UNIVERSIDADE DA CORUÑA

Escola Politécnica Superior

TRABAJO FIN DE GRADO

CURSO 2017/18

MEJORA DEL MODELO DE SIMULACIÓN DE UNA PLATAFORMA ROBÓTICA MÓVIL CONTROLADA MEDIANTE SMARTPHONE

Grado en Ingeniería en Tecnologías Industriales

ALUMNA

Vanesa Pérez Mc'Intosh

TUTORES

Francisco Javier Bellas Bouza

Martín Naya Varela

FECHA

DICIEMBRE 2017

1 TÍTULO Y RESUMEN

1.1 Título

Mejora del modelo de simulación de una plataforma robótica móvil controlada mediante Smartphone

1.2 Resumen

En el campo de la robótica, cada vez son más utilizados los modelos de simulación de robots mediante software. Esto permite por una parte, recrear un mayor número de escenas, realizar un mayor número de ensayos de un modo más rápido y económico. Por otra parte, el empleo de simuladores ayuda a reducir los riesgos y problemas derivados de trabajar con los dispositivos robóticos en entornos reales, minimizando los daños que se le puedan originar tanto al robot como a los elementos que se encuentren en su entorno.

En este Trabajo de Fin de Grado se tratará de mejorar el modelo de simulación existente del ROBOBO creado para la versión anterior. Dicho trabajo tiene por objeto comprender mejor el funcionamiento real de una parte de los elementos sensores y motores que lo integran, así como mejorar el modelo de simulación realizado en una versión anterior del mismo.

Título

Mellora do modelo de simulación dunha plataforma robótica móbil controlada mediante Smartphone.

Resumo

No campo da robótica, cada vez son máis empregados os modelos de simulación de robots mediante software. Esto permite, por unha parte, recrear un maior número de escenas e realizar un maior número de ensaios dun modo máis rápido e económico. Por outra parte, o emprego de simuladores axuda a reducir os riscos e problemas derivados de traballar cos dispositivos robóticos en entornos reais, minimizando os danos que se lle poidan orixinar tanto ao robot como aos elementos que se atopen no seu entorno.

Neste Traballo de Fin de Grao tratarase de mellorar o modelo de simulación existente do ROBOBO creado para a versión anterior. Dito traballo ten como obxecto comprender mellor o funcionamento real dunha parte dos elementos sensores e motores que o integran, así como mellorar o modelo de simulación realizado nunha versión anterior do mesmo.

Títle

Development of the simulation model of the infrared sensors and the motors of a mobile robotic platform.

Summary

In the field of robotics, the use of simulation models for robots in software is becoming more frequent. On the one hand, this leads to create a greater number of scenes, to perform a bigger number of tests in a more economic and fast way. On the other hand, the use of simulators helps to reduce the number of risks and problems caused by working with robotic devices on real environments, decreasing the damages that can be caused both, for the robot, and its surroundings.

On this project, we will try to improve the simulation model of a previous version of ROBOBO. The goal of this work is to get a better understanding of the real behavior of part of the integrated sensor and motors, as well as to upgrade the simulation model that already exists.

UNIVERSIDADE DA CORUÑA

Escola Politécnica Superior

TRABAJO FIN DE GRADO

CURSO 2017/18

MEJORA DEL MODELO DE SIMULACIÓN DE UNA PLATAFORMA ROBÓTICA MÓVIL CONTROLADA MEDIANTE SMARTPHONE

Grado en Ingeniería en Tecnologías Industriales

MEMORIA

2 ÍNDICE

3 INTRODUCCIÓN

En este capítulo se expondrá el campo en el que se enmarca este Trabajo Fin de Grado, partiendo desde una perspectiva global hasta concretar su posición en el ámbito de la robótica autónoma.

Empezando por la definición de esta disciplina, la robótica es una ciencia o rama de la tecnología que estudia el diseño y construcción de máquinas capaces de desempeñar tareas repetitivas, o en las que se necesita de una alta precisión, así como aquellas que resultan peligrosas para el ser humano o irrealizables sin intervención de una máquina.

La palabra robot encuentra su origen en la obra teatral del autor checo Karel Capek titulada *R.U.R (Robots Universales Rossum),* estrenada en el año 1921 en Praga. El término, creado por el hermano de Karel, Josef, se deriva de la palabra checa 'robota' que significa 'trabajo' o 'labor' y se aplica habitualmente al 'trabajo duro'. Históricamente, la robota era un periodo de trabajo de 6 meses que el siervo debía ofrecer a su señor. Será el escritor Isaac Asimov (1920-1992) el encargado de acuñar el término en 1942 al publicar la historia de ciencia ficción *Runaround* donde se dan a conocer las Tres Leyes de la Robótica:

1. Un robot no hará daño a un ser humano o, por inacción, permitir que un ser humano sufra daño.
2. Un robot debe hacer o realizar las órdenes dadas por los seres humanos, excepto si estas órdenes entrasen en conflicto con la 1ª Ley.
3. Un robot debe proteger su propia existencia en la medida en que esta protección no entre en conflicto con la 1ª o la 2ª Ley.

Así, la *Real Academia Española* define a un robot como una máquina automática programable capaz de realizar determinadas tareas de forma autónoma. Un robot se compone principalmente de tres partes: sensores, sistema de control y actuadores.

Los sensores son los dispositivos que le permiten al robot conocer su entorno, estos pueden ser, por ejemplo, sensores de presión, infrarrojos, cámaras, sensores de ultrasonido, etc. Estos sensores reciben variables físicas y las transforman en señales eléctricas que envían al sistema de control. Éste interpreta las señales recibidas y según su programación, enviará órdenes a los actuadores necesarios para realizar una acción deseada. Los actuadores son

los mecanismos que le permiten al robot comunicarse con el medio, o interferir en él, pueden ser motores, altavoces, pantallas, LEDs, etc.

Los robots tal y como los conocemos hoy en día fueron desarrollados después de la Segunda Guerra Mundial debido a la creciente demanda de automatización en la industria del automóvil. Cabe señalar que antes los robots no eran más que herramientas para la automatización. Estaban programados a realizar una tarea específica: transportar, cargar, descargar, pintar, etc. Actualmente existen los llamados robots inteligentes que además de detectar las modificaciones en el entorno y actuar en consecuencia, poseen memoria para archivar el resultado de sus acciones. De este modo, cuando se le presenta una tarea, el propio robot inspecciona su memoria y puede aprender de su experiencia. Aprende cómo lograr mejor su rendimiento y eficiencia.

Si clasificamos los robots en función de su ámbito de aplicación, a día de hoy, la robótica se divide en dos grandes áreas, la robótica industrial y la de servicios. La *Organización Internacional de Estándares (ISO)* define al robot industrial como un manipulador multifuncional reprogramable con varios grados de libertada, capaz de manipular materias, piezas, herramientas o dispositivos especiales según trayectorias variables programadas para realizar tareas diversas (Figura 1). Están altamente implantados en el sector industrial debido a las numerosas ventajas que ofrecen, entre las que se encuentran el aumento de la productividad y de la calidad, la mayor flexibilidad y la reducción de riesgos para los seres humanos.

Figura 1. Ejemplos de robots industriales**.**

Según la *Federación Internacional de Robótica* (IFR), un robot de servicio es un robot que opera de forma parcial o totalmente autónoma, para realizar servicios útiles para el bienestar de los humanos y del equipamiento, excluyendo operaciones de manufactura. Son empleados en numerosos aplicaciones, entre las que se encuentran las de tipo militar (exploración de zonas remotas, rescate, manejo de explosivos, etc.), sanitarias (sistemas de cirugía, cuidado de personas enfermas y/o mayores), domésticas (limpieza, preparación de comida), trabajos peligrosos (construcción, limpieza industrial, seguridad), educativas y de entretenimiento.

De entre los diferentes ámbitos de aplicación mencionados, este trabajo se centrará en los robots pertenecientes a los dos últimos, educación y entretenimiento. Estos han dado lugar a la aparición de un nuevo término *'edutainment'* (education and entertainment, en inglés). El *edutainment* es una combinación de métodos y tipos de formación que combinan la presentación de la información educativa con elementos de entretenimiento. Su objetivo es motivar al alumno en el proceso de aprendizaje para aumentar la cantidad de información retenida. Básicamente, gracias al *edutainment*, jóvenes y niños principalmente, pueden desarrollar conocimientos y habilidades en diferentes áreas, tales como la informática, la electrónica o la mecánica, mediante aprendizaje basado en proyectos. Por lo tanto, robots con propósitos educacionales son muy aplicados en escuelas e institutos. En esta línea cabe destacar la línea de robots educativos de LEGO Mindstorms[1] (ejemplo en la Figura 2).

Figura 2. Robot de LEGO Mindstorms.

Un gran número de los robots desarrollados en investigación se han creado para el uso dentro de este campo. En investigación se apuesta por una configuración del mismo con alta sensorización (sensores infrarrojos, sensores de ultrasonido…) para favorecer la obtención

de información, y diversas formas de actuación, como los LEDs o los motores, a la vez que se intenta conseguir un precio asequible, una fácil programación y un tamaño reducido, facilitando así su adquisición y manipulación. Otra característica de estos robots es que son autónomos, es decir, tienen capacidad de autogobierno y adaptación ante cambios del entorno. El robot sobre el que trata este Trabajo de Fin de Grado se encuentra en este grupo.

Se podría considerar que el *AIBO 'Artificial Intelligence Robot'*[2] (Figura 3a) es el primer robot en ser utilizado tanto en el ámbito del entretenimiento como en el de la investigación. Se trata de una serie emblemática de mascotas robóticas diseñadas y fabricadas por Sony. El prototipo fue anunciado a mediados de 1998, aunque el primer modelo de consumo se introdujo en 1999. Los AIBO fueron comercializados para su uso doméstico como 'Robots de Entretenimiento'. También fueron ampliamente adoptados por las universidades con fines educativos para investigaciones de robótica y para la interacción humano-robot puesto que integra un computador, un sistema de visión y actuadores en un paquete bastante más barato que los robots de investigación convencionales. Las mejoras en programación, sensores y actuadores han hecho que este robot-mascota haya ido evolucionando anualmente, consiguiendo dotarle, gracias al software AIBOLife, de una personalidad, la capacidad de caminar, de ver su entorno a través de la cámara y de reconocer comandos de voz (en inglés, en español o japonés), consiguiendo así realizar acciones y movimientos similares a los de mascotas reales. Son precisamente sus altas prestaciones en cuanto a sensorización y actuación, junto a su elevado precio para tratarse de un juguete, los motivos que lo acabaron relegando más a investigación que a entretenimiento. En enero de 2006, Sony anunció que iba a descontinuar AIBO y varios otros productos en un esfuerzo para hacer rentable la empresa. Este año han anunciado un nuevo relanzamiento del AIBO para el año 2018 que incorporará inteligencia artificial y conexión WIFI (Figura 3b)

Figura 3a. AIBO 1999

Figura 3b. AIBO 2018

Otra categoría dentro del campo de investigación y entretenimiento es la correspondiente a los robots humanoides, los cuales tienen una morfología que se asemeja a la humana. Debido a la mayor complejidad de este tipo de robots, los avances iniciales en estos han sido más lentos, aunque en la actualidad existe un gran número de robots humanoides dotados de una capacidad de aprendizaje que les permite mejor control y autonomía. Ejemplo de esto son los robots ASIMO[3] (Figura 4a) de Honda, creado principalmente para investigación y concebido para funcionar en un entorno humano real, siendo capaz de interactuar con las personas haciéndoles la vida más fácil. Para conseguir los movimientos de ASIMO, Honda ha estudiado y utilizado como modelos los movimientos coordinados y complejos del cuerpo humano. Este robot tiene una altura de 120 centímetros, es capaz de maniobrar entre los objetos en una habitación, y puede desplazarse por escaleras. Alcanza los interruptores de luz y picaportes, y puede realizar tareas en meas y superficies de trabajo. Es fácil de manejar y se puede mover sin problemas dentro de cualquier ambiente cotidiano.El robot NAO[4] (Figura 4b) de la compañía Aldebaran Robotics. Nació con la misma filosofía de ASIMO pero es más pequeño y económico de fabricar. Se ha utilizado para la enseñanza de habilidades sociales en niños autistas y como 'profesor' de robótica, entre otras tareas.

Figura 4a. Robot ASIMO.

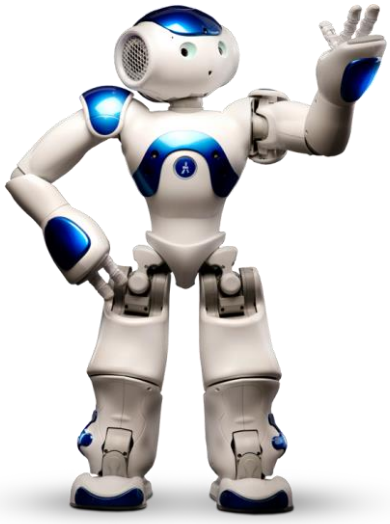


Figura 4b. Robot NAO**.**

Dentro de los robots humanoides cabe hacer una mención especial a Sophia[5] (Figura 5) creada por la compañía con sede en Hong Kong Hanson Robotics, es el robot de Inteligencia Artificial más avanzado actualmente. Ha sido diseñada para aprender y adaptarse al comportamiento humano, y trabajar con personas. En octubre de 2017, se convirtió en una ciudadana saudí, siendo el primer robot con ciudadanía de un país.

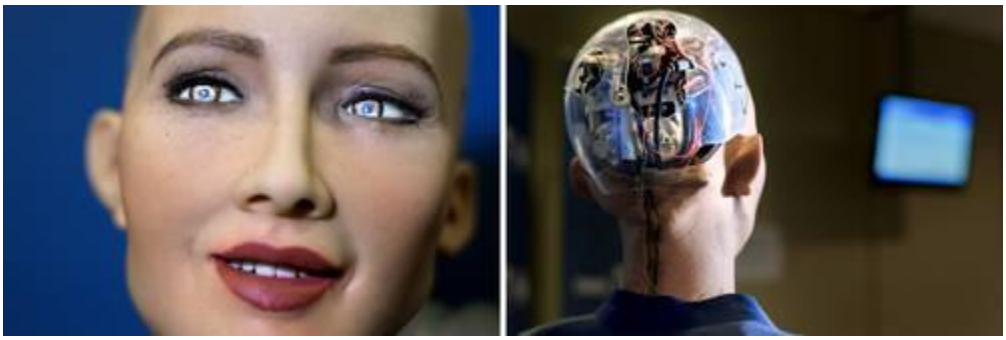


Figura 5. Robot Sophia..

Los robots anteriores han sido citados debido a su importancia en el campo de la investigación, aunque existen otros tipos de robots también importantes en este campo que se asemejan más al robot en el que se centra este trabajo. Se trata de robots con ruedas que carecen de la complejidad de movimiento de las articulaciones de los robots humanoides, como el THYMIO[6] (Figura 6) de la institución *École Polytechnique Féderale de Lausanne (EPFL)*. Es un robot autónomo utilizado en investigación y docencia, que consta de un gran número de sensores y actuadores, sencillo mecánicamente hablando, versátil y con un entorno de desarrollo abierto y compatible con los lenguajes de programación estándar. Además dispone de una carcasa a la cual se le pueden acoplar distintos accesorios LEGO.

Figura 6. Robot THYMIO.

No se debe olvidar al robot KEPHERA[7], desarrollado por la misma institución que el anterior, pues es uno de los más utilizados en experimentos de robótica. La última versión KEPHERA IV (Figura 7), sensores de seguimiento de líneas, un acelerómetro, ocho sensores de proximidad, un giroscopio, micrófonos y altavoces, cámara, distintas formas de conectividad, etc.

Figura 7. Robot KEPHERA IV.

El problema de estos tipos de robots es que necesitan una inversión considerable para su adquisición. Un bajo coste suele ser necesario en investigación, ya que el entorno de las inversiones no suele ser directo, es decir, los inversores tienden a interesarse en los proyectos una vez que empiezan a aparecer resultados positivos; y tampoco está asegurado un beneficio económico, además de que puede necesitarse un alto número de robots trabajando a la vez, como en el caso de la robótica colaborativa, en la que diferentes robots comunicados entre sí actúan juntos para lograr diferentes objetivos. Además, cuentan con sensores de baja calidad y procesadores con poca capacidad de cómputo.

3.1 Plataformas robóticas con Smartphone.

Para solucionar algunos de los problemas presentados por los robots anteriores surgió la idea de diseñar plataformas robóticas que puedan alojar y transportar un Smartphone (teléfono inteligente). La ventaja de esta solución es el constante avance de los Smartphone y la robótica, que permite ahorrar dinero en las actualizaciones del robot, ya que lo que se renovaría sería el teléfono móvil. El Smartphone aporta capacidad de cómputo, sensorización (cámara y giroscopio), actuación (sonido, llamadas, iluminación, etc) y conectividad inalámbrica. A continuación se mostrarán los ejemplos más representativos de estos robots.

En 2012, Romotive lanzó una campaña para poner en marcha un proyecto de robótica orientado a la educación infantil. El resultado es Romo[8] (Figura 8a), un robot personal para niños que utiliza un iPhone o un iPad como "cerebro". Se trata básicamente de un juguete orientado a niños a partir de los 8 años. Las aplicaciones para iOS permiten aprender el funcionamiento del robot y sirven para crear las bases de una programación de tareas básicas. Sin embargo, se puede considerar el menos completo de los robots que vamos a mencionar en este apartado puesto que posee un gran número de carencias. Por un lado, está el gran

inconveniente de que únicamente sea compatible con el sistema iOS, ya que no permite elegir un Smartphone distinto de un iPhone. Además, no puede ser empleado en investigación, puesto que carece de un software de programación de bajo nivel, y no soporta el protocolo de red del principal sistema operativo de la robótica actual de investigación y docencia, ROS[9]. Por último, otra gran desventaja es el hecho de que la plataforma carece de cualquier tipo de sensor, limitando al robot únicamente a los que posea el iPhone.

Otro ejemplo de este tipo de plataformas robóticas es el SMARTBOT[10] (Figura 8b) que puede ser controlada por cualquier Smartphone o incluso una placa de Arduino pero que tampoco soporta el protocolo de red de ROS ni tiene ningún sensor.

Figura 8a. Robot ROMO

Figura 8b. Robot SMARTBOT.

Solucionando el problema de la compatibilidad con el protocolo de red de ROS, aparecen las plataformas robóticas ODDWERX[11] (Figura 9a) de OLogic, lo que las hace más versátiles y las dota de mayor facilidad de cómputo. Son utilizadas como juguete, como mascota virtual, para aplicaciones educacionales y para aplicaciones de interacción con el entorno. ODDWERX también carece de sensores en la base por lo que no dispone de autonomía en el movimiento.

Será la plataforma WHEELPHONE[12] (Figura 9b), de Gctronic, la que cubra todas estas necesidades, contando con sensores de proximidad además de las características mencionadas anteriormente. Sus carencias serían que no existe un movimiento independiente del Smartphone respecto a la base, carece de LEDs que permitan interactuar con el usuario e informarle del estado de la batería, por ejemplo, y la conexión entre la plataforma y el Smartphone varía entre Android y Smartphone, puesto que se realiza mediante cable.

Figura 9a. Robot ODDWERX **Figura 9b. Robot WHEELPHONE**

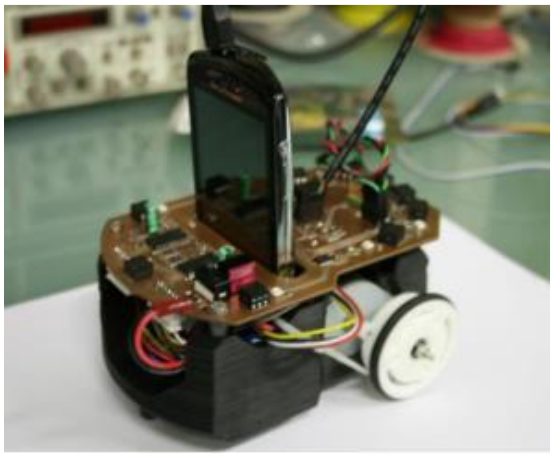
Desde el año 2013, el Grupo Integrado de Ingeniería de la UDC ha venido desarrollando una plataforma robótica para Smartphone que cubriese las carencias que se detectaban en los existentes en el mercado. Esto ha dado lugar al denominado ROBOBO que posee como como principales características:

- Tanto la carcasa como sus componentes no electrónicos pueden fabricarse de manera totalmente artesanal, mediante la utilización de una impresora 3D, algo totalmente revolucionario.
- Cuenta con una base móvil sensorizada con un entorno de programación integrado en el sistema operativo ROS, con la posibilidad de utilizar todas sus librerías.
- Consta con un alto número de sensores, gran capacidad de cómputo y comunicación (gracias al Smartphone).

Los puntos mencionados anteriormente se cumplían en la versión 1.0 del ROBOBO (Figura 10a) aunque este necesitaba algunas mejoras, es así como surgió el ROBOBO 2.0 (Figura 10b). Algunas de las mejoras incorporadas fueron:

- Cambio de motores por unos más pequeños y con un consumo de batería menor.
- Aumento del número de LEDs, utilizando 9 de ellos como una matriz para informar del estado del robot.
- Utilización del bluetooth para comunicar al Smartphone con la base.
- Modificaciones en la carcasa para hacerlo más atractivo visualmente a la hora de comercializar.

Hoy en día se está trabajando con el ROBOBO 2.0.2 (Figura 10c), que difiere del 2.0 principalmente en sus dimensiones y la nueva forma de la carcasa que le permite eliminar un sensor de la parte trasera.



(a)

(b)

(c)

Figura 10: (a) ROBOBO 1.0, (b) ROBOBO 2.0, (c) ROBOBO 2.0.2

Puesto que el ROBOBO está orientado a su uso en educación e investigación, resultaría especialmente útil la existencia de un modelo de simulación del mismo que permita predecir el comportamiento del robot real ante aplicaciones creadas por el usuario, sin necesidad de tener el robot presente. Esto permitiría, por ejemplo, que todos los alumnos de una clase pudieran trabajar al mismo tiempo en una misma tarea, aunque solo tuvieran un ROBOBO para toda la clase.

Para poder realizar el modelo de simulación del ROBOBO será necesario conocer su geometría, el peso de los diferentes componentes y el comportamiento de los infrarrojos y los motores del ROBOBO. Tratará de extraerse un modelo matemático que se ajuste lo más posible a este comportamiento para adaptar, así, los datos obtenidos al entorno de simulación. Precisamente será este el objetivo de este Trabajo de Fin de Grado.

4 OBJETIVOS

4.1 Objetivo principal

Este trabajo de fin de grado tiene como objetivo la realización de un modelo de simulación de los sensores infrarrojos y motores de una plataforma robótica móvil controlada por Smartphone.

4.2 Sub-objetivos

Para llevar a cabo dicho objetivo se han planteado los siguientes sub-objetivos:

- Determinar el comportamiento de los sensores infrarrojos en el entorno real. Esto nos permitirá implementar un modelo de simulación del funcionamiento de los sensores infrarrojos similar al real.
- Caracterización y modelado de los sensores infrarrojos en V-REP. Se trasladarán los datos obtenidos sobre el comportamiento de los sensores infrarrojos reales al modelo en V-REP.
- Determinar el comportamiento real de los motores de corriente continua de la plataforma robótica, con controlador de posición o sin él. Gracias a esto se podrá realizar un modelo de simulación del funcionamiento de los motores parecido al real.
- Caracterización y modelado de los motores de corriente continua en V-REP. Nos permitirá conseguir que la plataforma robótica se desplace de forma similar a como lo haría en la realidad.
- Pruebas y ensayos para validar el modelo en simulación con el modelo real. Con esto podremos afinar el modelo de simulación de modo que sea lo más parecido posible al robot real.

5 ANTECEDENTES

La evolución que ha experimentado la Robótica durante los últimos años dificulta cada vez más la evaluación de los robots en situaciones reales para probar su funcionamiento y continuar con su desarrollo, debido al aumento de la complejidad de los mismos.

