otro lado, las tecnologías del lado servidor (*backend*) se encargan de la administración del sitio web, usando bases de datos y gestores de contenidos.

Una de las principales ventajas de usar tecnologías web es que las aplicaciones creadas son multiplataforma y multidispositivo, funcionan tanto en ordenadores, móviles, tabletas, así como en distintos sistemas operativos. Otra ventaja es que no tenemos que instalar nada, sólo necesitamos el navegador y además la actualización del contenido es inmediata. El principal inconveniente es su dependencia de Internet, pero con los últimos avances tecnológicos el Wifi, la fibra óptica y el 5G han permitido que la mayoría de personas del mundo podamos acceder desde cualquier lugar y éste no sea un gran inconveniente.

### 1.2.1. Tecnologías Web lado cliente

Las tecnologías web del lado cliente permiten la interacción del usuario con la página web que corre en el navegador del usuario. Para ello, se usan principalmente estas tres tecnologías [8]:

- HTML5<sup>4</sup>: es un lenguaje de marcado de los contenidos de un sitio web, se usa para asignar la función de cada elemento. Es el esqueleto de la web.
- JavaScript: es un lenguaje de programación interpretado que se encarga del comportamiento de una página web y de su interactividad con el usuario.
- CSS3<sup>5</sup>: es un lenguaje de hojas de estilo creado para controlar la presentación de la página: colores, tipo de letra, tamaños, animaciones, colocación de los elementos...

Entraremos en más detalle en estas tecnologías en el capítulo 3 donde se habla de la Infraestructura utilizada.

---

<sup>4</sup>HyperText Markup Language

<sup>5</sup>Cascading Style Sheets

### 1.2.2. Tecnologías Web lado servidor

Las tecnologías web del lado servidor son las que permiten gestionar y servir las páginas web y acceder a bases de datos. En este caso las tecnologías son más flexibles y vamos a nombrar tres de las más utilizadas:

- Django: es un entorno para crear servidores web de alto nivel, que fomenta el desarrollo rápido con un diseño limpio y práctico en Python, destaca por su arquitectura basada en modelo-vista-controlador y el uso de plantillas. De esta forma puedes centrarte en crear tu aplicación web sin grandes complicaciones. Es gratis y de código abierto[9]. Un ejemplo de aplicación web que utiliza Django es Instagram [10].
- Node.js: es un entorno de ejecución para JavaScript orientado a eventos síncronos, construido con el motor de JavaScript V8 de Chrome. Diseñado para aplicaciones web escalables. De esta forma el cliente y el servidor pueden crearse con el mismo lenguaje de programación [11]. Netflix, Paypal o LinkedIn usan esta tecnología para sus servidores[12].
- PHP<sup>6</sup>: es un lenguaje de *scripting* de uso general popular que es especialmente adecuado para el desarrollo web [13]. Rápido, flexible y práctico, gracias a su capacidad de creación de webs dinámicas, desde blogs hasta sitios web como Facebook o Wikipedia [14].

En el lado servidor se utilizan bases de datos. Una base de datos es una colección de datos estructurados. Entre ellas destacan mySQL (Figura 1.11 (a))y MongoDB (Figura 1.11 (b)).

```
mysql>
mysql>
mysql> select * from books;
+----+-----+-----+-----+-----+
| id | title          | author        | sales | publication_date |
+----+-----+-----+-----+-----+
| 1  | Don Quijote    | Miguel de Cervantes | 500  | 1962 |
| 2  | Historia de los ciudades | Charles Dickens | 200  | 1859 |
| 3  | El Señor de los Anillos | J. R. R. Tolkien | 150  | 1954 |
| 4  | El principito   | Antoine de Saint-Exupéry | 150  | 1943 |
| 5  | El hobbit       | J. R. R. Tolkien | 100  | 1937 |
| 6  | Sueño en el pabellón rojo | Cao Xueqin | 100  | 1759 |
| 7  | Triple representatividad | Jiang Zemin | 100  | 2001 |
| 8  | Diez negritos   | Agatha Christie | 100  | 1930 |
| 9  | NULL            | NULL           | NULL  | NULL  |
| 10 | El código Da Vinci | Dan Brown | 80   | 2003 |
| 11 | El león, la bruja y el armario | C. S. Lewis | 85   | 1956 |
+----+-----+-----+-----+-----+
11 rows in set (0.00 sec)
```

(a) mySQL

```
{
  title: "Java in action",
  author: "author1",
  language: "English",
  publisher: {
    name: "My publications",
    founded: 1990,
    location: "SF"
  }
}

{
  title: "Hibernate in action",
  author: "author2",
  language: "English",
  publisher: {
    name: "My publications",
    founded: 1990,
    location: "SF"
  }
}
```

(b) MongoDB

Figura 1.11: Bases de Datos

<sup>6</sup>Hypertext Preprocessor

## 1.3. Robótica educativa

La Robótica educativa es un sector de aprendizaje multidisciplinar. Ayuda a desarrollar competencias y habilidades como: la innovación y espíritu emprendedor, la resolución de problemas y lógica, la toma de decisiones, conocimientos de herramientas relacionadas con las tecnologías digitales, el pensamiento crítico, creatividad, el trabajo colaborativo y cooperativo, y la flexibilidad y adaptabilidad al trabajo [20].

Ante la falta de estudiantes en carreras técnicas en la actualidad, la robótica educativa puede ofrecer una gran motivación a los alumnos de las primeras etapas de educación: Primaria, ESO y Bachillerato, para fomentar la creatividad y la curiosidad al mostrar la ciencia y la tecnología de una forma diferente e incrementar sus habilidades a la vez que sus conocimientos desde los fundamentos STEM (*Science, Technology, Engineering and Mathematics*).

Gracias a las tecnologías web son muchas las aplicaciones que ofrecen cursos de robótica para todos los niveles educativos, muchos ayuntamientos están comprando cursos y materiales para facilitar a los más pequeños su introducción al mundo de la robótica a través de clases extraescolares. En secundaria se está introduciendo poco a poco en las asignaturas de tecnología el uso de lenguajes de programación, en el que destaca el lenguaje Scratch. En la Figura 1.12 podemos ver la interfaz de Scratch para programar desde el navegador.

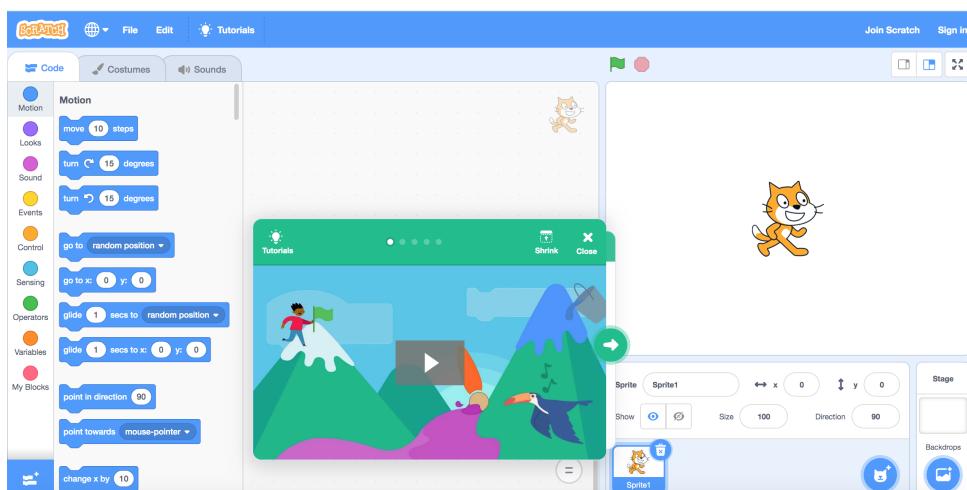


Figura 1.12: Scratch

Scratch es un lenguaje de programación visual basado en bloques, creado y mantenido por Lifelong Kindergarten group en el MIT Media Lab. Scratch además es una comunidad en línea

### 1.3. ROBÓTICA EDUCATIVA

---

donde los niños pueden programar y compartir sus historias, juegos y animaciones con gente de todo el mundo. Gracias a esto, los más pequeños aprenden a pensar con creatividad, trabajar en equipo y razonar [21]. Scratch posee un lenguaje de iniciación llamado Scratch Jr pensando para niños de 5 a 7 años siendo aún más sencillo, aunque Scratch está pensado para todas las edades. Actualmente se puede utilizar desde cualquier dispositivo al utilizar tecnologías web.

Junto con Scratch cada vez hay más plataformas y entornos STEM que se han dedicado al desarrollo de herramientas de aprendizaje enfocadas a los más pequeños. Destacan aplicaciones web como:

- *OpenRoberta*<sup>7</sup>: es una plataforma web creada por un instituto alemán perteneciente a la Fraunhofer Society. Tiene como objetivo simplificar conceptos de programación y facilitar a niños y profesores la codificación mediante el uso de robots como Lego Mindstorms y otros sistemas de hardware programables con su propia interfaz (Figura 1.13) [22].

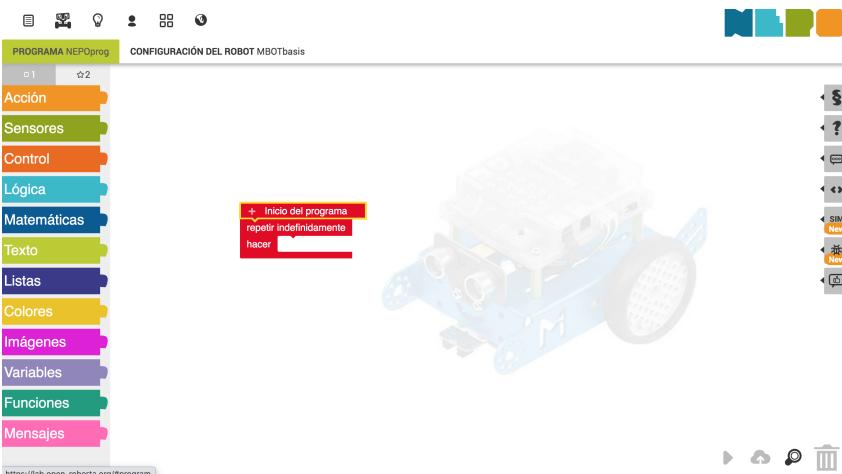


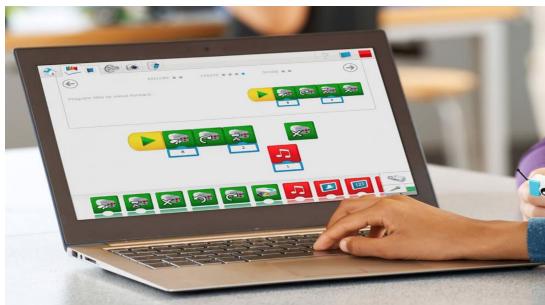
Figura 1.13: Open Roberta

- *LEGO Education*: la plataforma LEGO ofrece una amplia variedad de robots y *packs* para uso escolar. Sus kits de robótica educativa permiten a los más pequeños construir y programar robots mediante el uso de motores, sensores, engranajes, ruedas, ejes y otros componentes técnicos, además del uso de su propio software basado en bloques (Figura 1.14 (a)). Destacan modelos como el MINDSTORMS Education EV3 y LEGO Education WeDo 2.0 (Figura 1.14 (b)) [23] [24].

---

<sup>7</sup><https://lab.open-roberta.org/>

### 1.3. ROBÓTICA EDUCATIVA



(a) LEGO



(b) WeDo 2.0

Figura 1.14: LEGO EDUCATION

- *MBlock IDE*<sup>8</sup>: esta aplicación web está diseñada para la educación en ciencia, tecnología, ingeniería, artes y matemáticas (STEAM). Está inspirada en Scratch 3.0, es compatible con lenguajes de programación tanto gráficos (Scratch) como textuales (Python). Se pueden diseñar historias, juegos, animaciones (Figura 1.15) y programar dispositivos como robots Makeblock y microbit [25].

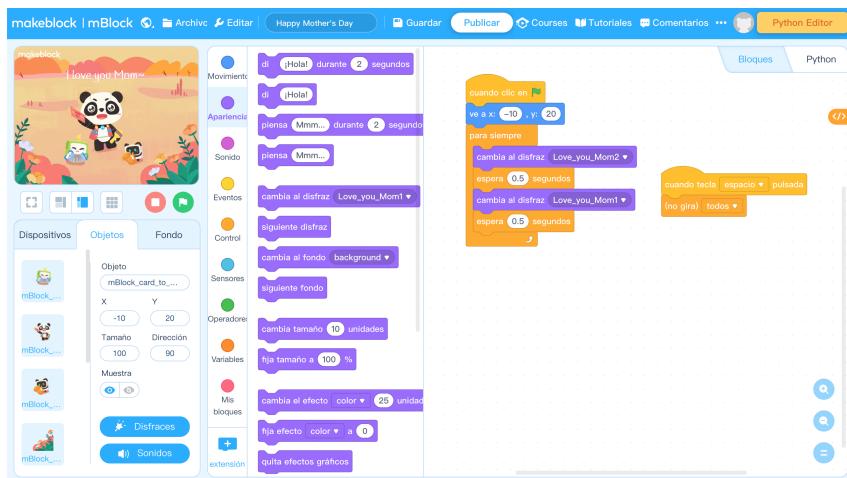


Figura 1.15: MBlock IDE

- Vex<sup>9</sup>, AppInventor<sup>10</sup>, Arduino Web Editor<sup>11</sup>, Kodu<sup>12</sup> o Snap!<sup>13</sup> son otras plataformas web de programación en robótica educativa.

<sup>8</sup><https://ide.mblock.cc/>

<sup>9</sup><https://www.vexrobotics.com/>

<sup>10</sup><https://appinventor.mit.edu/>

<sup>11</sup><https://store.arduino.cc/digital/create>

<sup>12</sup><http://www.kodugamelab.com/>

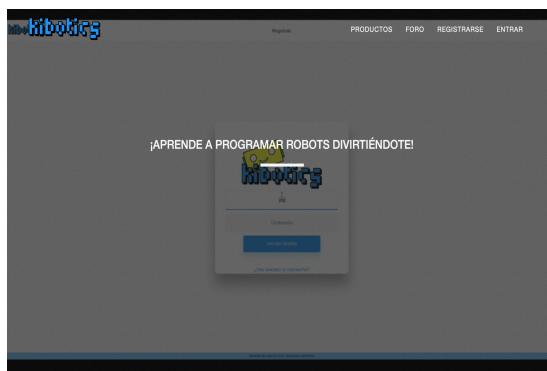
<sup>13</sup><https://snap.berkeley.edu/>

### 1.3. ROBÓTICA EDUCATIVA

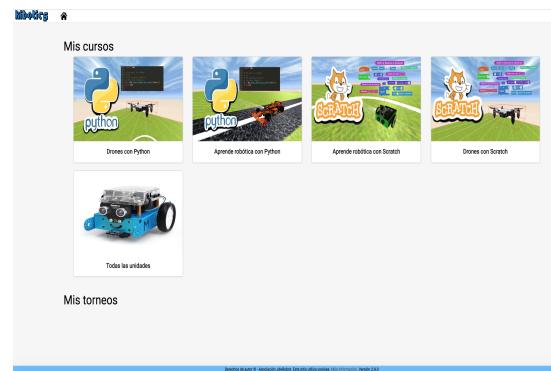
---

La plataforma de robótica educativa que vamos a utilizar en este TFG es Kibotics. Esta plataforma, basada en tecnologías web, enseña de manera atractiva conceptos básicos de tecnología e inicia a niños y adolescentes en robótica y programación. Sigue un enfoque práctico, fomentando el pensamiento y la organización para resolver un problema, además de formar el espíritu analítico de los alumnos [1].

Actualmente ofrece cursos de robótica en Scratch y Python (Figura 1.16). En el capítulo 3 se ampliará información sobre Kibotics y veremos cómo se introduce la *gamificación* en la plataforma a lo largo de este trabajo.



(a) kibotics.org



(b) Cursos disponibles en Kibotics.

Figura 1.16: Plataforma Kibotics.

#### 1.3.1. Importancia de los juegos y multimedia en el aprendizaje

El planteamiento actual del sistema educativo tiene carencias y la innovación en la educación cobra cada vez más importancia. Es por ello que los docentes buscan actividades enfocadas en evitar la recurrente pasividad de los alumnos [15].

Según concluyen el cono del aprendizaje de Edgar Dale y otros estudios, aprendemos un 10 % de lo que leemos, 20 % de lo que escuchamos, 75 % de lo que vemos y oímos y 90 % de lo que hacemos [17] [16]. Estos porcentajes indican que, por lo tanto, la introducción del vídeo y los juegos interactivos en la clase puede producir modificaciones primordiales en el ámbito educativo.

La multimedia es un valioso recurso en la enseñanza por su naturaleza interactiva. Estos materiales deben ser adecuados y facilitar el aprendizaje. Deben ser de fácil instalación y uso, interactivos, versátiles, mostrar información de calidad para motivar a los alumnos. Cada mate-

### **1.3. ROBÓTICA EDUCATIVA**

---

rial se ajusta a los usuarios para que cada uno trabaje a su ritmo [18].

Las nuevas tecnologías están cada vez más interiorizadas en las aulas gracias al uso de pantallas, proyectores y ordenadores. La pandemia vivida en este último año ha impulsado el uso de aplicaciones web para dar clase, como las plataformas de videoconferencia. Muchos profesores han optado por estas potentes plataformas, otros han grabado y compartido sus propios vídeos, donde los estudiantes pueden pararlos y verlos las veces necesarias para interiorizar mejor los conocimientos.

En los últimos años se han incorporando juegos en las aulas como la aplicación Kahoot! [19] en la que hacen juegos de encuestas, además se ha fomentado aprendizaje a través de vídeos y diapositivas más animadas, foros, así como el uso de plataformas educativas e interactivas como Moodle.

El término “*gamificación*” o ludificación se emplea para referirse al aprendizaje a través de juegos en el entorno educativo y profesional. Los juegos se utilizan para fomentar el aprendizaje de programación, mejorando los conocimientos y habilidades de los alumnos de una forma más dinámica y divertida. La *gamificación* facilita la interiorización de los conocimientos, generando una respuesta positiva al usuario por cumplir con un objetivo.

Enseñar a los más jóvenes cómo funcionan las últimas tecnologías y que además les parezca entretenido e interesante es lo que nos ha motivado a realizar este trabajo.

#### **1.3.2. Campeonatos de robótica educativa existentes**

El aprendizaje de la robótica educativa ha ido más allá. Actualmente existen competiciones a nivel nacional e internacional que utilizan juegos y ejercicios competitivos [26]. Estas son algunas de ellas:

- RoboCup Junior<sup>14</sup>: es una iniciativa educativa orientada a proyectos que patrocina eventos robóticos locales, regionales e internacionales para jóvenes estudiantes. Destaca la Liga de Rescate, Liga de Fútbol (Figura 1.17(b)) y Liga ONSTAGE [27].
- Eurobot Junior<sup>15</sup>: es una competición europea de robots para estudiantes de primaria,

---

<sup>14</sup><https://junior.robocup.org/>

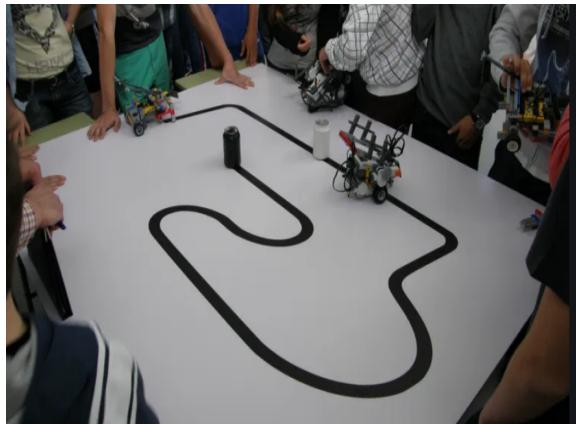
<sup>15</sup><http://www.eurobot.es/>

### 1.3. ROBÓTICA EDUCATIVA

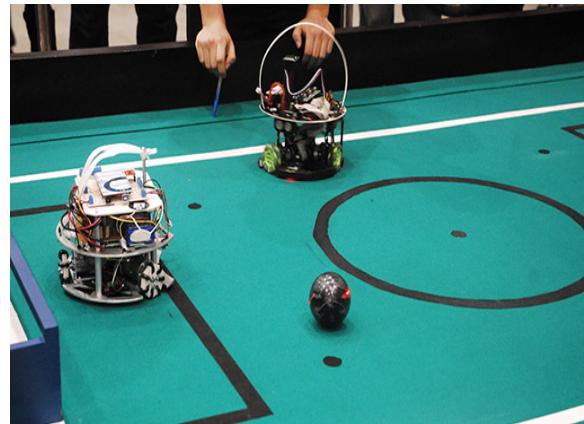
---

secundaria o clubs de robótica. El grupo de jóvenes debe diseñar, construir y programar un robot telecontrolado por cable. Además pueden tener un robot secundario autónomo [28].

- First Lego league<sup>16</sup>: es un programa internacional para jóvenes de 4 a 16 años a través de la resolución de problemas reales. Se adapta a cada edad con sus cursos Discover, Explore y Challenge [29].
- Torneo Nacional VEX Robotics IQ<sup>17</sup>: destinado a equipos de 4 a 8 miembros de secundaria junto un adulto. Ofrecen distintos retos y torneos con premios [30].
- RoboCampeones<sup>18</sup>: Creado en el RoboticsLabURJC de la Universidad Rey Juan Carlos. Los alumnos de instituto compiten en pruebas como sumo con robots LEGO y Arduino. Se celebra en Fuenlabrada y en los últimos años contó con más de 2000 participantes (Figura 1.17(a)) [31].



(a) Juego Pañuelo Robocampeones



(b) Fútbol en RoboCup Junior.

Figura 1.17: Campeonatos de robótica.

---

<sup>16</sup><https://www.firstlegoleague.es/>

<sup>17</sup><https://vexspain.com/>

<sup>18</sup><http://robocampeones.org/>

## 1.4. Estructura del documento

La estructura de este trabajo fin de grado está compuesta por los siguientes capítulos:

- *Capítulo 1 Introducción*: una breve introducción a la robótica y tecnologías web para ponernos en contexto y presentar el tema principal del trabajo.
- *Capítulo 2 Objetivos*: Se fijan los objetivos concretos y se explica la metodología y plan de trabajo seguidos a lo largo de este proyecto.
- *Capítulo 3 Infraestructura utilizada*: se describen lenguajes, tecnologías y herramientas empleadas en este trabajo.

Para una mejor exposición del trabajo que se ha realizado, el desarrollo se ha dividido en 3 capítulos. De esta forma nos centraremos en el enunciado, la infraestructura y solución de referencia de cada ejercicio realizado:

- *Capítulo 4 Ejercicio Aspiradora robótica atrapa confeti*
- *Capítulo 5 Ejercicio Juego del pañuelo*
- *Capítulo 6 Ejercicio Teleoperador Acústico y Banda Sonora*
- *Capítulo 7 Conclusiones y trabajos futuros*: Conclusiones del trabajo y futuros proyectos posibles a partir de éste.

# **Capítulo 2**

## **Objetivos**

En este capítulo se plantean los objetivos a cumplir con este proyecto, así como los requisitos, la metodología y el plan de trabajo que se ha seguido para alcanzarlos.

### **2.1. Objetivos**

El objetivo principal de este trabajo es introducir la *gamificación* en la plataforma Kibotics. Lo hemos articulado en los siguientes tres subobjetivos u objetivos específicos a cumplir:

- Diseñar y desarrollar un nuevo ejercicio sobre una aspiradora robótica que tiene que limpiar una habitación.
- Diseñar y desarrollar un nuevo ejercicio sobre un robot que juega al pañuelo, recorriendo una línea, recogiendo una lata que ejerce de pañuelo y regresando con ella al lugar de partida.
- Diseñar y desarrollar un nuevo juego que analice el audio en tiempo real y explorar la posibilidad de añadir bandas sonoras a los ejercicios actuales.

Para los tres ejercicios se desarrollará la infraestructura necesaria, modelos nuevos de robots, escenarios en el simulador, evaluadores automáticos, soluciones de referencia y se integrarán en la plataforma web de robótica educativa Kibotics.

## 2.2. Requisitos

Para cumplir con los objetivos citados anteriormente debemos tener en cuenta además los siguientes requisitos:

- Los robots y juegos desarrollados deben ser compatibles con la versión actual v.2.8 o superior de Kibotics.
- No se debe requerir de instalaciones adicionales. Todo debe correr en el navegador web del cliente.
- Uso de software de simulación Websim y A-Frame.

## 2.3. Metodología

La metodología que se ha seguido para la realización de este trabajo es la basada en el modelo de desarrollo software iterativo y creciente (Figura 2.1) [40]. Este modelo consiste en entregar a los usuarios y al equipo de desarrolladores de Kibotics una versión disponible lo antes posible y en continua actualización. En cada iteración se van solventando pequeños errores y mejoras que convergen en la versión final de cada ejercicio y del proyecto.

