UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Industrial

Diseño de un control bifrecuencia en red para seguimiento de trayectorias en un robot de dos ruedas.

Trabajo Fin de Máster

Máster Universitario en Ingeniería Industrial (Acceso desde Grado I. Mecánica)

AUTOR/A: Román Ortega, Juan Antonio

Tutor/a: Cuenca Lacruz, Ángel Miguel

Director/a Experimental: CARBONELL LAZARO, RAFAEL

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE MASTER EN INGENIERÍA INDUSTRIAL

DISEÑO DE UN CONTROL BIFRECUENCIA EN RED PARA SEGUIMIENTO DE TRAYECTORIAS EN UN ROBOT DE DOS RUEDAS

AUTOR: JUAN ANTONIO ROMAN ORTEGA

TUTOR: ANGEL CUENCA LACRUZ

COTUTOR: RAFAEL CARBONELL LAZARO

Curso Académico: 2021-22

AGRADECIMIENTOS

En primer lugar, quiero agradecer al profesor Ángel Cuenca Lacruz la oportunidad que me ha dado de participar en este proyecto. Su implicación en la docencia, sus grandes dotes para la enseñanza y su forma de involucrarse con sus alumnos fueron los motivos que me hicieron sencilla la elección de tutor para mi Trabajo de Final de Máster. De nuevo muchas gracias por todo Ángel, siempre guardaré un muy buen recuerdo de estos meses.

También quiero dar las gracias a Rafael Carbonell Lázaro por su apoyo en todo el desarrollo del proyecto, ha sido un placer contar con alguien con su interés, sus ganas de transmitir conocimiento y de trabajar. Su ayuda me ha permitido ampliar mi conocimiento enormemente y llegar a un nivel de desarrollo superior al planteado en un inicio.

Para terminar, dar las gracias a toda mi familia, amigos y a mi pareja por todos los ánimos recibidos durante todos estos años, por todos los momentos buenos y no tan buenos en los que siempre han creído y confiado en mí. Gracias por haberme acompañado en todo este recorrido.

A todos vosotros, muchas gracias.

RESUMEN

En la actualidad existe un fuerte nivel de desarrollo dedicado al ámbito de los coches autónomos, los cuales poseen un mayor o menor grado de autonomía en función de la aplicación concreta. Estas aplicaciones poseen un punto común, ya que todas buscan ejecutarse con el mayor ahorro energético y de ancho de banda posible, siendo éste otro de los temas presentes hoy en día.

Basándose en este enfoque, en el proyecto se lleva a cabo el desarrollo y la implementación práctica de un modelo que permita obtener los mejores resultados posibles de seguimiento de trayectorias con el mayor ahorro posible. Los mayores gastos de energía y consumo de red son producidos por el funcionamiento continuo de sensores, envíos de información y cálculos de computación, y la forma más fácil de reducirlo es disminuir la frecuencia con la que se llevan a cabo. De forma inversamente proporcional, la calidad de la respuesta obtenida es mayor cuanto más rápido funcionen todos estos elementos, por lo que el objetivo es buscar una solución de compromiso entre ambas situaciones.

Para lograr un resultado favorable, se lleva a cabo un desarrollo por etapas, en cada una de las cuales se va mejorando la respuesta obtenida y minimizando los diferentes consumos. Las tres primeras se enfocan en el desarrollo de un modelo funcional básico, diseñando entre otros el controlador PI necesario. Las tres últimas en cambio, están enfocadas directamente en los objetivos mencionados, implementando un algoritmo de *pure pursuit* que permite una ejecución más natural de las trayectorias, un Filtro de Kalman Extendido para reducir los ruidos que afectan al modelo y, por último, convertir el controlador en uno *multirate* haciendo que funcione de forma lenta fomentando el ahorro, pero obtenga señales de control rápidas asegurando una solución óptima.

PALABRAS CLAVE

Vehículo autónomo; control en red; algoritmo de búsqueda pura; Filtro de Kalman Extendido; controlador PI multirate.

ABSTRACT

Currently there is a strong level of development dedicated to the field of autonomous cars, these have a greater or lesser degree of autonomy depending on the specific application. These applications have a common point, since all of them seek to run with the greatest possible energy and bandwidth savings, this being another of the issues present today.

Based on this approach, the project carries out the development and practical implementation of a model that allows obtaining the best possible trajectory tracking results with the greatest possible savings. The biggest costs of energy and network consumption are produced by the continuous operation of sensors, information sending and computational calculations, and the easiest way to reduce it is to reduce the frequency with which they are carried out. In an inversely proportional way, the quality of the response obtained is greater the faster all these elements work, so the objective is to find a compromise between both situations.

To achieve a favourable result, a development is carried out in stages, in each of which the response obtained is improved and the different consumptions are minimized. The first three focus on the development of a basic functional model, designing, among others, the necessary PI controller. The last three, on the other hand, are focused directly on the objectives, implementing a pure pursuit algorithm that allows a more natural execution of the trajectories, an Extended Kalman Filter to reduce the noise that affects the model and, finally, convert the controller into one multirate so that it works slowly promoting savings, but get control signals that work fast ensuring optimum performance.

KEY WORDS

Autonomous vehicle; network monitoring; pure pursuit algorithm; Extended Kalman Filter; PI multirate controller.

ÍNDICE GENERAL

MEMORIA 1

PRESUPUESTOS 53

ANEXOS Y BIBLIOGRAFÍA 69

ÍNDICE MEMORIA:

1. OBJETO DEL PROYECTO 7
2. INTRODUCCIÓN Y MOTIVACIÓN 8
3. BASE TEÓRICA 9
4. DESARROLLO Y EXPERIMENTACIÓN 14
5. CONCLUSIONES 52

ÍNDICE PRESUPUESTOS:

1. RECURSOS Y COSTES ASOCIADOS 57
2. ACTIVIDADES 62
3. CRONOGRAMA 64
4. PRESUPUESTO 65
5. RESUMEN DEL PRESUPUESTO 67

ÍNDICE ANEXOS Y BIBLIOGRAFÍA:

1. ANEXO 1: INICIALIZACIÓN E INTERPOLACIÓN 73
2. ANEXO 2: PURE PURSUIT 75
3. ANEXO 3: EKF 76
4. ANEXO 4: PI MULTIRATE 79
5. BIBLIOGRAFÍA 82

MEMORIA

DISEÑO DE UN CONTROL BIFRECUENCIA EN RED PARA SEGUIMIENTO DE TRAYECTORIAS EN UN ROBOT DE DOS RUEDAS

Autor: Juan Antonio Román Ortega

Fecha: septiembre de 2022

ÍNDICE DE LA MEMORIA:

1. OBJETO DEL PROYECTO 7
2. INTRODUCCIÓN Y MOTIVACIÓN 8
3. BASE TEÓRICA 9
3.1. Método Ziegler-Nichols 9
3.2. Cinemática directa e inversa 10
3.2.1. Cinemática directa 10
3.2.2. Cinemática inversa 11
3.3. EKF 12
4. DESARROLLO Y EXPERIMENTACIÓN 14
4.1. Escenario de trabajo 14
4.1.1. Sistema de control 14
4.1.2. Instrumentación 14
4.1.3. Entorno de trabajo 18
4.2. Cálculos previos 19
4.2.1. Funcionamiento en bucle abierto y bucle cerrado 19
4.2.2. Relación velocidad-excitación 21
4.2.3. Sistemas de envío de información y pérdidas 22
4.2.4. Valores en régimen permanente 23
4.3. Etapas de simulación 25
4.3.1. PI 25
4.3.2. Cinemática directa 28
4.3.3. Cinemática directa e inversa 31
4.3.4. Pure pursuit 31
4.3.5. EKF 42
4.3.6. PI multirate 47
4.4. Análisis de resultados 50
5. CONCLUSIONES 52

ÍNDICE DE FIGURAS DE LA MEMORIA:

Figura 1. Ejemplo gráfico y ecuaciones Ziegler-Nichols [4]. 9
Figura 2. Kit robot LEGO® Mindstorms® EV3 [11]. 15
Figura 3. Modelo 3D del robot LEGO ensamblado [8]. 16
Figura 4. Cámara cenital [12]. 16
Figura 5. Sistema de balizas [13]. 16
Figura 6. Placa AZDelivery ESP32 [14]. 17
Figura 7. Sensor de orientación IMU [15]. 17
Figura 8. Moqueta con esquema del recorrido. 18
Figura 9. Conjunto robot LEGO, baliza móvil, IMU y ESP32. 18
Figura 10. Datos robot LEGO en bucle abierto para una excitación constante. 19
Figura 11. Datos de excitación en bucle cerrado para Ziegler-Nichols. 20
Figura 12. Datos de velocidad angular en bucle cerrado para Ziegler-Nichols. 20
Figura 13. Relación excitación-velocidad del robot. 21
Figura 14. Pérdidas en la recepción al PC. 22
Figura 15. Pérdidas en el envío del PC. 23
Figura 16. Relación entre la excitación y el tiempo de simulación. 23
Figura 17. Campana de Gauss de los valores de excitación. 24
Figura 18. Esquema Simulink de funcionamiento con controlador PI. 25
Figura 19. Subesquema Simulink que contiene cada rueda del AGV. 25
Figura 20. Relación velocidad angular-tiempo en experimentación PI. 26
Figura 21. Relación error-tiempo en experimentación PI. 26
Figura 22. Relación excitación-tiempo en experimentación PI. 27
Figura 23. Esquema Simulink de funcionamiento cinemática directa. 28
Figura 24. Subesquema Simulink del bloque de cinemática directa. 28
Figura 25. Relación velocidad angular-tiempo en cinemática directa. 29
Figura 26. Relación velocidad lineal-tiempo en cinemática directa. 29
Figura 27. Trayectoria X-Y en cinemática directa. 30
Figura 28. Posicionamiento del vehículo en cinemática directa. 30
Figura 29. Esquema Simulink de cinemáticas directa e inversa. 31
Figura 30. Subesquema Simulink del bloque de cinemática inversa. 31
Figura 31. Ejemplo gráfico de la distancia de Look Ahead [9]. 32
Figura 32. Esquema Simulink con sensado por encoders. 33
Figura 33. Trayectoria en “S” X-Y con encoders. 34
Figura 34. Posicionamiento del vehículo figura en “S” con encoders. 34
Figura 35. Velocidades lineal y angular de referencia figura en “S” con encoders. 34
Figura 36. Trayectoria cuadrada X-Y con encoders. 35
Figura 37. Posicionamiento del vehículo en figura cuadrada con encoders. 35
Figura 38. Velocidades lineal y angular de referencia figura cuadrada con encoders. 36
Figura 39. Esquema Simulink con sensado por cámara. 36
Figura 40. Subesquema Simulink bloque cámara cenital. 37
Figura 41. Trayectoria X-Y con cámara cenital. 37

Figura 42. Posicionamiento en simulación con cámara cenital. 37
Figura 43. Desviaciones angulares en simulación con cámara cenital. 38
Figura 44. Subesquema Simulink cámara cenital con ángulo calculado. 38
Figura 45. Pérdidas de información al recibir de la cámara. 39
Figura 46. Esquema Simulink con sensado por balizas+IMU. 40
Figura 47. Subesquema Simulink de las balizas y el IMU. 40
Figura 48. Posicionamiento X-Y con balizas+IMU. 40
Figura 49. Posicionamiento a través de los encoders con balizas+IMU. 41
Figura 50. Seguimiento de la referencia en X-tiempo con balizas+IMU 41
Figura 51. Seguimiento de la referencia en Y-tiempo con balizas+IMU. 41
Figura 52. Esquema Simulink EKF ejecutado por el PC. 42
Figura 53. Subesquema Simulink bloque EKF. 43
Figura 54. Esquema Simulink EKF ejecutado por el robot LEGO. 43
Figura 55. Trayectoria X-Y EKF con T=0,1s. 44
Figura 56. Seguimiento de la posición en X-tiempo con EKF y T=0,1s 44
Figura 57. Seguimiento de la posición en Y-tiempo con EKF y T=0,1s. 44
Figura 58. Pérdida de datos del sistema de balizas+IMU con T=0,1s. 45
Figura 59. Pérdida de datos de la cámara cenital con T=0,1s 45
Figura 60. Trayectoria X-Y con EKF y T=0,2s. 46
Figura 61. Seguimiento de la posición en X-tiempo con EKF y T=0,2s 46
Figura 62. Seguimiento de la posición en Y-tiempo con EKF y T=0,2s. 46
Figura 63. Pérdida de datos del sistema de balizas con T=0,2s. 47
Figura 64. Pérdida de datos de la cámara cenital con T=0,2s 47
Figura 65. Esquema Simulink PI multirate. 48
Figura 66. Subesquema Simulink PI multirate. 48
Figura 67. Trayectoria X-Y con PI multirate. 49
Figura 68. Seguimiento de la posición en X-tiempo con PI multirate 49
Figura 69. Seguimiento de la posición en Y-tiempo con PI multirate. 49
Figura 70. Comparación gráfica de la trayectoria final y la nominal. 51

ÍNDICE DE TABLAS DE LA MEMORIA:

Tabla 1. Componentes robot LEGO® Mindstorms® EV3 [11]..... 15
Tabla 2. Datos extraídos de aplicar el método Ziegler-Nichols bucle abierto. 20
Tabla 3. Datos extraídos de aplicar el método Ziegler-Nichols bucle cerrado. 21
Tabla 4. Relación excitación-velocidad del robot. 21
Tabla 5. Media y desviación estándar de la distribución de excitación. 24
Tabla 6. Parámetros del controlador PI. 26
Tabla 7. Intervalos de fiabilidad. 27
Tabla 8. Comparación de los índices de coste. 51

1. OBJETO DEL PROYECTO

El objeto del presente proyecto consiste en llevar a cabo de forma experimental el control de un vehículo de dos ruedas, de forma que éste sea capaz de seguir todo tipo de trayectorias que se le soliciten, corrigiendo cualquier tipo de desvío o alteración que pueda sufrir. Todo ello con el mayor ahorro computacional, energético y de ancho de banda de la red posible sin que esto suponga pérdidas de calidad excesivas en los resultados obtenidos.