El uso de plataformas de simulación de robots permite representar virtualmente el robot y su entorno de manera que este se evalúe de forma más simple a la que se haría con el modelo físico. Es posible recrear múltiples escenas, ensayos, experimentos, etc., donde además se incluyan las propiedades físicas que rigen el entorno. La creación de un modelo de simulación cuenta con numerosas ventajas:

- No se produce la manipulación física de los robots, lo que implicaría costes tanto de adquisición de los mismos como de reparación de posibles daños causados en sus componentes.
- Se evita el riego que constituye para los propios robots el utilizar el modelo físico durante la realización de pruebas en las que podría caerse, colisionar con otros objetos o incluso con otras partes de sí mismo.
- No existe peligro para las personas, o el entorno, en el caso de robots industriales de gran tamaño que podrían provocar daños graves en caso de un mal funcionamiento.

Resulta muy importante conocer el comportamiento de los sensores y actuadores de un robot para realizar un correcto modelo de simulación. Los simuladores ofrecen la capacidad de representar los sensores con los que se recogen datos del entorno de simulación, siendo los más comunes los sensores de distancia. Por su parte, los actuadores, como pueden ser los motores, precisan ser representados para simular las acciones del robot. Con esto es posible que el modelo de simulación interactúe con su entorno, ambos fácilmente modificables, ofreciendo la posibilidad de comparar diferentes morfologías de ambos. Además, se puede comprobar fácilmente los distintos comportamientos del robot antes escenarios diferentes. Esto constituye una de las ventajas principales que ofrece la simulación, ya que facilita detectar posibles problemas que puedan surgir en los diferentes casos, permitiendo sugerir estrategias para solucionarlos antes de que se produzcan en el robot físico.

Se debe tener en cuenta que existen limitaciones en la precisión con la que se pueden representar ciertas características del robot y de su entorno, y que por lo tanto el

comportamiento del modelo de simulación no va a ser exacto. Lo que se debe valorar es si el grado de similitud entre la realidad y la simulación es aceptable.

Muchos de los robots mencionados en el capítulo de introducción cuentan con un modelo de simulación, por ejemplo, ASIMO cuenta con una versión en Webots.

En el caso del ROBOBO, el modelo de simulación de su versión 2.0 (Figura 11) ha sido previamente realizado en el Trabajo de Fin de Grado de Cristina Castro Sequeiro titulado *Modelado del diseño y comportamiento de una plataforma robótica móvil controlada mediante Smartphone* .

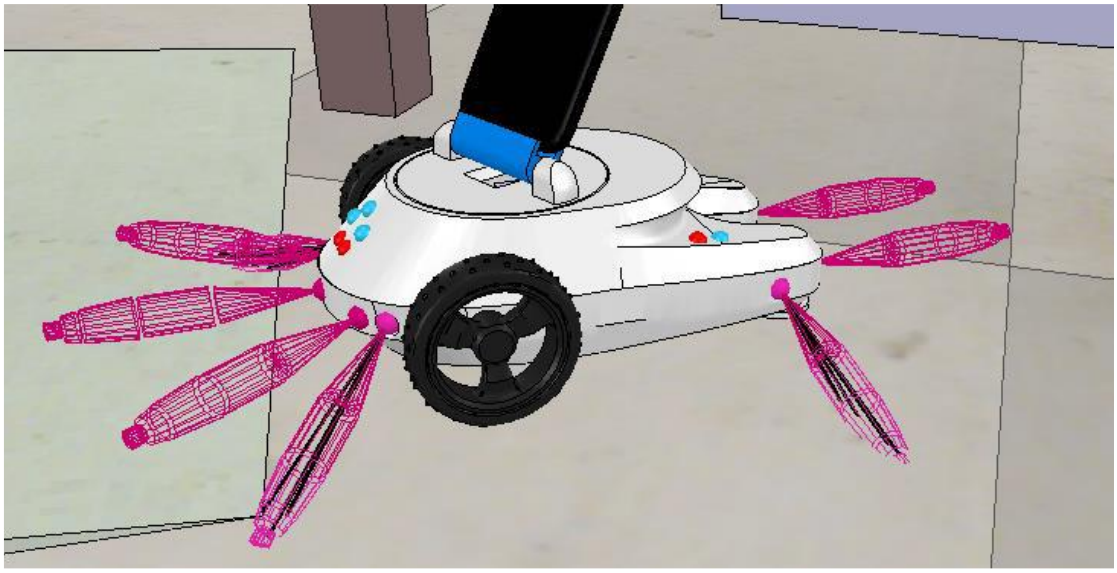


Figura 11. Modelo de simulación del ROBOBO 2.0

La construcción del modelo tridimensional del modelo de simulación del ROBOBO 2.0 es muy similar al realizado en el apartado 7 de este trabajo de fin de grado. No obstante, el empleo del modelo de simulación ha mostrado que existen ciertos aspectos del mismo que necesitan ser mejorados. Estas mejoras afectan tanto a los sensores infrarrojos como a los motores.

En cuanto a los sensores infrarrojos, este modelo posee dos carencias principales. Por un lado, el tamaño de los sensores está muy lejos del tamaño de los sensores reales, llegando a no detectar objetos a distancias o en posiciones que sí que detectaría el robot real. Además, no consta de un método de obtención del valor de intensidad de los infrarrojos, es decir, sólo indica si se está detectando un objeto o no y a qué distancia se haya, pero no puede calcular el valor que nos devolverían los sensores en Scratch (explicado en el capítulo 6 de este trabajo).

Pasando a los motores, la velocidad de los mismos se ha definido por medio de un script que permite que su regulación empleando controles visualizados en la pantalla durante la

simulación (Figura 12) lo que resulta muy cómodo para el manejo del modelo. Sin embargo, el problema vuelve a estar una vez más en que la relación existente entre el ROBOBO real y el Scratch no se ha tenido en cuenta. Cuando el usuario envía la orden a los motores para que se muevan mediante Scratch, lo hace mediante valores del 0% al 100% de la máxima que puede dar el motor, sin tener por qué conocer la velocidad real del mismo. Esto es así, para que los niños que puedan tener acceso al ROBOBO lo entiendan más fácilmente, puesto que es más sencillo entender la velocidad mediante un porcentaje que, por ejemplo, en metros/segundo.

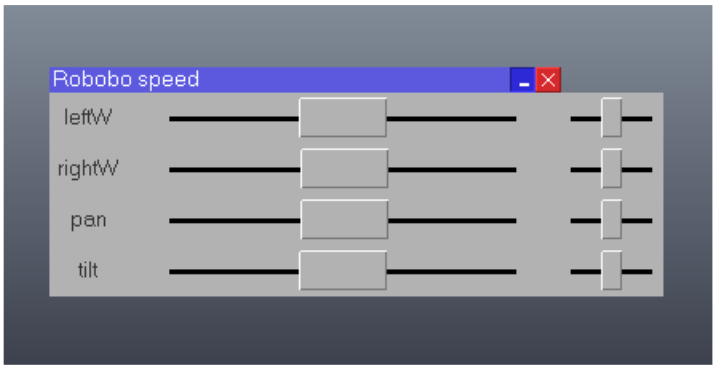


Figura 12. Controles de velocidad del modelo de simulación del ROBOBO 2.0.

El presente Trabajo de Fin de Grado ha surgido de la necesidad de mejorar las debilidades del modelo de simulación previo expuestas anteriormente.

6 FUNDAMENTOS TEÓRICOS

En este capítulo se va a hablar del funcionamiento teórico de parte de los componentes del ROBOBO que vamos a modelar, los sensores infrarrojos y los motores, así como de nociones básicas de los programas y plataformas empleadas en el proceso.

6.1 ROBOBO

La realización del modelado de un robot en un software de simulación robótica para probarlo virtualmente y solucionar posibles fallos de diseño antes de construirlo físicamente para evitar gastos de dinero innecesarios. En el caso de este proyecto, la realización del modelo de simulación se hace tomando los datos del modelo real ya existente con el propósito de facilitar la introducción del ROBOBO en colegios, permitiendo que los alumnos puedan probar sus aplicaciones en el modelo del robot sin necesidad de conectarse al ROBOBO físico, ya que es altamente probable que el aula carezca de un robot por alumno, agilizando así el desarrollo de las clases.

Después de la breve introducción sobre el ROBOBO que se ha hecho en los capítulos anteriores, se pasará a hablar de los elementos en cuyo modelado nos vamos a centrar en este trabajo.

6.1.1 Sensor infrarrojo.

Un sensor es un dispositivo capaz de detectar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. Un sensor de proximidad infrarrojo (comúnmente llamado sensor infrarrojo) es un dispositivo que puede detectar objetos que se encuentran en su entorno de visión, transformar la distancia a la que se encuentra el objeto en variables eléctricas y enviarlas a otros dispositivos.

El sensor infrarrojo está formado, básicamente, por un diodo LED emisor y un diodo LED receptor. El diodo emisor es un emisor de rayos infrarrojos, radiación electromagnética situada en el intervalo que va desde la luz visible a las ondas microondas. El diodo receptor es un fotodetector que trabaja como un transistor clásico, pero la base está reemplazada por un cristal fotosensible que al recibir la luz, produce una corriente y desbloquea el transistor. En el fototransistor la corriente circula sólo en un sentido y el bloqueo del transistor depende de la luz: cuanta más luz haya, más corriente se conducirá.

El ROBOBO consta de 8 sensores infrarrojos VCNL4040 de VISHAY (Figura 13).

Figura 13. Sensor infrarrojo VCNL4040**.**

Estos sensores tienen un alcance de hasta 20 cm. La luz emitida por el emisor de infrarrojos se refleja en un objeto y el fotodiodo recibe la luz que se refleja en el objeto y la convierte en corriente. La intensidad de la luz reflejada va a depender de diferentes parámetros:

- La distancia a la que se encuentre el objeto detectado del sensor, cuanto menor sea ésta, mayor será la intensidad.
- La reflectancia del objeto detectado. A mayor reflectancia, mayor será la intensidad del rayo reflejado. Esta propiedad varia tanto con el material del propio objeto como con el color de su superficie. Los materiales que mejor reflejan la luz son la madera y el plástico, mientras que los que peor lo hacen son el cristal transparente y el metal lacado. En cuanto a los colores, materiales más claros serán más fáciles de detectar por el sensor que los oscuros. Esta información la hemos obtenido del Trabajo de Fin de Grado de Alba Lema Martínez llamado *Diseño y adaptación de un sistema de sensorización infrarroja en una plataforma robótica móvil.*
- Ángulo de apertura del sensor. Se ha comprobado que para dos objetos de distinto tamaño pero del mismo material y color, el valor devuelto por el sensor no es siempre el mismo. Esto se debe al ángulo de apertura de los sensores (Figura 14).

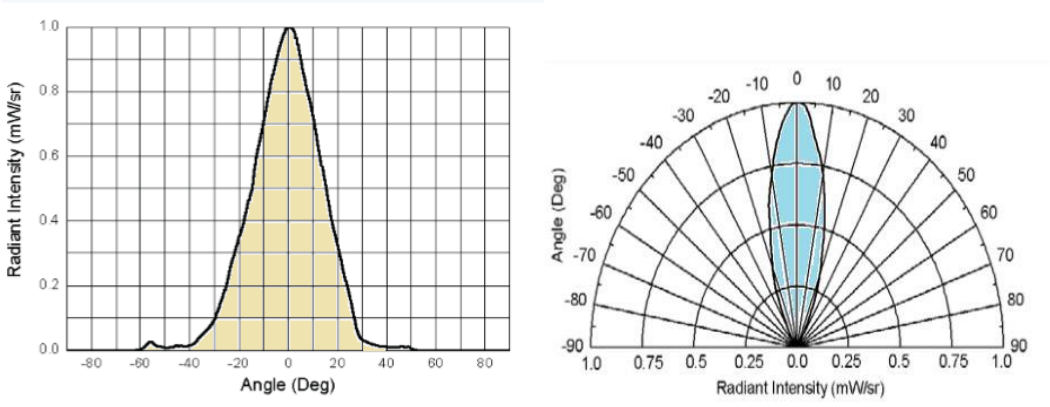


Figura 14. Perfiles IRED

Este ángulo de apertura hace que a ciertas distancias un objeto pequeño no sea capaz de abarcar todo el haz infrarrojo, devolviendo únicamente una parte, obteniendo valores de intensidad menores. No sólo sería así con objetos pequeños, sino que pasaría lo mismo con un objeto lo suficientemente grande como para abarcar el haz pero que no se encuentra posicionado correctamente.

La disposición de los sensores infrarrojos en el ROBOBO se indica en la figura 15.

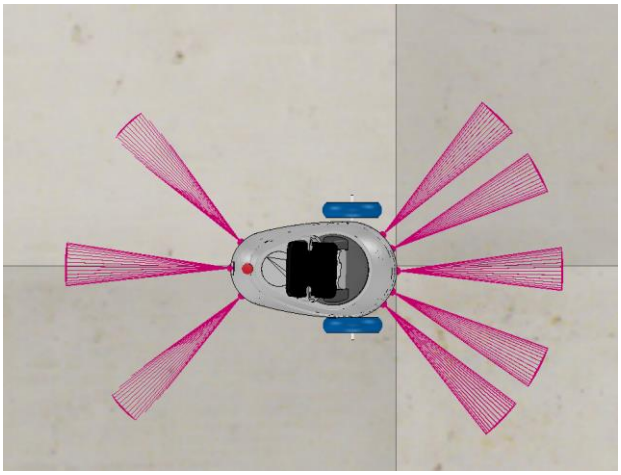


Figura 15. Disposición sensores infrarrojos en el ROBOBO

6.1.2 Sistema motriz

El motor que se utiliza es un motorreductor de corriente continua con eje extendido con escobillas como conmutador. En general estos motores están pensado para trabajar a 6V, sin embargo, pueden funcionar bien en el rango de 3V-9V. El giro comienza en menos de 0.5V pero no posee el suficiente torque como para ser utilizado de forma correcta. Con tensiones superiores a los 9V se reduciría la vida útil del motor.

El ROBOBO cuenta con dos motores encargados del desplazamiento de la plataforma, uno situado en cada rueda. Estos motores (Figura a) disponen de reducciones de 150:1, un encoder y un sistema de engranajes piñón-corona para la transmisión del movimiento a la base rotatoria. Un encoder es un dispositivo electromecánico cuya función es convertir un movimiento mecánico en una señal eléctrica. En este caso se encarga de traducir el movimiento giratorio del motor, actuando de interfaz entre este y el controlador. La reducción 150:1 permite aumentar el par máximo disponible y la aceleración del robot. Poseen un torque en vacío de 0.9 kg-cm. En la parte trasera del robot hay un deslizador que permite el movimiento omnidireccional del robot.

Adicionalmente, cuenta con otros dos motores que permiten el movimiento del Smartphone con respecto a la carcasa:

- Motor PAN (Figura 16a): permite la rotación de la plataforma del Smartphone con respecto al ROBOBO en su eje vertical. Se trata también un motor con reducción 150:1 con encoder y diseño piñón-corona. Esta reducción le da la posibilidad de girar de forma suave y con movimientos precisos.
- Motor TILT (Figura 16b): permite al Smartphone inclinarse hacia delante y atrás mediante la rotación de su soporte. Este motor cuenta con una reductora 1000:1 con encoder. La reductora es tan grande por dos motivos principales. Por un lado, permite que el móvil se sostenga sin necesidad de aplicar corriente. Por otro lado, proporciona el torque necesario para mover el Smartphone incluso en las peores condiciones. El torque en vacío es de 5kg-cm.

(a) **(b)**

Figura 16: Motor 100:1 en ruedas y motor PAN (a) y 1000:1 en motor TILT (b)

6.2 Simulador V-REP

El simulador V-REP[13] (Virtual Robot Experimentation Platform) es un simulador 3D compatible con Windows, Mac y Linux. Es gratuito y de código abierto mientras no se use para fines comerciales. Cuenta con numerosos tutoriales en la página web que permiten al usuario familiarizarse con el programa así como con un manual en la que se explican todos los aspectos necesarios para utilizarlo correctamente. Permite modelar un sistema entero o únicamente ciertos componente como sensores, mecanismos, etc, de modo que se pueda controlar cada elemento de forma individual.

Al iniciar el programa por primera vez se crea una escena vacía (Figura 17), en la que se verán los diferentes componentes de los robots u objetos que forman parte del entorno al ser creados. Las escenas son la parte principal de la simulación y en ellas se incorporarán todos los elementos necesarios para el modelado. Tal y como se ve en la figura 17, la ventana en la que podremos ver nuestra escena se encuentra en la parte de la derecha de la interfaz de usuario.

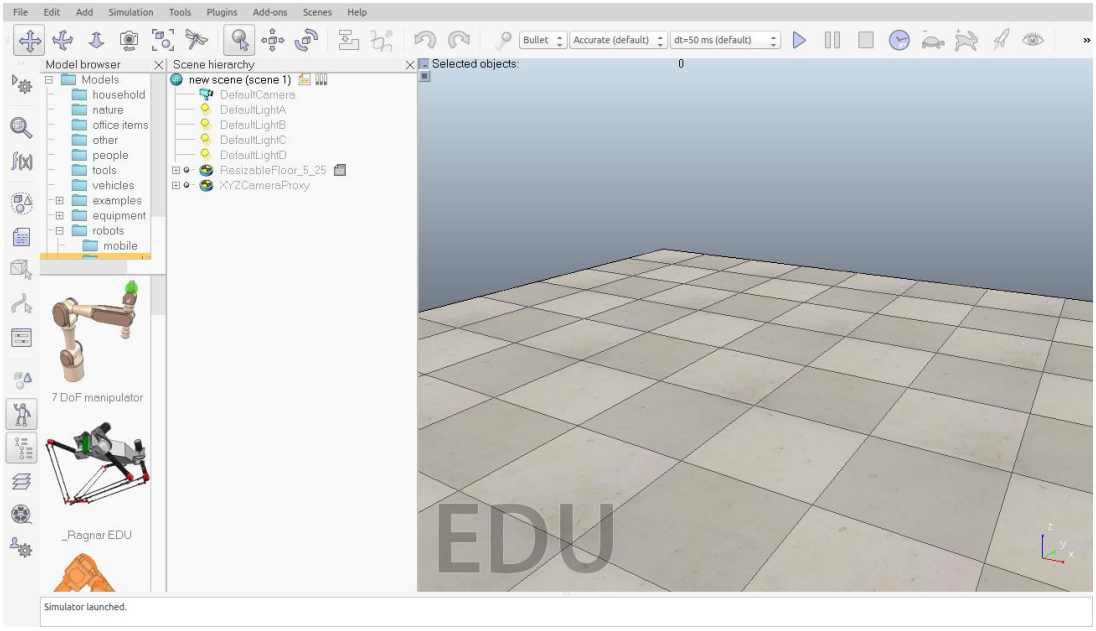


Figura 17. Interfaz de usuario de V-REP.

En la interfaz pueden ver también las barras de menú y de herramientas (Figura 18). Gracias a la primera se puede acceder a casi la totalidad de funcionalidades del simulador, activándose desde ella diferentes cuadros de diálogo. En cuanto a la barra de herramientas, en ella se encuentran las principales funciones de control de la escena, desde posicionamiento de objetos, hasta el control de las simulaciones.

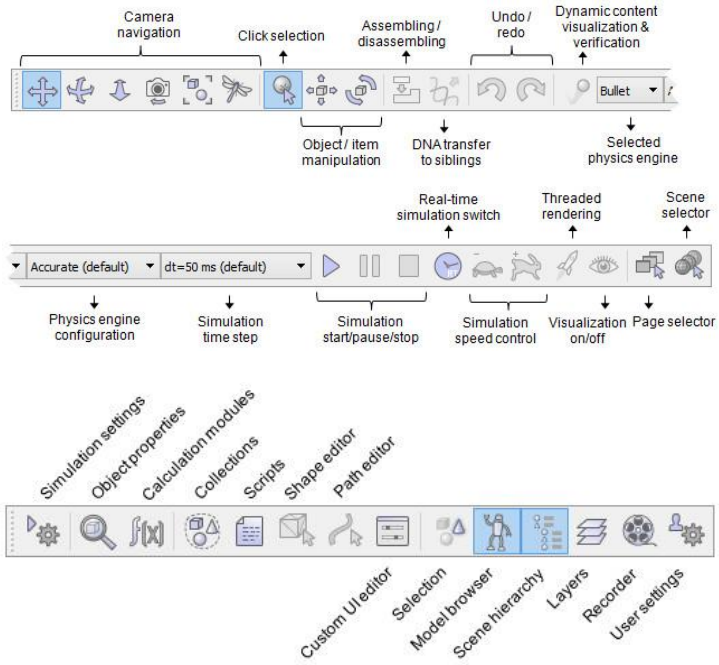


Figura 18 . Barra de herramientas.

Otro elemento importante de la interfaz de usuario es la ventana que nos muestra la jerarquía de la escena (Figura 19) que aparece a la izquierda de la interfaz de usuario. En ella aparece todo el contenido de nuestra escena. Como los objetos que forman parte de ella están construidos con una estructura jerárquica, esta ventana nos la muestra de forma detallada.

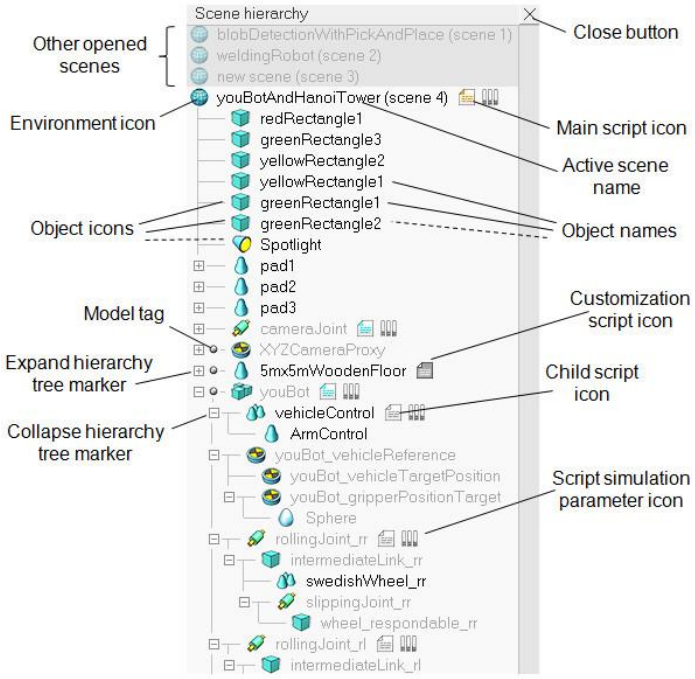


Figura 19. Ventana de Jerarquía de una escena.

En V-REP se puede crear un robot a partir de formas puras (cilindro, ortoedro, esfera...) que son las formas más sencillas que se pueden utilizar, o bien, se puede importar un modelo 3D en formatos que soporten mallas triangulares, como son STL, OBJ, 3DS, etc. El simulador posee algunos modelos de elementos reales que se pueden incorporar a la simulación (Figura 20) como son sensores de proximidad, cámaras, motores… Existe también la posibilidad de incorporar a la escena robots y objetos habituales que ya están construidos en V-REP como son los robots manipuladores, robots móviles (como el NAO mencionado anteriormente). Estos modelos también pueden ser modificados y adaptados en cualquier simulación.

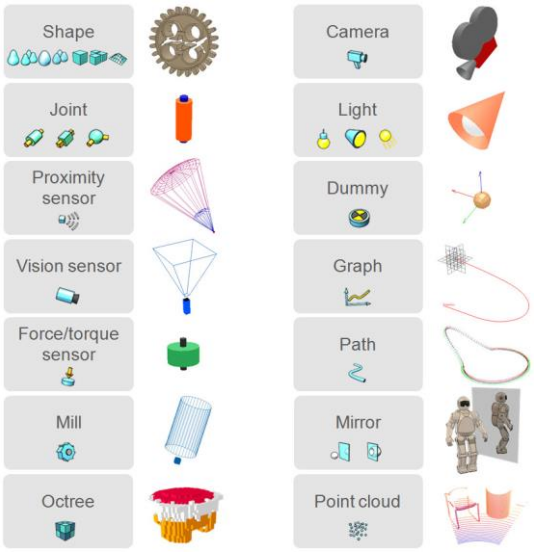


Figura 20. Objetos de V-REP y su representación**.**

Es posible controlar la escena o el modelo en V-REP de dos formas diferentes: desde el propio V-REP o de forma externa.