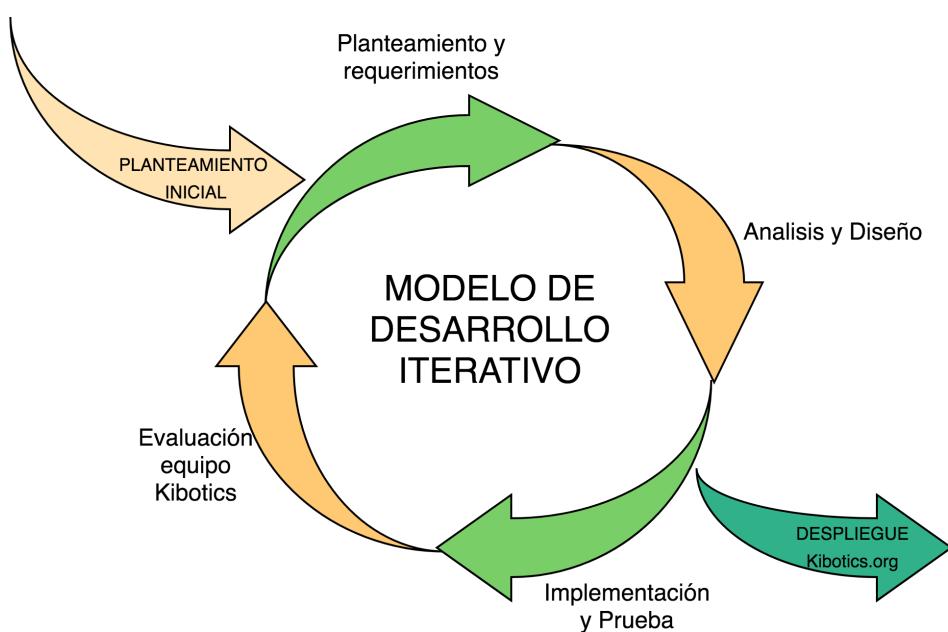


Figura 2.1: Modelo iterativo

## 2.3. METODOLOGÍA

El código desarrollado y las mejoras, se han integrado progresivamente en el código fuente oficial de Kibotics en GitHub<sup>1</sup>. Mediante su flujo de trabajo incidencia (*issue*), rama (*branch*) y parche (*pullrequest*), la plataforma oficial está siempre actualizada con los últimos cambios añadidos y verificados por el equipo de desarrolladores.

Para llevar acabo esta metodología se establecieron reuniones semanales con el tutor para la orientación de este trabajo fin de grado. A lo largo del proyecto se ha mantenido la comunicación con el tutor y el equipo de Kibotics a través de la plataforma Slack<sup>2</sup>.

Además se ha creado y mantenido una página web (Figura 2.2) tipo blog para llevar un seguimiento de las tareas realizadas y los objetivos semanales.<sup>3</sup>

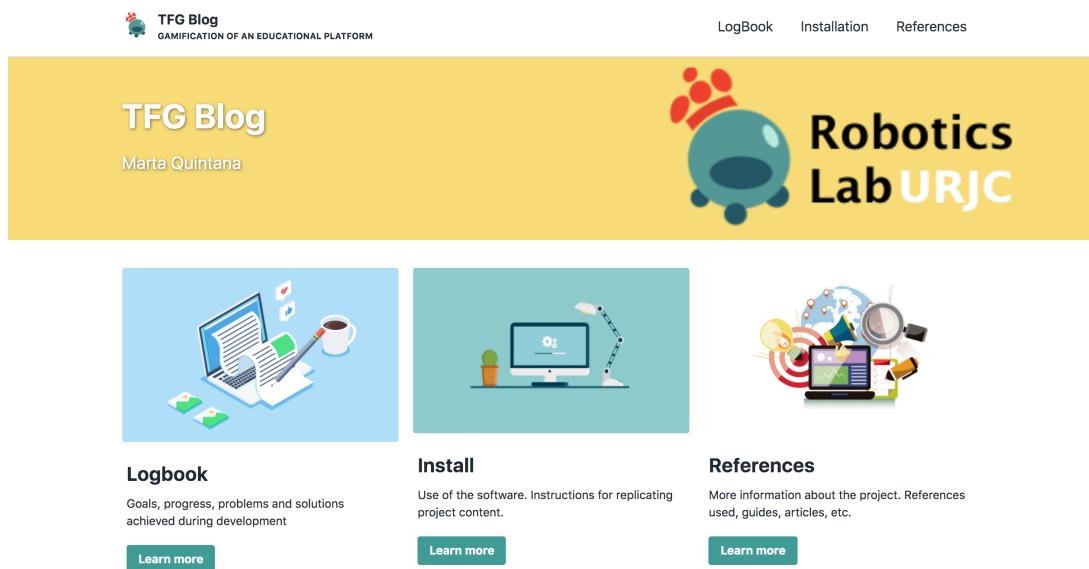


Figura 2.2: Página web de este TFG

<sup>1</sup><https://github.com/>

<sup>2</sup><https://slack.com/>

<sup>3</sup><https://roboticslaburjc.github.io/2020-tfg-marta-quintana/>

## 2.4. Plan de Trabajo

El plan de trabajo seguido durante este proyecto se puede dividir en las siguientes etapas:

- **Aterrizaje en Kibotics y repaso de tecnologías web:** Toma de contacto con Kibotics y repaso de HTML5, JavaScript, Django y otras tecnologías que se utilizan en la plataforma.
- **Estudio Web Audio API y Tensor FlowJS:** Realización de ejercicios y tutoriales de distintas herramientas para introducir sonido y reconocimiento de audio en Kibotics.
- **Diseño Teleoperador Acústico con Teachable Machine:** Creación de un teleoperador acústico con reconocimiento de audio.
- **Prototipo de aspiradora robótica:** Diseño y creación de modelos del nuevo ejercicio.
- **Diseño ejercicio aspiradora robótica confeti:** Desarrollo de una aspiradora robótica simulada, un nuevo actuador de absorción y piezas de confeti que puedan ser absorbidas por la aspiradora.
- **Prototipo de un robot Mbot con pinzas basado en A-Frame:** Estudio de un prototipo en A-Frame nativo para el estudio de las físicas y mallas de colisión.
- **Preparación del ejercicio del pañuelo:** Creación e implementación del ejercicio, desarrollo del circuito, una lata y un Mbot con pinzas realizados con Blender y JavaScript, para ofrecernos físicas más realistas.

# Capítulo 3

## Infraestructura utilizada

En este capítulo vamos a profundizar en las tecnologías y herramientas que se han empleado a lo largo de este trabajo. Por un lado se explican los lenguajes de programación (JavaScript, Python y Scratch) y los lenguajes de documentos (HTML5 y JSON) utilizados. Por otro lado, hablaremos de TensorFlowJS, Web Audio API y Teachable Machine, tecnologías que nos ofrecen procesamiento de audio en aplicaciones web. Otras aplicaciones como Blender y A-Frame han sido fundamentales para el modelado 3D y realidad virtual. Finalmente, hablaremos de la plataforma Kibotics, su estructura y las tecnologías web que la componen.

### 3.1. Lenguajes de programación y de documentos

#### 3.1.1. HTML5

HTML5 es la última versión de HTML cuyas siglas corresponden a *HyperText Markup Language* es el bloque de construcción más básico de la Web [42]. *HyperText* significa hipertexto, se refiere a enlaces que conectan páginas web entre sí, ya sea dentro de un único sitio web o entre sitios web. *Markup* significa marca o etiqueta, ya que todas las páginas web están construidas en base a etiquetas. Con ellas se anota texto, imágenes y otros contenidos para mostrar en un navegador web. El formato HTML5 incluye etiquetas como: <head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, <div>, <span>, <img>, <aside>, <audio>, <canvas>, <datalist>, <details>, <embed>, <nav>, <output>, <progress>, <video>, <ul>, <ol> y <li>, entre otras. En la Figura 3.1(a) podemos ver cómo es el uso de las etiquetas para

### 3.1. LENGUAJES DE PROGRAMACIÓN Y DE DOCUMENTOS

---

la creación de una página web y en la Figura 3.1(b) cómo se visualizaría en el navegador.

Hablamos de *Language* porque HTML es un lenguaje, tiene sus normas, tiene su estructura y una serie de convenciones que nos sirven para definir tanto la estructura como el contenido de una web. Esto no quiere decir que sea un lenguaje de programación. HTML no tiene estructuras de lenguaje de programación, como los bucles, las condiciones, las funciones, etcétera. HTML5 es un estándar que sirve para definir la estructura y el contenido de una página Web [43].

```
html5_image.png index.html ×  
1 <!DOCTYPE html>  
2 <html lang="es" dir="ltr">  
3   <head>  
4     <meta charset="utf-8">  
5     <title>Mi pagina web</title>  
6   </head>  
7   <body>  
8     <h1>Esto es una página web</h1>  
9       
10   </body>  
11 </html>
```



(a) Fuente de la página web

(b) Página web visualización

Figura 3.1: Ejemplo de página web con HTML5

Generalmente junto con HTML se utilizan otras dos tecnologías web: CSS para la apariencia/presentación y JavaScript para la funcionalidad/comportamiento de una página web.

Kibotics utiliza plantillas basadas en HTML5 en las páginas web que sirve. En este TFG se va a utilizar HTML5 para modificar y crear las plantillas de las páginas web que mostrarán los nuevos ejercicios que vamos a introducir en la plataforma.

#### 3.1.2. JavaScript

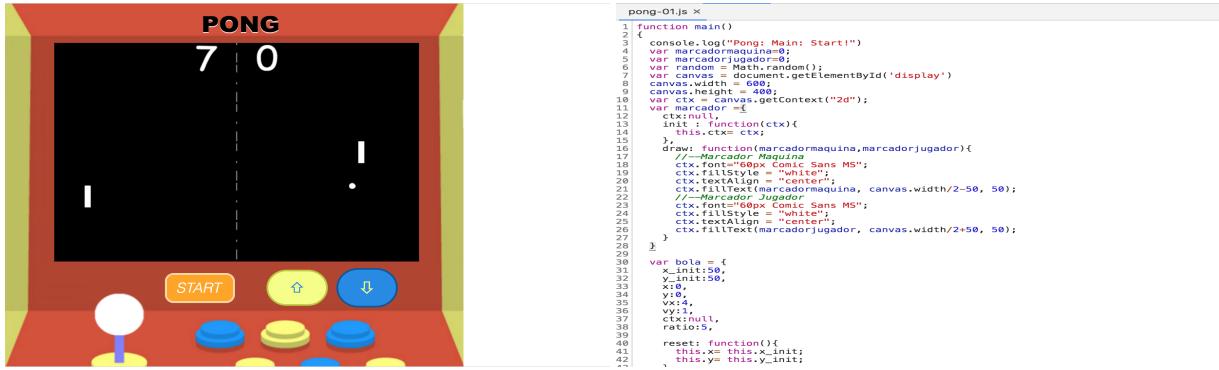
JavaScript es un lenguaje de programación interpretado, no necesita compilar los programas para ejecutarlos, pero sí un intérprete que en nuestro caso es el navegador. JavaScript sigue el estándar ECMAScript que se encarga de regir cómo debe ser interpretado, su operatividad y buen funcionamiento. Este lenguaje posee una sintaxis derivada de C y Java pero no tiene nada que ver con ellos.

Se utiliza principalmente para crear páginas web interactivas, que incorporan efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y

### 3.1. LENGUAJES DE PROGRAMACIÓN Y DE DOCUMENTOS

ventanas con mensajes de aviso al usuario [41].

El código JavaScript se ejecuta en el navegador del cliente y permite que éste interactúe con la página web. En la siguiente Figura 3.2 podemos ver un ejemplo de página web interactiva.<sup>1</sup>



(a) Página web visualización

(b) Código JavaScript

Figura 3.2: Ejemplo de página web interactiva con HTML5, CSS3 y JavaScript

En este trabajo JavaScript ha sido una pieza clave para su desarrollo y todos los ejercicios se han creado usando este lenguaje de programación.

#### 3.1.3. Python y Scratch

Python es un lenguaje de programación orientado a objetos, es un lenguaje interpretado o de *script*, y al igual que JavaScript necesita un programa intermedio que haga de intérprete. Es un lenguaje fuertemente tipado y multiplataforma. Su filosofía es que el código sea legible y tenga una sintaxis muy limpia. Sencillo pero potente, lenguaje muy usado en la universidad y en el ámbito laboral [44].

El servidor de Kibotics está programado en Python (usa la tecnología web Django). Los ejercicios disponibles en la plataforma se pueden resolver también con este lenguaje. Para usar Python, las páginas web de los ejercicios contienen el editor ACE. ACE es un editor de código incrustable escrito en JavaScript [45]. Esto permite al usuario escribir el código de la solución del ejercicio en Python e internamente este código se traduce a JavaScript para el correcto funcionamiento de los ejercicios.

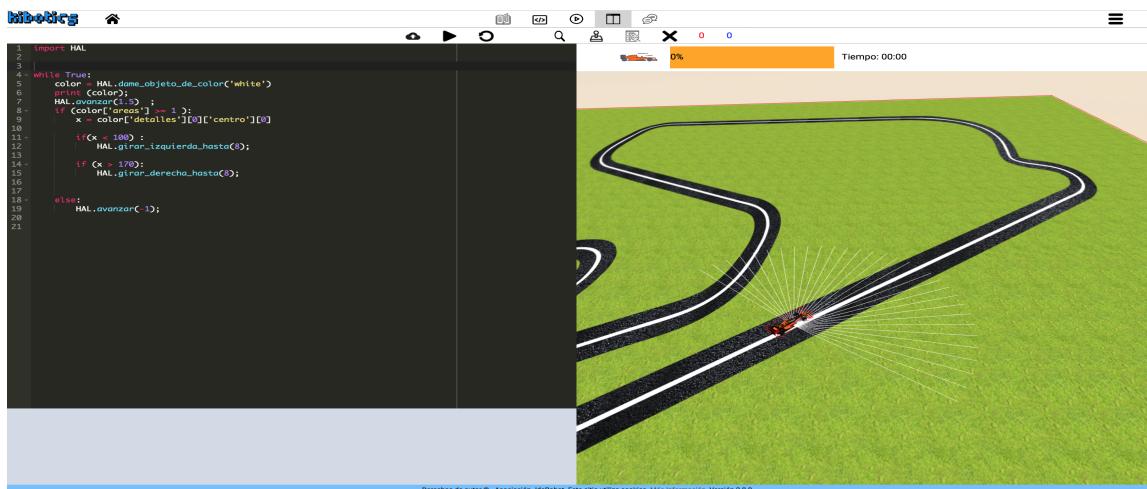
Como hemos comentado en la introducción, Scratch es un lenguaje de programación visual basado en bloques, creado y mantenido por Lifelong Kindergarten group en el MIT Media Lab.

<sup>1</sup><https://martaquintana.github.io/2018-19-CSAAI-Pong/pong-01.html>

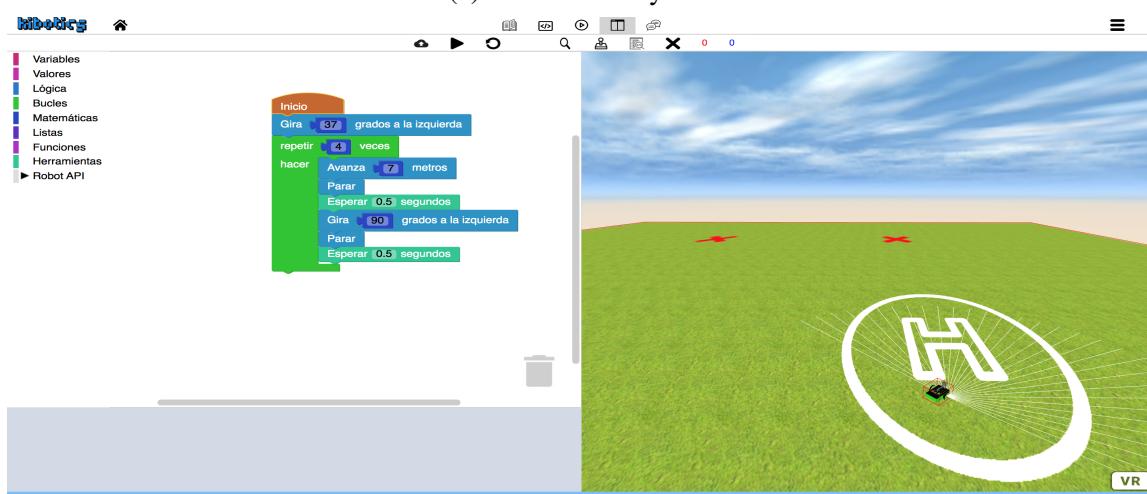
### 3.1. LENGUAJES DE PROGRAMACIÓN Y DE DOCUMENTOS

Para usar Scratch en Kibotics se usa Blocky, que es una librería de JavaScript desarrollada por Google. Permite usar en una página web un editor de código visual. Es compatible con Chrome, Firefox, Safari, Opera y otros navegadores. Corresponde a tecnologías del lado cliente y no tiene dependencia con el servidor [46].

Los ejercicios planteados en este TFG están resueltos tanto en Python como en Scratch. Al ser una plataforma para el ámbito educativo, estos dos lenguajes son una buena alternativa para empezar a programar porque son lenguajes sencillos y con una curva de aprendizaje muy suave. En la Figura 3.3 podemos ver unos ejemplos de soluciones, la Figura 3.3 (a) para el sigue líneas en Python y la Figura 3.3 (b) para el ejercicio del cuadrado en Scratch.



(a) Solución en Python



(b) Solución en Scratch

Figura 3.3: Ejemplo de código en Kibotics

### 3.1.4. JSON

JSON, cuyas siglas significan *JavaScript Object Notation*, es un formato de intercambio de datos muy ligero. Para nosotros es fácil de leer y escribir, además la interpretación y generación de ficheros es muy sencilla para las máquinas [47].

JSON es un formato de datos basado en texto estándar para representar datos estructurados con la sintaxis de objetos de JavaScript. Son archivos de texto plano con codificación UTF8, que son compatibles con todos los sistemas. Se utiliza para transmitir datos en aplicaciones web [48]. Este formato puede ser utilizado independientemente de JavaScript, y muchos entornos de programación poseen la capacidad de leer (convertir; *parsear*) y generar ficheros JSON. En nuestro caso vamos a leer ficheros JSON desde JavaScript.

Un fichero JSON típicamente está compuesto de dos estructuras:

- **Una colección de pares de nombre/valor:** En otros lenguajes son conocidos como objeto, registro, estructura, diccionario o lista de claves. Ejemplo objeto:

```
1  {
2      "id" : 7,
3      "name" : "Robot",
4      "type" : "Drone"
5 }
```

- **Una lista ordenada de valores:** En la mayoría de los lenguajes, esto se implementa como arreglos *arrays*, vectores o listas. Ejemplo *array*:

```
1 [ "blue", "yellow", "orange" ]
```

Estas estructuras son universales en todos los lenguajes de programación, es por esto que JSON es muy fácil utilizar para un programador. Todos los lenguajes disponen de funciones para interpretar cadenas JSON y convertir datos en cadenas JSON válidas.

En la Figura 3.4 podemos ver una pequeña parte de un fichero de configuración de un ejercicio con JSON en Kibotics. En esta plataforma se utiliza JSON para crear los ejercicios con las características específicas que le correspondan, en este TFG se han usado tanto para la creación de los mundos como para fijar las posiciones de los confetis en el ejercicio del aspirador robótico que veremos más adelante.

## 3.2. HERRAMIENTAS

---

```
1  {
2    "scene-parent-id": "myIFrame",
3    "scene": {
4      "id": "scene",
5      "gravity": -9.8,
6      "ground": "/static/websim/assets/textures/handkerchief.png",
7      "sky": "/static/websim/assets/textures/sky.png",
8      "background": "color: gray;",
9      "inspector": "url: https://aframe.io/releases/0.4.0/aframe-inspector.min.js",
10     "embedded": true,
11     "physics": "debug: true; friction: 0.0002",
12     "renderer": "colorManagement: true"
13   },
14   "robots_config": [
15     {
16       "controller": "user1",
17       "id": "a-pibot"
18     }
19   ],
20   "evaluator_config": {
21     "world_limits": {
22       "x": [-40, 40],
23       "z": [-44, 24]
24     },
25     "idle_time": 30
26   },
27 }
```

Figura 3.4: Parte de un fichero config.json de un ejercicio en Kibotics

## 3.2. Herramientas

### 3.2.1. TensorFlowJS, Web Audio API y Teachable Machine

Para procesar el audio en JavaScript se han tenido que analizar diferentes herramientas o APIs<sup>2</sup> de procesamiento de audio en la web. De esta forma hemos podido elegir la herramienta que mejor se adaptaba a nuestro problema. Entre ellas TensorFlowJS, Web Audio API y Teachable Machine (Figura 3.5) son las herramientas que se han estudiado para llevar a cabo el teleoperador acústico.

TensorFlowJS es una biblioteca de código abierto de JavaScript para el entrenamiento y la implementación de modelos de aprendizaje automático en navegadores y en Node.js [49].

La API de Audio Web provee un sistema poderoso y versatil para controlar audio en la Web, permitiendo a los desarrolladores escoger fuentes de audio, agregar efectos al audio, crear visualizaciones y aplicar efectos espaciales, entre otras cosas [50].

Teachable Machine es una herramienta basada en la Web que hace posible crear modelos de aprendizaje automático de manera rápida, sencilla y accesible para todos [51].

---

<sup>2</sup>Interfaz de programación de aplicaciones

### 3.2. HERRAMIENTAS

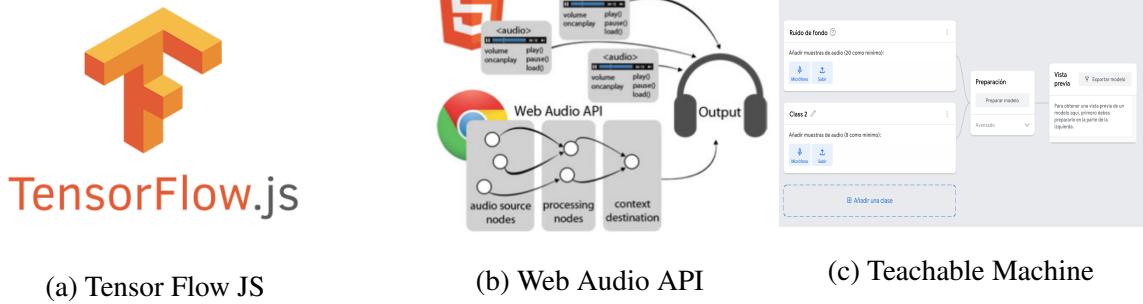


Figura 3.5: Herramientas de reconocimiento de audio

En el Capítulo 6 se explicará en profundidad cómo se han usado estas tecnologías y por qué después de probar con cada una de estas tres herramientas, hemos elegido Teachable Machine para el desarrollo final del teleoperador acústico.

#### 3.2.2. A-Frame

A-Frame es un marco web para crear experiencias de realidad virtual (VR). A-Frame utiliza HTML declarativo, lo que facilita bastante la creación de escenas en 3D y es accesible para todos, desde desarrolladores web, artistas y diseñadores, a educadores y estudiantes. Su estructura entidad-componente proporciona una infinidad de posibilidades y ofrece compatibilidad con *three.js*, una biblioteca escrita en JavaScript para crear y mostrar gráficos animados 3D en un navegador Web.

Sin instalar nada, A-Frame permite manejar modelos 3D para crear entornos de realidad virtual solamente usando las etiquetas *<script>* y *<a-scene>*. Es compatible con aplicaciones de realidad virtual como GearVR y Windows Mixed Reality entre otras. Además funciona perfectamente en ordenadores y teléfonos inteligentes [52].

Para crear escenas en una página web solo es necesario importar la librería de A-Frame de la siguiente forma:

```
1 <script src="https://aframe.io/releases/1.2.0/aframe.min.js"></script>
```

y poner las etiquetas correspondientes *a-scene* y los objetos que queramos que aparezcan en la escena con sus atributos. A continuación se muestra un ejemplo de código correspondiente con la visualización de la escena en el navegador en la Figura 3.6.

