UNIVERSIDADE DA CORUÑA

Escola Politécnica Superior

Trabajo Fin de Máster

CURSO 2018/19

DESARROLLO DE UN MODELO EN SIMULACIÓN EN V-REP DE UN ROBOT MÓVIL BASADO EN SMARTPHONE Y SOPORTE AL LENGUAJE PYTHON

Máster en Ingeniería Industrial

ALUMNO

Rafael Boado de la Fuente

TUTORES

Abraham Prieto García

Pedro José Trueba Martínez

FECHA

JUNIO 2019

1 TÍTULO Y RESUMEN

1.1 Título

Desarrollo de un modelo en simulación en V-REP de un robot móvil basado en smartphone y soporte al lenguaje Python.

1.2 Resumen

El, cada vez más presente, uso de robots en aplicaciones industriales, científicas o educativas, ha popularizado el uso de modelos de simulación para realizar experimentos con robots en distintos escenarios virtuales sin necesidad de tener el robot físico, evitando así posibles problemas derivados de probar el robot en entornos reales como por ejemplo choques, ambientes agresivos, caídas, etc. Estos modelos también permiten simular el comportamiento de los robots en entornos en los que, por distintas razones, no se podrían probar, y realizar las pruebas de forma más rápida

En cuanto a la robótica educativa, aparte de las ventajas comentadas anteriormente, estos modelos, con una programación idéntica al robot real, mejoran la accesibilidad de los alumnos a los robots al permitirles acceder a los conocimientos relacionados con su uso sin necesidad de que exista un robot por cada uno. Además, en el caso de este tipo de usuarios, es especialmente interesante aumentar la seguridad del robot debido al desconocimiento del funcionamiento del robot o su programación.

Este Trabajo de Fin de Máster tiene como objetivo desarrollar un modelo realista del Robobo real en el simulador V-REP, de forma que sus movimientos y su sensorización sean análogos a los del Robobo real.

Título

Desenvolvemento dun modelo en simulación en V-REP dun robot móvil basado en smartphone e soporte ao lenguaxe Python.

Resumo

O, cada vez máis presente, emprego de robots en aplicacións industriais, científicas ou educativas, popularizaron o uso de modelos de simulación para realizar experimentos con robots en distintos escenarios virtuais sen necesidade de ter o robot físico, evitanto así posibles problemas derivados de probar o robot en entornos reais como, por exemplo, choques, ambientes agresivos, caídas, etc. Estos modelos tamén permiten simular o comportamento dos robots en entornos nos que, por distintas razóns, non se poderían probar, e realizar as probas de maneira máis rápida.

En canto á robotica educativa, aparte das vantaxes comentadas anteriormente, estes modelos, cunha programación idéntica ao robot real, melloran a accesibilidade dos alumnos aos robots ao permitirlles acceder aos coñecementos relacionados co seu uso sen necesidades de que exista un robot para cada un. Ademáis, no caso deste tipo de usuarios, é especialmente interesante aumentar a seguridade do robot debido ao descoñecemento do funcionamento do robot e a súa programación..

Este Traballo de Fin de Máster ten como obxectivo desenvolver un modelo realista do Robobo real no simulador V-REP, de forma que os seus movementos e a sua sensorización sexan análogos aos do Robobo real.

Title

Development of a V-REP simulation model of a mobile robot based on smartphone and Python language support

Summary

The increasingly use of robots in industrial, scientific or educational applications, has popularized the use of simulation models to perform experiments with robots in different virtual scenarios without having the physical robot, which avoide possible problems derived from testing the robot in real environments such as shocks, aggressive environments, falls, etc. These models also allow to simulate the behavior of robots in environments where, for different reasons, they could not be tested, and test the experiments more quickly.

In terms of educational robotics, apart from the advantages discussed above, these models, with a programming identical to the real robot, improve the accessibility of students to robots by allowing them to access knowledge related to their use without needing a robot for each one. In the case of this type of users, it is especially interesting to increase the safety of the robot due to the lack of knowledge of the robot's operation or its programming.

This work aims to develop a realistic model of the real Robobo in the V-REP simulator, so that its movements and sensorization are analogous to those of the real Robobo.

UNIVERSIDADE DA CORUÑA

Escola Politécnica Superior

TRABAJO FIN DE GRADO/MÁSTER

CURSO 2018/2019

DESARROLLO DE UN MODELO EN SIMULACIÓN EN V-REP DE UN ROBOT MÓVIL BASADO EN SMARTPHONE Y SOPORTE AL LENGUAJE PYTHON

Máster en Ingeniería Industrial

MEMORIA

2 ÍNDICES

Índice de contenido:

Índice de figuras:

Índice de tablas:

3 INTRODUCCIÓN

Este Trabajo de Fin de Máster se enmarca en el campo de la robótica autónoma, en particular en la robótica educativa, y más especialmente en el proyecto Robobo [1].

Para empezar, la robótica autónoma está definida como el área de la robótica que desarrolla robots capaces de desplazarse y actuar sin intervención humana [2].

A partir de la robótica autónoma se puede definir un robot autónomo como un robot que puede actuar de forma autónoma, obteniendo información sobre el medio que lo rodea, actuando durante un tiempo prolongado sin intervención humana y aprendiendo y adquiriendo nuevos conocimientos. Esto implica que el robot debe percibir el entorno para actuar y comportarse de forma adecuada, lo cual hace necesario que el robot esté dotado de sensorización (sensores de presión, infrarrojos, cámaras, etc.).

La Federación Internacional de Robótica (IFR, International Federation of Robotics) reconoce dos tipos de robots atendiendo a su aplicación: los robots industriales y los de servicios. A mayores, los robots de servicio se dividen entre los de uso personal/doméstico y los de uso profesional.

3.1 Robots industriales: Industria 4.0

En la industria, el avance de la robótica, las necesidades de mejora de la calidad de la producción y el aumento de la eficiencia de producción son elementos que están muy relacionados entre sí, y el logro de cada uno depende en gran medida del logro del resto.

En los últimos años, la necesidad de mejorar la calidad de la producción, así como abaratar los costes, están siendo factores clave para las empresas para ganar competitividad en el mercado. Partiendo de estas necesidades, en 2011 se acuño el término de *Industria 4.0* que fue una iniciativa alemana del gobierno federal con universidades y empresas privadas. Fue un programa estratégico para desarrollar sistemas de producción avanzados con el objetivo de aumentar la productividad y la eficiencia de la industria nacional [3].

En este contexto gana un gran protagonismo la robótica industrial. Por un lado, la robotización de los procesos de producción y la automatización de los robots industriales serán muy importantes para incrementar la calidad de la producción, reducir costes y tiempos. Por otro lado, la inclusión de robots inteligentes con capacidad de aprender y adaptarse a los problemas que puedan surgir en una planta serán fundamentales para llevar a cabos trabajos más complejos.

Deloitte Touche Tohmatsy Limited, una de las principales firmas privadas de servicios profesionales a nivel mundial, publica en 2017 un informe donde define la Industria 4.0 como “una revolución que combina técnicas avanzadas de producción y operaciones con tecnologías inteligentes que se integrarán en las organizaciones, las personas y los activos”. Esta revolución se ha producido gracias a la aparición de nuevas tecnologías como la robótica, la inteligencia artificial, el “Internet of Things”, el “Big Data Análysis”, etc [4].

Es evidente que una parte importante de la Industria 4.0 implica la automatización, sensorización y robotización de la planta para mejorar y optimizar los procesos de producción, y, por lo tanto, la robótica industrial va a seguir incrementándose en los próximos años. Confirmando esta idea, la IFR, en el informe de ventas publicado en 2018 (Executive Summary World Robotics 2018 Industrial Robots), estima que el número de unidades de robots industriales previstas en 2021 alcanzará los 3,8 millones en todo el mundo [5] como se puede ver en la figura 1.

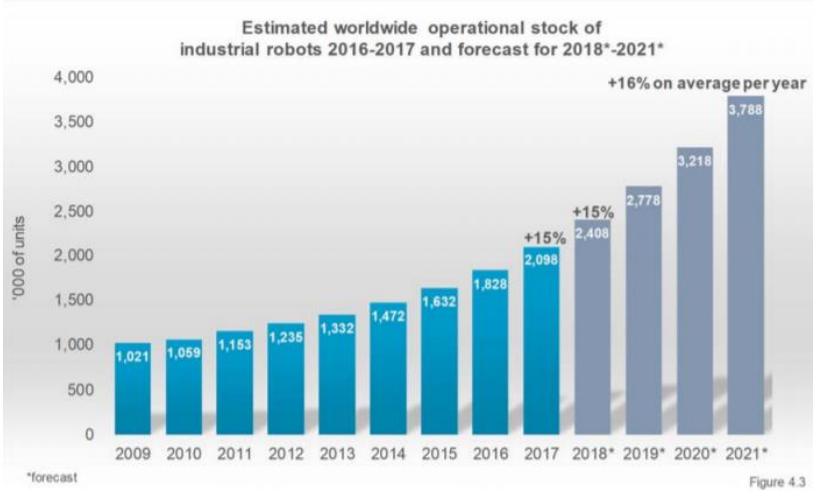

Figura 1: Previsión de unidades de robots industriales existentes hasta 2021.

3.2 Robots de servicio: Educación

Según la IFR, los robots de servicios pueden ser clasificados en función de su uso: personal/doméstico o profesional. Dentro de los robots de uso personal destacan los robots de tareas domésticas, de entretenimiento, de asistencia, etc. Por otro lado, la categoría de robots de uso profesional abarca los robots agrícolas, de limpieza profesional, inspección y mantenimiento de sistemas, construcción y demolición, médicos o militares, entre otros.

En concreto, y debido a que este Trabajo de Fin de Máster se enmarca en la robótica autónoma aplicada en la educación, este apartado introducirá el empleo de robots en la enseñanza.

La robótica educativa está basada en la teoría constructivista de Jean Piaget que propone que el conocimiento es un proceso activo de construcción del aprendizaje basado en experiencias [6].

Entre los objetivos de la robótica educativa destaca la participación activa de los alumnos en el proceso de aprendizaje, el desarrollo del razonamiento, desarrollo de la comprensión básica de programación o el trabajo cooperativo, al mismo tiempo que combinan el entretenimiento con el aprendizaje de forma lúdica y acercan las nuevas tecnologías a los más jóvenes. Actualmente, la robótica educativa está cada vez más presente en colegios e institutos, y por supuesto en universidades.

Actualmente, existe una gran variedad de robots educativos como por ejemplo el LEGO Mindstorms, el NAO, el KEPHERA o el EPUCK, entre muchos otros. En la figura 2 se puede ver un modelo de robot LEGO Mindstorms (izquierda) y del robot NAO (derecha).

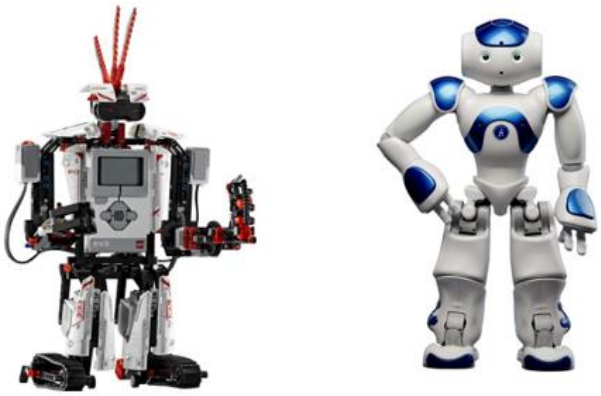

Figura 2: Modelos LEGO Mindstorms y NAO.

En esta línea, y centrándose más en el contexto de este trabajo, la empresa MINT, una spin-off del Grupo Integrado de Ingeniería (GII) de la Universidad de A Coruña (UDC), ha desarrollado desde el año 2016 una plataforma robótica para smartphone que ha culminado con el desarrollo del proyecto Robobo.

Robobo es un robot educativo basado en una base robótica móvil y un smartphone, que aprovecha la tecnología en sensores y procesadores del mismo. Robobo se ha convertido en una herramienta óptima para comenzar el aprendizaje en la robótica autónoma y está destinado tanto a estudiantes de secundaria como de bachillerato con alguna experiencia en robótica y programación [7]. En la figura 3 se observa una imagen del modelo de Robobo.

Figura 3: Robot Robobo.

3.3 Modelo de simulación de robots

Un modelo de simulación de un robot es una representación digital del robot real, que se comporta exactamente igual que él. Hay simuladores 2D y 3D, sin embargo, por cuestiones de realismo en la simulación, los 3D son los más empleados actualmente.

Centrándose ya en la robótica educativa, un modelo de simulación que represente al modelo real de forma fidedigna y que sea programable de la misma forma implica una serie de ventajas desde el punto de vista didáctico. La primera y más importante, supone una reducción de costes en modelos físicos, permitiendo a los alumnos acceder al material didáctico sin necesidad de que haya un robot para cada uno, lo cual en aulas con un gran número de estudiantes es muy significativo. Además, el uso de un modelo digital que funcione exactamente igual al real permite a los alumnos familiarizarse con el robot y su programación, de forme que cuando se emplee el robot real, los riesgos de averías debidas a un mal uso del robot o al desconocimiento del mismo sean mínimas. Por último, los modelos de simulación implican también una reducción de los tiempos de testeo en experimentación, sirven de base sobre la que probar de algoritmos de optimización, etc.

Hoy en día existe un gran número de simuladores robóticos 3D como el V-REP, Gazebo, Webots, ARGos, Aseba o Marilou Robotics Studio, entre otros.

En este contexto, el Grupo Integrado de Ingeniería de la UDC disponía de un modelo de simulación básico. En este proyecto se propone desarrollar un modelo de simulación preciso del comportamiento del Robobo. Para ello se implementan los modelos virtuales de todos los sensores de los que dispone el robot, así como los motores, y posteriormente se calibran con respecto a la respuesta del Robobo real para maximizar la fidelidad de la simulación. La idea es que el modelo sea complementario con el robot físico y que sirva de ayuda en las tareas didácticas para las que fue diseñado el Robobo, aprovechando las ventajas que ya se han comentado en este apartado acerca del modelo en simulación.

Entre todos los simuladores existentes, el GII ha elegido V-REP (Virtual Robot Experimentation Platform) para llevar a cabo el modelo. En la figura 4 se muestra el modelo ya exitente de Robobo en V-REP.

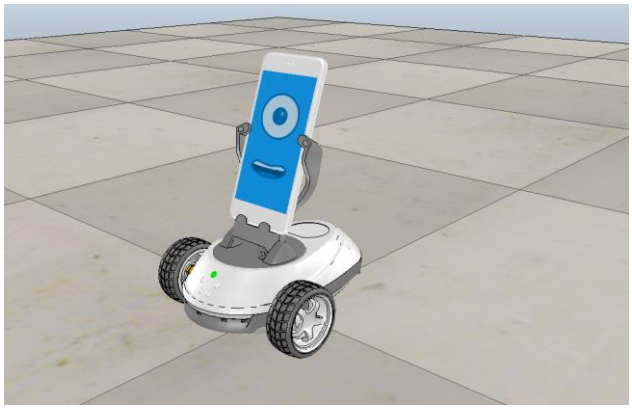

Figura 4: Modelo de Robobo en V-REP.

3.4 Simulador V-REP

En este apartado se explicará de forma introductoria qué es V-REP. Para una descripción más exhaustiva consutar el apartado 6.

V-REP es un simulador robótico desarrollado por la empresa Coppelia Robotics y actualmente es ampliamente utilizado con fines pedagógicos [8]. Esto es debido a que es un simulador muy versátil, que permite crear simulaciones 3D con facilidad y gratuito para fines educativos. Además, V-REP se basa en una arquitectura de control distribuida, lo que quiere decir que cada objeto en una simulación se puede controlar de forma individual por medio de un script, plugin, API (Application Programming Interface) externa, etc. Esto permitiría controlar varios robots de forma independiente posibilitando, entre otras cosas, la realización de experimentos de robótica colectiva.

La simulación de un modelo en V-REP se puede programar de varias formas: con scripts, plugins, add-on, API externa o interna, etc. Sin embargo, el uso de scripts, utilizando Lua como lenguaje de programación, es el método más fácil. Por otro lado, la API externa (remote API) tiene la ventaje de permitir controlar la simulación con varios lenguajes de programación como Python, C++, Java o Matlab, de forma externa a V-REP. En el apartado 6.2.4 se explicará de forma más detallada los métodos de programación de V-REP.

Las aplicaciones más habituales de V-REP son prototipado y verificación, desarrollo de algoritmos, robótica educativa, control de hardware o presentación de productos entre otros [9]. En la figura 5 se muestra la interfaz de V-REP y una escena de unos robots de brazo articulado para manejar cajas.

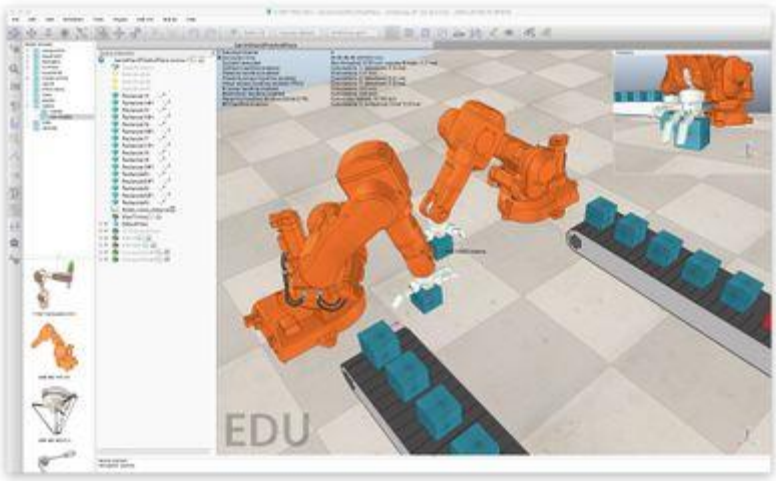

Figura 5: Interfaz de V-REP y modelos de robots de brazo articulado.

4 OBJETIVOS

4.1 Objetivo principal

El objetivo principal de este Trabajo de Fin de Máster es el desarrollo de un modelo de simulación 3D en VREP de la base del robot educativo Robobo, haciendo que su comportamiento sea lo más parecido posible al Robobo real, y permitiendo que se pueda programar con la librería Robobo.py.

4.2 Sub-objetivos

Para alcanzar el objetivo principal de este trabajo se deberán cumplir una serie de subobjetivos correspondientes al diseño, caracterización, integración y validación de los diferentes modelos implementados, y que se muestran a continuación:

- Diseñar los motores del Robobo en el modelo en simulación.
- Caracterizar el comportamiento de los motores del Robobo real e implementar este comportamiento en el modelo en simulación.
- Diseñar los sensores infrarrojos del Robobo en el modelo real.
- Caracterizar el comportamiento de los sensores infrarrojos del Robobo real e implementarlo en el modelo en simulación.
- Implementación de los encoders de los motores del modelo.
- Implementación de la cámara del smartphone en el modelo.
- Integración de los elementos comentados en un único modelo.
- Validación del modelo en simulación, comprobando el funcionamiento de todos los elementos del modelo, y comparación con el robot real.
- Estudio de discrepancias entre el modelo en simulación y el robot real.
- Integración de la librería de Python de Robobo con la librería de Python de V-REP, para que el robot real y el simulado se puedan controlar indistintamente.

5 ANTECEDENTES

En este apartado se describirán otros robots empleados tanto en educación como en investigación, a partir de los cuales también se han desarrollado modelos en simulación con el simulador V-REP, así como de los modelos de Robobo realizados con anterioridad a este trabajo.

5.1 LEGO Mindstorms EV3

LEGO (Lego System A/S) es una empresa danesa fundada en 1932 por Ole Kirk Christiansen. Aunque actualmente la línea principal de negocio de Lego es la producción y venta de juguetes, tiene una línea de robótica llamada Lego Mindstorms surgida de la colaboración de la empresa con el MIT (Massachusetts Institute of Technology) en 1985. Según este acuerdo, Lego financiaría investigaciones del grupo de epistemología y aprendizaje del MIT sobre cómo aprenden los niños y, a cambio, obtendría nuevas ideas para sus productos, que podría lanzar al mercado sin tener que pagar regalías al MIT [10].

LEGO Mindstorms es un set de construcción que permite construir, programar y controlar robots LEGO. El LEGO Mindstorms está basado en un bloque programable que cuenta con un microcontrolador con entradas y salidas a las que se le pueden conectar sensores, motores, etc. La última generación de este controlador es LEGO Mindstorms EV3, y se comercializó por primera vez en 2013 y contiene:

- Cuatro puertos de entrada rj12 modificada (para conectar los sensores al bloque EV3)
- Cuatro puertos de salida rj12 modificada (para conectar los motores al bloque EV3)
- Un puerto mini USB para PC (para conectar el bloque EV3 a un ordenador)
- Un puerto de host USB (para agregar un conector Wi-Fi y establecer conexiones en cadena)
- Un puerto para tarjetas Micro SD (para ampliar la memoria disponible en el bloque EV3)
- Un altavoz integrado
- Receptor de señales infrarrojas para recibir comandos
- Receptor Bluetooth y Wi-Fi.

A parte del controlador EV3 el set LEGO Mindstorms EV3 incluye cables conectores, elementos LEGO Technic (594 piezas), dos servomotores interactivos grandes, un servomotor interactivo mediano, un sensor táctil, un sensor de color, un sensor de infrarrojos y un transmisor de infrarrojos [11]. El controlador EV3 se muestra en la figura 6.

Figura 6: Controlador EV3 de LEGO Mindstorms.

La versión de LEGO EV3 permite construir una gran variedad de modelos. Los más básicos se muestran en la figura 7.

Figura 7: Modelos básicos de LEGO Mindstorms EV3.

Partiendo de estos modelos, en la Universidad de Málaga se ha desarrollado un modelo en simulación en V-REP de uno de los modelos de robot Lego Mindstorms EV3 y una toolbox de Matlab basada en las funciones del lenguaje NXC que permite controlarlo [12]. En la figura 8 se puede ver el modelo seleccionado para la simulación (izquierda), y su versión digital modelada en V-REP (derecha).

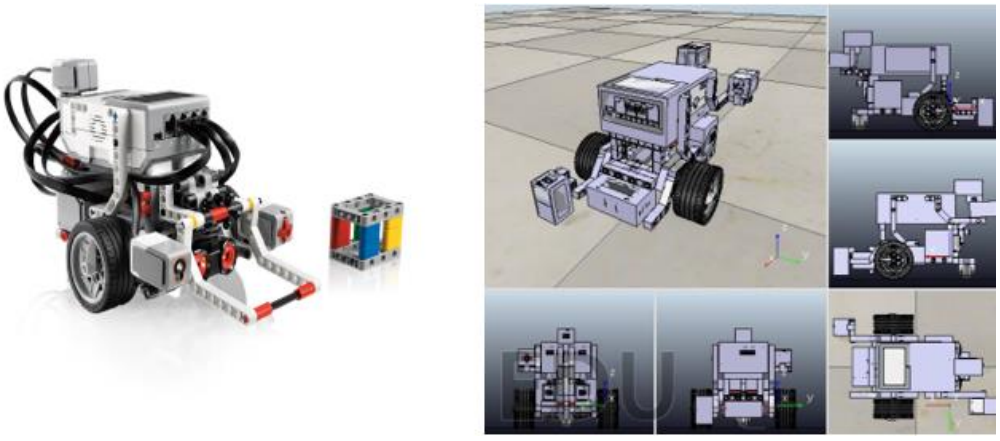

Figura 8: Modelo real LEGO Mindstorms EV3 y modelo simulado en V-REP.

5.2 TurtleBot

TurtleBot es un kit de robot personal de bajo coste con software de código abierto. Hasta ahora se han desarrollado tres generaciones de TurtleBot. La primera fue creada en Willow Garage por Melonee Wise (Fetch Robotics) y Tully Foote (Open Robotics) en noviembre de 2010 [13]. TurtleBot2 fue desarrollado por la empresa Yujin Robot basándose en el robot

iClebo Kobuki. Actualmente, el proyecto TurtleBot3, la última generación del TurtleBot, nace de una colaboración entre Open Robotics y ROBOTIS, además de muchos otros colaboradores.

TurtleBot 2 se compone de una base Yujin Kobuki, con una batería de 2200 mAh, un sensor Kinect, un Asus 1215N con procesador de dos núcleos, y un kit de montaje de hardware, que incluye los sensores y las piezas que compondrán el robot. Se estrenó en octubre de 2012 [14].

La última generación, TurtleBot3, consiste en una base con dos motores Dynamixel, una batería de 1800 mAh, un LIDAR (Light Detection and Ranging) de 360 grados, una cámara, una SBC (single board computer) Raspberry o Intel® Joule y un kit de montaje de hardware. Esta última generación salió a la luz en mayo de 2017. En la figura 9 se muestran las distintas versiones de TurtleBot.

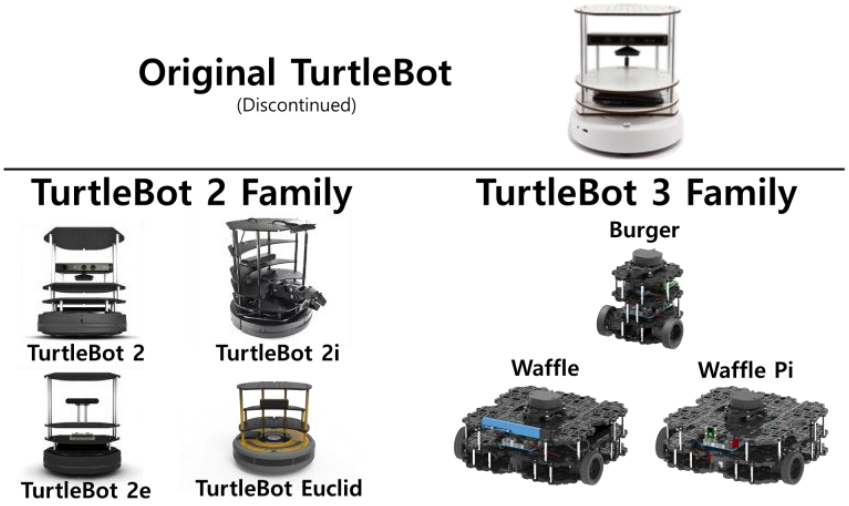

Figura 9: Versiones de TurtleBot (1,2 y 3).

TurtleBot2 en concreto, es un robot con un gran número de aplicaciones, desde investigaciones sobre robótica hasta la enseñanza en universidades e institutos o incluso en aplicaciones industriales.

Aunque en el ámbito de la simulación, los propios creadores de TurtleBot han realizado simulaciones de TurtleBot3 usando Fake Node y Gazebo, también hay proyectos de simulación de TurtleBot usando V-REP. Concretamente, en la Universidad de Málaga se ha desarrollado un modelo del TurtleBot 2 con un brazo manipulador WindowX usando este simulador [15]. Por otro lado, un equipo del Laboratorio de Computación Científica y Análisis Numérico de la Universidad Federal de Alagoas (Brasil) ha desarrollado un robot basándose en la base del TurtleBot1, desarrollándola con V-REP [16]. En la figura 10 se muestra el modelo de simulación de TurtleBot 1 en V-REP.

Figura 10: Modelo de TurtleBot 1 en V-REP.

5.3 E-puck

El proyecto de E-puck nació en la Escuela Politécnica Federal de Lausana (EPFL), en Suíza, como colaboración de varios grupos dentro de la universidad, concretamente, el Laboratorio de Sistemas Autónomos (Autonomous System Lab), el Grupo de Sistemas de Inteligencia Colectiva (Swarm-Intelligent Systems group) y el Laboratorio de Sistemas Inteligentes (Laboratory of Intelligent System) [17]. El objetivo del proyecto E-puck es desarrollar un robot móvil con fines didácticos, orientado al ámbito universitario y especialmente a la rama de ingeniería [18]. La figura 11 muestra una imagen del E-puck real.

Figura 11: Robot E-puck.

En la página web de e-puck, se muestra la última versión del robot, el e-puck2. Fue desarrollado por GCtronic y la EPFL incluye un microcontrolador STM32F4, un sensor WIFI, un USB de conexión y carga y una gran variedad de sensores (infrarrojo de proximidad, de sonido, 9 IMU (unidad de medida inercial), sensores de distancia ToF, cámara, etc .

Actualmente existen modelos desarrollados por la EPFL del E-puck compatibles con los simuladores como ENKI, WEBOTS, ARGoS y V-REP. El aspecto del modelo en V-REP se muestra en la figura 12.

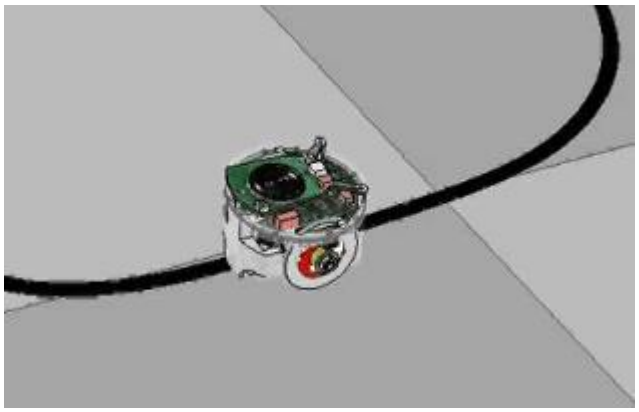

Figura 12: Simulación en V-REP de E-puck.