La primera opción se lleva a cabo mediante scripts de V-REP llamados 'embedded scripts'. Cada escena creada en el simulador tiene acoplado un script llamado 'script principal' sin el cual la simulación no podría ejecutarse. Cada objeto de la simulación puede llevar asociado un script denominado 'child script' que se suele emplearse para controlar un modelo, El script principal se encarga de controlar los child scripts llamándolos durante la simulación. El lenguaje de programación empleado por ellos es el Lua, que es un lenguaje de extensión, es decir, que no puede ser ejecutado por sí mismo, sino que debe ser incluido en un programa que lo utilice.

De forma externa, existe la posibilidad de hacer uso de plugins, ROS nodes o APIs remotas para controlar el modelo y comprobar diferentes comportamientos del mismo. Hay seis lenguajes de programación que se pueden utilizar: C/C++, Phyton, Java, Matlab, Octave y Urbi.

6.2.1 API externa

Una API es un conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software. Las siglas API vienen del inglés *Application Programming Interface*, es decir, Interfaz de Programación de Aplicaciones en español.

V-REP ofrece una API externa permitiendo el control de la simulación o del propio simulador desde una aplicación externa o un hardware externo. Las APIs externas de V-REP están compuestas por aproximadamente cien funciones específicas y una función genérica que

puede ser llamada desde una aplicación C/C++, un script de Python, una aplicación Java, un programa de Matlab/Octave o un script de Urbi o Lua.

Las APIs externas que empleamos en este Trabajo de Fin de Grado son scripts de Phyton, creados en Pycharm[14]. Pycharm es un IDE (Entorno de Desarrollo Integrado) especializado en el lenguaje Python. Ha sido desarrollado por la compañía checa JetBrains y su primera versión fue sacada en el año 2010. Es multiplataforma ya que es compatible con Windows, Linux y Mac OS X. Consta de dos licencias, una profesional y una comunitaria, que difieren entre ellas en algunas de sus características, precio y condiciones de uso.

Se ha empleado, además, el paquete de librerías de Anaconda[15]. Anaconda es una distribución de Python multiplataforma, desarrollada por Continuum Analytics. Contiene una gran colección de paquetes y librerías de análisis de datos, computación científica e ingeniería. Nos permite instalar paquetes y gestionar entornos virtuales fácilmente, además de tener otras ventajes:

- Ahorro de tiempo al instalar paquetes adicionales requeridos.
- Posee un administrador de paquetes y gestor de entornos virtuales,
- No necesita permisos de administrador.
- Es multiplataforma (Windows, Mac y Linux).
- Es completamente gratuita, incluso para uso comercial y redistribución,

6.3 Scratch

Scratch[16] es un lenguaje de programación visual desarrollado por el MIT Media Lab. Es utilizado por estudiantes, académicos, profesores y padres para crear fácilmente animaciones, juegos, interacciones, etc. Permite que los alumnos desarrollen las habilidades mentales básicas necesarias para programar (pensamiento computacional) sin necesidad de que sepan ningún otro lenguaje de programación.

Desde 2013, Scratch 2 está disponible en línea, de forma completamente gratuita, como aplicación de escritorio para Windows, OS X y Linux.

Cuando entramos en el entorno web de Scratch nos aparece una pantalla como la siguiente (Figura 21).

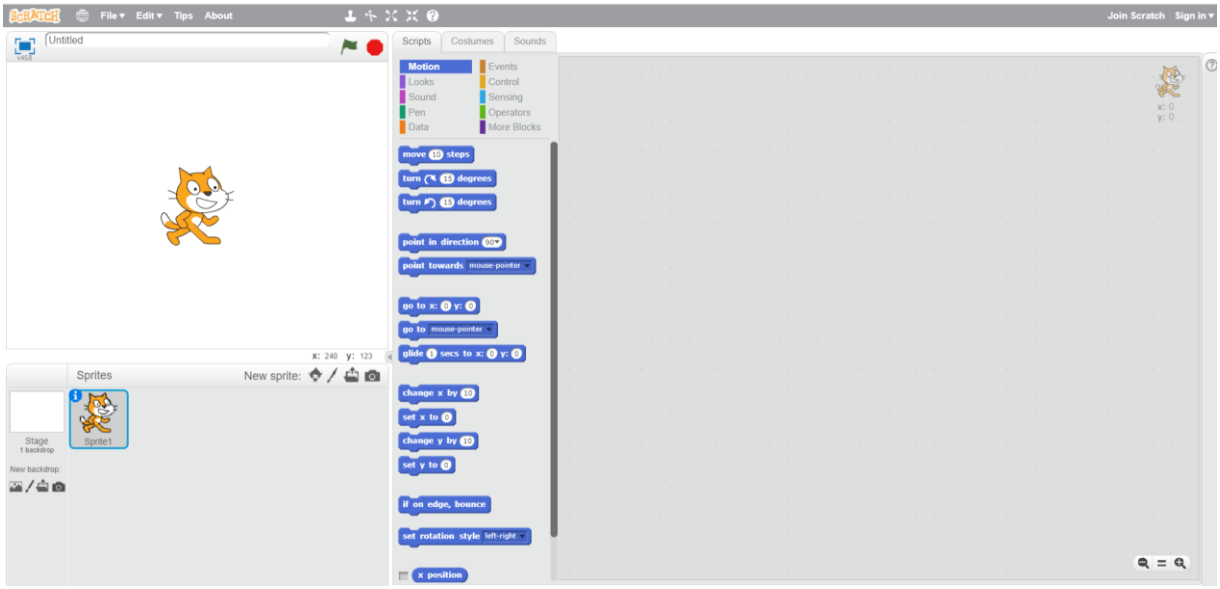


Figura 21. Interfaz de usuario de Scratch.

De izquierda a derecha, en la zona superior izquierda de la pantalla hay un escenario en el que se nos muestran los resultados del proyecto actual y todas las miniaturas de los 'sprites' (objetos en la versión en castellano) que aparecen listados en la zona inferior. El área de la derecha es la zona de programación, a donde se arrastrarán los bloques de la zona central para crear así los problemas. Los bloques de instrucciones se agrupan por colores en diferentes categorías, tal y como se aprecia en la Figura 21, éstas son:

- Movimiento (azul oscuro): mueve objetos y cambia ángulos.
- Apariencia (violeta): controla el aspecto visual del objeto, añade bocadillos de habla o pensamiento, cambia el fondo…
- Sonido (lila): reproduce ficheros de audio y secuencias programables.
- Lápiz (verde oscuro): control del ancho, color e intensidad del lápiz.
- Datos (naranja): creación de variables y listas.
- Eventos (marrón): contiene manejadores de eventos, se sitúan al principio de cada grupo de instrucciones.
- Control (ámbar): sentencias condicionales, bucles, detener…
- Sensores (azul claro): los objetos pueden interactuar con el ambiente creado por el usuario.
- Operadores (verde claro): operadores matemáticos, generador aleatorio de números, sentencias 'and' y 'or'…
- Más bloques (púrpura): control de bloques y dispositivos externos.

En nuestro caso emplearemos ScratchX para manejar el ROBOBO y poder obtener los datos necesarios para la realización de este trabajo. ScratchX permite la utilización de extensiones experimentales que no están avaladas por el equipo de Scratch y que por lo tanto no están disponibles en la web principal. El ROBOBO posee su propia plantilla de ScratchX que permite al usuario conectarse con él y controlar tanto el Smartphone como la base.

6.4 Red neuronal artificial

Las Redes Neuronales son un campo muy importante de la Inteligencia Artificial, inspirándose en el comportamiento de las neuronas humanas y sus conexiones para tratar de crear modelos artificiales que solucionen problemas difíciles de resolver empleando técnicas algorítmicas convencionales. Permiten extraer patrones y detectar tramas que son muy difíciles de apreciar por humanos u otras técnicas computacionales.

Se va a considerar una neurona como un módulo o unidad básica de la red que recibe información de otros módulos o del entorno, la integra, la computa y emite una única salida que se va a transmitir idéntica a múltiples neuronas posteriores.

En una red de neuronas existe un valor numérico, denominado peso o fuerza sináptica, que pondera las señales que reciben en su entrada. Este indica la fuerza de conexión que existe entre dos neuronas, cuanto mayor sea el peso, mayor será la influencia de la primera neurona sobre la segunda. Cuando se evalúa una neurona debe calcularse el conjunto de todas las fuerzas o valores que se reciben por sus entradas. Una vez obtenido ese valor, se aplica una función de activación que determinará el valor del estado interno de la neurona y que será lo que se trasmita a la salida.

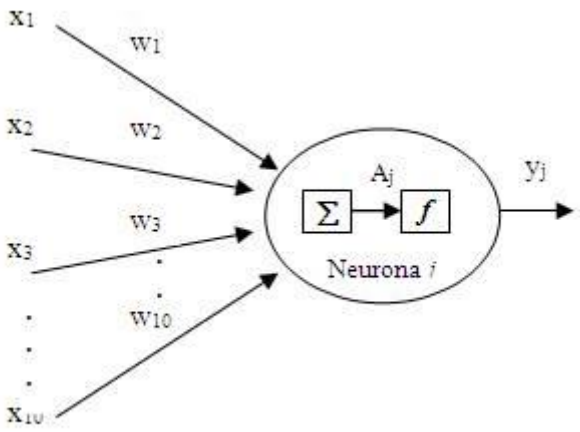


Figura 22. Neurona artificial.

6.4.1 Redes alimentadas hacia delante

Las redes neuronales alimentadas hacia delante o 'feedforward' (Figura 23), son aquellas en las que información se mueve en un único sentido, es decir, de la entrada a la salida. Estas redes se estructuran en capas. Cada capa agrupa un conjunto de neuronas que reciben sinapsis de la capa anterior y emiten salidas a las neuronas de la capa siguiente.

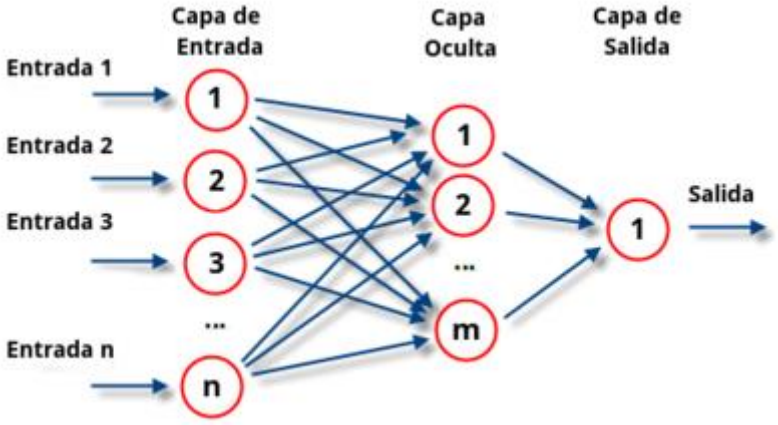


Figura 23. Red neuronal artificial alimentada hacia delante.

En este tipo de redes existe al menos una capa de entrada, formada por las neuronas que reciben las entradas del entorno; y una capa de salida, formada por una o más neuronas que emiten la respuesta de la red al exterior. Entre la capa de entrada y la de salida, se encuentran una o más capas intermedias u ocultas.

6.4.2 Redes con retroalimentación total o parcial.

En este tipo de redes, los elementos pueden enviar estímulos a neuronas de capas anteriores, de su propia capa o incluso a sí mismos (Figura 24). Cada neurona puede estar conectada a todas las demás, por ello, cuando se recibe información de entradas a la red, cada neurona necesitará calcular y recalcular su estado varias veces, hasta que se estabilice el sistema. Un estado estable es aquel en el que la salida del sistema se mantiene constante. Debido al tiempo que le lleva estabilizarse al sistema, este tipo de redes son más lentas que las anteriores, aunque su funcionamiento se asemeje más al del cerebro humano en donde los fenómenos de retroalimentación son fundamentales.

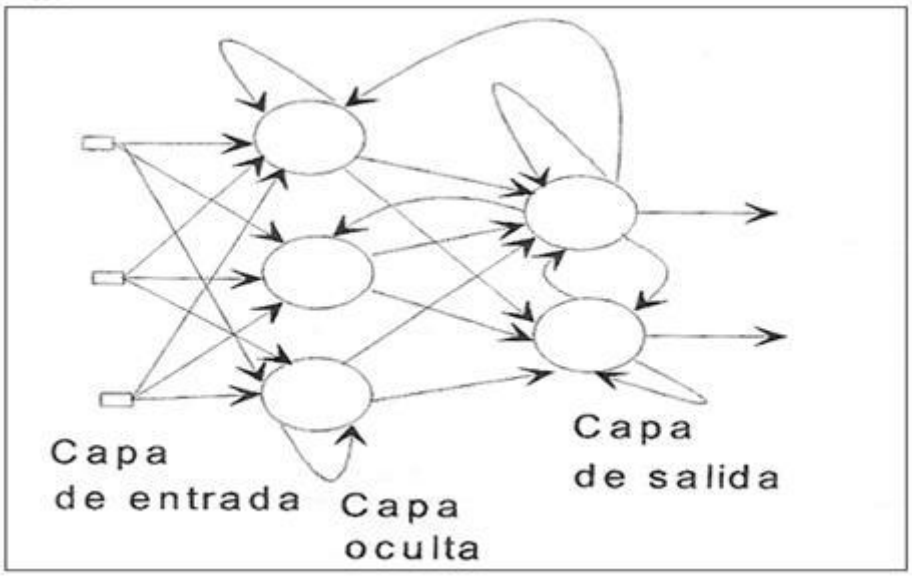


Figura 24. Red neuronal con retroalimentaciaón.

6.4.3 Aprendizaje de una red neuronal artificial

La capacidad de aprendizaje es la característica más importante de una red neuronal. Durante el aprendizaje o entrenamiento de la red se van ajustando internamente y de forma adecuada todos y cada uno de los pesos asociados a cada conexión entre las diferentes neuronas para conseguir la salida deseada. Para lograrlo, se le aportan a la red un conjunto de entradas hasta que la red pueda responder por sí sola a situaciones diferentes a las aprendidas. En casos sencillos los pesos podrían ajustarse manualmente, pero lo usual es utilizar algún tipo de algoritmo que lleve a cabo el proceso de entrenamiento.

La principal ventaja que presentan las redes neuronales es que, una red neuronal correctamente entrenada y ajustada, es capaz de trabajar con información incompleta o difícil de predecir, siempre y cuando esta entrada sea razonablemente parecida a aquellas con las que ha sido entrenada.

Hay múltiples algoritmos para realizar el entrenamiento de las redes neuronales y todos pueden ser englobados en dos categorías:

- Aprendizaje supervisado. En este tipo de entrenamiento se presentan dos vectores, uno de entrada y otro de salida deseada. La salida que obtiene la red se compara con la deseada y se modifican los pesos de la red con la intención de reducir el error cometido. Este proceso se repite iterativamente hasta que la diferencia entre la salida deseada y la computada sea aceptablemente pequeña.
- Aprendizaje no supervisado. Este caso de entrenamiento es el más parecido al funcionamiento del cerebro humano. En él no se emplea un vector con la salida

esperada y sólo hay vectores de entrada en el conjunto de entrenamiento. El algoritmo se encarga de modificar los pesos hasta que todas las salidas sean consistente, es decir, que a entradas muy parecidas, la red obtenga la misma salida.

7 DESARROLLO DEL MODELO DE SIMULACIÓN

En este capítulo se procederá con la explicación de la construcción del modelo de simulación del ROBOBO. Esto se hará dividiendo el desarrollo en varias partes.

7.1 Modelo tridimensional

En este apartado se procede a explicar de forma detallada la adaptación de las diferentes piezas que forman el robot al entorno de simulación para que los cálculos que realice el simulador sean lo más rápidos posibles y conseguir una simulación estable. Esto es necesario puesto que, a pesar de que ya existe un modelo de simulación previo para el ROBOBO 2.0, la versión 2.0.2 presenta cambios en la forma de su carcasa.

Para realizar esta primera tarea, se comienza creando una nueva escena en V-REP. Este simulador nos permite crear modelos robóticos importando las piezas que lo componen desde otros programas. En este caso, se dispone de un diseño del ROBOBO creado en Solid Works, del que se obtienen las piezas en formato .STL de forma independiente.

Cuando importamos la pieza, aparece una ventana que da la opción de determinar la escala del objeto importado. Se elige la opción de milímetros en todas las piezas para que se mantengan las proporciones de Solid Works.

De todas las piezas de las que consta la estructura 3D del ROBOBO, sólo vamos a importar únicamente aquellas que nos vayan a resultar útiles en el modelo de simulación. Esto es, no se importarán piezas internas que no se puedan visualizar desde el exterior y que no vayan a tener una función dinámica en el modelo. Esto se hace para reducir la complejidad de la puesta en marcha de la simulación, al disminuir la complejidad de los cálculos que se tendrán que realizar, evitando así que se ralentice y se vuelva inestable.

Cuando se importan objetos 3D desde otros programas, lo más importante es que la malla que lo forma no contenga un gran cantidad de triángulos (V-REP es compatible con formatos que describen los objetos mediante mallas de caras triangulares). En el manual de usuario de V-REP se recomienda que en total contengan entre 10000 y 20000.

Además, en V-REP existe la posibilidad de simplificar las piezas creando formas convexas a partir de ellas. Esto reduce en gran medida la densidad de triángulos que las forman, pero también provoca que los objetos pierdan definición, lo cual deteriora bastante el aspecto del

robot. Por ello, se crearán dos componentes de cada pieza: la componente visual y la dinámica. Esta última será la que esté enlazada con los motores para moverse como las partes reales del robot, representando su peso y reaccionando a colisiones con otros objetos. Es esta componente la que será empleada por el simulador para realizar los cálculos pertinentes durante la simulación debido a su mayor sencillez, haciendo así el proceso más rápido. Las componentes visuales estarán acopladas a las anteriores pero son empleadas únicamente para mejorar el aspecto del robot y para que sea detectable, por ejemplo, por los sensores de otro robot.

A continuación se nombrarán las piezas que constituyen el modelo y se mostrarán las diferentes componentes de cada una.

- **Carcasa.** Es el objeto base, a ella se acoplan el resto de los componentes.

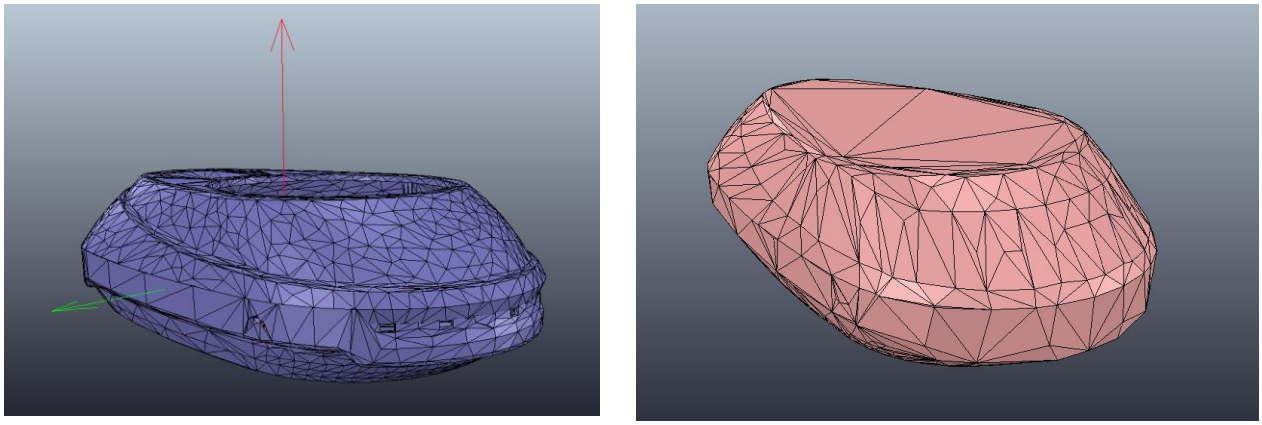


Figura 25. Carcasa. Componente visual y componente dinámica.

- **Ruedas:** Una situada a cado lado de la carcasa. Debido a que su forma es sencilla, se ha elegido que su componente dinámica esté formada por un cilindro en lugar de optar por su descomposición.

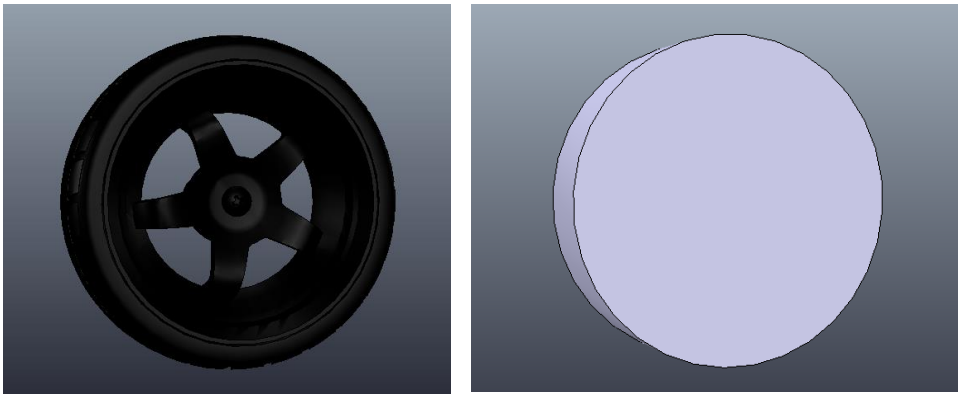


Figura 26. Componentes visual y dinámicas de las ruedas.

- **Base rotatoria.** Está situada en la parte superior de la carcasa y sobre ella se halla el soporte del Smartphone.

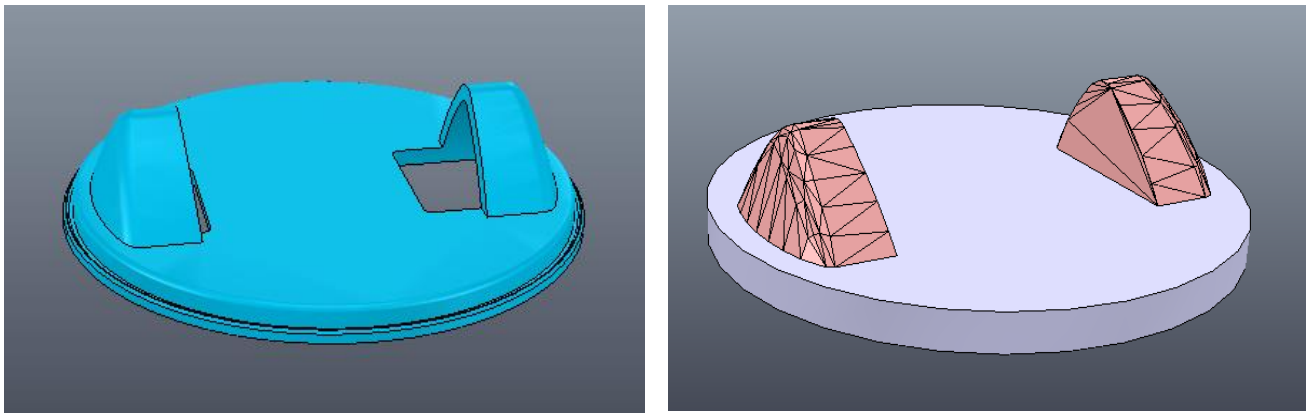


Figura 27. Base rotatoria. Componente visual y componente dinámica.

- **Soporte Smartphone.** Permite la inclinación del Smartphone sobre la base rotatoria. En este caso se ha decidido simplificar el soporte como un cilindro al que se le ha acoplado el teléfono móvil, tal y como se muestra.