### 3.2. HERRAMIENTAS

---

```
1  <html>
2    <head>
3      <script src="https://aframe.io/releases/1.2.0/aframe.min.js"></script>
4    </head>
5    <body>
6      <a-scene>
7        <a-box position="-1 0.5 -3" rotation="0 45 0" color="#4CC3D9"></a-box>
8        <a-sphere position="0 1.25 -5" radius="1.25" color="#EF2D5E"></a-sphere>
9        <a-cylinder position="1 0.75 -3" radius="0.5" height="1.5" color="#FFC65D"></a-
10       cylinder>
11      <a-plane position="0 0 -4" rotation="-90 0 0" width="4" height="4" color="#7BC8A4">
12        </a-plane>
13      <a-sky color="#ECECEC"></a-sky>
14    </a-scene>
15  </body>
16 </html>
```

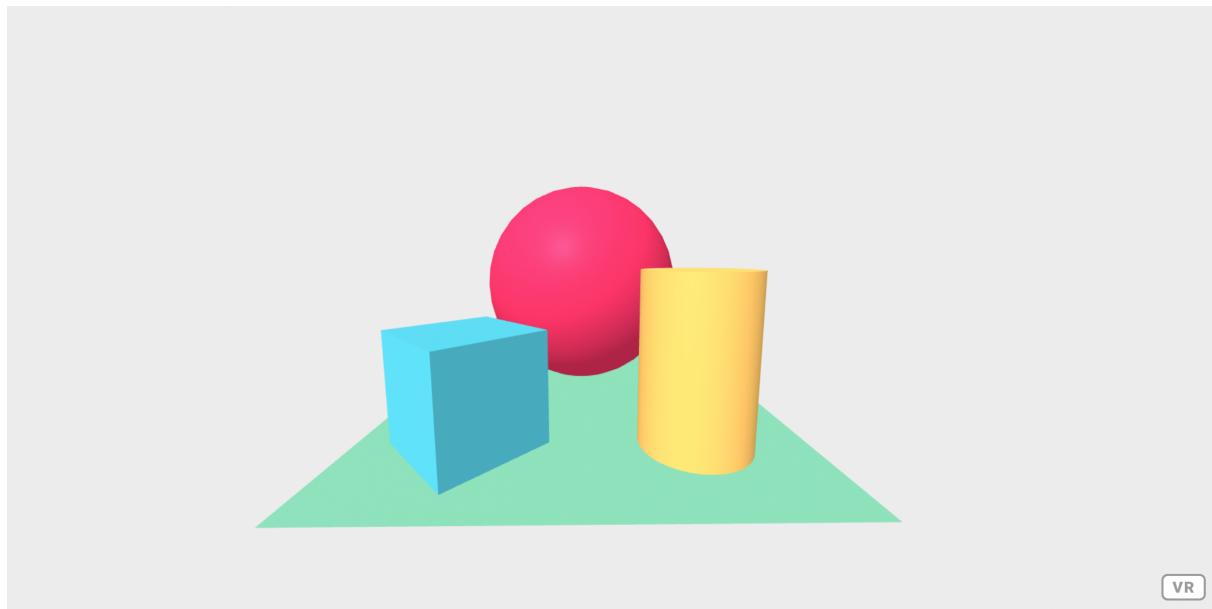


Figura 3.6: Ejemplo escena en A-Frame

A-Frame proporciona un inspector visual 3D incorporado (Figura 3.7). Presionando **ctrl + alt + i**, podemos inspeccionar la escena, esto es muy útil cuando necesitas la posición concreta de un objeto.

### 3.2. HERRAMIENTAS

---

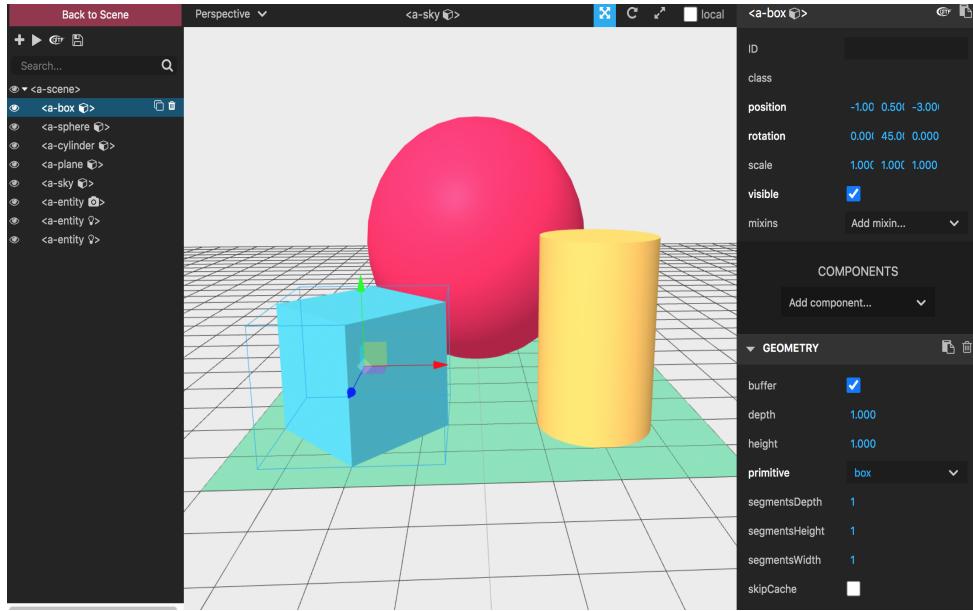


Figura 3.7: Inspector en A-Frame

En nuestro caso *a-entity* serán los modelos 3D de los robots que se crearán con Blender y se usarán componentes como *a-box*, *a-plane*, *a-cylinder* y *a-sphere* para crear componentes adicionales a la escena dándoles sus respectivas texturas. En este trabajo se ha usado la versión 1.1.0 de A-Frame.

#### 3.2.3. Blender

Blender es un software de creación de modelos 3D gratuito, de código abierto y multiplataforma. Este programa se utiliza para el modelado, montaje, animación, simulación y renderizado 3D, así como para la composición y seguimiento de movimiento, edición de vídeo y animación 2D [53]. En este trabajo se ha usado la versión 2.93.0 de Blender.

Para exportar los modelos de Blender se ha usado el formato glTF (*GL Transmission Format*). Es un formato de archivo para escenas y modelos 3D basado en JSON. De esta forma se pueden integrar los modelos en A-Frame de forma sencilla y rápida.

Este programa se ha utilizado en este proyecto para crear los nuevos robots, escenarios e introducir animaciones a la plataforma. En la Figura 3.8 se muestra un ejemplo de un cubo animado con Blender.

### 3.2. HERRAMIENTAS

---

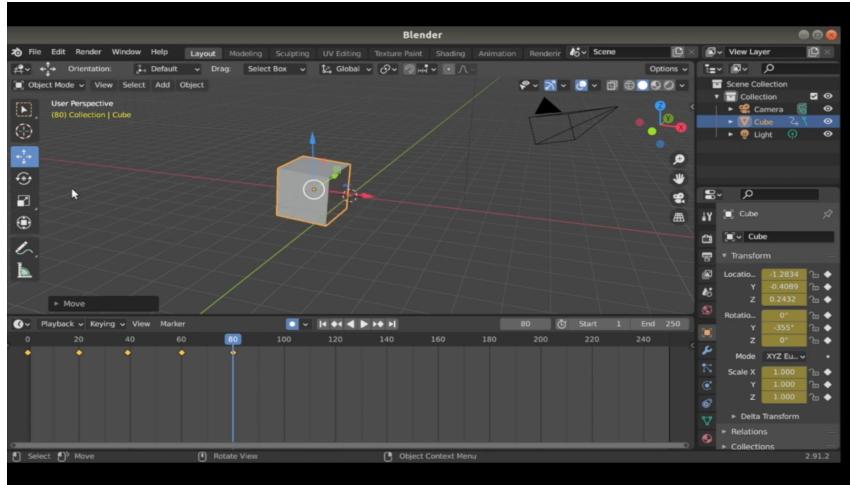


Figura 3.8: Blender

#### 3.2.4. Plataforma Kibotics

Kibotics es una plataforma web para docencia en robótica y programación. Esta plataforma se basa en tecnologías web como Django, para la parte servidor. Utiliza un simulador llamado Websim que se apoya en A-Frame para representar los escenarios de los ejercicios en el navegador del usuario.

Esta plataforma en línea ofrece contenidos educativos para facilitar el aprendizaje en programación a alumnos de primaria, secundaria y bachillerato. Ofrece cursos en los lenguajes Scratch y Python. Los ejercicios están disponibles tanto para robots físicos como simulados. Destacan Mbot, Dron Tello y Pibot entre otros.

La simulación de robots reales permite depurar el software al máximo antes de ejecutarlo en un robot físico y así reducir los costes, ya que no necesitas tener un robot para cada alumno y evitar posibles daños de los robots y accidentes.

Para usar esta plataforma no es necesario instalar nada, solo tener acceso a Internet. Al ser una aplicación web tenemos la ventaja de que es multiplataforma y podemos usarla en distintos dispositivos.

Kibotics tiene la filosofía *Learn by doing*, aprender haciendo. Las lecciones de teoría junto ejercicios prácticos hacen que los contenidos educativos se vayan adaptando según la complejidad de cada ejercicio. En la Figura 3.9 podemos ver algunos de los ejercicios que ofrecen.

### 3.2. HERRAMIENTAS

---

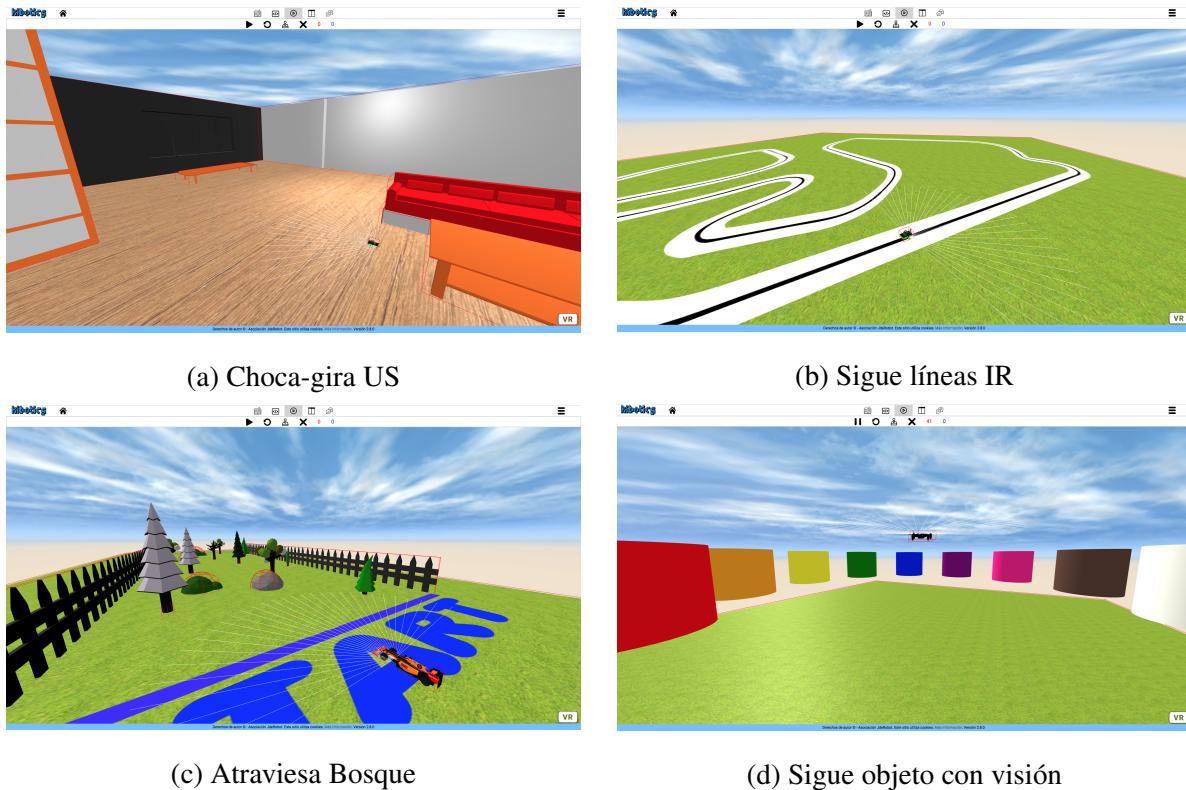


Figura 3.9: Ejercicios Kibotics

Como hemos mencionado anteriormente, la aplicación web Kibotics utiliza la tecnología Django y su estructura de plantillas para su servidor. En esta plataforma todos los ejercicios usan la misma plantilla. De esa forma, las páginas se crean dinámicamente con variables de Python y cada ejercicio tiene una página de web diferente con la teoría, el escenario y robot correspondiente. En estas plantillas o *templates* se incrusta el simulador Websim.

Los robots en Kibotics tienen dos partes: cuerpo y cerebro. El cuerpo se crea con Websim que analiza un fichero JSON (fichero de configuración) y lo materializa gracias a A-Frame. El cerebro lo programa el usuario en su editor en Python o Scratch, donde están implementados los sensores y actuadores gracias a HALRobotAPI<sup>3</sup>.

HALRobotAPI es una capa de abstracción hardware en la interfaz de programación del robot, ésta proporciona funciones sencillas para manejar los sensores y actuadores del robot. Al ejecutar el ejercicio, dando al botón play que aparece en la página web, el código del cerebro se envía al servidor por Websockets, lo traduce a JavaScript y renderiza el mundo. En la Figura 3.10 podemos ver cómo es la arquitectura de esta aplicación desde una visión general.

---

<sup>3</sup>HAL: Hardware Abstraction Layer

### 3.2. HERRAMIENTAS

---

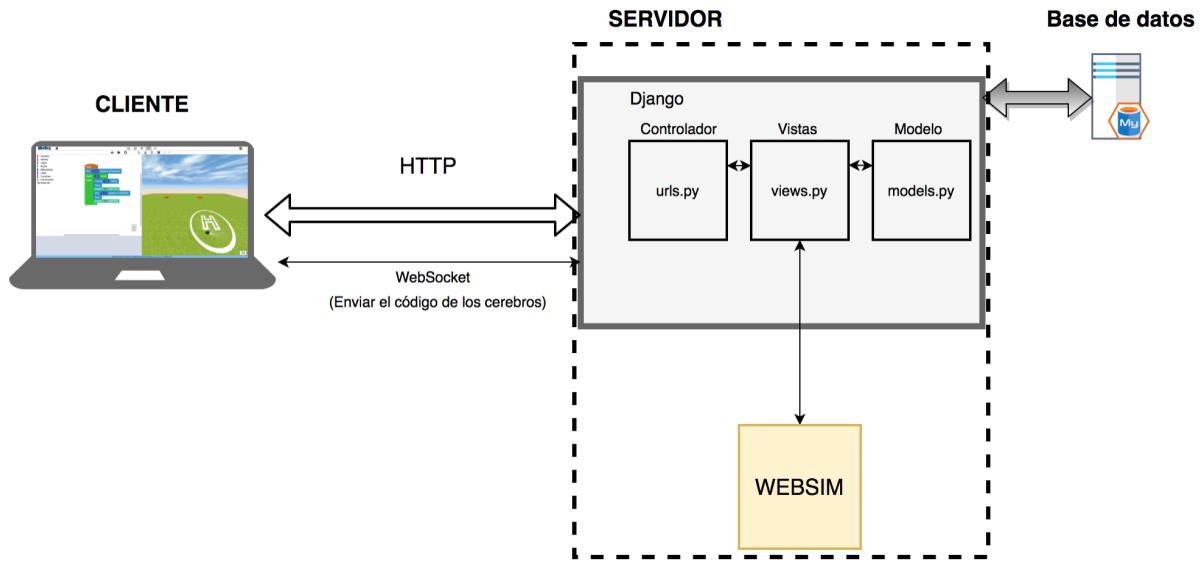


Figura 3.10: Arquitectura de la aplicación web Kibotics

Kibotics en los últimos años ha ofrecido cursos para la escuela de pensamiento computacional INTEF, el ayuntamiento de Fuenlabrada, la empresa Logix5, Universidad Rey Juan Carlos, IES Martínez Uribarri (Salamanca) y la Comunidad de Madrid [55].

#### 3.2.5. Navegadores web utilizados

En este Trabajo Fin de Grado se han usado los navegadores Chrome y Firefox por ser los navegadores más extendidos y que proporcionan gran rendimiento, estabilidad y seguridad. Se han utilizado para la visualización de la página web de Kibotics y la ejecución y pruebas de los distintos ejercicios.

# **Capítulo 4**

## **Ejercicio aspiradora robótica atrapa confeti**

En los siguientes capítulos se explica el proceso de desarrollo de los tres ejercicios realizados en este trabajo fin de grado. Este capítulo está dedicado al primero de ellos, que consiste en una aspiradora robótica que atrapa piezas de confeti que están dispersas por el suelo de una habitación. En primer lugar se expone el enunciado y los objetivos. En segundo lugar cómo se ha implementado y creado los modelos de robot aspiradora y escenario para la plataforma Kibotics. Finalmente, se muestran las soluciones de referencia creadas en Python y Scratch que satisfacen el objetivo del ejercicio.

Para hacer el ejercicio descrito a continuación, se utilizaron las herramientas Blender, el simulador Websim de Kibotics que contiene la tecnología A-Frame y los lenguajes JavaScript, HTML5 y JSON.

### **4.1. Enunciado**

La finalidad de este ejercicio era programar una aspiradora robótica que se encuentre en una habitación y que ésta fuera capaz de recoger piezas de confeti cuando pase por encima. El evaluador automático que se ha realizado sirve para contar el confeti recogido por los usuarios en un tiempo determinado.

El alumno deberá programar en Scratch o en Python un algoritmo de navegación para que consiga atrapar el mayor número de trozos de confeti en 5 minutos. El objetivo del usuario es

## 4.1. ENUNCIADO

---

que utilice los sensores y actuadores del robot para que cuando detecte un objeto próximo, por ejemplo una pared o una mesa, se mueva en ángulos aleatorios.

Para la visualización de este ejercicio y que los usuarios de Kibotics puedan acceder a él, se ha creado una página web. Gracias a esta página web, los usuarios tienen acceso a la teoría del ejercicio y disponen de un editor y el simulador Websim necesarios para hacer el ejercicio.

La teoría del ejercicio se creó en HTML5 y se modificaron las plantillas que utiliza el servidor Django de Kibotics. En las siguientes figuras podemos ver la página de teoría donde se explica el objetivo del ejercicio (Figura 4.1) y los requisitos (Figura 4.2). Para explicar al alumno los diferentes algoritmos de cobertura que pueden utilizar los aspiradores robóticos, se ha añadido un poco de teoría sobre esos algoritmos (Figura 4.3), unas pequeñas pistas (Figura 4.4) y un “¿Sabías que... ?” (Figura 4.5) hablando de las primeras aspiradoras y cómo funcionaban.

The screenshot shows a web-based exercise interface. At the top, there's a navigation bar with icons for home, search, and cart. Below it is a toolbar with icons for presentation, video, and other functions. On the right, there are two red buttons: 'Presentación' (selected) and 'Teoría'. The main content area has a title 'Aspirador Robótico 'Roomba' en Scratch' in bold. Below the title is a descriptive paragraph about programming a Roomba robot to collect confetti. It mentions using the HAL API and the goal of collecting the most confetti in 5 minutes. To the right of this text are two status indicators: 'Tiempo de estudio' (Study time) and 'Dificultad' (Difficulty), both showing '3 horas' (3 hours). Below these are two gear icons. Further down, there's another section titled 'Roomba atrapa confeti' with a sub-instruction about collecting confetti while avoiding obstacles. A small image of a room with colorful confetti on the floor is shown. At the bottom of the page is a blue footer bar with the text 'Derechos de autor © - Asociación JdeRobot. Este sitio utiliza cookies. Más Información. Versión 2.10.0'.

Figura 4.1: Página de teoría objetivos

## 4.1. ENUNCIADO

---

### 1 - Qué vas a aprender en esta unidad

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

### 2 - Requisitos de la práctica.

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

Figura 4.2: Página de teoría requisitos

### 3- Teoría

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

#### Algoritmos de cobertura

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

Clasificación Los algoritmos de cobertura se dividen en dos categorías:

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

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

Podemos usar un movimiento base: Movimiento en espiral: El robot sigue un patrón de círculo / cuadrado creciente. Movimiento Boustrophedon: El robot sigue un patrón en forma de S.

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

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

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

Figura 4.3: Teoría

### 4- Pistas

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

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

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

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

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

Figura 4.4: Pistas

### ¿Sabías que...?

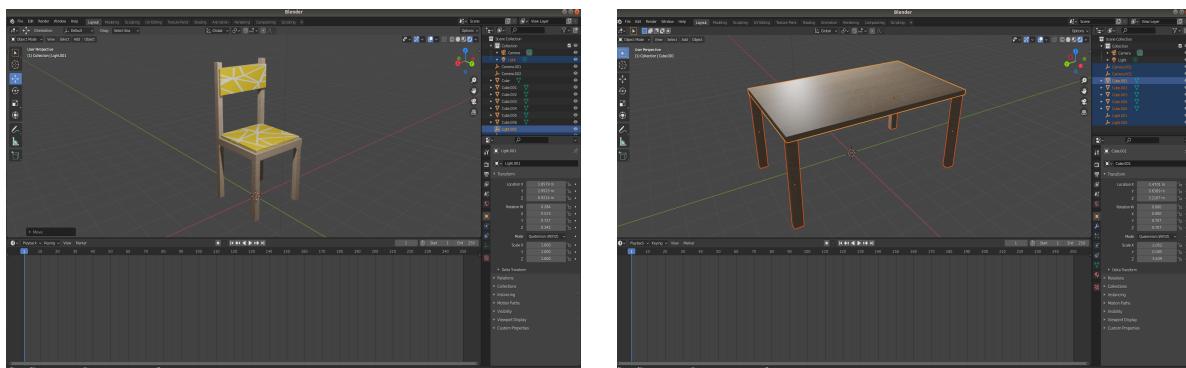
El primer aspirador mecánico fue creado por Huber Booth en 1901 y era tan grande que para trasladarlo de una casa a otra, tenía que ser arrastrado en un carro por caballos. Los tubos de succión se introducían en las viviendas por las puertas y las ventanas. Mucho más tarde aparecieron las aspiradoras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



Figura 4.5: ¿Sabías que ...?

## 4.2. Modelos del mundo

Para este ejercicio se crearon varios muebles con Blender (ver Figuras 4.6 y 4.7) para que el escenario pareciera una habitación realista (ver Figura 4.8).



(a) Modelo silla

(b) Modelo mesa

Figura 4.6: Modelos en Blender

## 4.2. MODELOS DEL MUNDO

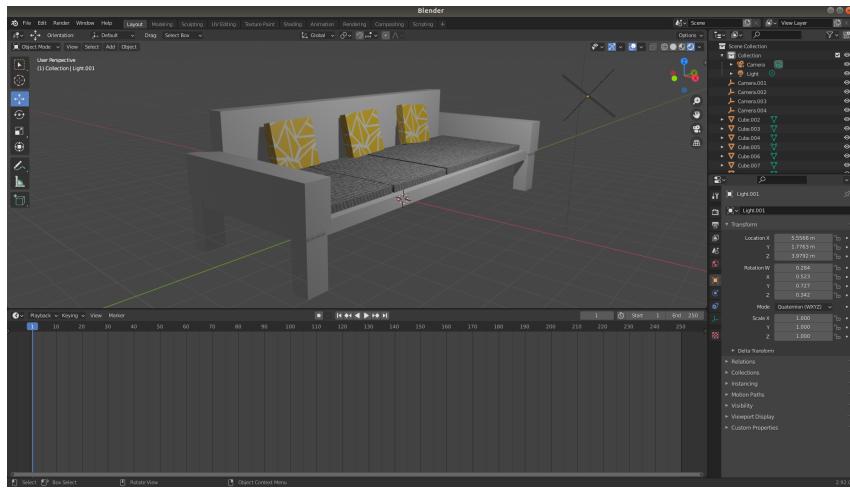


Figura 4.7: Modelo sofá en Blender



Figura 4.8: Habitación amueblada

Estos objetos se añadieron en el fichero de configuración JSON del ejercicio definiendo su identificador, modelo 3D gltf, la posición y la escala. El código que se muestra a continuación pertenece a la creación de la mesa pero la estructura de todos los elementos añadidos es la misma:

```
1 {      "tag": "a-entity",
2   "attr": {
3     "id": "model-table",
4     "gltf-model": "/static/websim/assets/models/mesa.gltf",
5     "position": { "x":40, "y":0, "z":0},
6     "scale": { "x":3, "y":3, "z":3}
7   }
8 },
```

## 4.2. MODELOS DEL MUNDO

---

También se añadieron a ese fichero de configuración, de forma muy similar, las paredes, una puerta y un cuadro. Se ajustó la posición, dimensiones y texturas correspondientes.

