UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

DISEÑO Y DESARROLLO DE CONTROLES DE ROBOTS MEDIANTE MÉTODOS NUMÉRICOS BASADOS EN TEORÍA DE ÁLGEBRA LINEAL. APLICACIÓN A ROBOTS MÓVILES.

AUTOR: SERGIO MORA CARRIÓN

TUTOR: ÁNGEL VALERA FERNÁNDEZ

COTUTORA: MARINA VALLÉS MIQUEL

Curso Académico: 2017-18

Agradecimientos

A mi familia, por su apoyo y su confianza en mí.

A mis amigos, por todos los momentos disfrutados juntos.

A Gustavo y Ángel, por todo lo aprendido y su ayuda durante el proyecto.

A Camila, por estar siempre a mi lado y darme fuerza en los peores momentos.

Diseño y desarrollo de controles de robots mediante métodos numéricos basados en teoría de álgebra lineal. Aplicación a robots móviles.

Diseño y desarrollo de controles de robots mediante métodos numéricos basados en teoría de álgebra lineal. Aplicación a robots móviles.

Resumen

En este Trabajo de Fin de Grado se han diseñado y desarrollado unos novedosos controladores de robots basados en álgebra lineal. Además, para confirmar la viabilidad de los mismos, han sido implementados en robots móviles diferenciales.

Para introducir al lector en el proyecto, se describen los objetivos del mismo así como la motivación que supone llevarlo a cabo. Se hace mención al pasado, presente, y futuro de la robótica, junto con el mercado que hoy en día ya ha conseguido y el potencial del mismo. También se hace un breve repaso de la historia de la robótica y se hace una clasificación morfológica de los robots. Una vez introducido el lector, se expone la estrategia de control a seguir y se describen minuciosamente los nuevos controladores a utilizar, así como todas las herramientas matemáticas requeridas en el proyecto.

Tras el desarrollo teórico, es momento de pasar al desarrollo práctico del proyecto. En primer lugar, se hace una presentación del material y del software utilizado. Posteriormente, se explica el procedimiento seguido para llevar a cabo la simulación del robot, así como las mejoras introducidas con los integradores. Tras esto, se exponen los códigos programados en las dos plataformas utilizadas y la generación de distintas trayectorias.

Al finalizar el trabajo práctico, se entra en la exposición de resultados y el análisis de los mismos. Se muestran varias figuras ilustrativas y se proponen una serie de trabajos futuros que surgen tras la finalización con éxito de este proyecto.

Por último se adjunta, como en cualquier proyecto de ingeniería, el presupuesto del proyecto.

Palabras claves: robótica, robot móvil, controladores, álgebra lineal, Matlab, Simulink, generación de trayectorias, simulación, programación, integradores.

Diseño y desarrollo de controles de robots mediante métodos numéricos basados en teoría de álgebra lineal. Aplicación a robots móviles.

Resum

En aquest Treball de Fi de Grau s'han dissenyat i desenvolupat una serie de nous controladors de robots basats en àlgebra lineal. A més, per a confirmar la viabilitat dels mateixos, han sigut implementats en robots mòbils diferencials.

Per introduir el lector en el projecte, es descriuen els objectius del mateix així com la motivació que suposa dur-lo a terme. Es fa esment al passat, present i futur de la robótica, junt al mercat que hui en dia ja ha aconseguit i el potencial del mateix. També es fa un breu repàs de la història de la robòtica i es presenta una classificació morfològica dels robots. Una vegada introduït el lector, s'exposa l'estratègia de control a seguir i es descriuen de forma minuciosa els nous controladors a utilizar, així com totes les ferramentes matemàtiques requerides en el projecte.

Després del desenvolupament teòric, es moment de passar al pràctic. En primer lloc, es presenta el material i el software utilitzat en el projecte. Després, s'explica el procediment seguit per a dur a terme la simulació del robot, així como les millores introduïdes amb els integradors. Per últim, s'exposen els còdigs programats en les dues plataformes i la generació de distintes trajectòries.

Al finalitzar el treball pràctic, s'exposen i s'analitzen els resultats del mateix. Es mostren vàries figures il·lustratives i es proposen una serie de treballs futurs que sorgeixen amb la finalització amb èxit d'aquest projecte.

Per últim s'adjunta, com en qualsevol treball d'enginyeria, el presupost del projecte.

Paraules clau: robòtica, robot mòbil, controladors, àlgebra lineal, Matlab, Simulink, generació de trajectòries, simulació, programació, integradors.

Diseño y desarrollo de controles de robots mediante métodos numéricos basados en teoría de álgebra lineal. Aplicación a robots móviles.

Abstract

In that Final Degree Project, new controllers based on linear algebra have been designed and developed. Besides, in order to confirm the feasibility of those, they have been implemented in mobile differential robots.

To introduce the reader to the project, the objectives and the motivation in carrying it out are described. Mention is made of the past, present and future of robotics, along with the market it has already achieved today and its potential. The history of robotics is also briefly reviewed and a morphological classification of robots is made. Once the reader is introduced, the control strategy and the controllers themselves are explained thoroughly, as well as all the mathematical tools required in the project.

After having exposed the theoretical development, it is time to move on the practical development of the project. First, a presentation of the material and software used is made. Afterwards, the procedure followed to carry out the simulation of the robot is explained, as well as the improvements that integrators confer. After this, the codes programmed in both robots and the generation of several trajectories are exposed.

At the end of the practical work, the results are presented and analyzed. Several illustrative figures are shown and a series of future works that emerge after the successful completion of this project are proposed.

Finally, as in any engineering project, the budget of the project is attached.

Keywords: robotics, mobile robots, controllers, linear algebra, Matlab, Simulink, trajectories generation, simulation, programming, integrators.

Diseño y desarrollo de controles de robots mediante métodos numéricos basados en teoría de álgebra lineal. Aplicación a robots móviles.

Documentos contenidos en el TFG

- Memoria
- Presupuesto

Índice de la memoria

Índice del presupuesto

Listado de figuras

Listado de tablas

Listado de ecuaciones

DOCUMENTO I: LA MEMORIA

1 INTRODUCCIÓN

En los últimos años el control de robots en general y la navegación de robots móviles en particular está siendo un área de interés para investigadores y desarrolladores industriales. De esta forma, si se aborda el control cinemático de robots móviles se pueden encontrar diversas estrategias de control que resuelven el control de trayectorias y el control de caminos. En la primera clase de controladores la posición y la orientación del robot está en función del tiempo, mientras que en la segunda clase de controladores el robot debe seguir una descripción geométrica del camino que debe seguir el robot, de manera que el tiempo no se considera como variable.

En el presente Trabajo Fin de Grado se propone desarrollar controladores cinemáticos de trayectorias para robots móviles. Los controladores que se proponen utilizar están basados en métodos numéricos de álgebra lineal. Se trata de un conjunto de técnicas novedosas que son muy prometedoras puesto que permiten abordar problemas complejos con un coste computacional reducido. Para validar los controladores desarrollados se piensa implementarlos tanto en un entorno de simulación como es Simulink, como en robots móviles basados en la plataforma Lego Mindstorms.

El proyecto se va a centrar en el control de robots móviles, pero posteriormente se estudiará la posibilidad de desarrollar controladores basados en métodos numéricos también para robots industriales como son los robots paralelos disponibles en el Laboratorio de Robótica del Departamento de Ingeniería de Sistemas y Automática. Este trabajo de fin de grado es sin duda una primera toma de contacto con estos novedosos controladores. La consecución con éxito del mismo abrirá un amplio abanico de trabajos futuros, que permitirán no solo utilizar los controladores en robots móviles, sino en otros muchos sistemas robotizados diferentes.

1.1 Objetivos

Una vez hecha la introducción al proyecto, es conveniente enumerar los principales objetivos que este persigue. La principal finalidad de exponer dichos objetivos es revisarlos tras la finalización del proyecto para comprobar que han sido alcanzados y con ello confirmar la viabilidad del TFG.

Los principales objetivos de este trabajo final de grado son:

- Diseñar y desarrollar los novedosos controladores de robots basados en teoría de álgebra lineal.
- Utilizar el software Simulink para realizar las primeras pruebas y confirmar el adecuado funcionamiento de los controladores.
- Utilizar integradores para minimizar el error en el seguimiento de la trayectoria.
- Implementar los controladores en dos robots diferenciales: Lego NXT y Lego EV3. Con ello se confirmará el adecuado funcionamiento de los mismos.
- Utilizar las curvas de Bézier como herramienta para generar trayectorias reales.
- Valorar el proyecto económicamente mediante el presupuesto del mismo.

1.2 Motivación

Desde el principio de nuestros días, los humanos hemos tratado de diseñar e implementar sistemas o máquinas que nos sustituyan en nuestras labores o trabajos. De hecho, hoy en día es impensable encontrar cualquier actividad, ya sea productiva o no, que carezca de estos medios de apoyo. Sin duda alguna hemos encontrado en la tecnología un aliado a la hora de realizar nuestras actividades.

Más en concreto, la automatización, la robótica y el control industrial han jugado y jugarán un papel esencial en nuestros trabajos. Para ello, es esencial que invirtamos tiempo y medios con el fin de lograr que los humanos dejemos de realizar tareas peligrosas o físicamente extenuantes.

Sin embargo, esta no es la única motivación para que se inviertan recursos en estos campos. La importancia del sector de la fabricación en la economía de los países se ha puesto de manifiesto en la crisis actual. Muchos expertos abogan por potenciar un sector clave que la sociedad necesita como motor de su economía y que en los últimos años ha sido sometido a una gran presión por parte de países con costes laborales reducidos. Las consecuencias son evidentes y ciertamente traumáticas; deslocalización de muchas empresas y lo que es peor, la desaparición de una parte importante del tejido industrial. [1]

En este difícil entorno, las empresas industriales se enfrentan al reto de buscar e implantar nuevas tecnologías productivas y organizativas que les permitan afrontar el futuro con garantía de éxito. Solo un aumento drástico de la productividad que afecte a los costes y a la calidad salvará al sector de la fabricación y sus empleos. Sin duda alguna dicha innovación pasa por implantar nuevas tecnologías que mejoren la productividad y optimicen el uso de recursos.

Sin embargo, en la práctica, la implantación de nuevos medios y/o procesos no es tan sencilla. La instauración de nuevas tecnologías requiere una fuerte inversión inicial, inadmisible en ocasiones para muchas de las pymes de nuestro país. Además, las máquinas no pueden sustituir por completo la mano de obra humana. Por tanto, aparece también el reto de formar a nuevos profesionales con el fin de que puedan supervisar y controlar estos nuevos procesos productivos.

1.2.1 La robótica y la industria

Otra de las motivaciones que invitan a investigar en los campos anteriormente mencionados es el mercado que ya actualmente tienen los robots y su prometedor futuro. Para comprobar su potencial se hará una breve revisión del mercado de la robótica y de la utilización de robots en la industria tanto en nuestro país como en el resto del mundo.

En primer lugar es necesario conocer cuáles son las compañías que dominan el mercado de la robótica industrial.

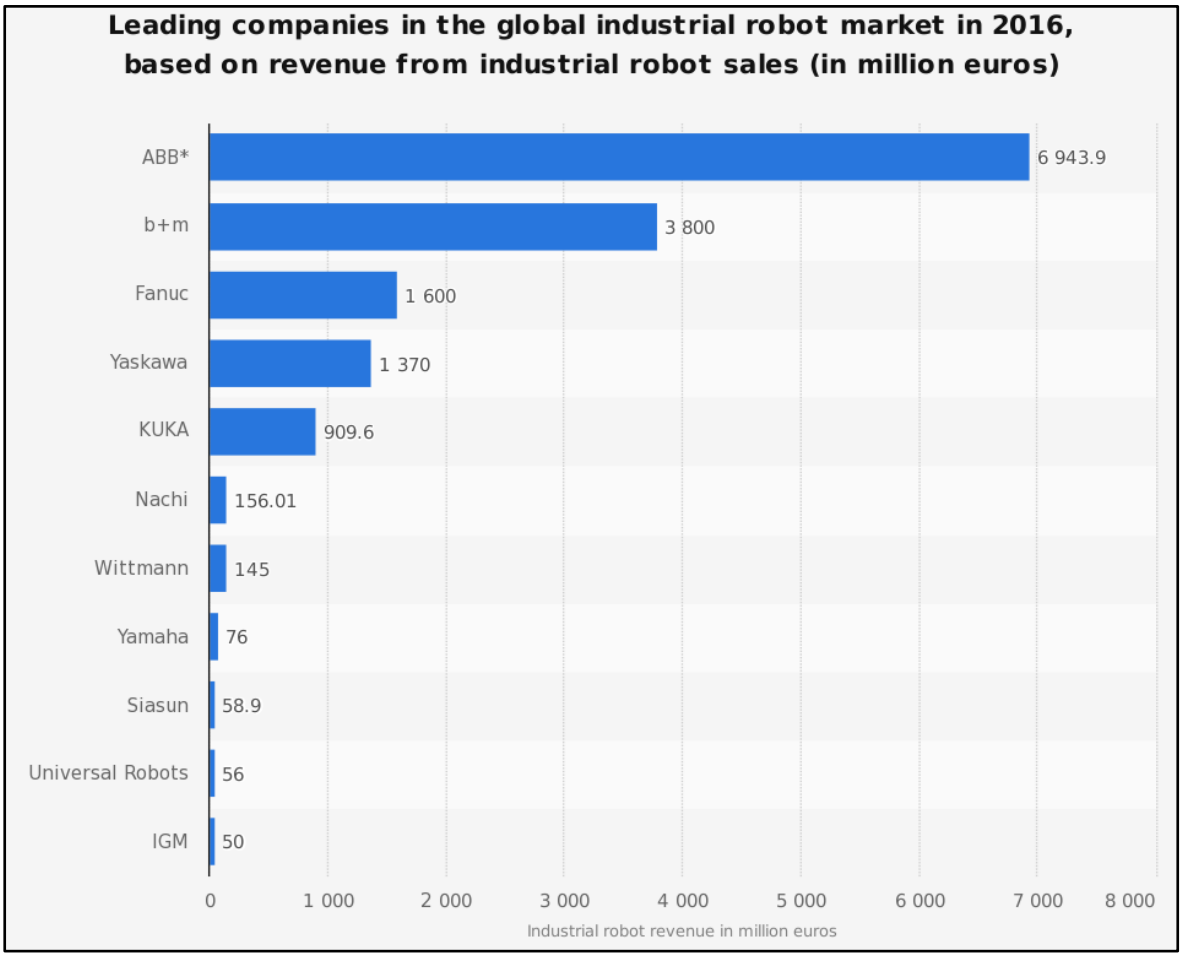


Figura 1. Principales empresas del sector de la robótica industrial

Fuente: Credit Suisse. "Leading Companies in The Global Industrial Robot Market in 2016, Based on Revenue from Industrial Robot Sales (in Million Euros)." Statista - The Statistics Portal, Statista, www.statista.com/statistics/257177/global-industrial-robot-market-share-by-company/

En el primer lugar de la lista figura de forma destacada la multinacional suiza ABB, con una facturación en el año 2016 de casi 7.000 millones de euros, superando holgadamente a su principal perseguidor, B+M.

Una vez vistas las principales empresas dominantes, es necesario conocer los países en los cuales se utiliza un mayor número de robots en la industria. Para ello disponemos de la siguiente figura, en la que se muestra la robotización de la industria en cada país.

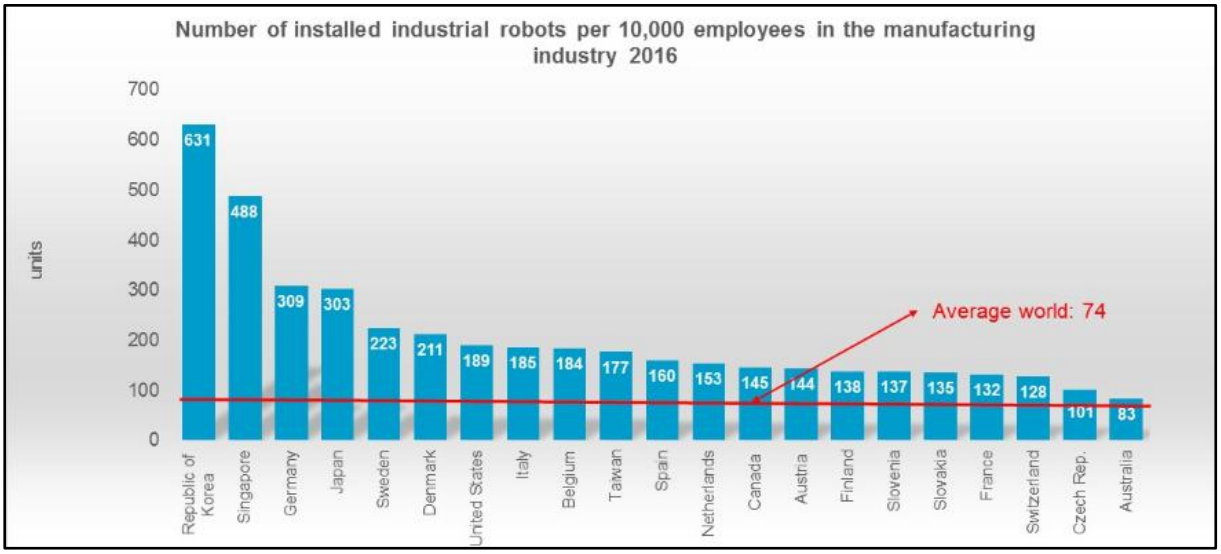


Figura 2. Número de robots empleados en la industria

Fuente: World robotics 2017. IFR. https://ifr.org/ifr-press-releases/news/robot-density-rises-globally

En este ranking elaborado en el año 2016 dominan dos países asiáticos: la República de Corea y Singapur. Le siguen grandes potencias como Alemania y Japón, y posteriormente, los países europeos más desarrollados y Estados Unidos.

Quizá pueda sorprender no localizar en estos primeros puestos a grandes productores como China, pero este gran país se encuentra en el puesto vigésimo tercero, con 68 robots por cada 10,000 trabajadores. Esta carencia de robots respecto a otros países productores se debe a que el gran gigante asiático está sufriendo en la actualidad un desarrollo considerable de su industria, por lo que la tecnología de los robots no se ha consolidado como si lo ha hecho en otros lugares.

También se puede echar en falta no encontrar al Reino Unido, una de las principales potencias europeas entre los veinte primeros puestos. El Reino Unido, con 71 robots cada 10,000 trabajadores se encuentra en el puesto número 22, justo por delante de China.

España, en el puesto número once, supera holgadamente la media mundial establecida en 74 robots por cada 10,000 trabajadores. Con 160 robots por cada 10,000, se coloca por encima de países más ricos como Suiza, Finlandia y el propio Reino Unido.

En cuanto a los sectores industriales que mayor número de robots utilizan destaca con fuerza la industria del automóvil, quedando por delante, pero cerca de la industria eléctrica y electrónica.

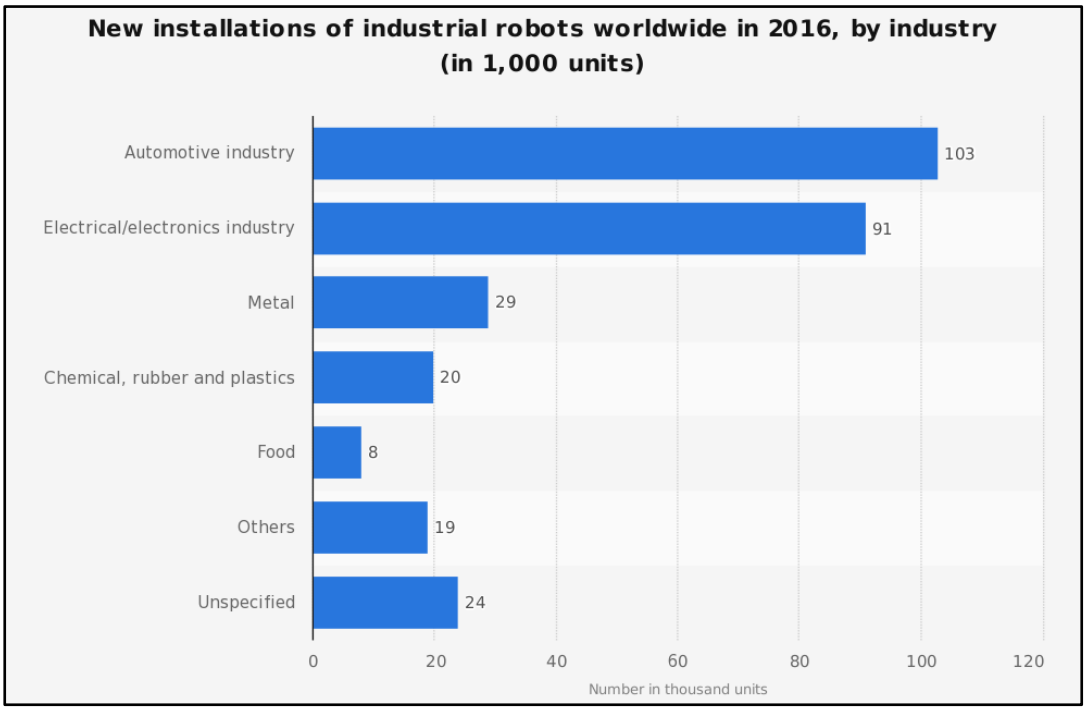


Figura 3. Los robots por sectores

Fuente: IFR. "New Installations of Industrial Robots Worldwide in 2016, by Industry (in 1,000 Units)." Statista - The Statistics Portal, Statista, www.statista.com/statistics/257080/new-installations-of-industrial-robots-worldwide-by-industry/

1.2.2 El futuro de la robótica

Finalmente, otra de las más importantes motivaciones para llevar a cabo este proyecto es el prometedor futuro de la robótica. Hace apenas unos años era impensable tener robots en nuestras casas, no obstante hoy en día es común encontrarlos en la mayoría de hogares. Es innegable, actualmente ya no vemos la robótica como una tecnología futura, sino que forma parte de nuestro presente. Pero esto no es todo, el crecimiento de la robótica va a más.

La industria 4.0 ya es una realidad, nos encontramos ante la cuarta revolución industrial. Por ello tenemos gran reto por delante, el cual consiste en saber que la tecnología ya no es una opción sino más bien un reto de adopción que las empresas deben tener. Se espera que los fabricantes de electrónica, así como la industria automotriz y la de alimentos y bebidas, sean las pioneras en la adopción de procesos de fabricación flexibles e individualizados. [2] Dentro de estos novedosos procesos, los robots juegan un papel crucial, ya que cada vez más tienden a ocupar puestos de trabajo que antes pertenecían a los humanos.

Figura 4. Industria 4.0

Fuente: AMETIC

Según el IFR, International Federation of Robotics, el Mercado de la robótica industrial en el mundo crecerá aproximadamente un 37% desde la actualidad hasta el año 2020. Llegando a alcanzar así las 520.9 mil unidades instaladas.

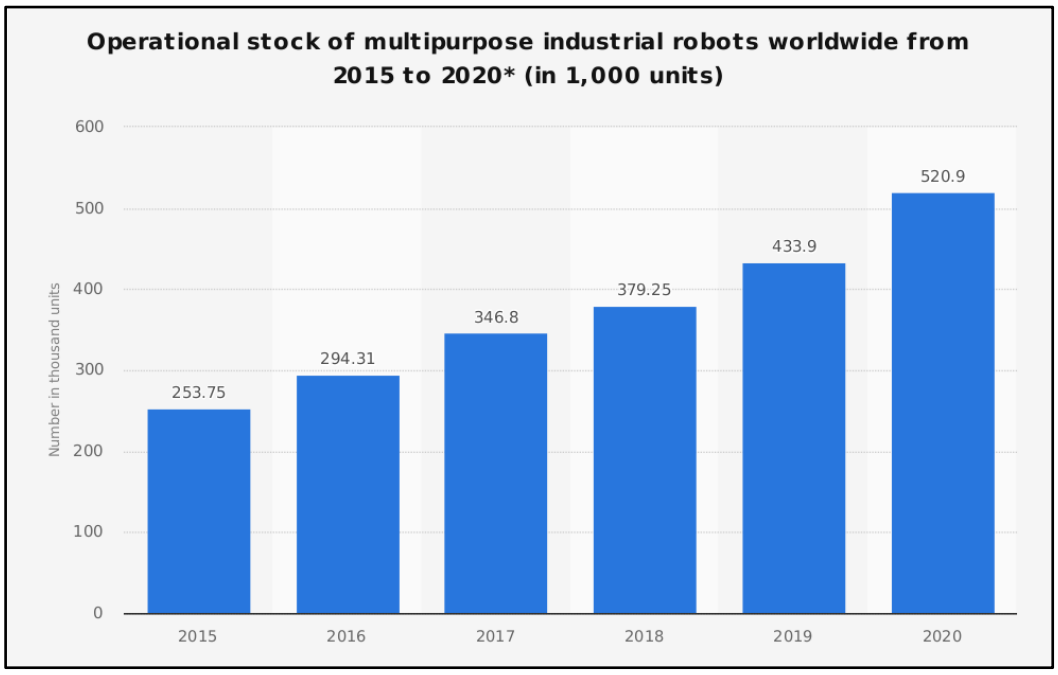


Figura 5. Previsión de la venta de robots industriales

Fuente: IFR. "Operational Stock of Multipurpose Industrial Robots Worldwide from 2015 to 2020 (in 1,000 Units)." Statista - The Statistics Portal, Statista, www.statista.com/statistics/281380/estimated-operational-stock-of-industrial-robots-worldwide/*

Además de los datos anteriormente mencionados, una motivación extra es resolver los retos que ya hoy en día propone la robótica. Muchos de estos retos quizá no tengan respuesta, pero muchos otros sí, y tarde o temprano se dará con ella.