Este objetivo se puede subdividir en otros secundarios, de forma que su seguimiento a lo largo del desarrollo del proyecto sea más sencillo. Éstos son los siguientes:

- Obtención de un entorno de trabajo eficiente para la ejecución de todas las simulaciones que se llevan a cabo.

- Desarrollo de los programas y modelos correspondientes, de MATLAB y Simulink respectivamente, que serán utilizados en las experimentaciones. Dentro de este objetivo se encuentra el de incluir en dichos modelos todas las mejoras posibles que faciliten el control.

- Cumplimiento de unos estándares de calidad en los siguientes ámbitos: estabilidad, precisión y ahorro computacional, energético y de ancho de banda de la red.

Para asegurar el cumplimiento de estos objetivos se llevan a cabo revisiones periódicas, apoyándose en todo momento de un cronograma de eventos y actuaciones que evoluciona a la par. Finalmente se evalúa su cumplimiento en el apartado de conclusiones.

2. INTRODUCCIÓN Y MOTIVACIÓN

Los vehículos o robots autónomos se tratan de vehículos que por sí solos, sin necesidad de la intervención humana, son capaces de imitar nuestras capacidades de conducción, pudiendo percibir en mayor o menor grado el mundo que les rodea y actuar en consecuencia.

La idea del vehículo de guiado automático (AGV) ha cobrado relevancia durante las últimas dos décadas. No obstante, se lleva investigando sobre este tipo de tecnología desde los años 30, donde se llevaron a cabo varios ensayos en Estados Unidos y con unos fuertes avances en los años 90, con proyectos financiados por la Comisión Europea bajo del nombre de Proyecto Eureka.

En la actualidad, la Sociedad de Ingenieros de Automoción (*SAE international*), clasifican la conducción autónoma según seis niveles [1], los tres primeros (del 0 al 2) incorporan elementos de asistencia a la conducción, mientras que los siguientes tres (del 3 al 5) poseen características de automatización propiamente dichas. A continuación, se muestra cada uno de los niveles junto con sus principales características:

- Nivel 0: el conductor es quien lleva a cabo la conducción y una serie de sensores le facilitan dicha acción.
- Nivel 1: el conductor es quien realiza la acción de conducir, pero en este caso incluye sistemas que controlan la dirección, frenado o la velocidad.
- Nivel 2: el vehículo está capacitado para conducirse de forma autónoma pero el conductor debe estar alerta en todo momento por si debe actuar.
- Nivel 3: el vehículo analiza el entorno gracias a una serie de sensores y está capacitado para la toma de decisiones.
- Nivel 4: utiliza algoritmos de inteligencia artificial y el conductor ya no interviene en ningún momento, se le denomina pasajero.
- Nivel 5: igual que el anterior pero además compartiendo información con su entorno y utilizando el Internet de las Cosas (IoT).

El presente trabajo surge de la idea de llevar a cabo una comprobación experimental de los trabajos teóricos de investigación siguientes: [2] y [3]. En este se lleva a cabo un control remoto para un vehículo autónomo, atendiendo a algunos de los principales problemas como las pérdidas de información debido a la red, las no linealidades del AGV, los ruidos, etc.

Además de llevar a cabo el desarrollo experimental, se busca ensayar varios escenarios detallando las diferencias existentes entre ellos, así como sus principales ventajas e inconvenientes. Para ello se ejecutan diferentes modelos de simulación, en los que cambian el tipo de sensores encargados de captar la información entre otras cosas.

Finalmente, como último punto a tener en cuenta está el ahorro energético, de computación y de ancho de banda de la red. Para lograr este objetivo se debe buscar el punto óptimo de funcionamiento en el que la información sensada y procesada sea mínima, manteniendo la exactitud y calidad de la respuesta obtenida del AGV lo más alta posible.

3. BASE TEÓRICA

En este apartado se busca desarrollar una introducción a algunos conceptos de tipo teórico, matemático o físico, ya que son utilizados en los siguientes apartados de esta memoria.

Antes de comenzar con estos apartados es necesario llevar a cabo una breve introducción de algunas de las variables, de forma que se pueda asegurar una correcta comprensión de todos lo descrito de aquí en adelante. Las de mayor relevancia son las siguientes:

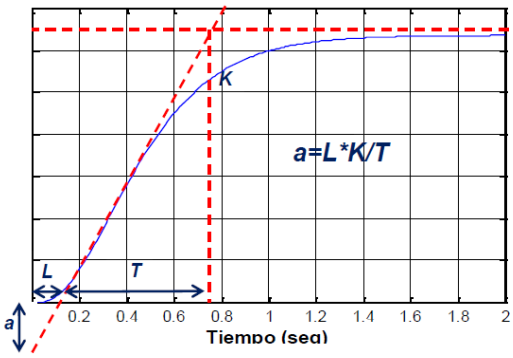
- Periodo (T): esta variable aparece en gran parte de la memoria y hace referencia al tiempo que existe entre dos instantes consecutivos de toma de datos, ejecución de acciones, etc. Puede aparecer sola o acompañada de un número o una N, que indicaría múltiplos de ese periodo. Suele estar colocado como superíndice de algunas variables.
- Instante de tiempo discreto (k): subíndice que indica si dicho valor es el de ese mismo instante (k), instantes anteriores (k-n) o instantes posteriores (k+n), siendo n cualquier número real y entero.

Por lo tanto, si aparece una variable con un superíndice T esto indica el periodo de muestreo o de cálculo de dicha variable, y si posee subíndice k indica el instante temporal al que pertenece dicho dato.

3.1. Método Ziegler-Nichols

Este método creado por John Ziegler y Nataniel Nichols sirve para sintonizar PID sin conocer la función de transferencia concreta, llevándose a cabo de forma experimental y empleando tablas a partir de las cuales se determinan los parámetros K_c, T_i y T_d requeridos [4]. Los valores característicos obtenidos no tienen por qué ser idóneos, pero son una buena aproximación que puede mejorar con un ligero reajuste.

Existen dos métodos para llevar a cabo la sintonización, el primero de ellos, el cual ha sido el más esclarecedor en este caso, se basa en la obtención de los parámetros *a* y *L* de la respuesta transitoria experimental en lazo abierto de la planta ante una entrada de tipo escalón, siendo estos dos parámetros los que se muestran en la *Figura 1*. El segundo método está basado en la respuesta oscilatoria experimental en lazo cerrado de la planta.



Controlador	K_c	T_i	T_d
P	$\frac{1}{a}$		
PI	$\frac{0.9}{a}$	$3L$	
PID	$\frac{1.2}{a}$	$2L$	$\frac{L}{2}$

Figura 1. Ejemplo gráfico y ecuaciones Ziegler-Nichols [4].

Por último, este método ha podido ser utilizado, puesto que la planta cumple las siguientes especificaciones:

- No posee integradores
- No posee polos dominantes complejos conjugados
- La respuesta no tiene oscilaciones
- Posee un retardo que forma una "s"

3.2. Cinemática directa e inversa

Algunos de los sensores utilizados proporcionan la información de forma que requiere algún tipo de conversión antes de ser utilizada. Dentro de estas conversiones es importante destacar dos de ellas: la cinemática directa y la cinemática inversa.

3.2.1. Cinemática directa

La cinemática directa [5] es utilizada para transformar variables de velocidad angular en otras de posición, de forma que, introduciendo los valores de velocidad angular de ambas ruedas, es posible obtener la posición X, Y y θ respecto del punto inicial para cualquier instante de tiempo.

A continuación, se muestran el conjunto de operaciones necesarias para llevar a cabo esta conversión, siendo los datos de partida la velocidad angular de cada una de las ruedas en radianes por segundo:

1) El primer paso de la cinemática directa consiste en obtener la velocidad lineal de cada una de las ruedas, lo cual se obtiene multiplicando la velocidad angular de cada rueda por su radio como se puede ver en las *ecuaciones 1 y 2*. El radio de la rueda derecha (r_r) es idéntico al de la izquierda (r_l), y en este proyecto tienen un valor de 0,028 cm.

$$(v_r)_k^T = r_r \cdot (\omega_r)_k^T \quad (1)$$
$$(v_l)_k^T = r_l \cdot (\omega_l)_k^T \quad (2)$$

2) A partir de la velocidad lineal de cada rueda es posible obtener las velocidades lineal y angular del robot, *ecuaciones 3 y 4*. Estos dos valores son de una elevada importancia, ya que interesa tenerlos controlados de forma que no superen unos determinados límites para cumplir unos requerimientos de estabilidad en las simulaciones. En el cálculo de la velocidad angular del robot es necesaria la distancia que existe entre el centro del vehículo y el punto de apoyo de ambas ruedas (b), que en este caso tiene un valor de 0,070 m.

$$v_k^T = \frac{(v_r)_k^T + (v_l)_k^T}{2} \quad (3)$$

$$\omega_k^T = \frac{(v_r)_k^T - (v_l)_k^T}{2 \cdot b} \quad (4)$$

3) Por último, para finalizar el estudio de la cinemática directa, se extraen los valores de posición en los ejes X e Y, así como de desviación angular, siendo todos los parámetros dimensionados con respecto a un sistema de ejes fijo que se define en la posición inicial del robot, más concretamente en el punto medio del eje de las ruedas motrices. Estas expresiones son las que se observan en las *ecuaciones 5, 6 y 7*.

$$X_k^T = X_{k-1}^T + v_k^T \cdot T \cdot \cos\left(\psi_{k-1}^T + \omega_k^T \cdot T\right) \quad (5)$$

$$Y_k^T = Y_{k-1}^T + v_k^T \cdot T \cdot \sin\left(\psi_{k-1}^T + \omega_k^T \cdot T\right) \quad (6)$$

$$\psi_k^T = \psi_{k-1}^T + \omega_k^T \cdot T \quad (7)$$

3.2.2. Cinemática inversa

La cinemática inversa [5] funciona justo al contrario que la cinemática directa, permitiendo obtener en cada instante de tiempo los valores de velocidad angular de cada una de las ruedas, a partir de valores de posición en X, Y y θ.