Partiendo de los modelos de e-puck se han realizado proyectos como el de Andrei George Florea que integró el modelo en V-REP del e-puck en ROS para ofrecerle a los estudiantes una plataforma de experimentación gratuita [19]. También se han usado los modelos de e-puck para realizar experimentos robótica colectiva y de lógica difusa donde los robots debían esquivar obstáculos [20].

5.4 Khepera (IV)

El robot Khepera es un robot móvil de tamaño reducido desarrollado en la Escuela Politécnica Federal de Lausana, en concreto, en el LAMI (Microprocessor Systems Lab) en 1991. De este proyecto surgió la empresa K-Team (1995), que se dedica a desarrollar y comercializar robots para educación e investigación. Actualmente, la empresa K-Team ha participado y dado soluciones a unas 600 universidades y centros industriales de investigación en todo el mundo [21].

Khepera IV, mostrado en la figura 13, es la última versión del robot Khepera y cuenta con un sistema operativo basado en Linux, WiFi, Bluetooth, acelerómetro, giroscopio, cámara a color y batería de mayor duración que las versiones anteriores. Incluye también 8 sensores infrarrojos y 5 sensores ultrasónicos y motores de corriente continua de alta calidad.

Figura 13: Modelo real del Khpera IV.

Partiendo del modelo de Khepera IV, se han realizado librerías [22] y modelos de este robot en V-REP para realizar experimentos con uno o varios robots y estudiar los posibles resultados antes de probarlos con el modelo real [23].

5.5 NAO

NAO, mostrado en la figura 14, es un robot humanoide, interactivo y totalmente programable desarrollado por la empresa Softbank Robotics en el año 2008. Actualmente, este robot ya ha alcanzado su quinta versión.

El NAO cuenta con una gran variedad de sensores: dos cámaras, cuatro micrófonos, nueve sensores táctiles, dos sensores de ultrasonidos, ocho sensores de presión, un acelerómetro y un giróscopo. Con el objetivo de ser más interactivo incluye elementos de expresión como LEDs RGB (53), un sintetizador de voz y dos altavoces.

Figura 14: Robot NAO.

Es programable en en C++, Python, JAVA, .NET y Matlab, aunque también tiene un software gráfico de programación para usuarios con menor conocimiento de programación, llamado Choreographe [24].

Actualmente NAO tiene una gran cantidad de aplicaciones especialmente en los ámbitos de educación, desde 5 años hasta la universidad y tanto para profesores como alumnos; en investigación, para realización de modelos conceptuales y teóricos y experimentos, trabajando con más de 400 universidades; en el hogar, como una de entretenimiento y para acercar la ciencia a los más jóvenes; y, por último, la industria, donde es muy útil en marketing y comunicación.

Hasta el día de hoy se han desarrollado un gran número de modelos y escenarios en V-REP simulado el robot NAO, con el objetivo de estudiar tareas de movimiento, aunque también de toma de decisiones o coordinación. Algunas de estas simulaciones se han usado, por ejemplo, para imitar distintas tareas humanas [25], para realizar experimentos de movimiento y evasión de obstáculos [26], mostrado en la figura 15, o incluso para simular humanoides jugando al fútbol [27].

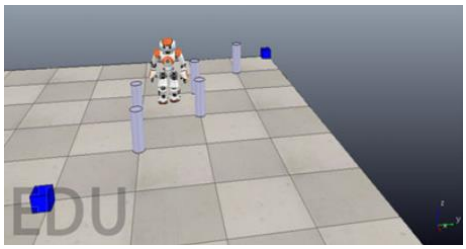

Figura 15: Simulación en V-REP de evasión de obstáculos con NAO.

5.6 Robobo

En relación con el Robobo, en los últimos años, alumnos de la Escuela Politécnica Superior han desarrollado modelos en V-REP en los que se mejoraban progresivamente distintos aspectos del modelo, especialmente los relacionados con el modelado de los motores y los sensores infrarrojos [28]. Por otro lado, el control de estos modelos se realizaba de forma externa con las APIs disponibles en V-REP, cosa que también se hará en este trabajo, aunque se incluirán cambios significativos.

Con este Trabajo de Fin de Máster, se pretende mejorar los modelos previamente realizados de Robobo en varios aspectos. Por un lado, se realiza una calibración de los motores distinta a la realizada en su trabajo tratando de encontrar una ecuación que se aproxime más al comportamiento del motor real, tanto los de las ruedas como la unidad PAN-TILT. Por otro lado, se buscará una forma alternativa de modelar los infrarrojos para mejorarlos en cuanto a similitud con los reales. Los modelos de sensores infrarrojos implementados en los desarrollos anteriores consisten en sensores de proximidad que forman volúmenes de detección y que miden la distancia mínima a un objeto utilizándola para estimar la respuesta del sensor infrarrojo real. Esta simplificación supone una pérdida de información de la posición del objeto respecto al sensor y, por lo tanto, no es posible diferenciar el área detectada por el sensor de dos objetos que estén a la misma distancia mínima, pero en una posición diferente o de distinta geometría. En este proyecto se ha realizado una redefinición de estos sensores usando modelos basados en haces de rayos que permiten abarcar el mismo volumen de detección que el sensor infrarrojo real y que proporcionan un array de distancias correspondientes a las intersecciones de estos haces de rayos con el objeto que se está detectando.

Otro aspecto a mejorar es el del uso del API externa. En el desarrollo previo se empleaba la API externa de V-REP para Python para controlar toda la simulación de forma completamente externa. En este trabajo, el control de la simulación y del modelo se realizará en V-REP, y se empleará la API externa de Python para llamar a las distintas funciones que han sido desarrolladas dentro de simulador. Esto implica que la simulación funciona con el tiempo del simulador, y tiene la ventaja de que el tiempo de ejecución de las funciones en la simulación no dependa de la carga de trabajo o velocidad del procesador. Por último, se realizará una conexión entre la librería de Python de Robobo y la del V-REP para poder controlar el modelo simulado y el real con los mismos comandos, aumentando así sus parecidos.

6 FUNDAMENTOS TECNOLÓGICOS

En este apartado se explicarán los aspectos tecnológicos que han sido necesarios para llevar a cabo el trabajo. Entre ellos, se detallará la configuración del Robobo, y se describirán especialmente los motores de corriente continua y los sensores infrarrojos que serán las partes del robot más estudiadas y en las que se han centrado los esfuerzos de este trabajo. Por otro lado, también se describirá el funcionamiento y las características de V-REP y de los lenguajes de programación empleados en este trabajo.

6.1 Robobo

Dado que el objetivo de este proyecto es llevar a cabo la simulación en V-REP del Robobo, para llevar a cabo dicha simulación será necesario conocer perfectamente los componentes físicos del Robobo así como su distribución. Debido a que en este proyecto se parte de un modelo ya existente, se prestará mayor atención a las partes del modelo que se volverán a calibrar y modelar para mejorar su comportamiento, que serán los motores de corriente continua, tanto de las ruedas como del conjunto Pan-Tilt, y los sensores infrarrojos.

Como ya se ha comentado en la introducción, Robobo cuenta con una base móvil a la que se le acopla un smartphone, conectados de forma inalámbrica mediante Bluetooth. La base cuenta con dos ruedas, una unidad PAN-TILT (en la que se acopla el smartphone), cinco leds frontales y dos traseros, cinco sensores infrarrojos y tres traseros, un conector micro USB (tipo B) y un botón de encendido. Los componentes de la base del Robobo aparecen indicados en la figura 16.

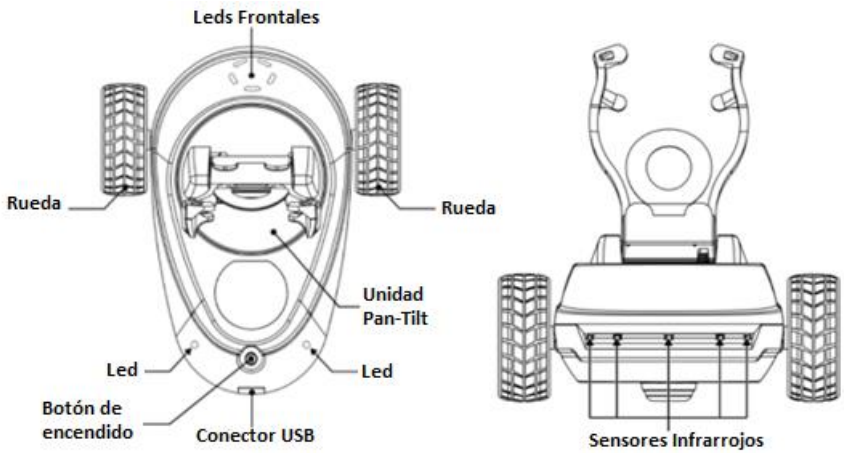

Figura 16: Componentes de la base del Robobo.

Una vez descrito el Robobo de forma general, se pasará a describir los motores y los sensores infrarrojos, ya que serán los elementos fundamentales que se incluirán en el modelo a desarrollar.

6.1.1 Sensores infrarrojos

El funcionamiento del sensor infrarrojo se basa en su capacidad para medir la radiación electromagnética infrarroja de los objetos que aparecen en su campo de detección, convirtiéndola en una señal eléctrica. Normalmente están compuestos por dos elementos principales, un LED infrarrojo (IRLED) y un fototransistor.

El IRLED es el encargado de emitir radiación infrarroja. Por otro lado, el fototransistor es un fotodetector que actúa como un transistor clásico. Está compuesto por un cristal fotosensible que cuando recibe luz produce una corriente y desbloquea el transistor. La corriente generada aumentará en función de la luz recibida.

Los sensores infrarrojos integrados en el Robobo, mostrados en la figura 17, son del modelo VCNL4040, de la marca VISHAY. Estos sensores integran un sensor de proximidad, un sensor de luz ambiente y un diodo LED infrarrojo (IRED) en un único elemento.

Figura 17: Sensores Infrarrojos modelo VCNL4040

La cantidad de luz que reciba el fototransistor y por lo tanto la medida final del mismo dependerá de varios factores, entre ellos la distancia a la que esté el objeto u objetos detectados, su color y material, el ángulo de incidencia de la radiación infrarroja en el objeto o el ángulo de apertura del sensor.

Concretamente, el modelo VCNL4040 tiene un alcance de 200 mm y su perfil de detección se muestra en la figura 18 [29]:

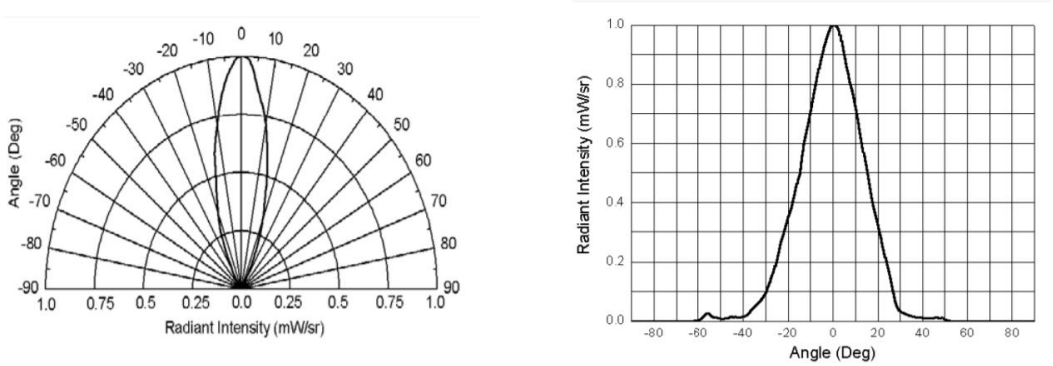

Figura 18: Perfil de detección teórico de los sensores infrarrojos.

6.1.2 Motores

Los motores integrados en el Robobo son motores de corriente continua de la marca TT MOTOR. El modelo en concreto es el TFF-N20VA-09220, con una tensión nominal de 5 V y una velocidad en vacío de 15000 revoluciones por minuto. Al motor se le acopla una reductora con un factor de reducción concreto, en función de las necesidades de par de cada actuador.

Para mover las ruedas y permitir el movimiento del Robobo, cada rueda tiene acoplado un motor con una relación de reducción de 150:1, lo que supone un par nominal de 0,22 kg.cm.

En el caso de la unidad Pan, que es la encargada de cambiar la orientación del smartphone mediante un movimiento rotativo de la base respecto al eje vertical, el factor de reducción es igual que el de las ruedas, 150:1.

Por último, la unidad Tilt se encarga de variar la inclinación del smartphone en relación a su eje, paralelo al eje de las ruedas. Dado a las necesidades de par de este motor, la relación de reducción de la reductora acoplada es de 1000:1. Los ejes del pan y del tilt están representados en la figura 19.

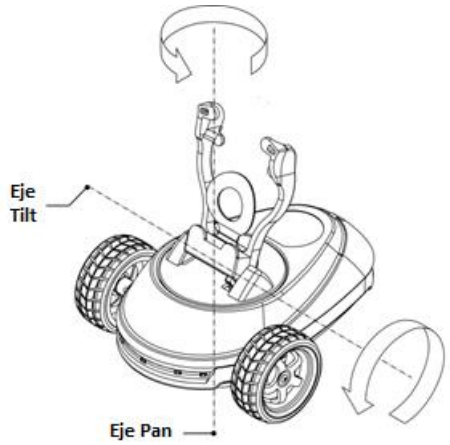

Figura 19: Ejes del pan y del tilt.

6.2 Simulador: Virtual Robot Experimentation Platform (V-REP)

V-REP es un simulador 3D desarrollado por la empresa Coppelia Robotics compatible con Windows, Linux y Mac. Actualmente existen tres versiones de V-REP: una versión educativa, V-REP PRO EDU, gratuita para fines educativos, una versión comercial, V-REP PRO, y una versión gratuita para ejecutar simulaciones, V-REP PLAYER.

Coppelia Robotics ofrece en su página web una gran cantidad de información acerca del programa para ayudar a los usuarios a iniciarse en el mismo, como tutoriales, manuales, vídeos de simulaciones y modelos realizados por otros usuarios y que son accesibles a todo el mundo. A mayores, existe un foro donde las dudas de los usuarios son contestadas por los técnicos de la empresa.

En los siguientes subapartados se describirán los aspectos más importantes del programa para dar una idea general de su funcionamiento y su interfaz, así como de los elementos que pueden componer un modelo o escena, sus módulos de cálculo dinámico o su programación. Estos apartados servirán de introducción para una mejor comprensión del desarrollo del modelo 3D así como la programación del modelo.

6.2.1 Interfaz

Los elementos principales de la interfaz que aparecen al abrir el programa son:

- Ventana de consola: no es interactivas y su única función es mostrar información, programándolo en plugins o scripts.
- La ventana de aplicación: es la ventana principal y en ella se muestra la escena a simular, permitiendo editarla e interactuar con ella con los botones o la rueda del ratón.
- Diálogos: hay múltiples diálogos para editar elementos de la escena y variar parámetros.
- Barra de aplicaciones: da información del tipo de licencia, el nombre del archivo o el estado de simulador, entre otras cosas.
- Barra menú: permite utilizar casi todas las funciones del simulados como guardar y abrir archivos, editar la escena o parámetros de simulación, cargar plugins o mallas, etc.
- Barra de herramientas: contiene las funciones para editar la escena. Permite desplazar objetos en la escena, variar su orientación, cambiar la cámara de visión, iniciar o pausar la simulación o editar las capas de la misma, entre otras cosas.

- Navegador del modelo: muestra la estructura de carpetas del modelo y lo que contiene cada una.
- Jerarquía de escena: contiene todo el contenido de la escena como los modelos, sus componentes y la relación entre ellos, o elementos independientes como el suelo. Desde ella se pueden editar los distintos elementos la escena.

Los distintos elementos de la interfaz mencionados en este punto aparecen indicados en la figura 20.

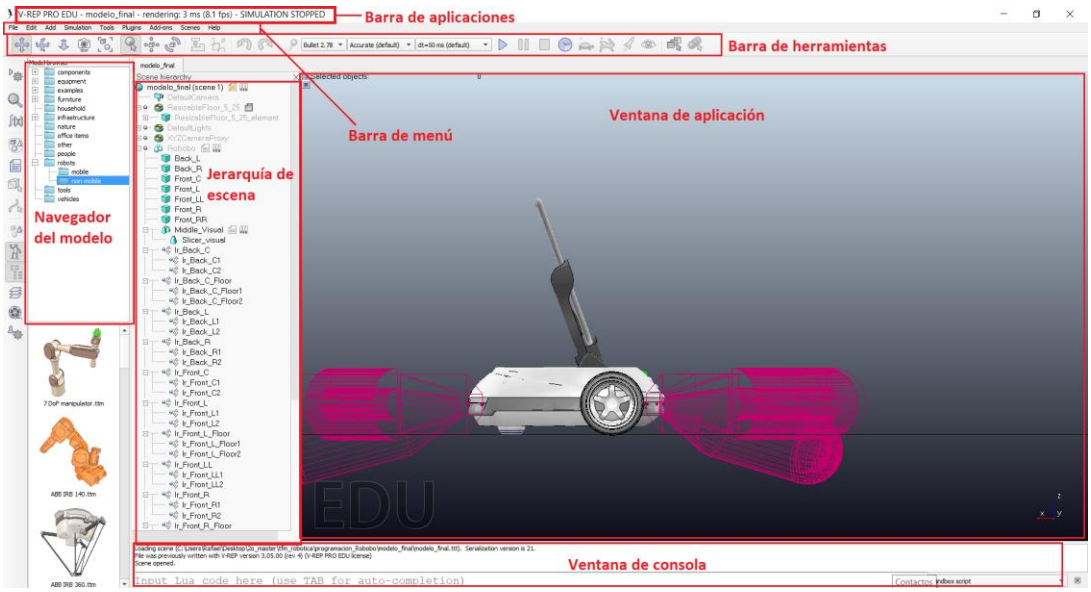

Figura 20: Elementos de la interfaz de V-REP.

6.2.2 Objetos de escena, modelos y escenas.

Los objetos de escena se emplean para crear la simulación de una escena. Los objetos aparecen disponibles en V-REP para generar una simulación se encuentran en la figura 21.

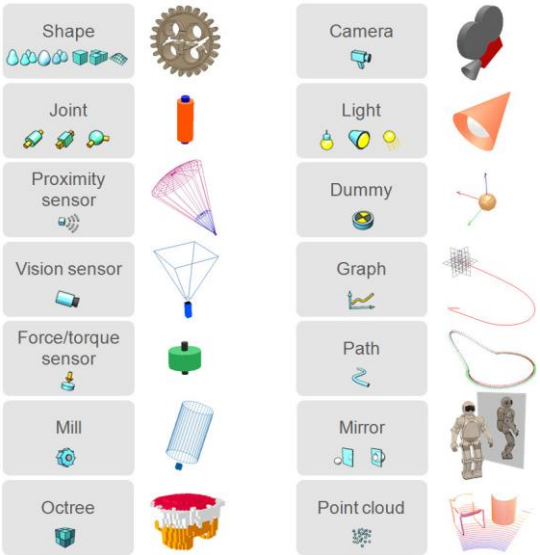

Figura 21: Objetos de escena.

En la simulación los objetos se pueden editar para que puedan colisionar con otros objetos, se pueda medir la distancia entre ellos, sean detectables por sensores de proximidad y de visión y sean visibles por las cámaras de simulación.

Los objetos de escena más importantes para realizar el modelo de simulación de este trabajo serán las *shapes*, los *joints*, *proximity sensor* y el *vision sensor*, por eso se explicarán con más detalle.

Por un lado, las *shapes* son mallas rígidas formadas por caras triangulares y conforman la estructura física de cualquier modelo, es decir, su forma. Para trabajar con este tipo de objetos se pueden editar directamente en V-REP o importar en un formato OBJ, DXF, STL, COLLADA o URDF. Hay cuatro tipos principales de *shapes*: las puras (pure shapes), las aleatorias (random shapes), las convexas (convex shapes) y las "heightfield shapes".

Las formas puras son formas sencillas como un cubo, un cilindro o una esfera, y dan mejores resultados en los cálculos de colisión que el resto de formas. Pueden ser simples o compuestas en función de si el objeto está formado por una malla o varias mallas agrupadas. El icono de las formas puras se muestra en la figura 22.

Figura 22: Icono de las formas puras (pure shapes).

Las formas aleatorias pueden ser cualquier malla que sea importada a V-REP desde un software de diseño 3D. En función de si tienen uno o varios atributos visuales pueden ser simples o compuestas. Este tipo de formas están desaconsejadas para hacer un cálculo de colisiones ya que relentizan mucho la simulación y son inestables. El icono de las formas aleatorias se muestra en la figura 23.

Figura 23: Icono de las formas aleatorias (random shapes).

El tercer tipo de formas son las formas convexas y son mallas importadas, como las *random shapes*, optimizadas para el cálculo de colisión. Al igual que las formas aleatorias, en función de los atributos visuales pueden ser simples o compuestas. El icono de las formas convexas se muestra en la figura 24.

Figura 24: Icono de las formas convexas (convex shapes).

El último tipo son las "heightfield shapes" y pueden equipararse a formas puras simples optimizadas para realizar cálculos de colisión. Su icono correspondiente se muestra en la figura 25.

Figura 25: Icono de las formas "heightfield".

Generalmente, las formas puras y las formas convexas se emplean para realizar el cálculo dinámico de colisión, y están situadas en una capa oculta, mientras que en la capa visible están las formas aleatorias asociadas a las formas empleadas para el cálculo, y su única función es visual, dándole más realismo a la simulación. En la figura 26 se muestra la diferencia entre las formas puras y convexas, y las aleatorias, para el modelo del Robobo.

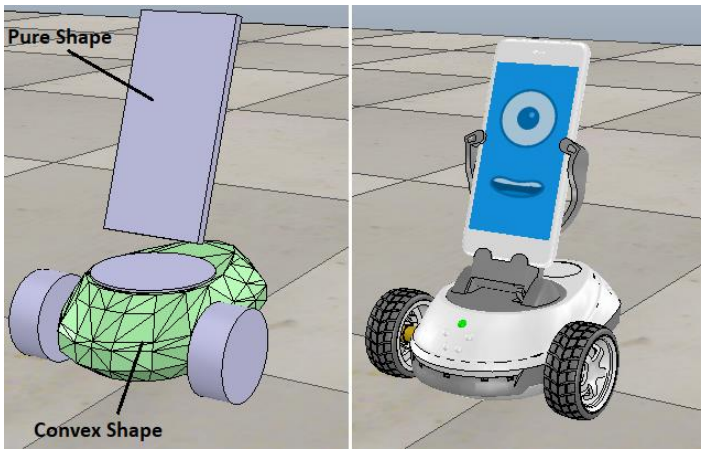

Figura 26: Diferencia entre formas puras, convexas y aleatorias.

Por otro lado, están los *joints* que son objetos que tienen como mínimo un grado de libertad y conectan dos objetos entre sí, permitiendo un movimiento relativo de uno de ellos respecto al otro.

Hay cuatro tipos de *joints* diferentes: de revolución, que tienen un grado de libertad y permiten girar alrededor de un eje; prismáticos, que tienen un grado de libertad y realizan un movimiento de traslación en el eje que se le asigne; esféricos, que tienen tres grados de libertad y permite un movimiento rotacional entre objetos; y de tornillo, que es una combinación de los joints de revolución y prismático. Cada uno de los tipos aparecen representados en la figura 27.

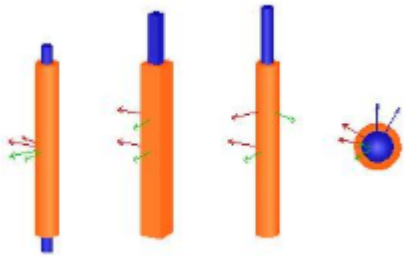

Figura 27: Tipos de *joints*.

Los *joints* tienen distintos modos de control, entre los que destacan el modo pasivo donde el *joint* actúa como una unión fija y el modo par/fuerza, que permite controlar el joint mediante la aplicación de una fuerza o par. Cuando el modo par/fuerza no está activado, el *joint* se puede controlar por posición mediante la configuración de un PID, mientras que si está activado se controlará aplicándole al *joint* una velocidad objetivo.

Otro de los objetos comentados al principio de este apartado son los sensores de proximidad. Estos objetos permiten detectar un objeto y devuelven la distancia mínima entre objeto que corta al volumen de detección y el sensor. El volumen de detección puede editarse, pero básicamente tiene las 5 formas que aparecen en la figura 28: rayo, pirámide, cilindro, disco o cono. En este trabajo se emplearán los sensores en forma de rayo.

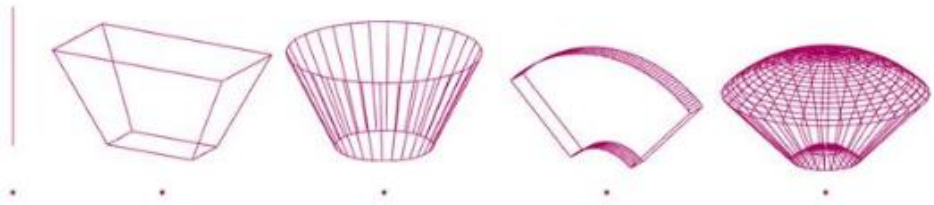

Figura 28: Formas del volumen de detección de los "proximity sensors".

Partiendo de estos objetos y sus propiedades, se modelizarán los sensores infrarrojos del Robobo.

El último elemento del que se hablará será el *vision sensor*. Este objeto será el encargado de modelar la cámara del smartphone, renderizando los objetos están en su campo de visión (siempre que estos objetos sean renderizables). En la figura 29 se muestra el icono del sensor.

Figura 29: Icono del sensor de visión.

Una vez introducidos los objetos de escena, el modelo simplemente es la creación de una versión digital del Robobo empleando *shapes* para darle la forma del modelo real al modelo de simulación, y a mayores añadiendo *joints* que permitan el movimiento del mismo, sensores, cámaras, etc. Para que el modelo se comporte como un único objeto, todos los elementos que lo conforman deben estar conectados entre sí de forma lógica mediante relaciones de jerarquía, y debe haber un elemento que sea la base del modelo y del que dependan el resto de los elementos.

Una vez obtenido el modelo, una escena consiste en un escenario, formado por los mismos elementos descritos anteriormente, en el que se desarrollará la simulación. Incluirá un suelo que soporte a los distintos elementos y puede incluir objetos como formas puras o mallas importadas, luces, cámaras, etc. como el mostrado en la figura 30.

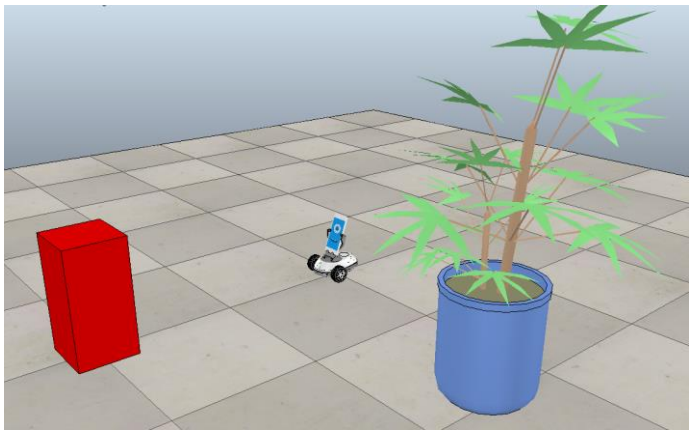

Figura 30: Escena con el modelo de Robobo en V-REP.

6.2.3 Módulos de cálculo

V-REP incluye cuatro módulos de cálculo: el de detección de colisión, cálculo de distancia mínima, cinemática inversa y dinámica.

El módulo de detección de colisión calcula las colisiones que se pueden producir entre dos objetos, aunque no calcula la respuesta de los mismo ante la colisión. Para que los objetos puedan colisionar deben estar definidos como *collidable*. La información obtenida de este módulo se puede extraer y representar en gráficos para su mejor visualización. La figura 31 muestra cómo el módulo de detección de colisiones no calcula las respuestas ante las mismas.

Figura 31: Ejemplo de funcionamiento del módulo de detección de colisiones.

El módulo de cálculo de distancia mínima calcula distancias mínimas entre dos objetos, definidos como medibles. Sin embargo, este módulo solo puede medir distancias, y no actuar en función de las medidas. En este caso, la información que aporta este módulo se puede guardar en gráficos. En la figura 32 se observa el funcionamiento del módulo de cálculo de mínima distancia.

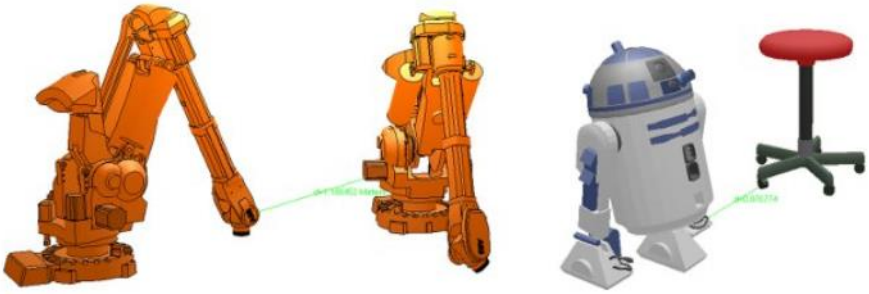

Figura 32: Módulo de cálculo de mínima distancia.