2 TRABAJO TEÓRICO

En este capítulo se describirán los fundamentos teóricos necesarios para el adecuado desarrollo del proyecto.

2.1 Historia de la robótica

La palabra robot fue usada por primera vez en el año 1920, cuando el escritor checo Karel Capek (1890-1938) estrenó en el teatro nacional de Praga su obra "Rossum´s Universal Robot". Su origen es la palabra eslava "robota", que significa servidumbre o trabajo realizado de manera forzada [3]. Aunque en sus primeros días la robótica era un término ligado a la ciencia ficción, actualmente el término robot encierra una gran cantidad de mecanismos y máquinas en todas las áreas de nuestra vida cotidiana. Aunque su principal uso se encuentra en la industria, poco a poco los robots van apareciendo más y más en numerosos aspectos de nuestro día a día.

El uso de máquinas diseñadas para sustituir a los humanos en sus labores se remonta a la antigüedad, casi al principio de la civilización. Se sabe que en la antigua Grecia se utilizaban grúas que se movían mediante poleas y bombas hidráulicas con fines estéticos y artísticos. Uno de los pioneros en la robótica fue Herón de Alejandría, que aplicó sus conocimientos en los campos de la ingeniería y las matemáticas con el fin de crear múltiples artilugios que iban desde teatros autónomos hasta la que es considerada como la primera máquina expendedora, que proporcionaba agua bendita al introducir una moneda.[4]

Figura 6. Teatro automático de Herón de Alejandría

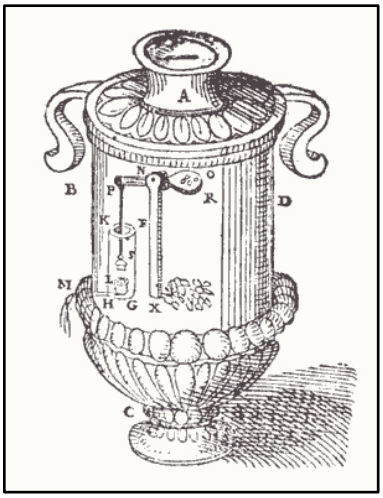


Figura 7. Primera máquina expendedora

Fuente: Primera máquina expendedora. Teknoplof http://www.teknoplof.com/2018/04/12/asi-la-primera-maquina-expendedora-dos-mil-anos/

Fuente: Teatro automático de Herón de Alejandría. Museo Kotsanas de la tecnología de la antigua Grecia http://kotsanas.com/gb/exh.php?exhibit=0101002

A lo largo de los años se siguieron implementando robots teleoperados, es decir, controlados por el ser humano. La primera implementación propiamente dicha no se dio hasta el año 1948, cuando el neurólogo Grey Walter construyó los primeros robots móviles utilizando el sentido común y algo de bricolaje: recibieron el nombre de las tortugas de Bristol. Estaban construidas utilizando

válvulas, sensores de luz y detectores de contacto. Tenían dos ruedas motrices y utilizaban como ojo un foto-tubo. A Walter Gray le atraía el concepto de que un robot, con conexiones simples era capaz de realizar comportamientos complejos al ser atraído por una luz. Fue en ese momento en el que surgió el concepto de comportamiento de los robots. [5]

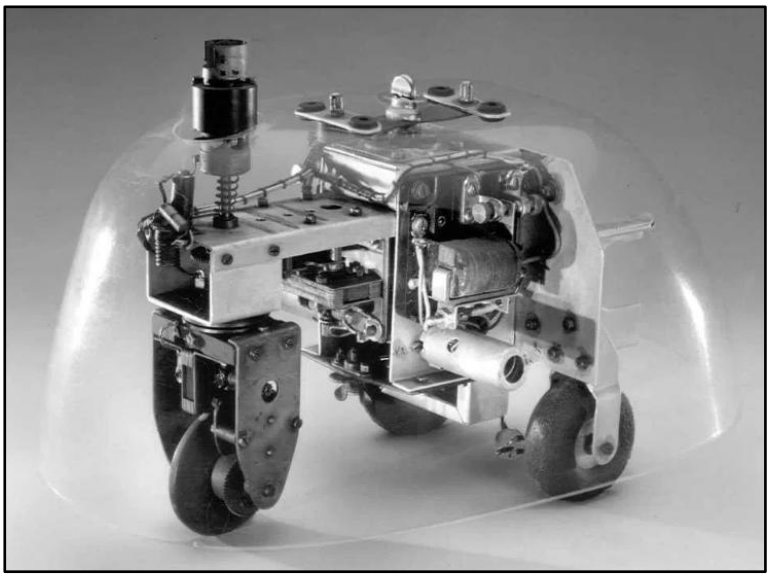


Figura 8. Las tortugas de Bristol

Fuente: APLOMA. https://alpoma.net/tecob/?p=11359lof

En 1954 el ingeniero George Devol patentó el primer robot programable. La programación surgió porque se dio cuenta que en la industria habían tareas que necesitaban ser hechas de forma repetitiva. En 1956, junto con el hombre de negocios Josef Engelberger, crean la primera empresa dedicada a la robótica: "Unimation" (Universal Automation).

En cuanto a la ciencia ficción, los robots también han sido objeto de numerosas obras literarias y películas. En 1950, Isaac Asimov publicaba su célebre obra "I, Robot" en español "Yo, robot".

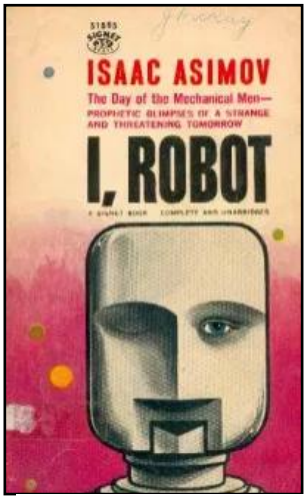


Figura 9. I, Robot

Fuente: Bookish Relish. http://bookish-relish.blogspot.com.es/2011/05/science-fiction-i-robot.html

En esta obra, el escritor expone las tres leyes de la robótica, a las que con posterioridad se añadió la ley cero. [6]

- Ley cero: Un robot no hará daño a la Humanidad o, por inacción, permitir que la Humanidad sufra daño.
- Ley uno: Un robot no hará daño a un ser humano o, por inacción, permitir que un ser humano sufra daño.
- Ley dos: Un robot debe obedecer las órdenes dadas por los seres humanos, excepto si estas órdenes entrasen en conflicto con la 1ª Ley.
- Ley tres: Un robot debe proteger su propia existencia en la medida en que esta protección no entre en conflicto con la 1ª o la 2ª Ley.

Con el fin de analizar el desarrollo de los robots a lo largo de la historia es interesante repasar las cinco generaciones que han existido hasta la época.

- Primera generación: Robots manipuladores.

Su desarrollo empieza en los años 50 y su morfología es similar a la de un brazo humano. [7] Disponen de cuerpos rígidos conectados con articulaciones que permiten su movimiento. Su sistema de control es relativamente sencillo, y pueden realizar tareas a partir de una serie de instrucciones programadas previamente. Dicho sistema de control está en lazo abierto, por lo que no se tienen en cuenta las perturbaciones del entorno. [8]

- Segunda generación: Robots de aprendizaje.

Los robots de aprendizaje se caracterizan por repetir una secuencia de movimientos que ha sido ejecutada previamente por un operador humano. Mediante un dispositivo mecánico, el robot memoriza la secuencia que realiza el operador. Estos robots disponen de un control en bucle cerrado y sensores, por lo que su comportamiento puede adaptarse a los cambios del entorno. [9] Aparecen en los años 80. [7]

- Tercera generación: Robots con control sensorizado.

En la tercera generación, el controlador es una computadora que ejecuta las órdenes de un programa y las envía al manipulador para que realice las acciones necesarias. Utiliza las computadoras para el control y tiene conocimiento del entorno a través del uso de sensores, los cuales miden las variables del ambiente y modifican las acciones de control cuando es necesario. Con esta generación se inicia la era de los robots inteligentes y aparecen los primeros lenguajes de programación para escribir los programas de control. [8] Su desarrollo se dio en las décadas de los 80 y los 90. [7]

- Cuarta generación: Robots inteligentes.

Los denominados robots inteligentes operan de manera similar a la tercera generación. La principal novedad es que incluyen sensores que envían información a la computadora de control sobre el estado del proceso. Además de utilizar sensores más sofisticados, hacen uso de métodos de análisis y obtención de datos con el fin de mejorar el comportamiento en tiempo real. Se puede decir por tanto, que la cuarta generación de los robots es capaz de comprender su entorno y actuar según las señales que recibe del mismo. [10] Aparecen a finales de los 90. [7]

- Quinta generación: Inteligencia artificial.

La quinta generación es la que se encuentra actualmente en desarrollo. Utilizan una nueva arquitectura basada en la organización y distribución de módulos de conducta. [8]

Figura 10. Robot Asimo de Honda

Fuente: Honda. http://asimo.honda.com/gallery.aspx

A modo de resumen, se presenta a continuación un eje cronológico con los principales hitos del desarrollo de la robótica desde los años 50 hasta hoy.

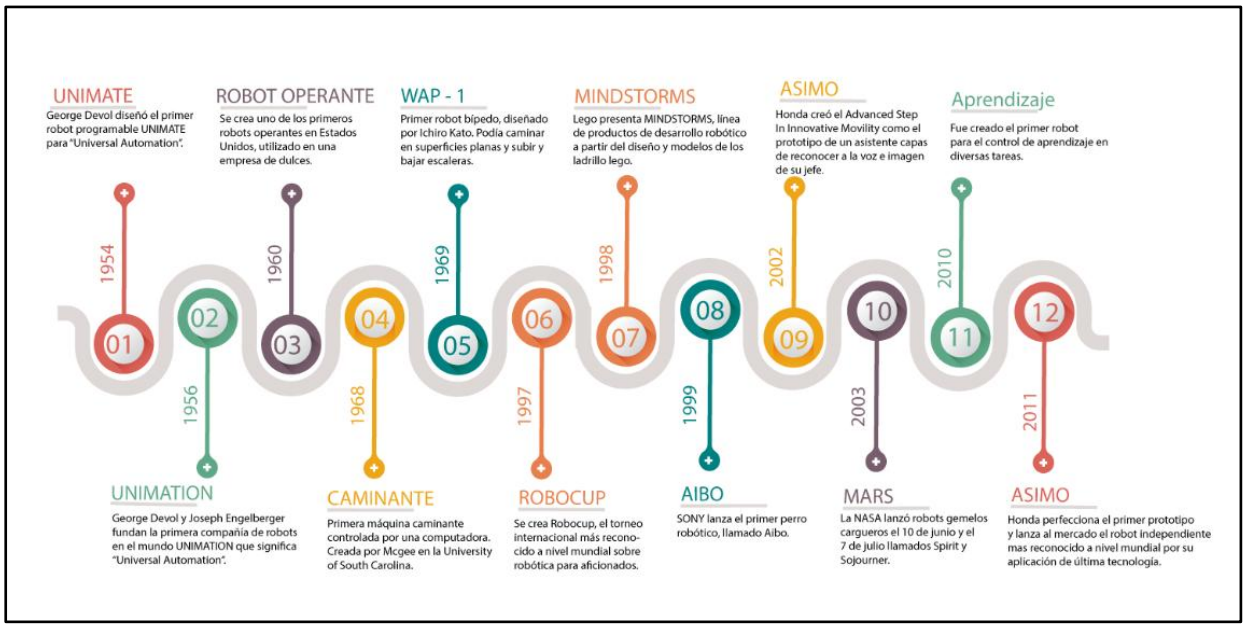


Figura 11. Cronología de la robótica

Fuente: ROBOCOL, http://robocol.co/robotica-en-el-mundo/

En los últimos años, los robots han tomado posición en casi todas las áreas productivas industriales ya que su desarrollo ha crecido de forma exponencial. En poco más de cuarenta años, se ha pasado de aquellos primeros modelos, simples y limitados, a sofisticadas máquinas capaces de sustituir al hombre en todo tipo de tareas. Pero esto no es todo, los robots consiguen hacerlo de forma más rápida, precisa y económica que el ser humano.

2.2 Definición y clasificación de los robots

En 1979, el "Robot Institute of America" definía robot como: "Un manipulador reprogramable y multifuncional diseñado para trasladar materiales, piezas, herramientas o aparatos específicos a través de una serie de movimientos programados para llevar a cabo una variedad de tareas". Veinte años más tarde, la enciclopedia Encarta, de Microsoft, daba la siguiente definición: "Máquina controlada por ordenador y programada para moverse, manipular objetos y realizar trabajos a la vez que interacciona con su entorno" [11]. En la actualidad, la RAE propone la siguiente definición: "Máquina o ingenio electrónico programable, capaz de manipular objetos y realizar operaciones antes reservadas solo a las personas" [12].

Debido a la diversidad de los robots hoy en día existentes, no es posible realizar una clasificación única que incluya a todos y cada uno de ellos. Por lo tanto, nos centraremos en dos grupos claramente diferenciados según la arquitectura de los robots: fijos y móviles. Para una información más extensa de las diferentes tipologías existentes se puede visitar la página web "wikirobotica". [5]

- Robots fijos: son aquellos que no pueden moverse en el espacio debido a que tienen fija su base. Principalmente suelen ser brazos o garras, aunque hay más clases, como por ejemplo los robots quirúrgicos. Están muy presentes en la industria y poseen articulaciones para poder desempeñar sus funciones.

- Robots móviles: esta clase de robots puede moverse en el espacio ya sea por un sistema de tracción o por una morfología zoomórfica. Dentro de los de robots zoomórficos destacamos los robots humanoides, aunque también los hay con morfología animal.

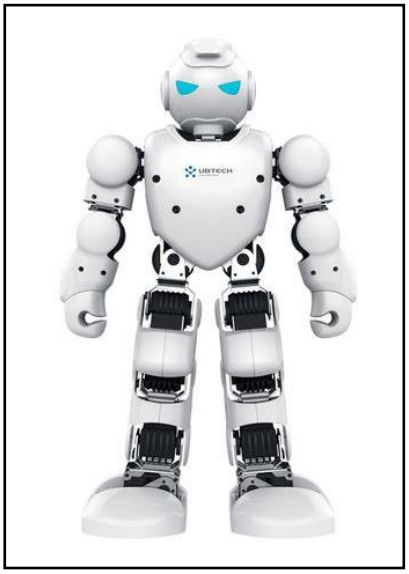


Figura 13. Robot humanoide

Fuente: Macnificos. https://www.macnificos.com/ubtech-alpha-1pro-robot-humanoide

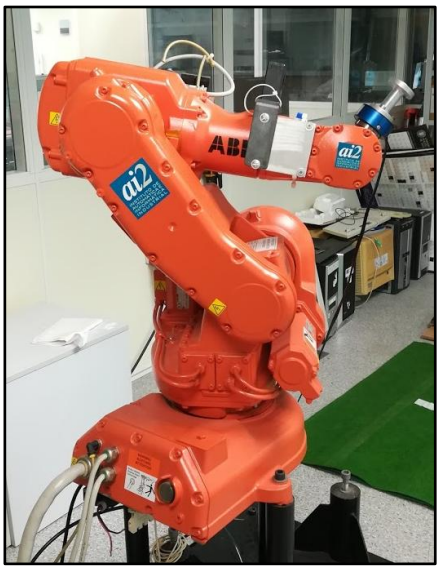


Figura 12. Robot fijo ABB

2.2.1 Robots móviles

Dado que el objetivo principal de este proyecto es implementar los nuevos algoritmos de control en un robot móvil con tracción diferencial, haremos un repaso de las principales configuraciones de los robots móviles que existen. [13]

- Configuración diferencial.

Es considerada como la configuración más sencilla de diseñar e implementar. Sin embargo, a veces puede resultar muy complicada de controlar debido a los deslizamientos o derrapes que suelen darse y a la buena sincronización de las ruedas requerida para describir una trayectoria rectilínea. Los robots diferenciales disponen de dos ruedas motrices situadas en la parte trasera y el centro de rotación está en el medio de este eje de tracción. Además, suelen tener una o dos ruedas locas en la parte delantera, estas sirven únicamente como punto de apoyo.

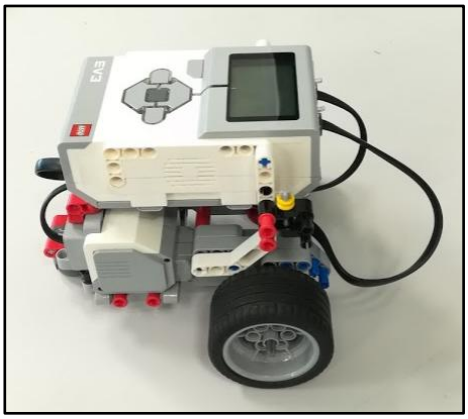


Figura 14. Configuración diferencial

- Configuración de oruga.

Un caso particular interesante de los robots diferenciales son los que tienen la configuración oruga. En lugar de tener dos ruedas tienen cadenas, y su modelo cinemático equivale a calcular unas ruedas equivalentes en el centro de las cadenas. El eje de rotación está por tanto en el centro del eje que una estas ruedas ficticias. Suelen carecer de las ruedas locas y aportan un mayor agarre dado a su mejor sistema de tracción. Esta configuración es muy útil en usos militares.

Figura 15. Configuración oruga

Fuente: Cafago. https://www.cafago.com/es/p-rm5076us.html

- Configuración triciclo.

En esta configuración se dispone de nuevo de dos ruedas traseras de tracción. Sin embargo, los giros no se controlan con la diferencia de velocidades entre ambas ruedas motrices, sino con la rueda delantera, que es orientable. Respecto a los robots diferenciales, tiene una mejor adherencia.

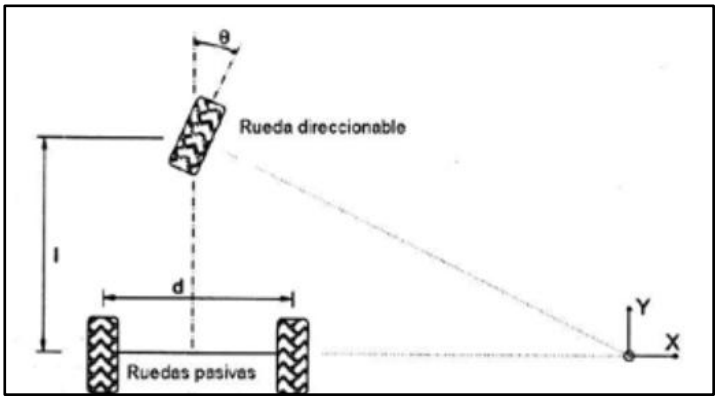


Figura 16. Configuración triciclo

Fuente: Mucho trasto. http://www.muchotrasto.com/TiposDePlataformas.php

- Configuración Ackerman.

Se trata de la configuración que tienen los automóviles con 4 ruedas. Respecto a los triciclos, en esta configuración son dos las ruedas orientables. Obteniendo de nuevo una rueda ficticia, esta vez entre las dos ruedas direccionales, su modelo cinemático coincide con el de los robots con configuración de triciclo.

Figura 17. Configuración Ackerman

Fuente: Robotnik, RB-Car. https://www.robotnik.es/robots-moviles/rbcar/

2.3 Estrategia de control

Con el fin de implementar el control de la trayectoria para robots móviles diferenciales se ha propuesto un control de dos niveles bien diferenciados: el control cinemático y el control dinámico.

Tal y como se ve en la figura, el control se corresponde con un bucle en cascada donde el control cinemático representa el bucle exterior y el dinámico el interior. [14]

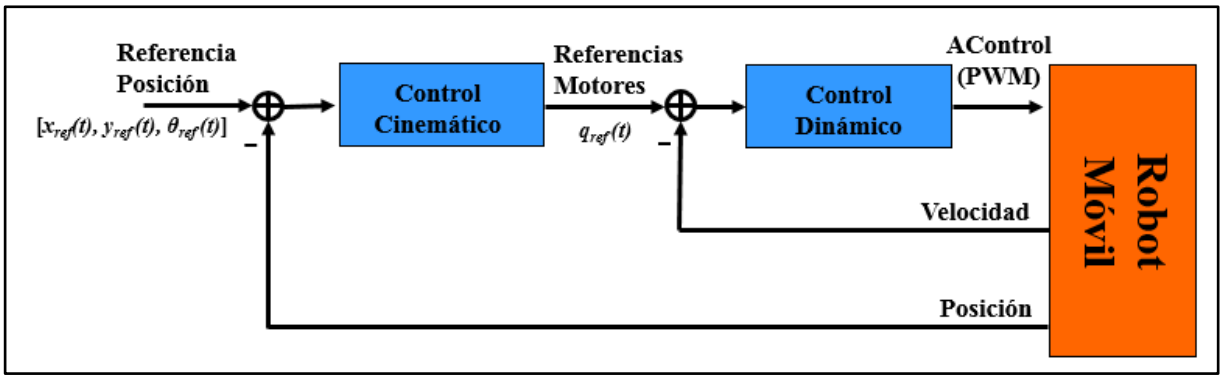


Figura 18. Estrategia de controla de control

Fuente: Ángel Valera. LAC. Diapositivas de clase.

2.3.1 Control cinemático

El control cinemático consiste en determinar las acciones del robot necesarias para llevarlo desde su posición actual a una posición final deseada, considerando las velocidades y la orientación del mismo. En general es un control más complejo que el dinámico.

Dentro del control cinemático se pueden seguir dos estrategias: el control de trayectoria y el control de camino. En el primero, se genera una curva temporal de cada una de las coordenadas del robot, por lo que el tiempo es un factor a tener en cuenta. En la segunda, se genera una curva en el espacio cartesiano, sin tener en cuenta el factor tiempo. Como se ha dicho, en el presente TFG se busca afrontar el problema del control de trayectoria. Para ello, se utilizarán dos estrategias: los métodos numéricos (objeto de este TFG) y el guiado por punto descentralizado. Como ambos serán explicados con posterioridad, ahora se describirá en sí mismo el control cinemático.

Como se ha explicado, el control cinemático será el encargado de generar las referencias para ajustarse mejor a las especificaciones de movimiento deseadas. Para ello, se compondrá de dos bloques claramente diferenciados: el control cinemático en sí mismo, y el modelo cinemático inverso.

- Bloque de control: es el primero de los dos bloques, y tiene como objetivo transformar la trayectoria que recibe en las velocidades del robot $\begin{bmatrix} \dot{x_p} \\ \dot{y_p} \end{bmatrix}$ ó $\begin{bmatrix} v \\ \omega \end{bmatrix}$ según la estrategia de control a utilizar. Este es el bloque más complicado de diseñar y es el que contiene los algoritmos de control en sí mismos.

- Bloque de cinemática inversa: este segundo bloque recibe el dato que ha generado el algoritmo de control y lo transforma en las velocidades lineales de referencia de cada una de

las ruedas. Al tratarse de un robot diferencial, estos datos serán de vital importancia ya que según la velocidad que lleven las ruedas, el robot realizará una trayectoria u otra.

En resumen, nos quedarán los siguientes diagramas:

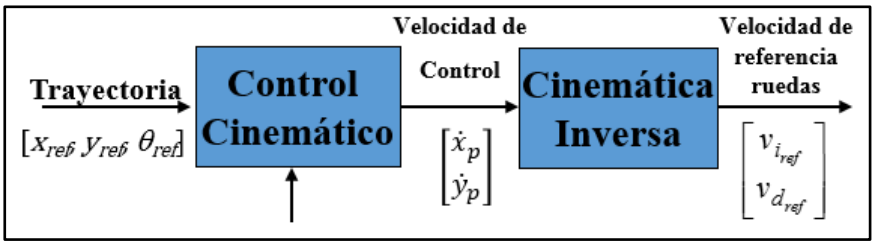


Figura 19. Control cinemático para guiado por punto descentralizado

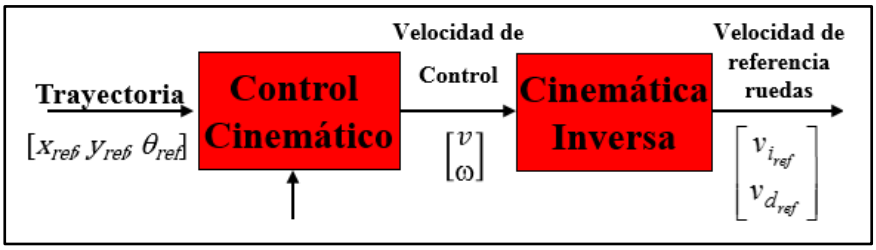


Figura 20. Control cinemático para guiado con métodos numéricos

Como vemos, la estructura de la estrategia de control es similar en ambos casos. En cambio, la programación de los bloques será muy diferente. Esto se explicará con posterioridad cuando se trate el control mediante métodos numéricos y con punto descentralizado.

2.3.2 Control dinámico

El controlador dinámico procura que la respuesta del robot sea lo más parecida posible a la referencia propuesta por el control cinemático. Por lo general es un controlador básico (tipo PID) de velocidad angular y/o de posición angular de los motores del robot. En nuestro caso, utilizaremos un control proporcional con una ganancia k=9 para la velocidad angular de los motores. La obtención de este valor está fuera del TFG ya que fue objeto de experimentos y simulaciones anteriores.