A continuación, se muestran el conjunto de operaciones necesarias para llevar a cabo esta conversión, siendo los datos de partida la posición en X e Y en metros y la desviación angular θ en radianes:

1) El primer paso consiste en obtener las velocidades lineal y angular del vehículo, *ecuaciones 8 y 9*, lo cual se obtiene a partir de los incrementos de posición en X e Y así como del incremento de la desviación angular de θ, todo ello para cada periodo de tiempo T.
2)

$$v_k^T = \frac{\sqrt{(X_k^T - X_{k-1}^T)^2 + (Y_k^T - Y_{k-1}^T)^2}}{T} \quad (8)$$

$$\omega_k^T = \frac{\psi_k^T - \psi_{k-1}^T}{T} \quad (9)$$

3) En segundo lugar, se calcula la velocidad lineal de cada una de las ruedas, empleando el valor de la distancia entre el centro del vehículo y el punto de apoyo de ambas ruedas (b), de valor 0,070 m, como se muestra en las *ecuaciones 10 y 11*.

$$(v_r)_k^T = v_k^T + b \cdot \omega_k^T \quad (10)$$

$$(v_l)_k^T = v_k^T - b \cdot \omega_k^T \quad (11)$$

4) Por último, se determina la velocidad angular de cada una de las ruedas dividiendo las velocidades lineales entre el radio de cada una de ellas (r_r y r_l), tal y como aparece en las *ecuaciones 12 y 13*.

$$(\omega_r)_k^T = \frac{(v_r)_k^T}{r_r} \quad (12)$$

$$(\omega_l)_k^T = \frac{(v_l)_k^T}{r_l} \quad (13)$$

3.3. EKF

El Filtro de Kalman (KF) es un algoritmo utilizado ampliamente en la actualidad para dar estimaciones de mayor precisión a la hora de obtener variables en presencia de incertidumbre, la cual suele coincidir con ruidos, [6] y [7]. El Filtro de Kalman Extendido (EKF) es el utilizado en este trabajo, ya que el funcionamiento del robot es no lineal y para ello se utiliza una matriz Jacobiana encargada de la linealización. La diferencia en la formulación matemática entre el filtro de Kalman normal y el extendido radica por lo tanto en esas no linealidades, siendo idénticas si las matrices F y G fuesen lineales.

En primer lugar, es necesario comenzar hablando de las expresiones que representan el modelo no lineal del proceso y a partir de las cuales se formula el filtro. Estas ecuaciones son las siguientes, definiéndose a continuación sus principales componentes: ξ es el estado del AGV, *ecuación 14*, el cual se compone de un vector con cinco componentes (velocidad angular de las ruedas derecha e izquierda, posición en X e Y y la rotación θ); n_1 y n_2 son ruidos del proceso y de la medición; u es la señal de control y z, *ecuación 15*, es la salida. En cuanto a los ruidos n_1 y n_2 se suponen ruidos gaussianos multivariados de media cero y cuya covarianza está determinada por Q_k^T y R_k^T respectivamente.

$$\begin{cases} \xi_k^T = f\left(\xi_{k-1}^T, (n_1)_{k-1}^T, u_{k-1}^T\right) & (14) \\ z_k^T = h\left(\xi_k^T, (n_2)_k^T\right) & (15) \end{cases}$$

En los siguientes puntos se define el procedimiento que lleva a cabo el EKF para ejecutar la predicción y corrección correspondiente, siendo su formulación matemática la siguiente:

- Predicción del siguiente estado, *ecuaciones 16 y 17*:

$$\hat{\xi}_{k|k-1}^{T} = f\left(\hat{\xi}_{k-1|k-1}^{T}(n_1)_{k-1}^{T}, u_{k-1}^{T}\right) \quad (16)$$

$$P_{k|k-1}^{T} = A_k^T P_{k-1|k-1}^{T}\left[A_k^T\right]^T + L_k^T Q_{k-1}^T\left[L_k^T\right]^{\top} \quad (17)$$

Donde A_k^T, *ecuación 18*, y L_k^T, *ecuación 19*, son las matrices Jacobianas encargadas de la linealización del proceso.

$$A_k^T = \left.\frac{\partial f}{\partial \tilde{\xi}}\right|_{\hat{\xi}_{k-1|k-1}^{T}, (n_1)_{k-1}^{T}, u_{k-1}^{T}} \quad (18)$$

$$L_k^T = \left.\frac{\partial f}{\partial n_1}\right|_{\hat{\xi}_{k-1|k-1}^{T}, (n_1)_{k-1}^{T}, u_{k-1}^{T}} \quad (19)$$

- Predicción de la salida futura y cálculo de la ganancia del EKF, *ecuaciones 20 y 21*:

$$\hat{z}_k^T = h\left(\hat{\xi}_{k|k-1}^{T}, (n_2)_k^T\right) \quad (20)$$

$$K_k^T = P_{k|k-1}^{T}\left[H_k^T\right]^{\top}\left(H_k^T P_{k|k-1}^{T}\left[H_k^T\right]^{\top} + M_k^T R_k^T\left[M_k^T\right]^{\top}\right)^{-1} \quad (21)$$

Donde H_k^T, *ecuación 22*, y M_k^T, *ecuación 23*, son las matrices Jacobianas encargadas de la linealización del modelo de salida.

$$H_k^T = \left.\frac{\partial h}{\partial \tilde{\xi}}\right|_{\hat{\xi}_{k|k-1}^{T}, (n_2)_k^T} \quad (22)$$

$$M_k^T = \left.\frac{\partial h}{\partial n_2}\right|_{\hat{\xi}_{k|k-1}^{T}, (n_2)_k^T} \quad (23)$$

- Modelo de corrección del estado, *ecuaciones 24 y 25*:

$$\hat{\xi}_{k|k}^{T} = \hat{\xi}_{k|k-1}^{T} + K_k^T\left(z_k^T - \hat{z}_k^T\right) \quad (24)$$

$$P_{k|k}^{T} = K_k^T R_k^T\left[K_k^T\right]^{\top} + \left(I - K_k^T H_k^T\right)P_{k|k-1}^{T}\left[\left(I - K_k^T H_k^T\right)\right]^{\top} \quad (25)$$

4. DESARROLLO Y EXPERIMENTACIÓN

4.1. Escenario de trabajo

Antes de comenzar con la explicación de las experimentaciones llevadas a cabo en el proyecto, es imprescindible establecer una breve explicación del escenario de trabajo, ya que tendrá una gran repercusión en las diferentes simulaciones como se verá en los siguientes apartados.

Para definir completamente el entorno de trabajo, se distinguen dos partes claramente diferenciadas: la primera de ellas hace referencia a las características del sistema de control utilizado, mientras que la segunda engloba todos los elementos físicos que han permitido su correcta ejecución.

Por último, se incluye un resumen del entorno de trabajo, indicando las interconexiones existentes entre los diferentes elementos que permiten llevar a cabo el control de forma correcta.

4.1.1. Sistema de control

El sistema de control empleado en las diferentes simulaciones es modificado en función de las características o requerimientos, entre los cuales se destaca la capacidad computacional al ser la que posee mayor relevancia.

En las primeras experimentaciones en las que está involucrado el robot LEGO, el sistema de control es de tipo convencional, donde el programa es cargado en su totalidad en el bloque EV3 del robot, el cual se encarga de la ejecución asumiendo toda la carga de computación. Con este tipo de configuración no se requiere el uso de una red para intercambio de información, de esta forma se puede asegurar que no existen ni retardos ni pérdidas de paquetes de datos.

Con la inclusión del Filtro de Kalman Extendido, los requerimientos computacionales aumentan considerablemente de forma que el bloque EV3 del robot LEGO es insuficiente y se requiere la utilización de un sistema de red para interconectar un ordenador, el LEGO y otros sistemas auxiliares como sensores. En este nuevo sistema de control el ordenador es el encargado de la parte de computación, recibiendo información de los diferentes sensores y enviando las acciones de control al bloque EV3 del robot LEGO, quien actúa sobre los servomotores que controlan ambas ruedas.

4.1.2. Instrumentación

A continuación, se muestra un listado con el conjunto de los elementos empleados en las diferentes simulaciones, así como algunas de sus principales características y, por último, un resumen del funcionamiento conjunto que llevan a cabo.

- Robot LEGO® Mindstorms® EV3:

Se trata de un kit de robótica, *Figura 2*, cuyo principal elemento es un ladrillo inteligente EV3, equipado con un procesador ARM9. Éste posee un puerto USB, a partir del cual se le puede proporcionar funciones WiFi y conexión a internet, así como un lector de tarjetas Micro SD y cuatro puertos de motor. Además, contiene tres servomotores y los siguientes tipos de sensores: ultrasónico, giroscópico, de colores y de contacto.

Figura 2. Kit robot LEGO® Mindstorms® EV3 [11].

De entre todas las piezas del kit se seleccionan algunas de ellas para llevar a cabo el montaje del modelo, dentro de las cuales hay que destacar:

Tabla 1. Componentes robot LEGO® Mindstorms® EV3 [11].

Bloque inteligente EV3		Puede conectarse al ordenador a través de USB, Bluetooth y WiFi, dependiendo este último de si la versión es o no compatible. Tiene una interfaz de seis botones, cuatro puertos de entrada y cuatro de salida.
Batería recargable CC EV3		Aporta mayor autonomía que las pilas AA, tiene una capacidad de 2050 mAh y posee un tiempo de carga de unas 3 horas.
Servomotor grande (x2)		Este servomotor utiliza la retroalimentación tacométrica para el control de velocidad con un grado de precisión. Además, tiene un sensor de rotación integrado.
Cables de conexión		Permiten conectar los dos servomotores con el bloque inteligente EV3.
Conjunto de ruedas		Dos pares de ruedas que poseen el mismo radio pero con diferentes características de adherencia para adecuarse a la superficie en cuestión.

Finalmente, una vez ensambladas todas las piezas y componentes correspondientes, el montaje posee el siguiente aspecto, *Figura 3*.

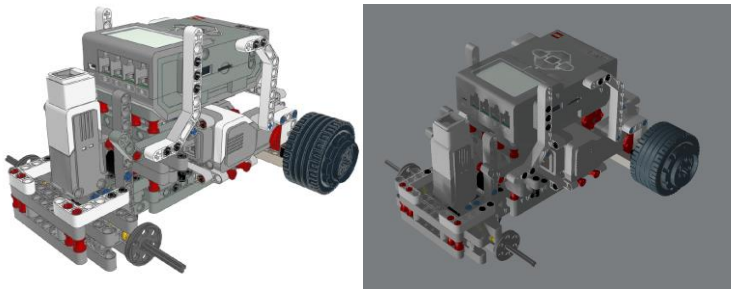


Figura 3. Modelo 3D del robot LEGO ensamblado [8].

- Cámara cenital [4K Full HD USB Webcam Auto Focusing]:

Esta cámara con conexión USB, ángulo de visión amplio de 120º y con alta calidad de imagen (4K y full HD), posee también enfoque automático, lo que permite utilizarla como cámara cenital o de difícil acceso y asegurar una correcta captación de imagen en todo momento. Una imagen de la misma se muestra en la *Figura 4*.

Figura 4. Cámara cenital [12].

- Sistema de balizas [Set HW v4.9-NIA]:

Este sistema de balizas para posicionamiento y navegación en interiores, *Figura 5*, está formado por un conjunto de cuatro balizas fijas, una baliza móvil y un receptor de señal. El conjunto proporciona datos de ubicación precisos, con unos márgenes de error de ± 2 cm más un porcentaje de la separación entre los diferentes dispositivos. La distancia entre las balizas no debe superar los 30 m y uno de sus principales usos es el posicionamiento de vehículos AGV.

El sistema de navegación está basado en balizas ultrasónicas estacionarias que están unidas por una interfaz de radio, calculándose la posición de la baliza móvil en función del retraso de propagación de la señal ultrasónica.

Figura 5. Sistema de balizas [13].

- Placa de desarrollo [AZDelivery ESP32 NodeMCU D1 R32]:

Esta placa de desarrollo con procesador ESP32 posee WiFi y Bluetooth, lo que permite establecer una conexión inalámbrica con el dispositivo. Su tensión de funcionamiento es de entre 5 y 12 V en corriente continua, su rango de funcionamiento en corriente es de 20 a 250 mA, posee un puerto micro USB, 4 MB de memoria flash, 20 entradas/salidas digitales de 3,3 V y 6 entradas analógicas. Una imagen de ésta es la que aparece en la *Figura 6*.

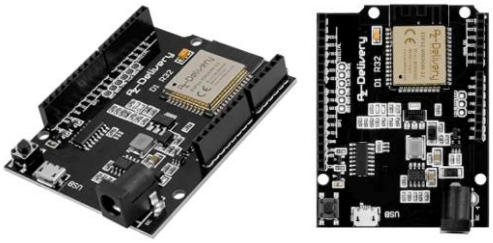


Figura 6. Placa AZDelivery ESP32 [14].

- Sensor de orientación [IMU BNO055 AHRS]:

Este sensor inteligente de orientación absoluta de 9 ejes, *Figura 7*, integra un acelerómetro triaxial de 14 bits, un giroscopio triaxial de 16 bits y un sensor geomagnético triaxial de 32 bits. Además, posee un microcontrolador ARM Cortex M0+ que ejecuta la fusión de sensores en un solo paquete. Requiere de alimentación entre 3 y 5 V.