El módulo de cinemática inversa permite resolver problemas cinemáticos de cualquier mecanismo, contando para esto con dos modos: el modo de cinemática inversa (IK mode) y el modo de cinemática directa (FK mode). El primero permite calcular los valores de posición y orientación de un joint partiendo del movimiento final. Por otro lado, el modo de cinemática directa permitiría calcular el efecto final de un mecanismo partiendo de los valores de sus joints. La figura 33 muestra un esquema del modo de cinemática directa, con el que se puede saber el movimiento final de un objeto a partirdel valor de los *joints*.

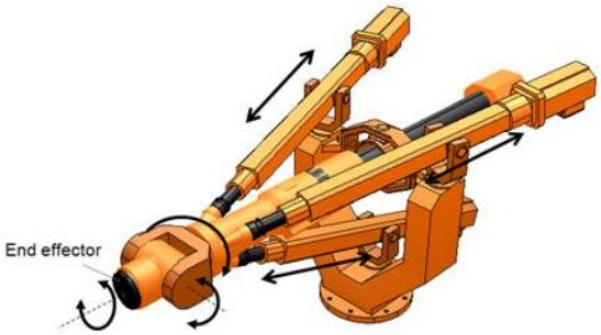

Figura 33: Módulo de cinemática inversa (modo FK).

Por último, el módulo de dinámica simula objetos dinámicos, y calcula las interacciones entre ellos como, por ejemplo, respuestas de colisión. Actualmente, V-REP cuenta con cuatro motores distintos para desarrollar la simulación: el Bullet, cuyo logo se muestra en la figura 34; el ODE (Open Dynamics Engine), el Vortex y el Newton. En este trabajo en concreto todas las simulaciones se harán con el motor Bullet, un motor de física 3D, de software libre para el cálculo de colisiones, dinámicas de cuerpo rígido y blando, usado sobre todo en videojuegos.

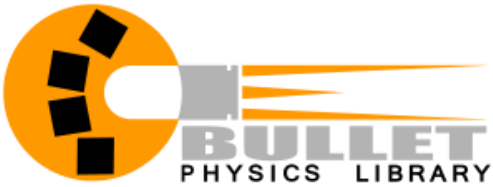

Figura 34: Logo del módulo de cálculo dinámico Bullet.

6.2.4 Programación en V-REP

Para programar el comportamiento de un modelo o una escena, V-REP ofrece varias alternativas: scripts, add-on, plugins, API (Application Programming Interface) externa, nodos de ROS y nodos BlueZero.

Dado que en este trabajo se han empleado los scripts y la API externa para programar el modelo, sólo se comentarán estos dos elementos.

Antes de hablar de los scripts, se comentará la importancia de Lua en V-REP. Lua es un lenguaje de programación gratuito y de software libre, empleado en programación de procedimientos, programación orientada a objetos, programación funcional, programación basada en datos, etc. Su sintaxis y su forma de escribirse y ejecutarse lo hace ideal para creación de scripts [30].

Los scripts embebidos (embedded scripts) son scripts programados en Lua, que permiten personalizar el comportamiento de un modelo y escena, y es el método más sencillo de personalizar la simulación en V-REP, en comparación con los plugin o los add-on. A parte de Lua, desde los *scripts* también se pueden llamar a las funciones de la API interna de V-REP, que permiten obtener datos de la simulación, personalizarla y actuar sobre ella, controlándola. Estos comandos comienzan por el prefijo “sim.”.

Los scripts embebidos se dividen en scripts de simulación, que son los ejecutados durante la simulación personalizando la misma o el modelo; y los scripts de personalización, que editan la escena o el simulador mientras la simulación está parada.

Centrándose en los scripts de simulación, es importante diferenciar el script principal, donde se edita el código necesario para que la simulación se pueda ejecutar, y los child scripts que son los asociados al modelo, o a componentes del mismo, y que controlan realmente el comportamiento del modelo en la simulación. Concretamente, los child scripts empleados en este trabajo son los *non-threaded scripts*, que pueden contener una colección de funciones bloqueantes, lo que significa que cuando son llamados deben realizar una tarea y devolver el control para que la simulación continue.

Por otro lado, el otro elemento de programación empleado en este trabajo fue la API externa. Una API (Application Programming Interface) es una interfaz de programación de aplicaciones, es decir, un conjunto de rutinas que permite acceder a las funciones de un determinado software desde una aplicación distinta al mismo. Esto significa que se puede controlar la simulación desde un software externo a V-REP.

El API externa de V-REP está disponible para para varios lenguajes de programación como Python, Matlab, Java, C++ o Lua.

En este proyecto, se empleará la API externa de Python, que incluye comandos que permiten obtener información de la simulación o personalizarla. Además, también permite ejecutar funciones definidas en los scripts de simulación que están programados dentro del modelo, lo que será muy útil para programar el comportamiento del robot en V-REP y controlarlo desde Python.

En este proyecto, se controlará el modelo en V-REP con los comandos de la librería Robobo.py, como ocurre con el robot real. Para esto se necesitará implementar un servidor Websocket, que se explicará en el apartado 8 de este trabajo. El hecho de que el modelo de

simulación se pueda controlar con los mismos comandos hace que la similitud entre el robot simulado y el real sea muy elevada.

La figura 35 muestra la estructura que siguen los elementos encargados de controlar la simulación en V-REP.

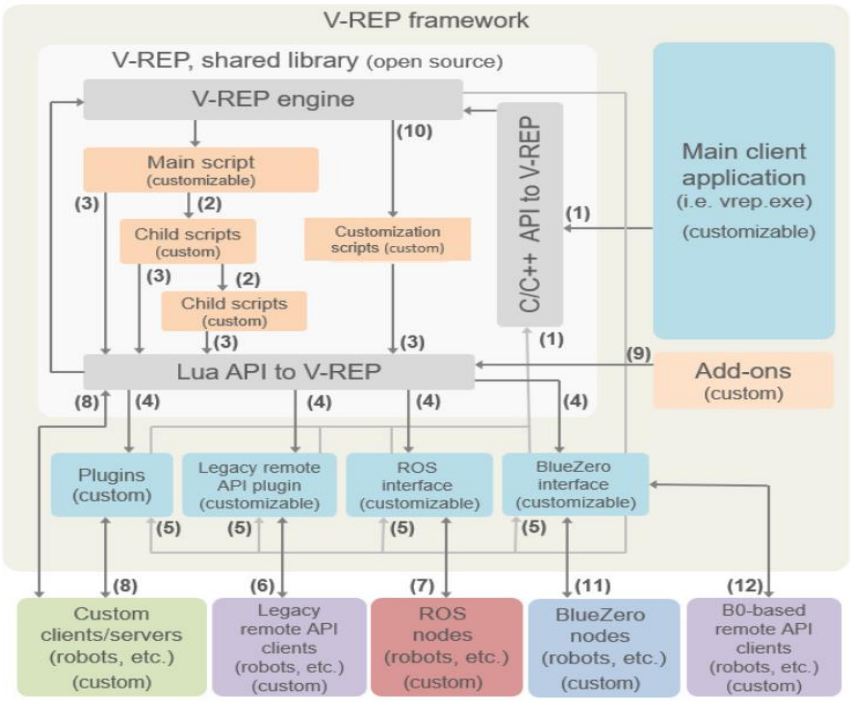

Figura 35: Estructura de la programación de V-REP.

7 DESARROLLO DEL MODELO DE ROBOBO

7.1 Modelo 3D

El modelo del Robobo en V-REP está formado por 3 componentes básicos. Por un lado, las *shapes*, de las que se hablaron en el apartado 6.2.2 de este trabajo, y que se emplearán para representar las ruedas, la base, la unidad pan-tilt, y el smartphone, con su soporte, del Robobo. Por otro lado, los *joints* conectarán los distintos componentes del modelo y se emplearán para modelar los motores del Robobo. Por último, los sensores de proximidad simularán los sensores infrarrojos.

7.1.1 Componentes del modelo 3D: Shapes.

En este apartado se explicará con más detalle las formas que componen el modelo.

Los distintos elementos que componen el modelo de Robobo (ruedas, base, smartphone, etc.) constan a su vez de dos componentes, una componente dinámica, empleada para realizar los cálculos de la simulación, y otra visual, como ya se ha explicado en el apartado 6.2.2. La parte dinámica puede ser *pure* o *convex shape*, mientras que la parte visual siempre será una *random shape*. Tanto la parte dinámica como la parte visual están diseñadas a escala real, respetando las medidas de las piezas reales.

Empezando por las ruedas, su componente dinámico consiste en una *pure shape* cilíndrica modelada directamente en V-REP. Tanto la parte visual (izquierda) como dinámica (derecha) de las ruedas se muestran en la figura 36.

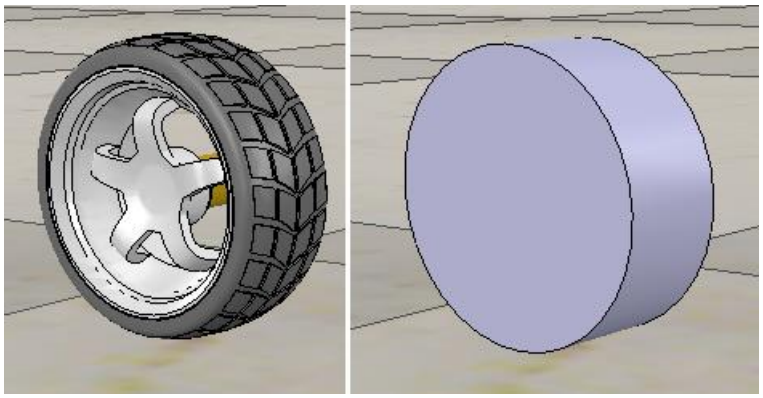

Figura 36: Componente visual y dinámica de las ruedas.

En el caso de la base, en el modelo de Robobo desarrollado previamente, el componente dinámico de la misma se dividía en dos elementos, la malla de la propia base (*convex shape*) y la deslizadera (*pure shape* cilíndrica), y estaban unidos mediante un *force sensor*, simulando una unión rígida. Sin embargo, esta configuración generaba errores de contacto en la unión entre los dos elementos dando como resultado movimientos no controlados del Robobo en la simulación.

Para solucionar este problema, la parte dinámica se modela con una *convex shape*, consistente en una malla creada e importada desde el software Blender y convertida en una malla de componentes triangulares más simple para reducir el coste del cálculo dinámico. Esta malla incluye tanto la base del Robobo como la deslizadera, mientras que la componente visual está formada por dos mallas, la de la deslizadera y la de la propia base.

En la figura 37 se muestra dicha malla, así como la componente visual.

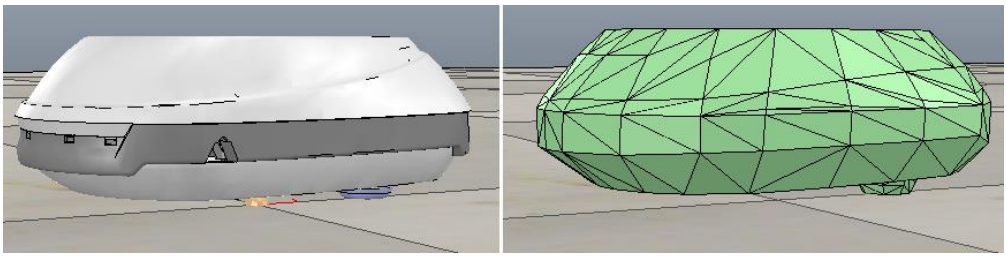

Figura 37: Componente visual y dinámica de la base.

Para modelar la unidad PAN se ha empleado una *pure shape* cilíndrica como la que aparece en la figura 38. En esa misma figura se compara con la componente visual del PAN, mucho más compleja.

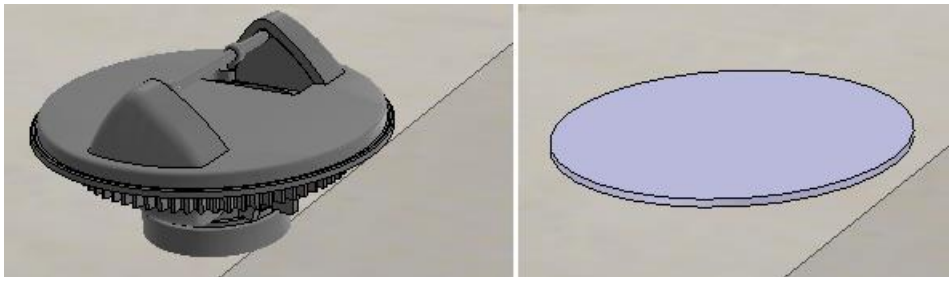

Figura 38: Componente visual y dinámica de la unidad PAN.

Por último, el smartphone, su soporte, y la unidad tilt están representados simplemente con un *pure shape* rectangular, modelando solo el smartphone, ya que a efectos de cálculo los otros elementos no son significativos y así se evita poner componentes dinámicos innecesarios. En la componente visual, sin embargo, se representan los 3 elementos. En la figura 39 se comparan las dos formas.

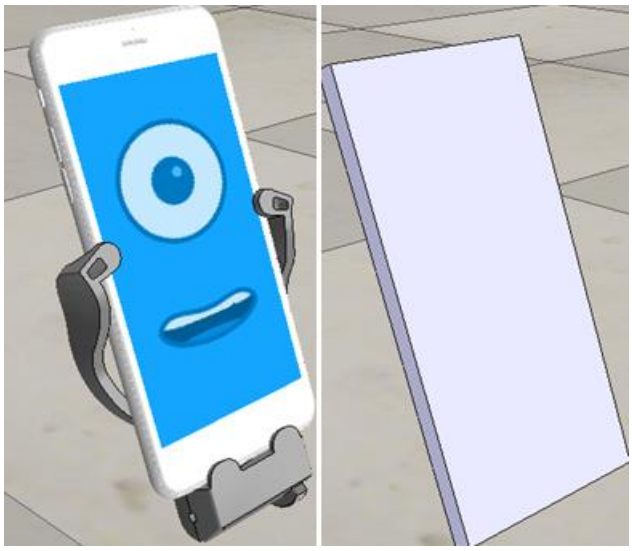

Figura 39: Componente visual y dinámica del smartphone y su soporte, y la unidad tilt.

Todos los componentes modelados para el cálculo dinámico ya sean *pure* o *convex shapes*, deben tener marcada las opciones *Body is respondable* y *Body is Dynamic*, en el diálogo de propiedades dinámicas. La primera implica que se producirán reacciones de colisión en caso de que dos objetos impacten. La segunda implica que los objetos variarán su posición y orientación durante la simulación, es decir, que serán dinámicos. Aunque no es parte del modelo, el suelo no debe cumplir la condición de dinámico, aunque si debe cumplir la primera condición. En ese mismo diálogo se personalizarán la masa y los momentos de

inercia de las distintas componentes, así como su material. Por defecto, todas las componentes del modelo tendrán el mismo material, excepto las ruedas y la deslizadera, a las que se les variará su valor de fricción para mejorar su comportamiento.

7.1.2 Componentes del modelo 3D: Joints.

Como ya se ha explicado en el apartado 6.2.2., hay cuatro tipos de joints. Sin embargo, en el modelo en cuestión sólo se empleará un tipo, el joint de revolución, ya que todos los motores generan un movimiento de giro alrededor de un eje.

En el modelo habrá 4 cuatro *joints* de revolución, dos para simular los motores de las ruedas, uno para el motor del pan y otro para el motor del tilt y su distribución se muestra en la figura 40, coincidiendo con el eje de cada elemento.

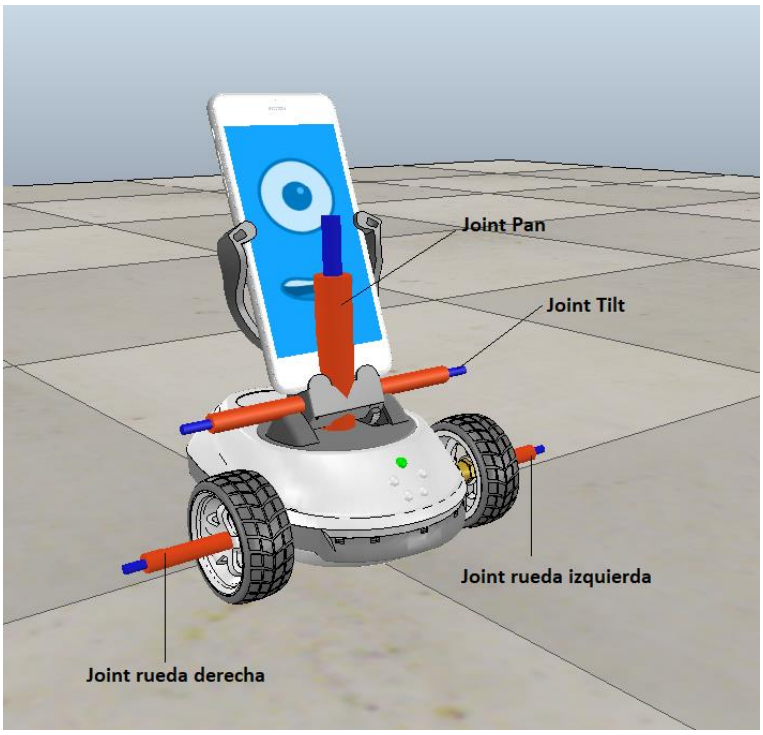

Figura 40: Distribución de los *joints* del modelo.

Todos los *joints* funcionarán en el modo *torque/force* y se controlarán por velocidad, asignándoles una velocidad objetivo (*target velocity)* en grados por segundo. Además, a cada *joint* se le asignará un par máximo que permita le permita alcanzar la velocidad objetivo de forma casi instantánea para no tener en cuenta la aceleración del motor y simplificar el cálculo (2.5 N.m). A mayores, los *joints* del pan y del tilt se bloquerán cuando su velocidad sea 0, para evitar movimientos no deseados debidos a la gravedad u otras causas. Estas características se pueden personalizar en el diálogo de propiedades dinámicas del *joint*.

7.1.3 Componentes del modelo 3D: sensores de proximidad.

Los sensores de proximidad, como ya se ha explicado en el punto 6.2.2, son sensores que detectan un objeto (siempre que esté configurado como detectable) y devuelven la distancia mínima al mismo. El modelo consta de 8 sensores de proximidad, cinco situados en la parte frontal de la base y otros tres en la parte trasera.

De los cinco sensores frontales, el que está situado en la posición central es paralelo al suelo y su dirección es perpendicular a la base del robot. Los dos sensores situados a los extremos están desviados 45 grados respecto al eje central en el plano paralelo al suelo y también son paralelos al suelo. Por otro lado, los dos sensores situados a ambos lados del

sensor central están desviados 18 grados respecto al sensor central en el plano paralelo al suelo, y están inclinados 10 grados hacia el suelo.

En el caso de los sensores traseros, el central está inclinado 7 grados hacia el suelo, mientras que los dos sensores que están a ambos lados del central son paralelos al suelo y se desvían 45 grados con respecto al sensor central. El objetivo de los sensores contenidos en el plano paralelo al suelo es detectar objetos que puedan estar en la trayectoria del Robobo para evitar colisiones, mientras que los que están inclinados hacia el suelo detectan la presencia o ausencia de suelo para evitar caídas del robot.

En la figura 41 se muestra la posición y orientación de los sensores respecto a la base del Robobo.

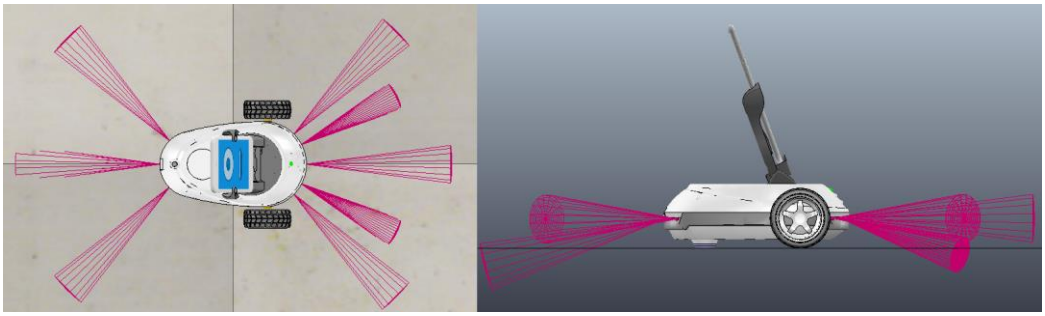

Figura 41: Distribución de los sensores de proximidad.

7.1.4 Componentes del modelo 3D: sensor de visión

Como ya se ha explicado anteriormente, el sensor de visión modelará la cámara del smartphone, de forma que en la simulación se podrá ver en una ventana secundaria lo que realmente está viendo la cámara. En la figura 42 se muestra el *vision sensor* implementado en el smartphone del modelo, así como los límites de su campo de detección.

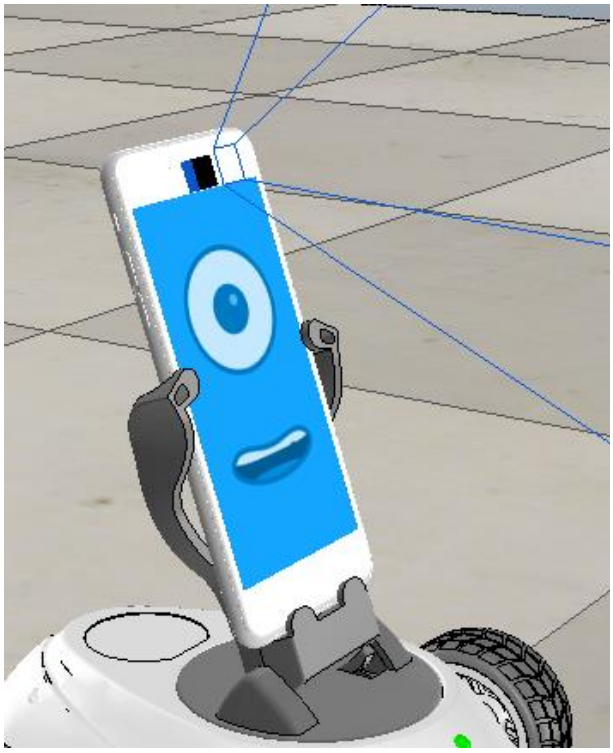

Figura 42: Integración del *vision sensor* en el modelo de V-REP.

Al implementar un *vision sensor* en el modelo se genera una ventana auxiliar en la que se muestran todos los elementos detectados por la cámara. En la figura 43 se muestra esta ventana, en la que se aparece la caja que el sensor detecta en el modelo.

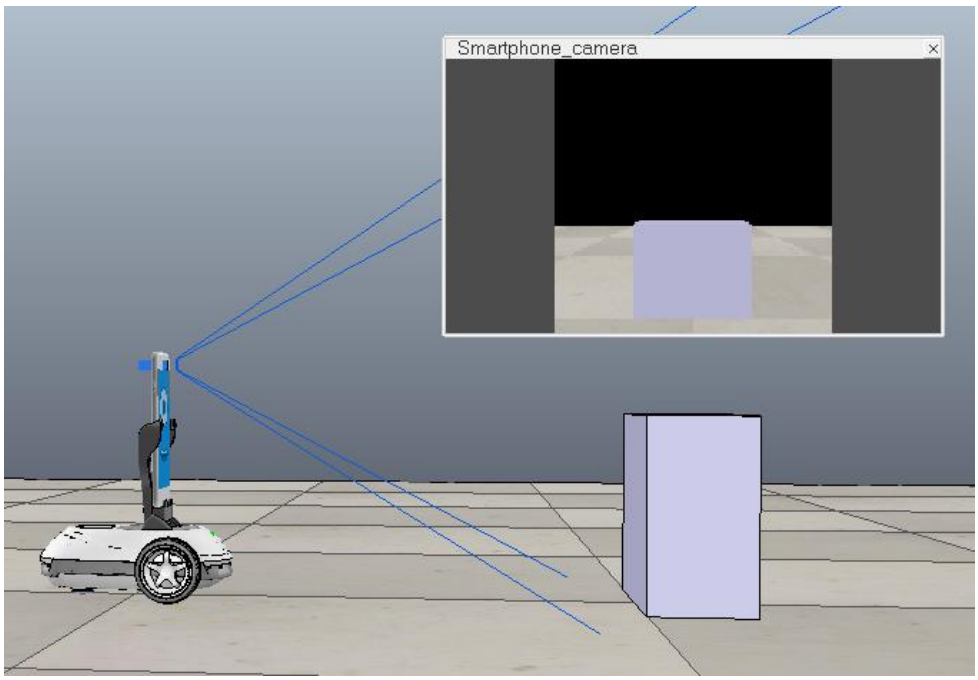

Figura 43: Modelo de Robobo con cámara integrada y ventana auxiliar de la misma.

7.1.5 Relación entre componentes

En este apartado se explicará cómo están conectados entre sí los distintos elementos que componen el modelo del Robobo para que se comporten de forma coordinada, cumpliendo las restricciones del modelo real.

Los objetos que forman un modelo conforman un árbol de jerarquía en donde todos los objetos dependen de un objeto principal que es considerado la base del modelo. En este modelo en concreto, la base del modelo será la malla de cálculo de la base del robot, llamada "Robobo". En la figura 44, se muestra una parte del árbol de jerarquía del modelo del Robobo.

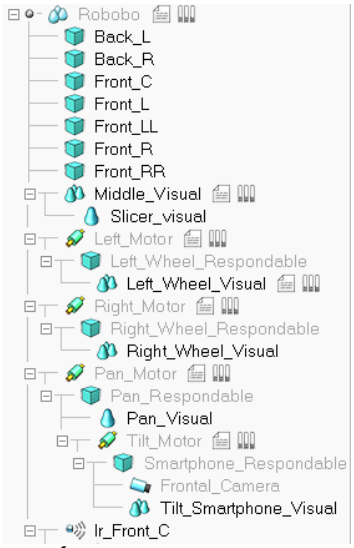

Figura 44: Árbol de jerarquía del modelo.

Hay dos configuraciones dentro del árbol de jerarquía que controlarán los movimientos del modelo, en función de la relación de dependencia de un objeto respecto a otro.

La primera es una dependencia directa, es decir, un objeto depende directamente de un objeto jerárquicamente superior, de forma que no habrá movimiento relativo entre ambos. En este caso, si el objeto superior se moviese, los que dependen de él se moverían con él. Esta relación se emplea, por ejemplo, entre los sensores infrarrojos y la base del modelo, o entre las mallas visuales y las mallas de cálculo.

La segunda relación de dependencia es una relación de dependencia indirecta, a través de un *joint*. Esta relación permite que el objeto inferior jerárquicamente tenga un movimiento relativo con respecto al superior. Este movimiento dependerá del tipo de *joint*. La estructura sería *objeto superior → joint → objeto inferior*. Este tipo de relación se usaría para unir las ruedas, el pan y el tilt con la base del modelo, como se ve en la figura 43.

7.2 Programación y caracterización de los motores de las ruedas

En este apartado se explicará cómo se ha llevado a cabo la programación de los *joints* que controlan el motor, así como la caracterización de los motores de las ruedas.

7.2.1 Programación de los motores en V-REP

En este apartado se explicará cómo se han programado los *non-threaded scripts* del modelo en V-REP para poder controlar los *joints* a través de Python, empleando la API externa.

En primer lugar, para controlar el movimiento de las ruedas del Robobo real, la librería de Python, robobo.py, tiene tres comandos: *moveWheels()*, *moveWheelsByDegrees()* y *moveWheelsByTime().* El primer comando tiene como argumentos las velocidades de la rueda derecha e izquierda, siendo un valor entre 0 y 100; los argumentos del segundo comando son la rueda que se desea mover (*left, right, both*), los grados que se desea moverla y la velocidad entre 0 y 100; y por último, el tercer comando tiene como argumentos las velocidades de la rueda derecha e izquierda entre 0 y 100 y el tiempo en segundos que se desea mover las ruedas. Sin embargo, internamente la librería Robobo.py trata la función *moveWheels()* como *moveWheelsByTime()* con un tiempo de 10.000 segundos.

Por otro lado, para entender el método de la programación en V-REP de las ruedas, hay que distinguir dos modos de lectura de los *non-threaded scripts*. Cuando se llama a una función definida en un *non-threaded script*, esta función es bloqueante, por lo que la simulación se detiene mientras se está ejecutando y se reanuda cuando se ejecuta la función y esta devuelve el control a la simulación. Cuando el script sólo contiene funciones, las funciones dentro del script se ejecutan una vez, sólo cuando son llamadas. Por otro lado, si el script sólo contiene código, sin una función definida, este código se ejecuta continuamente en cada paso de tiempo de la simulación, actuando como si fuese un bucle.

Basándose en lo dicho anteriormente, en primer lugar, se crea un *non-threaded script* que contenga las dos funciones equivalentes a los comandos que mueven las ruedas de la librería robobo.py. Las dos funciones se llamarán *moveWheelsByDegrees* y *moveWheelsByTime.* El objetivo de estas funciones es recibir valores desde un programa de Python, que se correspondan con los argumentos de cada función, mediante la API externa y generar variables globales con ellos.

Para la función *moveWheelByDegrees*, esta función recibirá el valor de velocidad de las ruedas (datos de tipo entero, de 0 a 100), los grados que deben moverse (datos de tipo entero) y qué rueda debe moverse (cadena de caracteres). Con el dato de porcentaje de velocidad y los grados que tiene que moverse la rueda se calculará el tiempo necesario para que la rueda se mueva ese ángulo. El tiempo se calculará a partir del polinomio que se obtendrá en el apartado 7.2.2 en el que se obtiene la posición en grados de la rueda en función del tiempo y la velocidad (de 0 a 100). En las variables globales se guardarán la velocidad de las ruedas en función de la cadena de caracteres que puede ser *left, right* o *both*. Si el string fuese *left*

solo se guardaría en una variable global la velocidad de la rueda izquierda, mientras que la velocidad de la rueda derecha sería 0. También se guarda en variables globales el tiempo de simulación en el que se llamó a la función (dato de tipo float) y el tiempo calculado en esta misma función. Como pasaba en la anterior función, estos valores de tiempo hay que guardarlos en variables distintas, cada una leída por el script de cada rueda.