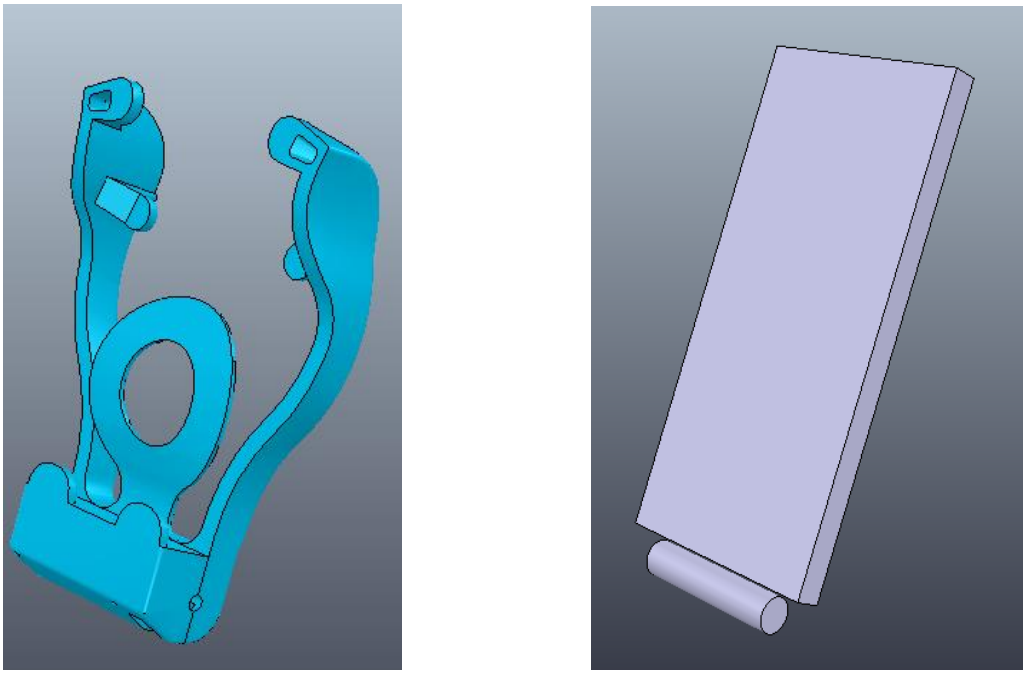


Figura 28. Soporte Smartphone. Componente visual y componente dinámica.

- **Motores y batería.** Carecen de componente visual y únicamente se crea su componente dinámica para representar su peso. Se han empleado ortoedros para representarlos. Estos irán situados en el interior de la carcasa en la posición aproximada de los objetos que representan.

- **Deslizadera.** También carece de componente visual, puesto que se ha acoplado a la carcasa. En cuanto a la componente dinámica, se empleará una pequeña esfera
- **Cámaras del Smartphone.** Son las cámaras frontal y trasera del mismo. El entorno de simulación cuenta con ventanas que permiten ver lo que enfocan las cámaras.

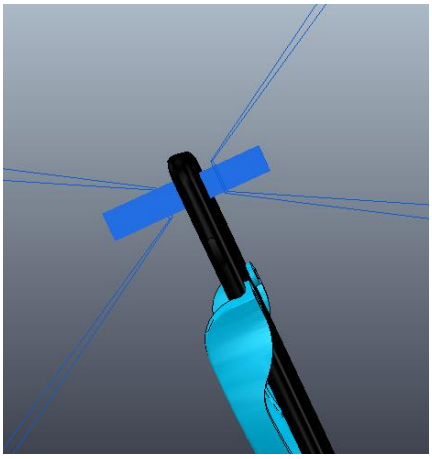


Figura 29. Cámaras frontal y trasera del Smartphone.

V-REP sitúa cada pieza importada en el origen de coordenadas, por lo que va a ser necesario tomar medidas en el modelo de Solid Works para conocer las posiciones relativas de cada una con respecto a la carcasa. A medida que se añaden las piezas a la escena, se colocarán en su posición correspondiente, ocupando el mismo lugar las dos componentes de cada una.

7.2 Composición del modelo dinámico

V-REP permite la simulación dinámica de tres tipos de objetos: *joints, shapes* y force *sensors.*

7.2.1 Joints

Los *joints* son los objetos encargados de representar a los motores. Pueden tener forma de cilindro, prisma o esfera en función del tipo de movimientos que posean:

- Cilindros. Describen movimientos rotatorios con un grado de libertad.
- Prismas. Describen movimientos de traslación con un grado de libertad.
- Cilindro mixto. Describen tanto movimientos de traslación como de rotación, de manera similar a un tornillo. Al igual que los anteriores, cuenta con un grado de libertada.
- Esfera. Describen movimientos de rotación con tres grados de libertad.

En el caso de este robot, se han escogido los siguientes tipos de *joints:*

- **Motores de las ruedas.** Al buscarse la rotación entorno a un eje se emplearán joints de tipo cilíndrico centrados en las mismas.

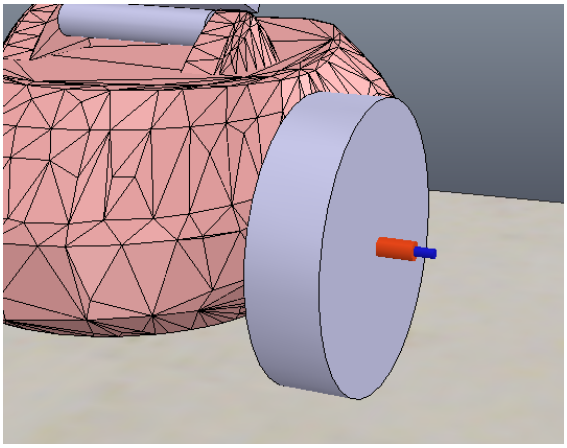


Figura 30. Joint ruedas.

- **Motor base rotatoria.** La rotación con respecto a la carcasa se realiza alrededor de un eje vertical, por lo que para nuestro modelo hemos vuelto a elegir un *joint cilíndrico* centrado en la plataforma y con eje perpendicular a su superficie.

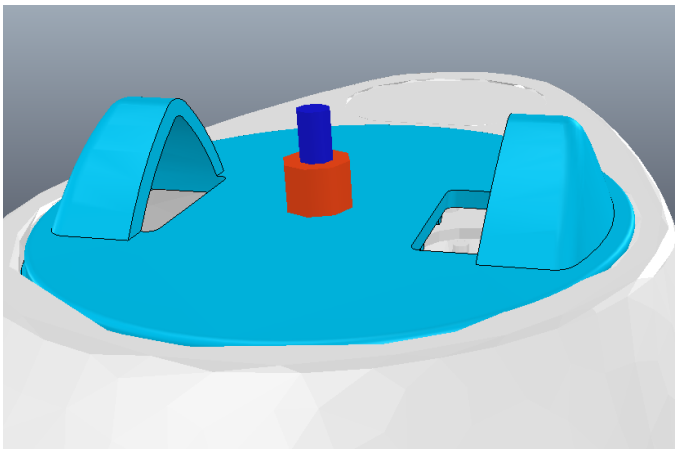


Figura 31. Joint de la base rotatoria.

- **Motor soporte Smartphone.** Igual que en los casos anteriores, se trata de un movimiento rotatorio con un grado de libertad, por lo que se volverá a emplear un *joint* cilíndrico. Estará colocado en horizontal a lo largo del soporte.

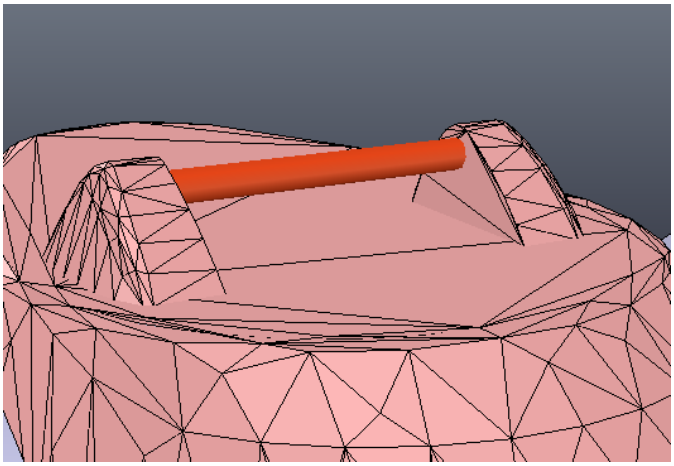


Figura 32. Joint del soporte del Smartphone.

La velocidad de rotación de los *joints* puede ajustarse a una *target velocity* (velocidad objetivo), que será alcanzada según el valor de par que haya sido configurado. Si este valor es muy alto, la velocidad objetivo será alcanzada al instante. Más adelante, se verá cómo se ha ajustado la velocidad en este modelo.

7.2.2 Shapes

Las piezas del apartado 7.1 son las *shapes* de las que se habla en este apartado y que participan en la simulación dinámica de forma activa o pasiva.

Como ya se ha explicado con anterioridad, existen componentes visuales y componentes dinámicas de las piezas del robot. A continuación se can a definir las características necesarias para que se comporten de una u otra forma (estáticamente o dinámicamente).

El programa V-REP ofrece la clasificación de sus elementos en cuatro tipos, dependiendo de su actuación durante la simulación:

- *Non-respondable and static.* No se ven afectados por la presencia de otros objetos ni reaccionan a colisiones con ellos. Carecen de masa. No se mueven, o únicamente lo hacen siguiendo el movimiento de los objetos con los que están emparentados. Este es el caso de las componentes visuales de las piezas del ROBOBO.
- *Non*-respondable and dynamic. Al igual que los anteriores, no reaccionan ante otros objetos, pero sí que tienen masa y se ven afectados por fuerzas y pares externos. Este es el caso de la batería y los motores creados para simular la masa de dichos componentes.
- *Respondable and static.* Provocan el movimiento de los objetos que colisionen con ellos mientras que permanecen estáticos al carecer de masa.

- *Respondable and dynamic.* Reaccionan ante el contacto con otros objetos, provocando movimiento y siendo movidos (tienen masa). En este tipo se engloban todos los componentes dinámicos del apartado 7.1.

Otra información que es importante aportar a la hora del correcto desarrollo del modelo es el material de las diferentes partes. V-REP asigna un material por defecto a todos los componentes marcados como *respondables.* Se ha modificado el material de las ruedas por *wheelMaterial* (posee un parámetro de fricción máximo), y el de la deslizadera por un *noFrictionMaterial* (tiene un parámetro de fricción con valor nulo).

7.2.3 Force sensors

Los *force sensors* constituyen uniones rígidas entre objetos y poseen la capacidad de medir fuerzas y pares trasmitidos en los 3 ejes. Estas uniones pueden llegar a romperse si se superan ciertos límites establecidos. En la siguiente figura se muestra un ejemplo de utilización de un *force sensor.*

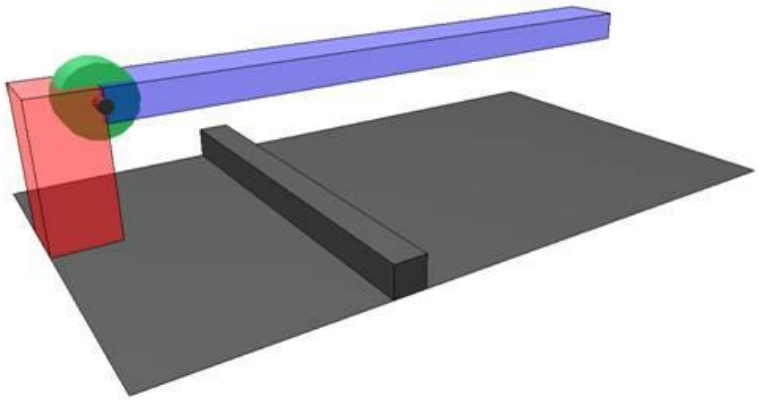


Figura 33. Unión realizada mediante f*orce sensor.*

En el siguiente apartado se muestra como serán utilizados en nuestro modelo del ROBOBO para unir ciertas piezas entre sí.

7.2.4 Estructura del modelo.

En este apartado se explica la forma en la que se han unido los diferentes componentes del modelo entre sí. Para ello, se realiza la estructura en forma de árbol jerarquizada de forma que los elementos de la parte más alta del mismo son considerados los "padres" de aquellos de los niveles inferiores.

No se pueden vincular dos objetos cualesquiera sin atender a sus características y existe un modo muy concreto de realizar las uniones.

Empezando por los objetos dinámicos (*joints, shapes* del tipo *non-respondable/respondable and dynamic* y *force sensors),* las dos configuraciones posibles serían:

- Objeto dinámico 1 + *force sensor* + objeto dinámico 2.
- Objeto dinámico 1 + *joint* + objeto dinámico 2.

Siendo el objeto dinámico 1 el padre del *joint* y este último el padre del objeto dinámico 2.

Por lo tanto, al elegir la componente dinámica de la carcasa como primer objeto de la cadena, el resto de componentes se enlazarán a ella mediante *force sensors* (batería, motores y deslizadera) y *joints (las ruedas y la base rotatoria).* A la base rotatoria se asocia, a su vez, el *joint* del soporte del Smartphone.

El modelo que hemos estructurado hasta ahora se corresponde con la que participa de forma activa en la simulación, y tiene el aspecto que se muestra en la siguiente figura:

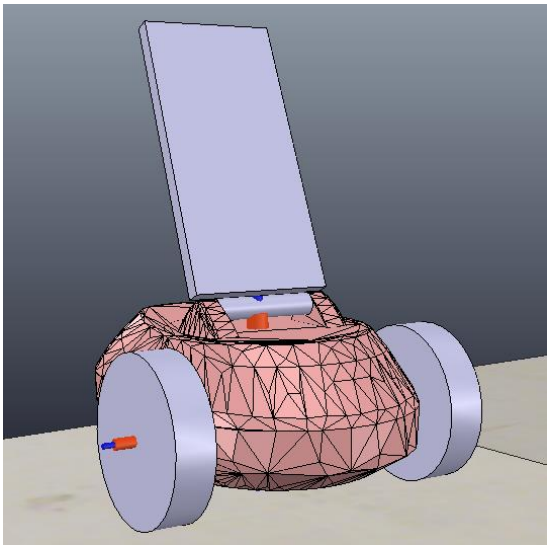


Figura 34. Componente dinámica.

La forma en la que se incluye la componente visual del modelo es haciéndola hija de la componente dinámica, tal y como se muestra en la Figura 34.

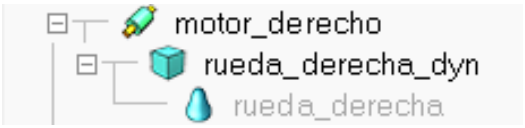


Figura 35. Relación entre joint, componente dinámica y componente visual.

V-REP permite trabajar con diferentes capas, de modo que puedes decidir qué componentes ver en cada una. Esto permite ocultar la capa dinámica, mostrando únicamente la capa visual tal y como se ve en la figura 36.

Figura 36. Componente visual del ROBOBO.

7.3 Representación de los volúmenes de detección de los sensores infrarrojos.

Como ya se ha comentado en capítulos precedentes, el ROBOBO consta de 8 sensores infrarrojos de unos 20 cm de alcance.

Ensayos realizados para verificar el alcance y rango de actuación del sensor infrarrojo indicado por la hoja de datos del mismo, nos ha permitido comprobar que el comportamiento y las capacidades del sensor infrarrojo real son diferentes a las especificadas por la hoja de datos del fabricante. Considerando una proyección en planta, tenemos que el área de detección del sensor infrarrojo es la siguiente.

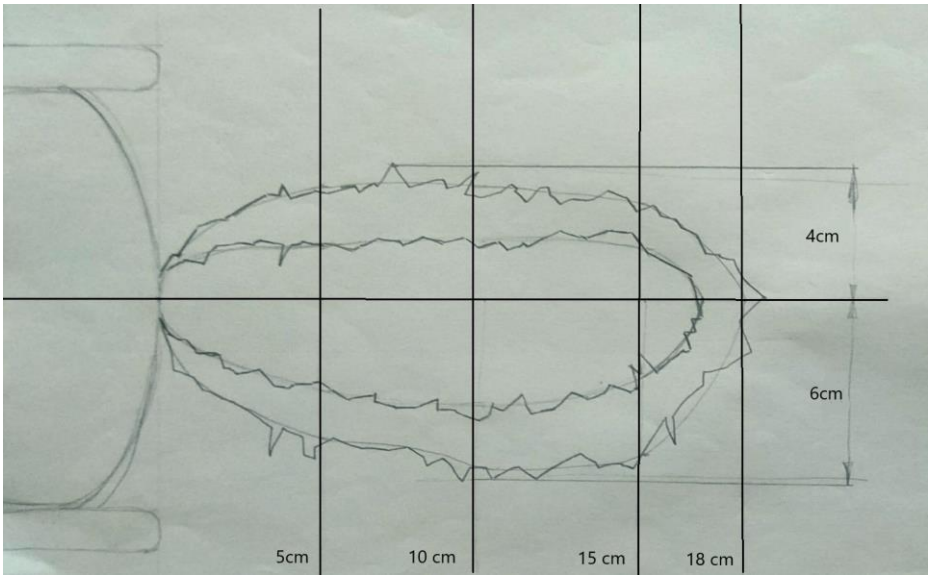


Figura 37. Forma del área de detección de los sensores infrarrojos.

En cuanto a la posición de los sensores en la carcasa, es la siguiente:

- Un sensor en el centro de la parte frontal de la carcasa. Es el representado en la figura anterior. Longitudinalmente, el volumen de detección se encuentra perpendicular a la superficie de la carcasa.
- A ambos lados de este se encuentran dos sensores que forman un ángulo de -15° con la horizontal. Están girados aproximadamente 18° con respecto al central. Sirven para detectar el suelo y evitar que el ROBOBO se caiga.
- Otros dos sensores se sitúan en la parte frontal del robot, próximos a las ruedas. Son perpendiculares a la carcasa y están girados 45° con respecto al central.
- En la parte posterior central, se encuentra un sensor inclinado -15° con respecto a la horizontal. Detecta el suelo y evita que el ROBOBO se caiga por la parte trasera.
- A los lados de este último se encuentran dos sensores que forman 0° con la horizontal y que verticalmente se desvían 45° con respecto al central.

En un primer momento se decidió representar el volumen de detección de los sensores según los datos anteriores en el V-REP, empleando los sensores cónicos y cilíndricos que nos ofrece el simulador, tal y como se ve en la figura 38.

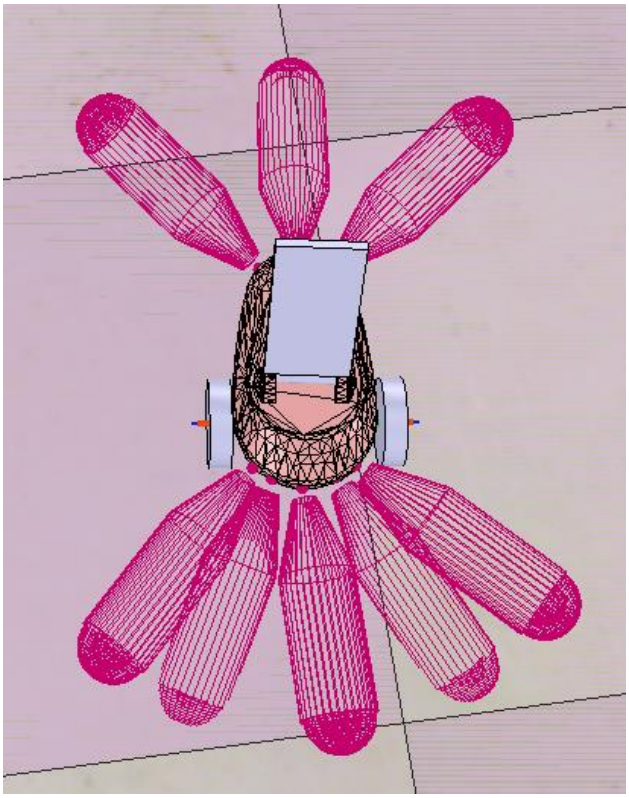


Figura 38. Primer intento de caracterización de los volúmenes de detección de los sensores infrarrojos.

Probando el funcionamiento de este modelo de los volúmenes de detección nos hemos encontrado con dos problemas principales, causados por el funcionamiento de los sensores de V-REP, que únicamente detectan aquel objeto que se encuentra más próximo al sensor:

Por un lado, el diámetro de los diferentes cilindros y conos que forman los volúmenes de detección ha tenido que reducirse con respecto al real, puesto que de aumentarse su tamaño, estos detectarían el suelo e ignorarían cualquier objeto que se introdujera en su volumen. No obstante, tratar de eliminar este problema dio lugar a la aparición de uno nuevo que consiste en que, al situar un objeto de tamaño conocido ante el ROBOBO real y ante su modelo de simulación, el número de sensores que lo detectan es diferente en ambos casos para algunas de las zonas. Para reducir este problema, se ha optado por sustituir los sensores cilíndricos y cónicos por sensores tipo *pyramid* y tipo *disc*, que permitan que la altura y el ancho del sensor sean distintos (Figura 39), pudiendo así abarcar un mayor ángulo horizontal, sin interceptar el suelo.

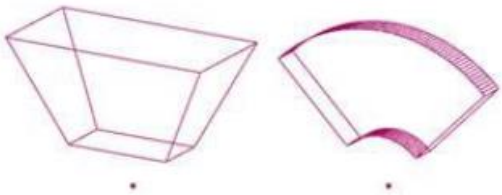


Figura 39. Sensores tipo *pyramid* y *disc*.

Además, esta disposición dio lugar a un problema con los sensores encargados para la detección del suelo. Puesto que al únicamente detectar aquel objeto que se encuentra más próximo a ellos, el suelo, no indicarían la presencia de un objeto dentro de su zona de detección. Este no es el caso de los sensores infrarrojos reales, en los cuales sí que se mostraría un aumento del valor de intensidad devuelto, puesto que hay un aumento de los rayos reflejados al encontrarse parte de ellos con la superficie del objeto.

Por lo tanto, se ha optado por dividir estos sensores en dos componentes, una encargada de la detección del suelo y otra que se encargaría de detectar objetos que entrasen en su volumen de detección.

Así, la forma final que se ha elegido para representar los sensores infrarrojos en el modelo de simulación es la representada en la figura 40.

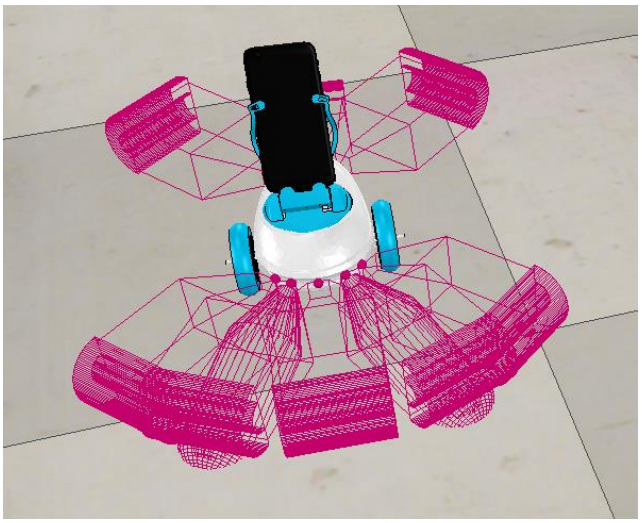


Figura 40. Representación final del volumen de detección de los infrarrojos.

Como puede verse, cada sensor se ha representado dividiéndolo en tres secciones diferentes, para así poder adaptarse mejor a la forma del volumen vista anteriormente. Los parámetros elegidos para los sensores que forman 0° con la horizontal en nuestro modelo son los indicados en la figura 41.