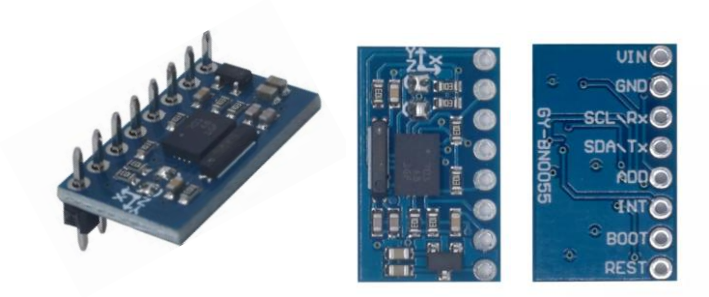


Figura 7. Sensor de orientación IMU [15].

- Otros elementos:

Dentro de este apartado se engloban el resto de los componentes necesarios para la realización de las simulaciones pero que no requieren de unas características específicas como son: el conjunto de baterías empleadas para cargar o dar tensión de forma continua a los diferentes dispositivos ya especificados, dos ordenadores y una moqueta que, junto con las ruedas del LEGO ya especificadas, permiten llevar a cabo las simulaciones evitando deslizamientos y otras situaciones indeseadas. En la siguiente imagen, *Figura 8*, se muestra la moqueta sobre la que se llevan a cabo las simulaciones en la cual se ha colocado el recorrido a seguir por el robot para facilitar su seguimiento.

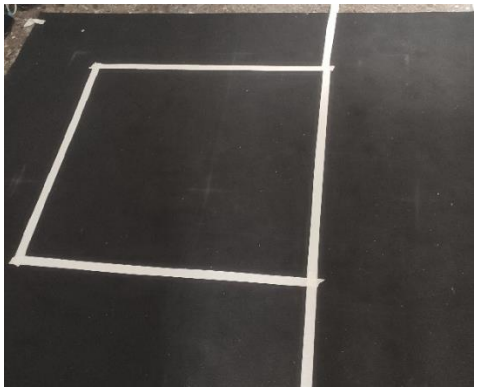


Figura 8. Moqueta con esquema del recorrido.

4.1.3. Entorno de trabajo

Este apartado busca mostrar la interconexión existente entre la instrumentación ya definida y cómo este conjunto de elementos consigue ejecutar el control deseado, presentando una figura del montaje final del AGV, *Figura 9*, donde se puede observar el EV3, el conjunto de piezas del LEGO, la baliza móvil, el IMU o sensor de orientación, la caja que almacena la placa ESP32 (caja blanca) y una batería que sirve de alimentación a estos dos últimos elementos.

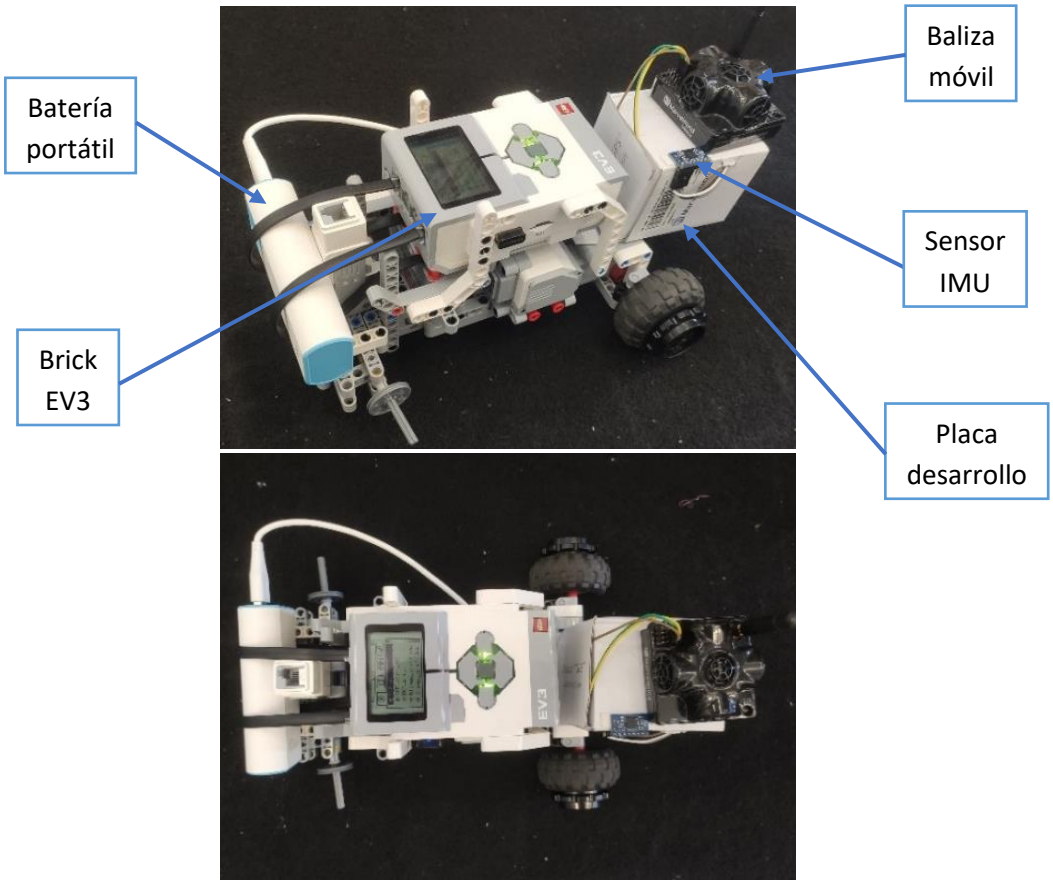


Figura 9. Conjunto robot LEGO, baliza móvil, IMU y ESP32.

4.2. Cálculos previos

En los siguientes subapartados se explica y justifica la obtención de algunos de los parámetros necesarios para el control del AGV, como son: el diseño del PI utilizado para el control, la relación entre la excitación enviada a los servomotores y la velocidad obtenida, el estudio de las pérdidas producidas en los intercambios de información y los valores en régimen permanente.

4.2.1. Funcionamiento en bucle abierto y bucle cerrado

Para llevar a cabo el diseño del PI encargado del control del AGV, se ejecutan dos ensayos, uno en bucle abierto y otro en bucle cerrado.

En el primero de ellos, se aplica una entrada de tipo escalón a los servomotores que controlan ambas ruedas, obteniendo la señal de los dos encoders situados en éstas, resultado que se muestra en la *Figura 10*. A partir de estos datos y empleando el método de Ziegler-Nichols [4] para respuesta ante entrada de tipo escalón, se obtienen gráficamente los valores de a, T, L y K de la *Tabla 2* y, con ellos, los parámetros del PI correspondiente, tal y como se detalla en el apartado 3.1. de este mismo documento.

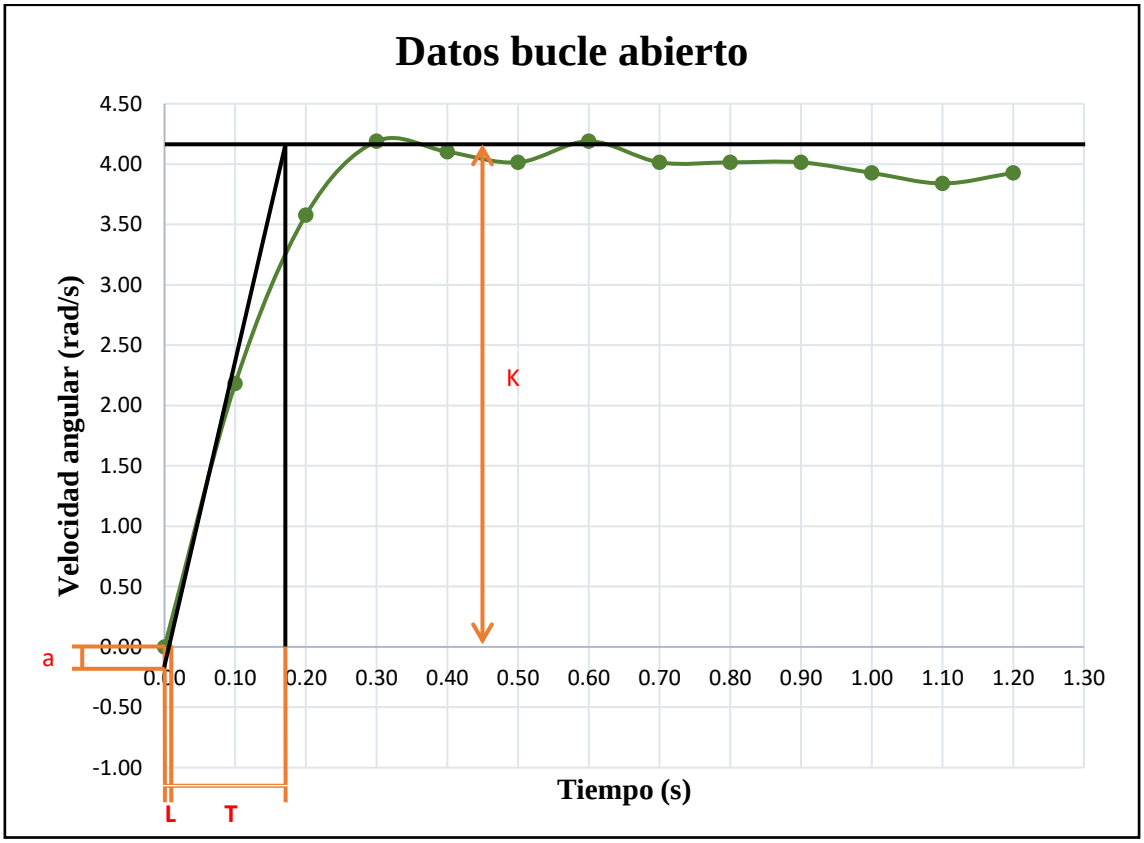


Figura 10. Datos robot LEGO en bucle abierto para una excitación constante.

Tabla 2. Datos extraídos de aplicar el método Ziegler-Nichols bucle abierto.

A	L	T	K		Kc	Ti	Td
0,25	0,01	0,17	4,20		3,60	0,51	0,00

De esta forma queda definido el controlador PI, cuyos valores serán reajustados manualmente si así se requiere, pero partiendo en todo caso de esta primera estimación obtenida. La función de transferencia obtenida adopta la siguiente forma, *ecuación 26*.

$$G_{PI}(s) = \frac{K_c\left(s + {}^{1}/_{T_i}\right)}{s} = \frac{3{,}60\left(s + {}^{1}/_{0{,}51}\right)}{s} \quad (26)$$

La segunda experimentación se lleva a cabo en bucle cerrado, en este caso se conducen a valor nulo las acciones integral y derivativa, elevándose poco a poco la ganancia proporcional hasta que el sistema oscile de forma mantenida. De esta forma se extraen los valores de la *Figura 11* y la *Figura 12*, y se procede de la misma forma que en el caso anterior, obteniendo gráficamente de la representación de los valores de ganancia crítica (Ku) y de periodo crítico (Pu), a partir de los cuales se calculan los parámetros de diseño del PI, *Tabla 3*.

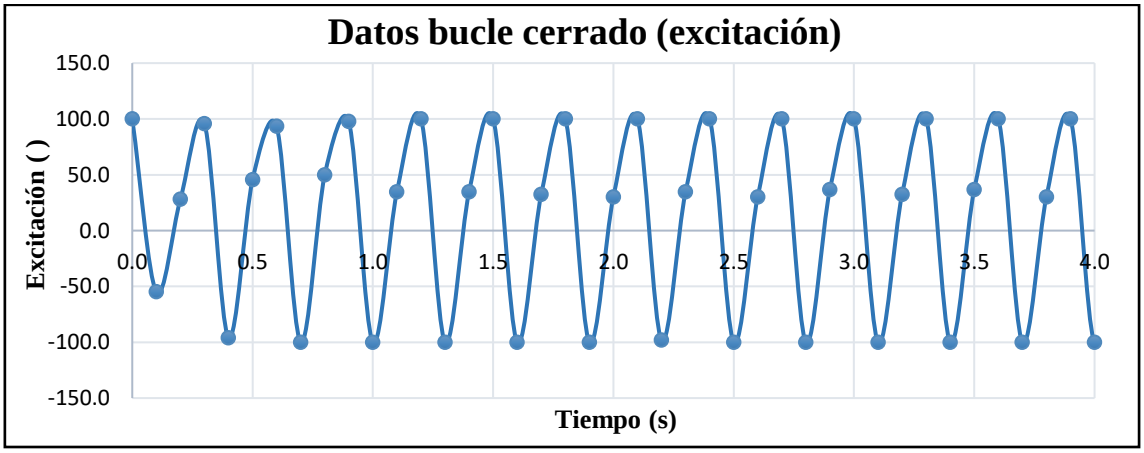


Figura 11. Datos de excitación en bucle cerrado para Ziegler-Nichols.