En el caso de *moveWheelsByTime*, la función recibirá un vector con los valores de las velocidades de las ruedas (dato de tipo entero, de 0 a 100) y el tiempo que debe moverse (dato de tipo float) y los guardará en variables globales, además del tiempo de simulación en el que se llama a la función. Como pasa en las otras funciones, debe haber dos variables globales para cada uno de los datos. Cuando desde Python se use el comando *moveWheels*, la función *moveWheelsByTime* recibirá los valores de las velocidades de las ruedas y un tiempo de 10.000 segundos.

Al llamar a las funciones de este script desde el API externa de Python se pasan los valores de los argumentos del comando real y que se le quieren enviar al modelo, a variables globales que pueden ser leídas por el resto de *scripts*.

A continuación, se crean dos *non-threaded scripts* más, cada uno asociado a un *joint* de las ruedas del modelo, en donde se desarrolla un código que lea las variables globales referentes a cada rueda. Estos scripts se ejecutan constantemente y actúan como un bucle. En cada uno de ellos, cada vez que se ejecutan, se comprueba el valor de la variable global que guarda el valor en porcentaje de la velocidad de cada rueda y cuando es distinto de 0, se asigna al *joint* que controla la rueda una velocidad calculada a partir del polinomio del apartado 7.2.2, moviendo de esta forma el modelo en la simulación. La velocidad se le asigna a los *joints* con comandos de la API interna de V-REP, que permiten controlar los elementos de la simulación. La velocidad se calcula como los grados recorridos según el polinomio para un porcentaje de velocidad y un tiempo, y dividiéndolos entre el tiempo. En cada paso de tiempo también se comprueba que la diferencia entre el tiempo actual de la simulación y el tiempo de la simulación en el que se ejecutó la función es menor que el tiempo que se deben mover las ruedas. Una vez que se supera el tiempo de duración, se le asigna al *joint* velocidad 0 y se reinician a 0 las variables globales hasta que llamando a una nueva función se les da otro valor.

En la figura 45 se muestra de forma esquemática la lógica de funcionamiento de los motores, explicada en este apartado.

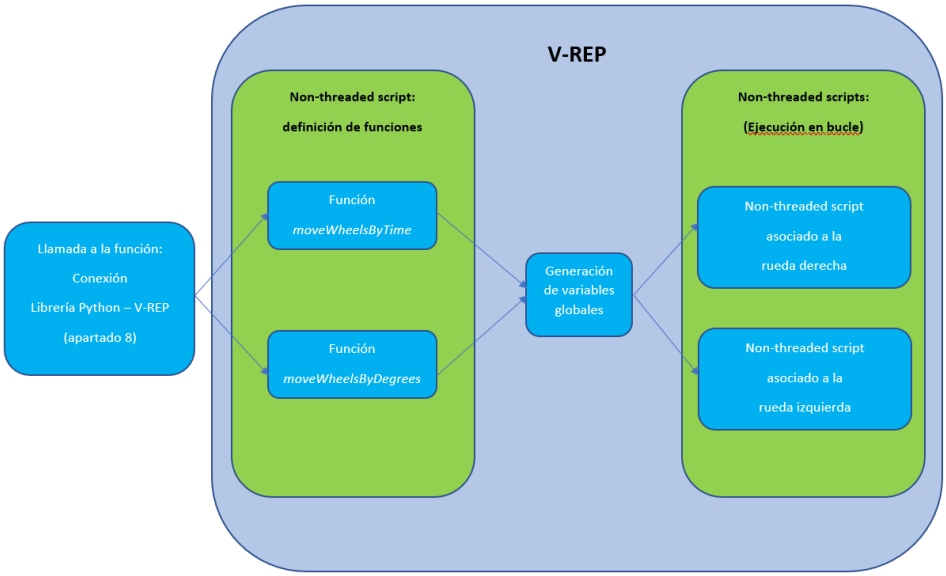

Figura 45: Lógica de la programación de los motores de las ruedas del modelo.

El código de las funciones anteriores, así como de los scripts que se encargan del movimiento de los *joints* de las ruedas, está recogido en el anexo 2.

7.2.2 Caracterización del comportamiento de los motores

El objetivo de la caracterización de los motores de las ruedas es obtener una expresión matemática que permita relacionar el valor en porcentaje de velocidad que se les pasa a los comandos encargados de mover las ruedas de la librería robobo.py, con un valor de velocidad en grados por segundo, para que cuando al modelo en V-REP se le envíe un porcentaje de velocidad, el motor se mueva a la velocidad real equivalente a ese porcentaje.

Para esto se llevan a cabo una serie de medidas con el Robobo real usando el comando *moveWheelsByTime()*. De esta forma se obtienen las distancias que se ha desplazado el robot para distintos valores de porcentaje de velocidad y tiempo. En la figura 46 se muestran cómo se obtienen las distancias en las pruebas realizadas.

Figura 46: Medida de la distancia recorrida por el Robobo.

Posteriormente las distancias se convertirán a ángulo recorrido por la rueda. Los porcentajes de velocidad escogidos para realizar las pruebas se muestran en la tabla 1.

Velocidades (%)	2	4	6	10	15	20	30	40	60	90

Tabla 1: Velocidades seleccionadas para la caracterización de los motores de las ruedas.

Se han escogido valores más próximos entre sí para velocidades muy bajas, teniendo así más información para esos puntos. Para cada velocidad, se realizan 8 pruebas con los valores de tiempo mostrados en la tabla 2.

Tiempos (s)	0.25	0.50	1.00	1.50	2.00	3.00	6.00	8.00

Tabla 2: Tiempos escogidos en la caracterización de los motores de las ruedas.

En este caso, también se cogen valores más próximos entre sí para tiempos muy pequeños para tener más información del comportamiento del motor en el momento del arranque para cada velocidad.

Una vez realizadas las pruebas, el objetivo es obtener una ecuación que permita calcular el ángulo recorrido por la rueda, Θ, en función del tiempo y el porcentaje de velocidad.

Se va a considerar que el ángulo recorrido por la rueda se puede expresar como una ecuación del tipo: $\theta = A * t + B$. En esta ecuación *A* sería el término de la velocidad angular, *t* el tiempo y *B* el término independiente.

Tanto el término *A* como el *B* se considerarán polinomios de grado 3 que dependan del porcentaje de velocidad, *p*. De esta forma, la ecuación final obtenida será:

$$\theta = (C * p^3 + D * p^2 + E * p + F) * t + (G * p^3 + H * p^2 + I * p + J)$$

Con el polinomio desarrollado y las medidas realizadas con el Robobo, se calcularán los coeficientes C, D, E, F, G, H, I y J de forma que el error cuadrático medio entre el valor de ángulo recorrido calculado y el obtenido en las medidas reales sea mínimo, mediante el complemento *Solver* de Excel. La única restricción que se establece es que los coeficientes tengan un valor entre -100 y 100. Los resultados obtenidos de esta forma para los coeficientes se muestran en la tabla 3.

C	D	E	F	G	H	I	J
-0.00005	0.00522	6.35708	51.36677	-0.00033	0.04285	-2.06405	-17.69727

Tabla 3: Coeficientes obtenidos como resultado de la optimización.

Con el polinomio que se obtiene sustituyendo los coeficientes por su valor, se podrán calcular los valores de velocidad angular para un porcentaje y un tiempo determinados. En el caso del comando *moveWheelsByDegrees*, conociendo el ángulo que se quiere recorrer y el porcentaje de velocidad, que son los argumentos que recibe la función, se puede despejar el tiempo, *t*, del polinomio y obtener la velocidad como Θ/t.

En la figura 47 se muestra la comparación entre la curva real y la curva calculada por el polinomio obtenido de la caracterización para la velocidad 60, en la que se aprecia el grado de aproximación del polinomio.

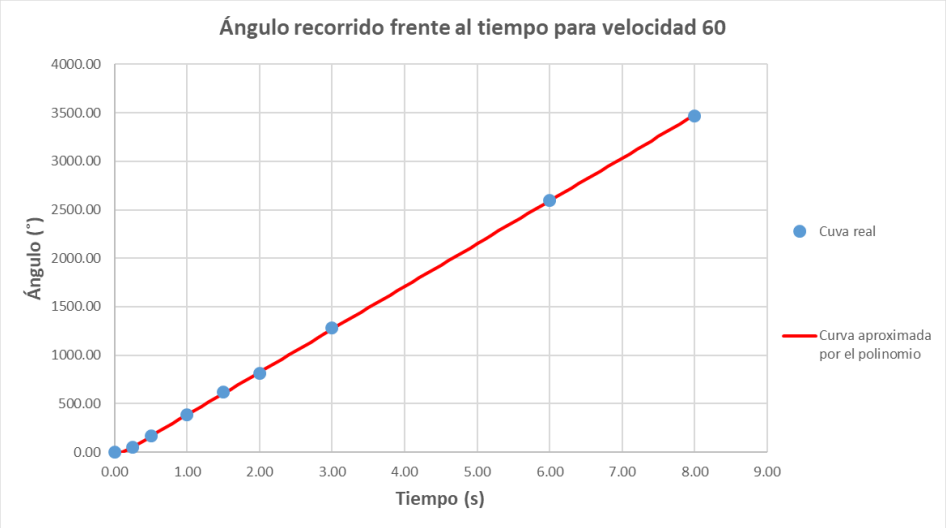

Figura 47: Comparación de los resultados de velocidad reales y aproximado por el polinomio para la velocidad 60.

En el anexo 1 está incluida la tabla con los resultados de la caracterización. En ella se muestran los valores de los coeficientes optimizados para minimizar el error cuadrático medio entre los grados girados por la rueda real y el valor calculado por el polinomio. también se muestran para cada velocidad y cada tiempo asignados al motor, la distancia recorrida por la rueda en cm y los grados equivalentes, los grados que gira la rueda según el polinomio y el error del polinomio, tanto en grados como en centímetros. En las dos últimas columnas se muestra el error cuadrático en grados y el valor absoluto del error, también en grados. Al final de la tabla se muestra un resumen que incluye el error cuadrático medio y el error medio, en grados, y el error máximo, tanto en grados de giro de la rueda como en cm recorridos por la misma.

7.3 Programación y caracterización del motor de la unidad pan

En los dos subapartados siguientes se explicará cómo se ha llevado a cabo la programación de los *joints* que controlan el pan, así como la caracterización del motor real del mismo.

7.3.1 Programación del motor en V-REP

El principio de funcionamiento de programación del motor del pan es el mismo que el de las ruedas, por lo tanto, ya no se explicará de nuevo.

En este caso, en el mismo script en el que están definidas las funciones que generan las variables globales de los motores de las ruedas se añade otra función llamada *movePanTo*. Esta función recibe como argumentos la posición objetivo del pan, que es un dato de tipo entero que puede variar en el siguiente intervalo [-160,160] grados; y el porcentaje de velocidad, que como ya se ha dicho anteriormente, es un entero entre 0 y 100. Antes de declarar las variables globales, los datos de la posición objetivo y de velocidad son filtrados para que sus valores estén entre siempre -160 y 160 y entre 0 y 100. En la función se guardan en variables globales la posición objetivo e inicial del pan, y el porcentaje de velocidad para que puedan ser leídas por un script asociado al *joint* del pan.

En el script asociado al pan comprueba la posición objetivo del pan con su posición inicial, para saber en qué sentido debe girar el motor. Para esto, se le da alguno de estos tres valores [-1, 0, 1] a un parámetro multiplica a la velocidad que se le asigna al *joint*. Conocido el sentido de giro del *joint*, se le asigna la velocidad angular equivalente al valor de porcentaje que recibe, calculada a través del polinomio del apartado 7.3.2, y se comprueba que el valor absoluto de la diferencia entre la posición objetivo y la actual sea menor a un límite. Cuando se supera ese límite, se le asigna al *joint* velocidad 0 y las variables globales se reinician con valor 0.

En la figura 48 se representa de forma esquemática la relación entre los scripts que controlan el pan.

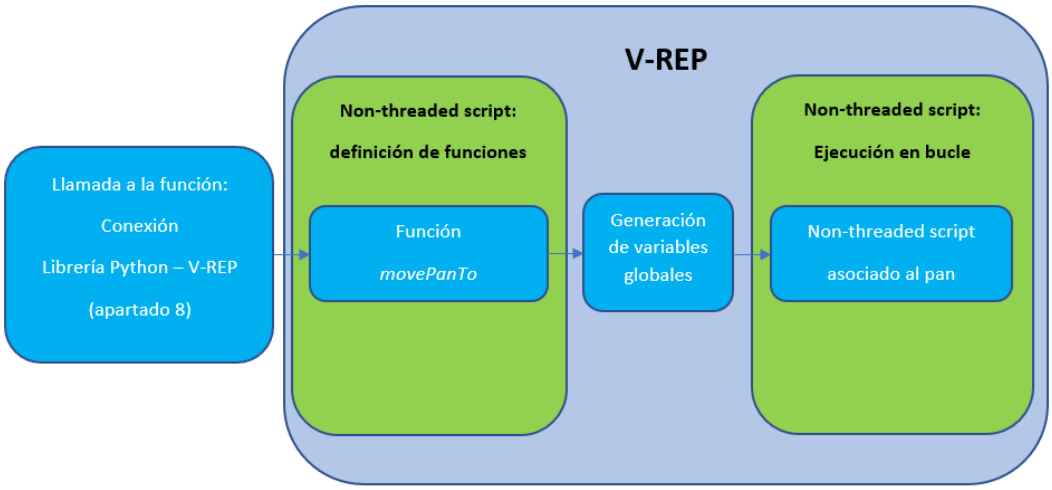

Figura 48: Lógica de programación del motor del pan en el modelo de V-REP.

La función *movePanTo* se incluye en el script en el que están incluidas las funciones de los *joints* de las ruedas. También se incluye en el anexo 2 el script que controla el *joint* de la unidad pan.

7.3.2 Caracterización del comportamiento del motor

Para llevar a cabo la caracterización del motor del pan, igual que se ha hecho en el caso de la caracterización de los motores de las ruedas, se van a realizar una serie de medidas con el Robobo real usando el comando *movePanTo* de la librería robobo.py. En este caso, se han realizado medidas cronometrando el tiempo que tarda el pan en recorrer un ángulo concreto a una velocidad determinada. Los porcentajes de velocidad seleccionados para los experimentos se muestran en la tabla 4.

Velocidades (%)	2	4	6	10	20	30	40	60	80

Tabla 4: Velocidades seleccionadas para la caracterización del motor del pan.

Para cada velocidad, se cronometrará el tiempo que tarde el pan en recorrer los ángulos motrados en la tabla 5.

Ángulos (˚)	15	30	45	60	75	90	110	135	160

Tabla 5: Tiempos escogidos en la caracterización del motor del pan.

Para cada combinación se toman tres valores de tiempo y se calcula el promedio.

Para obtener una ecuación que permita modelar el comportamiento del motor del pan se parte de la misma ecuación que en el caso de las ruedas, y considerando cada término como un polinomio de tercer grado se obtiene el mismo polinomio que en el caso de la caracterización de las ruedas:

$$\theta = (C * p^3 + D * p^2 + E * p + F) * t + (G * p^3 + H * p^2 + I * p + J)$$

En este caso se buscará minimizar el error cuadrático medio entre el tiempo cronometrado y el teórico, despejado del polinomio anterior:

$$t = \frac{\theta - (G*p^3 + H*p^2 + I*p + J)}{(C*p^3 + D*p^2 + E*p + F)}.$$

Empleando el complemento *Solver* de Excel para calcular los coeficientes que harían mínimo el error entre el tiempo real y el calculado, los coeficientes serían los mostrados en la tabla 6:

C	D	E	F	G	H	I	J
-0.00003	0.00388	0.84747	8.05468	-0.00002	0.00106	-0.33034	-0.89074

Tabla 6: Coeficientes obtenidos como resultado de la optimización.

A partir del polinomio obtenido, con el ángulo recorrido y el porcentaje de velocidad asignado se puede obtener el tiempo, y de ahí la velocidad angular.

En la figura 49 se muestran la comparación entre los valores de tiempo promediado obtenidos de los experimentos y los obtenidos de despejar el tiempo del polinomio calculado, para una velocidad de 60, dando una idea de la exactitud de la caracterización.

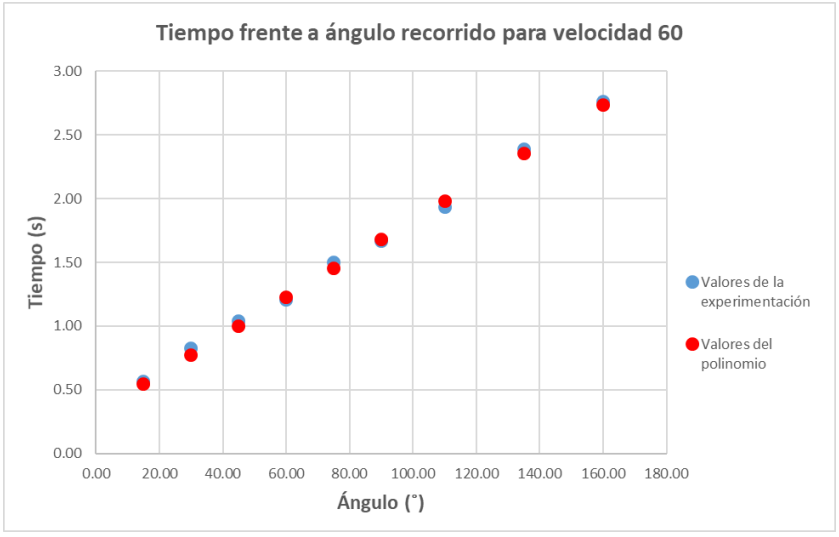

Figura 49: Comparación de los resultados de tiempo reales y aproximados para la velocidad 60.

La tabla de resultados de la caracterización está incluida en el anexo 1. En ella se muestra, de la misma forma que en el caso de la caracterización de los *joints* de las ruedas, los valores de los coeficientes optimizados para que error cuadrático medio entre los segundos que el pan tarda en llegar a la posición en la experimentación y el tiempo calculado según el polinomio sea mínimo. La tabla también incluye los distintos tiempos que tarda el pan en llegar a una posición objetivo, para una velocidad y una posición concretas, así como el tiempo promedio, y el tiempo ajustado con el polinomio obtenido. Las dos últimas columnas muestran el error en segundos y el cuadrático, para cada caso. Al final de la tabla se incluye un resumen que indica el error cuadrático medio, el error medio y el error máximo de los casos estudiados para realizar la caracterización.

7.4 Programación y caracterización del motor de la unidad tilt

En este apartado se describirá la programación de los *joints* que controlan el tilt, así como el método seguido para caracterizar el motor del tilt real.

7.4.1 Programación del motor en V-REP

La programación del motor del tilt sigue el mismo procedimiento que el motor del pan o y de las ruedas.

En el caso del tilt, en el script en el que están definidas las funciones que generan las variables globales de los motores de las ruedas y del motor del pan se añade otra función llamada *moveTiltTo*. Esta función recibe como argumentos la posición objetivo del tilt, que es un dato de tipo entero que varía entre 5 y 105 grados; y el porcentaje de velocidad. Antes de declarar las variables globales, los datos de la posición y el porcentaje de velocidad son filtrados para que sus valores estén entre 5 y 105 y entre 0 y 100. Posteriormente se guardan en variables globales la posición objetivo e inicial del tilt, y el porcentaje de velocidad para que puedan ser leídas por el script asociado al *joint* del tilt.

En el script asociado al tilt, igual que en el del pan, se comprueba la posición objetivo del tilt con su posición inicial, para saber en qué sentido debe girar el joint. A partir de ahí, se le asigna al *joint* la velocidad equivalente al porcentaje que se le manda, calculada con el polinomio del apartado 7.4.2, y se comprueba que el valor absoluto de la diferencia entre la posición actual y la objetivo sea menor a un límite. Cuando se supera ese límite, se le asigna al *joint* velocidad 0 y las variables globales se reinician con valor 0.

En la figura 50 se muestra cómo se estructuran los scripts que controlan la unidad tilt.

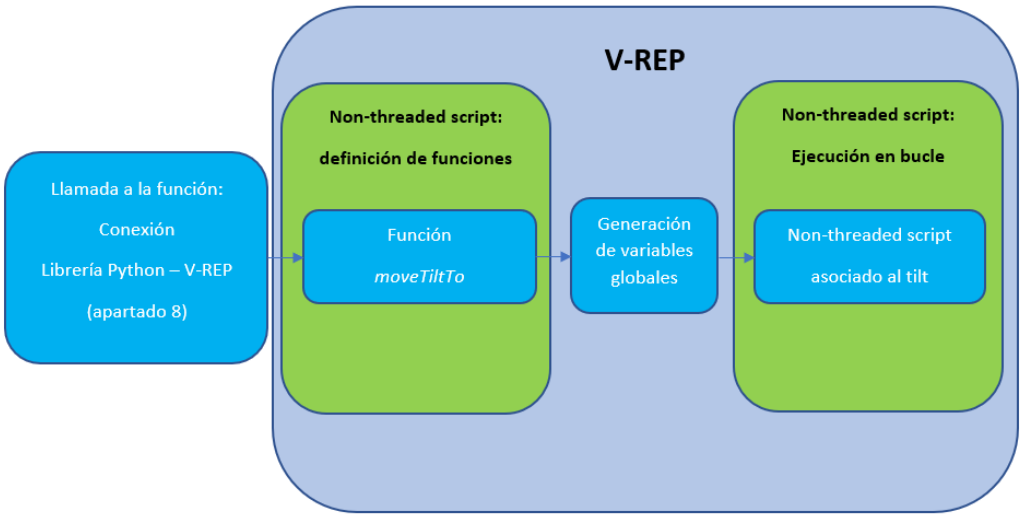

Figura 50: Lógica de programación del motor del tilt en el modelo de V-REP.

El código del script que controla el *joint* del tilt se encuentra en el anexo 2. Por otro lado, la función que se encarga de pasar los parámetros que le llegan, a variables globales está en el script que incluye el resto de funciones.

7.4.2 Caracterización del comportamiento del motor

Para llevar a cabo la caracterización del motor se sigue el mismo procedimiento empleado en el caso del pan. En las pruebas realizadas con el Robobo real se emplea el comando *moveTiltTo* de la librería robobo.py. Para cronometrar los tiempos que tarda el tilt en recorrer un ángulo determinado, los porcentajes de velocidad seleccionados para los experimentos se muestran en la tabla 7.

Velocidades (%)	2	4	6	10	20	30	40	60	80

Tabla 7: Velocidades seleccionadas para la caracterización del motor del tilt.

Para cada velocidad, se cronometrará el tiempo que tarde el tilt en recorrer los ángulos mostrados en la tabla 8. La posición inicial se considerará 5 ˚, es decir, el smartphone estará prácticamente paralelo al suelo.

Ángulos (˚)	15	30	45	60	75	90	100

Tabla 8:Tiempos escogidos en la caracterización del motor del tilt.

Para cada combinación se toman 3 valores de tiempo y se calcula el promedio, como en el caso del pan.

Para obtener una ecuación que permita modelar el comportamiento del motor del tilt se parte de la misma ecuación que en el caso del pan, y considerando cada término como un polinomio de grado 3 se obtiene el mismo polinomio:

$$\theta = (C * p^3 + D * p^2 + E * p + F) * t + (G * p^3 + H * p^2 + I * p + J)$$

Para el tilt también se comparará el tiempo real con el tiempo teórico que tardaría en recorrer los grados que se le indique en el comando, y se calcularán los coeficientes para minimizar el error cuadrático medio. El tiempo teórico se calcula despejando el tiempo del polinomio anterior:

$$t = \frac{\theta - (G*p^3 + H*p^2 + I*p + J)}{(C*p^3 + D*p^2 + E*p + F)}.$$

Empleando el complemento *Solver* de Excel para calcular los coeficientes que harían mínimo el error entre el tiempo real y el calculado los resultados Se muestran en la tabla 9:

C	**D**	**E**	**F**	**G**	**H**	**I**	**J**
0.00001	-0.00260	0.48094	3.18153	-0.00009	0.01233	-0.54953	4.73795

Tabla 9: Coeficientes obtenidos como resultado de la optimización.

A partir de este polinomio, con el ángulo recorrido y el porcentaje de velocidad asignado se puede obtener el tiempo, y de ahí la velocidad angular.

En la figura 51 se muestran la comparación entre los valores de tiempo promediado obtenidos de los experimentos y los obtenidos de despejar el tiempo del polinomio calculado, para una velocidad de 60, dando una idea de la exactitud de la caracterización.

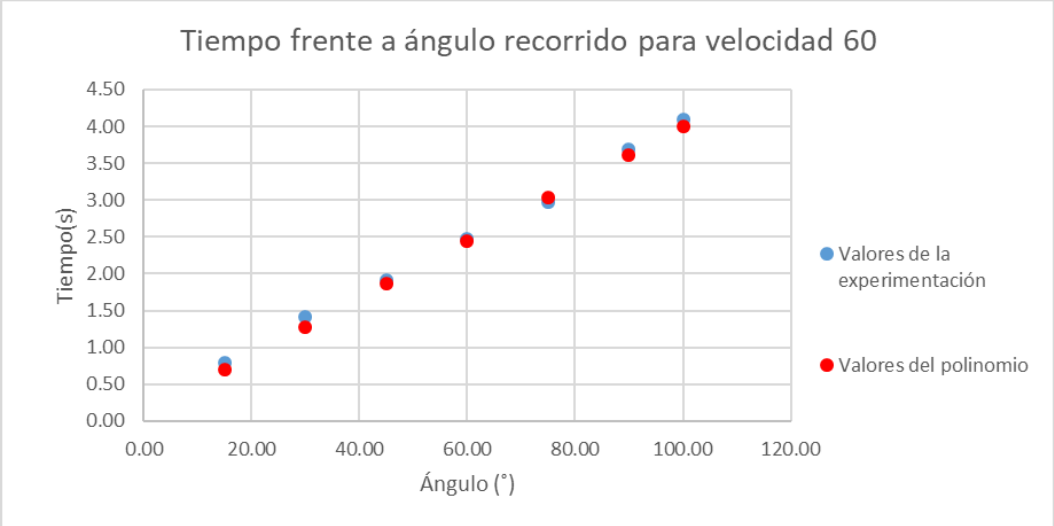

Figura 51: Comparación de los resultados de tiempo reales y aproximados para la velocidad 60.

La tabla de resultados de la caracterización está incluida en el anexo 1. En ella se muestra, los mismos campos que en el caso del pan, es decir, los valores de los coeficientes optimizados para que error cuadrático medio entre los segundos que el tilt tarda en llegar a la posición en la experimentación y el tiempo calculado según el polinomio sea mínimo, los tiempos que tarda el tilt en llegar a una posición objetivo para una velocidad y una posición concretas, el tiempo promedio, y el tiempo ajustado con el polinomio obtenido. Las 2 últimas columnas muestran el error en segundos y el cuadrático, para cada caso. Al final de la tabla se incluye un resumen que indica el error cuadrático medio, el error medio y el error máximo de los casos estudiados para realizar la caracterización.

7.5 Modelado, programación y calibración de los sensores de proximidad.

En este apartado se explicará cómo se han modelado los sensores de proximidad y cómo se han programado para que su comportamiento sea análogo al de los sensores infrarrojos reales, a partir de la calibración realizada.

7.5.1 Modelado de los sensores de proximidad

En la figura 41 se mostró la distribución y orientación de los sensores y una representación del modelo básico de detección. En este apartado se explicarán los cambios realizados en los modelos previamente desarrollados para obtener unos sensores que mejoren el comportamiento de los anteriores y aumenten su similitud con los sensores infrarrojos reales. A continuación, se describirán los distintos modelos probados para realizar la modelización de los sensores infrarrojos, sus ventajas y desventajas y el modelo final.

Como ya se ha explicado en el apartado 5.6, uno de los objetivos de este trabajo era mejorar los sensores infrarrojos implementados en el modelo de Robobo que había disponible. En él, los sensores infrarrojos fueron modelados en V-REP con sensores de proximidad. Para representar el sensor infrarrojo se realizó un volumen de detección que a su vez estaba formado por tres subvolúmenes. La forma del volumen modelado variaba en función de si el sensor detectaba el suelo o no. En la figura 52 se muestra el modelo de sensor para detectar el suelo (derecha) y para detectar el resto de objetos (izquierda).