Para que el ejercicio tuviera mayor dificultad se introdujeron 2 pelotas con movimiento. Para conseguir este movimiento se consideraron dos posibilidades: animación desde Blender o animación desde A-Frame. En el vídeo<sup>1</sup> se puede ver cómo hacer una animación no lineal sencilla en Blender. Para usar animaciones de Blender en Websim es necesario que sean animaciones no lineales. Estudiando ambas opciones, se optó por la animación en A-Frame desde el fichero de configuración, que era el método ideal para Websim. De esta forma dejamos toda la creación de las pelotas, asignación de texturas, posición y animación definido en la configuración.

Para animar un objeto desde A-Frame añadimos el atributo “*animation*” y completamos los valores necesarios como desde dónde a dónde quieras que se mueva el objeto o el tiempo que quieras que tarde en hacer esa animación. Por ejemplo, en el siguiente código se puede ver cómo se define la animación en diagonal de la pelota de baloncesto. Las dos pelotas tienen los mismos atributos pero con valores diferentes.

```
1 {      "tag": "a-sphere",
2     "attr": {
3         "id": "basketball",
4         "position": { "x": -40, "y": 1.5, "z": -40 },
5         "rotation": { "x": 0, "y": 0, "z": 0 },
6         "radius": 1.5,
7         "src": "#basketball",
8         "static-body": {
9             "mass": 2
10            },
11         "class": "collidable",
12         "animation": "property: position; from: -40 1.5 -40 ;to: 40 1.5 40; dir:
13             alternate; dur: 10000; loop: true"
14     }
},
```

En el escenario de este ejercicio se crearon un conjunto de 100 trozos de papel, confeti, repartidos por el suelo. Estos modelos se hicieron con la etiqueta *a-cylinder* de A-Frame.

El confeti no se crea desde el fichero de configuración, dado que se tendrían que crear uno a uno todos los confetis y extendería demasiado el fichero. Lo que se ha hecho es que, una vez esta cargado el mundo en el navegador del usuario, usando JavaScript, creamos dinámicamente todos los confetis.

---

<sup>1</sup><https://www.youtube.com/watch?v=1JPb3Mw8638>

## 4.2. MODELOS DEL MUNDO

---

Primero se generó un programa para fijar aleatoriamente las posiciones de los confetis en el mundo, así los confetis quedan esparcidos por la habitación. Estas posiciones x, y, z de los confetis se guardaron en otro fichero JSON llamado *data.json*, de esta forma, todos los alumnos cuentan con el mismo escenario. El escenario está formado por 100 confetis de colores y el color de cada uno se elige aleatoriamente cuando se crea desde JavaScript .

Parte del fichero *data.json*:

```
1 data = '[{"x": -39, "y": 0, "z": -14}, {"x": 7, "y": 0, "z": -11}, {"x": -6, "y": 0, "z": -9}, {"x": -33, "y": 0, "z": 28}, {"x": -11, "y": 0, "z": 41}, {"x": -11, "y": 0, "z": -13}, {"x": 19, "y": 0, "z": 23}, {"x": 19, "y": 0, "z": -30}, {"x": -4, "y": 0, "z": -27}, {"x": 25, "y": 0, "z": 20}, ... ]'
```

Para leer el fichero *data.json* se utiliza éste código en la plantilla HTML del ejercicio.

```
1 <script type="text/javascript" src="{% static '/data.json' %}"></script>
```

En el siguiente código podemos ver cómo se crea el confeti con la función *document.createElement('a cylinder')*. Una vez creado se le asignan los atributos con la función *setAttribute*. El identificador es confeti más un número n, n es un número de 0-99 que se le asigna para crear 100 confetis diferentes (ejemplo de *id* del confeti número 50 *id= "confeti50"*). Además se le asigna los atributos posición, con la posición correspondiente que se lee del *data.json*, el color que es aleatorio gracias a la función *getRandomColor()* y como es un cilindro, se define la altura y radio del confeti. Los confetis no cuentan con malla de colisión para que el aspirador pueda pasar por encima de ellos y absorberlos de una forma más natural. Estas son las funciones necesarias para la creación de las piezas de confeti:

```
1 function getRandomColor() {
2     var letters = '0123456789ABCDEF';
3     var color = '#';
4     for (var i = 0; i < 6; i++) {
5         color += letters[Math.floor(Math.random() * 16)];
6     }
7     return color;
8 }
9
10 document.addEventListener('robot-loaded', (evt)=>{
11     localRobot = evt.detail;
12     var sceneEl = document.querySelector('a-scene');
13     // CREATE CONFETI
14     var n = 0;
15     var n_confetis = 99;
16     score = 0;
17     var array = JSON.parse(data);
```

### 4.3. MODELO ASPIRADORA

```
18    for ( n = 0; n <=n_confetis ; n++) {  
19        var c = document.createElement('a-cylinder');  
20        var num_conf="confeti"+ String(n)  
21        c.setAttribute('id', num_conf);  
22        pos = {x:array[n].x, y:0,z:array[n].z}  
23        c.setAttribute('position',pos);  
24  
25        var color = getRandomColor();  
26        c.setAttribute('color', color);  
27        c.setAttribute('height', "0.25");  
28        c.setAttribute('radius', 1);  
29        sceneEl.appendChild(c);  
30    }
```

## 4.3. Modelo aspiradora

El robot propuesto en este ejercicio es una aspiradora robótica. Se crearon varios prototipos en Blender hasta que se llegó al modelo final que podemos ver en la Figura 4.9. Su diseño está inspirado en los aspiradores robóticos más conocidos como Roomba o Dyson. El logo que aporta color al aspirador pertenece a la asociación JdeRobot<sup>2</sup>, pilar fundamental en el desarrollo de la plataforma Kibotics.

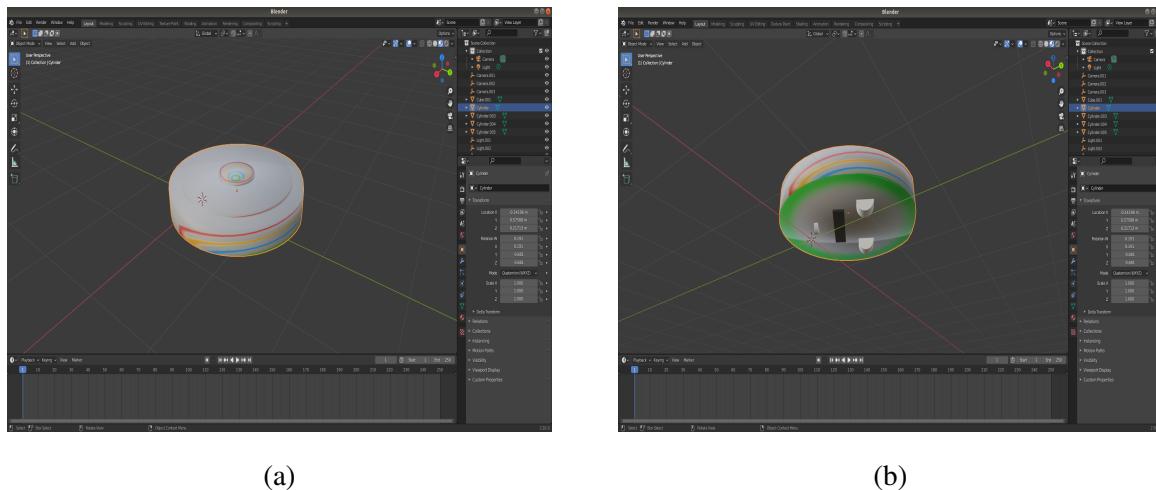


Figura 4.9: Modelo de la aspiradora robótica realizado en Blender.

Los robots en A-Frame están definidos por una apariencia 3D (geometría más texturas) y una malla de colisión.

<sup>2</sup><https://jderobot.github.io/>

### 4.3. MODELO ASPIRADORA

---

El simulador Websim analiza ficheros JSON para formar mundos con la tecnología A-Frame. De esta forma es sencillo crear los objetos con pares “atributo”: “valor”. A continuación podemos ver cómo se define el identificador del nuevo robot, se importa el modelo 3D (en formato gltf) que hemos creado en Blender de la aspiradora y se le asignan otros atributos como la posición, escala y rotación del robot. El *dynamic-body* de A-Frame nos facilita el movimiento del robot y el atributo *collide* asigna una malla de colisión cilíndrica para que la aspiradora pueda chocarse con los demás elementos del mundo usando físicas de A-Frame y todo sea mucho más realista.

```
1 {      "tag": "a-robot",
2       "attr": {
3           "id": "a-pibot",
4           "gltf-model": "/static/websim/assets/models/roombajderbotgrey.gltf",
5           "scale": { "x":2, "y":2, "z":2},
6           "position": { "x":0, "y":4, "z":30},
7           "rotation": { "x":0, "y":90, "z":0},
8           "dynamic-body":{ "mass": 10, "shape":"none",
9             "shape__handle":{ "shape": "cylinder",
10               "height": 0.36,
11               "radiusTop": 1.2,
12               "radiusBottom": 1.2,
13               "offset": "-0.1 0.2 -0.55"},
14             "collide":{}}
15     },
16 }
```

El robot que acabamos de describir es un objeto de A-Frame inanimado, la capacidad de movimiento autónomo se la proporciona el código del alumno y unos drivers que ofrece Kibotics para materializar desplazamientos en la posición del robot en A-Frame y giros en su orientación. Estos drivers ofrecen funciones al código del alumno para controlar el movimiento del robot tanto en velocidad como en posición, y tanto en traslación como en rotación. Gracias a este software el robot es capaz de moverse por la escena según ordene el código del usuario y está sujeto a las físicas realistas de A-Frame.

El robot también debe ser capaz de aspirar trozos de papel de la escena, por ello se ha tenido que implementar un actuador ficticio, el actuador de absorción. Este actuador debe hacer que cuando el robot pase encima de una pieza de confeti, la pieza en concreto desaparezca de la escena.

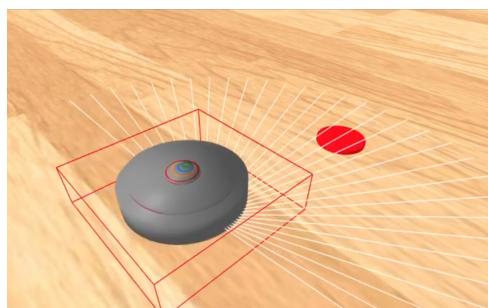
La absorción se implementó en JavaScript utilizando la función *setInterval()*. Esta función ejecuta las funciones que estén definidas dentro, cada un cierto periodo de tiempo indefinida-

### 4.3. MODELO ASPIRADORA

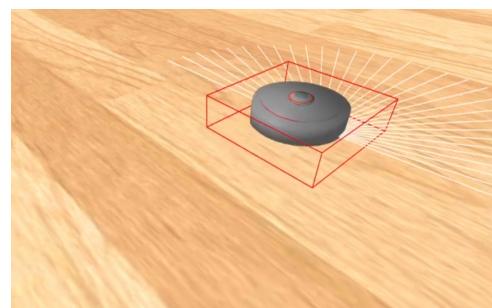
mente. Se ha establecido que cada 25ms este programa comprueba la distancia entre la aspiradora robótica y cada uno de los confetis. La posición del robot es su centro de masas, por eso se utilizó la distancia euclídea para calcular la distancia entre el centro del robot y el centro del confeti n, si esta distancia  $d$  es menor o igual a 2, el confeti n cambia su atributo ‘visible’ a false.

```
1 roomba=sceneEl.querySelector('#a-pibot')
2 setInterval(function(){
3     for ( n = 0; n <=n_confetis ; n++) {
4         d = Math.sqrt(Math.pow((array[n].z-roomba.getAttribute('position').z), 2)+Math.pow(
5             array[n].x-roomba.getAttribute('position').x), 2));
6         if ( d <= 2 ){
7             num_conf="#confeti"+ String(n)
8             confeti=sceneEl.querySelector(num_conf)
9             confeti.setAttribute('visible', false);
10        }
11    }
12 }, 25);
13 }) ;
```

Las pruebas de absorción se estudiaron tanto en función de la posición como por colisión. En este vídeo<sup>3</sup> se puede ver el primer prototipo por colisión y en este otro por posición<sup>4</sup>, al ser un efecto más parecido a lo que ocurre en la realidad se eligió implementar este ejercicio con la absorción basada en posición (ver Figura 4.10).



(a) Modelos aspiradora y confeti en A-Frame



(b) Confeti invisible

Figura 4.10: Absorción por posición

<sup>3</sup>[https://www.youtube.com/watch?v=xkC\\_qHXKUDs](https://www.youtube.com/watch?v=xkC_qHXKUDs)

<sup>4</sup><https://www.youtube.com/watch?v=If2XMcr1ci4>

## 4.4. Evaluador automático

Para contar el número de confetis que es capaz de recoger un usuario en un periodo de tiempo se creó un evaluador automático. Este evaluador automático se introdujo dentro de la absorción. Como en esa parte se calcula la distancia euclídea, ahora cada vez que  $d$  es menor o igual a 2 se suma un punto e indica que el confeti n ha sido “absorbido” por el aspirador. La puntuación máxima es de 100, dado que depende del número de confetis. También se establece una cuenta atrás de 5:00 minutos. Una vez acabado el tiempo (Tiempo 00:00), aunque la aspiradora pase por encima de otros confetis visibles, éstos no serán absorbidos ni se aumentará el contador del evaluador.

En el siguiente código podemos ver cómo se hace comprobación de las posiciones del confeti para la absorción con la distancia euclídea  $d$  y la puntuación del evaluador con la variable *score* que se incrementa cuando el confeti n se hace invisible y no se ha acabado el tiempo. En la Figura 4.11 se muestra cómo se visualiza el evaluador en la página web del ejercicio.

```
1 roomba=sceneEl.querySelector('#a-pibot')
2 setInterval(function(){
3     for ( n = 0; n <=n_confetis ; n++) {
4         d = Math.sqrt(Math.pow((array[n].z-roomba.getAttribute('position').z), 2)+Math.pow(
5             array[n].x-roomba.getAttribute('position').x), 2));
6         if ( d <= 2 ){
7             num_conf="#confeti"+ String(n)
8             confeti=sceneEl.querySelector(num_conf)
9             if (confeti.getAttribute('visible') == true) {
10                 var counter= document.getElementById('time').innerHTML;
11                 // Tiempo: 00:00
12                 if((counter[8] =='0' ) && (counter[9]=='0' ) && (counter[11]=='0' ) && (counter[12]=='0')) {
13                     score = score;
14                 }else{
15                     score+=1;
16                     document.getElementById('confeti_recogido').innerHTML = "Confetti recogido: "+ score;
17                 }
18                 confeti.setAttribute('visible', false);
19             }
20         }
21     }, 25);
22 }) ;
```

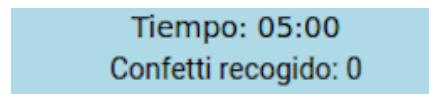


Figura 4.11: Evaluador automático para el ejercicio aspiradora robótica

## 4.5. Solución de referencia

Hay muchas formas diferentes de resolver este ejercicio, en este apartado se proponen dos soluciones en los dos lenguajes que soporta Kibotics. En la Figura 4.12 se muestra una solución realizada en Scratch y en la Figura 4.13 una para Python.



Figura 4.12: Solución en Scratch

```

1 import HAL
2 import time
3 # Pon aquí tu código
4 # ej:
5
6 i = 0;
7 while True:
8     HAL.avanzar(1);
9     time.sleep(i);
10    HAL.girar_izquierda_hasta(10+i);
11    i+=2;
12    if (i > 100 ):
13        i = 0;
14    if HAL.leer_us() < 7:
15        HAL.girar_izquierda_hasta(i)
16

```

Figura 4.13: Solución en Python

En estas soluciones se ha hecho un algoritmo de cobertura usando el sensor de ultrasonidos para detectar la distancia entre el robot y los objetos de la habitación. Cuando la distancia que

#### *4.5. SOLUCIÓN DE REFERENCIA*

---

devuelve el sensor es muy pequeña, el robot para, retrocede y gira un número aleatorio de grados para luego continuar avanzando y aspirando por la habitación.

Este ejercicio ya está disponible en la plataforma Kibotics. También se han realizado dos vídeos promocionales para presentar el nuevo ejercicio de la plataforma con los códigos que se han mostrado anteriormente en las Figuras 4.12 y 4.13. Estos vídeos<sup>5</sup> <sup>6</sup> sirven de ejemplo y motivación para los alumnos que vayan a realizarlo.

---

<sup>5</sup><https://www.youtube.com/watch?v=5Q0TmwunYWY>

<sup>6</sup><https://www.youtube.com/watch?v=Twc9wsPFjaY>

# Capítulo 5

## Ejercicio juego del pañuelo

En este capítulo se expone cómo ha sido el desarrollo del juego del pañuelo. Primero hablaremos del enunciado, seguidamente hablaremos de los modelos desarrollados. Para terminar, se ofrecen dos soluciones de referencia en Python y Scratch.

Para que nuestro ejercicio sea acorde con el juego del pañuelo que conocemos todos y con las competiciones actuales, se ha creado un robot con pinzas, una lata, que representa el pañuelo y un circuito. Las principales tecnologías que se han utilizado en este ejercicio son Blender, JavaScript, A-Frame y Websim .

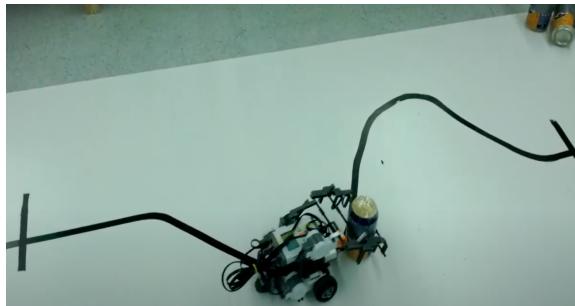
### 5.1. Enunciado

El propósito era crear un ejercicio con un nuevo robot que fuera capaz de coger objetos y moverlos por el escenario. El nuevo robot debía tener pinzas móviles para atrapar otros objetos.

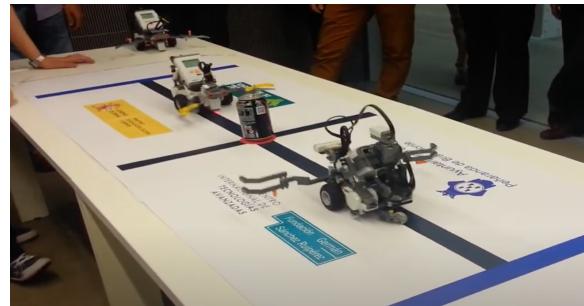
El juego del pañuelo es muy popular entre los más pequeños y también en el mundo de la robótica. En las competiciones de robots este juego incluye programar un sigue líneas. El robot debe recorrer un circuito, ser capaz de coger una lata que obstaculiza el camino y volver con ella a la posición de salida. En la Figura 5.1 (a) y 5.1 (b) podemos ver cómo es este juego en competiciones de robótica como Robocampeones.

## 5.1. ENUNCIADO

---



(a) Juego del pañuelo con un robot <sup>a</sup>



(b) Juego del pañuelo multirobot <sup>a</sup>

<sup>a</sup><https://www.youtube.com/watch?v=FsS2CSudl7c>

<sup>a</sup>[https://www.youtube.com/watch?v=6T\\_UaTaB6Q](https://www.youtube.com/watch?v=6T_UaTaB6Q)

Figura 5.1: Juego del Pañuelo en competiciones robóticas

En este trabajo fin de grado hemos hecho nuestra versión del juego del pañuelo basándonos en las competiciones existentes. De esta forma, los alumnos pueden adquirir experiencia y animarse en un futuro a presentarse a competiciones de alto nivel.

En este ejercicio el alumno deberá programar, en Python o en Scratch, un algoritmo que permita que el Mbot con pinzas avance siguiendo la línea negra del circuito hasta que se encuentre a poca distancia de la lata. Una vez se encuentre enfrente de la lata, el robot debe cerrar las pinzas y dar media vuelta para volver a la casilla de salida, llevando consigo la lata en todo momento. Gracias a un evaluador automático vamos a obtener la puntuación, ésta va a depender del porcentaje de circuito recorrido y si lleva o no la lata entre las pinzas.

Se creó la página web de teoría con HTML5 y se añadió a las plantillas de Django en Kibotics en su versión de Python y Scratch. La teoría se divide en 5 apartados, primero se hace una breve introducción del ejercicio y los objetivos que tiene que cumplir el alumno (Figura 5.2). En el primer punto se comenta qué se va a aprender en esta unidad (Figura 5.3). En el segundo punto se explica la teoría de los sensores que se van a utilizar, en este caso, son necesarios el sensor infrarrojos y el sensor ‘‘dame objeto en pinza’’ que se ha desarrollado en este trabajo (Figuras 5.3, 5.4, 5.5, 5.6, 5.7 y 5.8). En el tercer punto se nombran los actuadores de movimiento, junto con el abrir y cerrar de las pinzas necesarios para hacer el ejercicio y se indican las funciones y los bloques para poder utilizarlos en Python (Figura 5.9) y en Scratch (Figura 5.10). Finalmente, el punto 4 trata de un “¿Sabías que...?” que cuenta datos curiosos sobre este antiguo juego (Figura 5.11).

## 5.1. ENUNCIADO

The screenshot shows the hibotics platform interface. At the top, there are navigation icons for home, cart, and account. To the right are icons for presentation mode, full screen, code editor, video player, and exit. Below these are two red buttons: "Presentación" and "Teoría". The main title "Juego del Pañuelo Python" is displayed in large bold letters. Below the title is an image of a blue Mbot Pinza robot on a green grid floor, positioned on a red line that forms a circuit path. To the right of the image, there are two columns: "Tiempo de estudio" (Study time) with "3 horas" (3 hours) and "Dificultad" (Difficulty) with a gear icon.

El objetivo de este ejercicio es conseguir atrapar la lata siguiendo las líneas del circuito.

### Ejercicio del Pañuelo con Mbot Pinza

En este ejercicio deberás programar nuestro robot Mbot Pinza para que sea capaz de recorrer el circuito para coger la lata debemos usar los *sensores* para obtener información del entorno continuamente, y sus *actuadores* para ejecutar las acciones necesarias para poder ir moviéndose por el circuito, de acuerdo a la información obtenida por los sensores.

Concretamente, el programa debe hacer que el Mbot Pinza avance siguiendo la línea negra del circuito hasta que se encuentre a poca distancia de la lata, una vez te encuentres enfrente de la lata debes cerrar las pinzas y dar media vuelta para volver a la casilla de salida, llevaté contigo la lata en todo momento.

Figura 5.2: Página de teoría enunciado y requisitos

### 1 - Qué vas a aprender en esta unidad

En esta unidad vas a profundizar tus conocimientos sobre los sensores: infrarrojos y el nuevo sensor dame objeto en pinza.

### 2- Los sensores infrarrojos y dame objeto en pinza

#### Sensor Infrarrojos

Un sensor de infrarrojos es un detector de un tipo de luz que nuestro ojo no es capaz de ver que es la luz infrarroja. Como ya sabréis la luz del sol, en su espectro visible (es decir, lo que el ojo humano es capaz de ver), está compuesta de diferentes colores, los que vemos en el arcoíris. Cada color se caracteriza por una frecuencia diferente de la radiación solar.

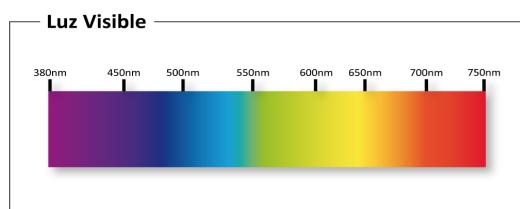


Figura 5.3: Qué vas a aprender y teoría infrarrojos

## 5.1. ENUNCIADO

---

Sin embargo, el sol no sólo emite radiación en esas frecuencias que nosotros vemos. Un intervalo de esas frecuencias es el infrarrojo, que se llama así porque si la viésemos estaría antes del rojo en el arcoíris.