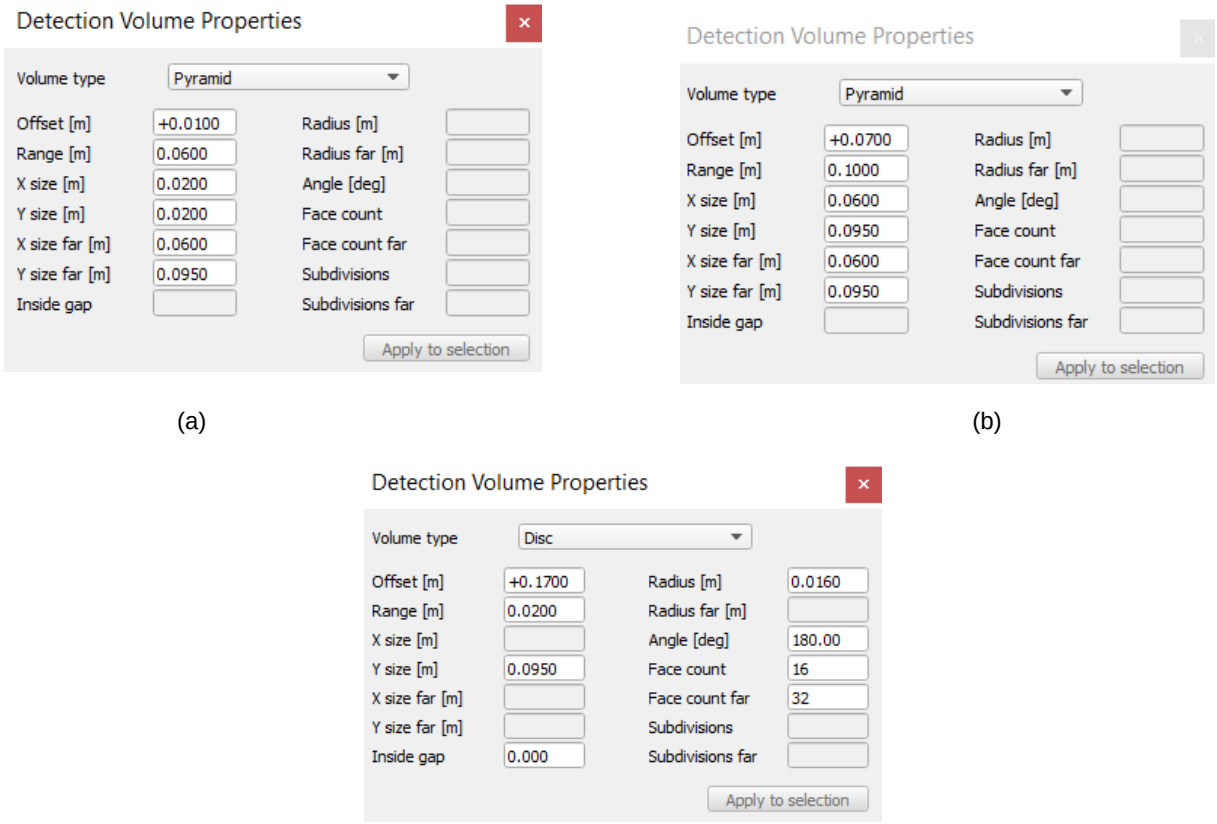
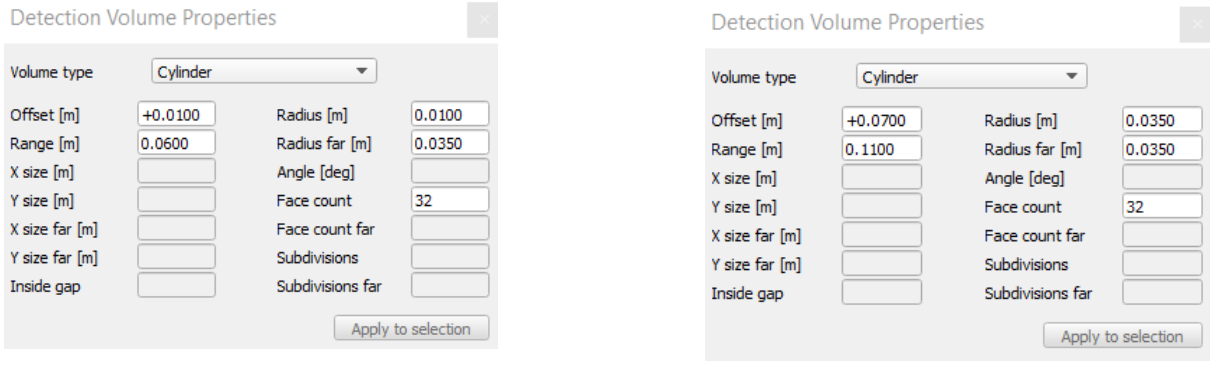
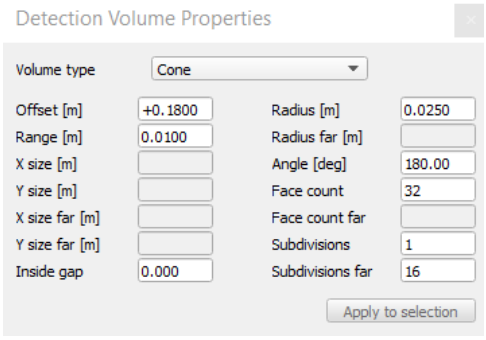


Figura 41. Características de las secciones de los sensores horizontales. (a) Sección más próxima. (b) Sección intermedia. (c) Sección más alejada.

Los sensores que se emplean para detectar el suelo también están divididos en tres secciones que poseen las siguientes características (Figura 42).





(c)

Figura 42. Características de las secciones de los sensores inclinados. (a) Sección más próxima. (b) Sección intermedia. (c) Sección más alejada.

7.4 Determinación del comportamiento de los sensores infrarrojos en el entorno real.

Para continuar desarrollando el modelo de simulación, será necesario conocer cómo se comportan los sensores infrarrojos reales. Esto nos permitirá implementar un modelo de simulación del funcionamiento de los sensores infrarrojos similar al real.

La determinación del comportamiento de los sensores infrarrojos ya se había hecho con anterioridad en el modelo de simulación de la versión 2.0 del ROBOBO, sin embargo se ha podido observar que esa caracterización no se ajusta al comportamiento real de los sensores y surge la necesidad de mejorarlo.

Gracias a la hoja de datos de los sensores sabemos que estos no poseen una intensidad de detección constante a lo largo de todo su volumen de detección manteniendo la corriente del emisor constante, sino que se experimenta una reducción de la misma con la distancia (Figura 43). Con la intención de conocer cómo se produce esta reducción de la intensidad de detección de los sensores con la distancia se ha llevado a cabo el primer ensayo, consistente en colocar un mismo objeto en puntos a diferentes distancias del ROBOBO, manteniendo a este en una posición fija, y comprobar para cada uno de estos puntos el valor de los sensores.

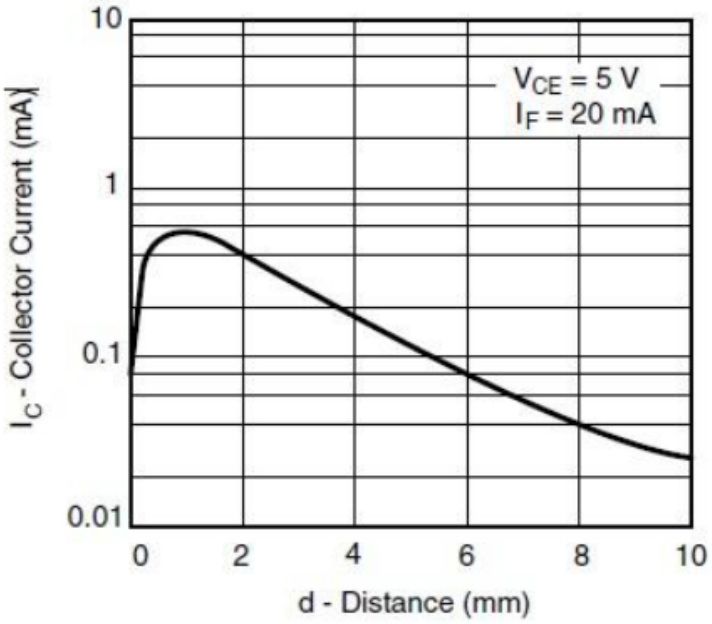


Figura 43. Curva de la intensidad del colector frente a la distancia.

7.4.1 Caracterización de la curva de intensidad de un sensor infrarrojo aislado.

En esta prueba se procederá a determinar el funcionamiento real de un sensor infrarrojo. Para realizarlo se sitúa el ROBOBO en una posición fija sobre una hoja en la que, tomando la posición del sensor frontal central como origen, estableciendo distancias en variaciones de 1 cm. Si se tiene en cuenta que el diámetro de la rueda es de 6.5cm, 1cm de distancia se correspondería con un giro de la rueda de 17.6°, un valor significativamente pequeño. Por lo tanto, y considerando la velocidad a la que se puede desplazar el robot (40 cm/s), se estima que 1cm es una distancia lo suficientemente pequeña para ser considerada como valor mínimo en la medición. Esta disposición se muestra en la figura 43.

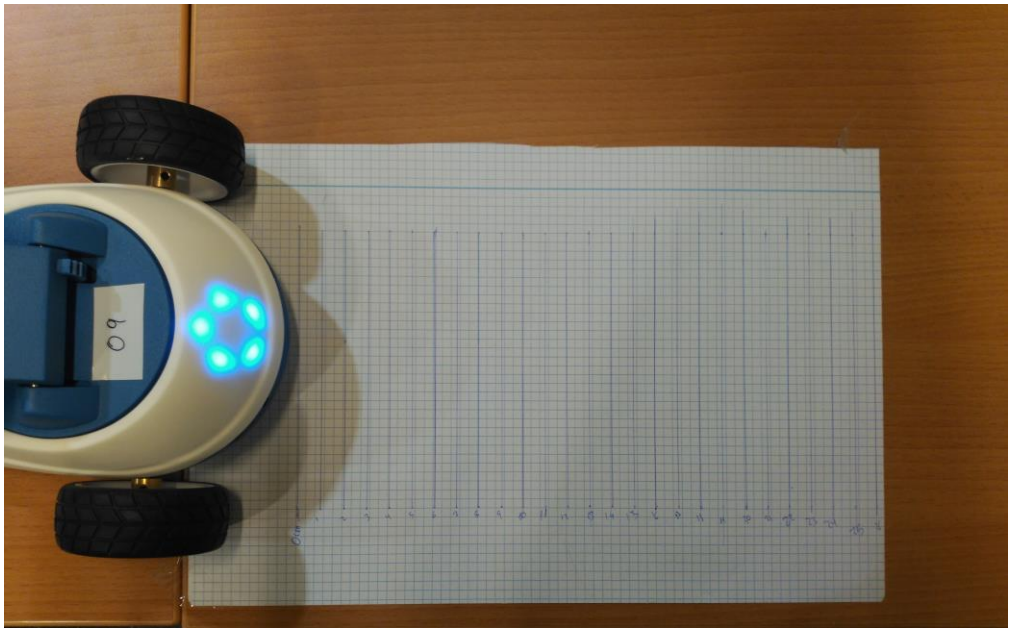


Figura 43. ROBOBO y hoja con las diferentes distancias.

Algo que va a ser muy importante a la hora de hacer las pruebas es que el objeto empleado sea siempre del mismo material, puesto que los la sensibilidad del sensor varía mucho de un material a otro (Figura 44). En este caso nos hemos decantado por elegir hojas de papel blancas debido a su buena reflectividad y este será el material que empleemos en todas las pruebas de este Trabajo de Fin de Grado.

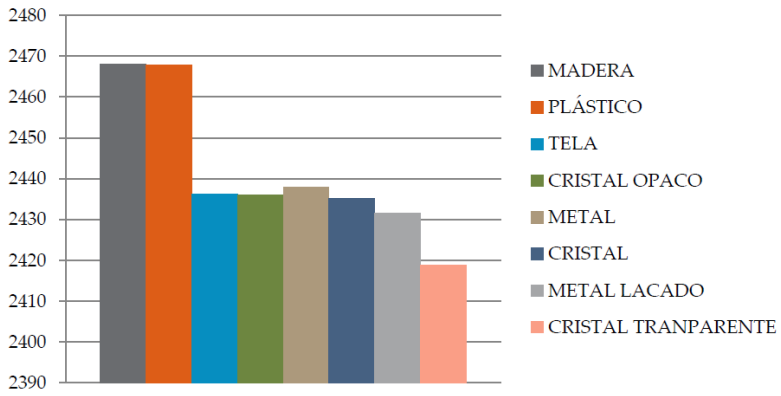


Figura 44. Sensibilidad del sensor en diferentes materiales. Fuente: TFG Alba Lema Martínez.

Se ha decidido emplear como objeto a detectar una caja de cartón con una de sus caras forrada con una hoja de papel (Figura 45). Esta nos facilita asegurar que la colocación del objeto es la adecuada en cada momento, es decir, que la hoja de papel está siempre orientada perpendicularmente al haz de emisión del sensor.

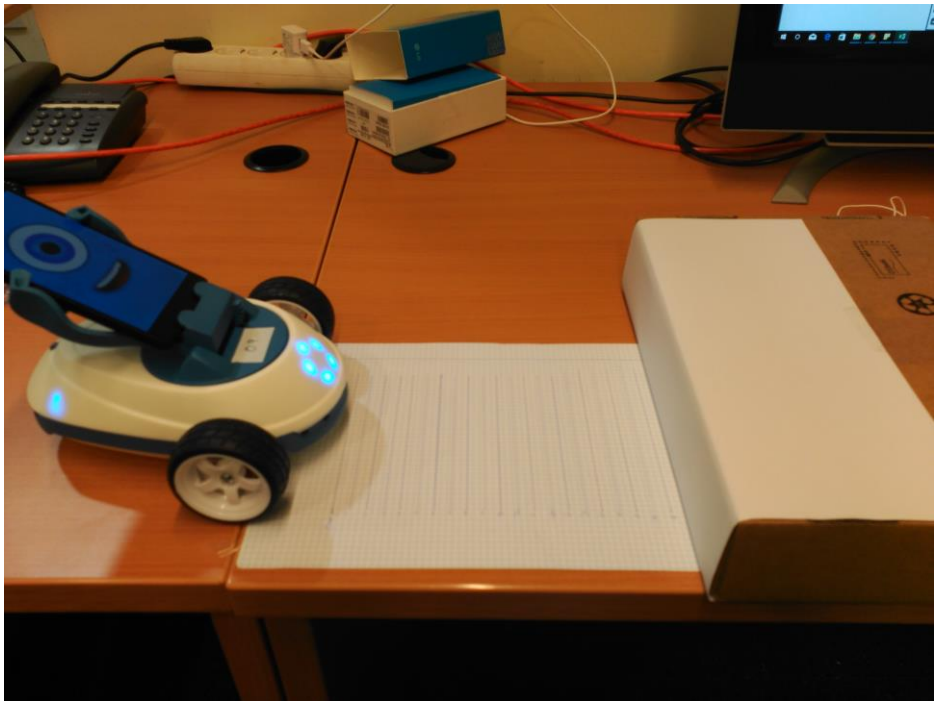


Figura 45. Nuevo objeto a detectar.

Una vez que está todo dispuesto, se conecta el ROBOBO al Scratch, plataforma que permite, entre otras cosas, poder ver el valor de intensidad devuelto por los sensores infrarrojos en cada momento. A continuación, se irá situando el objeto en los diferentes puntos marcados, anotando los valores de los sensores correspondientes a cada uno, obteniendo así la siguiente tabla.

Tabla 1. Intensidad devuelta por los sensores en función de la distancia.

Distancia (m)	Ensayo 1	Ensayo 2	Intensidad media
0,01	7534	7479	7506,5
0,02	1948	1983	1965,5
0,03	897	899	898
0,04	533	544	538,5
0,05	360	359	359,5
0,06	253	256	254,5
0,07	192	190	191
0,08	148	145	146,5
0,09	113	115	114
0,1	92	92	92
0,11	74	75	74,5
0,12	61	63	62
0,13	52	52	52
0,14	44	44	44
0,15	37	38	37,5
0,16	33	33	33
0,17	29	29	29
0,18	26	26	26
0,19	23	24	23,5
0,2	21	20	20,5

El siguiente paso será el de crear las gráfica que represente los datos de la tabla anterior, añadiendo en ella la línea de tendencia de los datos y obteniendo la ecuación de dicha línea empleando Excel (Figura 46)

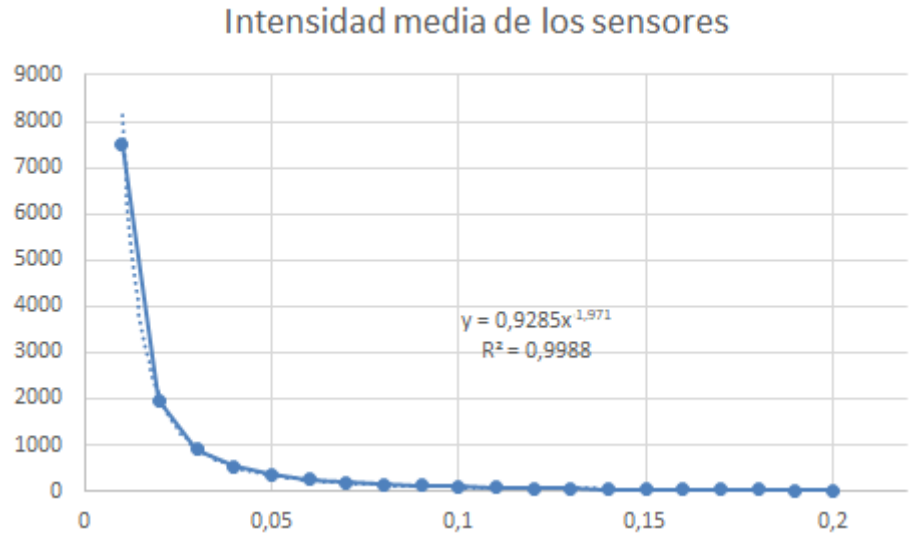


Figura 46. Curva de intensidad media de los sensores con la distancia.

Como se puede observar en esta figura, los valores de la intensidad varían de forma mucho más brusca en las regiones cercanas al sensor (de 0 a 4 cm) mientras que en las zonas más lejanas lo hace en menor grado.

Debido a que el ROBOBO es un robot autónomo diseñado para operar en entornos reales y heterogéneos, se decidió comprobar si estos valores sirven para medir la distancia del robot a cualquier objeto que se pueda encontrar en dicho entorno. Para ello, simplemente se reorientó el objeto empleado anteriormente de la forma indicada en la figura 47.

Figura 47. Objeto reorientado.

En la tabla 2 se muestran los resultados obtenidos junto con los proporcionados por el ajuste anterior y el error relativo cometido al aplicarlo.

Tabla 2. Nuevos valores de intensidad.

Distancia (m)	Valor ajuste	Intensidad media	Error ensayo
0,01	7392,49	11262	34,36%
0,02	1991,78	2601	23,42%
0,03	924,86	1122	17,57%
0,04	536,65	664	19,18%
0,05	351,84	423	16,82%
0,06	252,89	300	15,70%
0,07	193,20	231	16,37%
0,08	147,64	179	17,52%
0,09	113,99	144	20,84%
0,1	89,99	120	25,01%
0,11	73,39	103	28,75%
0,12	61,96	86	27,96%
0,13	53,44	75	28,75%
0,14	45,59	67	31,95%
0,15	36,17	59	38,70%
0,16	32,92	53	37,88%

A la vista de los errores que se observan en la tabla, se puede comprobar que la suposición de la hipótesis de que la superficie no afecta al valor de la detección no es válida. Por ello se necesitará dar un nuevo enfoque al método a seguir, para que ésta sí que se tenga en cuenta.

7.4.2 Relación del valor del sensor infrarrojo con la superficie del objeto.

A la vista de los resultados mostrados por la tabla 2, se ha visto necesario la realización de un nuevo ensayo. Este ensayo tiene como objetivo comprobar la relación existente entre el valor que proporciona un sensor infrarrojo, la superficie disponible de detección, y la distancia a la que se encuentra el objeto. Para ello se ha dividido el ensayo en dos partes diferentes: variación de altura y variación de anchura. Con la finalidad de llevar a cabo estos ensayos se han creado diferentes objetos como los que se nos muestran en la figura 48. En el ensayo de variación de altura, nos aseguramos de que el ancho del elemento a detectar fuese constante y lo suficientemente grande para que se pueda considerar como "infinito" desde el punto de vista del sensor infrarrojo. Lo mismo se ha hecho con la altura en el ensayo de variación constante. El motivo de esto es asegurarse de que la dimensión que no estamos poniendo a prueba no sea la responsable de ninguna variación en el resultado del valor indicado por los sensores.

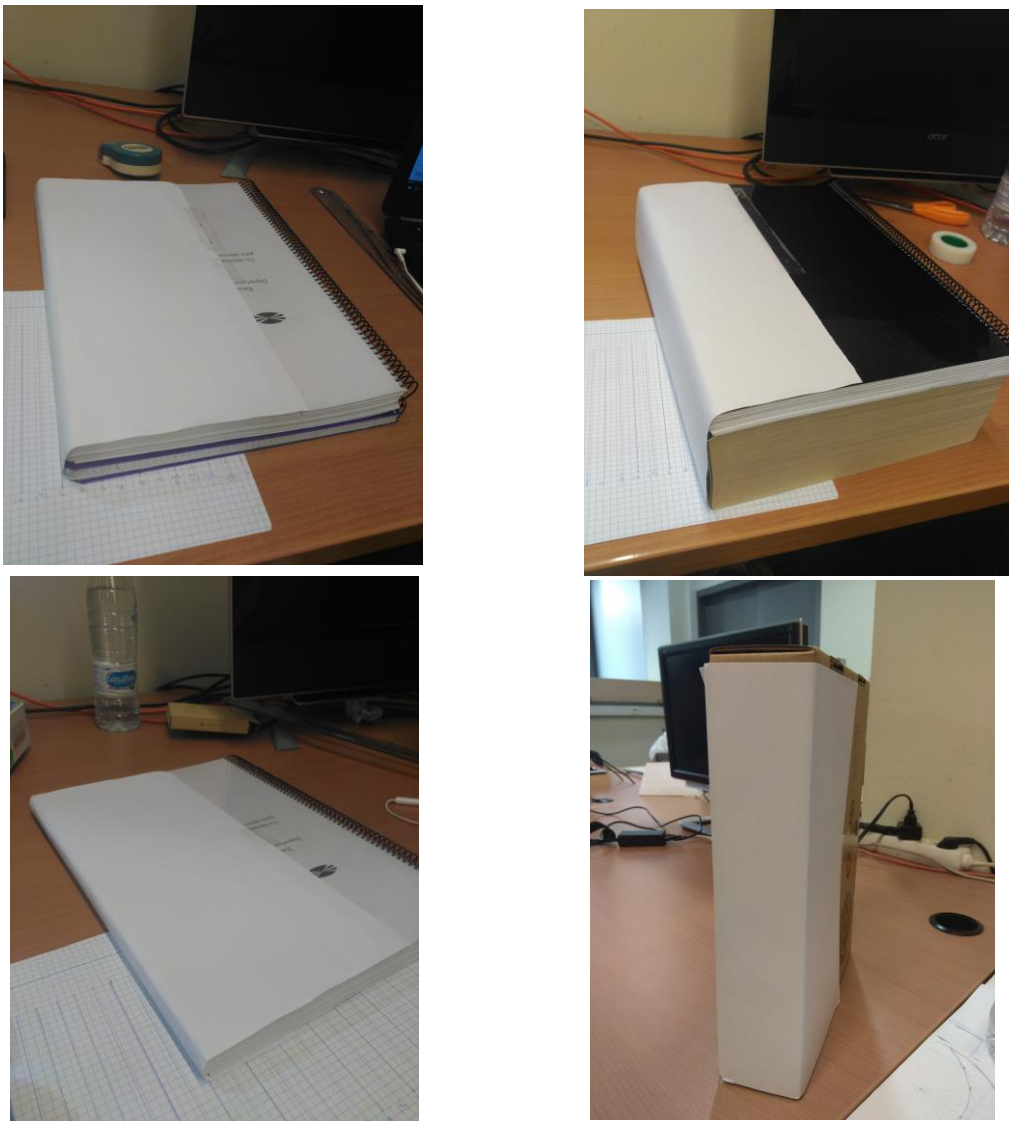


Figura 48. Ejemplos de diferentes objetos utilizados.