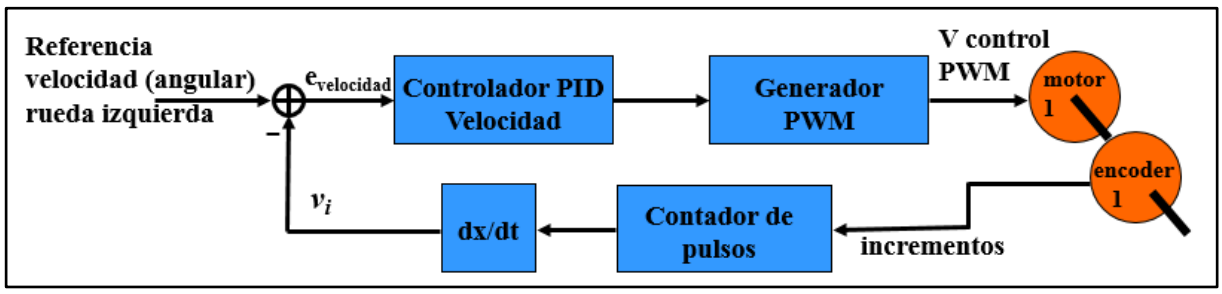


Figura 21. Controlador dinámico

Fuente: Ángel Valera. LAC. Diapositivas de clase.

2.3.3 Robot móvil

En este último apartado se describirá el último bloque que encontramos en el bucle de control: el robot móvil o cinemática directa del robot. Las entradas de este bloque son las acciones de control generadas por el control dinámico, es decir, las velocidades angulares de cada una de las ruedas. Sus salidas serán las componentes $[x, y, \theta]^t$, es decir, la trayectoria real que sigue el robot. Dicha trayectoria será la realimentación negativa del bucle de control y nos servirá para calcular las nuevas velocidades de control. Cabe mencionar que dicho bloque solamente será programado a la hora de llevar a cabo simulaciones.

Para deducir las ecuaciones que lo componen, partimos del modelo cinemático de los robots diferenciales:

$$\begin{cases} V = \dfrac{v_d + v_i}{2} \\ \omega = \dfrac{v_d - v_i}{2b} \end{cases}$$ ECUACIÓN 1

Siendo "b" la mitad de la separación entre las ruedas.

Como se sabe, las entradas serán las velocidades angulares de cada una de las ruedas, por lo que si se multiplican por el radio de la rueda, se obtienen "v_d" y "v_i". Conocidas las velocidades lineales de las ruedas y "b", ya se pueden obtener "V" y "ω".

Una vez halladas, se obtiene la orientación "θ" integrando "ω". Una vez conocida, se calcula el seno y el coseno y se multiplica por la velocidad lineal, quedando "v_y", "v_x". Bastará con integrar estas velocidades para obtener las posiciones reales en los ejes de coordenadas.

2.4 Guiado por Métodos Numéricos

En este capítulo se propondrá el desarrollo de controladores cinemáticos de trayectorias para robots móviles. Los controladores que se proponen están basados en métodos numéricos de álgebra lineal. Se trata de un conjunto de técnicas novedosas muy prometedoras, ya que permiten abordar problemas complejos con un coste computacional reducido. En resumen, se tratarán de proponer unos nuevos algoritmos de control, sencillos, versátiles, y con un comportamiento estable.

2.4.1 Métodos numéricos

En este apartado se describirán brevemente los métodos matemáticos que se utilizarán para diseñar y desarrollar los nuevos algoritmos de control y para probar la convergencia a cero del error cuadrático.

En el campo de las matemáticas, se conoce la interpolación como el proceso mediante el cual se estima el valor de un punto de una función siendo conocidos otros dos puntos. El caso más sencillo es la interpolación lineal, en la cual se aproxima la función f(x) mediante una recta. Para ello, se debe trazar una recta que pase por los puntos (x1, y1) y (x2, y2).

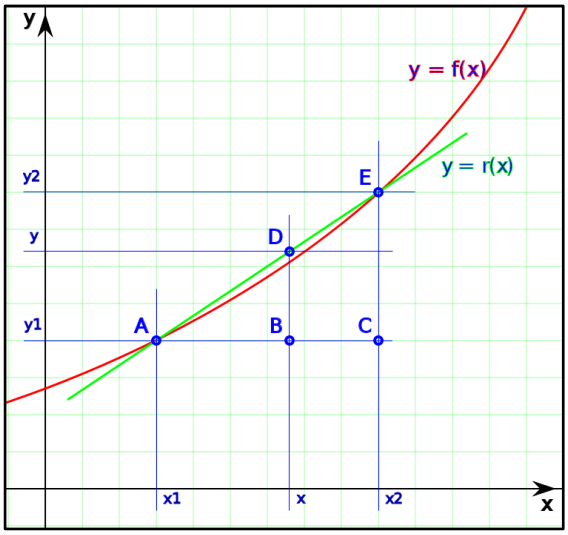


Figura 22. Interpolación lineal

Fuente: Interpolación lineal. Wikipedia.
https://es.wikipedia.org/wiki/Interpolaci%C3%B3n_lineal

Utilizando la ecuación punto pendiente de la recta

$$(\mathrm{y}-\mathrm{y1})=\mathrm{m}*(\mathrm{x}-\mathrm{x1})$$ ECUACIÓN 2

Y sabiendo que la pendiente m viene dada por

$$m=\frac{\Delta y}{\Delta x}=\frac{y2-y1}{x2-x1}$$ ECUACIÓN 3

Quedará la siguiente fórmula:

$$y=\frac{(y2-y1)}{(x2-x1)}\,(x-x1)+y1$$ ECUACIÓN 4

En segundo lugar se utilizan métodos de integración numéricos, en particular, la integración por el método del trapecio. Dicho método consiste en aproximar el área bajo la curva a un trapecio tal y como se ve en la siguiente figura.

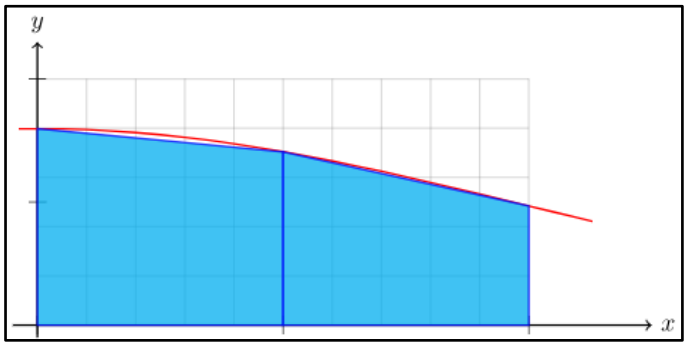


Figura 23. Integral por trapecios

Fuente: Aprender matemáticas, integral aproximada mediante la regla del trapecio.
https://www.aprendematematicas.org.mx/unit/integracion-aproximada-regla-del-trapecio/

En conclusión, la regla del trapecio es equivalente a realizar la integral del polinomio interpolador de grado 1, P(x).

$$I = \int_a^b f(x)\,dx \approx \int_a^b P(x)\,dx \approx \int_a^b \left[\frac{f(b)-f(a)}{b-a}(x-a) + f(a)\right]dx$$ ECUACIÓN 5

Por tanto:

$$I \approx (b-a)\frac{f(a)+f(b)}{2}$$ ECUACIÓN 6

Una vez entendido el método de integración trapezoidal, puede exponerse el método de Euler para la resolución de ecuaciones diferenciales ordinarias (EDO). Se hará uso de dicho método para deducir las ecuaciones del controlador. [15]

Sea una EDO de primer orden tal que:

$$\dot{y} = f(t,y) \qquad y(t_0) = y_0$$ ECUACIÓN 7

Y siendo conocido el valor de la solución en el instante inicial:

$$\dot{y}(t_0) = f(t_0\,, y_0)$$ ECUACIÓN 8

Con estas dos expresiones podemos escribir la ecuación de la recta tangente a la solución en el instante $t=t_0$.

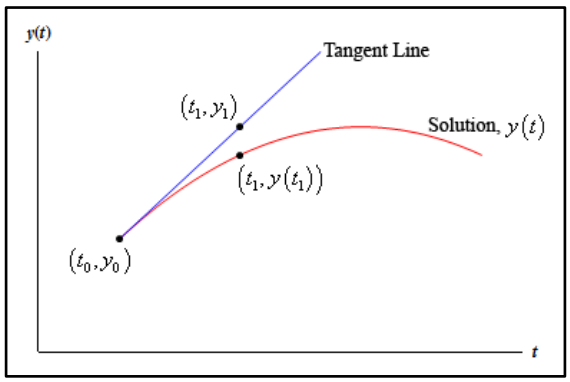


Figura 24. Método de Euler

Fuente: Paul's online math notes.
http://tutorial.math.lamar.edu/Classes/DE/EulersMethod.aspx

$$y = y_0 + f(t_0\,, y_0)(t - t_0)$$ ECUACIÓN 9

Si el t_1 es lo suficientemente cercano a t_0 quedará:

$$y_0 = y_0 + f(t_0\,, y_0)(t_1 - t_0)$$ ECUACIÓN 10

De igual forma se puede generalizar para t_2, t_3, hasta llegar a t_n. Ya se puede enunciar el método de Euler según la siguiente expresión:

$$y_{n+1} = y_n + f(t_n, y_n)(t_{n+1} - t_n)$$ ECUACIÓN 11

Por último, se comentan brevemente la definición y el uso de las series de Taylor. Una serie de Taylor es una aproximación de funciones mediante una suma de potencias de $(x-a)^n$. Dicha serie se calculará a partir de las derivadas de dicha función en un punto "a". Si el valor de "a" es 0, la serie en particular recibe el nombre de serie de McLaurin. Como se verá con posterioridad, las series de Taylor serán empleadas para demostrar la convergencia a cero del error cuadrático del controlador. Ahora veamos su definición.

Se define la serie de Taylor de una función cualquiera f(x) como:

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}\,(x-a)^n$$ ECUACIÓN 12

O lo que es lo mismo:

$$f(a) + \frac{f'(a)}{1!}\,(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$ ECUACIÓN 13

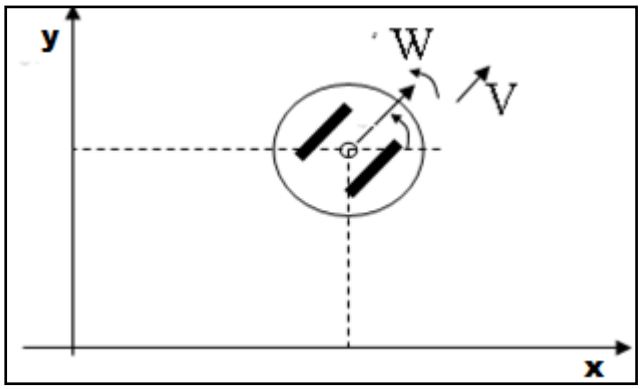
En este caso, se hará uso de la función coseno. Particularizando la serie de Taylor con solamente la primera derivada para la función f(x)= cos(x), queda la siguiente expresión:

$$\cos(x) \approx \cos(a) - \sin(a)\ (x-a)$$ ECUACIÓN 14

2.4.2 Diseño del controlador de trayectoria

En esta sección se diseñará el controlador de la trayectoria. Estos controladores fueron propuestos por Gustavo Scaglia, profesor de la Universidad de San Juan, Argentina y han sido implementados en robots móviles como el Pioneer P3-DX y en procesos de la industria química.

En primer lugar las ecuaciones de las velocidades lineal y angular [16].



$$\begin{cases} \dot{x} = V\cos(\theta) \\ \dot{y} = V\sin(\theta) \\ \dot{\theta} = W \end{cases}$$ ECUACIÓN 15

Figura 25. Robot diferencial. Velocidades.

Una vez conocidas, deben resolverse las ecuaciones diferenciales que lo componen según el método de Euler. (ECUACIÓN 11)

$$\begin{cases} x_{k+1} = x_k + v_k \cos(\theta_k)\,\Delta t \\ y_{k+1} = y_k + v_k \sin(\theta_k)\,\Delta t \\ \theta_{k+1} = \theta_k + \omega_k\,\Delta t \end{cases}$$ ECUACIÓN 16

Ahora el objetivo será calcular el valor de las acciones de control v_k y ω_k. Podemos reescribir las dos primeras ecuaciones del sistema anteriormente planteado con el fin de obtener la velocidad lineal de la siguiente forma:

$$\begin{bmatrix} \cos(\theta_k) \\ \sin(\theta_k) \end{bmatrix} v_k = \frac{1}{\Delta t}\begin{bmatrix} x_{k+1} - x_k \\ y_{k+1} - y_k \end{bmatrix}$$ ECUACIÓN 17

Considerando que la trayectoria seguida por el robot la es conocida; y es la diferencia entre la coordenada deseada en el instante posterior y el error multiplicado por una acción de control k [17], que como se expondrá con posterioridad debe estar contenida entre 0 y 1. Para la coordenada "x" sería:

$$\Omega_x = x_{ref,k+1} - k_x(x_{ref,k} - x_k)$$ ECUACIÓN 18

Operando de forma similar para la coordenada "y" y sustituyendo en la ECUACIÓN **17** se obtiene la siguiente formulación:

$$\begin{bmatrix} \cos(\theta_k) \\ \sin(\theta_k) \end{bmatrix} v_k = \frac{1}{\Delta t}\begin{bmatrix} x_{ref,k+1} - k_x(x_{ref,k} - x_k) - x_k \\ y_{ref,k+1} - k_y(y_{ref,k} - y_k) - y_k \end{bmatrix} = \frac{1}{\Delta t}\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$ ECUACIÓN 19

Por lo tanto para cumplir la ecuación se tiene que cumplir la siguiente relación de proporcionalidad:

$$\frac{\sin(\theta_k)}{\cos(\theta_k)} = \frac{\Delta y}{\Delta x} = \frac{y_{ref,k+1} - k_y(y_{ref,k} - y_k) - y_k}{x_{ref,k+1} - k_x(x_{ref,k} - x_k) - x_k}$$ ECUACIÓN 20

Por tanto el valor de la orientación de referencia "$\theta_{ez,k+1}$" será:

$$\theta_{ez,k+1} = \operatorname{atan}\left(\frac{y_{ref,k+1} - k_y(y_{ref,k} - y_k) - y_k}{y_{ref,k+1} - k_y(y_{ref,k} - y_k) - y_k}\right) = atan\left(\frac{\Delta y}{\Delta x}\right)$$ ECUACIÓN 21

Para despejar el valor de "v" se pasa al otro lado de la ECUACIÓN **19** la traspuesta de la matriz 2x1 que lo premultiplica.

$$v_k = \frac{1}{\Delta t}\begin{bmatrix} \cos(\theta_k) \\ \sin(\theta_k) \end{bmatrix}^t \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$ ECUACIÓN 22

Al realizar dicha operación queda:

$$v_k = \frac{1}{\Delta t}[\Delta x \cos(\theta_k) + \Delta y \sin(\theta_k)\,]$$ ECUACIÓN 23

Una vez calculada la velocidad lineal, debe calcularse la velocidad angular. Para ello, se despeja su valor de la ECUACIÓN 16 y se realiza el mismo razonamiento que en la ECUACIÓN **18**, donde se emplea la componente "x".

$$\Omega_\theta = \theta_{ref,k+1} - k_\theta(\theta_{ref,k} - \theta_k) \qquad \text{ECUACIÓN 24}$$

Quedará por tanto, la siguiente expresión para "ω":

$$\omega_k = \frac{1}{\Delta t}\left[\theta_{ez,k+1} - k_\theta\left(\theta_{ez,k} - \theta_k\right) - \theta_k\right] \qquad \text{ECUACIÓN 25}$$

Particularizando para los valores de la orientación de referencia (θ_{ez}) la ecuación del controlador se escribirá como:

$$\begin{cases} v_k = \dfrac{x_{ref,k+1} - k_x\left(x_{ref,k} - x_k\right) - x_k}{To}\cos\left(\theta_{ez,k}\right) + \dfrac{y_{ref,k+1} - k_y\left(y_{ref,k} - y_k\right) - y_k}{To}\sin\left(\theta_{ez,k}\right) \\ w_k = \dfrac{\theta_{ez,k+1} - k_\theta\left(\theta_{ez,k} - \theta_k\right) - \theta_k}{To} \end{cases} \qquad \text{ECUACIÓN 26}$$

Dónde $\theta_{ez,k+1} = arctan\left(\dfrac{y_{ref,k+1} - k_y\left(y_{ref,k} - y_k\right) - y_k}{x_{ref,k+1} - k_x\left(x_{ref,k} - x_k\right) - x_k}\right)$

2.4.3 Convergencia a cero del error cuadrático

Una vez diseñados los nuevos algoritmos de control, es necesario demostrar la convergencia a cero del error. Demostrar la estabilidad de dichos controles es imprescindible ya que si el error no tendiese a cero, el robot no seguiría las trayectorias de referencia de forma adecuada.

En primer lugar se define el error cuadrático como:

$$e_k = \left[x_{ref,k} - x_k \quad y_{ref,k} - y_k\right] = [\, e_{x,k} \quad e_{y,k}] \qquad \text{ECUACIÓN 27}$$

Y tomando su módulo:

$$\|e_k\| = \sqrt{e_{x,k}^2 + e_{y,k}^2} \qquad \text{ECUACIÓN 28}$$

Una vez expuesto, se parte del modelo cinemático del robot una vez resuelta la ecuación diferencial (ECUACIÓN 16**¡Error! No se encuentra el origen de la referencia.**), a modo de ejemplo se utilizará únicamente la componente "x". La demostración para las otras dos componentes se realizará de forma análoga.

Aplicando la expansión de Taylor al coseno de la ECUACIÓN 14:

$$x_{k+1} = x_k + v_k\,\Delta t\,[\cos\left(\theta_{ez,k}\right) - \sin\left(\theta_{ez,k}\right)(\,\theta_k - \theta_{ez,k})] \qquad \text{ECUACIÓN 29}$$

Esto es:

$$x_{k+1} = x_k + v_k\,\Delta t\cos(\theta_{ez,k}) + v_k\,\Delta t\,\sin(\theta_{ez,k})\,(\theta_{ez,k} - \theta_k) \qquad \text{ECUACIÓN 30}$$

Como se aprecia, el error de la posición angular está presente en esta ecuación, y se puede reescribir de forma más sencilla:

$$x_{k+1} = x_k + v_k\,\Delta t\cos(\theta_{ez,k}) + v_k\,\Delta t\ f\,e_{\theta,k} \qquad \text{ECUACIÓN 31}$$

Como se sabe, la orientación será proporcional a Δx y a Δy de acuerdo a la siguiente expresión.

$$\frac{\Delta y}{\Delta x} = \frac{\sin(\theta_{ez,k})}{\cos(\theta_{ez,k})} \rightarrow \Delta y = \Delta x\frac{\sin(\theta_{ez,k})}{\cos(\theta_{ez,k})} \qquad \text{ECUACIÓN 32}$$

Sustituyendo en la ecuación del controlador, (ECUACIÓN 26) en la que se tiene la expresión de la velocidad lineal quedará tras operar:

$$V_k = \frac{1}{\Delta t}\,\frac{\Delta x}{\cos(\theta_{ez,k})} \qquad \text{ECUACIÓN 33}$$

Sustituyendo esta velocidad en la expresión deducida anteriormente para el valor de la posición (ECUACIÓN 31) se llega a una nueva expresión:

$$x_{k+1} = x_k + \Delta x + v_k\,\Delta t\ f\,e_{\theta,k} \qquad \text{ECUACIÓN 34}$$

Ahora ya se puede sustituir el incremento de "x" de acuerdo a (ECUACIÓN 19).

$$x_{ref,k+1} - x_{k+1} = k_x(x_{ref,k} - x_k) - v_k\,\Delta t\ f\,e_{\theta,k} \qquad \text{ECUACIÓN 35}$$

Una vez hecho, se llega a una ecuación en función de los errores.

$$e_{x,k+1} = k_x e_{x,k} - v_k\,\Delta t\ f\,e_{\theta,k} \qquad \text{ECUACIÓN 36}$$

Siguiendo los mismos pasos pero para la componente "y", se obtiene el siguiente sistema expresado en forma matricial.

$$\begin{bmatrix} e_{x,k+1} \\ e_{y,k+1} \end{bmatrix} = \begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix}\begin{bmatrix} e_{x,k} \\ e_{y,k} \end{bmatrix} + \Delta t\ v_k\begin{bmatrix} f \\ g \end{bmatrix} e_{\theta,k} \qquad \text{ECUACIÓN 37}$$

Una vez obtenido el sistema, se observa que se tiene un término lineal sumado a otro no lineal. Es entonces cuando se tratará la orientación del robot para ver si realmente converge a cero el error del controlador.

Partiendo de nuevo de la resolución de la ecuación diferencial (ECUACIÓN 16**¡Error! No se encuentra el origen de la referencia.**) pero en este caso con la componente "θ" realizaremos la misma operación pero esta vez sustituyendo la velocidad angular "ω_k" sacada de la ecuación del controlador (ECUACIÓN 26). Se llegará de forma inmediata la siguiente ecuación:

$$\theta_{ez,k+1} - \theta_{k+1} = k_\theta(\theta_{ez,k} - \theta_k) \qquad \text{ECUACIÓN 38}$$

Que de nuevo queda solamente en función del error, esta vez, el error de la orientación.

$$e_{\theta,k+1} = k_\theta \, e_{\theta,k} \qquad \text{ECUACIÓN 39}$$

Si se acota la constante k_θ entre 0 y 1 tal y como proponen Scaglia y Chein [17], se observa que conforme "k" va creciendo, el error angular $e_{\theta,k}$ va disminuyendo hasta llegar a cero. Además, será posible controlar dicha disminución modificando el valor de la contante: si su valor es 0.6, el error disminuirá en un 40%, y si por ejemplo se desea que la bajada sea más suave y se toma un valor de 0.9, su valor decrecerá un 10% en cada iteración.

Volviendo al sistema obtenido (ECUACIÓN 37) y sabiendo que el error en la orientación tiende a cero es posible eliminar el término no lineal. Con ello queda un nuevo sistema, en este caso lineal. Siguiendo el mismo razonamiento, se acotan las constantes k_x y k_y entre 0 y 1, de nuevo los errores e_x, e_y convergerán a cero, y con ello el error cuadrático del controlador.

Se puede afirmar por tanto:

$$\|\boldsymbol{e_k}\| \xrightarrow{\boldsymbol{converge}} \mathbf{0} \qquad \boldsymbol{cuando} \qquad \boldsymbol{k} \xrightarrow{\boldsymbol{tiende}} \infty$$

2.4.4 Control cinemático

Como se ha explicado en el apartado 2.3.1 el control cinemático será diferente si se emplea el guiado con métodos numéricos o con punto descentralizado. Por ello se explicará con mayor detalle en este apartado y en el 2.5.1.

Recordemos en primer lugar cuál es el objetivo del control cinemático: generar las referencias para ajustarse mejor a las especificaciones de movimiento deseadas. Por tanto, en este caso utilizaremos el controlador deducido en la ECUACIÓN 26, por lo que generaremos las acciones de control, que son las velocidades lineales y angulares del robot.

$$\begin{cases} v_k = \dfrac{x_{ref,k+1} - k_x\left(x_{ref,k} - x_k\right) - x_k}{To}\cos\left(\theta_{ez,k}\right) + \dfrac{y_{ref,k+1} - k_y\left(y_{ref,k} - y_k\right) - y_k}{To}\sin\left(\theta_{ez,k}\right) \\ w_k = \dfrac{\theta_{ez,k+1} - k_\theta\left(\theta_{ez,k} - \theta_k\right) - \theta_k}{To} \end{cases}$$

Dónde $\theta_{ez,k+1} = arctan\left(\frac{y_{ref,k+1} - k_y\left(y_{ref,k} - y_k\right) - y_k}{x_{ref,k+1} - k_x\left(x_{ref,k} - x_k\right) - x_k}\right)$

2.4.5 Cinemática inversa

La cinemática inversa es la encargada de transformar las señales que ha generado el algoritmo de control y lo transforma en las velocidades lineales de referencia de cada una de las ruedas. Dado que el control cinemático ha proporcionado "v" y "ω" podemos despejar de forma inmediata el sistema de ecuaciones que forma el modelo cinemático del robot diferencial ECUACIÓN 1.

$$\begin{cases} v_d = V + b\omega \\ v_i = V - b\omega \end{cases}$$ ECUACIÓN 40

2.5 Guiado por Punto Descentralizado

El guiado por punto descentralizado es una de las diferentes estrategias existentes que se utilizan para el control de la posición de un robot móvil. Consiste en establecer el control a partir de la posición y de la velocidad de un punto que está separado una distancia "e" del centro del eje de tracción del robot, por lo que se asemeja a una situación de conducción real, en la cual el conductor se fija en un punto adelantado con el fin de anticiparse a los cambios. A continuación se deducen sus ecuaciones. [14]

En primer lugar, se parte de nuevo de la ecuación diferencial del modelo del robot.

$$\begin{cases} \dot{x} = V\cos(\theta) \\ \dot{y} = V\sin(\theta) \\ \dot{\theta} = W \end{cases}$$ ECUACIÓN 41

Expresándolo en forma matricial y sustituyendo el modelo cinemático del robot ECUACIÓN 1 quedará:

$$\begin{bmatrix} \dot{x_m} \\ \dot{y_m} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 1/2 \\ -1/2b & -1/2b \end{bmatrix} \begin{bmatrix} v_d \\ v_i \end{bmatrix}$$ ECUACIÓN 42

Ahora se calcula la posición y la velocidad del punto descentralizado.