Un sensor de infrarrojos está formado por dos partes:

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

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

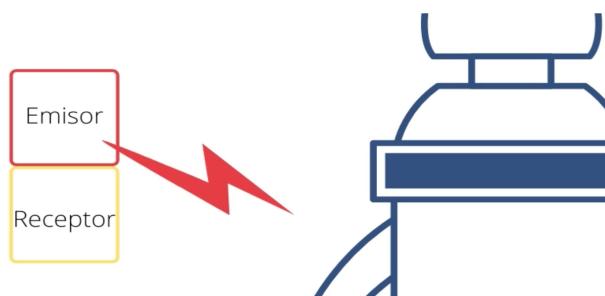


Figura 5.4: Teoría infrarrojos

### Detectando una línea negra con los sensores infrarrojos

Vamos a usar los 2 sensores de Infrarrojos (abreviado IR) que tiene el Mbot en su parte frontal apuntando al suelo, uno en la parte izquierda y otro en la parte derecha. Estos sensores son capaces de detectar si el suelo es oscuro (negro) o claro (blanco) por medio de la emisión/detección de rayos infrarrojos. Para ello, en el programa debes llamar a siguiente función:

FunciónDescripción

`leer_ir()` Lee los dos sensores de IR y nos devuelve un valor entre 0 y 3

El valor entre 0 y 3 que devuelve la función depende de las lecturas de los dos sensores de IR, de acuerdo a la siguiente tabla:

lectura IR izquierdolectura IR derechovalor de leer\_ir()

|        |        |   |
|--------|--------|---|
| negro  | negro  | 0 |
| negro  | blanco | 1 |
| blanco | negro  | 2 |
| blanco | blanco | 3 |

Estos cuatro valores que nos devuelve la función `get_ir()` se corresponden con las siguientes situaciones del Mbot con respecto a la línea negra:



Figura 5.5: Teoría sensor infrarrojos en Python

## **5.1. ENUNCIADO**

| Obtener el valor del infrarrojo |  |
|---------------------------------|--|
| Valor                           | Significado  |
| 0                               | Los dos sensores detectan negro. El robot está sobre la línea negra. |
| 1                               | La línea negra se encuentra a la izquierda del robot                 |
| 2                               | La línea negra se encuentra a la derecha del robot                   |
| 3                               | El robot está completamente fuera de la línea negra                  |

Figura 5.6: Teroría sensor infrarrojos en Scratch

## Sensor Dame objeto en pinza

Este sensor funciona gracias al láser del robot, el sensor `dame_objeto_en_pinza()` devuelve el valor 0 cuando no hay ningún objeto entre las pinzas y devuelve el valor 1 si detecta un objeto entre las pinzas.

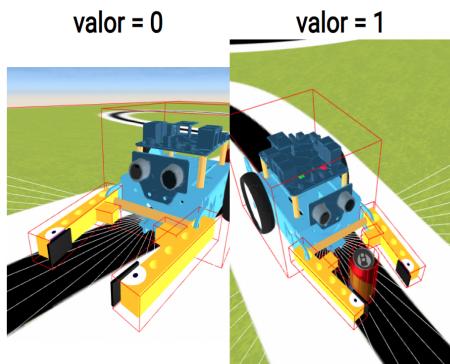


Figura 5.7: Sensor dame objeto en pinza en Python

## 5.1. ENUNCIADO

---

### Sensor Dame objeto en pinza

Este sensor funciona gracias al láser del robot, el sensor devuelve el valor 0 cuando no hay ningún objeto entre las pinzas y devuelve el valor 1 si detecta un objeto entre las pinzas.

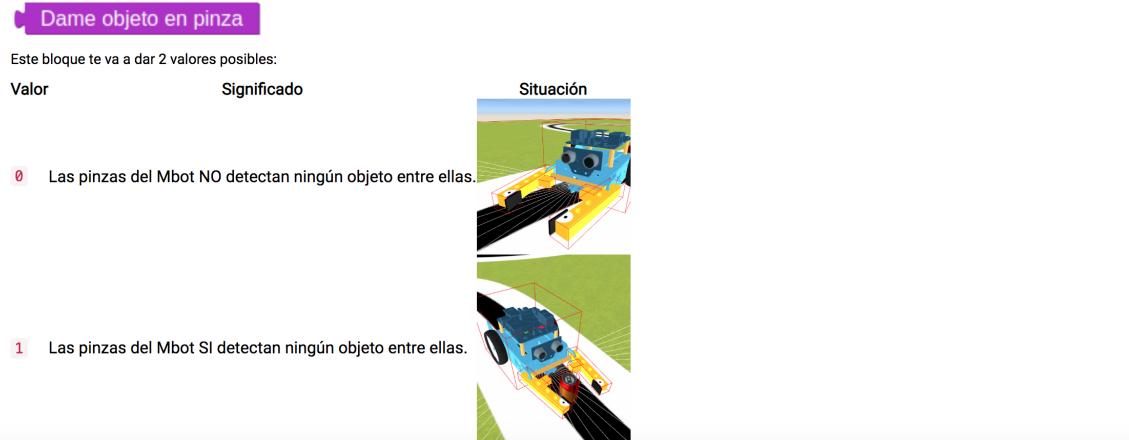


Figura 5.8: Sensor dame objeto en pinza en Scratch

## 3 - Actuadores a utilizar

Vamos a usar los motores que incorpora el Mbot Pinza (uno en cada rueda motriz) para hacer que se mueva según nuestras necesidades. Para ello, en el programa debes llamar a las siguientes funciones:

| Función                     | Descripción  |
|-----------------------------|--|
| avanzar( <i>v</i> )         | Hace que el Mbot avance en línea recta con una velocidad lineal igual a <i>v</i> expresada en metros/segundo                             |
| retroceder( <i>v</i> )      | Hace que el Mbot retroceda en línea recta con una velocidad lineal igual a <i>v</i> expresada en metros/segundo                          |
| girar_izquierda( <i>w</i> ) | Hace que el Mbot gire a la izquierda sobre su eje (sin avanzar) con una velocidad angular igual a <i>w</i> expresada en radianes/segundo |
| girar_derecha( <i>w</i> )   | Hace que el Mbot gire a la derecha sobre su eje (sin avanzar) con una velocidad angular igual a <i>w</i> expresada en radianes/segundo   |
| parar()                     | Hace que el Mbot se detenga (si estaba avanzando, retrocediendo o girando)   |
| abrir_pinza()               | Hace que la pinza del Mbot se abra.  |
| cerrar_pinza()              | Hace que la pinza del Mbot se cierre.  |

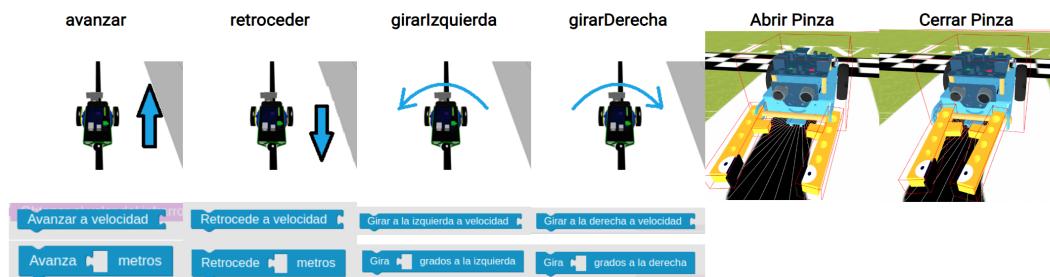
Combina los dos sensores y usa los actuadores para recoger la lata y completar el circuito ¡Inténtalo, es divertido!

Figura 5.9: Teoría actuadores en Python

### 3 - Actuadores del MBot Pinza a utilizar

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

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



Combina los dos sensores y usa los actuadores para recoger la lata y completar el circuito ¡Inténtalo, es divertido!

Figura 5.10: Teroría actuadores en Scratch

### 4 - ¿Sabías que...?

Este ejercicio está basado en el juego del pañuelo tradicional que jugábamos de pequeños.

Las normas del juego del pañuelo están diseñadas para premiar la rapidez y la agilidad. Ser el primero en reaccionar y saber colaborar con los compañeros es un juego educativo ¡muy divertido!.

El origen de este juego es incierto, algunas voces apuntan a que es un juego con más de 2000 años de antigüedad al que ya jugaban los niños en el Antiguo Egipto. Si bien no podía jugarse con un pañuelo, ya que este pequeño complemento no se usó de forma habitual hasta el siglo XV.

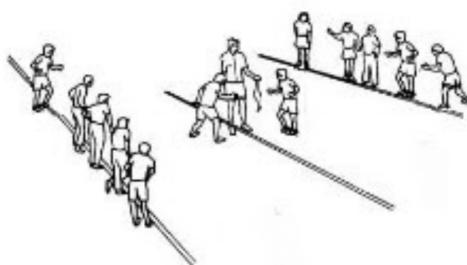
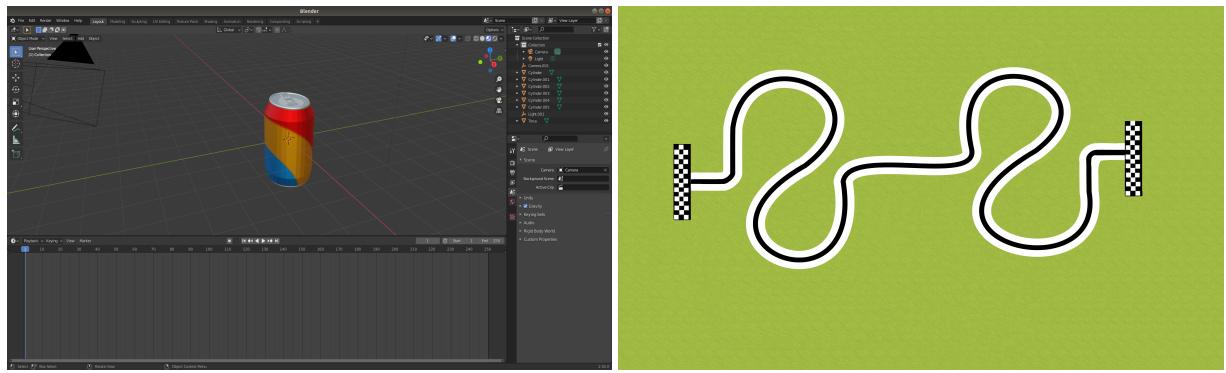


Figura 5.11: ¿Sabías que.. ?

## 5.2. Modelos del mundo

En este apartado se explica cómo se ha creado el modelo de la lata y del circuito de este ejercicio. El modelo 3D de la lata se creó con Blender (Figura 5.12 (a)), la referencia fue un bote de refresco tradicional y se añadió el logo de JdeRobot. Para hacer el circuito se utilizó GIMP, un programa de edición de imágenes gratuito y multiplataforma (versión 2.10), con él se diseñó la imagen .png del circuito (Figura 5.12 (b)).

## 5.2. MODELOS DEL MUNDO



(a) Modelo lata

(b) Circuito diseñado en GIMP del juego del pañuelo

Figura 5.12: Modelos en Blender

Los objetos se añadieron de la siguiente forma al fichero de configuración:

```
1 ...
2
3     "assets": [
4         {
5             "tag": "img",
6             "attr": {
7                 "id": "ground",
8                 "alt": "Texture for the scene ground",
9                 "src": "/static/websim/assets/textures/handkerchief.png"
10            }
11        },
12        {
13            "tag": "a-asset-item",
14            "attr": {
15                "id": "model-pibot"
16            }
17        },
18        {
19            "tag": "a-asset-item",
20            "attr": {
21                "id": "model-can"
22            }
23        }
24    ],
25    "objects": [
26        {
27            "tag": "a-entity",
28            "attr": {
29                "id": "object",
30                "gltf-model": "/static/websim/assets/models/can.gltf",
31                "position": { "x": -2.25, "y": 0.6, "z": -10.25 },
32                "rotation": { "x": 0, "y": 0, "z": 0 },
33                "scale": { "x": 0.3, "y": 0.3, "z": 0.3 },
34                "body": { "type": "dynamic", "mass": 20, "shape": "cylinder" },
35            }
36        }
37    ]
38 }
```

### 5.3. MODELO ROBOT CON PINZA

---

```
30     "class": "collidable"
31   }
32 },
33
34 { "tag": "a-plane",
35   "attr": {
36     "static-body": {
37       "mass": 100000
38     },
39     "position": { "x":0, "y":0, "z":-4 },
40     "rotation": { "x":-90, "y":0, "z":0 },
41     "width": "100",
42     "height": "100",
43     "src": "#ground"
44   }
45 },
46 ]...
```

A la hora de crear la escena en el fichero de configuración de este ejercicio, se tuvo que añadir el atributo `'renderer': 'colorManagement: true'` para que los colores en Websim fueran los mismos que se habían diseñado en Blender. Si no, los modelos 3D en Websim se verían más oscuros.

```
1 "scene": {
2   "id": "scene",
3   "gravity": -9.8,
4   "ground": "/static/websim/assets/textures/handkerchief.png",
5   "sky": "/static/websim/assets/textures/sky.png",
6   "background": "color: gray;",
7   "inspector": "url: https://aframe.io/releases/0.4.0/aframe-inspector.min.js",
8   "embedded": true,
9   "physics": "debug: false; friction:0.0002",
10  "renderer": "colorManagement: true" }, ...
```

## 5.3. Modelo robot con pinza

En este apartado se va a explicar, por un lado, el prototipo en A-Frame nativo hecho por Roberto Pérez, un colaborador de JdeRobot, y por otro, cómo se ha creado finalmente el nuevo robot con pinzas.

### 5.3.1. Antecedentes

El prototipo existente en la plataforma estaba realizado en A-Frame nativo, (HTML5, JavaScript y la librería de A-Frame) sin Websim. Este primer prototipo cogía el modelo del Mbot que ya se usaba en Kibotics para algunos ejercicios y lo representaba en la escena. El robot se movía mediante eventos de teclado con JavaScript.

Las pinzas eran dos ortoedros a los lados del robot. Estas pinzas estaban creadas con *a-box* y eran independientes del robot. Con este prototipo se estudió el movimiento de las pinzas con respecto a la posición del robot. El código JavaScript era muy complejo y todos los objetos dependían de la actualización continua explícita de la posición. Se encontraron muchas dificultades a la hora de realizar los giros del robot con las pinzas, dado que éstas estaban definidas por el centro de masas y al rotar las pinzas había que hacerlo con funciones senoidales para que se movieran en concordancia con la rotación del robot. En la Figura 5.13 y en estos vídeos<sup>12</sup> se puede ver el prototipo y los problemas con el giro.

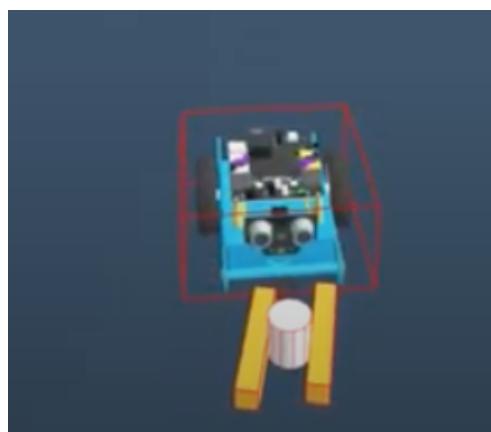


Figura 5.13: Prototipo robot con pinza en A-Frame nativo.

En este trabajo fin de grado el prototipo mencionado se ajustó y se llevó a Websim pero no funcionaba bien. Esta dependencia continua de la posición del robot con respecto a las pinzas y el cierre de las pinzas con las pinzas era muy complejo y muy poco realista. En este vídeo<sup>3</sup> se puede ver cómo era el funcionamiento en Websim con el JavaScript de este primer prototipo.

<sup>1</sup><https://www.youtube.com/watch?v=BpAujxcWx-Y>

<sup>2</sup>[https://www.youtube.com/watch?v=w5plHB\\_4G7Y](https://www.youtube.com/watch?v=w5plHB_4G7Y)

<sup>3</sup><https://www.youtube.com/watch?v=eVM9n4mziTQ&t=46s>

### 5.3.2. Modelo Mbot con pinzas

La idea principal de este ejercicio era que el robot pudiera coger con las pinzas una lata de la forma más natural posible. Para el robot base se usó el modelo del Mbot, ya que teníamos su modelo 3D en formato .gltf, para que fuera diferente a los anteriores que se usaban en la plataforma, se cambió el color del robot desde Blender con tonos amarillos y azules (Figura 5.14). Se insertó en el fichero de configuración del ejercicio de la siguiente forma:

```

1  {
2      "tag": "a-robot",
3
4      "attr": {
5          "id": "a-pibot",
6
7          "gltf-model": "/static/websim/assets/models/mbot_base.gltf",
8
9          "scale": { "x":25, "y":25, "z":25},
10         "position": { "x":-35, "y":0.5, "z":-6},
11         "rotation": { "x":0, "y":0, "z":0},
12         "dynamic-body": {"mass": 1, "shape": "none"},
13         "shape_main": {"shape": "box",
14             "halfExtents": "0.055 0.04 0.05",
15             "offset": "-0.025 0.04 0"
16         }
17     }
18 },

```

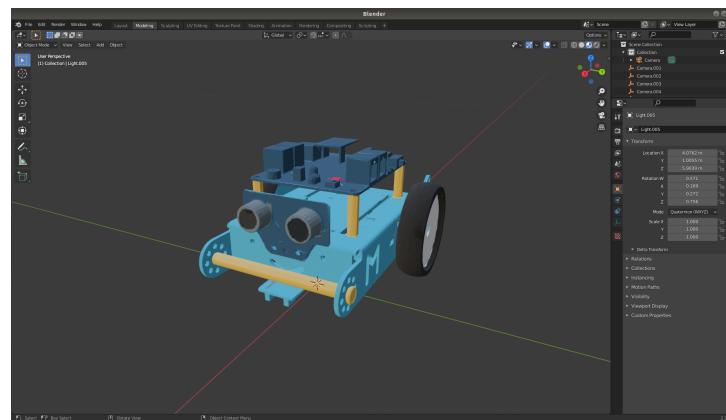


Figura 5.14: Modelo Mbot cambiado de color

Para las pinzas, primero se hicieron con ortoedros gracias a la etiqueta `<a-box>` de A-Frame. Pero finalmente se cambiaron por un modelo más moderno diseñado en Blender (Figura 5.15) .

### 5.3. MODELO ROBOT CON PINZA

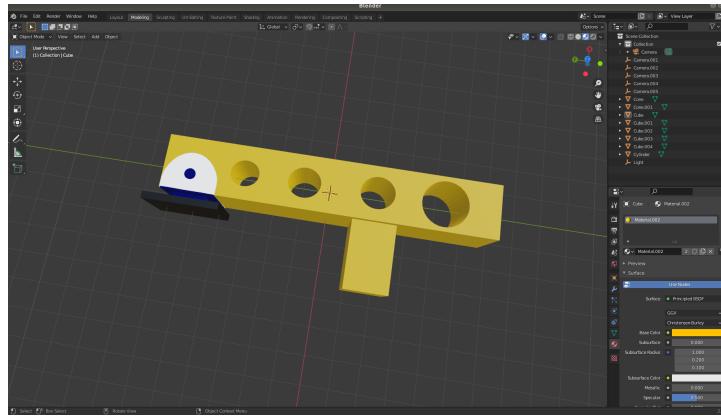


Figura 5.15: Modelo Pinza

Las pinzas tenían que ser dependientes del robot para que su movimiento y los giros fueran solidariamente coherentes. Para conseguir esta dependencia de las pinzas con el robot, desde el fichero de configuración, las pinzas se definieron como objetos hijos (`childs`) del objeto robot base (el Mbot). De esta forma las pinzas y el robot forman un mismo objeto y se mueven a la vez.

```
1 {      "tag": "a-robot",
2   "attr": {
3     "id": "a-pibot",
4     "gltf-model": "/static/websim/assets/models/mbot_base.gltf",
5     "scale": { "x":25, "y":25, "z":25},
6     "position": { "x":-35, "y":0.5, "z":-6},
7     "rotation": { "x":0, "y":0, "z":0},
8     "dynamic-body":{ "mass": 1, "shape":"none" },
9     "shape_main":{ "shape": "box",
10                   "halfExtents": "0.055 0.04 0.05",
11                   "offset": "-0.025 0.04 0"
12                 }
13   },
14
15   "childs": [
16     { "tag": "a-entity",
17       "attr": {
18         "id": "gripper-left",
19         "gltf-model": "/static/websim/assets/models/gripper.gltf",
20         "position": { "x":-0.033, "y":0, "z":-0.05},
21         "scale": { "x":0.005, "y":0.005, "z":0.006},
22         "rotation": { "x": 0, "y":-8, "z":180},
23         "body":{ "type": "static", "mass": 1, "shape":"none" },
24         "shape_main":{ "shape": "box",
25           "halfExtents": "1.5 1.5 6.5",
```

### 5.3. MODELO ROBOT CON PINZA

---

```
26             "offset": "0 0 0"},  
27             "shape__handle": {"shape": "box",  
28                 "halfExtents": "1.5 1 1",  
29                 "offset": "-2.2 -0.2 1.75"}  
30         }  
31     },  
32     { "tag": "a-box",  
33         "attr": {  
34             "id": "gripper-right",  
35             "gltf-model": "/static/websim/assets/models/gripper.gltf",  
36             "position": { "x": 0.033, "y": 0, "z": -0.05},  
37             "scale": { "x": 0.005, "y": 0.005, "z": 0.006},  
38             "rotation": { "x": 0, "y": 8, "z": 0},  
39             "body": {"type": "static", "mass": 1, "shape": "none"},  
40             "shape__main": {"shape": "box",  
41                 "halfExtents": "1.5 1.5 6.5",  
42                 "offset": "0 0 0"},  
43             "shape__handle": {"shape": "box",  
44                 "halfExtents": "1.5 1 1",  
45                 "offset": "-2.2 -0.2 1.75"}  
46         }  
47     },  
48 ]  
49 },
```

A continuación se explica cómo se han ajustado los atributos de los objetos mostrados en el código anterior. En A-Frame existen dos tipos de cuerpos: *dynamic-body* y *static-body*. Un objeto con *dynamic-body* se mueve libremente. Los cuerpos dinámicos tienen masa, chocan con otros objetos, rebotan o se ralentizan durante las colisiones y caen si la gravedad está habilitada. Los cuerpos estáticos son objetos animados o de posición fija. Otros objetos pueden chocar con cuerpos estáticos, pero los cuerpos estáticos en sí mismos no se ven afectados por la gravedad ni las colisiones.

El robot se definió como un cuerpo dinámico mientras que las palas de las pinzas eran cuerpos estáticos, de esta forma, las palas pueden estar elevadas del suelo y no les afecta radicalmente las colisiones que pueda tener con la lata a la hora de cerrarse. La lata en cambio es un objeto dinámico.

La malla de colisión se define por *shape\_main* y *shape\_handle*. En el atributo *body* se pone “shape: none” para poder ajustar la malla de colisión que viene por defecto con *shape\_main* y *shape\_handle*. *Shape\_main* es la malla de colisión principal en la que se puede elegir la forma, *shape*, lo que ocupa *halfExtents* y se puede desplazar del centro del objeto con *offset*.

### 5.3. MODELO ROBOT CON PINZA

---

Esto permite hacer la malla de colisión más grande o más pequeña que las dimensiones de nuestro objeto. `Shape_handle` permite añadir una malla de colisión extra independientemente de si dentro de ese objeto hay otro objeto o no. Esta malla de colisión extra se define respecto a la malla de colisión `shape_main`. Ambas mallas de colisión dependerán de la posición del objeto y se moverán lo mismo que él.

La malla de colisión extra se usó para la parte cúbica que tiene la parte central del modelo de la pinza usando la forma `“shape” : “box”`, ajustando la posición y dimensiones del cubo, mientras que la malla de colisión principal se asignó a la parte del modelo con forma de ortoedro, también con `“shape” : “box”` pero ajustándolo a las medidas del ortoedro.

En la Figura 5.16 se muestra el Mbot con pinzas con sus modelos, físicas y sus 5 mallas de colisión.

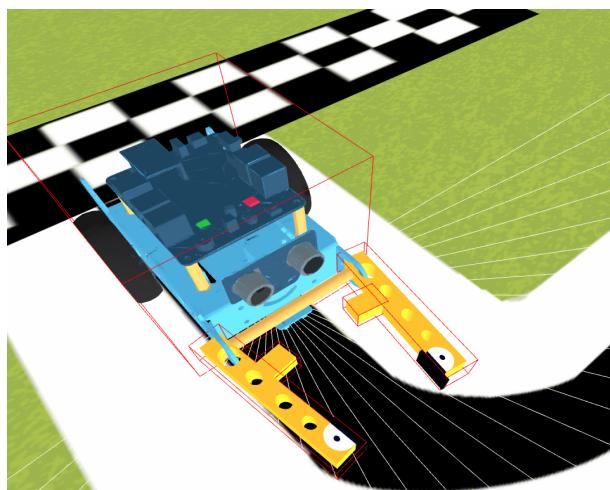


Figura 5.16: Mallas de colisión del robot

Los usuarios tienen que programar el Mbot para que éste abra y cierre las pinzas cuando la lata esté cerca. Para ello, implementaron las funciones `closeGripper()`, `openGripper()` y `getObjectInGripper()` en JavaScript con las que se materializan a su vez los bloques y funciones respectivas en Scratch y Python que los usuarios usan para manejar todas las operaciones del robot.