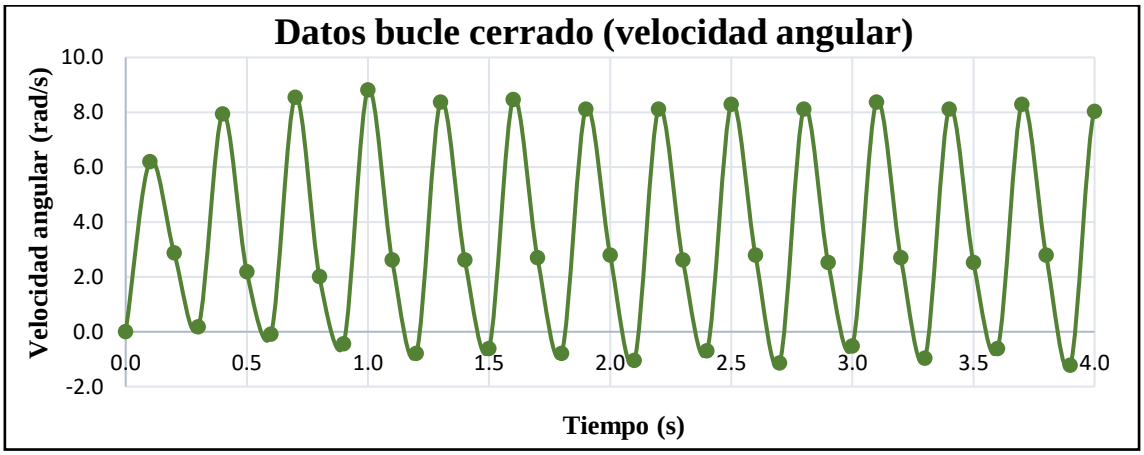


Figura 12. Datos de velocidad angular en bucle cerrado para Ziegler-Nichols.

Tabla 3. Datos extraídos de aplicar el método Ziegler-Nichols bucle cerrado.

Ku	Pu		Kc	Ti	Td
20	0,3		9,00	0,25	0,00

Aunque existe cierta discrepancia ente los valores esto no supone ningún inconveniente ya que, como se ha explicado antes, se llevarán a cabo reajustes manuales al ejecutar las primeras simulaciones.

4.2.2. Relación velocidad-excitación

El objetivo de este segundo cálculo es establecer una relación entre la señal de excitación que se le envía a los servomotores y la correspondiente velocidad angular y lineal de cada una de las ruedas. Para ello se ejecuta un programa en el que, cada cierto intervalo de tiempo se va aumentando el valor de la excitación en 10 unidades. De esta forma, y debido a que el valor de la excitación puede variar desde 0 hasta 100, se establecen diez intervalos de relación, los cuales aparecen representados en la *Figura 13* y en la *Tabla 4*.

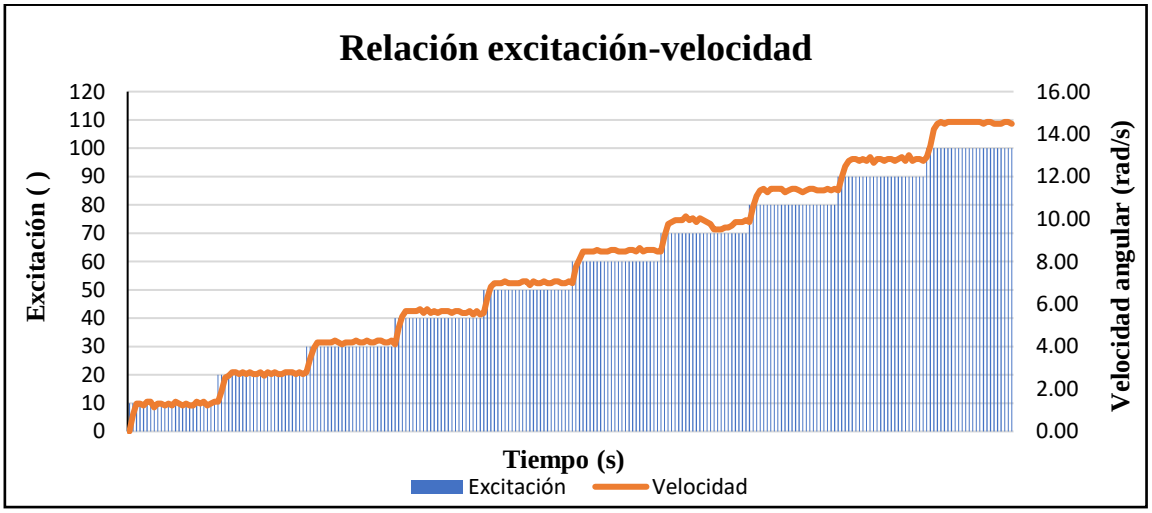


Figura 13. Relación excitación-velocidad del robot.

Tabla 4. Relación excitación-velocidad del robot.

Excitación ( )	10	20	30	40	50	60	70	80	90	100
Velocidad angular media (rad/s)	1,23	2,65	4,11	5,54	6,92	8,40	9,75	11,27	12,70	14,42
Velocidad lineal media (m/s)	0,03	0,07	0,12	0,16	0,19	0,24	0,27	0,32	0,36	0,40

4.2.3. Sistemas de envío de información y pérdidas

Debido a que parte de los envíos de información entre los diferentes componentes se lleva a cabo de forma remota (mediante protocolo UDP), existe la posibilidad de que se produzcan pérdidas de datos puntuales o continuadas. Por este motivo, se ejecuta un análisis previo de dichas redes, con el objetivo de asegurar un correcto funcionamiento y prevenir posibles fallos a futuro.

En las siguientes figuras se muestra para un intervalo de 50 segundos la cantidad de paquetes de datos que llegan incompletos para un periodo de envío y recepción de 0,1 segundos. En la *Figura 14* aparece la recepción de información en el PC desde los diferentes dispositivos y en la *Figura 15* la recepción en el LEGO.

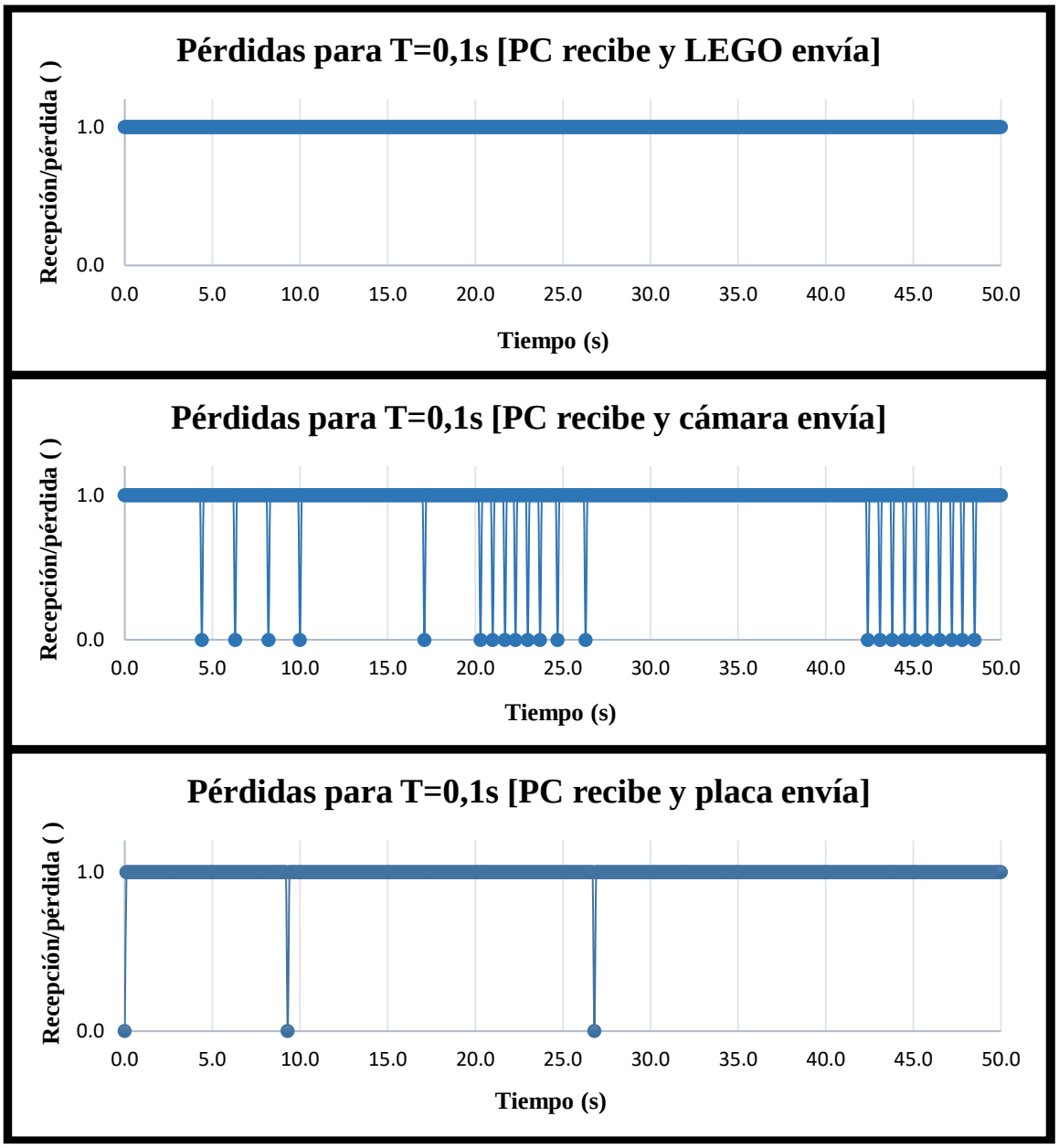


Figura 14. Pérdidas en la recepción al PC.

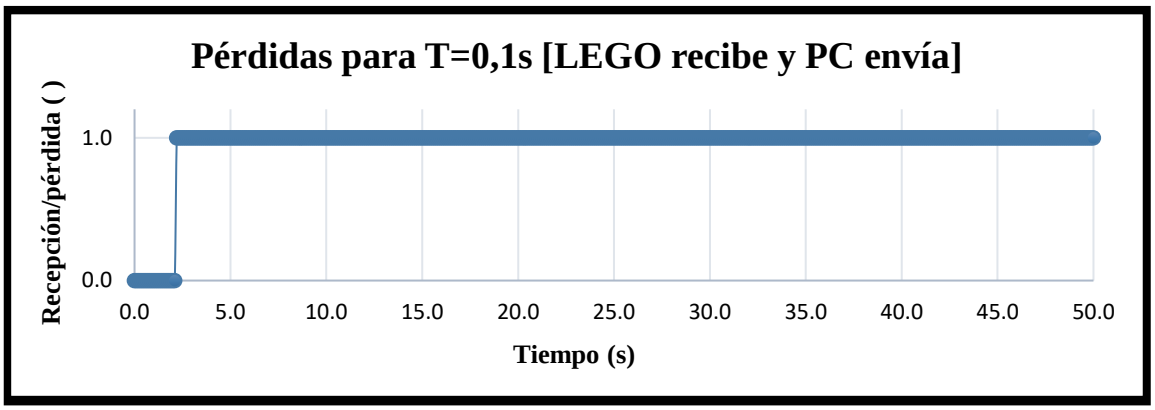


Figura 15. Pérdidas en el envío del PC.

Como se aprecia en las gráficas anteriores, las pérdidas de datos son prácticamente nulas, por lo que se estima que éstas no van a afectar al resultado obtenido y, por lo tanto, pueden despreciarse. No obstante, en los siguientes apartados se seguirá haciendo referencia a ellas ya que cuando se duplique el periodo se comprobará cómo afecta a estas pérdidas.

4.2.4. Valores en régimen permanente

Para finalizar los estudios previos al comienzo de la experimentación, se lleva a cabo un análisis de precisión en régimen permanente con el objetivo de determinar la exactitud con la que el AGV logra alcanzar la velocidad prevista. Para ello se ejecuta un programa simple en el que se solicita al vehículo que se mueva con un determinado valor de excitación, dicha curva se observa en la *Figura 16*.