Se tomaron los valores devueltos por el sensor frontal para un total de 14 objetos de diferentes alturas y anchos. No serán incluidos todos ellos puesto que finalmente muchos no han sido empleados en el desarrollo puesto que el ajuste que se realizará en este apartado se descartó antes de llegar a utilizarlos. Los valores más representativos serán los obtenidos para el cuerpo de 29.5 cm de ancho y 12.5 cm de alto. Esto es así porque se conocen las dimensiones del volumen de detección del sensor, cuyas secciones circulares no superan los 10 cm de diámetro, podemos deducir que el rayo del sensor estará contenido completamente en la superficie del elemento de dimensiones anteriormente mencionadas. Esto nos permite asumir que los valores devueltos por los sensores infrarrojos en estas circunstancias son los mayores que se pueden conseguir, puesto que están siendo reflejados la totalidad de los rayos infrarrojos. Gracias a esto, se creará una función "densidad" que represente la intensidad del

sensor por unidad de área del mismo. Para ilustrar mejor lo mencionado anteriormente se dispone de la Figura 49, en ella se ve un sensor con su volumen de detección orientado hacia un cubo. La zona sección circular rosa que se halla sobre la superficie del objeto, sería el área entre la que se dividiría la intensidad devuelta por el sensor para esa distancia.

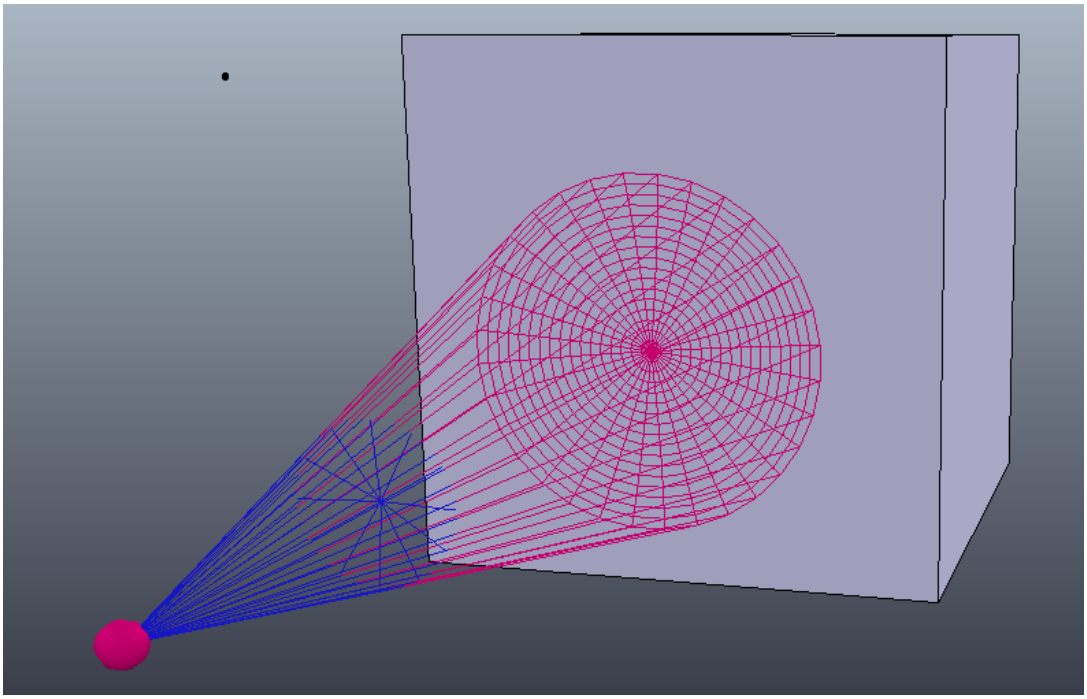


Figura 49. Ejemplo de volumen de detección de sensor contenido en la superficie detectada.

Para la creación de la función densidad, lo primero será crear una función que nos permita obtener el área del volumen de detección en función de la distancia. Esto se ha hecho midiendo los radios de las secciones circulares de este, representándolos gráficamente y obteniendo una línea de tendencia que se ajuste a los valores de forma apropiada, obteniendo así los resultados mostrados en la figura 50.

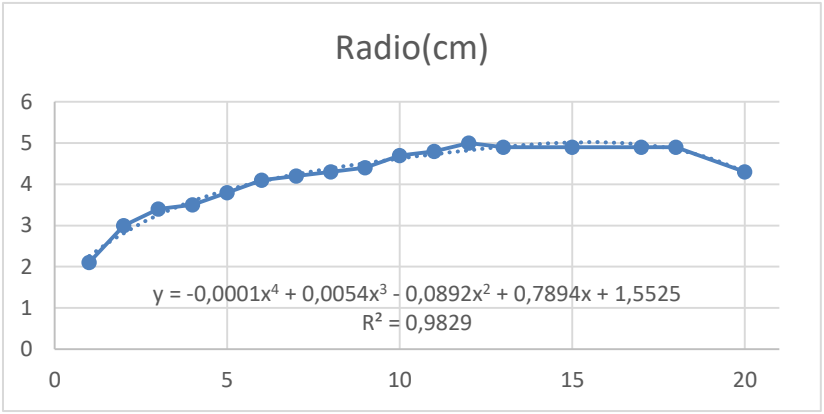


Figura 50. Representación del radio de las secciones circulare que forman el volumen de detección del sensor con la distancia.

Una vez obtenido esto, se calcula el área fácilmente (área= πr^2) y con esta se puede obtener el valor de la intensidad por unidad de área para cada distancia (Tabla 3).

Tabla 3. Intensidad media por unidad de área del sensor en función de la distancia. Se incluyen los valores necesarios para su obtención.

Distancia (cm)	Intensidad media	Radio detección	Área sensor (cm^2)	"densidad" ($1/cm^2$)
1	9844,67	2,37	17,69	556,54
2	2156,33	2,80	24,59	87,68
3	950,67	3,17	31,65	30,04
4	556,67	3,51	38,61	14,42
5	369,67	3,80	45,28	8,16
6	263,67	4,05	51,52	5,12
7	199,33	4,27	57,22	3,48
8	157,00	4,45	62,30	2,52
9	124,67	4,61	66,71	1,87
10	104,33	4,73	70,39	1,48
11	88,67	4,83	73,29	1,21
12	75,67	4,90	75,38	1,00
13	64,00	4,94	76,62	0,84
14	56,00	4,95	76,96	0,73
15	49,67	4,93	76,37	0,65
16	43,00	4,88	74,81	0,57
17	38,00	4,80	72,24	0,53
18	34,33	4,67	68,65	0,50
19	30,67	4,51	64,04	0,48
20	27,67	4,31	58,42	0,47

La representación gráfica de esta tabla se muestra en la Figura 51.

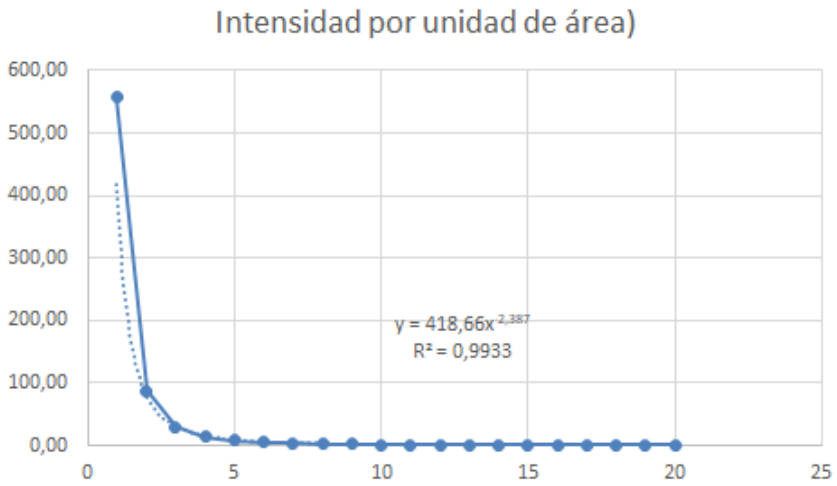


Figura 51. Intensidad por unidad de área en función de la distancia.

Como la finalidad de los sensores infrarrojos posicionados en el Robobo es la de detectar objetos y/o ausencia de superficie para aquellos orientados hacia el suelo, con este ensayo se pretende saber la distancia a la que se encuentra un objeto, conociendo las dimensiones del mismo y el valor devuelto por los sensores. Para ello aproximaremos el área de intersección de la superficie del objeto con el volumen de detección del sector. Esto se mediante un proceso iterativo, explicado a continuación.

1. Se calcula una aproximación del área máxima que se podría detectar. Esto es, se multiplica su ancho por su altura, pero en el caso de que alguna de esas dimensiones fuera superior a los 10 cm de diámetro máximo de las secciones del sensor, se emplearían los 10 cm en su lugar.
2. Con el valor del área obtenida y el valor de intensidad del sensor, obtenemos un valor posible de la "densidad".
3. Se introduce el valor de la densidad en la ecuación correspondiente de las obtenidas en la figura 49, obteniendo así una distancia.
4. Con esta distancia, se obtiene el radio del sensor mediante la ecuación de la figura 48. Si alguna de las dimensiones de nuestro objeto es superior a este, se empleará el diámetro en su lugar.
5. Así, se vuelve a calcular un nuevo valor de densidad, repitiendo el proceso el número de veces necesario para que la diferencia entre las distancias obtenidas en dos iteraciones consecutivas sea lo suficientemente pequeña.

Probando este método se puedo comprobar que tenía un comportamiento aceptable a distancias intermedias en objetos cuya altura era superior a los 3.5 cm a los que se encuentra situado el sensor. Esto es debido a que en objeto más bajos, el sensor detecta también la parte superior del mismo, aumentando el área detectada e incrementando el valor de la intensidad devuelto, con respecto al estimado

No obstante, nuestro modelo no contempla un posible giro o desplazamiento con relativo del objeto con respecto a la línea de acción del sensor. En ambos casos, el valor de los sensores puede variar, aumentando o disminuyendo dependiendo del tipo de giro, y disminuyendo cuando el objeto se desplace lo suficiente como parte de los rayos que forman el volumen de detección del sensor dejen de encontrarse con el objeto. En la situación de un giro en el que el valor proporcionado por el sensor aumente, como el objeto sigue teniendo la misma área, la densidad de nuestro modelo será mayor, lo que dará lugar a que obtengamos distancias más pequeñas que la real. Paralelamente, si se produjese un desplazamiento o un giro que

disminuyeran la intensidad del sensor, nuestro modelo nos lo situaría en una posición más lejana a la real.

Por esto, se ha concluido que no va a ser posible determinar la posición de un objeto empleando únicamente los datos facilitados por uno de los sensores ya que nos es posible conocer las orientaciones y posiciones relativas del objeto con respecto al sensor, haciendo imposible calcular las distancias de forma correcta.

Otra conclusión que se ha sacado de este ensayo es la influencia de la geometría del objeto en los valores que dan los sensores infrarrojos. Se ha visto que, varios objetos situados a una misma distancia, ofrecen valores diferentes en función de la forma geométrica que tienen.

7.4.3 Modelo con objeto de referencia fijo

Como se ha visto en el apartado anterior, existe una gran dificultad a la hora de calcular la distancia a la que se encuentra un objeto sin conocer su morfología. Como no se quiere que esta sea un parámetro a considerar dentro de la ecuación, para este ensayo se ha cambiado el planteamiento y se ha decidido emplear un objeto de morfología fija. Además, con objeto de tener una mayor idea de la orientación y posición del objeto se ha decidido añadir a la medición los restantes sensores infrarrojos y analizar la influencia que tienen en este ensayo. Por ello, el elemento empleado a partir de ahora será un cubo de arista 10 cm (Figura 52).

Figura 52. Cubo de arista 10 cm.

Además, tendremos en cuenta el valor de los 5 sensores en cada una de las posiciones en las que situemos el objeto. Con esto será más fácil hacerse una idea de su localización y orientación, puesto que como ya hemos mencionado en el apartado 7.3, existen varias zonas del volumen total de detección del ROBOBO en los que un objeto puede ser detectado por más de un sensor.

7.4.3.1. Validación con una red neuronal.

En este sub-apartado trataremos de determinar el comportamiento de los 5 sensores con la posición de nuestro objeto.

Debido a la posición de los sensores infrarrojos en la carcasa, se ha cambiado el sistema de representación de la posición del objeto, empleando ahora un mapa en coordenadas polares en el que el centro será el punto en el que se unen las líneas de acción de todos los sensores infrarrojos del robot. (Figura 53)

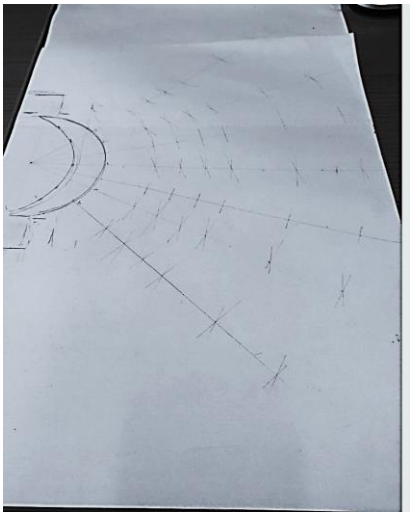


Figura 53. Plano de coordenadas polares sobre el que se realizarán las medidas.

Para realizar las mediciones, iremos desplazando nuestro objeto sobre los puntos de referencia que hemos especificado, manteniendo su superficie perpendicular a la línea de acción de los sensores y haciendo que el centro de la cara del cubo esté situada sobre dicha línea. A continuación se muestra un ejemplo de los valores obtenidos en los 5 sensores cuando el objeto se sitúa frente al sensor frontal central (Tabla 4).

Tabla 4 Valores de los sensores infrarrojos al colocar el cubo frente al sensor frontal central.

d(cm)	d1(cm)	d2(cm)	d3(cm)	v0	v1	v2	v3	v4	α2
4	10,96	**9,75**	10,96	18	246	547	306	23	**90**
6	12,77	**11,75**	12,77	16	90	252	130	19	**90**
8	14,63	**13,75**	14,63	15	55	153	68	17	**90**
10	16,52	**15,75**	16,52	15	36	102	50	16	**90**
15	21,34	**20,75**	21,34	15	25	43	35	15	**90**
20	26,23	**25,75**	26,23	15	22	23	31	16	**90**

Donde:

- d es la distancia del punto central del cubo a la carcasa del ROBOBO.
- d2 es la distancia del mismo punto central del cubo al origen de nuestro plano de coordenadas.
- d1 y d3 son las distancia respecto al centro de nuestro plano de las aristas izquierda y derecha del cubo respectivamente (vistas desde la posición del ROBOBO).
- v0, v1, v2, v3, v4 se corresponde a los valores devueltos por los sensores en cada caso, de izquierda a derecha.
- α2 es el ángulo en el plano polar del punto central de la cara del cubo.

Una vez obtenidos estos valores, se procede a introducirlos en el modelo de ajuste que hemos seleccionando con el fin de obtener un valor para las distancias de los tres puntos del objeto descritos anteriormente. El método de ajuste seleccionado es una red neuronal que será entrenada con estos datos, tomando los valores de los sensores como entradas y las distancias (d1, d2, d3) y el ángulo α2 como salidas deseadas. Se ha elegido la red neuronal como método de ajuste debido a que son capaces de conseguir trabajar con una gran cantidad de datos fácilmente.

La creación de las redes neuronales se ha realizado en Matlab Se ha elegido este programa debido a que posee una herramienta especial para la creación y el entrenamiento de redes neuronales, por lo que el proceso resulta más intuitivo que en otros programas como Python en los cuales se hace mediante código de programación. En esta herramienta de Matlab se puede elegir fácilmente el tipo de red neuronal, el número de capas de las que consta, el número de neuronas por capa y el tipo de entrenamiento que se quiere realizar.

En este punto se ha tratado principalmente de entrenar redes neuronales artificiales de dos capas ocultas, tal y como se ve en la figura 54.

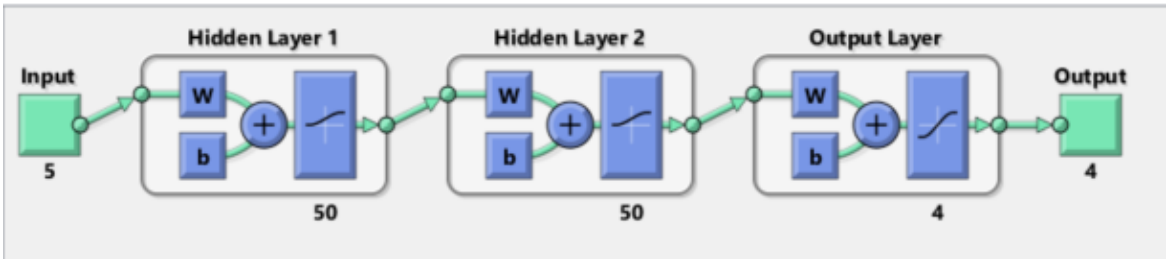


Figura 54. Ejemplo de red neuronal creada.

A continuación vamos a explicar el proceso a seguir para entrenar las diferentes redes neuronales que se han creado para intentar lograr el ajuste.

Primero, comenzamos creando una red neuronal con las capas deseadas, por recomendaciones, se han empleado redes neuronales feedforward con dos capas ocultas. En ejemplo, se cada una de ellas consta con 100 neuronas (Figura 55)

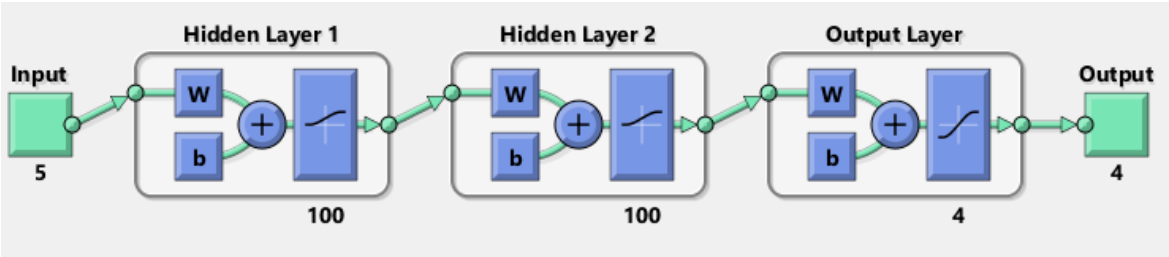


Figura 55. Red neuronal de 100 neurona por capa.

Como tipo de entrenamiento se ha seleccionado Bayesian Regularization puesto que aunque requiere un mayor esfuerzo computacional que los demás, ofrece resultados más correctos. Además se ha indicado que el proceso de ajuste se repita durante 100 iteraciones.

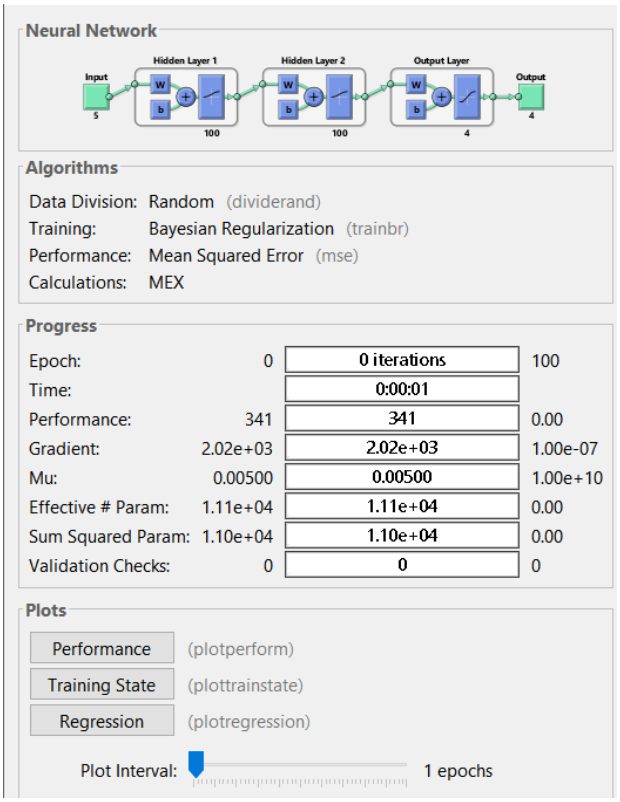


Figura 56. Pantalla de entrenamiento de red neuronal.

El programa tomará parte de los valores de entrada como datos de entrenamiento y parte como valores de ensayo. Los de entrenamiento los utiliza para ajustar los pesos y conseguir la salida deseada, mientras que los de ensayo se utilizan para ver el resultado que se obtiene en estos datos que no se han empleado para variar los pesos. A medida que aumentan las iteraciones, los errores en ambas muestras van disminuyendo (Figura 57)

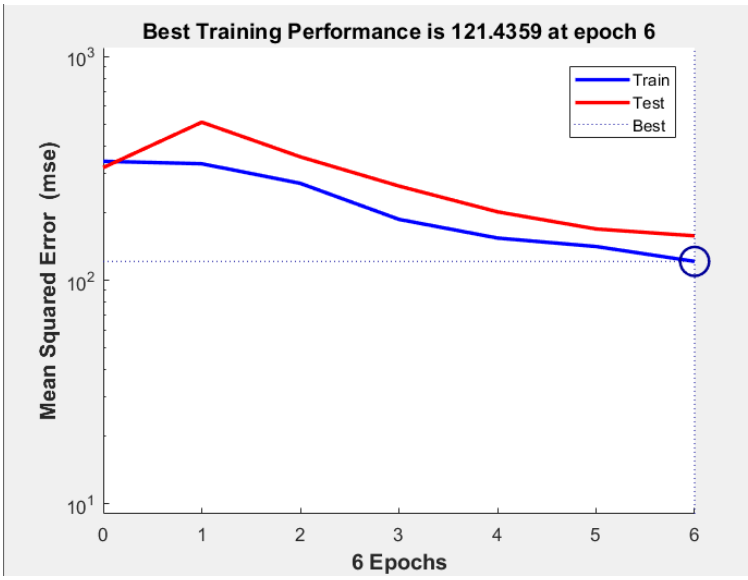


Figura 57. Progresión de los errores de las muestras de entrenamiento y ensayo.

En otra ventana se nos muestra la localización de los valores de salida con respecto al ajuste (Figura 58). Cuantos más puntos haya fuera de la línea de ajuste, peor será este.

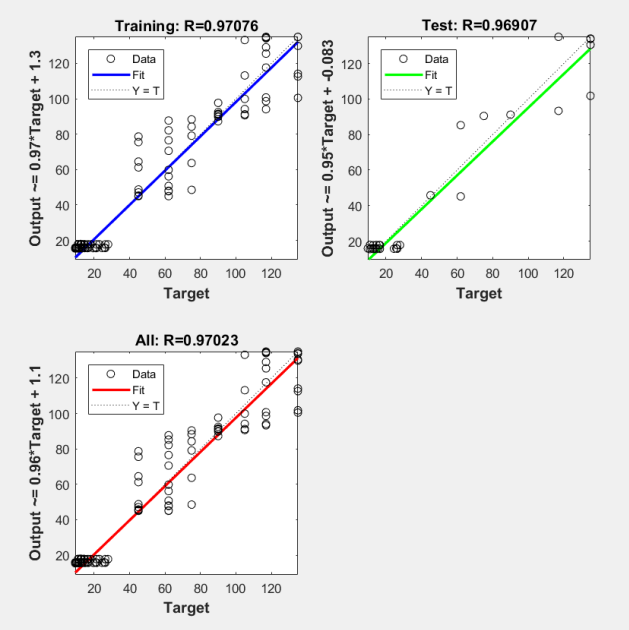


Figura 58. Representación gráfica del ajuste y los valores de salida deseados.