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} x_m + e\cos(\theta) \\ y_m + e\sin(\theta) \end{bmatrix}$$ ECUACIÓN 43

Derivando:

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} \dot{x}_m - e\sin(\theta)\dot{\theta} \\ \dot{y}_m + e\cos(\theta)\dot{\theta} \end{bmatrix}$$ ECUACIÓN 44

Y escribiendo en forma matricial:

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} 1 & 0 & -e\sin(\theta) \\ 0 & 1 & e\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{x_m} \\ \dot{y_m} \\ \dot{\theta} \end{bmatrix}$$ ECUACIÓN 45

Ahora ya podemos se puede sustituir la ECUACIÓN 42 en esta expresión que se acaba de obtener.

$$\begin{bmatrix}\dot{x}_p\\ \dot{y}_p\end{bmatrix}=\begin{bmatrix}1 & 0 & -e\sin(\theta)\\ 0 & 1 & e\cos(\theta)\end{bmatrix}\begin{bmatrix}\cos(\theta) & 0\\ \sin(\theta) & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}1/2 & 1/2\\ -1/2b & -1/2b\end{bmatrix}\begin{bmatrix}v_d\\ v_i\end{bmatrix}$$ ECUACIÓN 46

Finalmente se obtiene:

$$\begin{bmatrix}\dot{x}_p\\ \dot{y}_p\end{bmatrix}=\frac{1}{2b}\begin{bmatrix}b\cos(\theta)+e\sin(\theta) & b\cos(\theta)-e\sin(\theta)\\ b\sin(\theta)-e\cos(\theta) & b\sin(\theta)+e\cos(\theta)\end{bmatrix}\begin{bmatrix}v_d\\ v_i\end{bmatrix}$$ ECUACIÓN 47

2.5.1 Control cinemático

Para obtener la ley de control cinemático se parte de la ECUACIÓN 43. La estrategia a seguir será usar cuatro ganancias diferentes: dos para las velocidades de referencia y dos para el error de las posiciones de referencia. [14] Siguiendo dicha metodología se obtiene:

$$\begin{bmatrix}\dot{x}_p\\ \dot{y}_p\end{bmatrix}=\begin{bmatrix}k_{vx}\dot{x}_{ref}\\ k_{vy}\dot{y}_{ref}\end{bmatrix}+\begin{bmatrix}k_{px} & 0\\ 0 & k_{py}\end{bmatrix}\begin{bmatrix}x_{ref}-(x_m+e\cos(\theta))\\ y_{ref}-(y_m+e\sin(\theta))\end{bmatrix}$$ ECUACIÓN 48

2.5.2 Cinemática inversa

Para obtener las ecuaciones de la cinemática inversa solamente se deben despejar las velocidades de las ruedas de la ECUACIÓN 47.

$$\begin{bmatrix}v_d\\ v_i\end{bmatrix}=\frac{1}{e}\begin{bmatrix}e\cos(\theta)+b\sin(\theta) & e\sin(\theta)-b\cos(\theta)\\ e\cos(\theta)-b\sin(\theta) & e\sin(\theta)+b\cos(\theta)\end{bmatrix}\begin{bmatrix}v_d\\ v_i\end{bmatrix}$$ ECUACIÓN 49

2.6 Comparación de las estrategias de control

A modo de resumen podemos realizar una breve comparativa entre las dos estrategias de control de trayectorias anteriormente mencionadas. En primer lugar, una de las diferencias más significativas es que en el control con métodos numéricos no es necesario generar las velocidades de referencia, por lo que el coste computacional es más bajo. Además, el hecho de no tener que calcularlas evita que se produzcan problemas de derivabilidad, como por ejemplo en las esquinas de un cuadrado. La segunda diferencia a tener en cuenta es la diferente cinemática inversa que se programa en el algoritmo de control. Si comparamos la ECUACIÓN **49** y la ECUACIÓN **40** vemos que de nuevo es mucho más simple en el guiado por métodos numéricos. Sin embargo, el guiado por métodos numéricos requiere del cálculo de la orientación del robot y de referencia. Esto es un punto en su contra, y pueden aparecer problemas si no se resuelven los cambios de cuadrante de forma adecuada.

2.7 Curvas de Bézier

En este apartado se realizará una breve explicación de la teoría de las curvas de Bézier, una herramienta muy útil a la hora de generar trayectorias. Además, dado que nuestro controlador se comporta de forma adecuada cuando las trayectorias de referencia tienen la primera derivada continua, [18] la generación de estas curvas permitirá obtener trayectorias sin picos ni discontinuidades, las cuales podrán ser seguidas de forma excelente por el robot.

Las curvas de Bézier fueron creadas en los años 60 por el ingeniero francés Pierre Bézier. Estas fueron utilizadas para numerosos trabajos de dibujo técnico, pero fueron especialmente explotadas por Renault en el diseño de automóviles. [19] El objetivo de estas curvas es interpolar el primer y el último punto de control (P_0 y P_n) mediante una curva que se aproxime al resto de puntos intermedios. Por tanto las curvas de Bézier pasarán por el primer y último punto pero no por el resto de ellos. [20]

La ecuación para cualquier curva con n-1 puntos es la siguiente:

$$r(t) = \sum_{i=0}^{n} P_i \, f_i(t) \qquad t \in [0,1]$$ ECUACIÓN 50

Siendo $f_i(t)$ los polinomios de Berstein:

$$f_i(t) = B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i} = a_i \, t^i (1-t)^{n-i}$$ ECUACIÓN 51

Se debe recordar la expresión:

$$\binom{n}{i} = \frac{n!}{i!\,(n-i)!}$$ ECUACIÓN 52

Claramente, si nuestra curva tiene dos puntos, la curva de Bézier será una recta que los una, por lo que obviaremos este caso. Si se calcula la curva de Bézier 4 puntos se obtiene:

$$r(t) = P_0(1-t)^3 + P_1 3t(1-t)^2 + P_2 3t^2(1-t) + P_3 t^3$$ ECUACIÓN 53

En este proyecto se ha trabajado con curvas de Bézier de 4, 6 y 8 puntos. Para generar dichas curvas, se ha programado un breve script en Matlab con un bucle que va almacenando en dos vectores (x, y) los valores de la curva según las ecuaciones anteriores. Para las curvas de 6 y 8 puntos, los coeficientes (a_i) que multiplican a cada punto se tendrán que calcular según la ECUACIÓN 52.

N=6	a0	a1	a2	a3	a4	a5
Valor	1	5	10	10	5	1

Tabla 1. Bézier con n=6

N=8	a0	a1	a2	a3	a4	a5	a6	a7
Valor	1	7	21	35	35	21	7	1

Tabla 2. Bézier con n=8

n=4	x	y
P0	0	0
P1	2	0
P2	0	2
P3	2	2

Tabla 3. Coordenadas para n=4

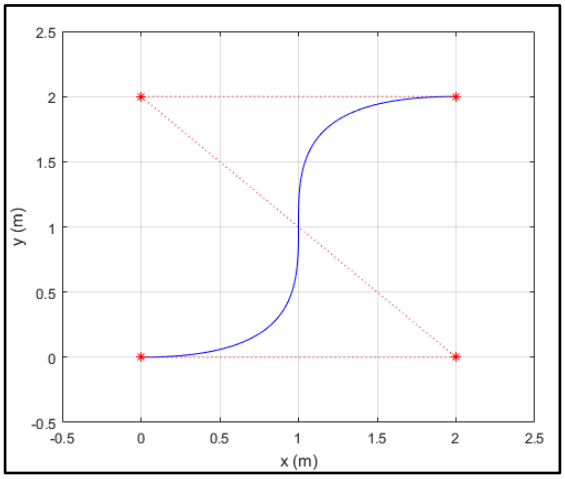


Figura 26. Bézier n=4

n=6	x	y
P0	1	1
P1	1	2.5
P2	1	5
P3	5	5
P4	5	2.5
P5	5	1

Tabla 4. Coordenadas para n=6

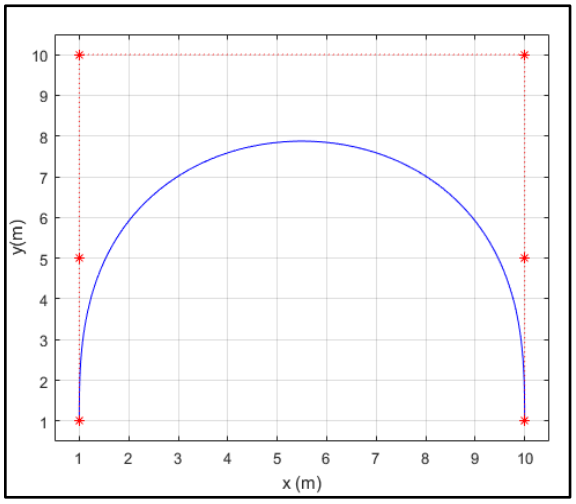


Figura 27. Bézier n=6

n=8	x	y
P0	0	0
P1	0	2
P2	0	3
P3	2	3
P4	3	2
P5	4	0
P6	4	2
P7	4	4

Tabla 5. Coordenadas para n=8

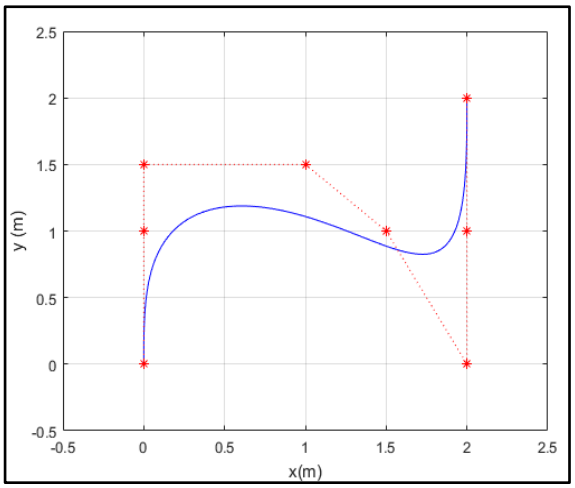


Figura 28. Bézier n=8

3 TRABAJO PRÁCTICO

Una vez conocidos los fundamentos teóricos, es hora de exponer todos los trabajos de índole práctica que se han desarrollado en este proyecto.

3.1 Material y software utilizado

Para el desarrollo de este TFG se han utilizado distintos recursos, todos ellos serán descritos a continuación.

3.1.1 Ordenador portátil

- Marca y modelo: HP 250 G4 Notebook PC
- Sistema operativo: Windows 10.
- Tipo de sistema: 64 bits.
- Procesador: Intel Core I5-62000 CPU @ 2.3GHz 2.7GHz
- Disco duro: 515 GB
- Memoria RAM: 8GB

3.1.2 Ordenador fijo

- Marca y modelo: Gigabyte Technology Co. Ltd. H67MA-USB3-B3
- Sistema operativo: Partición de Windows 7 y Ubuntu 3.13.
- Tipo de sistema: 64 bits.
- Procesador: Intel Core I7-2600 CPU @ 3.40GHz 3.70GHz.
- Disco duro: Dos discos duros de 542 GB y 222 GB. En total 764 GB.
- Memoria RAM: 12 GB

3.1.3 Robot Lego NXT

A continuación se muestra un resumen de las principales características del ladrillo NXT [21], la segunda generación de esta marca tras el bloque RCX. [22]

- Procesador principal: Atmel 32-bit ARM processor, AT91SAM7S256.
  - 256 KB Flash.
  - 65 KB RAM.
  - 48 MHz.
- Co-procesador: Atmel 8-bit AVR processor, ATmega48.
  - 4 KB Flash.
  - 542 Byte RAM.
  - 8 MHz.
- Dispositivos de entrada
  - 4 puertos de entrada.
  - 4 botones.
  - Nivel de tensión de las baterías.
  - Temporizadores internos.

Figura 29. Lego NXT

  - Comunicaciones USB y Bluetooth.
- Dispositivos de salida
  - 3 puertos.
  - Pantalla LCD 100x64 píxeles.
  - Altavoz. Resolución 8 bits, 2-16 KHz.

3.1.4 Robot Lego EV3

A continuación se revisarán de igual modo las principales especificaciones de la tercera generación, la EV3 [23]. Algunas de las diferencias más importantes respecto a la anterior es la comunicación mediante WI-FI o la posibilidad de trabajar con un entorno de Linux.

- Procesador: ARM9.
  - 16 MB Flash.
  - 64 MB RAM. Ampliable con micro SD hasta 32 GB.
  - 300 MHz.
  - SO Linux.
- Dispositivos de entrada
  - 4 puertos de entrada.
  - 4 botones.
  - Comunicaciones USB, Bluetooth y WI-FI.
- Dispositivos de salida
  - 4 puertos.
  - Pantalla monocromática 78x128 píxeles.
  - Altavoz.

Figura 30. Lego EV3

3.1.5 Matlab y Simulink

MATLAB es una herramienta o software matemático que ofrece un entorno de desarrollo integrado incluyendo su propio lenguaje de programación. Se puede instalar esta herramienta en plataformas Unix, Windows y Apple Mac OS X. Es un software muy potente, por lo que es utilizado ampliamente tanto en universidades como en centros de investigación y desarrollo.

Algunas de sus utilidades son: la manipulación de matrices y vectores, la representación de datos y funciones, la implementación de algoritmos o la creación de interfaces de usuario. El paquete MATLAB dispone de herramientas adicionales que expanden sus prestaciones, en este proyecto se hace uso de una de ellas: Simulink, una plataforma de simulación multidominio. [24]

Esta herramienta en particular también es ampliamente utilizada en industrias de todo tipo. Su principal utilidad reside en la simulación de procesos previa implementación para confirmar la viabilidad. El entorno de Simulink es bastante intuitivo, y permite simular casi cualquier tipo de proceso, pero es especialmente interesante para las áreas de robótica, control y automatización industrial.

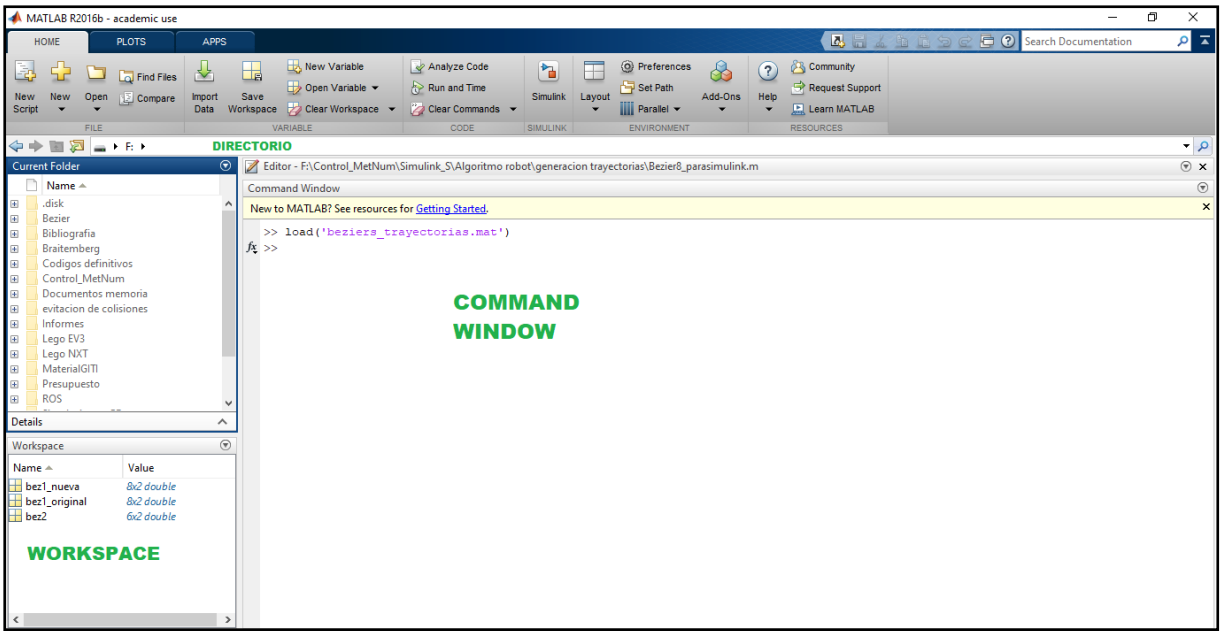


Figura 31. Entorno Matlab

3.1.6 RobotC

Para la programación de los robots se ha utilizado la herramienta RobotC. Se trata de un lenguaje de programación basado en C que contiene gran cantidad de librerías para los robots Lego Mindstorms. Se utiliza en un entorno Windows, y la elección de dicho lenguaje se debe a que supera en prestaciones al resto de alternativas. Para su utilización se requiere de la instalación del firmware del robot. La instalación del mismo y del propio programa se puede encontrar en el 69Anexo I: Descarga de Robot C y del firmware LEGO. [25] A continuación se muestra una imagen del entorno de programación.

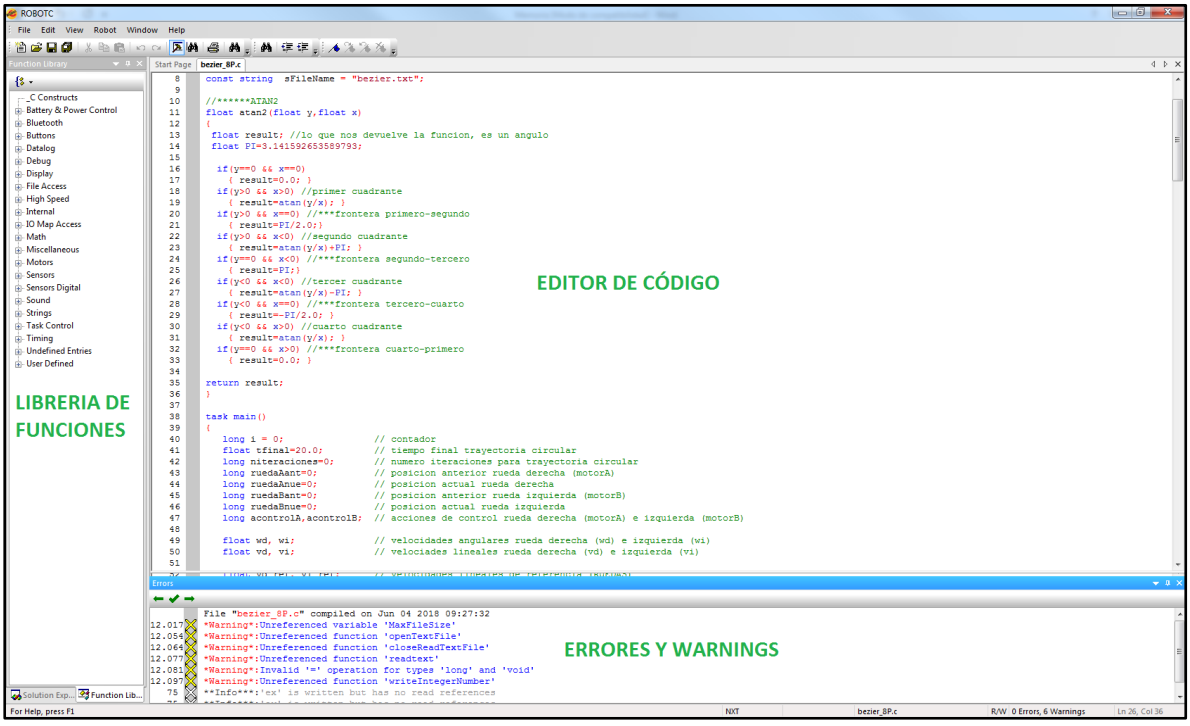


Figura 32. Entorno RobotC

3.2 Simulación del comportamiento del robot

Con el fin de demostrar el correcto funcionamiento de los nuevos algoritmos de control, se han llevado a cabo distintas simulaciones mediante el software "Simulink" de Mathworks. Dichas simulaciones son la primera aproximación de cara a implementar los nuevos controladores en un robot real.

3.2.1 Diagrama de la simulación

La siguiente figura representa la estructura utilizada para llevar a cabo las simulaciones.

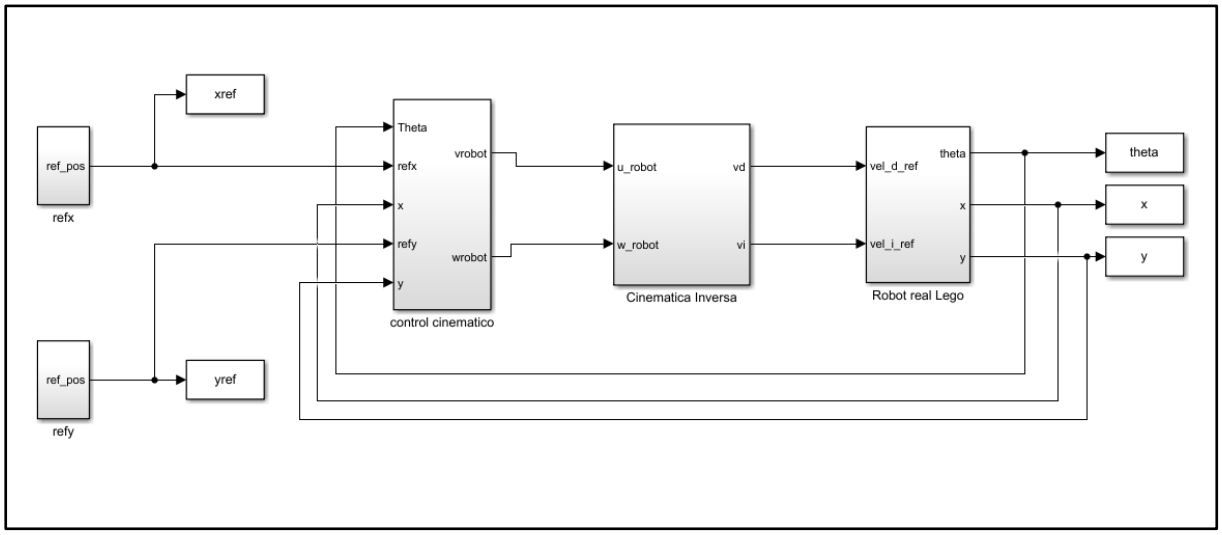


Figura 33. Diagrama de la simulación

Como se aprecia, el esquema tiene una forma similar al bucle de control, en el cual se realimenta el control cinemático con las posiciones en "x" e "y" y la orientación del robot "theta".

Además, se puede apreciar de forma clara que existen cuatro bloques bien diferenciados: generación de referencias, control cinemático, cinemática inversa y robot real. Todos ellos serán descritos con mayor detalle a continuación.

3.2.2 Generación de referencias

Para generar las trayectorias de referencia se han utilizado dos estrategias diferentes: el uso de scripts de Matlab o la generación directa dentro del propio esquema global.

- Uso de scripts de Matlab.

Se han utilizado para generar distintas trayectorias como por ejemplo la cuadrada o la que tiene forma de ocho. Son códigos sencillos, que constan básicamente de un bucle "for" y se deben ejecutar antes de poner en marcha la simulación.

Cada iteración generará un punto de la trayectoria a seguir y cada uno de dichos puntos se calcula con una diferencia temporal de 20 milisegundos, para que así coincida con el periodo de muestreo.

Finalmente se crean los vectores “yyref” y “xxref” los cuales contienen las posiciones y los instantes de tiempo que les corresponden.

A modo de ejemplo se expone el código para generar la trayectoria en ocho.

genera_ref_ocho.m | genera_ref_cuadradov4.m

```
1  clear
2
3  tt=0:0.02:19.98
4
5  k=1;
6
7  for k=1:1000
8
9      xref(k)=sin(2*pi*k/500);
10     yref(k)=cos(k*pi/500);
11
12 end;
13
14 xxref=[tt' xref']
15 yyref=[tt' yref']
```

Figura 34. Script para generación de la trayectoria en ocho

En cuanto al bloque de Simulink en sí mismo, solamente consta de una salida y un “source” que lee del workspace las referencias generadas.

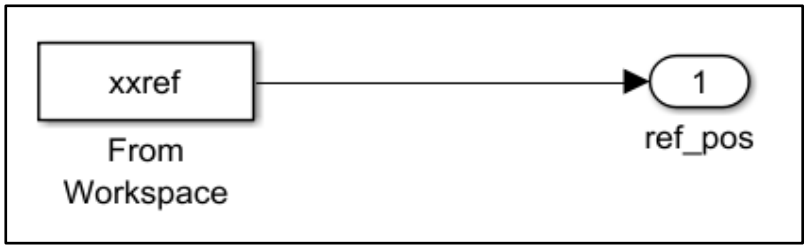


Figura 35. Bloque de generación de referencias

- Generación directa.

La otra estrategia seguida ha sido el generar la trayectoria directamente dentro del mismo bloque las trayectorias. Se ha utilizado principalmente para generar una trayectoria circular.

Lo que se ha hecho es añadir una señal de reloj para cada coordenada (x, y) que indicará cuando calcular el siguiente punto. Luego se multiplica dicho instante por 2*PI*0.06 para que los puntos estén lo suficientemente cercanos y se calcula el seno y el coseno del resultado obtenido. Por último, se multiplica por el radio de la circunferencia deseada.