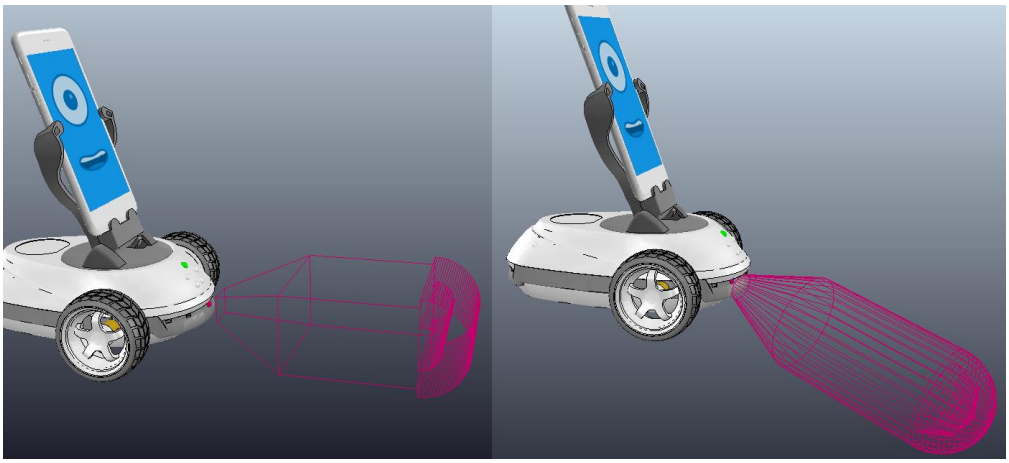

Figura 52: Modelo de sensores infrarrojos basados en volúmenes.

A pesar de que estos modelos son muy sencillos y fáciles de implementar, para este trabajo se consideró que, dado que un volumen de detección de un sensor de proximidad en V-REP sólo ofrece la distancia mínima al objeto, un sensor modelado de esta forma da muy poca información sobre la posición del objeto dentro del volumen de detección. Es decir, a medida que un objeto entra en el volumen de detección del sensor infrarrojo, el área del objeto que refleja luz es mayor y, por lo tanto, el valor devuelto por el sensor infrarrojo se incrementará. Sin embargo, el sensor modelado de esta manera no tendría en cuenta este aspecto, de forma que el modelo planteado en este proyecto debe resolver este inconveniente.

Para realizar un nuevo modelo del sensor, en primer lugar, se ha realizado un estudio del volumen de detección del sensor infrarrojo real. Antes de realizar las mediciones es necesario definir un material, tanto para el suelo sobre el que se está trabajando como para el objeto que vaya a ser detectado por el sensor infrarrojo. Esto es debido a que las mediciones registradas por el sensor infrarrojo variarán en función del material, del color o de la superficie de los objetos detectados, por lo que se hace necesario estandarizar estas características. El material empleado será papel blanco, debido a que tiene una buena reflectividad.

Para estudiar el volumen de detección se colocará el Robobo centrado sobre una hoja de papel cuadriculada como se muestra en la figura 53.

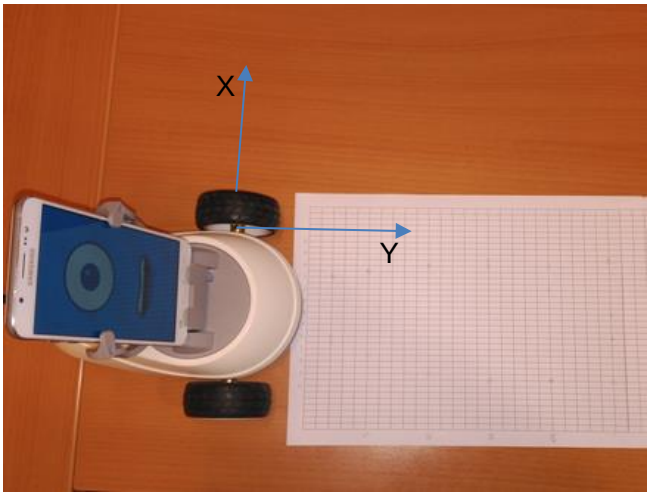

Figura 53: Método para calcular el volumen de detección del sensor IR.

Si en la figura 53 se considera el eje X paralelo al eje de las ruedas y el eje Y el perpendicular, para obtener el perfil de detección del sensor infrarrojo se desplazará una lámina de cartón, recubierta de papel blanco, a lo largo del eje X para encontrar el punto donde el sensor empieza a detectar. Esta operación se repetirá alejando la lámina cada 5 cm en el

eje Y. El motivo de que se emplee una lámina y no una caja, por ejemplo, es que las paredes laterales no influyan en la medida del sensor.

Se emplearán dos láminas de distintos tamaños para realizar estas pruebas, una de 23x18,5 cm y otra de 8,5x5,4 cm, dando cada una un perfil de detección distinto. La vista en planta del área de detección para cada lámina se muestra en la figura 54.

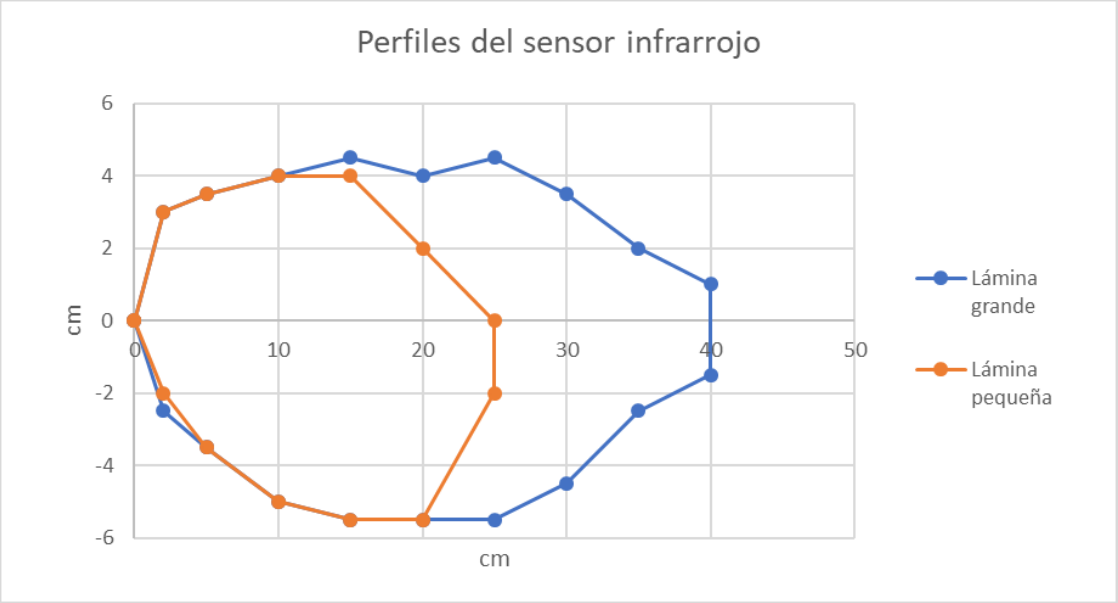

Figura 54: Perfiles de detección del sensor infrarrojo.

Para modelar el volumen de los nuevos sensores, se tomará de referencia el perfil azul, ya que es el menos restrictivo. Si se escogiese el perfil más pequeño habría puntos en los que un objeto no sería detectado cuando sí debería serlo. El perfil seleccionado tiene un alcance frontal de 40 cm, un alcance lateral derecho de 5,5 cm y un alcance lateral izquierdo de 4,5 cm.

Partiendo de ese perfil se propone un modelo para representar el volumen de detección del sensor infrarrojo, basado en la discretización del volumen en sensores con forma de rayo que ocupen aproximadamente el mismo espacio. De esta forma, un sensor estará formado por múltiples sensores que formarán un volumen. Cada sensor deberá estar definido como tipo rayo y se define según un rango (la distancia que abarca) y un offset (la distancia entre la posición origen del sensor y el punto a partir del cual el sensor empieza a medir realmente). El diálogo de propiedades se muestra en la figura 55.

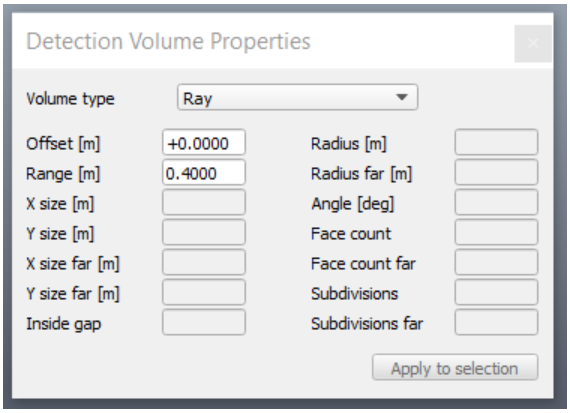

Figura 55: Diálogo de propiedades del sensor tipo rayo.

7.5.1.1 Modelo de haces paralelos

El primer modelo propuesto está formado por 13 sensores con una distribución en planta como la mostrada en la figura 56.

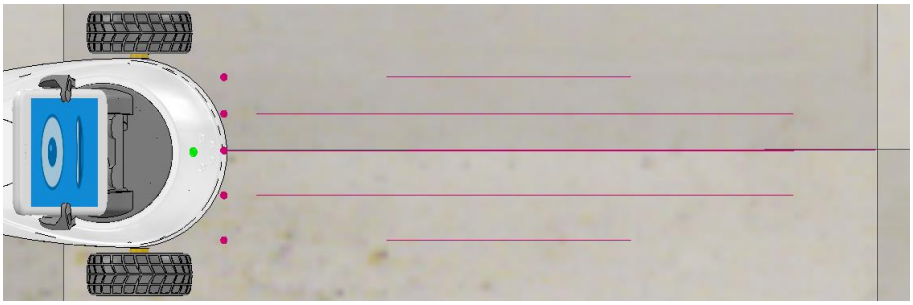

Figura 56: Vista en planta de los sensores de proximidad del primer modelo de infrarrojo.

El sensor central tiene un rango de 40 cm y un offset de 0. Por otro lado, los dos sensores que aparecen por encima en la imagen están separados entre sí y del censor central 2,25 cm y tienen un rango de 33 y 15 cm, y un offset de 2 y 10 cm respectivamente. Por último, los dos sensores que aparecen por debajo del sensor central en la imagen están separados entre sí y del sensor central 2,75 cm y tienen los mismos valores de rango y offset que sus equivalentes al otro lado del sensor central.

A mayores, se incluirán más sensores en planos situados tanto por encima como por debajo del sensor central, que se aprecian mejor en la figura 57.

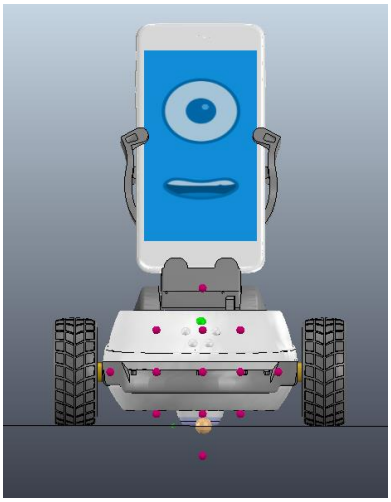

Figura 57: Vista frontal de los sensores de proximidad.

Los planos que están por encima y por debajo del plano central están separados entre sí a 2,75 cm, y las distancias laterales entre sensores siguen el mismo criterio que el explicado para la figura 42. Los sensores situados en los extremos superior e inferior tienen un rango de 15 cm y un offset de 10 cm, mientras que los 9 sensores que rodean al sensor central tienen un rango de 33 cm y un offset de 2 cm.

Para finalizar la explicación de este modelo, en la figura 58 se muestra una vista en perspectiva del Robobo y los sensores de proximidad que conforman el sensor central.

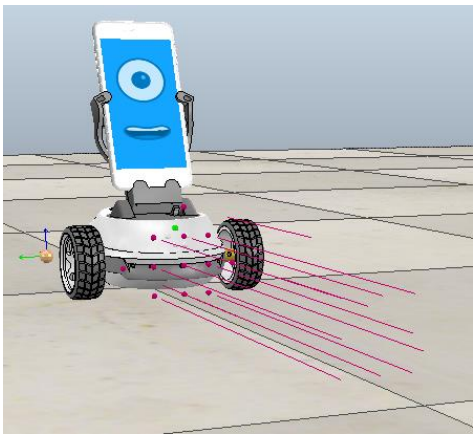

Figura 58: Vista en perspectiva del modelo de sensor IR.

Tras trabajar este modelo, se ha detectado una serie de inconvenientes. Uno de ellos es que la precisión de detección está limitada a la separación que hay entre los sensores, por lo tanto, cualquier movimiento del objeto a detectar que sea menor a la separación entre los sensores va a ser imperceptible. A mayores, la zona de detección del sensor más cercana al Robobo está muy poco definida debido a los offset de los distintos sensores y, por lo tanto, la detección de un objeto en esa zona sería también muy poco precisa. Además, el hecho de que los rayos de los sensores sean paralelos, tiene como consecuencia que no se detecten las paredes laterales de un objeto cuando su cara frontal sea perpendicular a los rayos, lo que supone que para el sensor sería lo mismo una lámina que una caja, por ejemplo. Esto no ocurre en el sensor real ya que las paredes laterales también reflejan luz, lo cual incrementa el valor devuelto por el sensor infrarrojo. También es evidente, como se ve en la figura 57, que los sensores no parten del mismo punto, sino del mismo plano, lo que supone otra diferencia con el robot real. Por otro lado, el sensor situado en la posición más baja en la figura 57 no va a detectar nada debido a que se encuentra por debajo del nivel del suelo. Por último, se detectó también otro factor que surge de una limitación del simulador. Dado que el origen de los sensores se separa del sensor central, podría darse la situación de que un sensor se introdujese en el interior de un objeto simulado, impidiendo que pueda detectar la intersección con el mismo. Esto produciría que las paredes paralelas a los rayos o aquellas que estén muy cercada no fuesen detectadas.

7.5.1.2 Modelo de haces radiales

Con el objetivo de mejorar los problemas descritos en el párrafo anterior se elabora un nuevo modelo de sensor, cuyo volumen de detección estará formado esta vez por 41 sensores de proximidad de tipo rayo. El modelo se genera partiendo de una distribución en planta como la mostrada en la figura 59, a la altura del sensor central (3,22 cm).

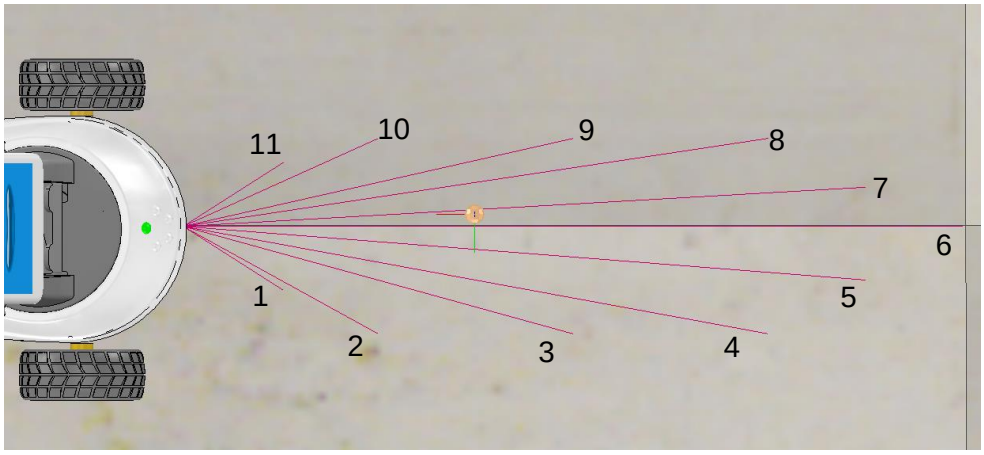

Figura 59: Punto de partida del nuevo modelo de sensor IR.

Si se empezasen a numerar los sensores empezando por los que están por debajo del sensor central en la figura 59, las longitudes de los sensores y su desviación respecto al sensor central en el plano mostrado en la figura 59 serían los mostrados en la tabla 10.

Nº de sensor	Longitud (cm)	Desviación respecto al sensor central (˚)
1	6,10	-32,47
2	1,41	-28,81
3	20,74	-15,38
4	30,50	-10,39
5	35,11	-4,49
6	40,00	0
7	35,06	3,67
8	30,64	8,53
9	20,50	12,68
10	10,96	24,22
11	6,10	32,47

Tabla 10: Dimensión y desviación de los sensores del modelo de sensor IR.

El resto del volumen de detección se formaría copiando y rotando los sensores 1, 2, 3, 4 y 5, sobre el eje del sensor central (sensor 6), 45, 90, -45 y -90 grados; y los sensores 7, 8, 9, 10 y 11, 45 y -45 grados, también sobre el eje del sensor central. El resultado final sería el mostrado en la figura 60.

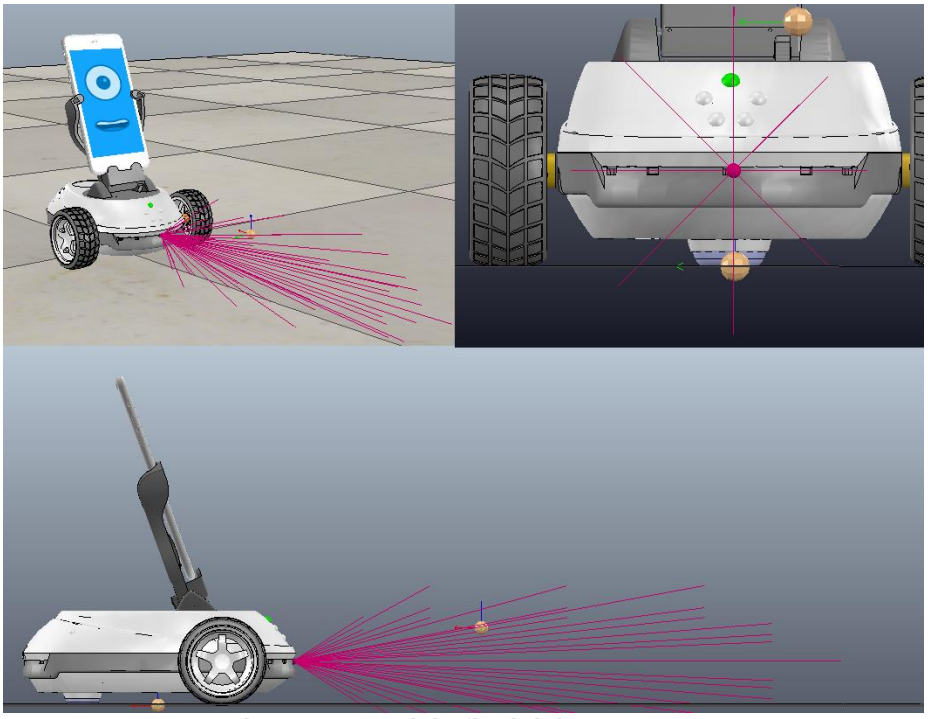

Figura 60: Modelo final del sensor IR.

De esta forma se mejora el modelo en cuanto a la densidad de rayos, aumentando la precisión de detección, y resolviendo el problema de detección de superficies laterales. Además, el hecho de que todos los rayos partan del mismo punto hace que el comportamiento sea más realista, pareciéndose más al comportamiento del infrarrojo real. Por último, también se resuelve el problema del sensor que quedaba enterrado en el suelo y que no aportaba información.

7.5.2 Programación de los sensores de proximidad

Como pasaba con la programación de las ruedas, del pan y del tilt, para programar la función de lectura de los sensores infrarrojos se crea una función en el mismo script en el que se han creado el resto de funciones, llamada *readAllIRSensor* y que solo se ejecutará una vez cuando se la llame, según el funcionamiento de los *non-threaded scripts*. En esta función se obtendrá la distancia que está leyendo cada uno de los rayos que forman el volumen de detección de cada sensor (41 rayos por sensor), y que serán las entradas a una función que calcule el valor que el sensor infrarrojo debería mostrar realmente para cada combinación de distancias. La función que relaciona los valores de las 41 distancias de los sensores de proximidad con el valor final del infrarrojo se explicará en el apartado 7.5.3. Esta operación se realizará para los ocho sensores y el valor final de cada uno se guardará en una posición de un vector de 8 posiciones, una para cada uno de los sensores. Es decir, cada vez que se llame a la función *readAllIRSensor* se calculará un vector con los ocho valores de los sensores infrarrojos, que será devuelto por la función. El valor de los infrarrojos, a pesar de que se calcula como un valor decimal, el valor que se guarda en el vector es un valor entero. A cada valor de sensor se le sumará un porcentaje aleatorio que variará entre el -5% y 5% de su valor para simular el ruido que afecta al sensor real.

La diferencia entre la programación de los sensores y la programación de los *joints* es que, en el caso de los sensores, no será necesario crear un script que actúe sobre el modelo, ya que solo será una función de cálculo y lectura.

El código de la función *readAllIRSensor* está contenido en el anexo 2, en el script que incluye el resto de funciones relacionadas con los joints de las ruedas, el pan y el tilt.

7.5.3 Caracterización de los sensores de proximidad

Partiendo de que todos los sensores integrados en el Robobo son iguales, para caracterizar los sensores de proximidad que forman cada uno de los sensores infrarrojos del modelo, las medidas se realizarán con el sensor central infrarrojo central del Robobo.

Para que la geometría del objeto detectable no influya en las mediciones del sensor, se empleará siempre el mismo objeto, que será la caja que contiene al propio Robobo, con unas medidas de 23 cm de ancho, 10 cm de largo y 18,5 cm de alto. Además, debido a que el color y la superficie del objeto también hará variar el valor medido por el infrarrojo, se escogerá un material estándar para realizar las mediciones. Como ya se ha explicado en el apartado 7.5.1, se emplearán hojas de papel blanco para recubrir las superficies detectadas por el sensor, dado a su buena reflectividad. Por lo tanto, la caja empleada en las detecciones se recubrirá de papel blanco.

El objetivo de la caracterización es encontrar una relación entre las medidas de los 41 sensores de proximidad del modelo de V-REP y el valor que mide el sensor infrarrojo real para un mismo objeto con la misma posición respecto al Robobo. Para esto, se realizarán 63 medidas con el Robobo real variando la posición del objeto lateralmente, así como la distancia entre el objeto y el robot desde 5 cm a 35 cm incrementando de 5 en 5 cm la separación, como se muestra en la figura 61.

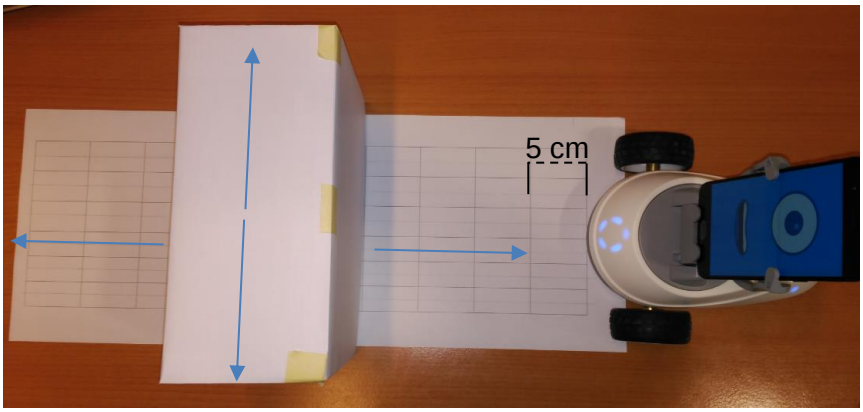

Figura 61: Medición con el sensor infrarrojo del Robobo.

Por otro lado, para las mismas posiciones del objeto que las consideradas para realizar las medidas del sensor infrarrojo, se realizan las medidas en V-REP para los sensores de proximidad, en las cuales se obtendrán las distancias de cada uno de los sensores.

Con este método se obtendrán 63 combinaciones de distancias de los 41 sensores y 63 valores medidos con el sensor infrarrojo real de forma que ya se puede establecer una relación entre las distancias y el valor medido.

Para tener una idea aproximada de cómo será la relación entre las distancias medidas por los sensores de proximidad en el modelo y el valor real del infrarrojo se representarán los valores medidos por el infrarrojo frontal del Robobo frente a las distancias entre la caja y dicho sensor, como se muestra en la tabla 11.

Distancia sensor-caja	Valor IR
5	414
10	117
15	57
20	35
25	26
30	21
35	18

Tabla 11: Distancias entre el sensor y la caja y valor del sensor IR correspondiente.

La gráfica obtenida a partir de estos datos se muestra en la figura 61.

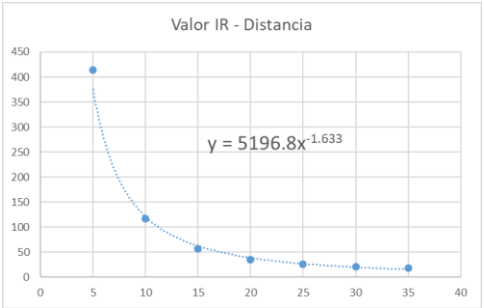

Figura 62: Valor leído por el sensor infrarrojo real frente a la distancia al objeto.

De la gráfica de la figura 61 se puede suponer que el valor que aportará cada sensor de proximidad al valor total del infrarrojo se podría aproximar como:

$$IR_n = a_n * {d_n}^{-b_n}$$

donde d es la distancia medida por el sensor. Para poder emplear esta fórmula, hay que tener en cuenta que los sensores de proximidad de V-REP dan un valor de 0 si no detectan nada, por lo que habría que transformar los valores de 0 en valores muy altos para que el valor que aporten al valor final del infrarrojo sea muy próximo a 0. Para esto, en el script de V-REP se considerará que cuando la distancia leída por un sensor sea igual a 0, se transforme en 10^6. El valor final calculado a partir de las distancias será el sumatorio de los valores aportados por cada uno de los sensores de proximidad que conforman el sensor infrarrojo, como se muestra en la siguiente ecuación.

$$IR = \sum_{n=1}^{41} a_n * {d_n}^{b_n}.$$

Con las 63 combinaciones de distancias para los 41 sensores y los correspondientes valores medidos por el sensor infrarrojo se emplea el complemento *Solver* de Excel para minimizar el error cuadrático medio entre el valor real medido por el infrarrojo y el calculado con la fórmula. Para eso, las restricciones empleadas fueron que el coeficiente a_n fuese mayor que 0 y que el exponente b_n fuese menor que 0.

En la figura 63 se muestran las intensidades leídas por el sensor modelado para los exponentes y coeficientes optimizados frente a los valores leídos por el sensor real. La línea de tendencia muestra una recta muy próxima a x=y, indicando que los valores leídos por el sensor real y el simulado son prácticamente iguales.

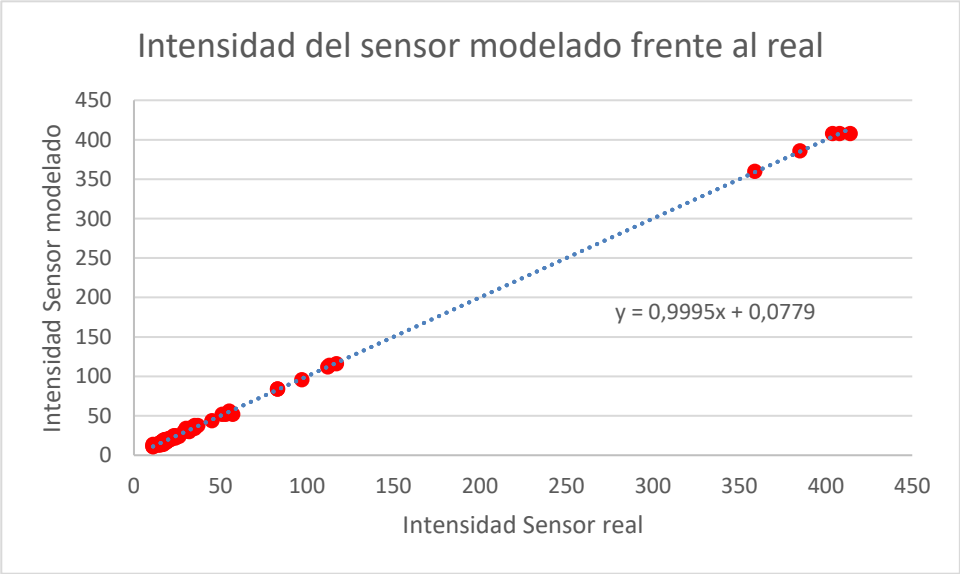

Figura 63: Valores de intensidad leídos por el sensor modelado frente al sensor real.

La tabla de resultados con los coeficientes y exponentes optimizados está recogida en el anexo 1. En esta tabla se muestran las distintas distancias medidas por los 41 sensores de proximidad para las 63 posiciones de la caja respecto al Robobo. A mayores, para cada sensor se indica el coeficiente y el exponente por el que hay que multiplicar y elevar a la distancia medida para calcular el valor de intensidad equivalente que aporta cada sensor. La tabla también incluye los valores reales del sensor infrarrojo y el valor calculado a través del polinomio para cada posición de la caja, el error entre ambos y el error cuadrático. Por último, se muestran el error cuadrático medio, el error medio y el error máximo.

7.6 Otras funciones de Robobo

A mayores de la programación del comportamiento de los motores de las ruedas, del pan y del tilt, y de los sensores de proximidad del Robobo, el modelo realizado en este trabajo también incluye otras funciones equivalentes a los comandos del Robobo real presentes en la librería robobo.py. Estos comandos son *readWheelPosition()*, *readPanPosition()*, *readTiltPosition(), readWheelSpeed()* y *resetWheelEncoders()*.