El proceso continuará hasta que se realicen todas las iteraciones o hasta que el usuario lo desee. Uno de los motivos para detenerlo antes de que finalice, es porque se produzca el sobreentrenamiento de la red, esto se vería en la gráfica que nos muestra los errores del ajuste, puesto que los errores cometidos con los valores de ensayo empezarían a aumentar (Figura 59)

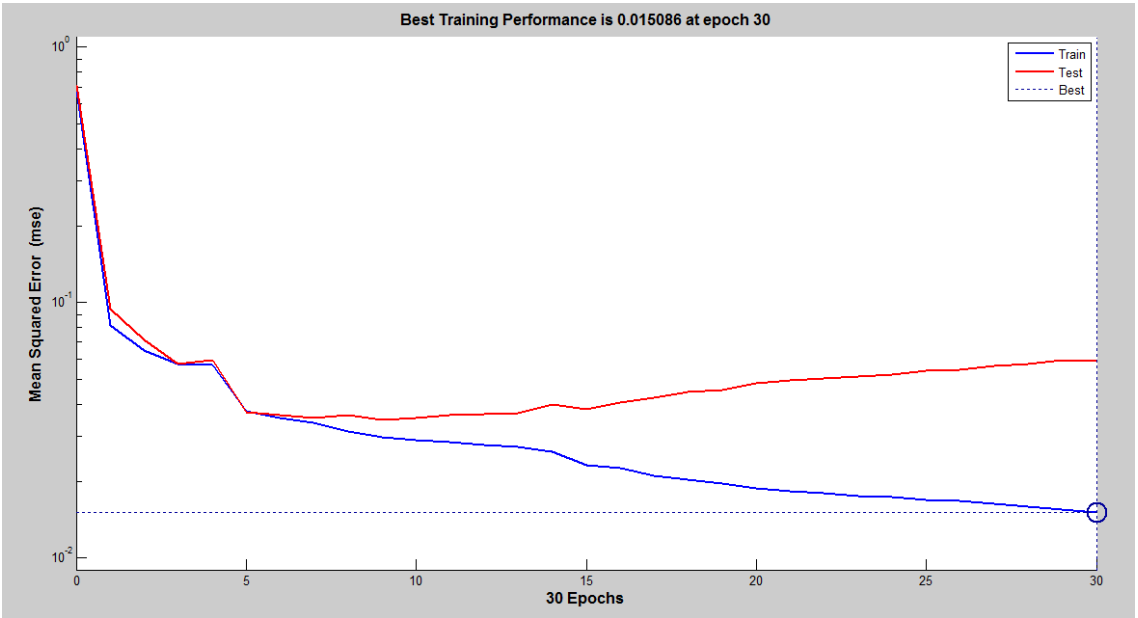


Figura 59. Red sobreentrenada.

Para la red utilizada como ejemplo, una vez finalizadas las 100 iteraciones, algunos de los errores (en tanto por cierto) que se cometen en las salidas son:

```
-6.1062   1.3747   -2.4874  -15.8011  -4.9662
-1.7436  -4.5531  -11.6815  -11.4903  -0.7570
 0.7639  -9.7298  -18.2436   -6.9024   1.5610
-15.1079 -0.1177    0.0007  -15.3839   0.0523
```

Figura 60. Ejemplos de errores que se cometieron con la red de 100 neuronas por capa..

Para una red de 50 neuronas por capa, como la de la Figura 61, entrenada del mismo modo que la red anterior, algunos valores de los errores obtenidos en la salida son:

```
 1.4482  -3.7629  -1.2410   2.3915
-3.5084 -14.2976  -8.1423  -8.9848
 3.5962  -5.9557 -11.8183 -15.0870
-12.9625  0.8402  -7.5224  20.4311
```

Figura 61. Ejemplos de errores cometidos con el ajuste de la red de 50 neuronas por capa.

Lamentablemente, independientemente del número de neuronas por capa que se seleccionaran, o del tipo de entrenamiento propuesto, no se llegó a conseguir una red neuronal que proporcionara valores a su salida aceptables. Esto puede explicarse debido a que el número de pesos que se necesitan ajustar en las redes que hemos utilizado es muy superior a la cantidad de datos que se han empleado como datos de muestra. En la red de la figura 52, que consta con dos capas ocultas de 50 neuronas cada una, más una capa de salida con 4 neuronas, el número de pesos es del orden de los 2700 (50x50+50x4=2700). Por lo tanto, debido a la gran cantidad de casos más que habría que medir, se ha optado por descartar la creación de una red neuronal artificial como medio de caracterización del comportamiento de los sensores infrarrojos.

7.4.3.2. Validación por coeficientes de corrección

En esta parte, seguiremos empleando los datos de la intensidad de los sensores infrarrojos obtenidos en el apartado 7.4.3.1. Utilizaremos la idea de la existencia de un sensor principal, es decir, aquel cuya intensidad de detección sea mayor para la posición del objeto a detectar, y sensores secundarios, aquellos que también detecten el cuerpo pero que posean unos niveles de intensidad menor.

Se comenzará obteniendo una representación gráfica de la variación de la intensidad de los diferentes sensores, tomando como datos el valor devuelto por únicamente en los casos en los que el objeto se halla situado sobre su línea de acción. Con esto obtenemos las 5 gráficas mostradas a continuación.

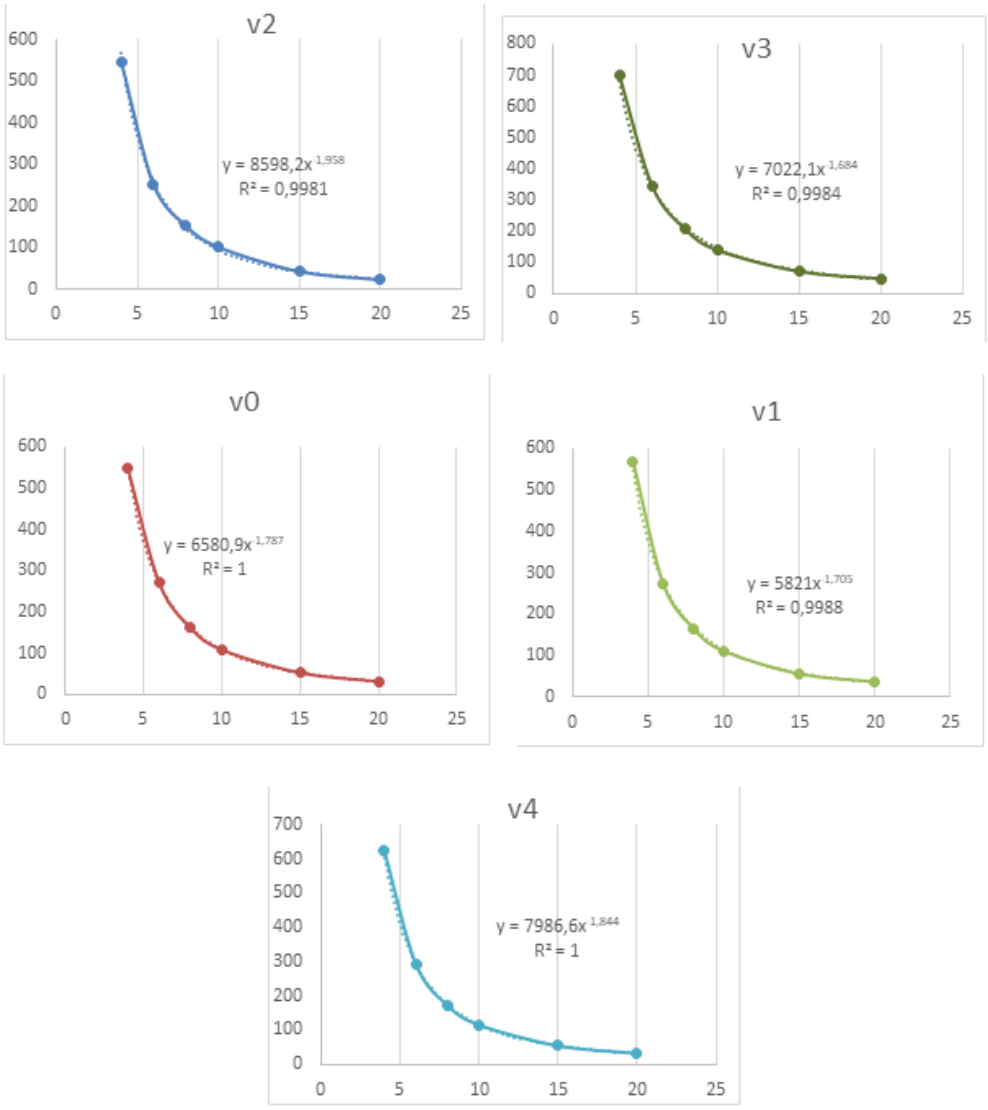


Figura 61. Representación de la intensidad de detección de los diferentes sensores en función de la distancia cuando el objeto se encuentra en su línea de acción.

Al igual que en el caso anterior, los valores v0, v1, v2, v3 y v4 hacen referencia a los sensores ordenándolos de izquierda a derecha.

En este apartado se procederá a crear un factor de corrección que ajuste los valores de los sensores secundarios para obtener así valores similares a los que se obtendrían en la realidad. Este factor de correlación viene dado por la siguiente expresión:

$$kps = \frac{vs - vst}{vpt - vst}$$

'vs' se corresponde con el valor obtenido al sustituir en la gráfica del sensor secundario la distancia entre dicho sensor y la arista del cubo más próxima a él (ds real).

'vpt' y 'vst' son los valores devueltos por el sensor principal y el secundario respectivamente.

La tabla 5 muestra los datos y resultados obtenidos para el caso en el que el sensor principal es el frontal central (2) y el secundario es el que está inmediatamente a su izquierda (1):

Tabla 5. Datos de partida y resultado obtenido en el cálculo del factor de corrección.

d2	d1 real	v2t	v1t	v1	k21
4	5,3	547,0	246,0	344,4	0,3
6	7,1	252,0	90,0	205,9	0,7
8	8,8	153,0	55,0	143,3	0,9
10	10,5	102,0	36,0	105,1	1,0
15	15,8	43,0	25,0	52,7	1,5
17	17,9	33,0	24,0	42,4	2,0

Siendo:

- d2 es la distancia a la que se encuentra el centro del cubo del sensor 2.
- v2t y v1t son los valores devueltos por los sensores.
- d1 real, en este caso, se refiere a la distancia existente entre el sensor secundario y la arista del cubo más cercana a él, en este caso, la arista izquierda (visto desde la posición del ROBOBO).
- v1 es el valor que se obtendría al sustituir en la gráfica correspondiente al sensor 1 el valor de la distancia real. Como se puede ver este es mayor que el devuelto por los sensores, esto es debido a que el objeto no está centrado en su línea de acción y por lo tanto, una mayor parte de los rayos que emite no rebotan en la superficie del cubo.
- k21 es el factor de corrección:

$$k21 = \frac{v1 - v1t}{v2t - v1t}$$

Este mismo procedimiento se realizaría con cada pareja de sensores contiguos. A la hora de representar las gráficas (Figura 62), esto se hará con respecto a la distancia entre el sensor principal y el centro del cubo, puesto que es la distancia más fácilmente calculable.

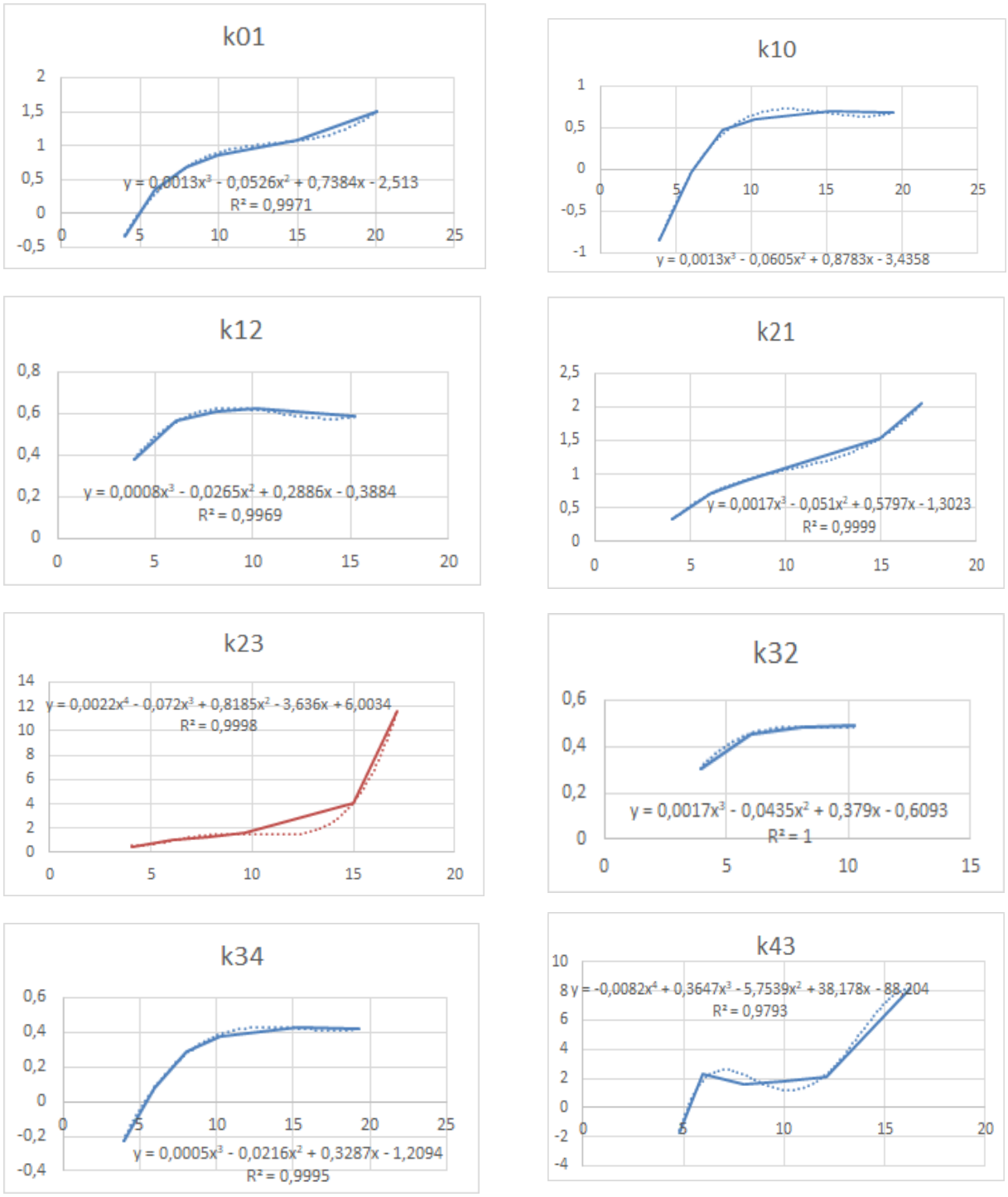


Figura 62. Coeficientes de correlación para todos los sensores.

Este factor de corrección se aplicaría al sensor secundario correspondiente para obtener un valor de intensidad del sensor corregido que al ser introducido en la gráfica de intensidad del sensor nos devuelva unos valores de las distancias más parecidos a los reales.

A continuación se mostrarán algunos de los errores cometidos con este método, para posiciones del objeto en las que se habían tomado las muestras:

Tabla 6. Ejemplos de errores cometidos empleando los coeficientes de correlación

v0	v1	v2	v3	v4	error d1 (%)	error d2 (%)	error d3 (%)
161	97	8	29	15	-2,82	-1,05	0,12
61	111	23	29	15	-1,66	-2	-1,43
15	36	102	50	16	2,98	3,6	0,2

7.5 Caracterización de los sensores infrarrojos del modelo de simulación

En este apartado nos encargaremos de caracterizar los sensores infrarrojos del modelo anterior, esto es, conseguir que nuestro modelo de simulación, al detectar un cubo de 10cm de arista en el entorno de simulación, obtenga valores semejantes a los que nos devolvería el ROBOBO real en un escenario equivalente.

Para caracterizar el funcionamiento de los sensores infrarrojos en el simulador, emplearemos la API externa de Python que ofrece el VREP. Se emplea este sistema de control y no el standard de VREP porque mediante el uso de una API externa se pueden emplear otras librerías o complementos que no están incluidos en VREP, como pueden ser las librerías incluidas en el paquete Anaconda de Python y que no se encuentran incluidas como tal en el VREP. Dentro de esta API externa, existen una serie de funciones, siendo interesante para nosotros las referentes a los sensores infrarrojos. En concreto, emplearemos la función, *simxReadProximitySensor* que nos permite conocer el estado de estos, que nos permite almacenar diferentes parámetros:

- *detectionState:* el valor *True* si ha detectado un objeto y *False* si no lo ha hecho.
- *detectedPoint:* las coordenadas del punto detectado (relativas al marco del sensor). Son estas coordenadas las que emplearemos para calcular la distancia a la que se situa el objeto detectado.
- *detectedObjectHandle:* el handle del objeto detectado.
- detectedSurfaceNormalVector: un vector normalizado perpendicular a la superficie detectada, en coordenadas relativas al marco del sensor.

Gracias a esta función se almacenará los valores de las distancias de aquellos puntos del objeto que estén más próximos a los sensores, puesto que en V-REP los sensores infrarrojos actúan como detectores de distancia mínimas. Consideraremos que el sensor principal es aquel que detecte el punto a menor distancia.

Teniendo en cuenta esto, se introducen las distancias obtenidas en las gráficas de valores de los sensores correspondientes (Figura 61) y se calculan los valores de los factores de

corrección (Figura 62), recordando emplear para estos últimos la distancia detectada por el sensor principal. Aplicando el factor de corrección, se obtienen los valores de los sensores estimados.

Llegados a este punto, pudimos comprobar que los valores obtenidos para los sensores secundarios difieren bastante de los reales. Esto se debe al hecho de que las distancias que se han tomado como referencia en el apartado anterior y en este no son las mismas, esto es, en la caracterización de los sensores, se ha empleado la distancia de los sensores reales a las aristas del cubo, pero en V-REP esto no tiene por qué ser cierto, ya que las arista del cubo es necesariamente el punto más cercano al sensor. En la Figura 63 se puede apreciar que ninguno de los sensores está detectando una arista del cubo (los puntos detectados son aquellos situados en el extremo de las líneas negras).

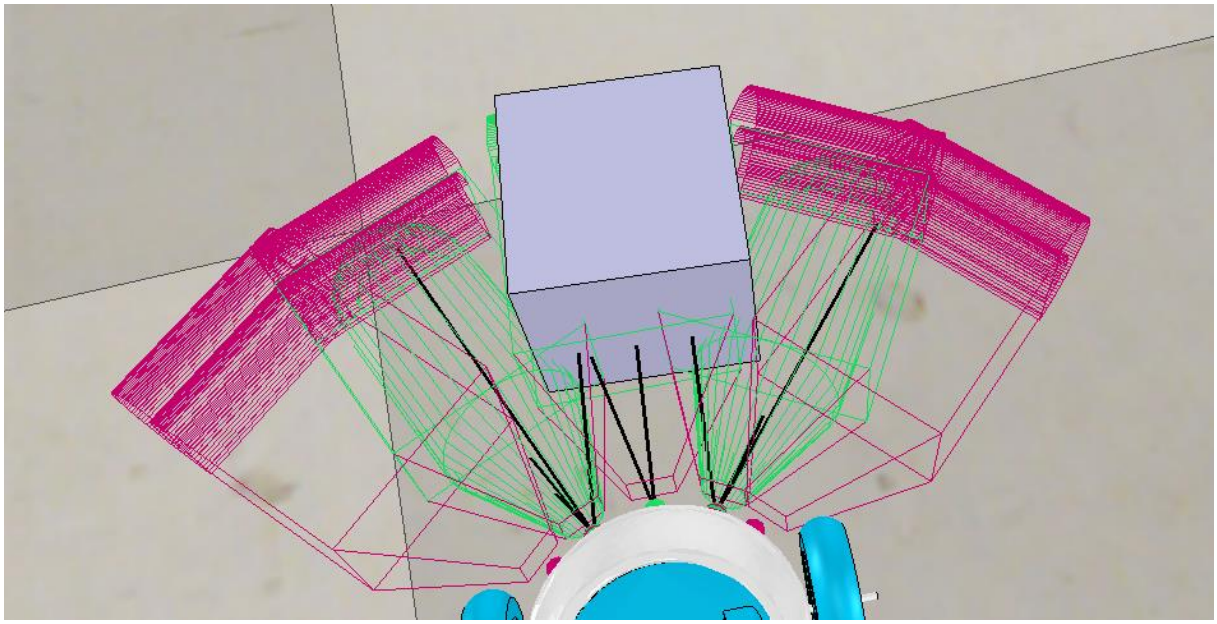


Figura 63. Objeto detectado en simulación.

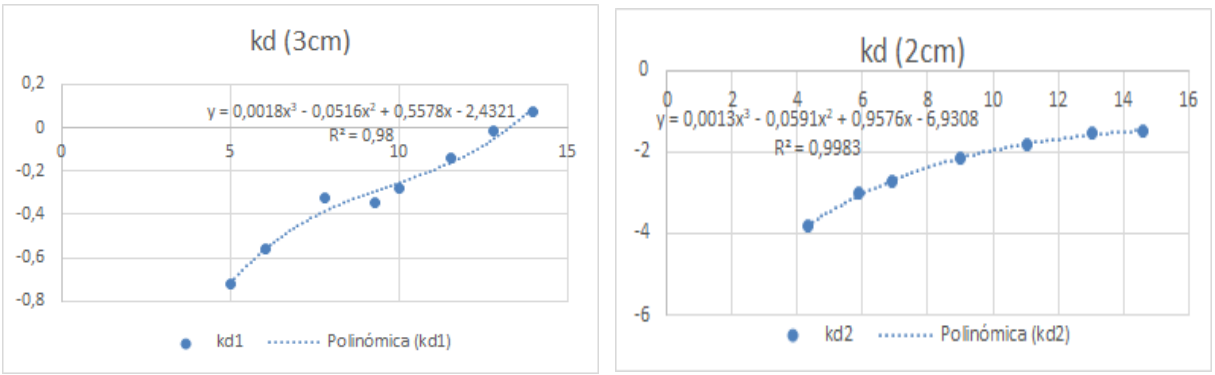
Con esto se dificulta el proceso, y será necesario introducir otro factor de corrección para asemejar en lo posible los valores obtenidos en un entorno real, con los que se puedan obtener en el V-REP. En este caso emplearemos un factor de corrección que modifique los valores de la distancia que nos devuelve el V-REP para que se asemejen a los reales. Vamos a trabajar con el objeto de simulación perpendicular a la línea de acción del sensor que se esté a evaluar. Como este parámetro depende principalmente de la geometría, esto es, la distancia entre los sensores (2 cm o 3 cm), vamos a suponer que solo será necesario crear dos factores de corrección. Se empleará uno u otro en función de cuál sea la distancia entre el sensor principal y el secundario a evaluar.

El factor de corrección que emplearemos vendrá dado por la siguiente expresión:

$$kd = \frac{d\ secundario\ real - d\ secundario\ simulación}{d\ principal - d\ secundario\ simulacion}$$

La distancia principal que nos devuelve el simulador sí que va a ser la del punto central, puesto que, al estar colocado el objeto en posición perpendicular, este equivaldrá a la distancia mínima.

A continuación, se ejecuta la simulación, desplazando el cubo a lo largo de la línea de acción de los sensores que nos interesan para obtener los valores de las distancias que nos ofrecen los sensores secundarios para distintos valores de los principales. Obteniendo así los coeficientes de corrección en función de la distancia del sensor principal indicados en la figura 64.



(a) (b)

Figura 64. Factores de corrección de distancia. (a) sensores separados 3cm. (b) sensores separados 2cm.