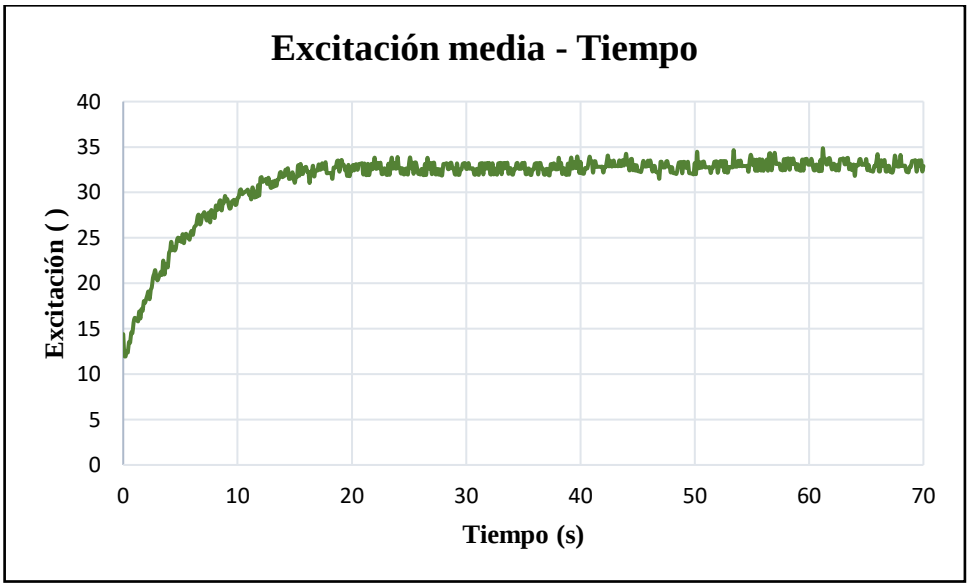


Figura 16. Relación entre la excitación y el tiempo de simulación.

A partir de estos valores, eliminando el periodo de establecimiento en el que la señal aún no ha alcanzado su valor final, se elabora el siguiente histograma, *Figura 17*, en el que se observan los valores de excitación y su repetitividad. Como se puede observar, existe cierta variación entre los valores a lo largo del tiempo de simulación, esto es debido a múltiples factores, entre los que se encuentra el rozamiento de las ruedas con el suelo. Los valores de desviación y media extraídos de esta prueba se encuentran recogidos en la *Tabla 5*.

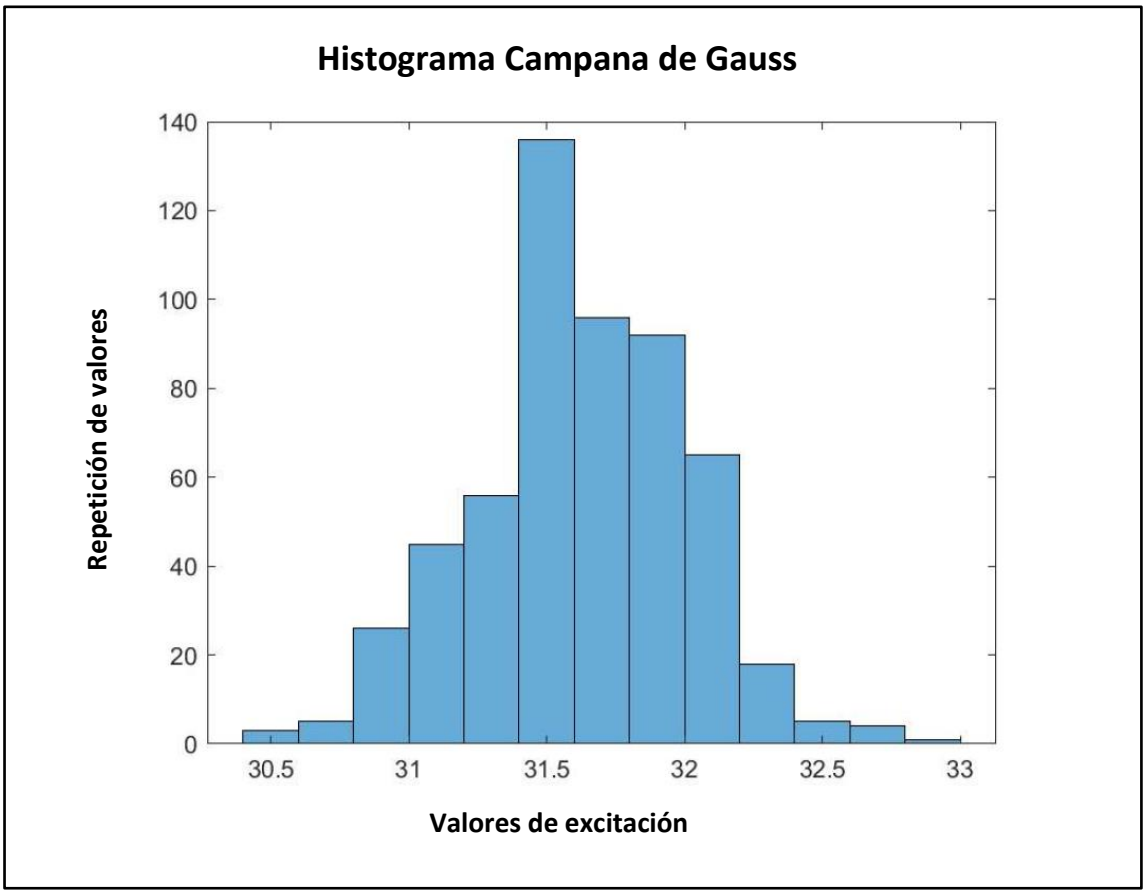


Figura 17. Campana de Gauss de los valores de excitación.

Tabla 5. Media y desviación estándar de la distribución de excitación.

Media	31,625
Desviación estándar	0,3819

4.3. Etapas de simulación

En este apartado se documentan de forma detallada cada uno de los pasos que se han llevado a cabo para conseguir el objetivo del proyecto, que es llevar a cabo el control de trayectorias del AGV de forma que ésta sea lo más precisa posible, evitando posibles irregularidades y con el mayor ahorro energético.

Las etapas en las que está dividido el desarrollo y la experimentación son un total de seis, comenzando por el funcionamiento del PI en bucle cerrado y terminando por el PI *multirate* con *pure pursuit* y EKF. El motivo principal de dividir en seis subapartados las simulaciones es la detección de fallos y la mejora continua, de forma que se puede ajustar con mayor precisión cada elemento por separado, haciéndolo finalmente de forma conjunta.

4.3.1. PI

Esta primera simulación tiene varios objetivos como son: comprobar que los parámetros de diseño del PI son correctos y ajustarlos, establecer unas velocidades y aceleraciones que permitan llevar a cabo el resto de los estudios dentro de unos límites de estabilidad y, por último, habilitar el entorno de estudio de la mejor forma posible para evitar deslizamientos y otros fenómenos indeseados.

En las siguientes imágenes, *Figura 18* y *Figura 19* , se muestra el esquema utilizado para esta primera simulación, donde se aplica una entrada escalón al sistema en bucle cerrado, midiéndose la salida de los dos encoders situados en cada uno de los servomotores.

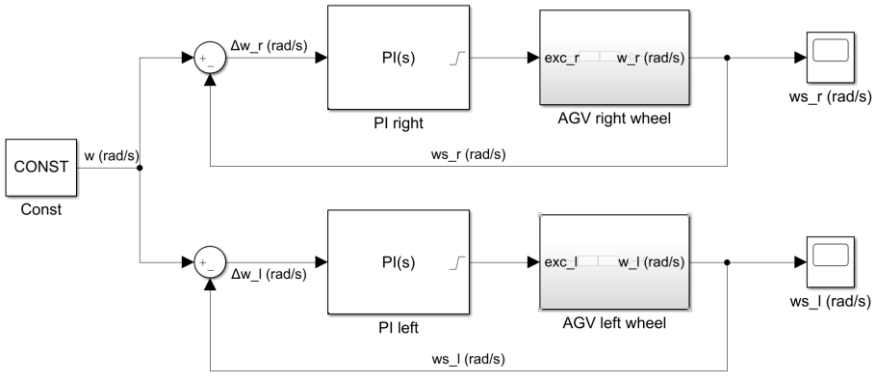


Figura 18. Esquema Simulink de funcionamiento con controlador PI.

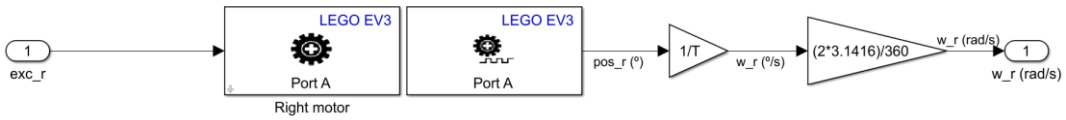


Figura 19. Subesquema Simulink que contiene cada rueda del AGV.

Con respecto a los parámetros del PI, se lleva a cabo un ligero reajuste manual a partir de los valores obtenidos previamente y los resultados de esta simulación, quedando finalmente los valores recogidos en la *Tabla 6*.

Tabla 6. Parámetros del controlador PI.

K_c	T_i	T_d
3,6	0,2	0,0

Como se aprecia en la *Figura 20*, *Figura 21* y *Figura* 22, los valores proporcional e integral son los adecuados, llevando el proceso a su valor final en un tiempo adecuado y de forma sobreamortiguada.

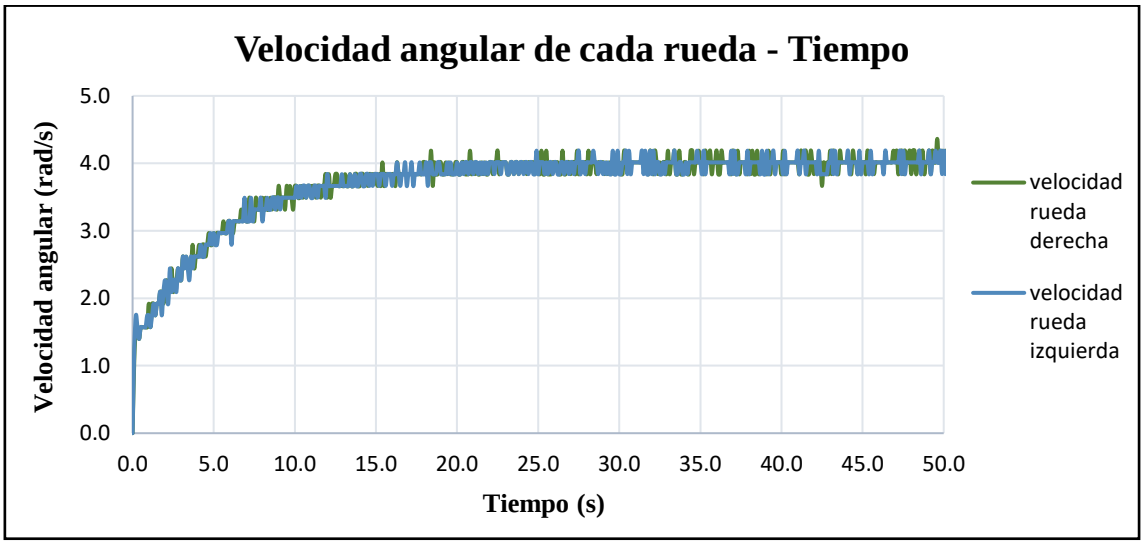


Figura 20. Relación velocidad angular-tiempo en experimentación PI.

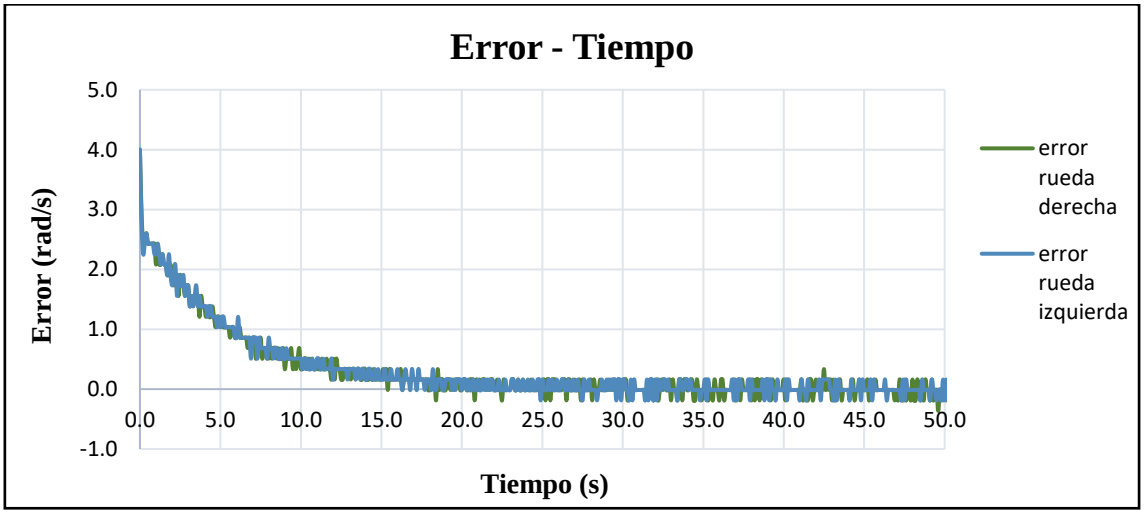


Figura 21. Relación error-tiempo en experimentación PI.