En Scracth se crearon nuevos bloques en los Motores: Abrir pinza, Cerrar Pinza y Dame objeto en pinza. En Python se crearon las nuevas funciones: `HAL.abrir_pinza()`, `HAL.cerrar_pinza()` y `HAL.dame_objeto_en_pinza()` que en el fon-

### 5.3. MODELO ROBOT CON PINZA

---

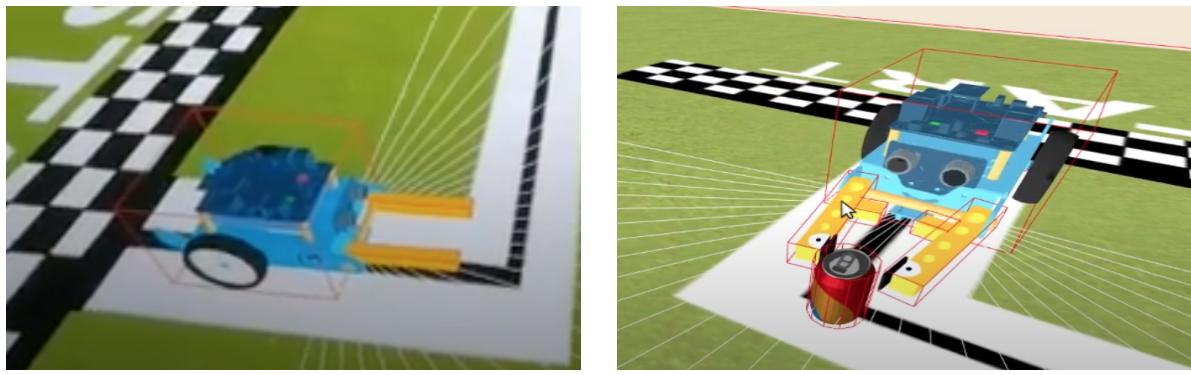
do todas ellas llaman las funciones JavaScript del siguiente código para cambiar la posición x de las pinzas y obtener la distancia entre la lata y el robot.

```
1 export async function closeGripper() {
2
3     let thread = getThread(this.myRobotID);
4     thread.blocking_instruction = true;
5     let gripperLeft = document.querySelector("#gripper-left");
6     let gripperLeftPos = gripperLeft.object3D.position;
7     let gripperRight = document.querySelector("#gripper-right");
8     let gripperRightPos = gripperRight.object3D.position;
9     let val = 0.025;
10    while (getThread(this.myRobotID).status !== "RELOADING" && gripperLeftPos.x < -0.020 &&
11          gripperRightPos.x > 0.020) {
12        val = val - 0.001;
13        await document.querySelector("#gripper-left").setAttribute('position', {x: -val, y: 0,
14                               z: -0.05});
15        await document.querySelector("#gripper-right").setAttribute('position', {x: val, y: 0,
16                               z: -0.05});
17        await this.sleep(0.2);
18    }
19    thread.blocking_instruction = false;
20 }
21
22 export function getObjectInGripper() {
23     let distance = this.getDistance();
24     if (distance < 1.5) {
25         return 1;
26     } else {
27         return 0;
28     }
29 }
30
31 export async function openGripper() {
32     await document.querySelector("#gripper-left").setAttribute('position', {x: -0.030, y: 0, z
33                           : -0.05});
34     await document.querySelector("#gripper-right").setAttribute('position', {x: 0.030, y: 0, z
35                           : -0.05});
36 }
```

Una vez creado el robot y las funciones necesarias, se estudió la posibilidad de que ajustando las mallas de colisión de las pinzas se pudiera atrapar una lata con la fuerza que ejercieran al cerrarse. Las mallas de colisión son las líneas rojas que delimitan el entorno de un objeto y permiten definir una forma física alrededor de modelos personalizados. Como hemos comentado anteriormente, esto hace que funcionen mejor y se comporten con mayor realismo.

## 5.4. EVALUADOR AUTOMÁTICO

Al cerrar las pinzas de esta forma, la lata colisionaba con las mallas de colisión de las pinzas pero no se quedaba atrapada. Para solucionar ésto, se rotaron las pinzas 8 grados en el eje y. También se ajustaron las físicas del mundo (*friction:0.0002*) para que el robot pudiera arrastrar la lata sin problemas por el escenario. En la Figura 5.17 (a) podemos ver cómo eran las pinzas sin rotar y con ortoedros y en la Figura 5.17 (b), las nuevas pinzas con el modelo de Blender rotado 8 grados que finalmente se utilizó.



(a) Primeras pinzas con ortoedros rectas

(b) Pinzas rotadas 8 grados

Figura 5.17: Robot Mbot con pinzas

## 5.4. Evaluador automático

Por último, se añadió un evaluador automático para mostrar el porcentaje del circuito recorrido con puntos de control y unos iconos para indicar que la lata está cogida o no. Esto se hizo en un fichero .js que se importa al ejercicio llamado `gripper_evaluator.js`. En la Figura 5.18 se muestra la visualización del evaluador. En el siguiente código se puede ver cómo se ha implementado la comprobación de los puntos de control y si el robot coge la lata o no.

```
1 var checkpoints= [[-32,0,-6], [-31,0,-11], [-31,0,-17], [-30,0,-28], [-27,0,-32],
2   [-22,0,-34], [-19,0,-33], [-17,0,-32], [-15,0,-28], [-14,0,-24],
3   [-14,0,-20], [-15,0,-16], [-18,0,-12], [-24,0,-4], [-27,0,2], [-27,0,6],
4   [-26,0,12], [-18,0,16], [-13,0,8], [-12,0,0], [-13,0,-5], [-8,0,-10], [-5,0,-10], [-4,0,-10],
5   [-5,0,-10], [-8,0,-10], [-13,0,-5], [-12,0,0], [-13,0,8], [-18,0,16], [-26,0,12],
6   [-27,0,6], [-27,0,2], [-24,0,-4], [-18,0,-12], [-15,0,-16], [-14,0,-20],
7   [-14,0,-24], [-15,0,-28], [-17,0,-32], [-19,0,-33], [-22,0,-34], [-27,0,-32],
8   [-30,0,-28], [-31,0,-17], [-31,0,-11], [-32,0,-6], [-37,0,-6]
9 ];
10 let robot = Websim.simulation.getHalAPI(robID);
11 var x = Math.round(robot.getPosition().x)
```

## 5.4. EVALUADOR AUTOMÁTICO

---

```
5   var z = Math.round(robot.getPosition().z)
6   var element = document.getElementById("a-car1bar");
7   var totalpuntuation = document.getElementById("totalpuntuation");
8   var status = false;
9
10  var can = document.getElementById('object')
11  var canx = can.getAttribute('position').x
12  var canz = can.getAttribute('position').z
13  var takencan_distance = Math.sqrt(Math.pow((canx-x), 2)+Math.pow((canz-z), 2));
14  if (takencan_distance <= 2.5){
15    extrapoint.innerHTML= "&#x1f3c5 &#x2728";
16  }else{
17    extrapoint.innerHTML= "";
18  }
19
20  var d = Math.sqrt(Math.pow((checkpoints[n][0]-x), 2)+Math.pow((checkpoints[n][2]-z), 2));
21  var d1 = Math.sqrt(Math.pow((checkpoints[n][0]+1-x), 2)+Math.pow((checkpoints[n][2]+1-z),
22    2));
22  var d2 = Math.sqrt(Math.pow((checkpoints[n][0]-1-x), 2)+Math.pow((checkpoints[n][2]-1-z),
23    2));
23  var d3 = Math.sqrt(Math.pow((checkpoints[n][0]+1-x), 2)+Math.pow((checkpoints[n][2]-1-z),
24    2));
24  var d4 = Math.sqrt(Math.pow((checkpoints[n][0]-1-x), 2)+Math.pow((checkpoints[n][2]+1-z),
25    2));
25
26  if (d <= 1 || d1 <= 1 || d2 <= 1 || d3 <= 1 || d4 <= 1) {
27    n+=1
28    if(n >= 48) {
29      if (takencan_distance <= 2.5){
30        totalpuntuation.innerHTML=100;
31      }
32      status = true;
33      element.style.width = 100 + ' %';
34      element.innerHTML = 100 + ' %';
35    }
36  }
37
38  var completed = Math.round(n/checkpoints.length*100)
39
40  if (n!=48) {
41    element.style.width = Math.round(completed) + ' %';
42    element.innerHTML = Math.round(completed) + ' %';
43    totalpuntuation.innerHTML=Math.round(completed*0.66);
44    if (takencan_distance <= 2.5){
45      totalpuntuation.innerHTML=Math.round(completed*0.66)+14;
46    }
47  }
```

## 5.5. SOLUCIÓN DE REFERENCIA

---

Los puntos de control están repartidos por toda la mitad del circuito que es el recorrido que tiene que hacer el robot. Los puntos se van comparando uno a uno, de modo que si no ha pasado por el primer punto y el robot pasa por el segundo, no se tiene en cuenta, porque no ha recorrido el circuito de forma correcta. Esto se ha realizado con la distancia euclídea, si el robot se encuentra en un radio cercano al punto de control que le corresponda del circuito y ya ha pasado por los anteriores, se aumenta el porcentaje de circuito recorrido, si no, no.

Cuando la lata está entre las pinzas se muestra una medalla y unas estrellas. Esto se ha realizado también con la distancia euclídea, donde calculamos la distancia entre el robot y la lata y si es menor o igual a 2'5 se añade esa puntuación extra.

En la Figura 5.18, podemos ver que el robot ha recorrido bien un 6 % del circuito pero a partir de ahí no lo ha recorrido correctamente, al llegar a la lata debería tener alrededor del 50 % de circuito recorrido. A pesar de que no ha seguido bien la línea del circuito, ha cogido la lata, por eso se muestran los iconos.



Figura 5.18: Evaluador automático del juego del pañuelo

## 5.5. Solución de referencia

Este ejercicio se puede resolver de muchas formas, unas posibles soluciones son las que se muestran en la Figura 5.19(a) para Python y en la Figura 5.19(b) para Scratch. En estas soluciones reflejan el código necesario para hacer un sigue líneas con el sensor de infrarrojos que dependiendo de su valor, el robot avanzará, girará a la izquierda o a la derecha hasta llegar

## 5.5. SOLUCIÓN DE REFERENCIA

a la lata. Con el sensor *Dame objeto en pinza o HAL.dame\_objeto\_en\_pinza()* se obtiene un 1 cuando la lata está entre las pinzas o un 0 si no lo está. Cuando el sensor detecta que la lata está entre las pinzas (devuelve un 1), el robot se para, cierra las pinzas, espera un segundo para que le dé tiempo a cerrar bien las pinzas y gira 180 grados. En este momento se vuelve a usar el sensor infrarrojos para detectar la línea y el robot vuelve por donde ha venido llevando la lata consigo.

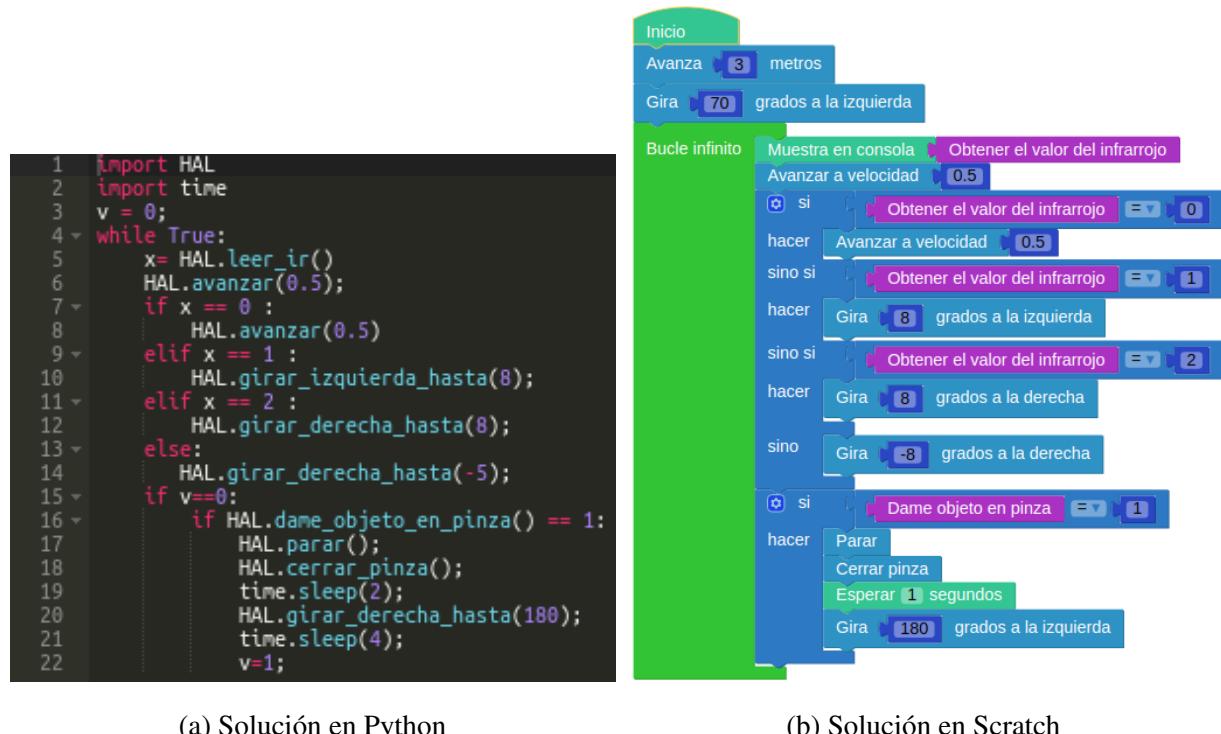


Figura 5.19: Soluciones de referencia

En estos videos promocionales que se han realizado<sup>4</sup> <sup>5</sup>, se muestra el funcionamiento de estas dos soluciones.

<sup>4</sup><https://www.youtube.com/watch?v=3VKYRPnS3Vw>

<sup>5</sup><https://www.youtube.com/watch?v=LkTpuYItZmE>

# Capítulo 6

## Ejercicio teleoperador acústico y banda sonora

En este capítulo describe cómo se ha realizado el ejercicio del teleoperador acústico, que aborda el análisis de audio de entrada y se nombran las herramientas utilizadas. También se muestra cómo se ha añadido la posibilidad de incluir efectos de sonido y bandas sonoras (audio de salida) a los ejercicios ya existentes.

La entrada y salida de audio desde aplicaciones web es muy interesante para conseguir una mayor interactividad con el usuario. El sonido es muy importante en el mundo real y llevarlo a Websim permite que la simulación sea más inmersiva y la experiencia del usuario sea más agradable. El audio de entrada o *audio in* en este trabajo lo vamos a utilizar para obtener las muestras del micrófono y analizarlo para reconocer las palabras que entiende nuestro robot y actúe de forma adecuada. El audio de salida o *audio out* se usará para completar los ejercicios con bandas sonoras y el sonido de colisión.

### 6.1. Enunciado

En este ejercicio se va a desarrollar un teleoperador acústico. El objetivo era crear un algoritmo de reconocimiento de voz con JavaScript, para poder dirigir un robot de Kibotics con la voz. El alumno deberá indicarle al robot órdenes claras para que él reconozca cada orden y pueda moverse por el escenario. Hay dos escenarios disponibles, un muro de piedra y una portería de rugby. El robot elegido es un dron que deberá moverse por el mundo sin chocar con los objetos.

## 6.2. Modelos del mundo

Para empezar con este ejercicio se utilizó un fichero de configuración sencillo en el que sólo aparecía un dron y un plano con una textura de césped. Este fichero es necesario para que el simulador Websim construya la escena 3D. Ésta sería una parte de ese fichero donde se muestra la configuración de la escena (*scene*), los robots (*robots\_config*), las texturas (*assets*) y los objetos (*objects*).

```
1 { "scene-parent-id": "myIFrame",
2   "scene": {
3     "id": "scene",
4     "gravity": 0,
5     "sky": "../../assets/textures/sky.png",
6     "background": "color: gray;",
7     "inspector": "url: https://aframe.io/releases/0.4.0/aframe-inspector.min.js",
8     "embedded": true,
9     "physics": "debug: true"
10   },
11   "robots_config": [
12     { "controller": "user1",
13       "id": "a-pibot"
14     }
15   ],
16   "assets": [
17     { "tag": "img",
18       "attr": {
19         "id": "ground",
20         "alt": "Texture for the scene ground",
21         "src": "../../assets/textures/escenarioLiso-min.png"
22       }
23     ],
24   "objects": [
25     { "tag": "a-plane",
26       "attr": {
27         "static-body": {
28           "mass": 100000
29         },
30         "position": { "x":0, "y":0, "z":0 },
31         "rotation": { "x":-90, "y":0, "z":0 },
32         "width": "100",
33         "height": "100",
34         "src": "#ground"
35       }
36     },
37     { "tag": "a-robot",
```

## **6.2. MODELOS DEL MUNDO**

```
38     "attr": {
39         "id": "a-pibot",
40         "gltf-model": "../../assets/models/drone_animation.gltf",
41         "scale": { "x":0.5, "y":0.5, "z":0.5},
42         "position": { "x":0, "y":4, "z":0},
43         "rotation": { "x":0, "y":90, "z":0},
44         "dynamic-body": {"mass": 1}
45     },
46 }
```

Para este ejercicio se crearon dos escenarios: uno con portería de rugby y otro con una pared.

La portería se creó con Blender y la pared con A-Frame. Ver Figura 6.1 y Figura 6.2.

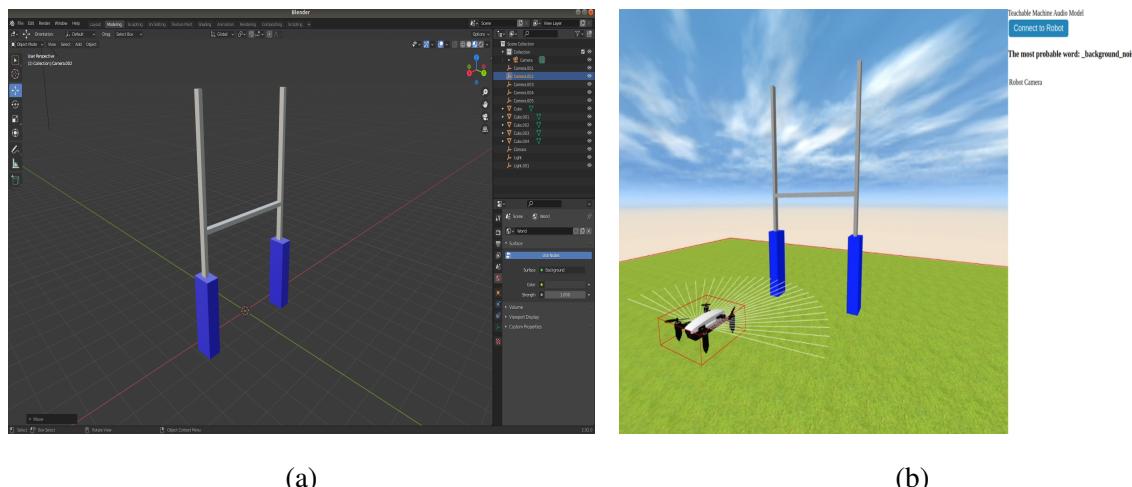


Figura 6.1: Modelos portería en Blender y en Websim

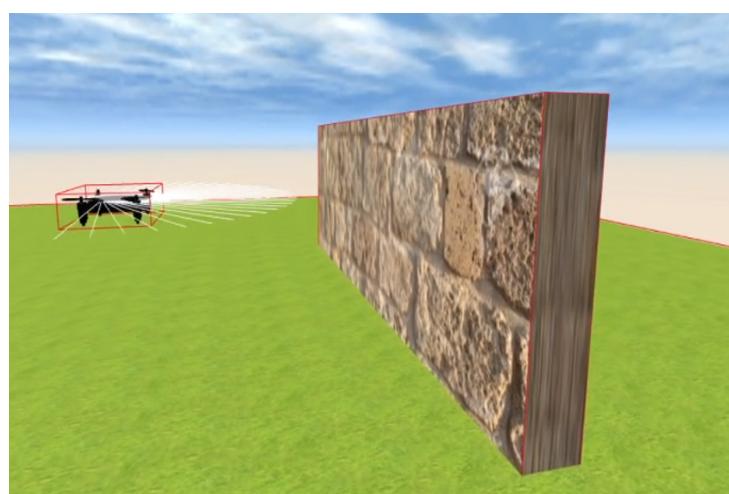


Figura 6.2: Pared de piedra en Websim

Estos modelos se han añadido a los respectivos ficheros de configuración de la siguiente forma:

```

1 ...
2     "assets": [
3         { "tag": "a-asset-item",
4             "attr": {
5                 "id": "porteria_rugby"
6             },
7             "tag": "img",
8                 "attr": {
9                     "id": "wall",
10                    "alt": "Texture for the scene wall",
11                    "src": "../../assets/textures/wall.jpg"
12                }
13 ...
14 ],
15 "objects": [
16     { "tag": "a-entity",
17         "attr": {
18             "id": "porteria_rugby",
19             "gltf-model": "../../assets/models/porteria_rugby.gltf",
20             "position": { "x":0, "y":0, "z":-15},
21             "scale": { "x":3, "y":3, "z":3}
22         }
23     },
24     { "tag": "a-box",
25         "attr": {
26             "id": "wall",
27             "width": "30",
28             "height": "20",
29             "depth": "2",
30             "position": { "x":0, "y":0, "z":-15},
31             "src": "#wall",
32             "static-body": {
33                 "mass": 1
34             }
35         }
36     }, ...

```

## 6.3. Procesamiento de audio con *Teachable Machine*

El procesamiento de audio es muy complejo y existen muchas herramientas diferentes para implementarlo desde JavaScript. En este trabajo se han investigado tres herramientas y final-

### 6.3. PROCESAMIENTO DE AUDIO CON TEACHABLE MACHINE

---

mente se ha decidido usar Teachable Machine, que es una herramienta basada en la Web para crear modelos de aprendizaje automático. Destaca por usar modelos de clasificación de imágenes, sonidos y posturas. Teachable Machine internamente usa TensorFlowJS .

La creación de un modelo en Teachable Machine tiene tres fases: recopilación, entrenamiento y exportación del modelo a tu proyecto.

- *Recopilación :*

Para hacer el modelo primero hay que recopilar muestras. Para ello fue necesaria la participación de 10 personas, para que el modelo generado fuera más completo y pudiera reconocer la palabra independientemente de si la voz del usuario es más aguda o más grave. Se usaron aproximadamente 10 muestras de cada palabra por persona. El modelo cuenta con 105 muestras por cada palabra (10 muestras aproximadamente x 10 personas). Las palabras que queremos reconocer son *go, stop, back, right, left, up, down* y el ruido de fondo. Esto ha supuesto un total de 840 muestras.

Las muestras se grabaron una a una en la página de creación de modelos de Teachable Machine. En la Figura 6.3 se puede ver el modelo y las muestras grabadas.

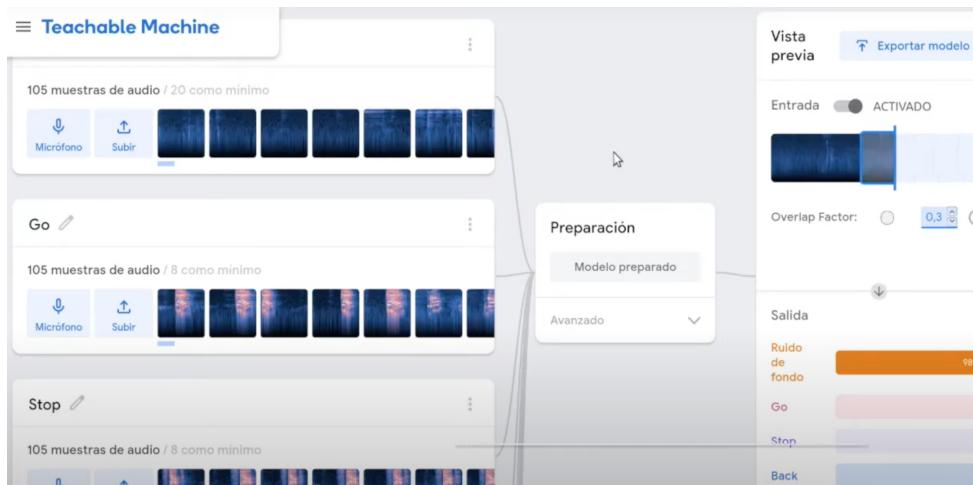


Figura 6.3: Modelo realizado en Teachable Machine

- *Entrenamiento :*

Una vez conseguidas todas las muestras, hay que entrenar el modelo, que se hace en ‘épocas’. Teachable Machine permite cambiarlas. Gracias a unas tablas de precisión y pérdidas por época puedes guiarte para que en función de tus datos puedas ajustar este parámetro.

### 6.3. PROCESAMIENTO DE AUDIO CON TEACHABLE MACHINE

---

Para este proyecto se utilizaron los valores predeterminados, los cuales proporcionaron buenos resultados.