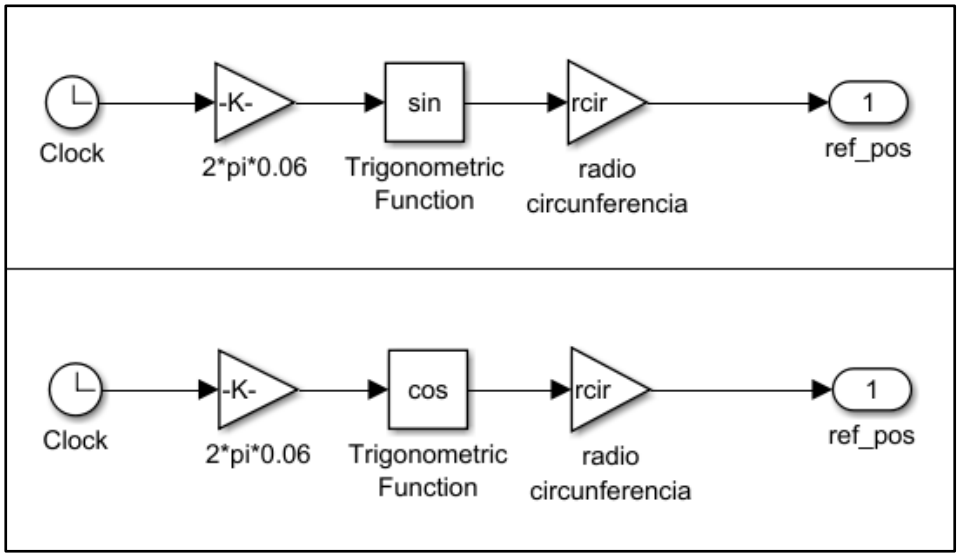


Figura 36. Generación directa de una trayectoria circular

3.2.3 Control cinemático

Una vez generadas las referencias, éstas son enviadas al segundo bloque: el control cinemático. Como se ha explicado, es el bloque más complicado de programar y el que contiene el controlador en sí mismo. Esto significa que calculará las salidas "v" y "ω" a partir de las posiciones y orientaciones reales y de referencia, tal y como se indica en la ECUACIÓN 26. Veamos que contiene dicho bloque.

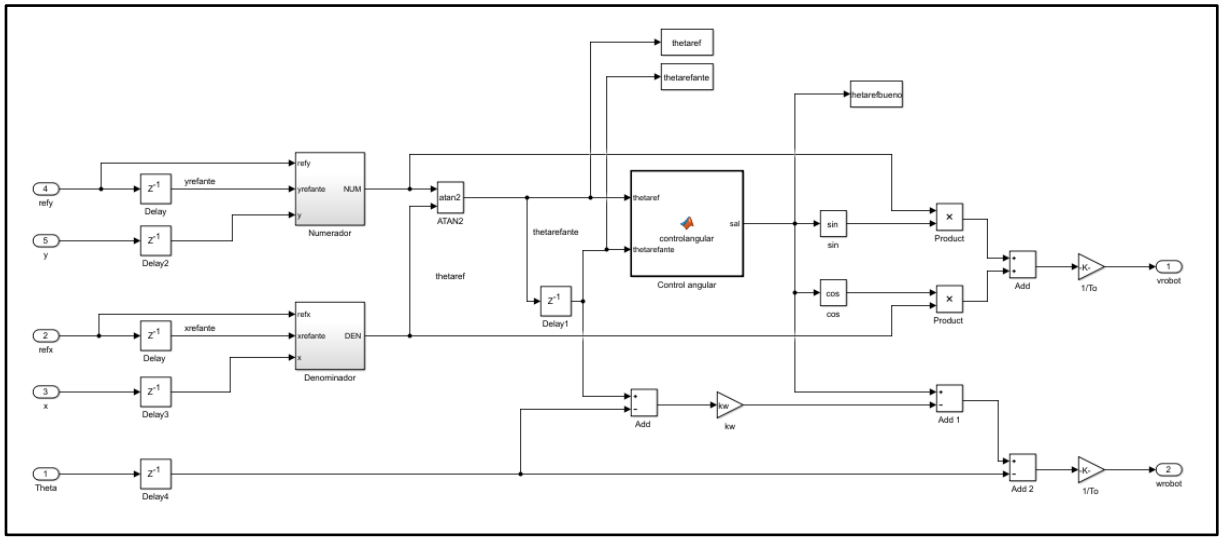


Figura 37. Control cinemático Simulink

En primer lugar se dan como entradas las siguientes variables: xref, yref, x, y, theta. Se retrasan las referencias y se llevan a dos nuevos bloques. Éstos calculan el numerador y el denominador; dos variables auxiliares para obtener la orientación de referencia mediante la operación atan2.

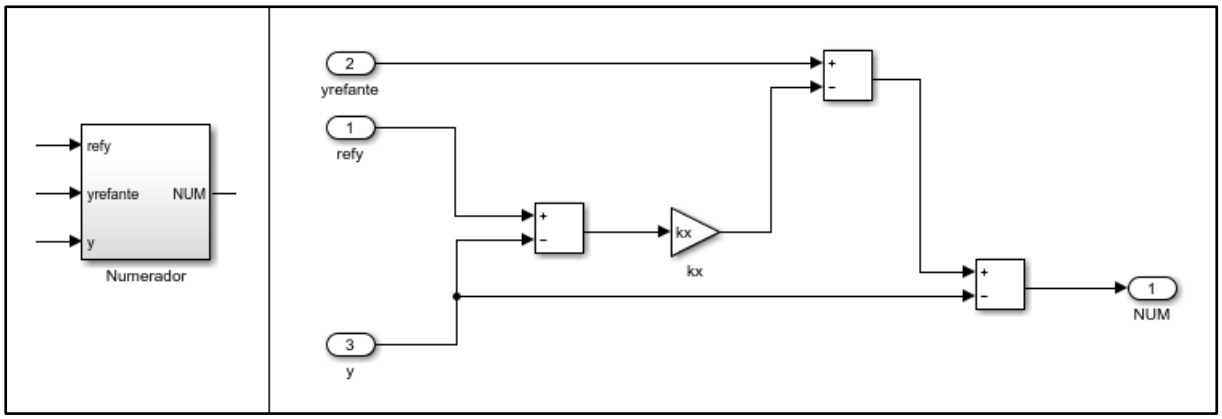


Figura 38. Variable auxiliar: numerador

Una vez obtenidas la orientación de referencia, aparece el principal problema de los controladores: los cambios bruscos de dicha orientación al cambiar de cuadrante. Por ejemplo, al pasar del segundo al tercer cuadrante el valor de "thetaref" será aproximadamente -π debido a que ya ha pasado al otro cuadrante. Sin embargo el valor de "thetarefante" todavía en el segundo cuadrante, estará cercano a π. La resta de ambos valores nos dará un valor cercano a 2π, un valor muy grande si lo comparamos con el del instante anterior. Esto provocará en la simulación un comportamiento inesperado, por lo que se producirán en los cambios de cuadrantes comportamientos indeseados.

Para solucionar dicha problemática se ha hecho es utilizado un bloque que contiene la corrección de la orientación del robot. El código programado en el mismo se muestra a continuación.

control cinematico/Control angular

```
1  function thetaref = controlangular(thetaref,thetarefante)
2
3  sal=thetaref;
4
5   if (thetaref-thetarefante < -5.0)  % cambio 2-3
6      sal=thetaref+2*3.141592653589793;
7   end
8
9   if (thetarefante-thetaref > 5.0) % cambio 4-1
10     sal=thetaref+2*3.141592653589793;
11  end
12
13  if (thetaref-thetarefante > 5.0)  % cambio 3-2
14     sal=thetaref-2*3.141592653589793;
15  end
16
17  if (thetarefante-thetaref < -5.0) % cambio 1-4
18     sal=thetaref-2*3.141592653589793;
19  end
```

Figura 39. Programación del control angular

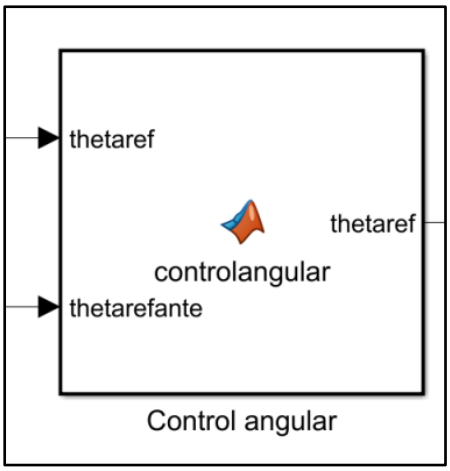


Figura 40. Bloque control angular

En la siguiente figura se aprecian los resultados obtenidos al corregir el valor de la orientación. Como se aprecia, ya no hay saltos bruscos y el comportamiento del robot es el adecuado.

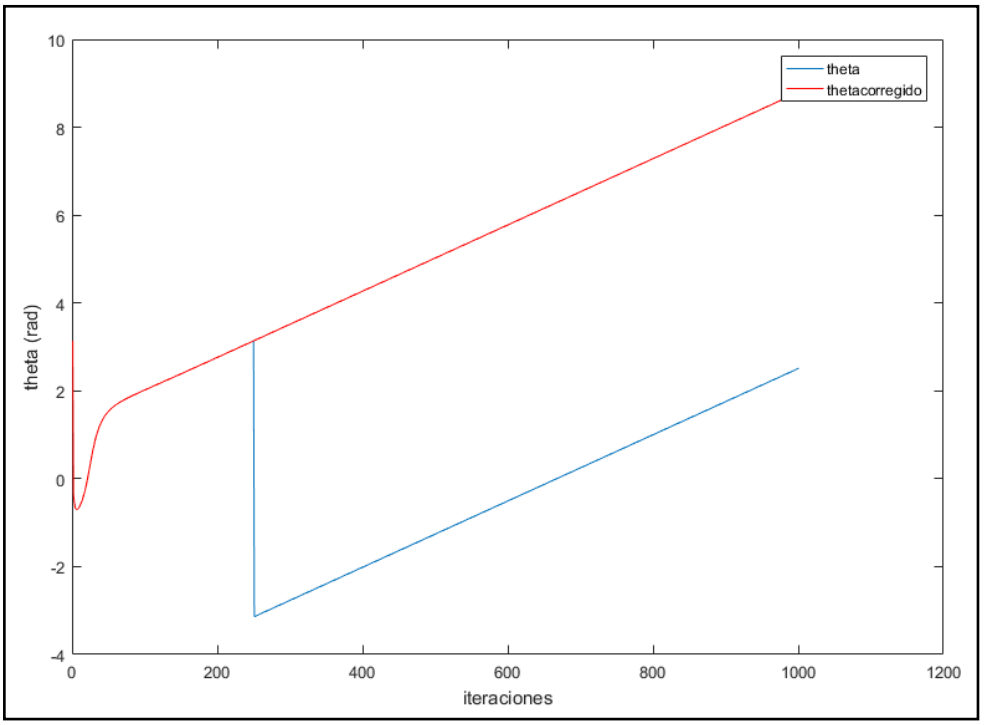


Figura 41. Corrección de la orientación

3.2.4 Cinemática inversa

Este bloque contiene la programación de la cinemática inversa del robot. Se obtendrán las velocidades lineales de las ruedas derecha e izquierda a partir de las acciones de control anteriormente calculadas. La programación de dicho bloque se corresponde a la ECUACIÓN 40.

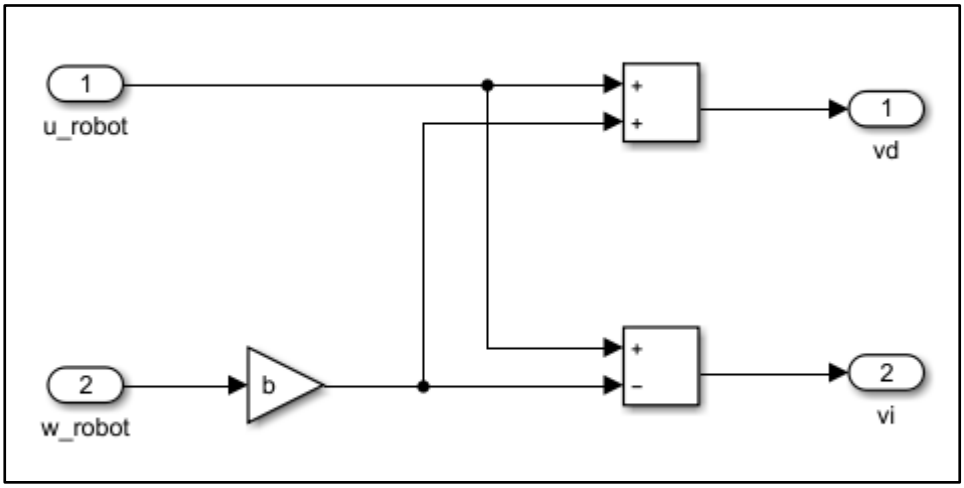


Figura 42. Cinemática inversa

3.2.5 Robot real Lego

En este bloque se obtienen las posiciones reales del robot a partir de las velocidades lineales de las ruedas anteriormente calculadas. Para ello, el primer paso es obtener las velocidades angulares de las ruedas, ya que el control se establece sobre dicha variable. Para ello se multiplica la velocidad lineal por el radio de la rueda.

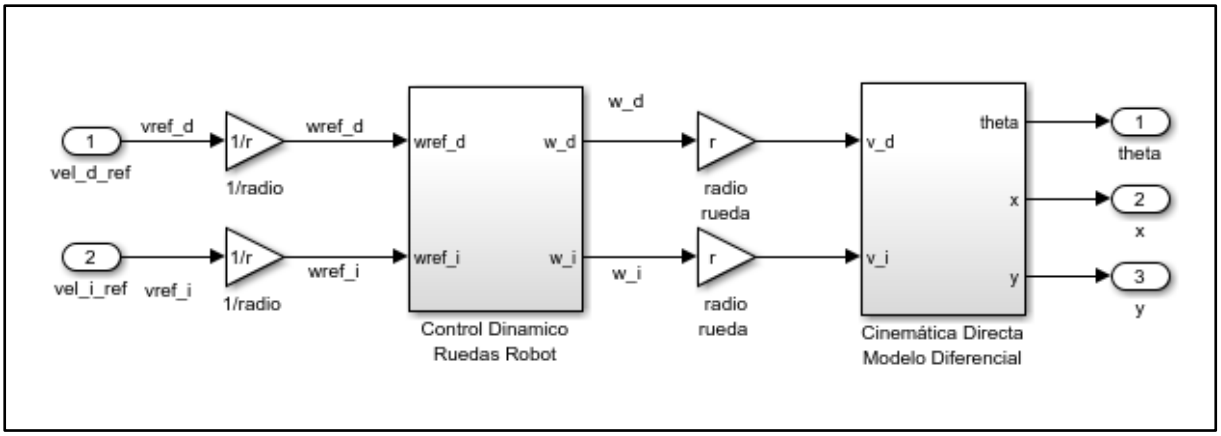


Figura 43. Robot Real Lego

- Control dinámico.

Una vez calculadas las velocidades angulares, se entra en el control dinámico propiamente dicho. Como se sabe, dicho control será del tipo PID, y en particular, un controlador proporcional de ganancia 9. Como vemos en la figura, se trata de un bucle cerrado bastante simple en el cual se coloca el controlador y la función de transferencia del motor. La realimentación debe ser negativa, y se utilizará para calcular la acción de control. La función de transferencia del motor se obtuvo mediante el método de la respuesta ante escalón y fue un dato proporcionado por experimentos previos al TFG.

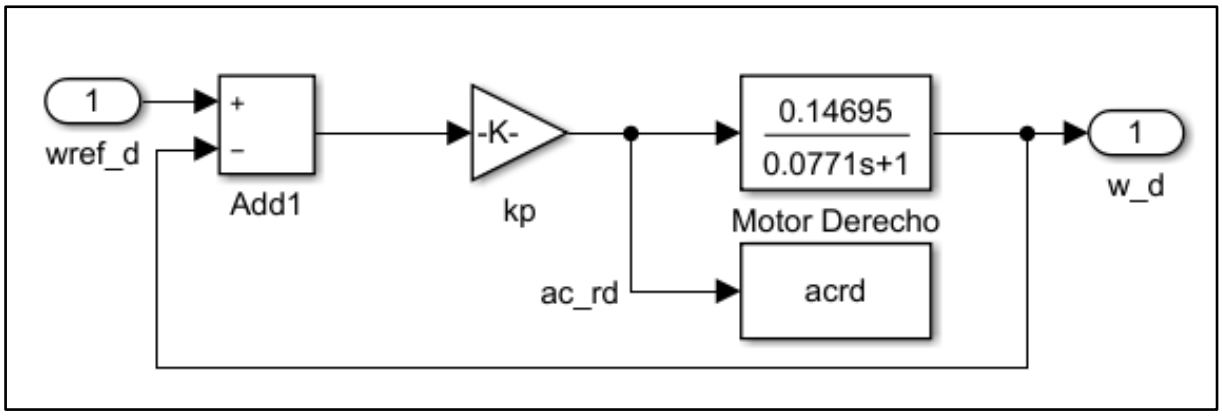


Figura 44. Control dinámico. Bucle de control

- Modelo cinemático directo.

Una vez calculada la velocidad angular real que debe llevar cada rueda se entra en el último bloque: la cinemática directa. En este se obtiene a partir de las velocidades reales de las ruedas del robot la posición y la orientación del mismo. Dichos datos servirán para realimentar el bucle principal de control y calcular las futuras acciones de control. La programación de este bloque es inmediata a partir del modelo cinemático del robot (ECUACIÓN 1). Una vez obtenida la velocidad angular, se utilizará un integrador para obtener la orientación. Con la velocidad lineal se realizará un proceso similar, solo que en este caso habrá que multiplicar por el coseno y el seno de la orientación para descomponer las referencias según los ejes “x” e “y”.

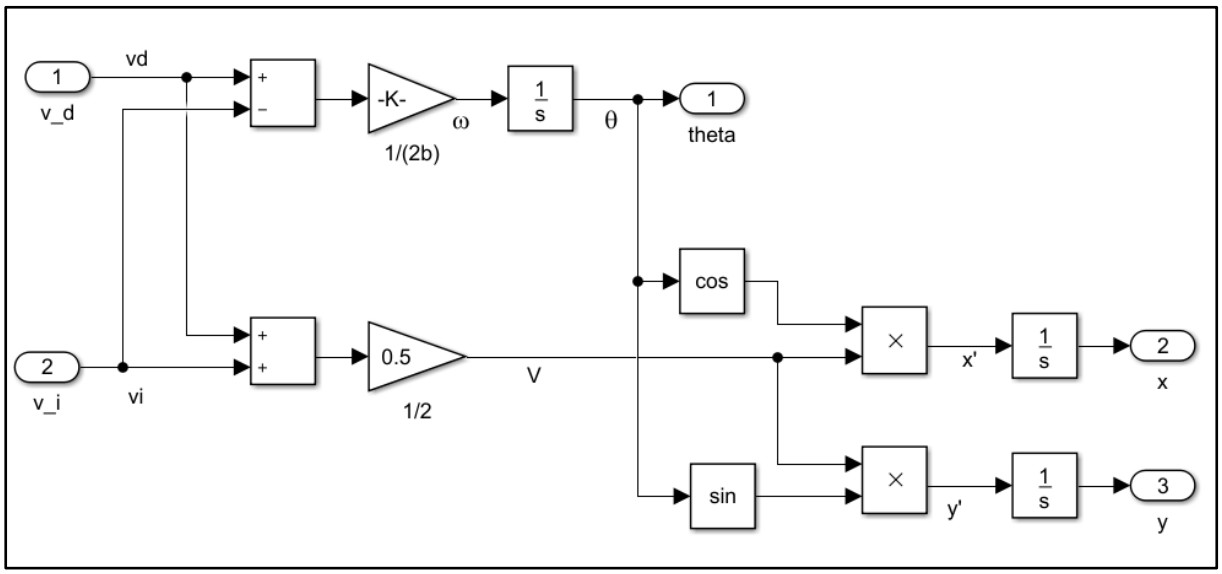


Figura 45. Cinemática directa

3.2.6 Resultados obtenidos en la simulación

En este apartado se mostrarán los distintos resultados obtenidos al simular el comportamiento del robot con distintas trayectorias de referencia. Pero en primer lugar se presentan los valores de los parámetros de los controladores con los que mejor resultado se ha obtenido.

Parámetro	Valor	Comentarios
b	0.056	Mitad de la distancia entre las ruedas (m.)
r	0.028	Radio de las ruedas (m.)
kmp	9	Ganancia del control proporcional
kx	0.94	Constante que pondera el error en x
ky	0.94	Constante que pondera el error en y
kw	0.8	Constante que pondera el error en la orientación
To	0.02	Periodo de muestreo (s.)

Tabla 6. Parámetros de la simulación

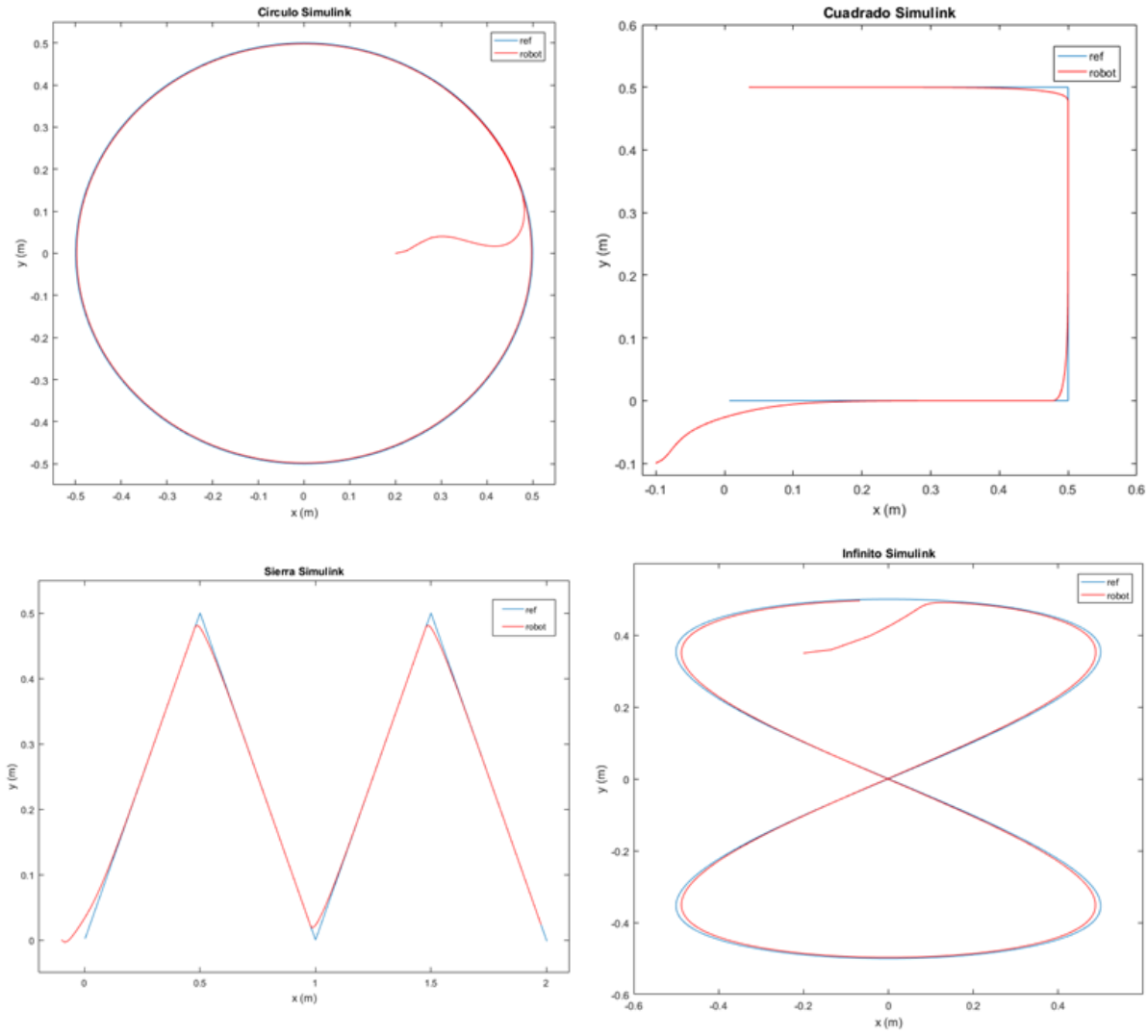


Figura 46. Resultados de la simulación

Una vez vistas las figuras básicas, se pasará a presentar los resultados obtenidos al generar las trayectorias con las curvas de Bézier. Estas trayectorias son más interesantes que las figuras anteriormente vistas ya que permitirán que el robot describa trayectorias más realistas, como podría ser un estacionamiento, o incluso evitar obstáculos con los que en una trayectoria rectilínea colisionaría.