Las funciones *readWheelPosition* y *readWheelSpeed* en el modelo de V-REP se integrarán en una misma función llamada *readWheels*, que mide las posiciones y velocidades de ambas ruedas y las guarda en un vector como un valor entero. Como pasa con la función que se encarga de medir los sensores de proximidad, esta función no actuará en el modelo, sino que se emplea para obtener información del mismo. La programación del cálculo de los grados que se desplazó la rueda consiste en obtener la diferencia entre la posición inicial de una rueda (según el argumento que reciba en comando), en grados, y la posición que tiene en el momento de la llamada a la función. Para esto, la posición inicial de cada rueda se guardará en una variable global que se iniciará en el main script del modelo con valor 0. Este valor no cambiará a menos que se emplee la función *resetWheelEncoders.* Al llamar a esta función, se cambiará el valor de las variables globales en las que están guardados los valores de las posiciones de las ruedas derecha e izquierda, y se substituirán por los valores de posición en el momento de la llamada. De esta forma, los nuevos valores de incremento de posición de las ruedas empezarán desde cero. Para obtener la velocidad de las ruedas simplemente se leerá la variable global en la que se guarda la velocidad de cada una, como se explicó en el apartado 7.2.1.

Por otro lado, las funciones *readPanPosition* y *readTiltPosition* simplemente leen del modelo la posición actual del pan y del tilt y hacen su conversión a grados, devolviendo un valor entero.

En el anexo 2 de este trabajo se incluyen las funciones descritas en este apartado, en el script donde se definen todas las funciones explicadas en el apartado 7.

8 CONEXIÓN DE LA LIBRERÍA DE PYTHON CON V-REP.

La conexión de la librería Python con el simulador V-REP se muestra en la figura 64.

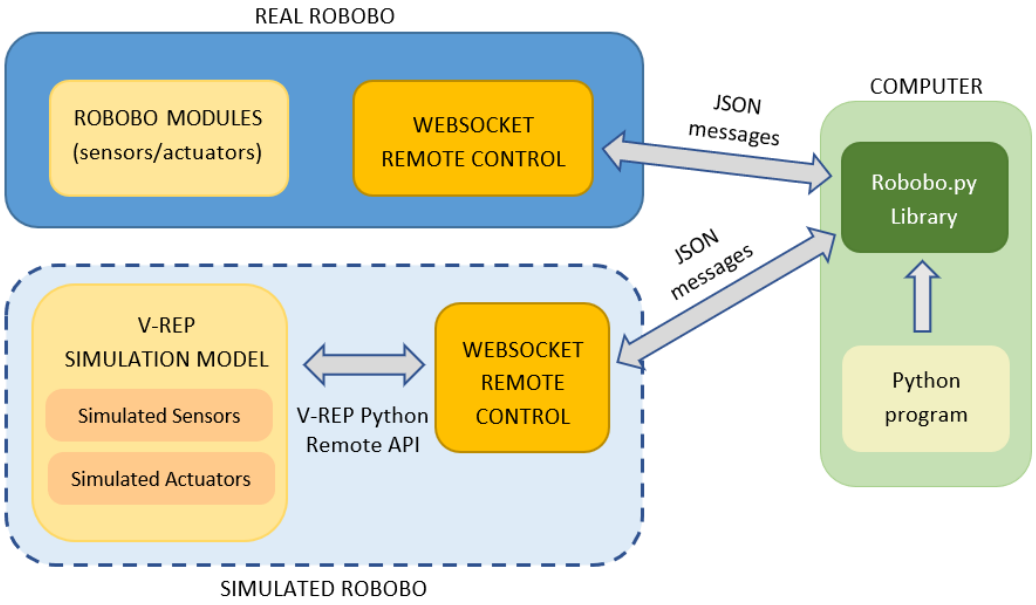

Figura 64: Esquema de funcionamiento de la conexión de la librería Robobo.py con V.REP mediante servidor Websocket.

En el esquema se observa que el Robobo real cuenta con un servidor Websocket, que está implementado en el smartphone. Este servidor recibe comandos en formato JSON, usados en el programa de Python diseñado para controlar los actuadores y sensores del robot. Al mismo tiempo, el servidor también envía los estados de los sensores del robot al ordenador, en formato JSON, para que puedan ser leídos por el programa. De esta forma, la librería Robobo.py actúa como cliente del servidor Websocket.

Para que el Robobo simulado en V-REP funcione de la misma forma que el real se implementa un servidor Websocket que permita recibir los comandos de la librería Robobo.py y enviar estados de los sensores del modelo, para que puedan ser leídos por el programa que controla el comportamiento del Robobo simulado. A mayores, hace falta una conexión entre los comandos que recibe el servidor Websocket y los comandos de la API remota de Python de V-REP que ejecuten las funciones que están implementadas en el modelo de simulación, definidas en el apartado 7. De esta forma se controla los actuadores y se leen los sensores implementados en el modelo.

El servidor Websocket y la conexión con la API externa de V-REP fueron desarrollados por el personal del GII para la realización de este proyecto. La validación de su implementación se llevará a cabo en la prueba descrita en el apartado 9, en donde se desarrollará un programa en Python que permita controlar el robot real y el simulado, empleando los comandos de la librería Robobo.py.

9 VALIDACIÓN

Para validar el modelo se realizará una prueba en la que estén implicados los motores y los sensores infrarrojos, así como el pan y el tilt, y se compararán con los sistemas del robot real.

La prueba consistirá en esquivar una caja situada a 40 cm del Robobo. Para esto se creará un programa que use los comandos de la librería Robobo de Python, y que hará que el robot avance con velocidad 15 hacia la caja en línea recta hasta que el sensor detecte un valor superior a un umbral, por ejemplo 100. Llegado a este punto, el robot debería parar moverse hacia atrás 2,5 segundos, girar mover rueda derecha durante 2 segundos y repetir el proceso hasta esquivar completamente la caja. En las figuras 65 y 66 se muestra la posición de partida de la caja en el caso real y en el simulador.

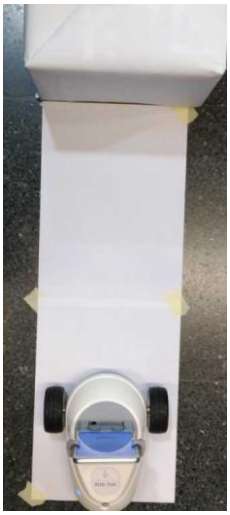

Figura 65: Posición de partida del robot real.

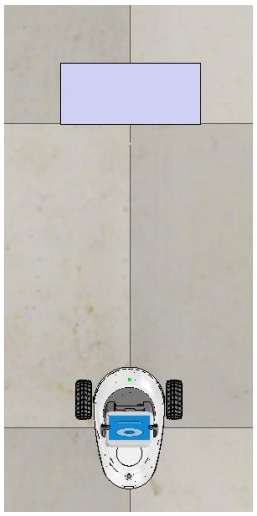

Figura 66: Posición de partida del robot simulado.

Para comprobar el buen funcionamiento del pan y del tilt, antes de que el robot empiece a moverse se rotará el pan hasta la posición de 90 y -90 grados antes de volver a la posición inicial, y el tilt hasta 5 grados y volverá a la posición de 90. En el momento en el que el pan y el tilt alcancen las posiciones que se le indican en el programa se medirán sus posiciones.

Cuando el Robobo se detenga al detectar la caja, se medirán los encoders de las ruedas, para saber cuántos grados han girado las ruedas.

Tras realizar la prueba descrita anteriormente, los resultados, tanto para el Robobo real como para el Robobo simulado en V-REP se muestran en la tabla 12:

Resultados	Robobo real	Robobo V-REP
Posición del pan (1)	88	88
Posición del pan (2)	-89	90
Posición del pan (3)	-2	-1
Posición del tilt (1)	7	7

Posición del tilt (2)	89	89
Valor del sensor IR frontal	107	126
Lectura encoder rueda derecha	630	573

Tabla 12: Resultados de las pruebas de validación.

Las posiciones finales tanto del robot real como el del modelo de V-REP se muestran en las figuras 67 y 68. Si se ejecutase la función en bucle, el robot avanzaría esquivando la caja hasta encontrarse con otro obstáculo, en cuyo caso repetiría el mismo procedimiento.

Figura 67: Posición final del Robobo real.

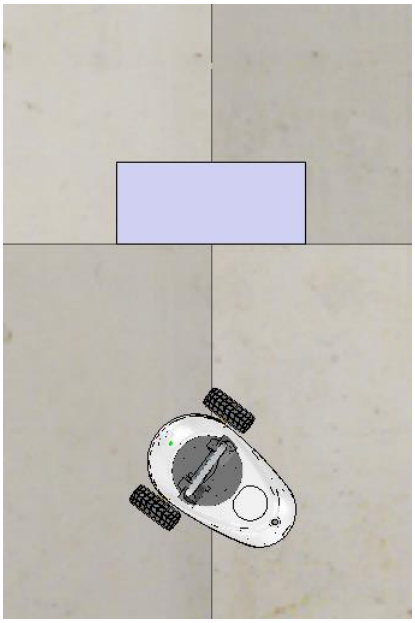

Figura 68: Posición final del Robobo simulado.

Comparando las pruebas realizadas en los dos modelos, se observa que los comportamientos son muy similares. La principal diferencia es la lectura del sensor infrarrojo frontal, que es el encargado de detectar la caja. Partiendo de que la caracterización de los sensores infrarrojos se realizó con el mismo material que con el que se hace esta prueba, la diferencia de resultados puede deberse a varios motivos como la luz ambiente, el error de lectura de los sensores infrarrojos reales, el ruido introducido en el cálculo de intensidad del sensor infrarrojo en el modelo de V-REP, o a que la caracterización no es capaz de ajustarse totalmente al comportamiento del sensor infrarrojo real. A mayores, el valor leído de los encoders de las ruedas muestra que el robot real recorrió más distancia que el simulado, en concreto, 57 grados, es decir, 3 cm más. Esta situación se debe a que el Robobo real ha necesitado acercarse más a la caja para que la intensidad del sensor infrarrojo superase el umbral de 100.

A parte de los infrarrojos, los motores se comportaron de una manera muy similar a los motores del Robobo real. Salvando las diferencias de distancia recorrida, explicada por una diferencia de medida entre el sensor del robot real y el del simulado. Los motores del pan y del tilt también funcionan bien ante los comandos recibidos, y por los resultados se observa que alcanzan la posición objetivo con precisión. Esto implica también que las funciones de lectura de las posiciones de los motores están bien implementadas.

10 CONCLUSIONES

En este trabajo se han implementado modelos de los motores de las ruedas, del pan y el tilt, que comportan de manera similar a los del robot real. A partir de las pruebas realizadas para la caracterización del pan, el tilt y las ruedas, se obtuvieron polinomios que permiten obtener la posición de cada *joint* ajustándose muy bien a los casos reales, y permitiendo modelar los motores para que se moviesen a una velocidad similar a los motores físicos. Además, los encoder se han incluido en el modelo permitiendo conocer las posiciones de los motores.

Por otro lado, los sensores infrarrojos desarrollados han mejorado los modelos anteriores en cuanto precisión de detección de objetos, permitiendo no solo obtener un valor de intensidad en función de la distancia entre el objeto y el sensor, sino también teniendo en cuenta el área del objeto que es detectada por el sensor. La dificultad de modelar los sensores infrarrojos se basaba en su propio funcionamiento, ya que la idea de encontrar un valor de intensidad de luz equivalente a una distancia, que es lo que miden los sensores del simulador, es muy compleja.

A mayores, para mejorar el modelo del smartphone se ha integrado una cámara que simule la cámara del teléfono móvil real.

Todos estos elementos, los motores, los sensores infrarrojos y la cámara se han integrado en un modelo 3D de Robobo que funciona conjuntamente, como se demuestra en la prueba de validación.

Por último, se ha conseguido controlar el modelo de Robobo en V-REP a partir de un programa de Python que emplee los comandos de la librería Robobo.py, de forma que se puede usar un mismo programa para controlar tanto el Robobo real como el simulado. Esto tiene una gran importancia ya que permitiría aprender a programar el Robobo a partir de un simulador, y comprender su funcionamiento, lo que será de gran ayuda tanto en el ámbito de la investigación como de la educación.

Como conclusión, y partiendo de la prueba de validación explicada en el apartado 9 de este trabajo, se puede concluir que se cumple con el objetivo principal de este trabajo que es el desarrollo de un modelo en simulación 3D en V-REP de Robobo con un comportamiento y unas respuestas idénticas al modelo real y que se pudiese controlar con la librería Robobo.py, de la misma forma que el robot real.

11 ANEXOS

11.1 ANEXO 1: Tablas de resultados

Tabla de resultados de la caracterización de los joints de las ruedas

Valor de los coeficientes optimizados							
C	**D**	**E**	**F**	**G**	**H**	**I**	**J**
-4.625E-05	5.219E-03	6.357E+00	5.137E+01	-3.253E-04	4.285E-02	-2.064E+00	-1.770E+01

Velocidad (%)	**Tiempo (s)**	**Distancia Recorrida (cm)**	**Grados Recorridos (˚)**	**Ajuste con el polinomio (˚)**	**Error (˚)**	**Error (cm)**	**Error Cuadrático (˚²)**	**Valor Absoluto del error (˚)**
2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2.00	0.50	0.00	0.00	10.39	10.39	0.59	108.04	10.39
2.00	1.00	3.00	53.30	42.44	-10.85	-0.61	117.80	10.85
2.00	1.50	4.60	81.72	74.50	-7.23	-0.41	52.25	7.23
2.00	2.00	6.10	108.37	106.55	-1.83	-0.10	3.34	1.83
2.00	3.00	10.20	181.21	170.65	-10.57	-0.59	111.66	10.57
2.00	6.00	20.70	367.76	362.95	-4.81	-0.27	23.11	4.81
2.00	8.00	27.70	492.12	491.15	-0.97	-0.05	0.94	0.97
4.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4.00	0.50	0.00	0.00	13.15	13.15	0.74	172.90	13.15
4.00	1.00	3.30	58.63	51.59	-7.04	-0.40	49.58	7.04
4.00	1.50	5.30	94.16	90.02	-4.14	-0.23	17.10	4.14
4.00	2.00	7.00	124.36	128.46	4.10	0.23	16.81	4.10
4.00	3.00	12.50	222.08	205.34	-16.74	-0.94	280.18	16.74
4.00	6.00	24.70	438.82	435.96	-2.86	-0.16	8.17	2.86
4.00	8.00	33.90	602.27	589.72	-12.56	-0.71	157.65	12.56
6.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6.00	0.50	0.80	14.21	16.23	2.02	0.11	4.09	2.02
6.00	1.00	3.30	58.63	61.08	2.45	0.14	6.00	2.45
6.00	1.50	6.30	111.93	105.92	-6.01	-0.34	36.06	6.01
6.00	2.00	7.80	138.58	150.76	12.19	0.69	148.57	12.19
6.00	3.00	13.50	239.84	240.45	0.61	0.03	0.37	0.61
6.00	6.00	28.30	502.78	509.51	6.73	0.38	45.32	6.73
6.00	8.00	38.60	685.77	688.89	3.11	0.18	9.70	3.11
10.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10.00	0.50	1.00	17.77	23.33	5.56	0.31	30.94	5.56

10.00	1.00	4.10	72.84	81.03	8.19	0.46	67.14	8.19
10.00	1.50	7.30	129.69	138.74	9.05	0.51	81.88	9.05
10.00	2.00	10.80	191.87	196.45	4.57	0.26	20.92	4.57
10.00	3.00	16.80	298.47	311.86	13.39	0.75	179.30	13.39
10.00	6.00	36.30	644.91	658.10	13.19	0.74	173.98	13.19
10.00	8.00	48.10	854.55	888.93	34.38	1.93	1181.73	34.38
15.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
15.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00
15.00	0.50	1.80	31.98	33.76	1.78	0.10	3.15	1.78
15.00	1.00	6.10	108.37	107.63	-0.75	-0.04	0.56	0.75
15.00	1.50	10.00	177.66	181.50	3.83	0.22	14.71	3.83
15.00	2.00	13.80	245.17	255.37	10.19	0.57	103.92	10.19
15.00	3.00	22.60	401.51	403.11	1.59	0.09	2.54	1.59
15.00	6.00	47.70	847.44	846.33	-1.11	-0.06	1.24	1.11
15.00	8.00	64.50	1145.92	1141.81	-4.10	-0.23	16.82	4.10
20.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20.00	0.25	0.60	10.66	0.61	-10.05	-0.57	100.91	10.05
20.00	0.50	3.10	55.08	45.67	-9.40	-0.53	88.44	9.40
20.00	1.00	8.30	147.46	135.78	-11.68	-0.66	136.31	11.68
20.00	1.50	13.10	232.74	225.90	-6.84	-0.38	46.78	6.84
20.00	2.00	18.10	321.57	316.01	-5.56	-0.31	30.88	5.56
20.00	3.00	28.80	511.66	496.24	-15.43	-0.87	238.05	15.43
20.00	6.00	59.20	1051.76	1036.91	-14.84	-0.84	220.26	14.84
20.00	8.00	79.60	1414.18	1397.37	-16.82	-0.95	282.86	16.82
30.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30.00	0.25	0.60	10.66	11.54	0.88	0.05	0.78	0.88
30.00	0.50	4.60	81.72	72.92	-8.80	-0.50	77.46	8.80
30.00	1.00	11.50	204.31	195.69	-8.62	-0.49	74.36	8.62
30.00	1.50	18.20	323.34	318.45	-4.89	-0.28	23.94	4.89
30.00	2.00	24.30	431.72	441.21	9.50	0.53	90.21	9.50
30.00	3.00	38.30	680.44	686.74	6.30	0.35	39.68	6.30
30.00	6.00	80.20	1424.84	1423.33	-1.52	-0.09	2.31	1.52
30.00	8.00	108.00	1918.74	1914.38	-4.36	-0.25	19.02	4.36
40.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40.00	0.25	1.40	24.87	25.23	0.36	0.02	0.13	0.36
40.00	0.50	5.20	92.38	102.99	10.61	0.60	112.59	10.61
40.00	1.00	14.10	250.50	258.52	8.01	0.45	64.20	8.01
40.00	1.50	22.80	405.07	414.04	8.97	0.50	80.42	8.97
40.00	2.00	31.40	557.86	569.56	11.70	0.66	136.87	11.70
40.00	3.00	49.50	879.42	880.60	1.17	0.07	1.38	1.17
40.00	6.00	102.50	1821.03	1813.72	-7.31	-0.41	53.44	7.31
40.00	8.00	136.50	2425.08	2435.80	10.72	0.60	114.98	10.72
60.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
60.00	0.25	2.60	46.19	52.83	6.64	0.37	44.12	6.64
60.00	0.50	9.20	163.45	163.23	-0.22	-0.01	0.05	0.22
60.00	1.00	21.70	385.53	384.03	-1.50	-0.08	2.24	1.50

60.00	1.50	35.10	623.59	604.82	-18.77	-1.06	352.24	18.77
60.00	2.00	45.90	815.47	825.62	10.15	0.57	103.09	10.15
60.00	3.00	72.00	1279.16	1267.21	-11.95	-0.67	142.85	11.95
60.00	6.00	146.10	2595.63	2591.98	-3.65	-0.21	13.32	3.65
60.00	8.00	195.20	3467.95	3475.16	7.22	0.41	52.06	7.22
90.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
90.00	0.25	2.70	47.97	64.45	16.48	0.93	271.57	16.48
90.00	0.50	12.90	229.18	222.46	-6.72	-0.38	45.15	6.72
90.00	1.00	30.50	541.87	538.50	-3.37	-0.19	11.36	3.37
90.00	1.50	48.70	865.21	854.53	-10.68	-0.60	114.11	10.68
90.00	2.00	66.00	1172.56	1170.56	-2.00	-0.11	4.02	2.00
90.00	3.00	101.00	1794.38	1802.62	8.25	0.46	67.98	8.25
90.00	6.00	208.30	3700.69	3698.82	-1.87	-0.11	3.49	1.87
90.00	8.00	279.30	4962.08	4962.95	0.87	0.05	0.75	0.87

Error Cuadrático Medio (°²)	**Error Medio (°)**	**Error máximo (°)**	**Error máximo (cm)**
73.18	8.55	34.38	1.93

Tabla de resultados de la caracterización de los joints del pan

Valor de los coeficientes optimizados							
C	**D**	**E**	**F**	**G**	**H**	**I**	**J**
-3.060E-05	3.877E-03	8.475E-01	8.055E+00	-1.990E-05	1.064E-03	-3.303E-01	-8.907E-01

Velocidad	**Posicion (˚)**	**Tiempo1 (s)**	**Tiempo2 (s)**	**Tiempo3 (s)**	**Tiempo medio (s)**	**Ajuste con el polinomio (s)**	**Error (s)**	**Error Cuadrático (s^2)**
2.00	15.00	1.75	1.67	1.72	1.71	1.69	-0.02	0.00
2.00	30.00	3.34	3.12	3.50	3.32	3.23	-0.09	0.01
2.00	45.00	4.61	4.50	4.53	4.55	4.77	0.22	0.05
2.00	60.00	6.28	6.22	6.72	6.41	6.30	-0.10	0.01
2.00	75.00	7.79	7.83	7.83	7.82	7.84	0.02	0.00
2.00	90.00	9.28	9.23	9.65	9.39	9.38	-0.01	0.00
2.00	110.00	11.81	11.99	11.65	11.82	11.42	-0.39	0.15
2.00	135.00	14.00	13.89	13.93	13.94	13.98	0.04	0.00
2.00	160.00	16.40	16.56	16.52	16.49	16.54	0.05	0.00
4.00	15.00	1.45	1.60	1.49	1.51	1.49	-0.02	0.00
4.00	30.00	2.73	3.07	2.67	2.82	2.80	-0.02	0.00
4.00	45.00	3.91	4.11	3.90	3.97	4.10	0.13	0.02
4.00	60.00	5.26	5.26	5.46	5.33	5.41	0.08	0.01
4.00	75.00	6.63	6.57	6.58	6.59	6.71	0.12	0.01
4.00	90.00	7.90	7.92	7.84	7.89	8.01	0.13	0.02
4.00	110.00	9.63	9.54	9.51	9.56	9.75	0.19	0.04
4.00	135.00	12.37	11.96	12.27	12.20	11.93	-0.27	0.08
4.00	160.00	13.83	13.81	13.88	13.84	14.10	0.26	0.07
6.00	15.00	1.48	1.18	1.58	1.41	1.34	-0.07	0.00
6.00	30.00	2.31	2.40	2.33	2.35	2.47	0.13	0.02
6.00	45.00	3.91	3.34	3.47	3.57	3.60	0.03	0.00
6.00	60.00	4.71	4.61	4.53	4.62	4.73	0.12	0.01
6.00	75.00	6.10	5.92	6.21	6.08	5.86	-0.21	0.04
6.00	90.00	7.18	7.00	6.82	7.00	6.99	-0.01	0.00
6.00	110.00	8.69	8.67	8.38	8.58	8.50	-0.08	0.01
6.00	135.00	10.09	10.24	10.04	10.12	10.39	0.26	0.07
6.00	160.00	12.58	12.26	12.16	12.33	12.27	-0.06	0.00
10.00	15.00	1.17	1.16	1.35	1.23	1.13	-0.10	0.01
10.00	30.00	2.01	1.99	1.99	2.00	2.02	0.02	0.00
10.00	45.00	2.80	2.92	2.81	2.84	2.91	0.06	0.00
10.00	60.00	3.80	3.80	3.80	3.80	3.80	0.00	0.00
10.00	75.00	4.91	4.76	4.64	4.77	4.68	-0.09	0.01
10.00	90.00	5.97	5.72	6.17	5.95	5.57	-0.38	0.14
10.00	110.00	7.32	6.82	7.12	7.09	6.76	-0.33	0.11
10.00	135.00	8.22	8.22	8.15	8.20	8.24	0.04	0.00
10.00	160.00	9.78	9.61	9.66	9.68	9.72	0.03	0.00
20.00	15.00	0.86	0.83	0.88	0.86	0.84	-0.01	0.00
20.00	30.00	1.47	1.30	1.36	1.38	1.42	0.04	0.00

20.00	45.00	1.94	1.84	2.05	1.94	1.99	0.04	0.00
20.00	60.00	2.44	2.43	2.45	2.44	2.56	0.12	0.01
20.00	75.00	3.38	3.14	2.96	3.16	3.13	-0.03	0.00
20.00	90.00	3.48	3.57	3.47	3.51	3.70	0.19	0.04
20.00	110.00	4.82	4.41	4.34	4.52	4.46	-0.07	0.00
20.00	135.00	5.22	5.31	5.86	5.46	5.41	-0.06	0.00
20.00	160.00	6.12	6.66	6.22	6.33	6.36	0.02	0.00
30.00	15.00	0.77	0.65	0.72	0.71	0.70	-0.01	0.00
30.00	30.00	1.10	1.07	1.03	1.07	1.12	0.05	0.00
30.00	45.00	1.52	1.52	1.52	1.52	1.53	0.01	0.00
30.00	60.00	2.03	1.88	1.84	1.92	1.95	0.03	0.00
30.00	75.00	2.34	2.35	2.31	2.33	2.36	0.03	0.00
30.00	90.00	2.70	3.03	2.72	2.82	2.78	-0.04	0.00
30.00	110.00	3.38	3.26	3.19	3.28	3.33	0.05	0.00
30.00	135.00	4.00	3.79	4.11	3.97	4.02	0.06	0.00
30.00	160.00	4.55	4.66	4.56	4.59	4.71	0.12	0.02
40.00	15.00	0.63	0.64	0.66	0.64	0.62	-0.02	0.00
40.00	30.00	0.84	0.94	0.97	0.92	0.95	0.03	0.00
40.00	45.00	1.30	1.28	1.29	1.29	1.27	-0.02	0.00
40.00	60.00	1.61	1.61	1.59	1.60	1.59	-0.01	0.00
40.00	75.00	1.92	1.88	2.00	1.93	1.92	-0.01	0.00
40.00	90.00	2.25	2.26	2.16	2.22	2.24	0.02	0.00
40.00	110.00	2.73	2.75	2.81	2.76	2.68	-0.09	0.01
40.00	135.00	3.21	3.14	3.22	3.19	3.22	0.03	0.00
40.00	160.00	3.70	4.12	3.69	3.84	3.76	-0.08	0.01
60.00	15.00	0.54	0.54	0.62	0.57	0.55	-0.02	0.00
60.00	30.00	0.77	0.83	0.87	0.82	0.77	-0.05	0.00
60.00	45.00	0.96	0.97	1.19	1.04	1.00	-0.04	0.00
60.00	60.00	1.21	1.23	1.18	1.21	1.23	0.02	0.00
60.00	75.00	1.49	1.51	1.50	1.50	1.45	-0.05	0.00
60.00	90.00	1.70	1.63	1.68	1.67	1.68	0.01	0.00
60.00	110.00	1.91	1.99	1.91	1.94	1.98	0.04	0.00
60.00	135.00	2.31	2.49	2.37	2.39	2.36	-0.03	0.00
60.00	160.00	2.77	2.76	2.75	2.76	2.73	-0.03	0.00
80.00	15.00	0.52	0.52	0.51	0.52	0.54	0.02	0.00
80.00	30.00	0.80	0.70	0.70	0.73	0.71	-0.02	0.00
80.00	45.00	0.87	0.90	0.87	0.88	0.89	0.01	0.00
80.00	60.00	0.98	1.05	1.06	1.03	1.07	0.04	0.00
80.00	75.00	1.21	1.21	1.21	1.21	1.24	0.03	0.00
80.00	90.00	1.38	1.42	1.44	1.41	1.42	0.01	0.00
80.00	110.00	1.71	1.69	1.69	1.70	1.66	-0.04	0.00
80.00	135.00	1.97	1.85	1.93	1.92	1.95	0.03	0.00
80.00	160.00	2.25	2.21	2.30	2.25	2.24	-0.01	0.00

Error Cuadrático Medio (s^2)	Error Medio (s)	Error máximo (s)
0.01	0.11	0.39

Tabla de resultados de la caracterización de los joints del tilt

Valor de los coeficientes optimizados							
C	**D**	**E**	**F**	**G**	**H**	**I**	**J**
1.409E-05	-2.598E-03	4.809E-01	3.182E+00	-8.840E-05	1.233E-02	-5.495E-01	4.738E+00