- *Exportación del modelo a Websim :*

Una vez entrenado, la página web de Teachable Machine muestra una vista previa en el que puedes probar y ver el porcentaje de aciertos de cada palabra y con esto decidir si necesitas añadir más muestras o cambiar los parámetros de tu modelo. En la Figura 6.4 podemos ver la visualización de prueba.

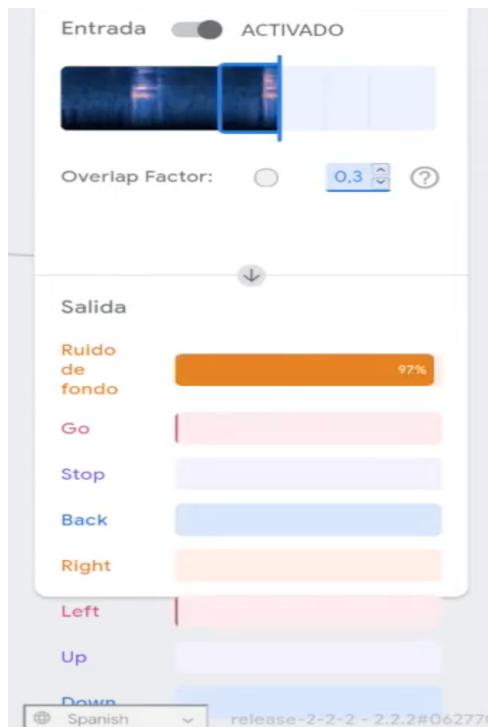


Figura 6.4: Vista previa del modelo

Una vez ajustado y con las 105 muestras por palabra, se exportó el modelo. Para exportarlo, Teachable Machine puede alojarlo en sus servidores y proporcionarte un enlace de forma gratuita o puedes descargarlo. Se optó por guardarlo en la nube de Teachable Machine para que no ocupara mucho espacio en Kibotics, ya que la forma de importarlo a JavaScript era similar en ambas opciones. Una vez subido a la nube, Teachable Machine proporciona una URL con la dirección de tu modelo y el código necesario para importarlo a tu proyecto desde JavaScript.

### 6.3. PROCESAMIENTO DE AUDIO CON TEACHABLE MACHINE

---

Con estos tres pasos ya tenemos creado nuestro modelo en Teachable Machine. Para probar el funcionamiento de este modelo se hizo una primera prueba de reconocimiento de audio en una página web propia y específica, todavía sin fusionarlo con el comportamiento del robot en Websim. En este enlace <sup>1</sup> se puede probar el funcionamiento de este código basado en HTML5, CSS3, JavaScript y Teachable Machine (TensorFlowJS) (Ver Figura 6.5). Esta página web pide permiso para acceder al micrófono, cuando lo tiene, analiza el audio y muestra la probabilidad de cada palabra en tiempo real.

Este es el código de esa página web de prueba del modelo, que incluye el código que nos facilita Teachable Machine a la hora de exportar el modelo:

```
1 <html>
2 <head>
3   <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.3.1/dist/tf.min.js"></script>
4   <script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/speech-commands@0.4.0/dist/
5     speech-commands.min.js"></script>
6   <style type="text/css">
7     button{
8       text-decoration: none;
9       padding: 10px;
10      font-weight: 600;
11      font-size: 20px;
12      color: #ffffff;
13      background-color: #AAA;
14      border-radius: 6px;
15      border: 2px solid black;
16    }
17    button:hover{
18      color: black;
19      background-color: #ffffff;
20    }
21    button:active {
22      box-shadow: 0 2px #666;
23      transform: translateY(2px);
24    }
25   </style>
26 </head>
27 <body>
28 <div>Teachable Machine Audio Model</div>
29 <button type="button" onclick="init()">Start</button>
30 <h3 id = "prediction" > The most probable word: </h3>
31 <div id="label-container" style="background-color: #D3D3D3;"></div>
```

---

<sup>1</sup>[https://martaquintana.github.io/Audio\\_Recognition/index.html](https://martaquintana.github.io/Audio_Recognition/index.html)

### 6.3. PROCESAMIENTO DE AUDIO CON TEACHABLE MACHINE

---

```
32
33 <script type="text/javascript">
34     // more documentation available at
35     // https://github.com/tensorflow/tfjs-models/tree/master/speech-commands
36
37     // the link to your model provided by Teachable Machine export panel
38 const URL = "https://teachablemachine.withgoogle.com/models/P3XdF5r5d/";
39
40 async function createModel() {
41     const checkpointURL = URL + "model.json"; // model topology
42     const metadataURL = URL + "metadata.json"; // model metadata
43
44     const recognizer = speechCommands.create(
45         "BROWSER_FFT", // fourier transform type, not useful to change
46         undefined, // speech commands vocabulary feature, not useful for your models
47         checkpointURL,
48         metadataURL);
49
50     // check that model and metadata are loaded via HTTPS requests.
51     await recognizer.ensureModelLoaded();
52
53     return recognizer;
54 }
55
56 async function init() {
57     const recognizer = await createModel();
58     const classLabels = recognizer.wordLabels(); // get class labels
59     const labelContainer = document.getElementById("label-container");
60     const word = document.getElementById("prediction");
61     for (let i = 0; i < classLabels.length; i++) {
62         labelContainer.appendChild(document.createElement("div"));
63     }
64
65     // listen() takes two arguments:
66     // 1. A callback function that is invoked anytime a word is recognized.
67     // 2. A configuration object with adjustable fields
68     recognizer.listen(result => {
69         const scores = result.scores; // probability of prediction for each class
70         var word_index = 0;
71         // render the probability scores per class
72         for (let i = 0; i < classLabels.length; i++) {
73             const classPrediction = classLabels[i] + ": " + result.scores[i].toFixed(2);
74             labelContainer.childNodes[i].innerHTML = classPrediction;
75             //The most probable word
76             if (result.scores[i].toFixed(2) >= result.scores[word_index].toFixed(2)) {
77                 word_index = i;
78             }
79         }
80     });
81 }
```

### 6.3. PROCESAMIENTO DE AUDIO CON TEACHABLE MACHINE

```
79      }
80      var prediction = classLabels[word_index];
81      word.innerHTML = "The most probable word: " + prediction;
82      //console.log(prediction);
83    },
84    includeSpectrogram: true, // in case listen should return result.spectrogram
85    probabilityThreshold: 0.75,
86    invokeCallbackOnNoiseAndUnknown: true,
87    overlapFactor: 0.50 // probably want between 0.5 and 0.75. More info in README
88  );
89}
90</script>
91</body>
92</html>
```

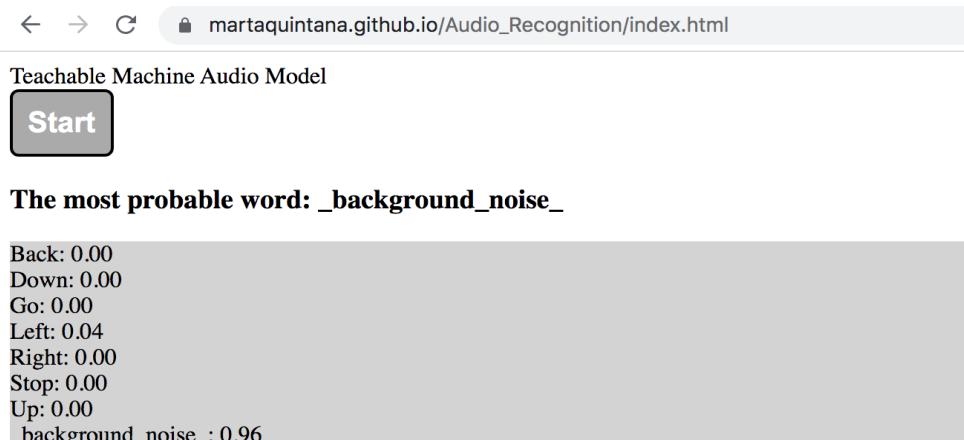


Figura 6.5: Página web de prueba de reconocimiento de audio

En la Figura 6.6 se muestra el diagrama simplificado de cómo *Teachable Machine* analiza el audio que viene del micrófono, es capaz de devolver una predicción, la palabra con mayor probabilidad.

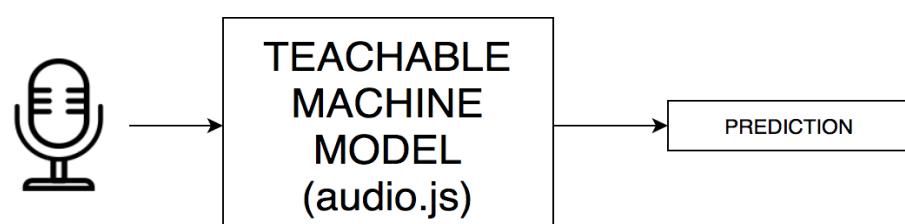


Figura 6.6: Análisis audio de entrada con Teachable Machine

### 6.3. PROCESAMIENTO DE AUDIO CON TEACHABLE MACHINE

---

Una vez se comprobó que el modelo de reconocimiento de audio funcionaba satisfactoriamente, se incorporó a Websim. Para ello, se creó el siguiente fichero `audio.js` que se importa en el HTML del ejercicio de la siguiente forma:

```
1 <script type="text/javascript" src="js/audio.js" >
```

Código de `audio.js`:

```
1 document.addEventListener('robot-loaded', (evt)=>{
2     localRobot = evt.detail;
3     console.log(localRobot);
4 });
5
6 // the link to your model provided by Teachable Machine export panel
7 const URL = "https://teachablemachine.withgoogle.com/models/P3XdF5r5d/";
8 var connection = false;
9 var noise_times = 0;
10
11 async function createModel() {
12     const checkpointURL = URL + "model.json"; // model topology
13     const metadataURL = URL + "metadata.json"; // model metadata
14
15     const recognizer = speechCommands.create(
16         "BROWSER_FFT", // fourier transform type, not useful to change
17         undefined, // speech commands vocabulary feature, not useful for your models
18         checkpointURL,
19         metadataURL);
20
21     // check that model and metadata are loaded via HTTPS requests.
22     await recognizer.ensureModelLoaded();
23
24     return recognizer;
25 }
26
27 async function analizar(recognizer,classLabels,word) {
28     // listen() takes two arguments:
29     // 1. A callback function that is invoked anytime a word is recognized.
30     // 2. A configuration object with adjustable fields
31     recognizer.listen(result => {
32         console.log(noise_times)
33         const scores = result.scores; // probability of prediction for each class
34         var word_index = 0;
35
36         // render the probability scores per class
37         for (let i = 0; i < classLabels.length; i++) {
38             const classPrediction = classLabels[i] + ":" + result.scores[i].toFixed(2);
39             //The most probable word
40             if (result.scores[i].toFixed(2) >= result.scores[word_index].toFixed(2)) {
41                 word_index = i;
```

### 6.3. PROCESAMIENTO DE AUDIO CON TEACHABLE MACHINE

---

```
41         }
42     }
43     var prediction = classLabels[word_index];
44     word.innerHTML = "The most probable word: " + prediction;
45     if (String(prediction) == "_background_noise_"){
46         noise_times = noise_times + 1;
47     }else{
48         noise_times = 0;
49     }
50     if (connection){
51         if (String(prediction) == "Go") {
52             localRobot.setV(0.9);
53             //que se mueva el robot
54         }else if (String(prediction) == "Stop") {
55             localRobot.setV(0);
56             localRobot.setW(0);
57             localRobot.setL(0);
58             //que se pare
59         }else if (String(prediction) == "Back"){
60             localRobot.setV(-0.9);
61         }else if (String(prediction) == "Right"){
62             localRobot.setW(-0.005);
63
64         }else if (String(prediction) == "Left"){
65             localRobot.setW(0.005);
66
67         }else if (String(prediction) == "Up"){
68             localRobot.setL(0.25);
69
70         }else if (String(prediction) == "Down"){
71             noise_times = 0;
72             console.log(localRobot.velocity);
73             if (localRobot.velocity.y = 0.25) {
74                 localRobot.setL(-0.25);
75             }
76         }
77         //For stop going down or going up, turn left or turn right say the word : STOP
78     }
79     //console.log(prediction);
80 }, {
81     includeSpectrogram: true, // in case listen should return result.spectrogram
82     probabilityThreshold: 0.75,
83     invokeCallbackOnNoiseAndUnknown: true,
84     overlapFactor: 0.50 // probably want between 0.5 and 0.75. More info in README
85 });
86
87 // Stop the recognition in 10 seconds.
```

### 6.3. PROCESAMIENTO DE AUDIO CON TEACHABLE MACHINE

---

```
88     setTimeout(() => {recognizer.stopListening(); }, 10000);
89 }
90
91 async function listen() {
92     document.getElementById("listen_again").style.visibility = 'hidden';
93     //-----TEACHABLE MACHINE-----
94     const recognizer = await createModel();
95     const classLabels = recognizer.wordLabels(); // get class labels
96     const word = document.getElementById("prediction");
97
98     var id = setInterval(() => {if(noise_times <= 10){analizar(recognizer,classLabels,word)
99         }else{
100             setTimeout(() => {console.log("Se ha parado de analizar")
101                 word.innerHTML = "The most probable word: " ;
102                 clearInterval(id);
103                 document.getElementById("listen_again").style.visibility = 'visible';
104             }, 10000);
105         } }, 10150);
106 }
107
108 async function Back\_To\_Listen(){
109     document.getElementById("prediction").innerHTML= 'The most probable word: <br> '
111     noise\_times = 0;
112     listen();
113 }
114
115 async function Connect\_To\_Robot () {
116     if (connection){connection= false;
117         document.getElementById("connection").style.backgroundColor= "red";
118         console.log( "Desconectado");
119     }else{
120         connection = true;
121         listen();
122         console.log( "CONECTADO AL ROBOT");
123         document.getElementById("connection").style.backgroundColor= "green";
124     }
}
```

Cuando se pulsa el botón *Conect\_To\_Robot* se llama a la función del mismo nombre que internamente ejecuta la función *listen()*, inicializa el reconocimiento y analiza el audio cada 10 segundos. Este botón se pone en verde cuando está analizando, cuando se vuelve a pulsar, el botón se pone en color rojo y se desconecta del análisis de audio. Mientras el botón esté de ese color el robot no reconoce ninguna palabra.

La función `listen()` inicializa el modelo y crea las etiquetas que necesita. Se ha creado un `setTimeout()` para esperar 10 segundos hasta que cargue el modelo. Cuando éste carga, se establece un `setInterval()` para que cada 10 segundos se ejecute la función `analizar()` que devuelve la palabra estimada.

La función `analizar()` es la que invoca al analizador/*recognizer*, que predice la palabra que es más probable que haya dicho el usuario. *Prediction* es la palabra que tiene la probabilidad más alta. Una vez tenemos la *prediction* se compara esta palabra con las 8 palabras que hemos entrenado el modelo: *go, stop, back, right, left, up, down y \_background\_noise\_*.

Gracias al evento `robot-loaded`, cuando el robot está cargado en el escenario, obtenemos el objeto `localRobot`, que en nuestro ejercicio es el dron. Kibotics ofrece una API<sup>2</sup> llamada HAL API<sup>3</sup> con funciones para controlar el movimiento del robot simulado. Cuando se predice una de las 8 palabras posibles, en los condicionales `if() else if()` se llama a una de estas funciones de velocidad :

- `LocalRobot.setV()`, para asignar la velocidad lineal: se utiliza para ir hacia delante *go*, ir hacia atrás *back* o parar *stop*.
- `LocalRobot.setW()` para la velocidad angular: se usa para girar a la derecha *right* o a la izquierda *left*.
- `LocalRobot.setL()` para asignar la velocidad de subida y bajada: subir *up* o bajar *down*.
- Si se predice que es ruido de fondo no afecta en el comportamiento del dron.

Con el fichero de configuración, el HTML y este código JavaScript el ejercicio se visualiza como se muestra en la Figura 6.7.

---

<sup>2</sup>Application Programming Interfaces

<sup>3</sup>Hardware Abstraction Layer

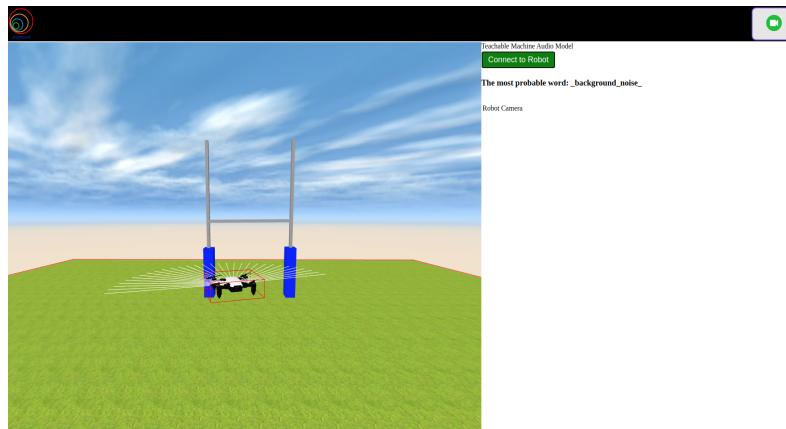


Figura 6.7: Ejercicio teleoperador acústico

La implementación del ejercicio con *Teachable Machine* nos ha aportado sencillez y más precisión que otros modelos que se estudiaron. Una vez realizado el ejercicio se optimizó para que el navegador del usuario no estuviera todo el tiempo analizando el audio si no tenemos nada que decirle al dron y aprovechar mejor así los recursos del ordenador del usuario.

La optimización se realizó añadiendo el contador (*noise\_times*) que aparece en el código anterior: éste cuenta las veces que se predice ruido de fondo. Se estableció un máximo de 10 las veces seguidas que la predicción es *\_background\_noise\_*. Como se analiza el audio cada 10 segundos, ésto quiere decir que si han pasado 100 segundos y siempre ha detectado ruido de fondo, la función *analizar()* para de ejecutarse y no se predice ninguna palabra. Cuando ésto ocurre, se muestra en el navegador un nuevo botón (*Continue analyzing*) que al pulsarlo desaparece y se reinicia el procesamiento de audio. Ver Figura 6.8.

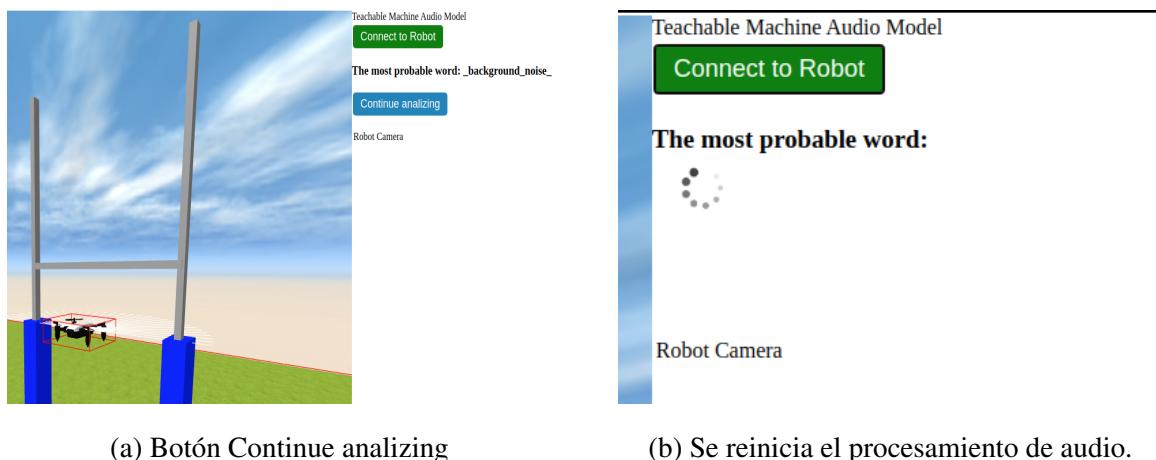


Figura 6.8: Optimización del modelo

## 6.4. Solución de referencia

Este ejercicio es un juego, el objetivo para el usuario no es programar nada, sino dirigir al dron con tu voz para cruzar por la portería de rugby o pasar de un lado al otro de la pared sin chocarte con ellas. Se ha realizado un vídeo promocional<sup>4</sup> y este otro vídeo<sup>5</sup> donde se muestran unas posibles soluciones para guiar al dron en los distintos escenarios. Ver Figura 6.9.

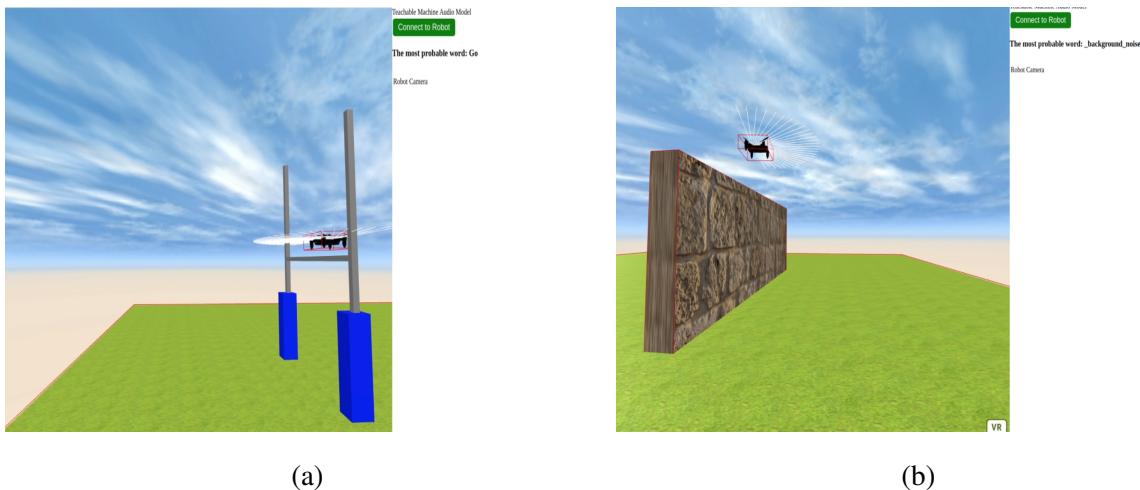


Figura 6.9: Ejemplo solución teleoperador acústico

## 6.5. Alternativas probadas

Para llevar a cabo este ejercicio se estudiaron otras dos herramientas de procesamiento de audio además de Teachable Machine: Web Audio API y TensorFlow JS.

Web Audio API permite escoger fuentes de audio, agregar efectos de sonido, crear visualizaciones, efectos espaciales, filtros y grabaciones, entre otras cosas. Se ha estudiado la posibilidad de hacer una red neuronal con esta API. Con esta tecnología, en este TFG se hizo un grabador donde el audio se grababa temporalmente en la memoria del navegador y un analizador en frecuencia en tiempo real [50]. En estos vídeos se puede ver el grabador<sup>6</sup> y el analizador<sup>7</sup> mencionados.

<sup>4</sup><https://youtu.be/sr54MxMw954>

<sup>5</sup><https://www.youtube.com/watch?v=85oTxgQQrck>

<sup>6</sup><https://www.youtube.com/watch?v=u9aerlWpCdM>

<sup>7</sup><https://www.youtube.com/watch?v=OZk4l7WFTZw>

## 6.5. ALTERNATIVAS PROBADAS

Web Audio API es muy útil para visualizado y efectos de sonido pero no nos proporcionaba la suficiente información para poder hacer procesamiento de audio y reconocer palabras.

TensorFlow JS es una biblioteca de JavaScript para el aprendizaje automático. Esta herramienta se utiliza para clasificación de imágenes, detección de objetos, segmentación del cuerpo, estimación de pose, detección de rostros y, entre otras, aplicaciones de reconocimiento y clasificación de comandos de voz [56]. Para empezar a entender cómo funcionan los modelos, capas y tensores, en definitiva, las redes neuronales en esta biblioteca, se realizaron distintos tutoriales de Youtube y de TensorFlow JS tanto de clasificación de imágenes como de reconocimiento de audio. En este video <sup>8</sup> se ven las diferentes pruebas que se hicieron. En la Figura 6.10 se puede ver un ejemplo de modelo de detección de objetos.