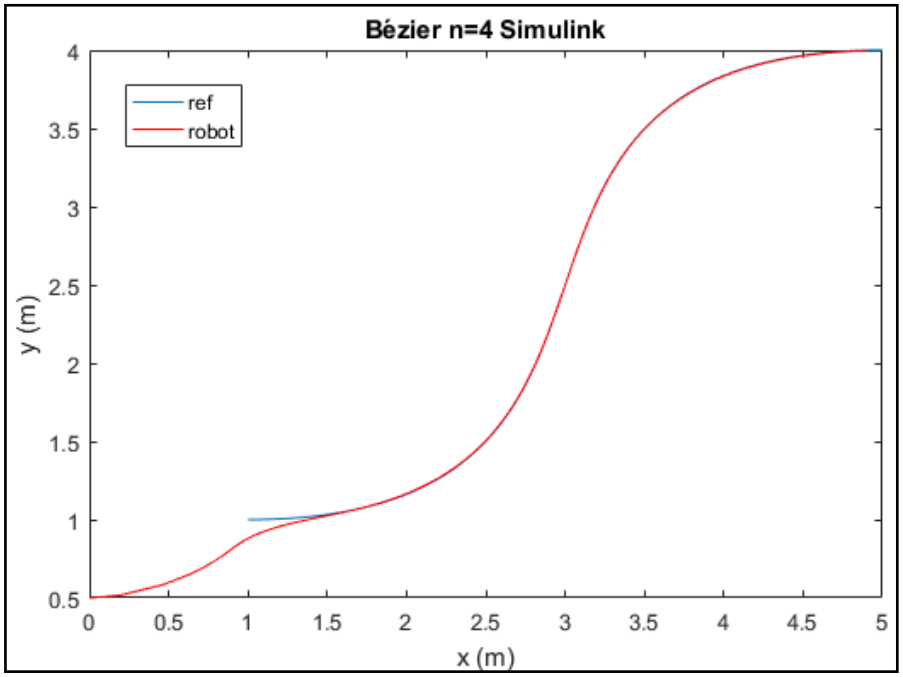


Figura 47. Simulación curva de Bézier n=4

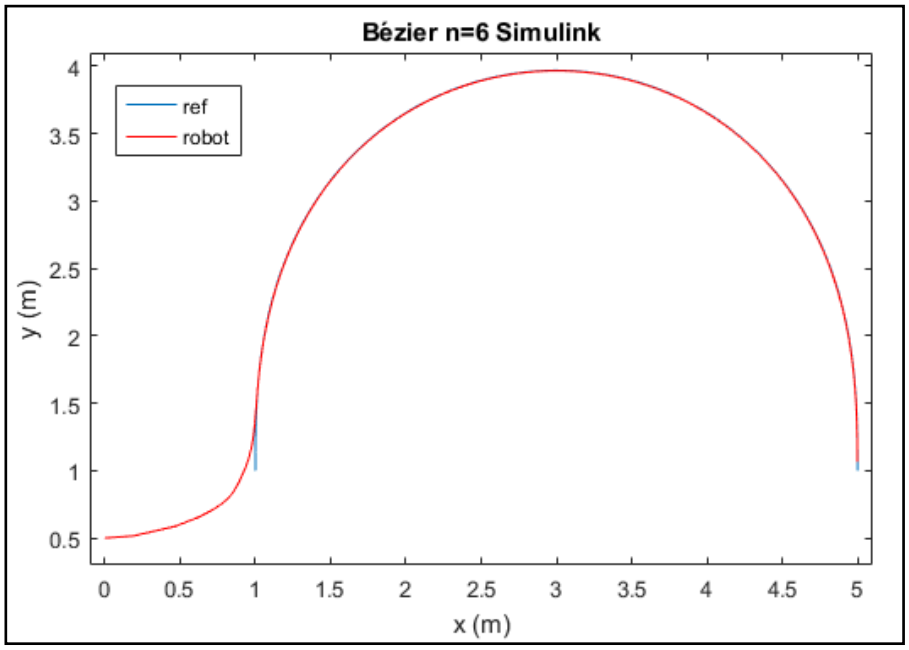


Figura 48. Simulación curva de Bézier n=6

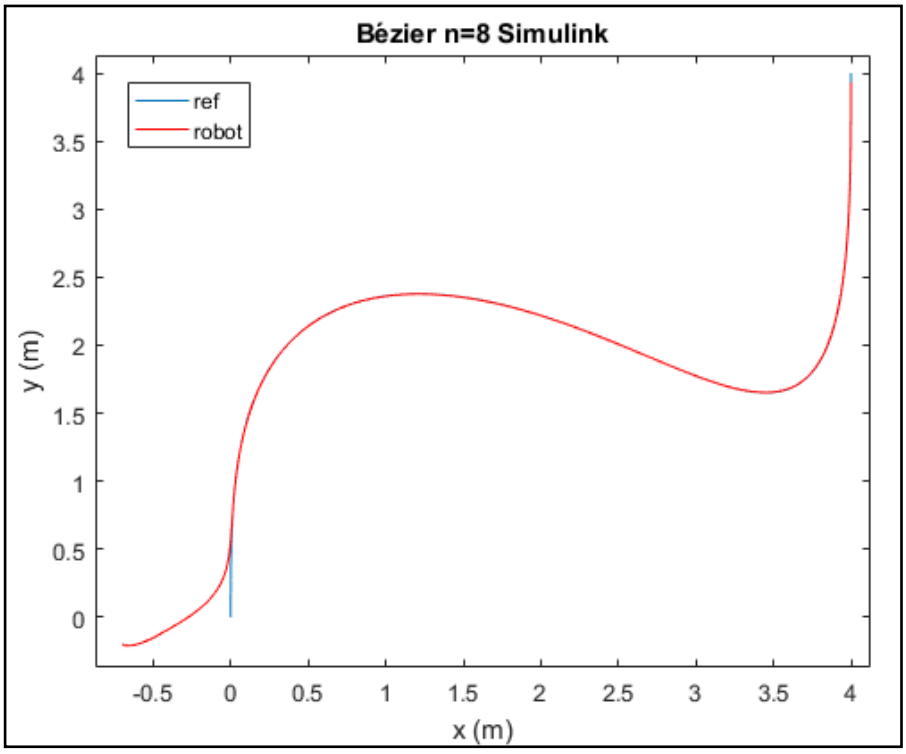


Figura 49. Simulación curva de Bézier n=8

Como se aprecia, en todas las trayectorias se ha conseguido seguir fielmente la referencia, incluso partiendo de un punto que no pertenece a la trayectoria. Para confirmar que el error tiende a cero tal y como se ha demostrado anteriormente, es conveniente presentar la evolución de los errores.

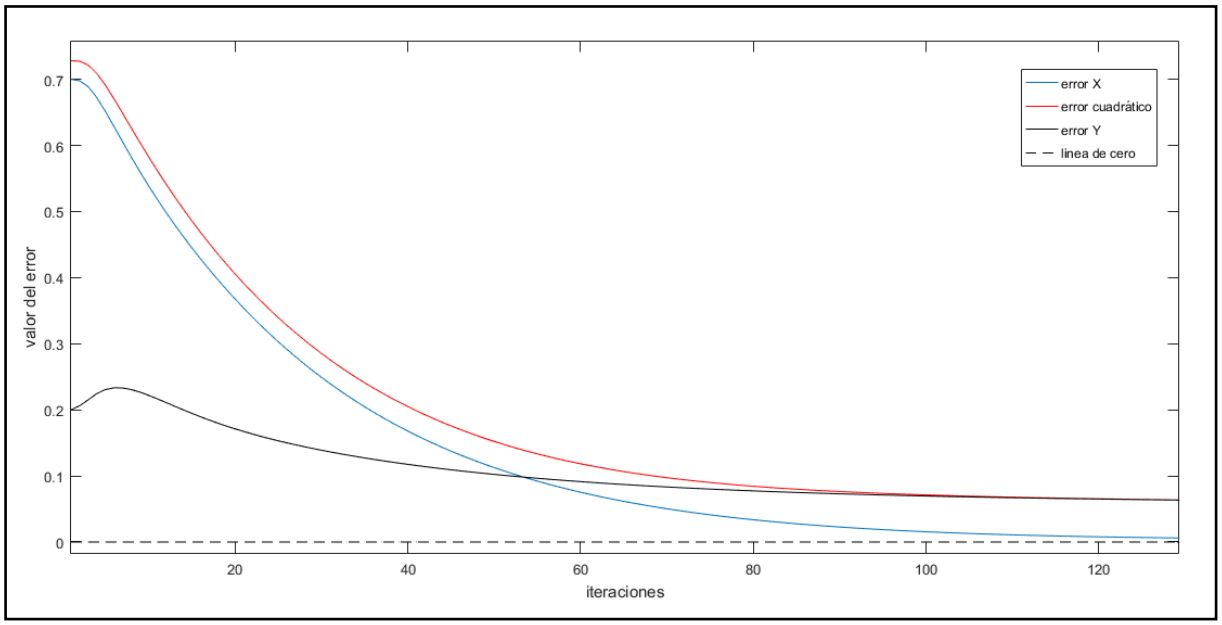


Figura 50. Convergencia a cero del error

Tras ver el adecuado funcionamiento de los controladores, se puede ponerlos a prueba cambiando las condiciones iniciales del robot: o bien alejándolo o acercándolo al primer punto de la trayectoria, o incluso cambiando la orientación inicial. Veamos algunos ejemplos.

- Cambio en la posición inicial

Se considera que el primer punto en la referencia es el (0, 0) y la orientación del robot es 0 rad.

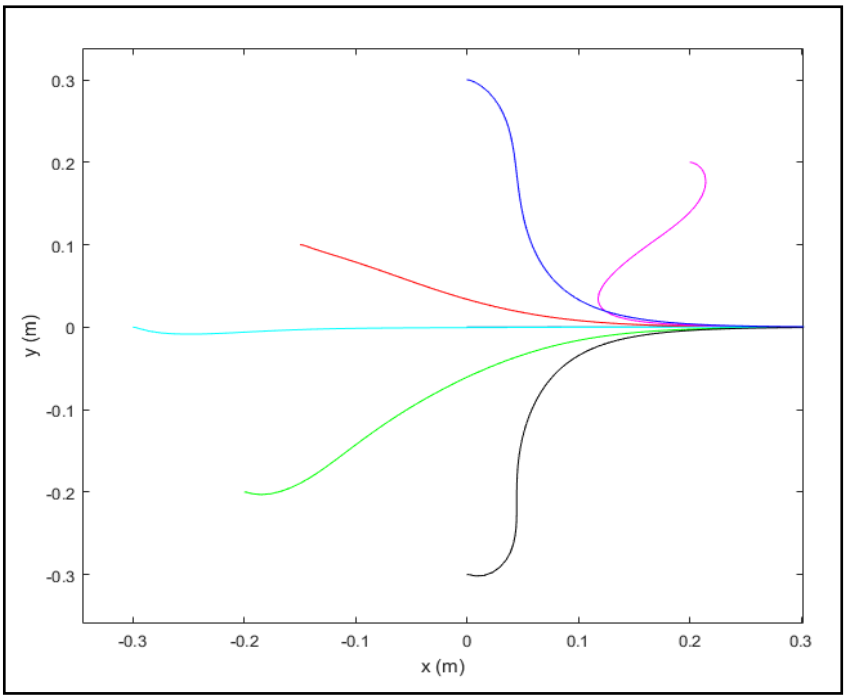


Figura 51. Cambios en las CI (I)

- Cambio en la orientación

Ahora se considerará un punto inicial fijo (-0.2, -0.2) y se cambiarán las orientaciones iniciales. El primer punto de la trayectoria de referencia sigue siendo el (0, 0).

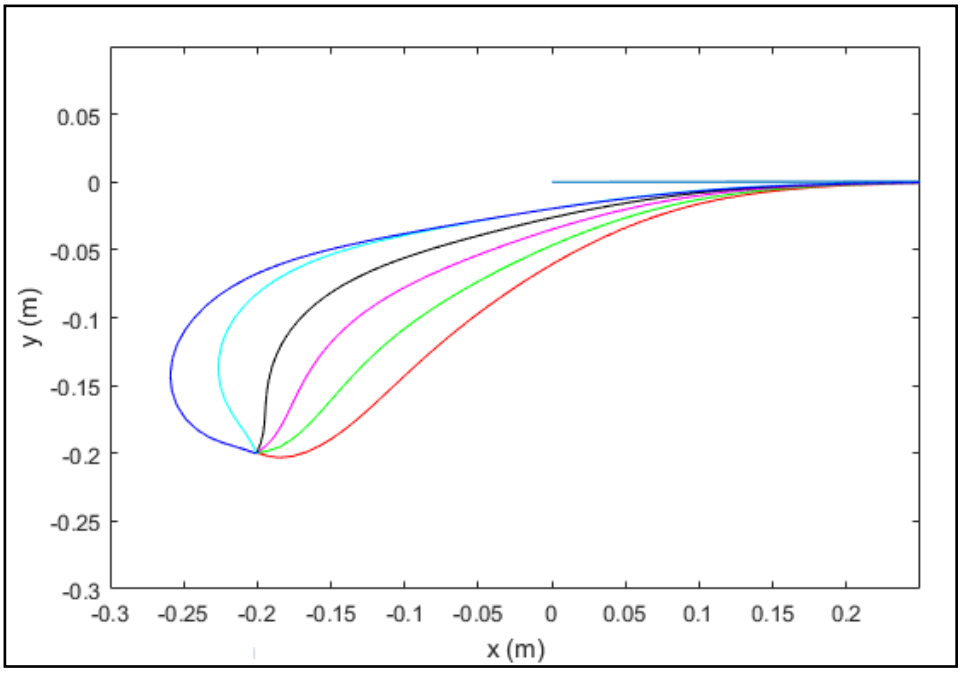


Figura 52. Cambios en las CI (II)

3.2.7 Mejoras del comportamiento: Integradores

A modo de simulación, y con el fin de ajustar lo máximo posible la trayectoria del robot a la de referencia se ha propuesto añadir un integrador a la ecuación del controlador. Esta propuesta fue sugerida por Gustavo Scaglia, que en diversas de sus investigaciones ha utilizado los controladores basados en álgebra lineal junto con los integradores para controlar procesos de la industria química.

En primer lugar, se deben calcular las expresiones de dichos integradores. Para ello, lo que se hace es calcular la integral del error en cada instante mediante el método de Euler. Para "x" queda:

$$U1x_{n+1} = \int_0^{(n+1)To} e_x(t)dt = U1x_n + To\ e_{xn}$$ ECUACIÓN 54

Efectuando lo mismo para "y" y para la orientación, se obtienen las siguientes expresiones de los controladores modificadas.

$$\begin{cases} v_k = \dfrac{x_{ref,k+1} - k_x\left(x_{ref,k} - x_k\right) - x_k + k1_x U_{xn+1}}{To}\cos\left(\theta_{ez,k}\right) + \\ \quad \dfrac{y_{ref,k+1} - k_y\left(y_{ref,k} - y_k\right) - y_k + k1_y U_{yn+1}}{To}\sin\left(\theta_{ez,k}\right) \\ w_k = \dfrac{\theta_{ez,k+1} - k_\theta\left(\theta_{ez,k} - \theta_k\right) - \theta_k + k1_\theta U_{\theta n+1}}{To} \end{cases}$$ ECUACIÓN 55

Dónde $\theta_{ez,k+1} = arctan\left(\frac{y_{ref,k+1} - k_y\left(y_{ref,k} - y_k\right) - y_k + k1_y U_{yn+1}}{x_{ref,k+1} - k_x\left(x_{ref,k} - x_k\right) - x_k + k1_x U_{xn+1}}\right)$

De acuerdo a los trabajos de Gustavo Scaglia, la convergencia del error a cero se mantiene, y se demuestra de forma similar al apartado 2.4.3. Como en el apartado anterior, la siguiente tabla presenta los parámetros utilizados en la simulación.

Parámetro	Valor	Comentarios
b	0.056	Mitad de la distancia entre las ruedas (m.)
r	0.028	Radio de las ruedas (m.)
kmp	9	Ganancia del control proporcional
kx	0.94	Constante que pondera el error en x
ky	0.94	Constante que pondera el error en y
kw	0.8	Constante que pondera el error en la orientación
K1x	0.9	Constante que pondera el efecto del integrador en X
K1y	0.9	Constante que pondera el efecto del integrador en Y
K1w	0.9	Constante que pondera el efecto del integrador en θ
To	0.02	Periodo de muestreo (s.)

Tabla 7. Parámetros de la simulación con integradores

Una vez vistos dichos parámetros, es hora de demostrar la mejoría que introducen los integradores mediante algunas figuras. A modo de ejemplo, se ha elegido la trayectoria en infinito.

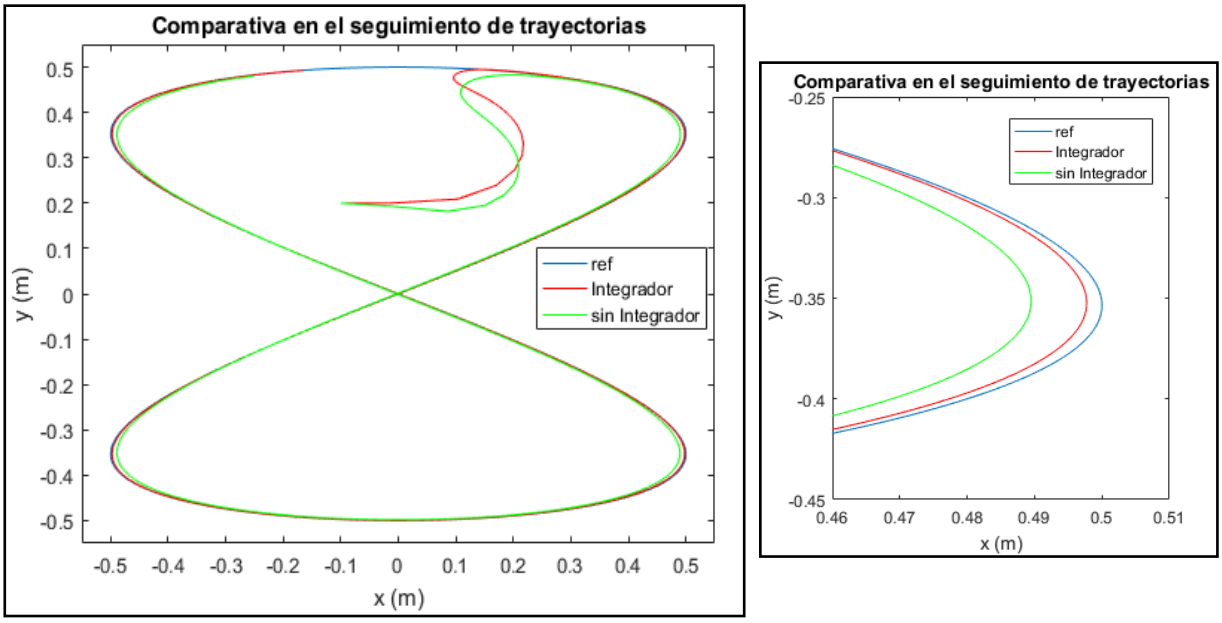


Figura 53. Comparativa en el seguimiento de trayectorias

Como se aprecia en el seguimiento de las trayectorias, si se añade el integrador al controlador según la ECUACIÓN 55, la trayectoria del robot se acerca mucho más a la de referencia. La figura de detalle muestra con mayor precisión este hecho.

Otra evidencia de la mejora introducida por los integradores se puede apreciar en la gráfica que compara los errores cuadráticos. En el caso del controlador con integrador los picos son más bajos y suaves. Además, se aproxima más al valor deseado de cero.

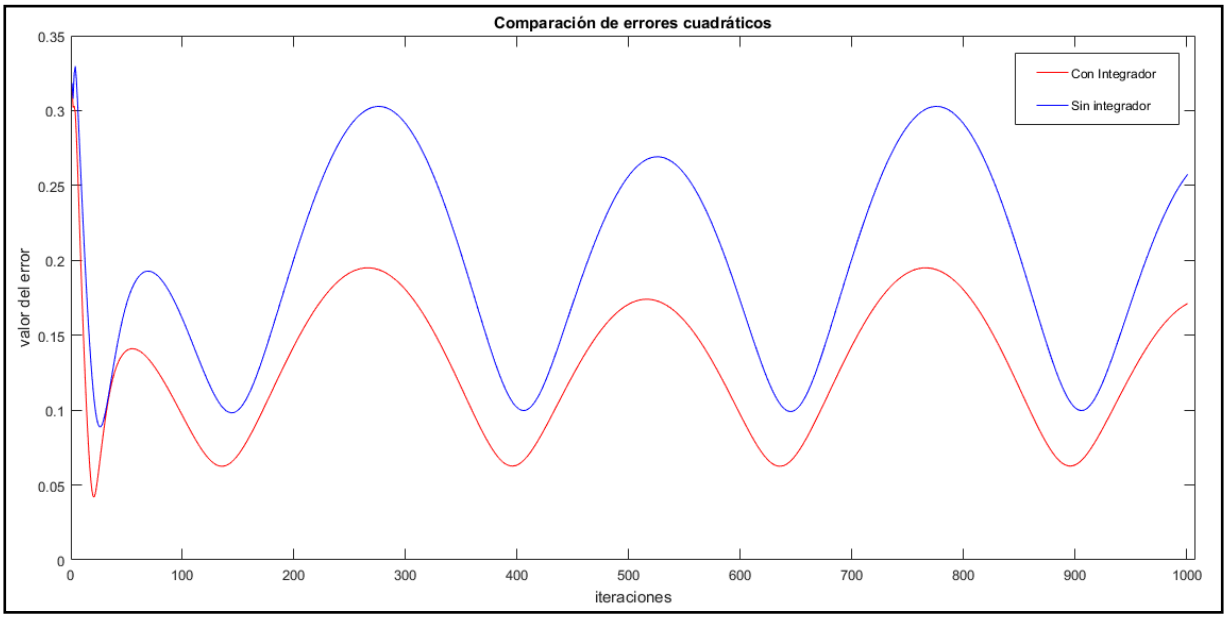


Figura 54. Comparación de los errores con integrador

3.3 Programación en los robots reales

En este apartado se expondrán los códigos en lenguaje C que se han programado en las dos plataformas utilizadas. Además, se comentarán los principales problemas que han ido apareciendo y las soluciones que se han propuesto e implementado.

A modo de ejemplo se expone el código que genera una trayectoria circular, pero en el apartado 3.3.5 se verá como generar otras trayectorias y los resultados que se obtienen en el robot real al generar las mismas.

3.3.1 Estructura del código

En primer lugar, es importante describir cómo será la estructura principal del código. El cuerpo del código será el siguiente:

```
1. #librerias
2. programación de la función ATAN2 (si procede)
3.
4.       main
5.   declaración de variables
6.   reseteamos encoders
7.   creación del nuevo fichero
8.   calculo del número de iteraciones
9.
10.        Bucle de control
11.          generación de la trayectoria
12.          leer encoders para saber cuánto se ha movido
13.          cálculo de la vel. angular y lineal de las ruedas
14.          cálculo de la vel. angular y lineal del robot
15.          cálculo de la posición y orientación real del robot
16.          cálculo de la primera parte del control cinemático
17.          cálculo de la orientación de referencia
18.          control para evitar cambios bruscos en la orientación
19.          cálculo expresión del control cinemático (v, w)
20.          cálculo de la cinemática inversa(vrueda_d, vreuda_i)
21.          cálculo vel. angular ruedas
22.          cálculo del error en la vel. angular de las ruedas
23.          control proporcional (obtenemos acciones de control)
24.          saturamos las acciones de control
25.          aplicamos las acciones de control en los motores
26.          almacenar valor de los encoders para la siguiente iteración
27.          actualizar valores para la próxima iteración
28.          cálculo del error cuadrático (opcional)
29.          escribir los valores en el fichero
30.          verificar periodo de muestreo e incrementamos contador
31.        Cierre del bucle de control
32.
33.      cierre fichero
34.      cierre main
```

3.3.2 Plataforma Lego NXT

A continuación se presenta el código que se ha programado para que el robot Lego NXT siga la trayectoria propuesta. En primer lugar se ha programado la función “atan2” ya que esta no está

disponible en las librerías de Lego NXT. La programación y justificación de la necesidad de esta función están presentes en el Anexo II: Programación de la función atan2.