Velocidad	**Ángulo recorrido (˚)**	**Tiempo1 (s)**	**Tiempo2 (s)**	**Tiempo3 (s)**	**Tiempo medio (s)**	**Ajuste con el polinomio (s)**	**Error (s)**	**Error Cuadrático (s)**
2.00	15.00	2.66	2.77	2.74	2.72	2.74	0.01	0.00
2.00	30.00	6.57	6.75	6.53	6.62	6.37	-0.25	0.06
2.00	45.00	9.88	9.83	9.89	9.87	10.00	0.13	0.02
2.00	60.00	13.44	13.61	13.42	13.49	13.62	0.13	0.02
2.00	75.00	17.52	17.56	17.70	17.59	17.25	-0.34	0.12
2.00	90.00	20.57	20.50	20.93	20.67	20.88	0.22	0.05
2.00	100.00	23.40	23.64	23.17	23.40	23.30	-0.10	0.01
4.00	15.00	2.45	2.37	2.44	2.42	2.42	0.00	0.00
4.00	30.00	5.69	5.89	5.83	5.80	5.38	-0.42	0.18
4.00	45.00	8.17	8.18	8.20	8.18	8.35	0.16	0.03
4.00	60.00	11.17	11.18	11.90	11.42	11.31	-0.11	0.01
4.00	75.00	14.14	14.03	14.16	14.11	14.27	0.16	0.03
4.00	90.00	17.81	17.83	17.79	17.81	17.23	-0.58	0.34
4.00	100.00	18.97	19.05	18.92	18.98	19.21	0.23	0.05
6.00	15.00	2.05	2.26	2.14	2.15	2.20	0.05	0.00
6.00	30.00	4.51	4.50	4.50	4.50	4.71	0.20	0.04
6.00	45.00	7.04	6.98	7.07	7.03	7.22	0.19	0.04
6.00	60.00	9.63	9.44	9.53	9.53	9.73	0.19	0.04
6.00	75.00	12.29	12.31	12.13	12.24	12.24	-0.01	0.00
6.00	90.00	14.46	14.45	14.39	14.43	14.75	0.31	0.10
6.00	100.00	16.19	16.09	16.09	16.12	16.42	0.30	0.09
10.00	15.00	1.74	1.63	1.74	1.70	1.89	0.18	0.03
10.00	30.00	4.09	3.89	4.29	4.09	3.82	-0.27	0.07
10.00	45.00	5.73	5.93	5.74	5.80	5.76	-0.04	0.00
10.00	60.00	7.78	7.59	7.57	7.65	7.70	0.05	0.00
10.00	75.00	9.53	9.43	9.60	9.52	9.63	0.11	0.01
10.00	90.00	12.01	11.73	11.76	11.83	11.57	-0.26	0.07
10.00	100.00	12.80	12.97	13.27	13.01	12.86	-0.15	0.02
20.00	15.00	1.23	1.27	1.14	1.21	1.43	0.22	0.05
20.00	30.00	2.64	2.55	2.73	2.64	2.70	0.06	0.00
20.00	45.00	4.06	3.80	3.88	3.91	3.96	0.05	0.00
20.00	60.00	5.23	5.24	5.16	5.21	5.22	0.01	0.00
20.00	75.00	6.52	6.55	6.54	6.54	6.49	-0.05	0.00
20.00	90.00	7.81	7.75	7.91	7.82	7.75	-0.07	0.01
20.00	100.00	8.61	8.72	8.62	8.65	8.59	-0.06	0.00
30.00	15.00	1.17	1.07	1.04	1.09	1.15	0.06	0.00
30.00	30.00	2.07	2.26	2.17	2.17	2.11	-0.06	0.00
30.00	45.00	3.28	3.04	3.06	3.13	3.07	-0.06	0.00

30.00	60.00	4.10	3.95	3.99	4.01	4.03	0.01	0.00
30.00	75.00	5.04	4.88	4.99	4.97	4.99	0.02	0.00
30.00	90.00	5.86	6.16	6.23	6.08	5.94	-0.14	0.02
30.00	100.00	6.56	6.59	6.49	6.55	6.58	0.04	0.00
40.00	15.00	0.95	0.97	0.95	0.96	0.95	-0.01	0.00
40.00	30.00	1.61	1.72	1.64	1.66	1.73	0.07	0.01
40.00	45.00	2.51	2.41	2.51	2.48	2.51	0.04	0.00
40.00	60.00	3.31	3.31	3.21	3.28	3.30	0.02	0.00
40.00	75.00	4.02	4.13	4.11	4.09	4.08	-0.01	0.00
40.00	90.00	4.82	4.82	4.91	4.85	4.86	0.01	0.00
40.00	100.00	5.62	5.32	5.63	5.52	5.38	-0.14	0.02
60.00	15.00	0.80	0.80	0.80	0.80	0.70	-0.10	0.01
60.00	30.00	1.42	1.40	1.41	1.41	1.28	-0.13	0.02
60.00	45.00	1.91	1.91	1.91	1.91	1.86	-0.05	0.00
60.00	60.00	2.41	2.51	2.51	2.48	2.45	-0.03	0.00
60.00	75.00	3.02	2.91	3.01	2.98	3.03	0.05	0.00
60.00	90.00	3.62	3.82	3.61	3.68	3.61	-0.07	0.00
60.00	100.00	4.12	4.02	4.12	4.09	4.00	-0.08	0.01
80.00	15.00	0.70	0.60	0.70	0.67	0.64	-0.03	0.00
80.00	30.00	1.20	1.31	1.11	1.21	1.10	-0.10	0.01
80.00	45.00	1.61	1.51	1.51	1.54	1.57	0.03	0.00
80.00	60.00	2.01	2.11	2.01	2.04	2.03	-0.01	0.00
80.00	75.00	2.31	2.41	2.51	2.41	2.50	0.09	0.01
80.00	90.00	2.91	2.81	2.81	2.84	2.97	0.12	0.01
80.00	100.00	3.12	3.11	3.11	3.11	3.28	0.16	0.03

Error Cuadrático Medio (s^2)	**Error Medio (s)**	**Error máximo (s)**
0.03	0.16	0.58

Tabla de resultados de la caracterización de los sensores de proximidad

	1	2	3	4	5	6	7	8
C	5.00	10.00	15.00	20.00	25.00	30.00	35.00	5.00
C1	6.10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C2	5.71	11.41	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C3	5.19	10.37	15.56	20.74	1000000.00	1000000.00	1000000.00	5.19
C4	5.08	10.17	15.25	20.33	25.42	30.50	1000000.00	5.08
C5	5.02	10.03	15.05	20.06	25.08	30.09	35.11	5.02
C6	5.01	10.02	15.03	20.04	25.05	30.06	35.07	5.01
C7	5.06	10.11	15.17	20.22	25.28	30.34	1000000.00	5.06
C8	5.13	10.25	15.38	20.50	1000000.00	1000000.00	1000000.00	5.13
C9	5.48	10.97	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	5.48
C10	6.10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	6.10
C11	6.10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	6.10
C12	5.71	11.41	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	5.71
C13	5.19	10.37	15.56	20.74	1000000.00	1000000.00	1000000.00	5.19
C14	5.08	10.17	15.25	20.33	25.42	30.50	1000000.00	5.08
C15	5.02	10.03	15.05	20.06	25.08	30.09	35.11	5.02
C16	5.01	10.02	15.03	20.04	25.05	30.06	35.07	5.01
C17	5.06	10.11	15.17	20.22	21.71	21.71	21.71	5.06
C18	5.13	10.25	14.67	14.67	14.67	14.67	14.67	5.13
C19	5.48	7.85	7.85	7.85	7.85	7.85	7.85	5.48
C20	5.62	5.62	5.62	5.62	5.62	5.62	5.62	5.62
C21	6.10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C22	5.71	11.41	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C23	5.19	10.37	15.56	20.74	1000000.00	1000000.00	1000000.00	5.19
C24	5.08	10.17	15.25	20.33	25.42	30.50	1000000.00	5.08
C25	5.02	10.03	15.05	20.06	25.08	30.09	35.11	5.02
C26	5.01	10.02	15.03	20.04	25.05	30.06	35.07	5.01
C27	5.06	10.11	15.17	20.22	25.28	30.34	1000000.00	5.06
C28	5.13	10.25	15.38	20.50	1000000.00	1000000.00	1000000.00	5.13
C29	5.48	10.97	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	5.48
C30	6.10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	6.10
C31	6.10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	6.10
C32	5.48	10.97	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	5.48
C33	5.13	10.25	15.38	20.50	1000000.00	1000000.00	1000000.00	5.13
C34	5.06	10.11	15.17	20.22	25.28	30.34	1000000.00	5.06
C35	5.01	10.02	15.03	20.04	25.05	30.06	35.07	5.01
C36	5.02	10.03	15.05	20.06	25.08	30.09	35.11	5.02
C37	5.08	10.17	15.25	20.33	25.25	25.25	25.25	5.08
C38	5.19	10.37	15.56	17.17	17.17	17.17	17.17	5.19
C39	5.71	9.45	9.45	9.45	9.45	9.45	9.45	9.45
C40	6.10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00

	9	10	11	12	13	14	15	16
C	10.00	15.00	20.00	25.00	30.00	35.00	5.00	10.00
C1	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	6.10	1000000.00
C2	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	5.71	1000000.00
C3	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	5.19	10.37
C4	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	5.08	10.17
C5	10.03	15.05	1000000.00	1000000.00	1000000.00	1000000.00	5.02	10.03
C6	10.02	15.03	20.04	25.05	30.06	35.07	5.01	10.02
C7	10.11	15.17	20.22	25.28	30.34	1000000.00	5.06	10.11
C8	10.25	15.38	20.50	1000000.00	1000000.00	1000000.00	5.13	10.25
C9	10.97	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	5.48	10.97
C10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	6.10	1000000.00
C11	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	6.10	1000000.00
C12	11.41	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	5.71	11.41
C13	10.37	15.56	20.74	1000000.00	1000000.00	1000000.00	5.19	10.37
C14	10.17	15.25	20.33	25.42	30.50	1000000.00	5.08	10.17
C15	10.03	15.05	20.06	25.08	30.09	35.11	5.02	10.03
C16	10.02	15.03	20.04	25.05	30.06	35.07	5.01	10.02
C17	10.11	15.17	20.22	21.71	21.71	21.71	5.06	10.11
C18	10.25	14.67	14.67	14.67	14.67	14.67	5.13	10.25
C19	7.85	7.85	7.85	7.85	7.85	7.85	5.48	7.85
C20	5.62	5.62	5.62	5.62	5.62	5.62	5.62	5.62
C21	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	6.10	1000000.00
C22	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	5.71	11.41
C23	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	5.19	10.37
C24	10.17	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	5.08	10.17
C25	10.03	15.05	20.06	1000000.00	1000000.00	1000000.00	5.02	10.03
C26	10.02	15.03	20.04	25.05	30.06	35.07	5.01	10.02
C27	10.11	15.17	20.22	25.28	30.34	1000000.00	5.06	10.11
C28	10.25	15.38	20.50	1000000.00	1000000.00	1000000.00	5.13	10.25
C29	10.97	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	5.48	10.97
C30	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	6.10	1000000.00
C31	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	6.10	1000000.00
C32	10.97	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	5.48	10.97
C33	10.25	15.38	20.50	1000000.00	1000000.00	1000000.00	5.13	10.25
C34	10.11	15.17	20.22	25.28	30.34	1000000.00	5.06	10.11
C35	10.02	15.03	20.04	25.05	30.06	35.07	5.01	10.02
C36	10.03	15.05	20.06	1000000.00	1000000.00	1000000.00	5.02	10.03
C37	10.17	25.25	25.25	25.25	25.25	25.25	5.08	10.17
C38	17.17	17.17	17.17	17.17	17.17	17.17	5.19	10.37
C39	9.45	9.45	9.45	9.45	9.45	9.45	5.71	9.45
C40	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	6.10	1000000.00

	17	18	19	20	21	22	23	24
C	15.00	20.00	25.00	30.00	35.00	1000000.00	1000000.00	1000000.00
C1	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C2	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C3	15.56	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C4	15.25	20.33	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C5	15.05	20.06	25.08	30.09	35.11	1000000.00	1000000.00	1000000.00
C6	15.03	20.04	25.05	30.06	35.07	1000000.00	17.54	17.54
C7	15.17	20.22	25.28	30.34	1000000.00	7.58	10.11	15.17
C8	15.38	20.50	1000000.00	1000000.00	1000000.00	5.13	10.25	15.38
C9	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	5.48	10.97	1000000.00
C10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	6.10	1000000.00	1000000.00
C11	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C12	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C13	15.56	20.74	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C14	15.25	20.33	25.42	30.50	1000000.00	1000000.00	1000000.00	1000000.00
C15	15.05	20.06	25.08	30.09	35.11	1000000.00	1000000.00	1000000.00
C16	15.03	20.04	25.05	30.06	35.07	1000000.00	1000000.00	1000000.00
C17	15.17	20.22	21.71	21.71	21.71	21.71	21.71	21.71
C18	14.67	14.67	14.67	14.67	14.67	14.67	14.67	14.67
C19	7.85	7.85	7.85	7.85	7.85	7.85	7.85	7.85
C20	5.62	5.62	5.62	5.62	5.62	5.62	5.62	5.62
C21	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C22	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C23	15.56	20.74	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C24	15.25	20.33	25.42	30.50	1000000.00	1000000.00	1000000.00	1000000.00
C25	15.05	20.06	25.08	30.09	35.11	1000000.00	1000000.00	1000000.00
C26	15.03	20.04	25.05	30.06	35.07	1000000.00	1000000.00	24.80
C27	15.17	20.22	25.28	30.34	1000000.00	10.73	10.73	15.17
C28	15.38	20.50	1000000.00	1000000.00	1000000.00	7.25	10.25	15.38
C29	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	5.48	10.97	1000000.00
C30	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	6.10	1000000.00	1000000.00
C31	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	6.10	1000000.00	1000000.00
C32	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	5.48	10.97	1000000.00
C33	15.38	20.50	1000000.00	1000000.00	1000000.00	7.25	10.25	15.38
C34	15.17	20.22	25.28	30.34	1000000.00	10.73	10.73	15.17
C35	15.03	20.04	25.05	30.06	35.07	1000000.00	1000000.00	24.80
C36	15.05	20.06	25.08	30.09	35.11	1000000.00	1000000.00	1000000.00
C37	15.25	20.33	25.25	25.25	25.25	25.25	25.25	25.25
C38	15.56	17.17	17.17	17.17	17.17	17.17	17.17	17.17
C39	9.45	9.45	9.45	9.45	9.45	9.45	9.45	9.45
C40	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00

	25	26	27	28	29	30	31	32
C	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C1	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C2	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C3	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C4	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C5	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C6	20.04	25.05	30.06	35.07	1000000.00	1000000.00	1000000.00	1000000.00
C7	20.22	25.28	30.34	1000000.00	1000000.00	1000000.00	22.75	22.75
C8	20.50	1000000.00	1000000.00	1000000.00	15.38	15.38	15.38	20.50
C9	1000000.00	1000000.00	1000000.00	1000000.00	8.22	10.97	1000000.00	1000000.00
C10	1000000.00	1000000.00	1000000.00	1000000.00	6.10	1000000.00	1000000.00	1000000.00
C11	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C12	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C13	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C14	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C15	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C16	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C17	21.71	21.71	21.71	21.71	21.71	21.71	21.71	21.71
C18	14.67	14.67	14.67	14.67	14.67	14.67	14.67	14.67
C19	7.85	7.85	7.85	7.85	7.85	7.85	7.85	7.85
C20	5.62	5.62	5.62	5.62	5.62	5.62	5.62	5.62
C21	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C22	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C23	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C24	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C25	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C26	24.80	25.05	30.06	35.07	1000000.00	1000000.00	1000000.00	1000000.00
C27	20.22	25.28	30.34	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C28	20.50	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C29	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C30	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C31	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C32	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C33	20.50	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C34	20.22	25.28	30.34	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C35	24.80	25.05	30.06	35.07	1000000.00	1000000.00	1000000.00	1000000.00
C36	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C37	25.25	25.25	25.25	25.25	25.25	25.25	25.25	25.25
C38	17.17	17.17	17.17	17.17	17.17	17.17	17.17	17.17
C39	9.45	9.45	9.45	9.45	9.45	9.45	9.45	9.45
C40	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00

	33	34	35	36	37	38	39	40
C	1000000.00	1000000.00	1000000.00	5.00	10.00	15.00	20.00	25.00
C1	1000000.00	1000000.00	1000000.00	6.10	1000000.00	1000000.00	1000000.00	1000000.00
C2	1000000.00	1000000.00	1000000.00	5.71	11.41	1000000.00	1000000.00	1000000.00
C3	1000000.00	1000000.00	1000000.00	5.19	10.37	15.56	20.74	1000000.00
C4	1000000.00	1000000.00	1000000.00	5.08	10.17	15.25	20.33	25.42
C5	1000000.00	1000000.00	1000000.00	5.02	10.03	15.05	20.06	25.08
C6	1000000.00	1000000.00	1000000.00	5.01	10.02	15.03	20.04	25.05
C7	25.28	30.34	1000000.00	5.06	10.11	15.17	20.22	1000000.00
C8	1000000.00	1000000.00	1000000.00	5.13	10.25	15.38	1000000.00	1000000.00
C9	1000000.00	1000000.00	1000000.00	5.48	1000000.00	1000000.00	1000000.00	1000000.00
C10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C11	1000000.00	1000000.00	1000000.00	6.10	1000000.00	1000000.00	1000000.00	1000000.00
C12	1000000.00	1000000.00	1000000.00	5.71	11.41	1000000.00	1000000.00	1000000.00
C13	1000000.00	1000000.00	1000000.00	5.19	10.37	15.56	20.74	1000000.00
C14	1000000.00	1000000.00	1000000.00	5.08	10.17	15.25	20.33	25.42
C15	1000000.00	1000000.00	1000000.00	5.02	10.03	15.05	20.06	25.08
C16	1000000.00	1000000.00	1000000.00	5.01	10.02	15.03	20.04	25.05
C17	21.71	21.71	21.71	5.06	10.11	15.17	20.22	21.71
C18	14.67	14.67	14.67	5.13	10.25	14.67	14.67	14.67
C19	7.85	7.85	7.85	5.48	7.85	7.85	7.85	7.85
C20	5.62	5.62	5.62	5.62	5.62	5.62	5.62	5.62
C21	1000000.00	1000000.00	1000000.00	6.10	1000000.00	1000000.00	1000000.00	1000000.00
C22	1000000.00	1000000.00	1000000.00	5.71	11.41	1000000.00	1000000.00	1000000.00
C23	1000000.00	1000000.00	1000000.00	5.19	10.37	15.56	20.74	1000000.00
C24	1000000.00	1000000.00	1000000.00	5.08	10.17	15.25	20.33	25.42
C25	1000000.00	1000000.00	1000000.00	5.02	10.03	15.05	20.06	25.08
C26	1000000.00	1000000.00	1000000.00	5.01	10.02	15.03	20.04	25.05
C27	1000000.00	1000000.00	1000000.00	5.06	10.11	15.17	20.22	25.28
C28	1000000.00	1000000.00	1000000.00	5.13	10.25	15.38	20.50	1000000.00
C29	1000000.00	1000000.00	1000000.00	5.48	10.97	1000000.00	1000000.00	1000000.00
C30	1000000.00	1000000.00	1000000.00	6.10	1000000.00	1000000.00	1000000.00	1000000.00
C31	1000000.00	1000000.00	1000000.00	6.10	1000000.00	1000000.00	1000000.00	1000000.00
C32	1000000.00	1000000.00	1000000.00	5.48	10.97	1000000.00	1000000.00	1000000.00
C33	1000000.00	1000000.00	1000000.00	5.13	10.25	15.38	20.50	1000000.00
C34	1000000.00	1000000.00	1000000.00	5.06	10.11	15.17	20.22	25.28
C35	1000000.00	1000000.00	1000000.00	5.01	10.02	15.03	20.04	25.05
C36	1000000.00	1000000.00	1000000.00	5.02	10.03	15.05	20.06	25.08
C37	25.25	25.25	25.25	5.08	10.17	15.25	20.33	25.25
C38	17.17	17.17	17.17	5.19	10.37	15.56	17.17	17.17
C39	9.45	9.45	9.45	5.71	9.45	9.45	9.45	9.45
C40	1000000.00	1000000.00	1000000.00	6.10	1000000.00	1000000.00	1000000.00	1000000.00

	41	42	43	44	45	46	47	48
C	30.00	35.00	5.00	10.00	15.00	20.00	25.00	30.00
C1	1000000.00	1000000.00	6.10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C2	1000000.00	1000000.00	5.71	11.41	1000000.00	1000000.00	1000000.00	1000000.00
C3	1000000.00	1000000.00	5.19	10.37	15.56	20.74	1000000.00	1000000.00
C4	30.50	1000000.00	5.08	10.17	15.25	20.33	25.42	30.50
C5	30.09	35.11	5.02	10.03	15.05	20.06	25.08	30.09
C6	30.06	35.07	5.01	10.02	15.03	1000000.00	1000000.00	1000000.00
C7	1000000.00	1000000.00	5.06	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C8	1000000.00	1000000.00	5.13	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C9	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C11	1000000.00	1000000.00	6.10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C12	1000000.00	1000000.00	5.71	11.41	1000000.00	1000000.00	1000000.00	1000000.00
C13	1000000.00	1000000.00	5.19	10.37	15.56	20.74	1000000.00	1000000.00
C14	30.50	1000000.00	5.08	10.17	15.25	20.33	25.42	30.50
C15	30.09	35.11	5.02	10.03	15.05	20.06	25.08	30.09
C16	30.06	35.07	5.01	10.02	15.03	20.04	25.05	30.06
C17	21.71	21.71	5.06	10.11	15.17	20.22	21.71	21.71
C18	14.67	14.67	5.13	10.25	14.67	14.67	14.67	14.67
C19	7.85	7.85	5.48	7.85	7.85	7.85	7.85	7.85
C20	5.62	5.62	5.62	5.62	5.62	5.62	5.62	5.62
C21	1000000.00	1000000.00	6.10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C22	1000000.00	1000000.00	5.71	11.41	1000000.00	1000000.00	1000000.00	1000000.00
C23	1000000.00	1000000.00	5.19	10.37	15.56	20.74	1000000.00	1000000.00
C24	30.50	1000000.00	5.08	10.17	15.25	20.33	25.42	30.50
C25	30.09	35.11	5.02	10.03	15.05	20.06	25.08	30.09
C26	30.06	35.07	5.01	10.02	15.03	20.04	1000000.00	1000000.00
C27	30.34	1000000.00	5.06	10.11	1000000.00	1000000.00	1000000.00	1000000.00
C28	1000000.00	1000000.00	5.13	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C29	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C30	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C31	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C32	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C33	1000000.00	1000000.00	5.13	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C34	30.34	1000000.00	5.06	10.11	1000000.00	1000000.00	1000000.00	1000000.00
C35	30.06	35.07	5.01	10.02	15.03	20.04	1000000.00	1000000.00
C36	30.09	35.11	5.02	10.03	15.05	20.06	25.08	30.09
C37	25.25	25.25	5.08	10.17	15.25	20.33	25.25	25.25
C38	17.17	17.17	5.19	10.37	15.56	17.17	17.17	17.17
C39	9.45	9.45	5.71	9.45	9.45	9.45	9.45	9.45
C40	1000000.00	1000000.00	6.10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00

	49	50	51	52	53	54	55	56
C	35.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C1	1000000.00	6.10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C2	1000000.00	5.71	11.41	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C3	1000000.00	5.19	10.37	15.56	20.74	1000000.00	1000000.00	1000000.00
C4	1000000.00	7.63	10.17	15.25	20.33	25.42	30.50	1000000.00
C5	35.11	1000000.00	17.55	17.55	20.06	25.08	30.09	35.11
C6	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C7	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C8	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C9	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C11	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C12	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C13	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C14	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C15	35.11	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C16	35.07	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C17	21.71	21.71	21.71	21.71	21.71	21.71	21.71	21.71
C18	14.67	14.67	14.67	14.67	14.67	14.67	14.67	14.67
C19	7.85	7.85	7.85	7.85	7.85	7.85	7.85	7.85
C20	5.62	5.62	5.62	5.62	5.62	5.62	5.62	5.62
C21	1000000.00	6.10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C22	1000000.00	5.71	11.41	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C23	1000000.00	7.33	10.37	15.56	20.74	1000000.00	1000000.00	1000000.00
C24	1000000.00	10.78	10.78	15.25	20.33	25.42	30.50	1000000.00
C25	35.11	1000000.00	1000000.00	24.83	24.83	25.08	30.09	35.11
C26	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C27	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C28	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C29	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C30	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C31	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C32	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C33	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C34	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C35	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C36	35.11	1000000.00	1000000.00	24.83	24.83	25.08	30.09	35.11
C37	25.25	10.78	10.78	15.25	20.33	25.25	25.25	25.25
C38	17.17	7.33	10.37	15.56	17.17	17.17	17.17	17.17
C39	9.45	5.71	9.45	9.45	9.45	9.45	9.45	9.45
C40	1000000.00	6.10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00

	57	58	59	60	61	62	63
C	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C1	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C2	8.56	11.41	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C3	15.56	15.56	15.56	20.74	1000000.00	1000000.00	1000000.00
C4	1000000.00	1000000.00	22.88	22.88	25.42	30.50	1000000.00
C5	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C6	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C7	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C8	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C9	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C10	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C11	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C12	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C13	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C14	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C15	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C16	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C17	21.71	21.71	21.71	21.71	21.71	21.71	21.71
C18	14.67	14.67	14.67	14.67	14.67	14.67	14.67
C19	7.85	7.85	7.85	7.85	7.85	7.85	7.85
C20	5.62	5.62	5.62	5.62	5.62	5.62	5.62
C21	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C22	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C23	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C24	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C25	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C26	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C27	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C28	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C29	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C30	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C31	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C32	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C33	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C34	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C35	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C36	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00
C37	25.25	25.25	25.25	25.25	25.25	25.25	25.25
C38	17.17	17.17	17.17	17.17	17.17	17.17	17.17
C39	9.45	9.45	9.45	9.45	9.45	9.45	9.45
C40	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00	1000000.00

Sensor	C	C1	C2	C3	C4	C5	C6	C7	C8
Coeficiente	72.377	93.802	51.468	9.995	75.963	140.411	108.351	108.817	64.858
Exponente	-1.527	-3.512	-1.075	-0.276	-1.002	-1.252	-1.223	-1.291	-1.015

Sensor	C9	C10	C11	C12	C13	C14	C15	C16	C17
Coeficiente	21.622	36.997	379.624	145.575	99.672	76.428	70.685	73.074	98.785
Exponente	-0.939	-10.161	-0.569	-1.206	-1.759	-1.924	-1.551	-1.562	-0.936

Sensor	C18	C19	C20	C21	C22	C23	C24	C25	C26
Coeficiente	29.611	8.919	4.708	88.014	59.921	71.716	66.052	69.734	63.024
Exponente	-3.682	-4.787	-5.996	-0.903	-0.535	-27.490	-1.641	-1.386	-1.869

Sensor	C27	C28	C29	C30	C31	C32	C33	C34	C35
Coeficiente	53.491	69.375	23.529	63.317	70.189	33.417	68.051	53.771	74.928
Exponente	-2.774	-2.293	-0.627	-2.007	-1.215	-2.228	-2.348	-3.111	-1.789

Sensor	C36	C37	C38	C39	C40
Coeficiente	75.528	108.935	22.956	0.001	15.760
Exponente	-1.465	-0.940	-72.747	-3.170	-22.707

	1	2	3	4	5	6	7	8	9	10	11	12
Valor IR real	414	117	57	35	26	21	18	359	83	37	24	17
Valor IR polinomio	408.45	116.20	51.51	37.95	24.28	21.56	16.24	359.63	83.74	38.16	25.18	17.03
Error	5.55	0.80	5.49	-2.95	1.72	-0.56	1.76	-0.63	-0.74	-1.16	-1.18	-0.03
Error Cuadrático	30.78	0.64	30.11	8.68	2.94	0.32	3.09	0.40	0.55	1.34	1.40	0.00

	13	14	15	16	17	18	19	20	21	22	23	24
Valor IR real	15	14	404	112	51	30	22	17	15	55	35	24
Valor IR polinomio	15.73	13.70	408.45	112.44	51.51	33.83	21.32	19.09	16.24	55.69	34.48	22.44
Error	-0.73	0.30	-4.45	-0.44	-0.51	-3.83	0.68	-2.09	-1.24	-0.69	0.52	1.56
Error Cuadrático	0.54	0.09	19.82	0.19	0.26	14.70	0.46	4.38	1.54	0.48	0.27	2.42

	25	26	27	28	29	30	31	32	33	34	35	36
Valor IR real	18	16	15	14	18	17	19	17	15	11	11	408
Valor IR polinomio	19.78	15.39	14.51	12.81	18.24	17.53	17.18	16.15	12.89	12.53	11.20	408.45
Error	-1.78	0.61	0.49	1.19	-0.24	-0.53	1.82	0.85	2.11	-1.53	-0.20	-0.45
Error Cuadrático	3.16	0.37	0.24	1.41	0.06	0.28	3.33	0.72	4.47	2.35	0.04	0.20

	37	38	39	40	41	42	43	44	45	46	47	48
Valor IR real	113	53	33	23	18	16	385	97	45	29	21	19
Valor IR polinomio	113.92	51.51	34.93	22.60	20.23	16.24	385.76	96.30	43.93	29.77	20.09	18.26
Error	-0.92	1.49	-1.93	0.40	-2.23	-0.24	-0.76	0.70	1.07	-0.77	0.91	0.74
Error Cuadrático	0.84	2.21	3.71	0.16	4.99	0.06	0.58	0.49	1.14	0.59	0.82	0.55

	49	50	51	52	53	54	55	56	57	58	59	60
Valor IR real	15	83	55	32	23	19	16	15	20	20	19	18
Valor IR polinomio	14.63	83.82	55.30	29.94	25.46	18.46	17.03	13.75	20.79	19.43	18.97	18.61
Error	0.37	-0.82	-0.30	2.06	-2.46	0.54	-1.03	1.25	-0.79	0.57	0.03	-0.61
Error Cuadrático	0.14	0.67	0.09	4.25	6.07	0.29	1.07	1.57	0.63	0.32	0.00	0.38

	61	62	63
Valor IR real	17	11	11
Valor IR polinomio	14.17	13.67	11.20
Error	2.83	-2.67	-0.20
Error Cuadrático	8.01	7.15	0.04