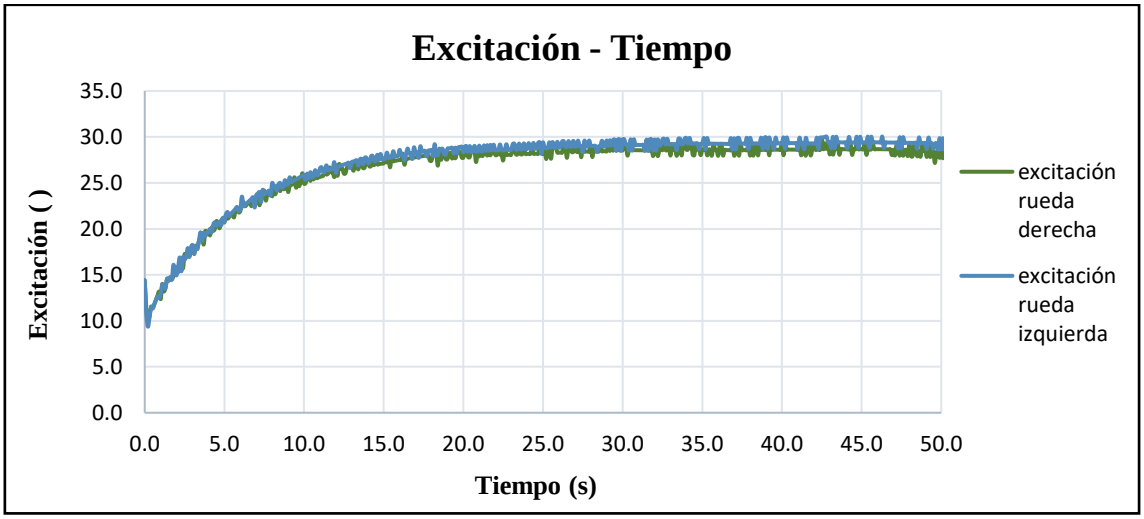


Figura 22. Relación excitación-tiempo en experimentación PI.

Para conseguir unos parámetros de velocidades y aceleraciones que permitan mantener unas condiciones de experimentación idóneas, se determina por observación directa unos límites entre los cuales se pueden asumir esas características. Estos valores numéricos son los mostrados en la *Tabla 7*. El estudio y análisis de estos parámetros es de vital importancia, ya que una de las suposiciones que se lleva a cabo a nivel teórico es que toda la dinámica del sistema es despreciable, lo cual es totalmente factible para unos valores de velocidades y aceleraciones bajos. Esta simplificación es muy positiva ya que conlleva un ahorro computacional muy elevado.

Tabla 7. Intervalos de fiabilidad.

	Velocidad lineal (m/s)	**Aceleración lineal (m/s^2)**	**Velocidad angular (rad/s)**	**Aceleración angular (rad/s^2)**
Intervalo de fiabilidad	De 0,08 a 0,16	Máximo de 0,65	De 0,50 a 5,00	Máximo de 15,00

Finalmente, dentro de los intervalos de confianza definidos se escogieron valores relativamente bajos, ya que estos se ajustan mejor a las especificaciones requeridas y conllevan un menor error por la simplificación realizada.

En relación con lo anterior, es necesario destacar la selección de unos elementos para el entorno de simulación que evitan o reduzcan la aparición de fenómenos indeseados como deslizamientos, rozamientos excesivos, detenciones, etc. Tras varios ensayos se llega a la conclusión de que las condiciones idóneas se alcanzan empleando una moqueta y las ruedas cuya cara de apoyo no es plana. Con estas condiciones y respetando los límites cinemáticos ya definidos es como se producen menos errores en la experimentación.

4.3.2. Cinemática directa

El siguiente paso, una vez obtenidos unos resultados acordes del PI en bucle cerrado, es la implementación de la cinemática directa [5] cuya parte teórica ha sido introducida en el apartado 3.2.1. de este documento. Este paso permite obtener para cada instante de tiempo la posición en X e Y, así como el ángulo girado tomando como 0 la posición de partida del vehículo.

El objetivo de este apartado es comprobar el funcionamiento de este nuevo bloque, siendo de vital importancia para experimentaciones posteriores. Además, gracias a la obtención de los parámetros de posición, es posible demostrar de forma gráfica la elección de los dispositivos explicada anteriormente.

Para llevar a cabo este análisis se emplea el siguiente programa de Simulink, *Figura* 23, donde se omite el PI para hacer más sencillo dicho estudio, introduciendo directamente al AGV la excitación deseada en bucle abierto y observando los resultados obtenidos.

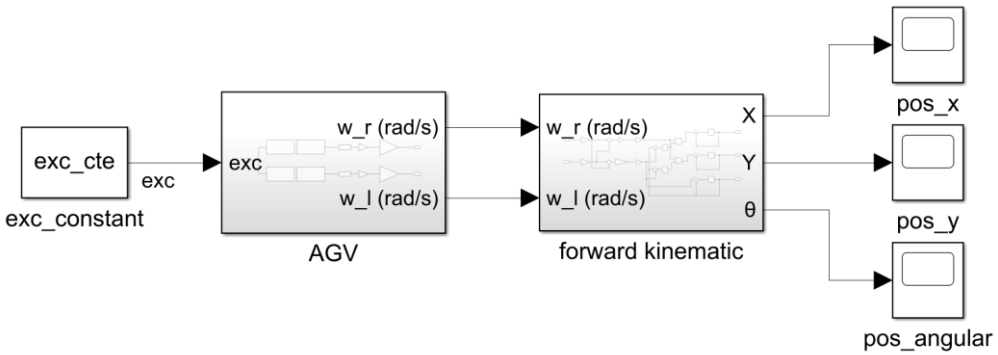


Figura 23. Esquema Simulink de funcionamiento cinemática directa.

En la *Figura* 24 se muestra el contenido del bloque de cinemática directa (*forward kinematic*) de la *Figura* 23, lo cual es una transcripción a lenguaje de bloques de lo expuesto en la parte teórica.

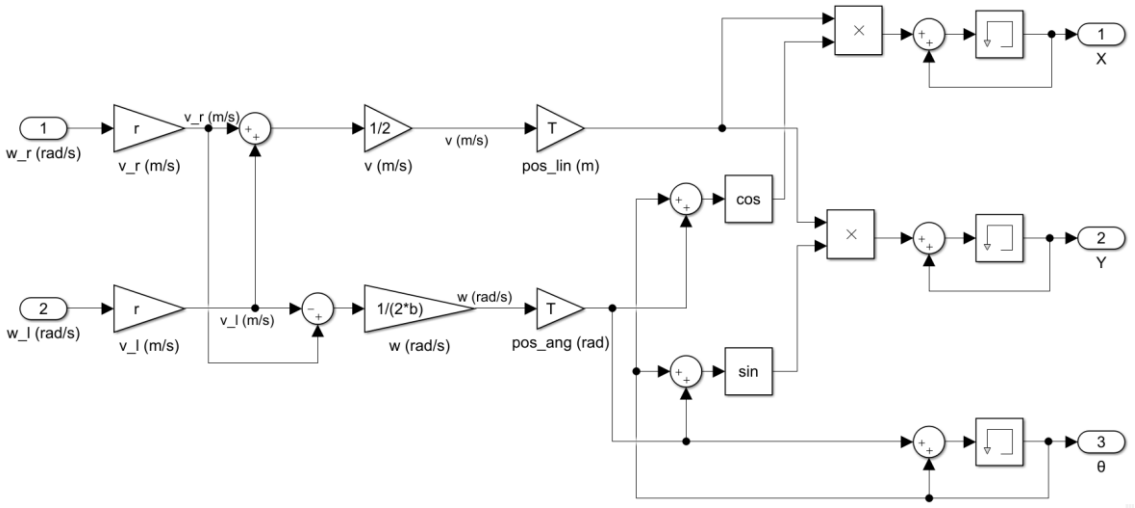


Figura 24. Subesquema Simulink del bloque de cinemática directa.

En las siguientes figuras se muestran los resultados obtenidos en una experimentación en la que no se utilizó la moqueta y estaban equipadas las ruedas planas.

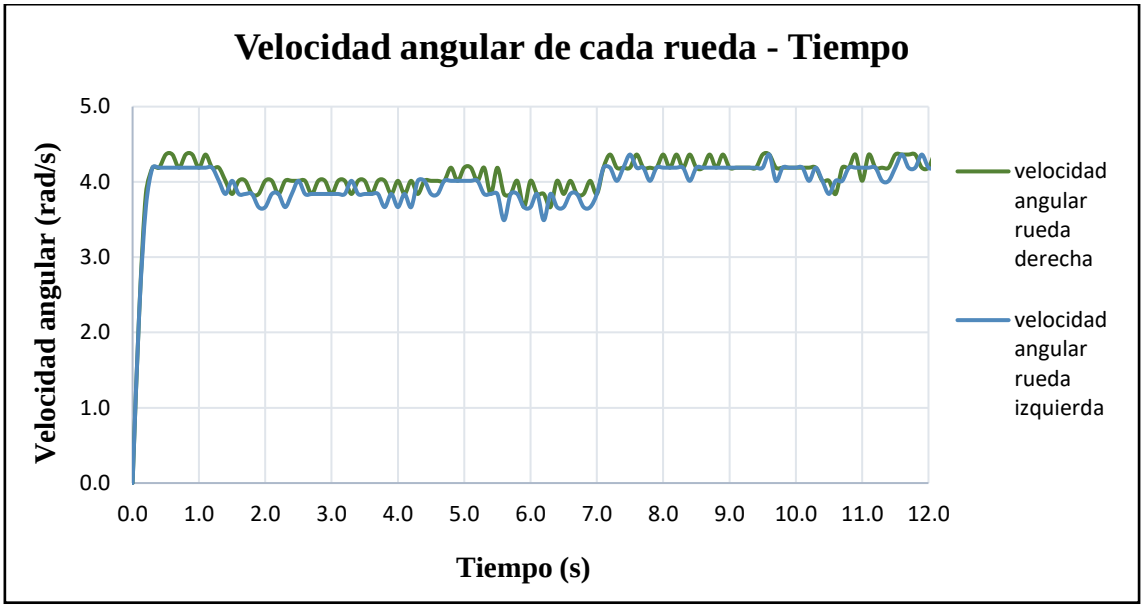


Figura 25. Relación velocidad angular-tiempo en cinemática directa.

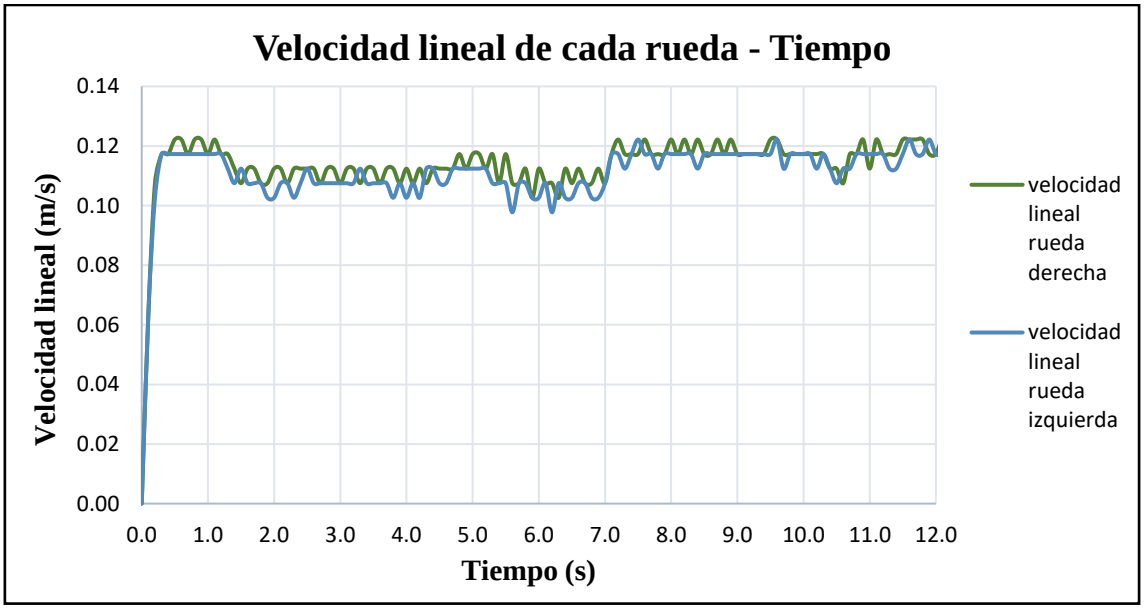


Figura 26. Relación velocidad lineal-tiempo en cinemática directa.