Cabe también mencionar que se ha utilizado el fichero "WriteFile3" para escribir los valores deseados en el fichero de texto. Este fichero permite escribir números con hasta tres decimales. La programación de dicho fichero fue proporcionada, por lo que simplemente se ha utilizado como herramienta para obtener datos del robot real.

```
1. #pragma config(Motor,  motorA,  tmotorNormal, openLoop, encoder)
2. #pragma config(Motor,  motorB,  tmotorNormal, openLoop, encoder)
3. #pragma config(Motor,  motorC,  tmotorNormal, openLoop, encoder)
4. //*!!Code automatically generated by 'ROBOTC' configuration wizard
               !!*//
5. #pragma  platform(NXT)
6.
7. #include "writeFile3.c"
8. const string  sFileName = "circulo_mnum.txt";
9.
10.      //******  Funcion ATAN2  ******
11.      float atan2(float y,float x)
12.      {
13.       float result; //lo que nos devuelve la funcion, es un angulo
14.       float PI=3.141592653589793;
15.
16.        if(y==0 && x==0)
17.          { result=0.0; }
18.        if(y>0 && x>0) //primer cuadrante
19.          { result=atan(y/x); }
20.        if(y>0 && x==0) //***frontera primero-segundo
21.          { result=PI/2.0;}
22.        if(y>0 && x<0) //segundo cuadrante
23.          { result=atan(y/x)+PI; }
24.        if(y==0 && x<0) //***frontera segundo-tercero
25.          { result=PI;}
26.        if(y<0 && x<0) //tercer cuadrante
27.          { result=atan(y/x)-PI; }
28.        if(y<0 && x==0) //***frontera tercero-cuarto
29.          { result=-PI/2.0; }
30.        if(y<0 && x>0) //cuarto cuadrante
31.          { result=atan(y/x); }
32.        if(y==0 && x>0) //***frontera cuarto-primero
33.          { result=0.0; }
34.
35.      return result;
36.      }
37.
38.      task main()
39.      {
40.            long i = 0;                 // contador
41.            float tfinal=25.0;          // tiempo final
42.            long niteraciones=0;        // numero iteraciones
43.            long ruedaAant=0;           // posición anterior rueda
   derecha (motorA)
44.            long ruedaAnue=0;           // posición actual rueda
   derecha
```

```
45.             long ruedaBant=0;           // posición anterior rueda
   izquierda (motorB)
46.             long ruedaBnue=0;           // posición actual rueda
   izquierda
47.             long acontrolA,acontrolB;   // acciones de control rueda
   derecha (motorA) e izquierda (motorB)
48.
49.             float wd, wi;               // velocidades angulares
   rueda derecha (wd) e izquierda (wi)
50.             float vd, vi;               // velocidades lineales rueda
   derecha (vd) e izquierda (vi)
51.
52.             float vd_ref, vi_ref;       // velocidades lineales de
   referencia (RUEDAS)
53.             float wd_ref,wi_ref;        // velocidades angulares de
   referencia (RUEDAS)
54.
55.             float velLin_robot;         // velocidad lineal robot
56.             float velAng_robot;         // velocidad angular robot
57.
58.             float vrobot=0.0;           // vel lineal cont cinem
59.             float wrobot=0.0;           // vel angular cont cinem
60.      //***Control
61.             float kx=0.94;              // ganancia velocidad lineal
62.             float ky=0.94;              // ganancia velocidad lineal
63.             float kw=0.8;               // ganancia velocidad angular
64.             float xref, xrefante=0.0;   // posición de referencia X
65.             float yref, yrefante=0.0;   // posición de referencia Y
66.             float thetaref, thetarefante=0.0;  // orientación de ref
67.             float theta=0.0;            // orientación inicial
68.             float x=0.3;                // posición inicial eje X
69.             float y=0.0;                // posición inicial eje Y
70.      //***Auxiliares
71.             float numerador;            // auxiliar para atan2
72.             float denominador;          // auxiliar para atan2
73.      //***Otras variables
74.             float error_wd, error_wi;   // errores vels ang ruedas
75.             float ex=0.0, ey=0.0;       // errores posición eje X e Y
76.
77.             float Kmp=9.0;              // ganancia control dinámico
78.             float rcir=0.410;           // radio de la circunferencia
79.             float b=0.056;              // (separación ruedas)/2
80.             float PI=3.141592653589793;
81.             float pul2rad=0.0174533;    //
   pul2rad=2*pi/360=0.0174533: cad vuelta: 360 pulsos de encoder
82.             float radiorueda=0.028;
83.             float T0=0.02;              // periodo de muestreo
84.             nMotorEncoder[motorA] = 0; // reseteamos el encoder de
   la rueda derecha
85.             nMotorEncoder[motorB] = 0; // reseteamos el encoder de
   la rueda izquierda
86.         Delete(sFileName,nIoResult);
87.         createTextFile(sFileName, 30000);
88.
89.         i=0;
90.         niteraciones=(int)((tfinal)/T0);
91.
92.         while(i < niteraciones)     //bucle de control
```

```
93.             {
94.               ClearTimer(T1);
95.
96.              // cálculo referencias de posición: tray. CIRCULAR
97.                       xref=rcir*cos(2*PI*0.06*T0*i);
98.                       yref=rcir*sin(2*PI*0.06*T0*i);
99.
100.             // obtenemos el valor de los encoders para ver cuánto se
    ha movido
101.                      ruedaAnue=nMotorEncoder[motorA];
102.                      ruedaBnue=nMotorEncoder[motorB];
103.
104.             // calculamos las velocidades angulares de las ruedas
105.                      wd=pul2rad*(ruedaAnue-ruedaAant)/T0;
106.                      wi=pul2rad*(ruedaBnue-ruedaBant)/T0;
107.
108.             // calculamos las velocidades lineales de las ruedas
109.                      vd=wd*radiorueda;
110.                      vi=wi*radiorueda;
111.
112.             // calculamos la velocidad lineal del robot
113.                      velLin_robot=(vd+vi)/2;
114.
115.             // calculamos la velocidad angular del robot
116.                      velAng_robot=(vd-vi)/(2*b);
117.
118.             // calculamos la posición X-Y y la orientación del robot
119.                      x=x+velLin_robot*T0*cos(theta);
120.                      y=y+velLin_robot*T0*sin(theta);
121.                      theta=theta+velAng_robot*T0;
122.
123.                      numerador=yref-ky*(yrefante-y)-y;
124.                      denominador=xref-kx*(xrefante-x)-x;
125.                      thetaref=atan2(numerador, denominador);
126.
127.                  if( (thetaref-thetarefante) < -5.0 ) // CUAD2-CUAD3
128.                  {
129.                     thetaref=thetaref+2*PI;
130.                  }
131.
132.                  if( (thetarefante-thetaref) > 5.0 ) // CUAD4-CUAD1
133.                  {
134.                     thetaref=thetaref+2*PI;
135.                  }
136.
137.                  if( (thetaref-thetarefante) > 5.0 ) // CUAD3-CUAD2
138.                  {
139.                     thetaref=thetaref-2*PI;
140.                  }
141.
142.                  if( (thetarefante-thetaref) < -5.0 ) // CUAD1-CUAD4
143.                  {
144.                     thetaref=thetaref-2*PI;
145.                  }
146.
147.
148.             // calculamos la expresión del control cinemático
```

```
149.                          Vrobot=(1/T0)*
   (denominador*cos(thetarefante) + numerador*sin(thetarefante));
150.      wrobot= (1/T0)* (thetaref-kw*(thetarefante-theta)-theta);
151.
152.
153.            // calculamos el modelo cinemático inverso del robot
154.                    vd_ref=vrobot+b*wrobot;
155.                    vi_ref=vrobot-b*wrobot;
156.
157.            // calculamos las velocidades angulares de referencia
   para el control dinámico
158.                    wd_ref=vd_ref/radiorueda;
159.                    wi_ref=vi_ref/radiorueda;
160.
161.            // calculamos errores velocidad angular ruedas
162.                    error_wd=wd_ref-wd;
163.                    error_wi=wi_ref-wi;
164.
165.            // calculamos las acciones de control a aplicar
166.                    acontrolA=error_wd*Kmp;
167.                    acontrolB=error_wi*Kmp;
168.
169.            // saturamos las acciones de control
170.                    if (acontrolA>100)
171.                          {
172.                                acontrolA=100;
173.                          }
174.                    if (acontrolA<-100)
175.                          {
176.                                acontrolA=-100;
177.                          }
178.                    if (acontrolB>100)
179.                          {
180.                                acontrolB=100;
181.                          }
182.                    if (acontrolB<-100)
183.                          {
184.                                acontrolB=-100;
185.                          }
186.
187.            // aplicamos las acciones de control a los motores
188.                    motor[motorA] = acontrolA;
189.                    motor[motorB] = acontrolB;
190.
191.            // almacenamos los valores de los encoder para la
   próxima iteración
192.                    ruedaAant=ruedaAnue;
193.                    ruedaBant=ruedaBnue;
194.
195.            // damos los valores para la próxima iteración
196.                    xrefante=xref;
197.                    yrefante=yref;
198.                    thetarefante=thetaref;
199.
200.            // calculo índice integral error cuadrático
201.                    ex=xref-x;
202.                    ey=yref-y;
203.
```

```
204.            // escribimos los valores importantes en el fichero
205.                    writeFloatNumber(xref);
206.                    writeFloatNumber(yref);
207.                    writeFloatNumber(x);
208.                    writeFloatNumber(y);
209.                    writeFloatNumber(thetaref);
210.                    writeNewLine();
211.
212.            // verificamos To e incrementamos el contador
213.               while(time1(T1)<20)
214.                     {
215.                         i++;
216.                         wait1Msec(4);    //  espera 4 MS
217.                     }
218.       } // cierre bucle de control
219.       closeWriteTextFile();
220.     } // cierre main
```

3.3.3 Plataforma Lego EV3

En el caso de la plataforma Lego EV3, ya encontramos en sus librerías la función atan2, por lo que su programación no es necesaria. En cuanto a la escritura de datos en un fichero de texto, se ha utilizado la función DATALOG. Para mayor detalle, se recomienda ver el Anexo III: Uso de la función Datalog.

```
1. #pragma config(StandardModel, "EV3_REMBOT")
2. //*!!Code automatically generated by 'ROBOTC' configuration wizard
              !!*//
3.
4. //const string  sFileName = "circulo_mnum.txt";
5. char * filename = "datalog-0.txt"; //nombre del fichero
6.
7. task main()
8. {
9.        // Datalog
10.          long fileHandle;
11.          char * string1 = "circulo EV3:"; //Frase fichero
12.          int strlen1 = strlen(string1); //longitud de la frase
13.          fileHandle = fileOpenWrite(filename); // Abrir archivo
14.
15.       //
16.          long i = 0;                 // contador
17.          float tfinal=25.0;          // tiempo final
18.          long niteraciones=0;        // numero iteraciones
19.          long ruedaAant=0;           // posición anterior rueda
   derecha (motorA)
20.          long ruedaAnue=0;           // posición actual rueda derecha
21.          long ruedaBant=0;           // posición anterior rueda
   izquierda (motorB)
22.          long ruedaBnue=0;           // posición actual rueda
   izquierda
23.          long acontrolA,acontrolB;   // acciones de control
24.          float wd, wi;               // velocidades angulares ruedas
25.          float vd, vi;               // velocidades lineales ruedas
26.          float vd_ref, vi_ref;       // velocidades lineales ref
```

```
27.         float wd_ref,wi_ref;        // velocidades angulares ref
28.         float velLin_robot;         // velocidad lineal robot
29.         float velAng_robot;         // velocidad angular robot
30.         float vrobot=0.0;           // velocidad lineal c. cinem.
31.         float wrobot=0.0;           // velocidad angular c. cinem.
32.       //***Control
33.         float kx=0.94;              // ganancia velocidad lineal
34.         float ky=0.94;              // ganancia velocidad lineal
35.         float kw=0.8;               // ganancia velocidad angular
36.         float xref, xrefante=0.0;   // posición de referencia eje X
37.         float yref, yrefante=0.0;   // posición de referencia eje Y
38.         float thetaref, thetarefante=0.0;  // orientación de ref
39.         float theta=0.0;            // orientación inicial
40.         float x=0.3;                // posición inicial eje X
41.         float y=0.0;                // posición inicial eje Y
42.       //***Auxiliares
43.         float numerador;            // auxiliar para atan2
44.         float denominador;          // auxiliar para atan2
45.       //***Otras variables
46.         float error_wd, error_wi;   // errores velocidades angulares
47.         float ex=0.0, ey=0.0;       // errores posición eje X e Y
48.         float Kmp=9.0;              // ganancia control dinámico
49.         float rcir=0.410;           // radio de la circunferencia
50.         float b=0.0625;             //(separación entre las ruedas)/2
51.         float PI=3.141592653589793;
52.         float pul2rad=0.0174533;    // pul2rad=2*pi/360=0.0174533:
    cada vuelta: 360 pulsos de encoder
53.         float radiorueda=0.028;
54.         float T0=0.02;              // periodo de muestreo
55.         resetMotorEncoder(leftMotor);// reseteamos el encoder de la
    rueda derecha
56.         resetMotorEncoder(rightMotor); // reseteamos el encoder de
    la rueda izquierda
57.
58.      fileWriteData(fileHandle, string1, strlen1 + 1);
59.      if (!datalogOpen(0, 4, true))        //4 es el número de
    columnas
60.      writeDebugStreamLine("Unable to open datalog");
61.
62.         i=0;
63.         niteraciones=(int)((tfinal)/T0);
64.
65.         while(i < niteraciones)     //bucle de control
66.         {
67.            clearTimer(T1);
68.
69.             // cálculo referencias de posición: trayectoria CIRCULAR
70.                    xref=rcir*cos(2*PI*0.06*T0*i);
71.                    yref=rcir*sin(2*PI*0.06*T0*i);
72.
73.             // obtenemos el valor de los encoders para ver cuanto se
    ha movido
74.                    ruedaAnue=getMotorEncoder(leftMotor);
75.                    ruedaBnue=getMotorEncoder(rightMotor);
76.
77.             // calculamos las velocidades angulares (rad/s) de las
    ruedas
78.                    wd=pul2rad*(ruedaAnue-ruedaAant)/T0;
```

```
79.                     wi=pul2rad*(ruedaBnue-ruedaBant)/T0;
80.
81.             // calculamos las velocidades lineales de las ruedas
82.                     vd=wd*radiorueda;
83.                     vi=wi*radiorueda;
84.
85.             // calculamos la velocidad lineal del robot
86.                     velLin_robot=(vd+vi)/2;
87.
88.             // calculamos la velocidad angular del robot
89.                     velAng_robot=(vd-vi)/(2*b);
90.
91.             // calculamos la posición X-Y y la orientación del robot
92.                     x=x+velLin_robot*T0*cos(theta);
93.                     y=y+velLin_robot*T0*sin(theta);
94.                     theta=theta+velAng_robot*T0;
95.
96.                     numerador=yref-ky*(yrefante-y)-y;
97.                     denominador=xref-kx*(xrefante-x)-x;
98.                     thetaref=atan2(numerador,denominador);
99.
100.                if( (thetaref-thetarefante) < -5.0 ) // CUAD2-CUAD3
101.                {
102.                   thetaref=thetaref+2*PI;
103.                }
104.
105.                if( (thetarefante-thetaref) > 5.0 ) // CUAD4-CUAD1
106.                {
107.                   thetaref=thetaref+2*PI;
108.                }
109.
110.                if( (thetaref-thetarefante) > 5.0 ) // CUAD3-CUAD2
111.                {
112.                   thetaref=thetaref-2*PI;
113.                }
114.
115.                if( (thetarefante-thetaref) < -5.0 ) // CUAD1-CUAD4
116.                {
117.                   thetaref=thetaref-2*PI;
118.                }
119.
120.            // calculamos el control cinemático del robot
121.            vrobot=(1/T0)*
    (denominador*cos(thetarefante)+numerador*sin(thetarefante));
122.            wrobot=(1/T0)*(thetaref-kw*(thetarefante-theta)-theta);
123.
124.            // calculamos el modelo cinemático inverso del robot
125.            vd_ref=vrobot+b*wrobot;
126.            vi_ref=vrobot-b*wrobot;
127.
128.            // calculamos las velocidades angulares de referencia
129.                    wd_ref=vd_ref/radiorueda;
130.                    wi_ref=vi_ref/radiorueda;
131.
132.            // calculamos los errores de la velocidad angular ruedas
133.                    error_wd=wd_ref-wd;
134.                    error_wi=wi_ref-wi;
135.
```

```
136.            // calculamos las acciones de control a aplicar
137.                    acontrolA=error_wd*Kmp;
138.                    acontrolB=error_wi*Kmp;
139.
140.            // saturamos las acciones de control
141.                    if (acontrolA>100)
142.                            {
143.                                    acontrolA=100;
144.                            }
145.                    if (acontrolA<-100)
146.                            {
147.                                    acontrolA=-100;
148.                            }
149.                    if (acontrolB>100)
150.                            {
151.                                    acontrolB=100;
152.                            }
153.                    if (acontrolB<-100)
154.                            {
155.                                    acontrolB=-100;
156.                            }
157.
158.            // aplicamos las acciones de control a los motores
159.                    setMotorSpeed(leftMotor,acontrolA);
160.                    setMotorSpeed(rightMotor,acontrolB);
161.
162.            // almacenamos los valores de los encoder para la
   próxima iteración
163.                    ruedaAant=ruedaAnue;
164.                    ruedaBant=ruedaBnue;
165.
166.            // damos los valores para la próxima iteración
167.                    xrefante=xref;
168.                    yrefante=yref;
169.                    thetarefante=thetaref;
170.
171.            // calculo índice integral error cuadrático
172.                    ex=xref-x;
173.                    ey=yref-y;
174.
175.            // escribimos los valores importantes en el fichero
176.                    datalogAddFloat(0, xref);
177.                    datalogAddFloat(1, yref);
178.                    datalogAddFloat(2, x);
179.                    datalogAddFloat(3, y);
180.
181.
182.            // verificamos To e incrementamos el valor del contador
183.               while(time1(T1)<20)
184.                       {
185.                           i++;
186.                           wait1Msec(4);    // espera 4 MS
187.                       }
188.       }// cierre bucle de control
189.      datalogClose();
190.     } cierre main
```

3.3.4 Cambios brucos de orientación

Una de las particulares de estos nuevos controladores es que requieren de la orientación del robot para aplicar la acción de control. Como se ha explicado, esto conlleva el problema del cambio brusco que se produce al cambiar de cuadrantes. Por ejemplo, si pasamos del segundo al tercer cuadrante estaremos pasando en una sola iteración de un ángulo π a uno –π. Además, si restamos los dos valores tal y como propone el controlador (ECUACIÓN **26**) nos dará un valor cercano a 2π, un valor muy grande y que causará en el robot un comportamiento inesperado e indeseado.

Para resolver esto se ha programado un fragmento de código que corrige el valor de la orientación con el fin de no obtener grandes saltos en los valores de la misma.

```
1.              if( (thetaref-thetarefante) < -5.0 ) // CUAD2-CUAD3
2.              {
3.                 thetaref=thetaref+2*PI;
4.              }
5.
6.              if( (thetarefante-thetaref) > 5.0 ) // CUAD4-CUAD1
7.              {
8.                 thetaref=thetaref+2*PI;
9.              }
10.
11.             if( (thetaref-thetarefante) > 5.0 ) // CUAD3-CUAD2
12.             {
13.                thetaref=thetaref-2*PI;
14.             }
15.
16.             if( (thetarefante-thetaref) < -5.0 ) // CUAD1-CUAD4
17.             {
18.                thetaref=thetaref-2*PI;
19.             }
```

3.3.5 Generación de distintas trayectorias

En este apartado se mostrará el modo que se ha empleado para generar las distintas trayectorias utilizadas para demostrar la viabilidad de los controladores. La estrategia que se ha utilizado consiste en generar la trayectoria punto a punto, siendo la variable “incre” la que indica la separación entre un punto y el posterior. La programación de estas referencias será idéntica para ambas plataformas.

- Trayectoria cuadrada.

```
1.        if ((i>=0) && (i<375))
2.              {
3.              xref=incre*i;
4.              yref=0;
5.              }
6.        if ((i>375) && (i<750))
7.              {
8.              xref=incre*375.0;
9.              yref=(i-375.0)*incre;
10.             }
11.        if ((i>750) && (i<1125))
12.             {
13.             xref=(incre*375.0)-(incre*(i-750.0));
```

```
14.             yref=incre*375.0;
15.             }
16.        if ((i>1125) && (i<1500))
17.             {
18.             xref=0;
19.             yref=(incre*375.0)-(incre*(i-1125.0));
20.             }
```

- Trayectoria en sierra.

```
1.         if ((i>=0) && (i<251))
2.               {
3.               xref=incre*i;
4.               yref=incre*i;
5.               }
6.         if ((i>250) && (i<501))
7.               {
8.               xref=incre*i;
9.               yref=incre*250.0-(i-250.0)*incre;
10.              }
11.         if ((i>500) && (i<751))
12.              {
13.              xref=incre*i;
14.              yref=incre*(i-500.0);
15.              }
16.         if ((i>750) && (i<1001))
17.              {
18.              xref=incre*i;
19.              yref=(incre*250.0)-(i-750)*incre;
20.              }
```

- Trayectoria en infinito.

Para generar esta trayectoria, simplemente se debe utilizar la circular y reducir a la mitad la frecuencia de una de las componentes tal y como se muestra a continuación.

```
1.              xref=rcir*sin(2*PI*0.06*T0*i);
2.              yref=rcir*cos(2*PI*0.03*T0*i);
```

- Curvas de Bézier

Por último, para calcular las trayectorias de referencia cuando se quieran describir curvas de Bézier se utilizarán las expresiones anteriormente deducidas. Una de las ventajas de las curvas de Bézier es que se pueden generar por separado las referencias para cada uno de los ejes de coordenadas. Para cuatro puntos quedaría:

```
1.         xref=P0x*(1-t)^3 +3*P1x*t*(1-t)^2 +3*P2x*t^2*(1-t) +P3x+t^3;
2.         yref=P0y*(1-t)^3 +3*P1y*t*(1-t)^2 +3*P2y*t^2*(1-t) +P3y+t^3;
```

4 RESULTADOS EN ENTORNO REAL

En este apartado se presentarán los resultados obtenidos al programar al robot para que siga una serie de trayectorias. En primer lugar, se mostrarán los resultados obtenidos al describir las figuras que también han sido utilizadas en la simulación. Una vez vista, se pasará a utilizar las curvas de Bézier para generar distintas trayectorias.

4.1 Resultados Lego NXT

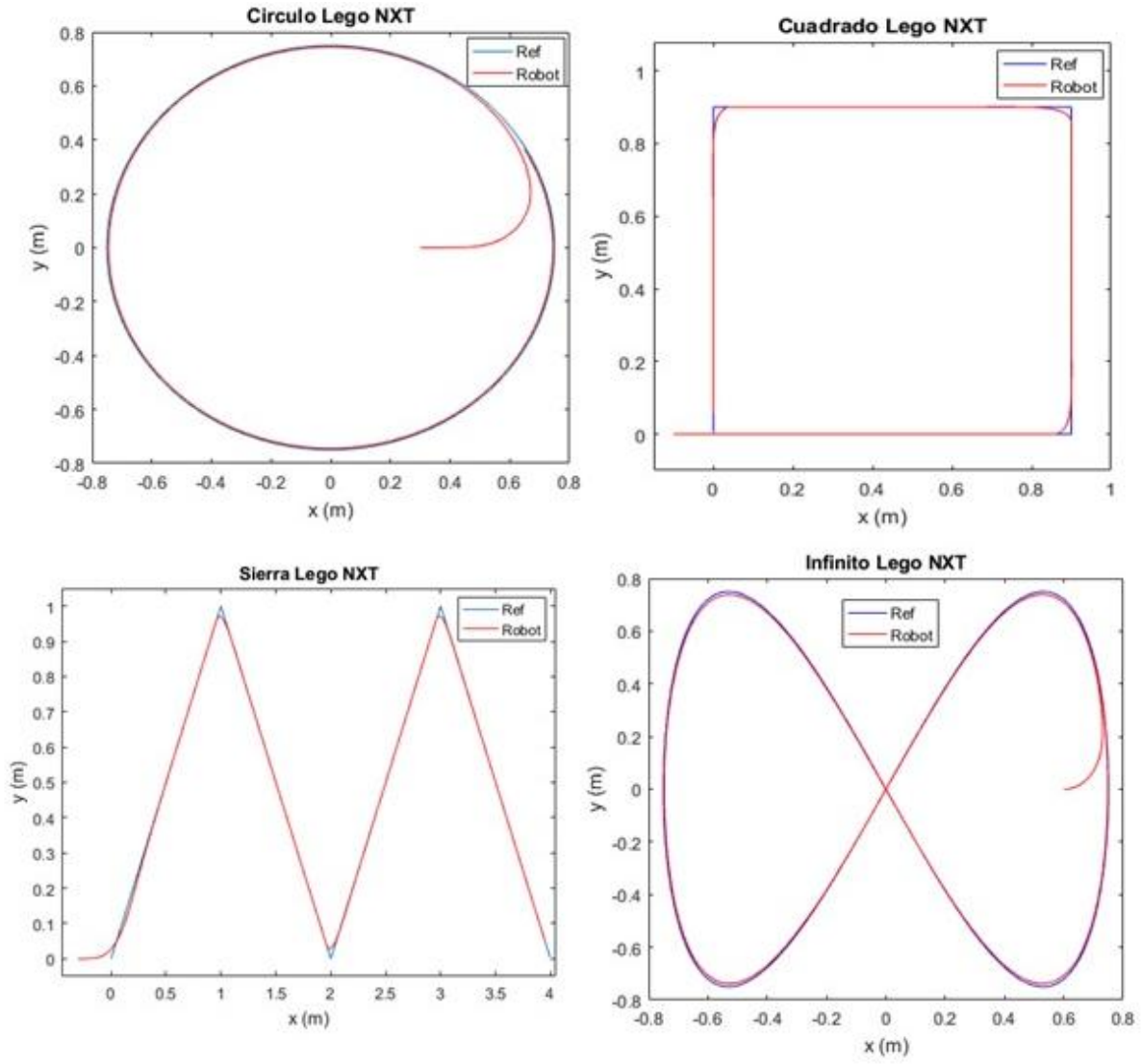


Figura 55. Resultados Lego NXT

Al utilizar la plataforma Lego NXT se observa que se siguen todas las trayectorias de manera adecuada. Además, el controlador es capaz de hacer al robot anticiparse a los cambios bruscos como son las esquinas. De todas formas, son más interesantes las trayectorias suaves y curvilíneas, ya que como se ha dicho, el controlador no garantiza un buen comportamiento ante referencias con derivadas discontinuas. [18]

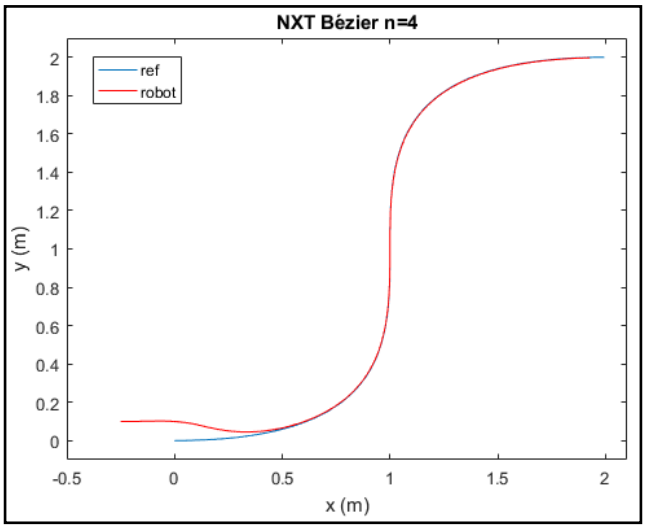


Figura 56. NXT Bézier n=4

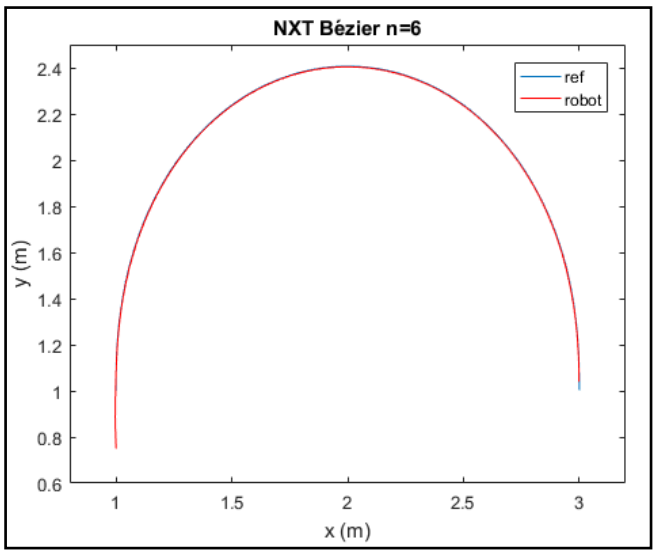


Figura 57. NXT Bézier n=6

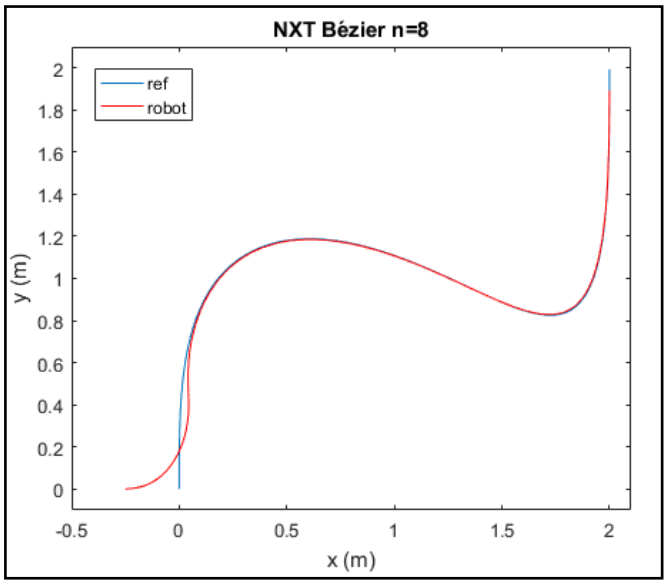


Figura 58. NXT Bézier n=8

Como se aprecia en las figuras generadas, el robot Lego NXT se ha comportado de excelente forma al seguir las trayectorias anteriormente generadas. Las figuras de 4 y 8 puntos podrían simular un estacionamiento de un vehículo mientras que la de 6 puntos podría simular un cambio de sentido. Aunque estas trayectorias han servido como ejemplo, la facilidad para trazar nuevas rutas basadas en curvas de Bézier permitirá que el robot pueda describir casi cualquier trayectoria.