Error Cuadrático Medio	Error Medio	Error máximo
3.00	1.73	5.55

11.2 ANEXO 2: Scripts

SCRIPT ASOCIADO A ROBOBO DONDE SE DEFINEN LAS FUNCIONES

```
moveWheelsByTime=function(velocidad,tiempo,inString,inBuffer)
sim.setFloatSignal('duracionI',tiempo[1])
   sim.setFloatSignal('referencia_tiempo_I',sim.getSimulationTime())
   sim.setFloatSignal('duracionD',tiempo[1])
   sim.setFloatSignal('referencia_tiempo_D',sim.getSimulationTime())
   sim.setIntegerSignal('Bloqueado',1)
   if velocidad[2]>100 then
      sim.setIntegerSignal('velocidad_rueda_I',100)
   else
      if velocidad[2]<-100 then
         sim.setIntegerSignal('velocidad_rueda_I',-100)
      else
         if velocidad[2]>=-100 and velocidad[2]<=100 then
            sim.setIntegerSignal('velocidad_rueda_I',velocidad[2])
         end
      end
   end
   if velocidad[1]>100 then
      sim.setIntegerSignal('velocidad_rueda_D',100)
   else
      if velocidad[1]<-100 then
         sim.setIntegerSignal('velocidad_rueda_D',-100)
      else
         if velocidad[1]>=-100 and velocidad[1]<=100 then
            sim.setIntegerSignal('velocidad_rueda_D',velocidad[1])
         end
      end
   end
return {},{},{},''
end
moveWheelsByDegrees=function(parametros,inFloats,rueda,inBuffer) -- parametros(Grados,Velocidad)
   rightMotor=sim.getObjectHandle("Right_Motor")
   tiempo=math.max((parametros[1]-(-0.000325*(math.abs(parametros[2])^3)
               +0.042847*(math.abs(parametros[2])^2)-2.064049*(math.abs(parametros[2]))-17.697271))
               /(-0.000046*(math.abs(parametros[2])^3)+0.005219*(math.abs(parametros[2])^2)
               +6.357077*(math.abs(parametros[2]))+51.366765),0)
   sim.setFloatSignal('duracionI',tiempo)
   sim.setFloatSignal('referencia_tiempo_I',sim.getSimulationTime())
   sim.setFloatSignal('duracionD',tiempo)
   sim.setFloatSignal('referencia_tiempo_D',sim.getSimulationTime())
```

```
        sim.setIntegerSignal('Bloqueado',1)
        if parametros[2]>100 then
            if rueda[1]=='left' then
                sim.setIntegerSignal('velocidad_rueda_I',100)
                elseif rueda[1]=='right' then
                    sim.setIntegerSignal('velocidad_rueda_D',100)
                    elseif rueda[1]=='both' then
                        sim.setIntegerSignal('velocidad_rueda_I',100)
                        sim.setIntegerSignal('velocidad_rueda_D',100)
            end
            elseif parametros[2]<-100 then
                if rueda[1]=='left' then
                    sim.setIntegerSignal('velocidad_rueda_I',-100)
                    elseif rueda[1]=='right' then
                        sim.setIntegerSignal('velocidad_rueda_D',-100)
                        elseif rueda[1]=='both' then
                            sim.setIntegerSignal('velocidad_rueda_I',-100)
                            sim.setIntegerSignal('velocidad_rueda_D',-100)
                end
                elseif (parametros[2]>=-100 and parametros[2]<=100) then
                if rueda[1]=='left' then
                    sim.setIntegerSignal('velocidad_rueda_I',parametros[2])
                    elseif rueda[1]=='right' then
                        sim.setIntegerSignal('velocidad_rueda_D',parametros[2])
                        elseif rueda[1]=='both' then
                            sim.setIntegerSignal('velocidad_rueda_I',parametros[2])
                            sim.setIntegerSignal('velocidad_rueda_D',parametros[2])
                end
        end
    return {},{},{},''
    end
    movePanTo=function(parametros,inFloat,inString,inBuffer) --parametros(Grados,Velocidad)
            panMotor=sim.getObjectHandle("Pan_Motor")
        sim.setFloatSignal('referencia_tiempo_pan',sim.getSimulationTime())
        sim.setIntegerSignal('posicion_pan_inicial',math.floor(sim.getJointPosition(panMotor)*180/math.pi))
        sim.setIntegerSignal('Bloqueado',1)
        if (parametros[1]-180)>160 then
            sim.setIntegerSignal('posicion_pan',-1*160)
        else
            if (parametros[1]-180)<-160 then
                sim.setIntegerSignal('posicion_pan',-1*-160)
            else
                if (parametros[1]-180)>-160 and (parametros[1]-180)<160 then
                    sim.setIntegerSignal('posicion_pan',-1*(parametros[1]-180))
                end
            end
```

```
    end
    if parametros[2]>100 then
        sim.setIntegerSignal('velocidad_pan',100)
    else
        if parametros[2]<1 then
            sim.setIntegerSignal('velocidad_pan',0)
        else
            sim.setIntegerSignal('velocidad_pan',parametros[2])
        end
    end
return {},{},{},''
end
moveTiltTo=function(parametros,inFloat,inString,inBuffer) --parametros(Grados,Velocidad)
        tiltMotor=sim.getObjectHandle("Tilt_Motor")
    sim.setFloatSignal('referencia_tiempo_tilt',sim.getSimulationTime())
    sim.setIntegerSignal('posicion_tilt_inicial',math.floor(sim.getJointPosition(tiltMotor)*180/math.pi+70))
    sim.setIntegerSignal('Bloqueado',1)
    if parametros[1]>105 then
        sim.setIntegerSignal('posicion_tilt',105)
    else
        if parametros[1]<5 then
            sim.setIntegerSignal('posicion_tilt',5)
        else
            if parametros[1]>=5 and parametros[1]<=105 then
                sim.setIntegerSignal('posicion_tilt',parametros[1])
            end
        end
    end
    if parametros[2]>100 then
        sim.setIntegerSignal('velocidad_tilt',100)
    else
        if parametros[2]<1 then
            sim.setIntegerSignal('velocidad_tilt',0)
        else
            sim.setIntegerSignal('velocidad_tilt',parametros[2])
        end
    end
return {},{},{},''
end
readAllIRSensor=function(inIntegers,inFloats,inStrings,inBuffer)
    distancia={}
    IR={}
    coef={72.3768, 93.8025, 51.4681, 9.9951, 75.9630, 140.4109,
          108.3509, 108.8175, 64.8584, 21.6218, 36.9968, 379.6236, 145.5747,
          99.6719, 76.4279, 70.6854, 73.0736, 98.7847, 29.6114, 8.9193, 4.7080,
          88.0139, 59.9209, 71.7158, 66.0515, 69.7341, 63.0244, 53.4914, 69.3750,
```

```
            23.5295, 63.3173, 70.1891, 33.4174, 68.0506, 53.7709, 74.9276, 75.5282,
            108.9353, 22.9556, 0.0010,15.7604}
        exp={-1.5266, -3.5119, -1.0750, -0.2755, -1.0023, -1.2522, -1.2228, -1.2909,
            -1.0154, -0.9392, -10.1613, -0.5689, -1.2057, -1.7593, -1.9242, -1.5508, -1.5622,
            -0.9355, -3.6815, -4.7873, -5.9959, -0.9032, -0.5352, -27.4898, -1.6408, -1.3861,-1.8691,
            -2.7744, -2.2928, -0.6267, -2.0070, -1.2153, -2.2282, -2.3477, -3.1106, -1.7889, -1.4648,
            -0.9396, -72.7472, -3.1703, -22.7072}
        IR_Front_C=sim.getObjectHandle('Ir_Front_C')
                                                                (…)
        IR_Front_C40=sim.getObjectHandle('Ir_Front_C_40')
        if sim.readProximitySensor(IR_Front_C)==0 then
            distancia[1]=1000000.00
            else
            resultado,distancia[1]=sim.readProximitySensor(IR_Front_C)
         end
                                                                (…)
        if sim.readProximitySensor(IR_Front_C40)==0 then
            distancia[41]=1000000
            else
            resultado,distancia[41]=sim.readProximitySensor(IR_Front_C40)
         end
        valorIR=0
        for i=1,41,1 do
            valorIR=valorIR+coef[i]*((100*distancia[i])^exp[i])
        end
        valorIRInt=math.floor(valorIR)
        IR[1]=valorIRInt+math.random(-5,5)/100*valorIRInt
        valorIR=0
                                                                (…)
(Se repite lo mismo para todos los sensors: IR_Front_C, IR_Front_L, IR_Front_R, IR_Front_L_FLOOR, IR_Front_R_FLOOR,
                                            IR_Back_C, IR_Back_L, IR_Back_R)
        return IR,{},{},''
    end

    readPanPosition=function(inIntegers,inFloats,inStrings,inBuffer)
    posicionPanInt={}
        PanMotor=sim.getObjectHandle("Pan_Motor")
        posicionPanFloat=-sim.getJointPosition(PanMotor)*180/math.pi
        posicionPanInt[1]=math.floor(posicionPanFloat)
        print(posicionPanInt)
        return posicionPanInt,{},{},''
    end
    readTiltPosition=function(inIntegers,inFloats,inStrings,inBuffer)
        posicionTiltInt={}
        TiltMotor=sim.getObjectHandle("Tilt_Motor")
        posicionTiltFloat=sim.getJointPosition(TiltMotor)*180/math.pi+70
```

```
    posicionTiltInt[1]=math.floor(posicionTiltFloat)
    print(posicionTiltInt)
    return posicionTiltInt,{},{},''
end
readWheel=function(inIntegers,inFloats,rueda,inBuffer)
    medidasRuedas={} -- (posicionRuedaI,posicionRuedaD,velocidadRuedaI,velocidadRuedaD)
    LeftMotor=sim.getObjectHandle("Left_Motor")
    RightMotor=sim.getObjectHandle("Right_Motor")
    posicionMotorIzdoFloat=sim.getJointPosition(LeftMotor)*180/math.pi-sim.getFloatSignal('posicionInicialLeftWheel')
    medidasRuedas[1]=math.floor(posicionMotorIzdoFloat)
    posicionMotorDchoFloat=sim.getJointPosition(RightMotor)*180/math.pi-sim.getFloatSignal('posicionInicialRightWheel')
    medidasRuedas[2]=math.floor(posicionMotorDchoFloat)
    velocidadMotorIzdo=sim.getIntegerSignal('velocidad_rueda_I')
    if velocidadMotorIzdo == nil then
        medidasRuedas[3]=0
    else
        medidasRuedas[3]=velocidadMotorIzdo
    end
    velocidadMotorDcho=sim.getIntegerSignal('velocidad_rueda_D')
    if velocidadMotorDcho == nil then
        medidasRuedas[4]=0
    else
        medidasRuedas[4]=velocidadMotorDcho
    end
    return medidasRuedas,{},{},''
end
resetWheelEncoders=function(inIntegers,inFloats,inStrings,inBuffer)
    LeftMotor=sim.getObjectHandle("Left_Motor")
    RightMotor=sim.getObjectHandle("Right_Motor")
    sim.setFloatSignal('posicionInicialLeftWheel',math.floor(sim.getJointPosition(LeftMotor)*180/math.pi))
    sim.setFloatSignal('posicionInicialRightWheel',math.floor(sim.getJointPosition(LeftMotor)*180/math.pi))
    return {},{},{},''
end
```

SCRIPT ASOCIADO AL JOINT DE LA RUEDA DERECHA

```
rightMotor=sim.getObjectHandle("Right_Motor")
if sim.getIntegerSignal('velocidad_rueda_D')==nil then
   sim.setJointTargetVelocity(rightMotor,0*math.pi/180)
elseif sim.getIntegerSignal('velocidad_rueda_D')==0 then
   sim.setJointTargetVelocity(rightMotor,sim.getIntegerSignal('velocidad_rueda_D')*math.pi/180)
   if (sim.getSimulationTime()-sim.getFloatSignal('referencia_tiempo_D'))>=sim.getFloatSignal('duracionD') then
      sim.setJointTargetVelocity(rightMotor,0)
      sim.clearFloatSignal('duracionD')
      sim.clearFloatSignal('referencia_tiempo_D')
      sim.clearIntegerSignal('velocidad_rueda_D')
   end
   elseif sim.getIntegerSignal('velocidad_rueda_D')>0 and sim.getIntegerSignal('velocidad_rueda_D')<=100  then
      sim.setJointTargetVelocity(rightMotor,(math.max(-0.000046*(sim.getIntegerSignal('velocidad_rueda_D')^3)
                        *sim.getFloatSignal('duracionD')+0.005219*(sim.getIntegerSignal('velocidad_rueda_D')^2)
                        *sim.getFloatSignal('duracionD')+6.357077*(sim.getIntegerSignal('velocidad_rueda_D'))
                        *sim.getFloatSignal('duracionD')+51.366765*sim.getFloatSignal('duracionD')
                        -0.000325*(sim.getIntegerSignal('velocidad_rueda_D')^3)
                        +0.042847*(sim.getIntegerSignal('velocidad_rueda_D')^2)
                        -2.064049*(sim.getIntegerSignal('velocidad_rueda_D'))-17.697271,0)
                        /sim.getFloatSignal('duracionD')*math.pi/180)
      if (sim.getSimulationTime()-sim.getFloatSignal('referencia_tiempo_D'))>=sim.getFloatSignal('duracionD') then
         sim.setIntegerSignal('Bloqueado',0)
         sim.setJointTargetVelocity(rightMotor,0)
         sim.clearFloatSignal('duracionD')
         sim.clearFloatSignal('referencia_tiempo_D')
         sim.clearIntegerSignal('velocidad_rueda_D')
      end
      elseif sim.getIntegerSignal('velocidad_rueda_D')<0 and sim.getIntegerSignal('velocidad_rueda_D')>=-100  then
         sim.setJointTargetVelocity(rightMotor,-(math.max(-0.000046*((-sim.getIntegerSignal('velocidad_rueda_D'))^3)
                        *sim.getFloatSignal('duracionD')+0.005219*((-sim.getIntegerSignal('velocidad_rueda_D'))^2)
                        *sim.getFloatSignal('duracionD')+6.357077*((-sim.getIntegerSignal('velocidad_rueda_D')))
                        *sim.getFloatSignal('duracionD')+51.366765*sim.getFloatSignal('duracionD')
                        -0.000325*((-sim.getIntegerSignal('velocidad_rueda_D')^3))
                        +0.042847*((-sim.getIntegerSignal('velocidad_rueda_D'))^2)
                        -2.064049*((-sim.getIntegerSignal('velocidad_rueda_D')))-17.697271,0)
                        /sim.getFloatSignal('duracionD')*math.pi/180))
         if (sim.getSimulationTime()-sim.getFloatSignal('referencia_tiempo_D'))>=sim.getFloatSignal('duracionD') then
            sim.setIntegerSignal('Bloqueado',0)
            sim.setJointTargetVelocity(rightMotor,0)
            sim.clearFloatSignal('duracionD')
            sim.clearFloatSignal('referencia_tiempo_D')
            sim.clearIntegerSignal('velocidad_rueda_D')
         end
end
```

SCRIPT ASOCIADO AL JOINT DE LA RUEDA IZQUIERDA

```
leftMotor=sim.getObjectHandle("Left_Motor")
if sim.getIntegerSignal('velocidad_rueda_I')==nil then
    sim.setJointTargetVelocity(leftMotor,0*math.pi/180)
elseif sim.getIntegerSignal('velocidad_rueda_I')==0 then
    sim.setJointTargetVelocity(leftMotor,sim.getIntegerSignal('velocidad_rueda_I')*math.pi/180)
    if (sim.getSimulationTime()-sim.getFloatSignal('referencia_tiempo_I'))>=sim.getFloatSignal('duracionI') then
        sim.setJointTargetVelocity(leftMotor,0)
        sim.clearFloatSignal('duracionI')
        sim.clearFloatSignal('referencia_tiempo_I')
        sim.clearIntegerSignal('velocidad_rueda_I')
    end
    elseif sim.getIntegerSignal('velocidad_rueda_I')>0 and sim.getIntegerSignal('velocidad_rueda_I')<=100  then
        sim.setJointTargetVelocity(leftMotor,(math.max(-0.000046*(sim.getIntegerSignal('velocidad_rueda_I')^3)
                        *sim.getFloatSignal('duracionI')+0.005219*(sim.getIntegerSignal('velocidad_rueda_I')^2)
                        *sim.getFloatSignal('duracionI')+6.357077*(sim.getIntegerSignal('velocidad_rueda_I'))
                        *sim.getFloatSignal('duracionI')+51.366765*sim.getFloatSignal('duracionI')
                        -0.000325*(sim.getIntegerSignal('velocidad_rueda_I')^3)
                        +0.042847*(sim.getIntegerSignal('velocidad_rueda_I')^2)
                        -2.064049*(sim.getIntegerSignal('velocidad_rueda_I'))
                        -17.697271,0)/sim.getFloatSignal('duracionI')*math.pi/180))
        if (sim.getSimulationTime()-sim.getFloatSignal('referencia_tiempo_I'))>=sim.getFloatSignal('duracionI') then
            sim.setIntegerSignal('Bloqueado',0)
            sim.setJointTargetVelocity(leftMotor,0)
            sim.clearFloatSignal('duracionI')
            sim.clearFloatSignal('referencia_tiempo_I')
            sim.clearIntegerSignal('velocidad_rueda_I')
        end
        elseif sim.getIntegerSignal('velocidad_rueda_I')<0 and sim.getIntegerSignal('velocidad_rueda_I')>=-100  then
            sim.setJointTargetVelocity(leftMotor,-(math.max(-0.000046*((-sim.getIntegerSignal('velocidad_rueda_I'))^3)
                        *sim.getFloatSignal('duracionI')+0.005219*((-sim.getIntegerSignal('velocidad_rueda_I'))^2)
                        *sim.getFloatSignal('duracionI')+6.357077*((-sim.getIntegerSignal('velocidad_rueda_I')))
                        *sim.getFloatSignal('duracionI')+51.366765*sim.getFloatSignal('duracionI')
                        -0.000325*((-sim.getIntegerSignal('velocidad_rueda_I'))^3)
                        +0.042847*((-sim.getIntegerSignal('velocidad_rueda_I'))^2)
                        -2.064049*((-sim.getIntegerSignal('velocidad_rueda_I')))
                        -17.697271,0)/sim.getFloatSignal('duracionI')*math.pi/180))
            if (sim.getSimulationTime()-sim.getFloatSignal('referencia_tiempo_I'))>=sim.getFloatSignal('duracionI') then
                sim.setIntegerSignal('Bloqueado',0)
                sim.setJointTargetVelocity(leftMotor,0)
                sim.clearFloatSignal('duracionI')
                sim.clearFloatSignal('referencia_tiempo_I')
                sim.clearIntegerSignal('velocidad_rueda_I')
            end
    end
```

SCRIPT ASOCIADO AL JOINT DEL PAN

```
PanMotor=sim.getObjectHandle("Pan_Motor")
parametro=0.0
if sim.getIntegerSignal('velocidad_pan')==nil or
sim.getIntegerSignal('velocidad_pan')<=0 or
sim.getIntegerSignal('posicion_pan')==nil or parametro==nil then

   sim.setJointTargetVelocity(PanMotor,0*math.pi/180)
else
   if sim.getJointPosition(PanMotor) > sim.getIntegerSignal('posicion_pan')/180*math.pi then
      parametro=-1.0
      elseif sim.getJointPosition(PanMotor) < sim.getIntegerSignal('posicion_pan')/180*math.pi then
         parametro=1.0
         elseif sim.getJointPosition(PanMotor) >= sim.getIntegerSignal('posicion_pan')/180*math.pi
              -0.0001 and sim.getJointPosition(PanMotor) <= sim.getIntegerSignal('posicion_pan')/180*math.pi+0.0001 then
            parametro=0.0
   end
   if (sim.getIntegerSignal('velocidad_pan')>0 and sim.getIntegerSignal('velocidad_pan')<=100) then
      sim.setJointTargetVelocity(PanMotor,parametro*(math.abs((sim.getIntegerSignal('posicion_pan_inicial'))
                        -sim.getIntegerSignal('posicion_pan'))/
                        ((math.abs((sim.getIntegerSignal('posicion_pan_inicial'))
                        -(sim.getIntegerSignal('posicion_pan')))-
                        (-0.000020*(sim.getIntegerSignal('velocidad_pan')^3)
                        +0.001064*(sim.getIntegerSignal('velocidad_pan')^2)
                        -0.330338*(sim.getIntegerSignal('velocidad_pan'))-0.890735))/
                        (-0.000031*(sim.getIntegerSignal('velocidad_pan')^3)
                        +0.003877*(sim.getIntegerSignal('velocidad_pan')^2)
                        +0.847465*(sim.getIntegerSignal('velocidad_pan'))+8.054684)))*math.pi/180)
      if (math.abs(sim.getJointPosition(PanMotor)-sim.getIntegerSignal('posicion_pan')/180*math.pi))<0.03 then
         sim.setIntegerSignal('Bloqueado',0)
         sim.setJointTargetVelocity(PanMotor,0)
         sim.clearIntegerSignal('velocidad_pan')
         sim.clearIntegerSignal('posicion_pan')
         --sim.clearIntegerSignal('posicion_pan_inicial')
         sim.clearFloatSignal('referencia_tiempo_pan')
         parametro=0.0
      end
   end
end
```

SCRIPT ASOCIADO AL JOINT DEL TILT

```
tiltMotor=sim.getObjectHandle("Tilt_Motor")
parametro=0.0
if sim.getIntegerSignal('velocidad_tilt')==nil or sim.getIntegerSignal('velocidad_tilt')<=0 or
                    sim.getIntegerSignal('posicion_tilt')==nil or parametro==nil then
   sim.setIntegerSignal('velocidad_tilt',0)
   sim.setJointTargetVelocity(tiltMotor,sim.getIntegerSignal('velocidad_tilt')*math.pi/180)
else
   if sim.getIntegerSignal('posicion_tilt')*math.pi/180 > (sim.getJointPosition(tiltMotor)*180/math.pi+70)*math.pi/180 then
      parametro=1.0
      elseif sim.getIntegerSignal('posicion_tilt')*math.pi/180 < (sim.getJointPosition(tiltMotor)*180/math.pi+70)*math.pi/180
then
         parametro=-1.0
         elseif ( (sim.getJointPosition(tiltMotor)*180/math.pi+70)<=sim.getIntegerSignal('posicion_tilt') +0.001
         and (sim.getJointPosition(tiltMotor)*180/math.pi+70)) >= sim.getIntegerSignal('posicion_tilt')-0.001 then
            parametro=0.0
   end
   if (sim.getIntegerSignal('velocidad_tilt')>0 and sim.getIntegerSignal('velocidad_tilt')<=100) then
      sim.setJointTargetVelocity(tiltMotor,parametro*math.max(((math.abs((sim.getIntegerSignal('posicion_tilt_inicial'))
                        -(sim.getIntegerSignal('posicion_tilt')))/
                        ((math.abs((sim.getIntegerSignal('posicion_tilt_inicial'))
                        -(sim.getIntegerSignal('posicion_tilt')))-
                        (-0.000088*(sim.getIntegerSignal('velocidad_tilt')^3)
                        +0.012325*(sim.getIntegerSignal('velocidad_tilt')^2)
                        -0.549530*(sim.getIntegerSignal('velocidad_tilt'))
                        +4.737946))/(0.000014*(sim.getIntegerSignal('velocidad_tilt')^3)
                        -0.002598*(sim.getIntegerSignal('velocidad_tilt')^2)
                        +0.480940*(sim.getIntegerSignal('velocidad_tilt'))
                        +3.181534))*math.pi/180))),0)
      if math.abs((sim.getJointPosition(tiltMotor)*180/math.pi+70)*math.pi/180
               - sim.getIntegerSignal('posicion_tilt')*math.pi/180)<0.01 then
         sim.setIntegerSignal('Bloqueado',0)
         sim.setJointTargetVelocity(tiltMotor,0)
         sim.clearIntegerSignal('velocidad_tilt')
         sim.clearIntegerSignal('posicion_tilt')
         sim.clearIntegerSignal('posicion_tilt_inicial')
         sim.clearFloatSignal('referencia_tiempo_tilt')
      end
   end
end
```

12 REFERENCIAS

[1] F. Bellas, M. Naya, G. Varela, L. Llamas, A. Prieto, J.C. Becerra, M. Bautista, A. Faiña, R.J. Duro. (2017). The Robobo Project: Bringing Educational Robotics Closer to Real-World Applications (Best paper award), Proceedings of 8th International Conference on Robotics in Education (RIE 2017), pp 226-237

[2] Robótica autónoma. (2018). Wikipedia, La enciclopedia libre. Fecha de consulta: 10 de abril de 2019. Desde https://es.wikipedia.org/w/index.php?title=Rob%C3%B3tica_aut%C3%B3noma&oldid=106690590.

[3] Alejandro Germán Frank, Lucas Santos Dalenogare, Néstor Fabián Ayala. (2019). Industry 4.0 technologies: Implementation patterns in manufacturing companies. International Journal of Production Economics. 210. 15-26.

[4] Deloitte Touche Tohmatsu Limited. (2017). Forces of change: Industry 4.0. Desde https://www2.deloitte.com/es/es/pages/manufacturing/articles/que-es-la-industria-4.0.html.

[5] International Federation of Robotics (2018). Executive Summary World Robotics 2018 Industrial Robots. Desde https://ifr.org/downloads/press2018/Executive_Summary_WR_2018_Industrial_Robots.pdf

[6] Bravo Sánchez, F., Forero Guzmán, A. (2012). La robótica como un recurso para facilitar el aprendizaje y desarrollo de competencias generales. Teoría de la Educación. Educación y Cultura en la Sociedad de la Información, 13 (2), 120-136.

[7] Robobo. (2019). Desde https://therobobоproject.com/

[8] E. Fabregas, G. Farias, E. Peralta, H. Vargas and S. Dormido. (2016) "Teaching control in mobile robotics with V-REP and a Khepera IV library". IEEE Conference on Control Applications (CCA), Buenos Aires, 2016, pp. 821-826.

[9] Coppelia Robobtics. (2019). Desde http://www.coppeliarobotics.com/assets/v-repoverviewpresentation.pdf.

[10] Lego Mindstorms. (2019). Wikipedia, La enciclopedia libre. Fecha de consulta: 10 de abril de 2019. Desde https://es.wikipedia.org/w/index.php?title=Lego_Mindstorms&oldid=116245967.

[11] Mindstorms Lego (2019). Desde https://www.lego.com/https://www.lego.com/es-es/mindstorms/support.

[12] Martín Domínguez, A. (2016). Modelado y Simulación de un Robot LEGO Mindstorms EV3 mediante V-REP y Matlab. Universidad de Málaga, Málaga.

[13] TurtleBot (2018). En Wikipedia, La Enciclopedia Libre. Fecha de consulta: 10 de abril de 2019. Desde https://en.wikipedia.org/w/index.php?title=TurtleBot&oldid=869524119.

[14] TurtleBot. (2019). Desde https://www.turtlebot.com/about.

[15] Fernández Vega, I. (2016). Desarrollo de un entorno de programación para un robot simulado TurtleBot-2 con brazo manipulador Widowx mediante la conexión de V-REP y Matlab. Universidad de Málaga, Málaga.

[16] Oliveira, Marco & Borges, João Victor & B Leite, Joilnen & G O E Silva, Wagner & Lopes, Ricardo & Viana, Leonardo & Lima, Adeilson & de O S D'Amato, Flavio

& A de Amorim, José. (2017). ASH/UFAL @Home 2017 Team Description Paper-LARC/CBR.

[17] e-puck education robot. (2019). Desde http://www.e-puck.org/

[18] Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stéphane Magnenat, Jean-christophe Zufferey, Dario Floreano, Alcherio Martinoli. (2009). The e-puck, a robot designed for education in engineering.

[19] George Florea, Andrei. (2018). Integrating a V-REP simulated mobile robot into Ros. UPB. Bucarest.

[20] A. Shitsukane, W. Cheruiyot, C. Otieno and M. Mvurya, "Fuzzy Logic Sensor Fusion for Obstacle Avoidance Mobile Robot," 2018 IST-Africa Week Conference (IST-Africa), Gaborone, 2018, pp. Page 1 of 8-Page 8 of 8.

[21] K-Team. (2019). Consultado el 10 de abril de 2019 desde https://www.k-team.com/khepera-iv.

[22] E. Peralta, E. Fabregas, G. Farias, H. Vargas, S. Dormido. (2016). Development of Khepera IV Library for the V-REP Simulator. Pontificia Universidad Católica de Valparaíso. Chile.

[23] G. Farias, E. Fabregas, E. Peralta, H. Vargas, S. Dormido-Canto, S. Dormido. (2019). "Development of an Easy-to-Use Multi-Agent Platform for Teaching Mobile Robotics," in IEEE Access, vol. 7, pp. 55885-55897.

[24] Robot NAO - Aliverobots. (2019). Desde https://aliverobots.com/nao/.

[25] M. Al-Hami, R. Lakaemper. (2014). "Sitting pose generation using genetic algorithm for NAO humanoid robots," 2014 IEEE International Workshop on Advanced Robotics and its Social Impacts, Evanston, IL, pp. 137-142.

[26] Kumar, PB, Mohapatra, S, Parhi, DR. (2018). An intelligent navigation of humanoid NAO in the light of classical approach and computational intelligence. Comput Anim Virtual Worlds.

[27] S. Michieletto, D. Zanin and E. Menegatti. (2013). NAO Robot Simulation for Service Robotics Purposes. 2013 European Modelling Symposium, Manchester, pp. 477-482

[28] Pérez Mc'Intosh, Vanesa. (2016). Mejora del modelo de simulación de una plataforma robótica móvil controlada mediante smartphone. Universidade da Coruña, Ferrol.

[29] Vishay. (2019). Desde https://www.vishay.com/docs/84274/vcnl4040.pdf

[30] The Programming Language Lua. (2019). http://www.lua.org/about.html.