A continuación, incorporamos estos coeficientes al programa que hemos realizado, de modo que, después de almacenar las distancias medidas por los diferentes sensores, corrija los valores dados por los sensores secundarios, utilizando la expresión:

$$d\ secundario = \ d\ secundario\ sim + kd * (\ d\ principal - d\ secundario\ sim)$$

Después simplemente se procede a calcular con estas distancias los valores de los sensores, aplicando también los coeficientes de corrección necesarios. En la figura 65 se muestra una porción del código escrito en Pycharm, con comentarios que explican lo que se hace en cada sección. Este código se corresponde con el ajuste del sensor 1 cuando el sensor 0 es el principal. El resto de los casos serían equivalentes.

```
sensores = [14, 21, 7, 29, 15]  # offset


#previamente se calculado las distacias devueltas por cada sensor y se ha determinado el sensor principal
if nprinc == 0:
    #si el sensor principal es el 0, se calcula el valor de este sensor con su curva correspondiente:
    sensores[nprinc] = 6580.9 * distSensor[nprinc] ** (-1.787)
    if detecta[nprinc+1] == 1:
        #en caso de que el sensor contiguo detecte, se calcula su kd y la distancia corregida:
        kd = 0.0013 * distSensor[nprinc] ** 3 - 0.0591 * distSensor[nprinc] ** 2 + 0.9576 * distSensor[nprinc] - 6.9308
        distSensor[nprinc + 1] = distSensor[nprinc + 1] + kd * (distSensor[nprinc] - distSensor[nprinc + 1])

        #se calcula el valor del sensor previo a la aplicacion del factor de correcion
        senmas = 5821 * distSensor[nprinc + 1] ** (-1.705)
        #se calcula el factor de correccion empleando su expresión y se corrige el valor del sensor
        k = 0.0013 * distSensor[nprinc] ** 3 - 0.0526 * distSensor[nprinc] ** 2 + 0.7384 * distSensor[nprinc] - 2.513
        sensores[nprinc + 1] = (senmas - k * sensores[nprinc])/(1-k)
```

Figura 65. Ejemplo de proceso de cálculo para una de las parejas de sensores.

7.6 Determinación del comportamiento real de los motores de corriente continua.

En este apartado procederemos a determinar el comportamiento de los motores de corriente continua encargados del desplazamiento de la plataforma robótica. Estos motores, como hemos mencionado anteriormente, disponen de reducciones 150:1 y tienen un torque en vacío de 0.9kg-cm.

Se puede variar la velocidad de los motores del ROBOBO utilizando Scratch. La plantilla para su utilización que existe para esta plataforma, permite asignar un valor de 0 a 100 a los motores, siendo estos valores porcentajes de la velocidad máxima que pueden alcanzar. En la figura 66 se muestra una imagen del bloque de Scratch que representa el movimiento de los motores.

Figura 66. Bloque de Scratch para control de velocidad de motores.

La regulación de la velocidad se hace de esta forma por la dificultad que entraña especificar un valor constante para la velocidad. Existen numerosos factores que afectan a la velocidad de desplazamiento del ROBOBO, como es desplazarse por un plano inclinado, patinaje de las ruedas, peso del robot, etc. Por ello, para realizar este experimento situaremos el robot en una superficie plana, lisa y sin obstáculos, en la que marcaremos la posición inicial del ROBOBO. A continuación, lo conectaremos al Scratch y le asignaremos un valor a la velocidad de los motores. Podemos indicar también el tiempo que queremos que los motores estén

activos. Con esto, medimos la distancia que se ha desplazado para diferentes velocidades y durante tiempos diferentes, obteniendo los datos mostrados en la siguiente tabla.

Tabla 7. Distancias y velocidades en cm/s y en grados/s

velocidad 10 %				velocidad 20 %			
tiempo	**distancia**	**v (cm/s)**	**w(deg/s)**	**tiempo**	**distancia**	**v (cm/s)**	**w(deg/s)**
1	5,0	5,0	88,1	**1**	10,0	10,0	176,3
2	12,0	6,0	105,8	**2**	19,5	9,8	171,9
3	18,0	6,0	105,8	**3**	30,0	10,0	176,3
4	24,0	6,0	105,8	**4**	40,5	10,1	178,5
5	30,0	6,0	105,8	**5**	50,5	10,1	178,1

velocidad 30 %				velocidad 40 %			
tiempo	**distancia**	**v (cm/s)**	**w(deg/s)**	**tiempo**	**distancia**	**v (cm/s)**	**w(deg/s)**
1	13,5	13,5	238,0	**1**	16,5	16,5	290,9
2	27,5	13,8	242,4	**2**	34,0	17,0	299,7
3	41,0	13,7	240,9	**3**	51,5	17,2	302,6
4	55,0	13,8	242,4	**4**	69,0	17,3	304,1
5	70,5	14,1	248,6	**5**	87,5	17,5	308,5

velocidad 50 %				velocidad 60 %			
tiempo	**distancia**	**v (cm/s)**	**w(deg/s)**	**tiempo**	**distancia**	**v (cm/s)**	**w(deg/s)**
1	21,0	21,0	370,2	**1**	24,0	24,0	423,1
2	41,5	20,8	365,8	**2**	49,0	24,5	431,9
3	64,0	21,3	376,1	**3**	74,5	24,8	437,8
4	85,0	21,3	374,6	**4**	100,0	25,0	440,7
5	106,0	21,2	373,7	**5**	121,0	24,2	426,6

velocidad 70 %				velocidad 80 %			
tiempo	**distancia**	**v (cm/s)**	**w(deg/s)**	**tiempo**	**distancia**	**v (cm/s)**	**w(deg/s)**
1	27,5	27,5	484,8	**1**	31,0	31,0	546,5
2	55,5	27,8	489,2	**2**	64,0	32,0	564,1
3	83,0	27,7	487,7	**3**	96,0	32,0	564,1
4	112,0	28,0	493,6	**4**	128,0	32,0	564,1
5	140,0	28,0	493,6	**5**	160,0	32,0	564,1

velocidad 90 %				velocidad 100 %			
tiempo	**distancia**	**v (cm/s)**	**w(deg/s)**	**tiempo**	**distancia**	**v (cm/s)**	**w(deg/s)**
1	34,5	34,5	608,2	**1**	41,0	41,0	722,8
2	70,0	35,0	617,0	**2**	81,0	40,5	714,0
3	108,0	36,0	634,7	**3**	118,0	39,3	693,4
4	141,5	35,4	623,6	**4**	157,5	39,4	694,2
5	178,5	35,7	629,4	**5**	197,0	39,4	694,6

Será necesario calcular la velocidad angular a partir de la lineal, puesto que los motores de V-REP trabajan con ella. Para ello se tiene en cuenta que el diámetro de las ruedas del ROBOBO es de 6.5 cm.

Se asignará como resultado de velocidad para cada porcentaje, el valor promedio de los obtenidos para los diferentes tiempos. Además, al V-REP trabaja en radianes por segundo, por lo que en la gráfica que represente el comportamiento de los motores (Figura 67), se utilizarán estas unidades.

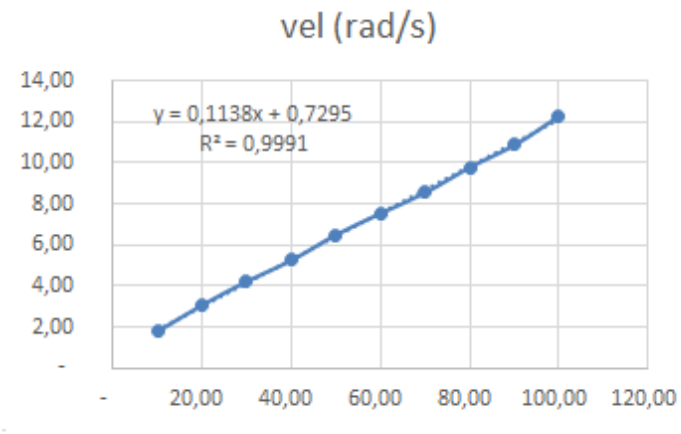


Figura 67. Velocidad de los motores en radianes por segundo en función de los valores del Scratch.

Como se puede ver, los motores presentan un comportamiento lineal. En el próximo sub-apartado, intentaremos transmitir este comportamiento al V-REP.

7.7 Modelado de los motores de corriente continua en V-REP.

En este apartado se procederá adaptar los resultados obtenidos en el apartado 7.6 para conseguir que el robot del modelo de simulación se comporte de forma similar al real.

Para conseguir este objetivo, lo primero que se hará sea crear un pequeño programa que permita controlar los motores mediante una API externa. En este caso, y como se ha comentado antes, se emplea el lenguaje de programación Python con el IDE PyCharm. Este programa permite trasmitir la velocidad angular correcta a los motores del modelo según la ecuación mostrada en la figura 67 y que nos devuelva la distancia recorrida por éste en el periodo de tiempo indicado. Tras ejecutarlo, se pudo comprobar que la distancia recorrida por la plataforma en la simulación es muy superior a la real y por ello será necesario introducir un coeficiente de corrección. Se ha decidido calcular este coeficiente como una relación entre las

distancias real y de simulación, puesto que esta relación debería de mantenerse con las velocidades si el tiempo en los dos casos es el mismo:

$$coef = \frac{d_{real}}{d_{sim}} = \frac{v_{real}\, t}{v_{\ sim}\, t}$$

En la tabla 8 se muestran los resultados obtenidos para este coeficiente, junto con las distancias medias obtenidas en varias simulaciones previas a la aplicación del mismo.

Tabla 8. Distancias de simulación y coeficientes de corrección de la misma.

velocidad 10 %				velocidad 30%			
time	**distance REAL**	**distance SIM**	**coef**	**time**	**distance**	**distance SIM**	**coef**
1	5,25	17,2	0,291	**1**	13,5	38	0,355
2	12	35,4	0,339	**2**	27,5	79,7	0,345
3	18	52,6	0,342	**3**	41	113,6	0,361
4	24	70,0	0,343	**4**	55	163,1	0,337
5	30	89,7	0,334	**5**	70,5	203,7	0,346

velocidad 60%				velocidad 100 %			
time	**distance**	**distance SIM**	**coef**	**time**	**distance**	**distance SIM**	**coef**
1	24	70,9	0,339	**1**	41	116	0,353
2	49	144,2	0,340	**2**	81	236,3	0,343
3	74,5	217,8	0,342	**3**	118	339	0,348
4	100	285	0,351	**4**	157,5	440	0,358
5	121	354,3	0,342	**5**	197	568,2	0,347

Como se puede observar en esta tabla, los valores de los coeficientes son muy similares para los distintos casos. Por este motivo y el hecho de que tanto el robot real como su modelo de simulación no se comporten del mismo modo en cada repetición, se ha decidido calcular el coeficiente de ajuste como un promedio de todos los obtenidos anteriormente.

$$coef = 0.343$$

Este coeficiente se incorpora a nuestro programa de Phyton (Figura 68) multiplicándolo por la velocidad obtenida con la ecuación de la figura 67.

```python
import vrep
import vrepConst
import math

import time

vrep_host = '127.0.0.1'
vrep_port = 19997

vrep.simxFinish(-1)
clientID = vrep.simxStart(vrep_host, vrep_port, True, True, 5000, 5)  # Connect to V-REP
if clientID != -1:
    vrep.simxStartSimulation(clientID, vrep.simx_opmode_oneshot)
    print ('Connected to remote API server')
else:
    print ('Failed connecting to remote API server')

#Valores introduciriamos en scratch
vel_izq_scratch=10
vel_der_scratch=10
t=1
#coeficiente de modificacion de velocidad  obtenido
coef = 0.343

#pasamos estas velocidades a velocidades angulares en rad/s
vel_izq = coef*(0.1138*vel_izq_scratch+0.7295)
vel_der = coef*(0.1138*vel_der_scratch+0.7295)



#  Obtencion de handlers
returnCode, leftMotorHandler = vrep.simxGetObjectHandle(clientID, 'motor_izquierdo', vrep.simx_opmode_blocking)
returnCode, rightMotorHandler = vrep.simxGetObjectHandle(clientID, 'motor_derecho', vrep.simx_opmode_blocking)
returnCode, objeto = vrep.simxGetObjectHandle(clientID, 'Cuboid0', vrep.simx_opmode_blocking)
returnCode, carcasaHandler = vrep.simxGetObjectHandle(clientID,'carcasa0',vrep.simx_opmode_blocking)

time.sleep(0.5)

# Primer uso de las funciones

returnCode,posicion_inicial = vrep.simxGetObjectPosition(clientID,carcasaHandler,-1,vrep.simx_opmode_streaming)


time.sleep(0.05)

valor_medio = 4
array_distancia = [0 for n in range(0, valor_medio)]
for n in range(0, valor_medio):
    # Segunda lectura de la posicion
    returnCode, posicion_inicial = vrep.simxGetObjectPosition(clientID, carcasaHandler, -1, vrep.simx_opmode_buffer)
    print "Posicion inicial", posicion_inicial
    targetVelocity = [-vel_izq, -vel_der]
    returnCode = vrep.simxSetJointTargetVelocity(clientID, leftMotorHandler, targetVelocity[0], vrep.simx_opmode_oneshot)
    returnCode = vrep.simxSetJointTargetVelocity(clientID, rightMotorHandler, targetVelocity[1], vrep.simx_opmode_oneshot)
    time.sleep(t)
    print "velocidad", targetVelocity
    targetVelocity = [0, 0]
    returnCode = vrep.simxSetJointTargetVelocity(clientID, leftMotorHandler, targetVelocity[0],vrep.simx_opmode_oneshot)
    returnCode = vrep.simxSetJointTargetVelocity(clientID, rightMotorHandler, targetVelocity[1],vrep.simx_opmode_oneshot)
    time.sleep(1)
    returnCode,posicion_final = vrep.simxGetObjectPosition(clientID,carcasaHandler,-1,vrep.simx_opmode_buffer)

    returnCode,end_position = vrep.simxGetObjectPosition(clientID,carcasaHandler,-1,vrep.simx_opmode_streaming)

    print "Posicion final", end_position

    array_distancia[n] = posicion_inicial[0] - end_position[0]
    print "Distance:", array_distancia[n]

average = sum(array_distancia)/valor_medio

print "Valor medio distancia:", average

time.sleep(1)
```

Figura 68. Programa Python para el control de los motores del modelo de simulación.

Se ha empleado el coeficiente obtenido para calcular las distancias recorridas por el ROBOBO de simulación para diferentes valores de velocidades y tiempos. En vista de los errores mostrados en la Tabla 9, se ve que este modelo sirve para representar el comportamiento de los motores del ROBOBO.

Tabla 9. Distancias y errores obtenidos en la simulación.

velocidad (%)	tiempo (s)	distancia real (cm)	distancia sim (cm)	error
10	2	12	12,21	1,75%
10	4	24	24,1	0,42%
30	1	13,5	13,8	2,22%
30	5	70,5	72,9	3,40%
60	3	74,5	76,6	2,82%
60	5	121	123,2	1,82%
100	1	41	42,1	2,68%
100	4	157,5	163,1	3,56%

8 RESULTADOS

En esta parte de la memoria se van a dar a conocer los resultados obtenidos del modelo de simulación y la valoración de los mismos.

8.1 Resultado de la creación de los modelos reales.

En este apartado se comentarán diferentes resultados obtenidos durante la creación de los modelos de caracterización de los sensores reales.

Durante la realización de este trabajo, se ha determinado que el valor devuelto por los sensores infrarrojos depende no sólo de la morfología del objeto sino que también de su orientación (apartado 7.4.2).

Los valores obtenidos en la tabla 6, nos muestran que el modelo de ajuste del comportamiento real para una posición y morfología conocida, es bastante preciso.

Para demostrar que la orientación del objeto es un factor clave en el valor de los sensores, colocamos el cubo de aristas de 10 cm, girado 45° y con la arista central a 4cm de distancia del ROBOBO, tal y como se ve en la figura 69.

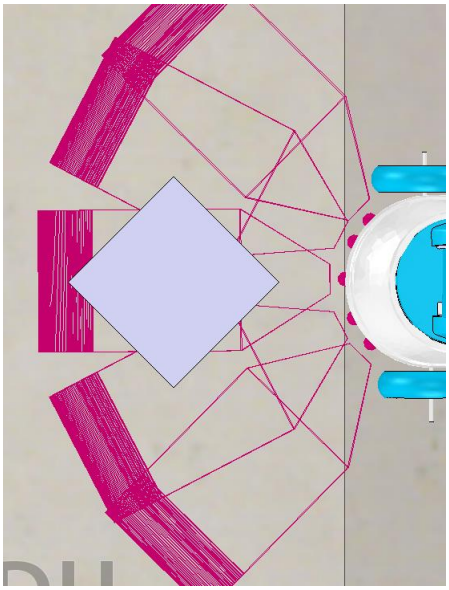


Figura 69. Representación del ensayo realizado.

En esta situación, los valores devueltos por los sensores infrarrojos son introducidos en los diferentes métodos de caracterización que se han ido creando en el desarrollo de este trabajo para conocer los valores de la distancia que nos devolverían en este caso.

- Un único sensor aislado situaría el objeto a 8.11 cm de distancia.
- La red neuronal artificial nos localiza el punto central a una distancia de 12.1 cm y los dos laterales a 13.15 cm.
- El método de validación mediante factores de corrección devuelve unas distancias de 8.5 cm para el punto central, y de 9.3 y 9.8 para los otros dos puntos.

Como se puede ver, el valor calculado que nos ofrecen los diferentes métodos varía notablemente entre ellos, al mismo tiempo que con esta disposición, nos ofrecen una valor erróneo de la posición más cercana del sólido, pues este se encuentra a 4cm y no a 8cm ni a 12cm.

8.2 Resultados del modelo de simulación de los sensores infrarrojos.

En este apartado se comentarán diferentes aspectos observados acerca del modelo de simulación de los sensores infrarrojos.

Para las distancias cortas este modelo ajusta relativamente bien los valores de los sensores, no obstante si se dispone de un margen de error elevado. A continuación, se muestran dos tablas, una con ejemplos de valores reales devueltos por de los sensores del ROBOBO número P07 (Tabla 10) y los resultados obtenidos en Pycharm (Tabla 11)

Tabla 10. Datos de los sensores del ROBOBO P07 en distancias cortas.

d(cm)	v1	v2	v3	v4	v5
4	18	248	547	306	23
6	16	90	252	130	19
8	15	55	153	68	17

Tabla 11. Valores obtenidos con Pycharm en distancias cortas.

d(cm)	v1	v2	v3	v4	v5
4	18	258	569	311	15
6	14	108	257	148	15
8	14	71	146	58	15

Para distancias mayores, el modelo presenta un comportamiento erróneo. Se muestran los resultados en las tablas siguientes.

Tabla 12. Datos de los sensores del ROBOBO P07 en largas distancias.

d(cm)	v1	v2	v3	v4	v5
10	15	36	102	50	16
15	15	25	43	35	15
20	15	22	23	31	16

Tabla 13. Datos obtenidos con Pycharm.

d(cm)	v1	v2	v3	v4	v5
10	14	21	94,7	29	15
15	14	21	42,81	29	15
20	14	21	24,37	29	15

Además, se ha comprobado que realizando distintas mediciones con distintos ROBOBOs, los valores que ofrecen los sensores infrarrojos varían considerablemente de uno a otro. Como un ejemplo, se muestran los datos de los sensores obtenidos del ROBOBO P08.

Tabla 14. Datos de los sensores del ROBOBO P08

d(cm)	v1	v2	v3	v4	v5
4	25	208	520	247	38
6	23	95	259	130	32
8	23	68	156	89	29

Por esto, para futuros trabajos se recomienda utilizar siempre el mismo robot o encontrar algún método que permita incluir las variaciones en las medidas que ofrecen las diferentes unidades de ROBOBO en el modelo de simulación.

8.3 Resultados del modelo de simulación de los motores de corriente continua.

En esta sección se analizarán los resultados obtenidos por el modelo de simulación de los motores de corriente continua del ROBOBO.

Como se mostró en el apartado 7.7, el ajuste empleado ofrece buenos resultados para aquellas velocidades que se utilizaron para calcular el coeficiente de ajuste. A modo de validación del mismo, se realizaron ensayos para valores de velocidad diferentes a las anteriores, obteniendo los resultados que se muestran en la tabla 15.

Tabla 15. Distancias y errores obtenidos en la simulación.

velocidad (%)	tiempo (s)	distancia real (cm)	distancia sim (cm)	error
20	2	19,5	19,7	1,026%
30	3	41	41,3	0,732%
40	4	69	67,5	-2,174%
50	5	106	106,8	0,755%
70	1	27,5	28,1	2,182%
80	3	96	94,9	-1,146%
90	5	178,5	177,3	-0,672%

Los errores cometidos con este método son muy bajos incluso para velocidades que no han formado parte del ajuste, por lo que podemos de determinar que nuestro modelo de simulación de los motores es válido.

9 CONCLUSIONES

Una vez realizado el trabajo para la consecución del objetivo principal de este proyecto y tras el análisis de los resultados obtenidos una vez finalizado el desarrollo del mismo, en este capítulo se repasarán los objetivos propuestos inicialmente para evaluar en qué medida se han llegado a cumplir.

El objetivo global del presente trabajo fin de grado era el de realizar un modelo de simulación, para ello se subdividió el mismo en una serie de objetivos secundarios. El primero de ellos consistía en determinar el comportamiento real de los sensores infrarrojos. Tras los experimentos realizados para calcular la distancia del ROBOBO a diferentes objetos con diferente forma y morfología hemos sido capaces de determinar el funcionamiento del sensor infrarrojo en la realidad, y la gran influencia que tiene la morfología del objeto a detectar, así como su orientación. Por otra parte, el traslado de la información obtenida sobre el funcionamiento del sensor real al VREP requiere de gran complejidad. No obstante, se ha creado una versión mejorada de la distribución de los sensores infrarrojos con respecto al trabajo previamente realizado. Está es más sencilla y abarca un volumen de detección más cercano al real del comparado con el modelo anterior. Además se han analizado dos modelos matemáticos para determinar a qué distancia se encuentran los objetos de los sensores infrarrojos, tanto en la realidad como en el VREP.

Con respecto a los objetivos que conciernen a los motores, la determinación del funcionamiento real del motor de corriente continua se ha realizado sin tener en cuenta el controlador. Esto es así porque, considerando el intervalo de tiempo recomendado para trabajar con el simulador (50ms) más las latencias entre la comunicación que pueda haber entre la API externa y el VREP, hacen que considerar el efecto del controlador del motor no tenga relevancia en la simulación, pues el motor estabiliza aproximadamente en 100ms. Por otra parte, se ha realizado un modelo de simulación del funcionamiento de los motores para representar el desplazamiento del ROBOBO en el VREP.

Por lo tanto, y a la vista de los resultados, se puede considerar que se ha cumplido con los objetivos planteados en el presente trabajo fin de grado.

10 REFERENCIAS

1. LEGO Mindstorms. https://www.lego.com/es-es/mindstorms
2. AIBO. https://elpais.com/tecnologia/2017/11/07/actualidad/1510056060_501578.html
3. Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., Higaki, N., & Fujimura, K. (2002). The intelligent ASIMO: System overview and integration. In Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on (Vol. 3, pp. 2478-2483). IEEE.
4. NAO. http://aliverobots.com/
5. Sophia. http://sophiabot.com/about-me/
6. THYMIO. Riedo, F., Chevalier, M., Magnenat, S., & Mondada, F. (2013, November). Thymio II, a robot that grows wiser with children. In *Advanced Robotics and its Social Impacts (ARSO), 2013 IEEE Workshop on* (pp. 187-193). IEEE.
7. KEPHERA. https://www.k-team.com/mobile-robotics-products/khepera-iv/introduction
8. ROMO. http://www.tecnoadicto.net/romo-robot-iphone.html
9. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In ICRA workshop on open source software (Vol. 3, No. 3.2, p. 5)
10. SMARTBOT. https://www.xatakamovil.com/aplicaciones/smartbot-un-pequeno-robot-programable-basado-en-tu-smartphone
11. ODDWERX. http://www.oddwerx.com/
12. WHEELPHONE. http://www.wheelphone.com/
13. V-REP. Rohmer, E., Singh, S. P., & Freese, M. (2013, November). V-REP: A versatile and scalable robot simulation framework. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on* (pp. 1321-1326). IEEE.
14. Pycharm. https://www.jetbrains.com/pycharm/
15. Anaconda. https://anaconda.org/anaconda/python
16. RESNICK, Mitchel, et al. Scratch: programming for all. *Communications of the ACM*, 2009, vol. 52, no 11, p. 60-67.