4.2 Resultados Lego EV3

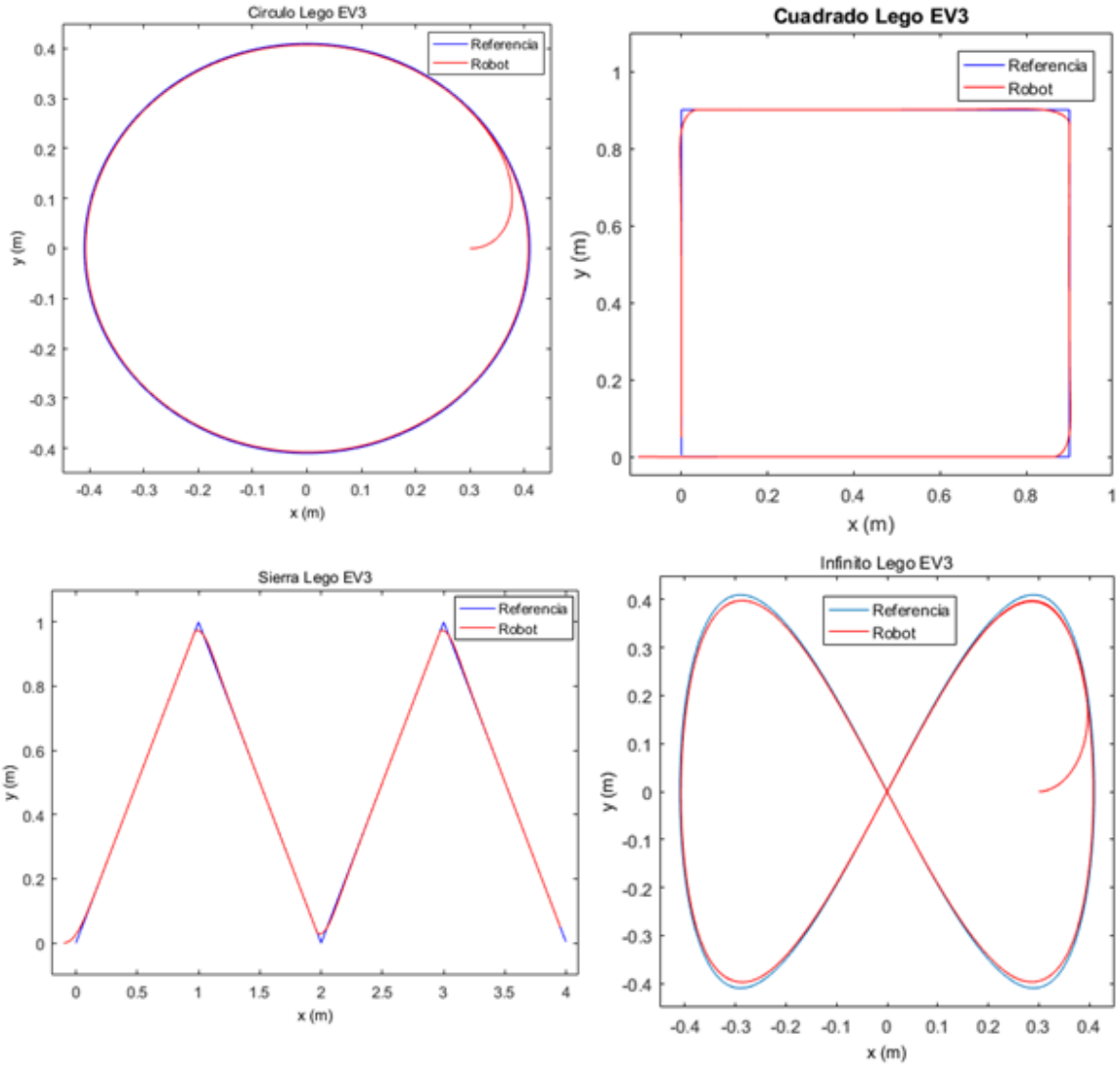


Figura 59. Resultados Lego EV3

De igual forma, se obtienen excelentes resultados con la plataforma más moderna Lego EV3. Ahora se utilizarán las curvas de Bézier, cuyos resultados en el seguimiento de la trayectoria, como se observa a continuación, siguen siendo muy precisos.

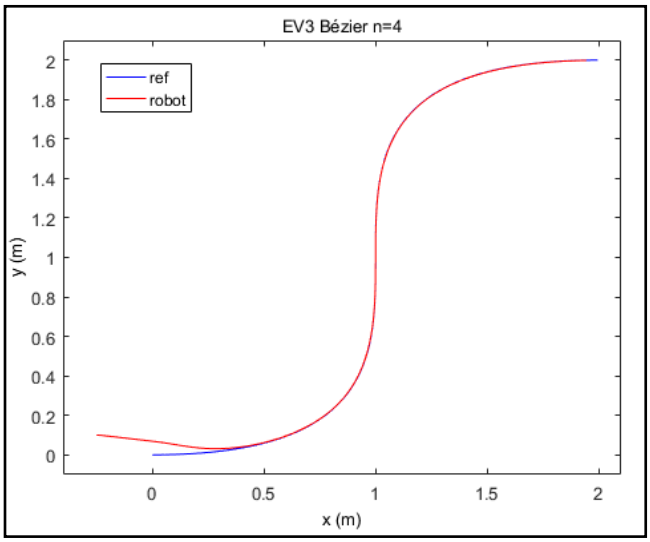


Figura 60. EV3 Bézier n=4

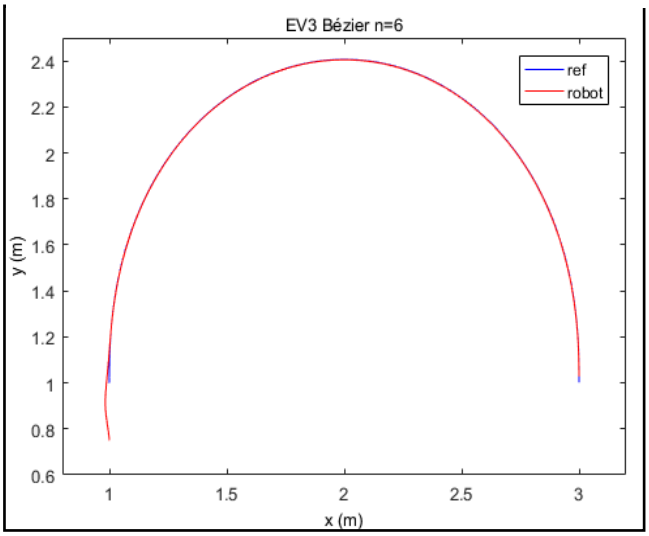


Figura 61. EV3 Bézier n=6

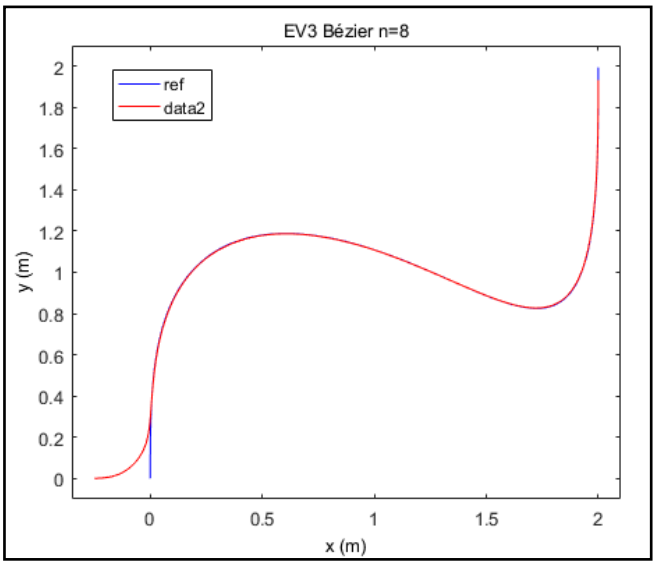


Figura 62. EV3 Bézier n=8

5 CONCLUSIONES Y TRABAJOS FUTUROS

En el presente TFG se han conseguido diseñar, desarrollar e implementar unos controladores sencillos y novedosos para el guiado de robots móviles. Repasando los objetivos enumerados al principio del proyecto, se han logrado alcanzar con éxito todos y cada uno de ellos. La consecución de dichos objetivos abre un abanico de trabajos futuros relacionados con el control de robots que van más allá de los robots móviles.

En primer lugar, se aplicarán los nuevos algoritmos de control de trayectorias y la generación de curvas de Bézier a la evitación de colisiones entre robots móviles. Para ellos se trabajará también en las comunicaciones entre robots.

Tras esto, se implementarán los controladores en otras plataformas con mayores prestaciones y complejidad que los Lego Mindstorms. Dichas plataformas serán el RBCar o el Summit, ambos disponibles en el Instituto de Automática e Informática Industrial (AI2).

Se desarrollarán los controladores basados en álgebra lineal para implementarlos en los robots paralelos del Departamento de Sistemas y Automática. Dichos robots se están utilizando para la rehabilitación de pacientes con problemas en la rodilla o el tobillo. Previamente se probarán en un entorno simulado, y se incluirán los integradores en el controlador con el fin de mejorar el comportamiento.

Otro posible trabajo futuro sería la aplicación de los controles y de las trayectorias generadas a vehículos no tripulados de vigilancia aérea. Con ello se podrían programar por ejemplo drones para realizar rondas de vigilancias periódicas en polígonos industriales u otras instalaciones.

6 ANEXOS

6.1 Anexo I: Descarga de Robot C y del firmware LEGO

En primer lugar se debe descargar el producto desde la propia web www.robotc.net/download/nxt/ e instalarlo en el ordenador.

La descarga del firmware será necesaria si acabamos de descargar o actualizar Robot C o si es la primera vez que se utiliza un robot. La descarga del mismo es sencilla: en la pestaña "robot" abrimos el desplegable y elegimos la opción "Download Firmware, Standard File".

Aparecerá una pestaña como esta, en la cual se indica el nombre del ladrillo. Simplemente tendremos que pulsar "F/W Download". Se abrirá otra ventana en la que aparece un archivo con formato .rfw, seleccionándolo comenzará la instalación del firmware en el ladrillo.

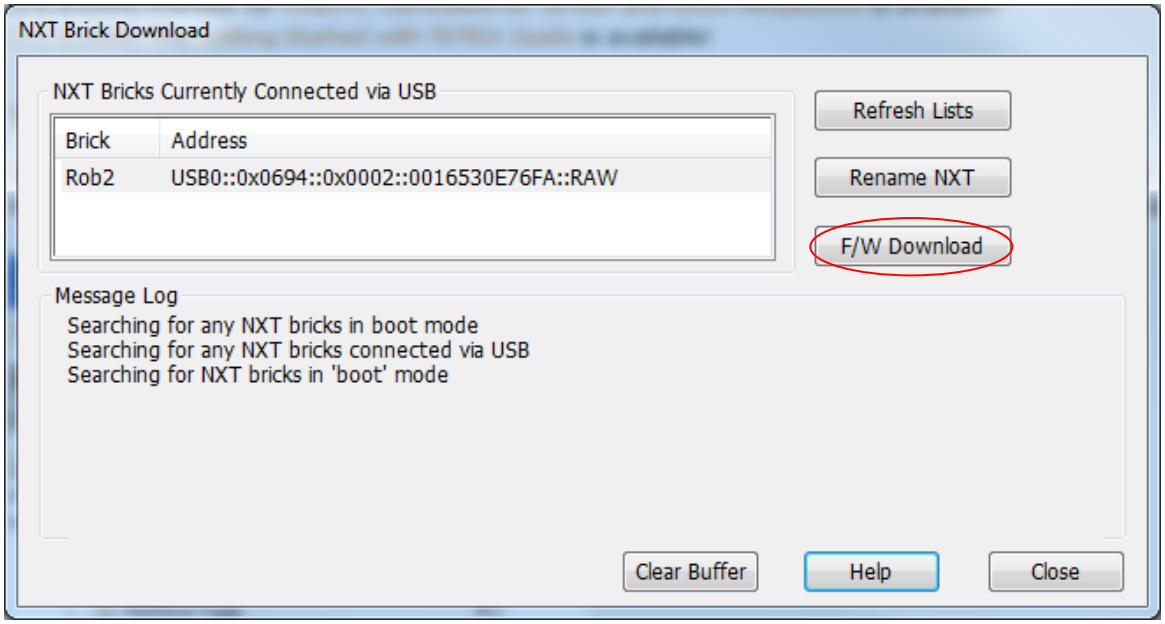


Figura 63. Firmware Lego

6.2 Anexo II: Programación de la función atan2

La función atan2 se diferencia de la función atan en que puede distinguir el cuadrante en el cual está el ángulo buscado. Mientras que la función atan da un ángulo que bien está en el primer o el cuarto cuadrante, la atan2 proporciona ángulos diferentes según sea la coordenada "x" e "y".

En robótica en general y más en concreto en el control de trayectorias, es imprescindible conocer la orientación del robot para poder aplicar correctamente las acciones de control pertinentes.

```
1. //******  Función ATAN2  ******
2.
3. float atan2(float y,float x)
4.       {
5.       float result; //lo que nos devuelve la función, es un ángulo
```

```
6.        float PI=3.141592653589793;
7.
8.          if(y==0 && x==0)
9.          { result=0.0; }
10.         if(y>0 && x>0) //primer cuadrante
11.           { result=atan(y/x); }
12.         if(y>0 && x==0) //***frontera primero-segundo
13.           { result=PI/2.0;}
14.         if(y>0 && x<0) //segundo cuadrante
15.           { result=atan(y/x)+PI; }
16.         if(y==0 && x<0) //***frontera segundo-tercero
17.           { result=PI;}
18.         if(y<0 && x<0) //tercer cuadrante
19.           { result=atan(y/x)-PI; }
20.         if(y<0 && x==0) //***frontera tercero-cuarto
21.           { result=-PI/2.0; }
22.         if(y<0 && x>0) //cuarto cuadrante
23.           { result=atan(y/x); }
24.         if(y==0 && x>0) //***frontera cuarto-primero
25.           { result=0.0; }
26.
27.       return result;
28.       }
```

6.3 Anexo III: Uso de la función Datalog

Como se ha explicado anteriormente, se ha utilizado la función Datalog para obtener los datos de la posición del robot Lego EV3. Gracias a esta herramienta se han podido obtener las figuras que confirman la viabilidad de los controladores en esta plataforma.

En primer lugar se declara como cadena de caracteres el nombre del fichero de texto en el que se escribirán las variables deseadas.

```
char * filename = "ejemplo.txt";
```

Posteriormente, se declara una variable tipo long, que se utilizará para abrir dicho fichero y se escribe en el fichero la frase que se desee como encabezado. También se debe calcular la longitud de la misma.

```
long fileHandle;

char * string1 = "controlador_Lego_EV3";  //Frase que queremos poner

int strlen1 = strlen(string1); //longitud de la frase

fileHandle = fileOpenWrite(filename); // Abrir el archivo
```

Una vez declaradas el resto de variables y justo antes de entrar en el bucle de control, se escribe lo siguiente:

```
fileWriteData( fileHandle, string1, strlen1 + 1);

if (!datalogOpen(0, 4, true))        //4 es el número de columnas

writeDebugStreamLine("Unable to open datalog"); //mensaje de error
```

En este fragmento de código se abre la función Datalog. Cabe mencionar que en la acción condicional se puede modificar el número de columnas y escribir el que se desee. En este caso, solamente se hará uso de cuatro, dos para las posiciones de referencia “x” e “y”, y otras dos para las posiciones reales del robot en los mismos ejes.

Para escribir los valores deseados en nuestro fichero de texto se debe utilizar la siguiente sintaxis, donde el número corresponde a la columna en la que se escribe:

```
datalogAddFloat(0, xref);
datalogAddFloat(1, yref);
datalogAddFloat(2, x);
datalogAddFloat(3, y);
```

Por último, se debe cerrar antes de finalizar el programa.

```
datalogClose();
```

7 BIBLIOGRAFÍA

[1] Ministerio de industria, energía y turismo (2015). Sistemas de automatización y robótica para las pymes españolas.
[2] CTFormación. http://ctformacion.com/industria-4-0-una-realidad-enfrentar/
[3] José Antonio Velásquez. La robótica y sus beneficios.
[4] Historia de los autómatas http://historiautomatas.blogspot.com.es/2010/06/grecia-iii-heron-de-alejandria.html
[5] Wikirobotica. http://wiki.robotica.webs.upv.es
[6] Piktochart. https://create.piktochart.com/output/17277976-leyes-de-asimov
[7] Robótica para todos. http://www.roboticaparatodos.es/tipos-de-robots-segun-su-cronologia/
[8] Conozcamos la robótica. http://conozcamoslarobotica.blogspot.com.es/p/generaciones-de-la-robotica.html
[9] Luis Ricardo Franco, Robótica. http://slideplayer.es/slide/3455360/
[10] Informática aplicada Ana blog. https://informaticaaplicadaanablog.wordpress.com/2016/01/15/la-5-generaciones-de-los-robotica/
[11] Cristina Urdiales García. Introducción a la robótica.
[12] RAE, diccionario de la lengua española http://dle.rae.es/?id=WYRlhzm
[13] Ángel Valera. LAC. Seminario introducción al control de robots.
[14] Ángel Valera. LAC. Diapositivas de clase.
[15] Paul's online math notes. http://tutorial.math.lamar.edu/Classes/DE/EulersMethod.aspx
[16] Gustavo Scaglia, Instituto de ingeniería química, Universidad de San Juan, Argentina. Estrategias para el seguimiento de trayectorias (PPT)
[17] Chein and Scaglia (2013) Trajectory Tracking controller design for unmanned vehicles.
[18] Gustavo Scaglia, Emanuel Serrano, Andrés Rosales and Pedro Albertos. Linear interpolation based controller design for trajectory tracking under uncertainties: Application to mobile robots.
[19] Wikipedia. Curvas de Bézier. https://es.wikipedia.org/wiki/Curva_de_B%C3%A9zier
[20] Ángel Valera. DISA-UPV. Modelado de curvas: aproximación de puntos.
[21] Ángel Valera. LAC. Seminario: Introducción a Lego NXT y RobotC.
[22] Wikipedia. https://es.wikipedia.org/wiki/Lego_Mindstorms
[23] RO-BOTICA. http://ro-botica.com/Producto/LEGO-MINDSTORMS-EV3-31313/
[24] Wikipedia. https://es.wikipedia.org/wiki/MATLAB
[25] RobotC. http://www.robotc.net/

DOCUMENTO II: EL PRESUPUESTO

Índice del presupuesto

8 Necesidad del presupuesto

En este documento se estudia el presupuesto del proyecto "Diseño y desarrollo de controles de robots mediante métodos numéricos basados en teoría de álgebra lineal. Aplicación a robots móviles". Como se ha explicado, uno de los objetivos del TFG es valorarlo económicamente, por ello es necesario realizar un presupuesto del mismo. A pesar de ser un trabajo académico, como cualquier proyecto de ingeniería debe incluir su presupuesto, cosa que facilitará una posible replicación futura del proyecto.

Para la realización del presupuesto se han tomado como referencia las "Recomendaciones en la Elaboración de Presupuestos en Actividades de I+D+I" (revisión del 2015) de la UPV de acuerdo al artículo 83 de la Ley Orgánica de Universidades.

9 Contenido del presupuesto

El contenido del presupuesto ha sido dividido en tres conceptos diferenciados: la mano de obra, la maquinaria y los materiales fungibles utilizados durante el desarrollo del proyecto.

9.1 Mano de obra

En primer lugar, se ha hecho una estimación de las horas empleadas por el graduado en Ingeniería en Tecnologías Industriales en cada una de las tareas que del proyecto.

Actividad	**Tiempo (h)**
Instalación Firmware Lego	2
Búsqueda bibliográfica	15
Simulaciones	100
Programación del código Lego NXT	40
Programación del código Lego EV3	10
Generación de trayectorias Matlab	15
Generación de trayectorias en C	5
Deducción de los controladores	25
Tutorías	20
Redacción de informes	10
Redacción y maquetación final	80
Reuniones vía Skype	8
TOTAL	**330**

Tabla 8. Estimación de horas Ingeniero en Tecnologías Industriales

Se han estimado también de forma aproximada las horas empleadas por el ingeniero senior y por el ingeniero consultor, siendo estas 50 y 15 respectivamente. Entre las tareas realizadas por el primero se pueden incluir tutorías, reuniones, revisión de documentos, etc. En cuanto al ingeniero consultor, sus principales tareas han sido las reuniones vía Skype y la aportación de información.

A continuación se muestra el cuadro de precios de la mano de obra.

Concepto	Coste (€/h)	Rendimiento (h)	Importe (€)
Ingeniero Tec. Industriales	20	330	6600
Ingeniero senior	50	50	2500
Ingeniero consultor	50	15	450
Coste total mano de obra			**9850**

Tabla 9. Mano de obra

9.2 Maquinaria y licencias software

Tras contabilizar la mano de obra, es necesario que nos centremos en la maquinaria. Sin embargo, al tratarse un proyecto en el que se han utilizado varios medios informáticos se tendrán que contabilizar también los importes de las licencias de software.

Para realizar este cuadro de precios se ha utilizado la siguiente fórmula:

$$Coste = \frac{coste\ del\ concepto * tiempo\ de\ uso\ (meses)}{12 * tiempo\ de\ amortización\ (años)}$$

Se han tomado los periodos de amortización siguientes:

- 12 años para equipos y útiles.
- 6 años para medios informáticos.

Concepto	Tiempo de uso (meses)	Periodo amortización (años)	Coste (€)	Coste total (€)
Ordenador portátil	5	6	489	33,96
Ordenador fijo	4	6	1200	66,67
Robot Lego NXT	2,5	12	549	9,53
Robot Lego EV3	2,5	12	400	6,94
Wifi	4	6	25	1,39
Windows 7	4	6	100	5,56
Windows 10	4	6	145	8,06
Robot C	2,5	6	30	1,04
Matlab y Simulink	4	6	500	27,78
Coste total maquinaria y licencias software				**160,92**

Tabla 10. Maquinaria y software

9.3 Material fungible

El material fungible utilizado en el proyecto es el siguiente:

Concepto	Unidades	Coste (€)	Coste total (€)
Folios	70	0,01	0,7
Encuadernación	2	4	8
Impresión	2	12	24
Bolígrafos	2	1	2
Memoria USB	1	10	10
Coste total material fungible			**44,7**

Tabla 11. Material fungible

10 Resumen del presupuesto

A partir de los resultados anteriores se ha calculado el presupuesto de ejecución material (PEM), tomándose un 13% de gastos generales y un 6% de beneficio industrial se ha obtenido el presupuesto de ejecución por contrata (PEC) y tras aplicar un 21% de IVA se ha calculado el presupuesto base de licitación (PBL).

Concepto	Coste total
Mano obra	9.850,00 €
Maquinaria y software	160,92 €
Material fungible	44,70 €
Presupuesto de Ejecución Material	**10.055,62 €**

Gastos generales 13%	1.307,23 €
Beneficio Industrial 6%	603,34 €
Presupuesto de ejecución por contrata	**11.966,19 €**

IVA 21%	2512,90
Presupuesto base de licitación	**14479,09**

Tabla 12. Resumen del presupuesto

El coste del proyecto asciende a un total de **CATORCE MIL CUATROCIENTOS SETENTA Y NUEVE EUROS CON NUEVE CÉNTIMOS.**