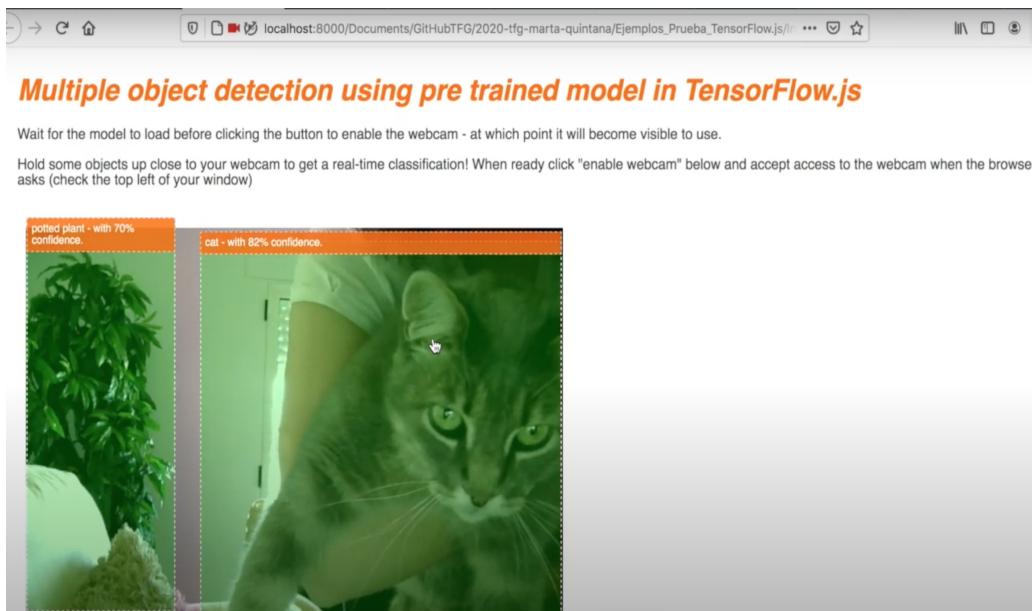


Figura 6.10: TensorFlow JS ejemplo detección de objetos.

Con TensorFlow JS se hicieron dos prototipos. El primero se realizó con un modelo pre-entrenado que reconocía 10 palabras entre ellas *go*, *stop*, *right*, *left* y *seven* (que se utilizó para que el robot fuera hacia atrás). Este modelo al ser pre-entrenado y definido por TensorFlow JS no era muy exacto y el ruido de fondo afectaba demasiado a la hora de detectar cada palabra en nuestro ejercicio.

El segundo prototipo se hizo con un modelo que se entrenaba con la voz del usuario del ejercicio. En este segundo prototipo cada palabra necesitaba grabar sus muestras, al menos unas

<sup>8</sup><https://www.youtube.com/watch?v=x8sUG1CyLdc>

## 6.6. BANDA SONORA

---

100 muestras por palabra, incluidas las muestras de ruido de fondo para que la detección fuera más precisa. Este modelo funcionaba más o menos bien pero no era lo más adecuado para el ejercicio que se había planteado. Se descartó porque era una tarea muy laboriosa grabar todas las muestras cada vez que entrabas en el ejercicio. En estos videos<sup>9</sup> <sup>10</sup> se puede ver cómo era el modelo.

Investigando el mundo de reconocimiento de audio y JavaScript se descubrió *Teachable Machine*, la tecnología que finalmente se eligió en este proyecto.

## 6.6. Banda Sonora

Para añadir banda sonora a los ejercicios existentes en Kibotics, se estudió cómo insertar audio desde HTML5 e implementarlo con JavaScript en Websim.

El elemento de HTML5 `<audio>` se usa para reproducir un fichero de audio en una página web. Los atributos que destacan de este elemento son *controls*, *source*, *loop* y *autoplay*. El atributo *controls* proporciona controles como reproducir, parar y subir o bajar el volumen. *Source* permite especificar la fuente de audio, en este caso un fichero en formato mp3, ogg o wav. *Loop* permite volver a reproducir el audio automáticamente cuando se termine, de esta forma el audio estará en bucle. Y *autoplay* se utiliza para que una vez se cargue el fichero de audio, éste empiece a reproducirse.

Para hacer esta mejora a Kibotics, en la carpeta *assets* de Websim se añadió una nueva carpeta llamada *soundtracks* con temas musicales para poner en los ejercicios. Las canciones usadas son de Patrick De Artega Royalty Free Music<sup>11</sup> y los temas que se han añadido son: Chiptronical, Common Fight, Goliath's Foe, Resilience, Spring Village, TheTrueStory of Beelzebub y Vals de su jardín.

La banda sonora de cada ejercicio se define en su fichero de configuración. En ese fichero se crea un objeto con la etiqueta “*soundtrack*”, el identificador “*audio*” y la dirección del fichero fuente del audio (“*src*”) que se quiere asignar como banda sonora, como se muestra en el siguiente código:

---

<sup>9</sup><https://www.youtube.com/watch?v=DcwJzCOE4kI>

<sup>10</sup><https://www.youtube.com/watch?v=nPeCAb47jiU>

<sup>11</sup> <https://patrickdearteaga.com> Credits to Patrick De Artega Royalty Free Music 2020

## 6.6. BANDA SONORA

---

```
1 { "tag": "soundtrack",
2   "attr": {
3     "id": "audio",
4     "src": "../assets/soundtracks/Spring_Village.ogg"
5   },
6 }
```

Websim tiene un analizador de ficheros JSON llamado `config-parser.js`, con este fichero JavaScript se analiza el fichero de configuración (incluyendo el objeto que hemos mencionado anteriormente) y lo traduce a A-Frame. Se ha mejorado este analizador de ficheros de Websim para materializar la banda sonora de cada ejercicio si está configurada.

Cuando Websim lee el fichero de configuración (formato JSON) se llama a `parseSoundtrack()` que recibe los objetos de la escena para analizar si hay banda sonora o no en el ejercicio.

```
1 await parseSoundtrack(sceneObjects);
```

En el siguiente código se muestra cómo se analizan todos los objetos de la escena. Si algún objeto tiene la etiqueta “soundtrack”, una variable también llamada `soundtrack`, que por defecto siempre es `false`, pasa a ser `true` y se asigna a la variable `audio` el atributo “src” del objeto, que en este caso es “`../assets/soundtracks/Spring_Village.ogg`”. Si `soundtrack` es `true`, se crea un objeto `Audio()` de HTML5 desde JavaScript, se le asigna el `src` de la variable `audio` y se empieza a reproducir. Como `reproducir.loop` es `true`, el audio estará en bucle todo el tiempo.

```
1 export function parseSoundtrack(sceneObjects){
2   var soundtrack = false;
3   for (var i = 0; i < sceneObjects.length; i++) {
4     if (sceneObjects[i]['tag'] == 'soundtrack') {
5       soundtrack = true;
6       var audio = sceneObjects[i]['attr']['src'];
7     }
8   }
9   if (soundtrack) {
10     var reproducir = new Audio();
11     reproducir.src = audio;
12     reproducir.loop = true;
13     reproducir.play();
14   }
15   return;
16 }
```

En la Figura 6.11, se puede ver un diagrama representando el análisis del fichero de configuración desde Websim y en función de si está definida la banda sonora, se reproduce un audio o no.

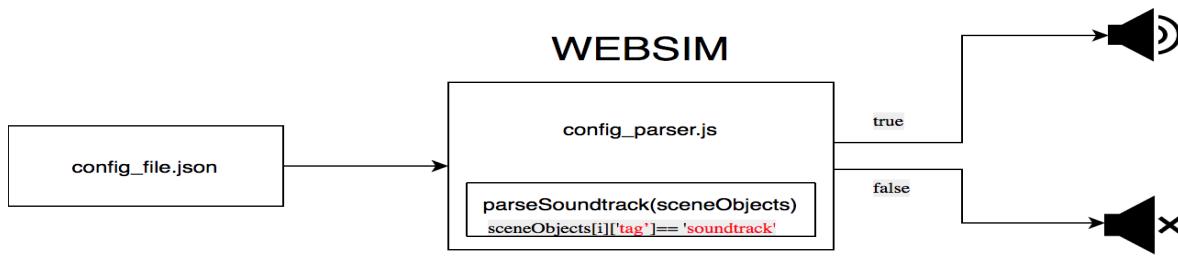


Figura 6.11: Diagrama audio de salida

En este video <sup>12</sup> se puede escuchar la banda sonora que se ha asignado a un ejercicio de ejemplo.

### 6.6.1. Efecto de sonido por colisión

Los efectos de sonido son muy importantes en los juegos, es por ello que se pensó mejorar la experiencia del usuario añadiendo el efecto de sonido de colisión.

Se implementó con el evento '*collide*' de A-Frame. Cuando se detecta una colisión este evento llama a una función que reproduce un audio y pone a 0 todas las velocidades V, W y L del robot. En este caso la reproducción del audio se hace una única vez por choque, como pasaría en la realidad.

```

1  document.addEventListener('collide', function (e) {
2      console.log('Robot has collided!');
3      var reproducir = new Audio();
4      reproducir.src= 'http://sonidosmp3gratis.com/sounds/000215403\_\_prev.mp3'; //collision
5      sound
6      reproducir.play();
7      localRobot.setV(0);
8      localRobot.setW(0);
9      localRobot.setL(0);
}) ;
  
```

Esta función sirve para todos los ejercicios de la plataforma, con o sin bandas sonoras, y para el teleoperador acústico. En el video de las soluciones <sup>13</sup>, en el escenario de la pared de piedra, se puede escuchar el efecto de sonido por colisión cuando el dron choca con la pared.

<sup>12</sup>[https://www.youtube.com/watch?v=c\\_BayBCSX4](https://www.youtube.com/watch?v=c_BayBCSX4)

<sup>13</sup><https://www.youtube.com/watch?v=85oTxgQQrck>

# Capítulo 7

## Conclusiones y trabajos futuros

Para finalizar, en este capítulo hablaremos de las conclusiones a las que se ha llegado en este proyecto, si se han cumplido o no los objetivos, los conocimientos adquiridos, posibles mejoras y futuros proyectos.

### 7.1. Conclusiones

El objetivo principal de este trabajo era introducir la *gamificación* en la plataforma Kibotics y se ha cumplido con éxito. Veámoslo analizando los subobjetivos propuestos y comprobando si se han cumplido o no:

- *Diseñar y desarrollar un nuevo ejercicio sobre una aspiradora robótica que tiene que limpiar una habitación.* Este subobjetivo se ha cumplido dado que se ha creado un robot capaz de aspirar pedazos de papel del suelo. Se ha programado un ejercicio en el que una aspiradora robótica tiene que limpiar una habitación llena de trozos de confeti y en la que hay obstáculos animados para aumentar la dificultad. También se ha programado un evaluador automático para que el usuario lo vea como un juego.
- *Diseñar y desarrollar un nuevo ejercicio sobre un robot que juega al pañuelo, recorriendo una línea, una lata que ejerce de pañuelo y regresando con ella al lugar de partida.* Este subobjetivo también se ha cumplido, se ha creado un nuevo escenario y un robot con pinzas capaz de coger objetos. Gracias a su diseño y las mallas de colisión de A-Frame, las pinzas son capaces de atrapar una lata y moverla por el circuito. El circuito es similar

## 7.1. CONCLUSIONES

---

a los que aparecen en las competiciones de robótica. Para la resolución de este juego el usuario tiene que usar varios sensores, el sensor de distancia para atrapar la lata y el sensor infrarrojos para detectar la línea negra del circuito. Con el evaluador automático desarrollado los usuarios pueden competir por obtener la máxima puntuación.

- *Diseñar y desarrollar un nuevo juego que analice el audio en tiempo real y explorar la posibilidad de añadir bandas sonoras a los ejercicios actuales.* Este subobjetivo se ha cumplido. Gracias a *Teachable Machine* tenemos un ejercicio de reconocimiento de audio, en el que el usuario puede dirigir mediante su voz al dron por los dos escenarios que se han creado. También se ha desarrollado la incorporación de bandas sonoras en los ejercicios ya disponibles en la plataforma y adicionalmente se ha añadido un efecto de sonido por colisión.

Los requisitos también se han satisfecho, los robots y juegos desarrollados son compatibles con la versión actual v.2.8 o superior de Kibotics. No se requieren instalaciones adicionales, el usuario entra a la plataforma Kibotics.org y todo se ejecuta desde su navegador. En este trabajo se ha usado el software de simulación Websim y A-Frame, cabe destacar el lenguaje JavaScript, el lenguaje de documentos JSON y el programa de modelado 3D Blender.

En conclusión, podemos decir que el objetivo principal de este trabajo se ha cumplido y los ejercicios del juego del pañuelo y el aspirador robótico están actualmente disponibles en la plataforma Kibotics, con ellos los alumnos de diferentes institutos están aprendiendo a programar.

A nivel personal este trabajo me ha dado la oportunidad de ampliar mucho mis conocimientos sobre tecnologías web (Django, HTML5, JavaScript y CSS3) y me ha entusiasmado conocer tecnologías como A-Frame para realidad virtual y Blender para modelado 3D, así como herramientas de reconocimiento de audio a través de la web con TensorFlow.JS y Teachable Machine. Con este proyecto también he aprendido a manejar GitHub para el desarrollo software en equipo y LaTeX con el que se ha escrito esta memoria.

También he tenido la oportunidad de dar un curso con Kibotics a un grupo de chicos usando el lenguaje Scratch. Desde mi experiencia, los comentarios hacia estos ejercicios orientados a juegos han sido muy positivos y he percibido un mayor interés por la robótica y programación.

## 7.2. Trabajos futuros

La *gamificación* es un mundo por explorar, se pueden crear miles de juegos con muchas temáticas diferentes. Estos son algunos de los futuros proyectos que se proponen relacionados con este trabajo:

- Mejorar el teleoperador acústico: El teleoperador acústico no está integrado aún en los cursos de Kibotics. El reconocimiento de audio es aceptable pero seguro que se puede mejorar con TensorFlowjs u otras tecnologías. También se puede mejorar creando un escenario más complejo y modelando una cueva o un castillo del que tiene que salir el dron u otros robots con las órdenes orales que le transmita el usuario.
- Mejorar juego del pañuelo: Se puede mejorar el juego del pañuelo añadiendo dos robots a la escena que compitan simultáneamente para ver quién es el primero en llegar a la meta con la lata. Este segundo robot podría ser un oponente automático con distintos niveles de dificultad proporcionado por Kibotics, o incluso el código de otro usuario de la plataforma, para permitir la competición directa.
- Juegos con reconocimiento de imágenes o de poses : *Teachable Machine* facilita el modelo de reconocimiento de audio pero también para imágenes y poses, se podría crear un modelo que basado en procesamiento de imágenes pueda guiar a un robot, por ejemplo dibujando una flecha e indicarle para qué lado quieras que se mueva. Con poses también se podría investigar cómo guiar al robot con los gestos que detecte el modelo e indicarle las órdenes al robot.
- Juegos multirobot en línea: Introducir más *gamificación* a la plataforma, una opción podría ser crear juegos en los que puedan competir directamente entre los usuarios y poder retransmitirlo por plataformas como Twitch o Youtube para que lo vean otros alumnos y organizar competiciones entre ellos.

# Bibliografía

- [1] “Kibotics.” <https://kibotics.org/> (accedida Abr. 18, 2021).
- [2] R. Barrientos Sotelo, Víctor Ricardo; García Sánchez, José Rafael; Silva Ortigoza, “Robots Móviles: Evolución y Estado del Arte,” Polibits, vol. 19, pp. 228–237, 2007, doi: 10.3233/978-1-60750-530-3-228.
- [3] “Abex.” <https://www.abexsl.es/es/sistema-robotico-da-vinci/que-es> (accedida Abr. 18, 2021).
- [4] “Los mejores KITS de ROBÓTICA y ROBOTS para NIÑOS por edades en 2020.” <https://revistaderobots.com/robotica-educativa/comprar-kits-de-robotica-y-robots-programables-para-ninos/> (accedida Abr. 18, 2021).
- [5] “mBot Explorer Kit,Makeblock ROBOTIX.” <https://www.robotix.es/es/mbot> (accedida Abr. 18, 2021).
- [6] “¿Qué es la World Wide Web (www) y cómo funciona?” <https://www.fotonostra.com/digital/paginasweb.htm> (accedida Abr. 18, 2021).
- [7] “Introducción a las tecnologías web · myTeachingURJC/2018-19-CSAAI Wiki.” <https://github.com/myTeachingURJC/2018-19-CSAAI/wiki/Introducción-a-las-tecnologías-web#tecnologías-en-el-lado-del-servidor> (accedida Abr. 18, 2021).
- [8] “Tecnologías y herramientas para el desarrollo web.” [http://cv.uoc.edu/annotation/a9c35c372dcee6e6b92afad6993cd048/620334/PID\\_00250214/PID\\_00250214.html](http://cv.uoc.edu/annotation/a9c35c372dcee6e6b92afad6993cd048/620334/PID_00250214/PID_00250214.html) (accedida Abr. 18, 2021).
- [9] “The Web framework for perfectionists with deadlines — Django.” <https://www.djangoproject.com/> (accedida Abr. 18, 2021).

## BIBLIOGRAFÍA

---

- [10] “Web Service Efficiency at Instagram with Python — by Instagram Engineering — Instagram Engineering.” <https://instagram-engineering.com/web-service-efficiency-at-instagram-with-python-4976d078e366> (accedida Abr. 18, 2021).
- [11] “Acerca — Node.js.” <https://nodejs.org/es/about/> (accedida Abr. 18, 2021).
- [12] “Top Companies That Use Node.JS in Production: Netflix, Trello, and Co.” <https://youteam.io/blog/top-companies-that-used-node-js-in-production/> (accedida Abr. 18, 2021).
- [13] “¿Qué es PHP? y ¿Para qué sirve? Un potente lenguaje de programación para crear páginas web. (CU00803B).” [https://www.aprenderaprogramar.com/index.php?option=com\\_content&view=article&id=492:ique-es-php-y-ipara-que-sirve-un-potente-lenguaje-de-programacion-para-crear-paginas-web-cu00803b&catid=70&Itemid=193](https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=492:ique-es-php-y-ipara-que-sirve-un-potente-lenguaje-de-programacion-para-crear-paginas-web-cu00803b&catid=70&Itemid=193) (accedida Abr. 18, 2021).
- [14] “Top 10 websites built with PHP technology - Facebook, Yahoo...” <https://cybercraftinc.com/blog/top-10-projects-developed-with-php-technology> (accedida Abr. 18, 2021).
- [15] “La importancia del contenido multimedia en educación.” <https://www.cursosfemxa.es/blog/14089-la-importancia-del-contenido-multimedia-en-educacion> (accedida Abr. 18, 2021).
- [16] E. Andrade & E. Chacón, “Implicaciones teóricas y procedimentales de la clase invertida,” Pulso, vol. 41, pp. 251–268, 2018.
- [17] “El Vídeo Educativo como recurso dinamizador del Aprendizaje - EVirtualplus.” <https://www.evirtualplus.com/video-educativo-como-recurso-aprendizaje/> (accedida Abr. 20, 2021).
- [18] “La Importancia Del Contenido Multimedia En La Educación.” <https://es.calameo.com/read/005984440640dded50f70> (accedida Abr. 18, 2021).
- [19] “Play Kahoot! - Enter game PIN here!” <https://kahoot.it/> (accedida Abr. 18, 2021).

## BIBLIOGRAFÍA

---

- [20] “Robótica educativa: ¿qué es y cuáles son sus ventajas?” <https://www.unir.net/educacion/revista/robotica-educativa/> (accedida Abr. 20, 2021).
- [21] “Scratch - Imagine, Program, Share.” <https://scratch.mit.edu/> (accedida Abr. 18, 2021).
- [22] E. P. Morales & M. F. G. Muñoz, “Manual Open Roberta.” [http://robomatrix.org/wp-content/uploads/2021/Manual\\_OpenRoberta.pdf](http://robomatrix.org/wp-content/uploads/2021/Manual_OpenRoberta.pdf) (accedida May 09, 2021).
- [23] “LEGO MINDSTORMS Education EV3 — LEGO® Education.” <https://www.robotix.es/es/lego-mindstorms-education-ev3> (accedida May 09, 2021).
- [24] “Recursos, Software y Actividades GRATIS — LEGO Education - ROBOTIX.” <https://www.robotix.es/es/descargar-software-lego-education> (accedida May 09, 2021)
- [25] “What Is mBlock 5?” <https://www.yuque.com/makeblock-help-center-en/mblock-5/overview> (accedida May 09, 2021).
- [26] “Competiciones de robótica educativa a los que apuntarse.” <https://blog.juguetronica.com/competiciones-de-robotica-educativa/#robocampeones> (accedida Abr. 18, 2021).
- [27] “RoboCupJunior – Creating a learning environment for today, fostering technological advancement for tomorrow.” <https://junior.robocup.org/> (accedida Abr. 20, 2021).
- [28] “EUROBOT JR.” <http://www.eurobot.es/index.php/eurobot-jr> (accedida Abr. 20, 2021).
- [29] “FIRST LEGO League.” <https://www.firstlegoleague.es/> (accedida Abr. 20, 2021).
- [30] “Vex Spain — Torneos.” <https://vexspain.com/vex-iq/torneos/> (accedida Abr. 20, 2021).
- [31] “Robocampeones” <http://robocampeones.org/> (accedida Abr. 20, 2021).
- [32] Imágenes de: “Pexels.” <https://www.pexels.com/es-es/> (accedida Abr. 18, 2021).
- [33] Imágenes de :“ Pixabay.” <https://pixabay.com/es/> (accedida Abr. 18, 2021).
- [34] “Programación para niños con vídeos de Scratch y Arduino.” <https://revistaderobots.com/robots-y-robotica/lenguajes-de-programacion-para-ninos-y-ninas/> (accedida Abr. 18, 2021).

## BIBLIOGRAFÍA

---

- [35] “Tecnologías para el desarrollo web más actuales — proun Madrid - Asturias.” <https://www.proun.es/blog/tecnologias-web-actuales/> (accedida Abr. 18, 2021).
- [36] “Conceptos básicos sobre tecnologías de desarrollo web - ingeniovirtual.com.” <https://www.ingeniovirtual.com/conceptos-basicos-sobre-tecnologias-de-desarrollo-web/> (accedida Abr. 18, 2021).
- [37] “Competencia docente de la robótica educativa: ¿una realidad o un nuevo reto para el profesorado? - Equipamiento para centros educativos.” <https://www.interempresas.net/Tecnologia-aulas/Articulos/156527-Competencia-docente-de-robotica-educativa-realidad-o-nuevo-reto-para-profesorado.html> (accedida Abr. 18, 2021).
- [38] “Así se enseña robótica y programación en las aulas españolas.” <https://www.educaciontrespuntocero.com/noticias/robotica-y-programacion-espana/> (accedida Abr. 18, 2021).
- [39] “Historia de robotica - el mundo de la robotica 604.” <https://sites.google.com/site/elmundodelarobotica604/historia-de-robotica> (accedida Abr. 18, 2021).
- [40] “Modelos de desarrollo de software” <https://www.elconspirador.com/2013/08/19/modelos-de-desarrollo-de-software/> (accedida Abr. 25, 2021).
- [41] J. Eguílez Pérez, “Introducción a JavaScript.” [www.librosweb.es](http://www.librosweb.es) (accedida Abr. 26, 2021)
- [42] “HTML: lenguaje de marcado de hipertexto — MDN.” <https://developer.mozilla.org/en-US/docs/Web/HTML> (accedida May 02, 2021).
- [43] “Qué es HTML5: Definición y funcionamiento — OpenWebinars.” <https://openwebinars.net/blog/que-es-html5/> (accedida May 02, 2021).
- [44] R. G. Duque, “Python PARA TODOS.” <http://mundogeek.net/tutorial-python/> (accedida May 03, 2021).
- [45] “Ace - The High Performance Code Editor for the Web.” <https://ace.c9.io/> (accedida May 03, 2021).

## BIBLIOGRAFÍA

---

- [46] “Blockly — Google Developers.” <https://developers.google.com/blockly> (accedida May 03, 2021).
- [47] “Trabajando con JSON Aprende sobre desarrollo web — MDN.” <https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects/JSON> (accedida May 03, 2021).
- [48] “JSON.” <https://desarrolloweb.com/home/json> (accedida May 03, 2021).
- [49] “TensorFlow Core — Aprendizaje automático para principiantes y expertos.” <https://www.tensorflow.org/overview?hl=es-419> (accedida May 03, 2021).
- [50] “Web Audio API - Referencia de la API Web — MDN.” [https://developer.mozilla.org/es/docs/Web/API/Web\\_Audio\\_API](https://developer.mozilla.org/es/docs/Web/API/Web_Audio_API) (accedida May 03, 2021).
- [51] “Teachable Machine.” <https://teachablemachine.withgoogle.com/> (accessed May 03, 2021).
- [52] “Introduction – A-Frame.” <https://aframe.io/docs/1.2.0/introduction/#getting-started> (accedida May 03, 2021).
- [53] “blender.org - Home of the Blender project - Free and Open 3D Creation Software.” <https://www.blender.org/> (accedida May 03, 2021).
- [54] “Vista de Navegadores web — El Tecnológico.” <https://revistas.utp.ac.pa/index.php/el-tecnologico/article/view/1287/html> (accedida May 03, 2021).
- [55] J.M. Cañas. “Robótica del siglo XXI y educación” <https://gsyc.urjc.es/jmplaza/activities/charla-prosegur2020.pdf> (accedida May 05, 2021 )
- [56] “Modelos de TensorFlow.js.” <https://www.tensorflow.org/js/models?hl=es-419> (accedida May 28, 2021).