Como se puede apreciar en la *Figura 25* y la *Figura 26*, tanto la velocidad lineal como la angular de cada rueda sufren ligeras variaciones a lo largo del tiempo, pese a que sus valores están dentro del intervalo de estabilidad ya definido. Esto ocurre debido a una fricción inadecuada entre los tres puntos de apoyo y la superficie de ensayo, ocurriendo tanto deslizamientos como obstrucciones que afectan a las simulaciones.

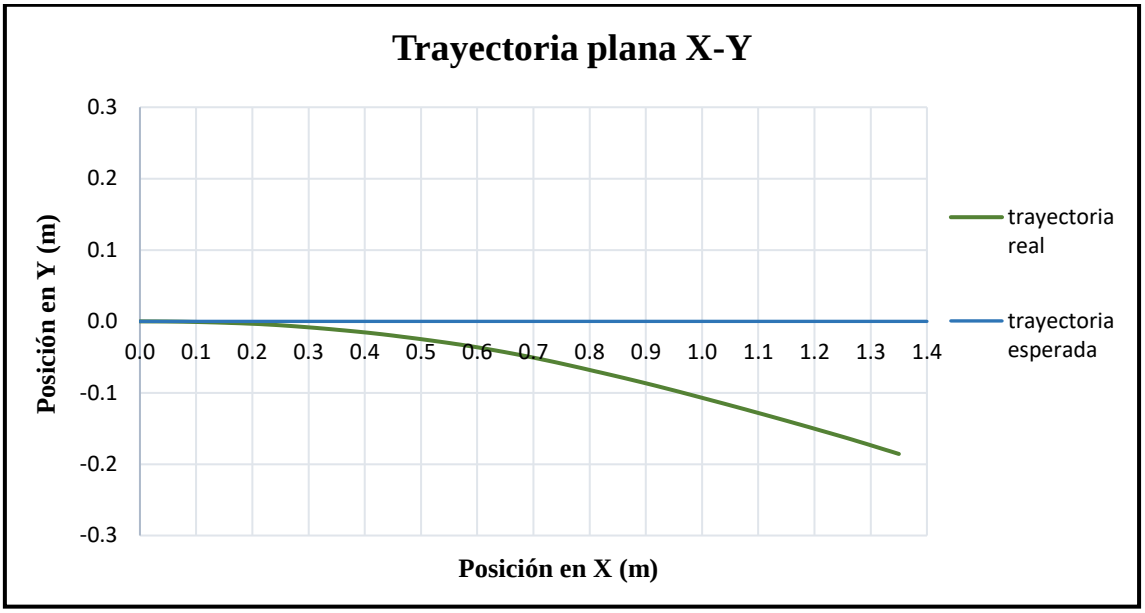


Figura 27. Trayectoria X-Y en cinemática directa.

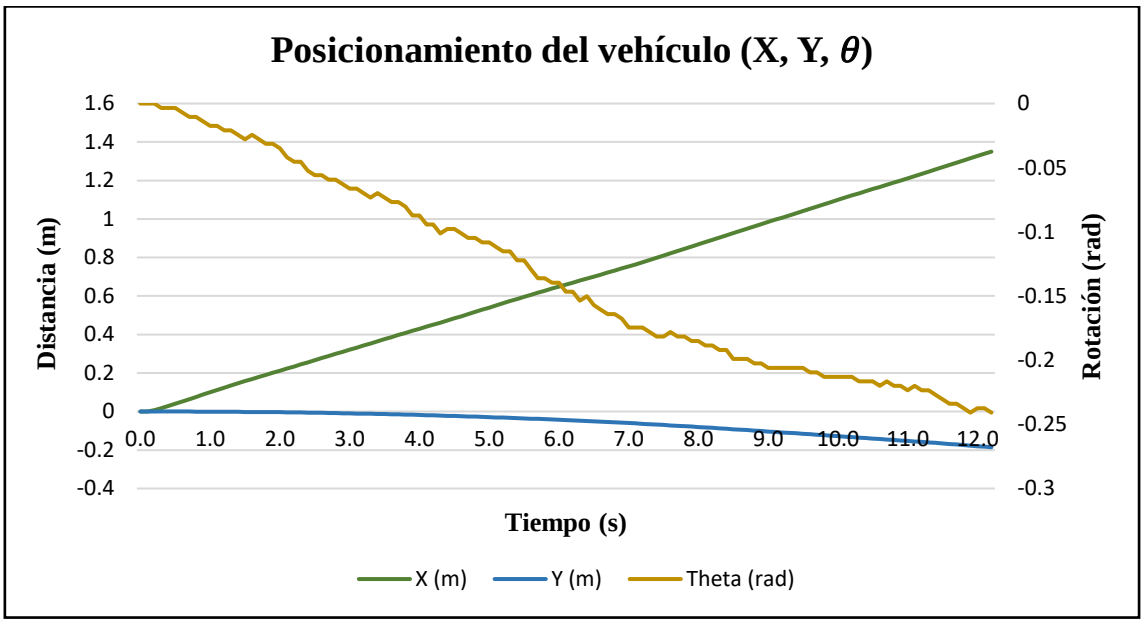


Figura 28. Posicionamiento del vehículo en cinemática directa.

En las gráficas anteriores se aprecia a simple vista el problema de fricción ya mencionado. La *Figura* 27 muestra las trayectorias real e ideal medidas con los encoders y transformadas en variables de posición, en la cual se puede ver como el AGV se desvía hacia valores negativos del eje Y, cuando realmente debería realizar una trayectoria recta entre los valores de 0 y 1,4 m del eje X. Por último, en la *Figura* 28 se muestran al igual que antes las variables X e Y con respecto al tiempo, pero se añade la variable de desviación angular (θ) donde se observa el movimiento ligeramente oscilatorio del AGV.

4.3.3. Cinemática directa e inversa

Este apartado tiene una finalidad similar al anterior, la cual consiste en implementar al modelo existente un componente de cinemática inversa (*inverse kinematic*) [5], cuya función es convertir una entrada de tipo posición en variables de velocidad angular que, transformadas en excitación por el PI, sean leídas y ejecutadas por el AGV. Este paso inverso al desarrollado en el apartado anterior es de gran utilidad, ya que lo más habitual es que un vehículo reciba datos de posicionamiento y no de velocidad.

En la siguiente imagen, *Figura* 29, se muestra el esquema global en Simulink, siendo la *Figura* 30 el bloque de cinemática inversa, en el cual entran variables de posición y desviación angular para dar como resultado las de velocidad angular necesarias para alcanzar el posicionamiento buscado. La parte teórica correspondiente se encuentra en el apartado 3.2.2.

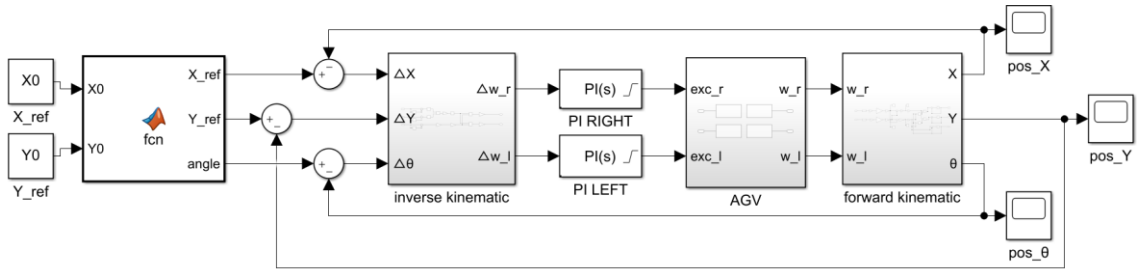


Figura 29. Esquema Simulink de cinemáticas directa e inversa.

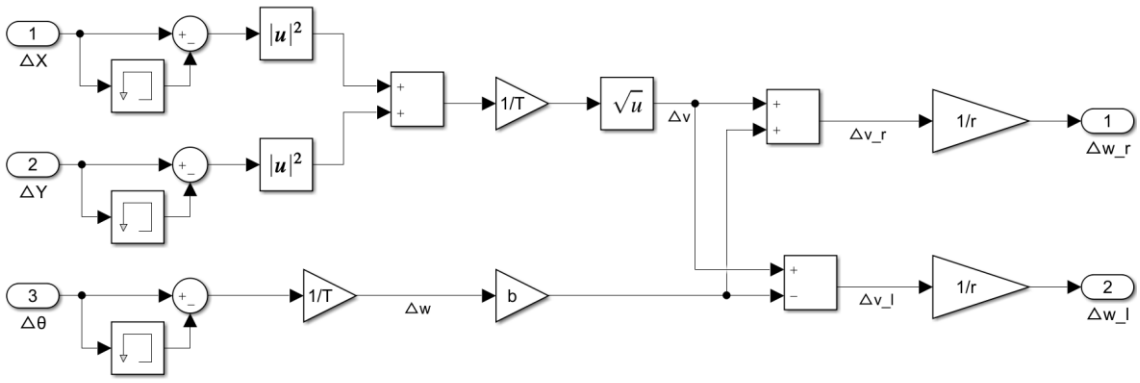


Figura 30. Subesquema Simulink del bloque de cinemática inversa.

4.3.4. Pure pursuit

Partiendo del modelo anterior, el siguiente paso consiste en añadir un algoritmo de *pure pursuit* o algoritmo de búsqueda pura o seguimiento de rutas [9]. Éste es el encargado de calcular la velocidad lineal y angular que debe tener el robot en la situación actual para alcanzar una determinada posición futura. Esta posición o punto futuro se encuentra en la trayectoria que se le ha solicitado seguir al robot, y se obtiene utilizando el *look-ahead distance*, variable de valor constante con unidades de longitud que define en cada instante el siguiente punto de la trayectoria que debe buscar el robot. Esta constante, representada con una L, es uno de los

valores que mayor importancia tiene en la obtención de resultados, ya que de ser muy pequeña el robot no tendría tiempo de reaccionar ante modificaciones en la trayectoria, y de ser muy grande se saltaría puntos de ésta, dirigiéndose directamente al final. El resto de las variables de entrada al *pure pursuit* son la velocidad lineal del robot, el estado o posición actual (variables X, Y y θ) y los valores de la trayectoria a seguir (definidos como referencia o ref).

Al introducir este nuevo bloque es necesaria la inclusión de nuevas variables como ya se ha explicado, algunas de estas como la velocidad lineal del robot ya han sido definidas y caracterizadas (intervalo numérico), pero otras son totalmente nuevas o bien han sido modificadas. Hay que recordar que la variable de velocidad lineal debe ser un valor constante que el robot tratará de mantener en todo momento, asegurando el cumplimiento de los límites de estabilidad definidos.

En cuanto al *look-ahead distance*, se trata de una variable que está directamente relacionada con la velocidad lineal del robot, siendo ambas directamente proporcionales. El motivo es claro, en caso de que la velocidad de movimiento del robot aumente, éste tardará menos tiempo en alcanzar el siguiente punto de referencia, lo que implica que dispondrá de un menor tiempo para adaptarse a los cambios en la trayectoria. Por ello, la solución radica en aumentar la distancia de *look-ahead*, de forma que el tiempo que tarde en alcanzar dicho punto sea aproximadamente el mismo para cualquier velocidad. En este caso, como la velocidad lineal utilizada es siempre la misma, se caracteriza el rango de valores del *look-ahead* de forma experimental en función de dicha variable. Un ejemplo gráfico se muestra en la *Figura* 31.

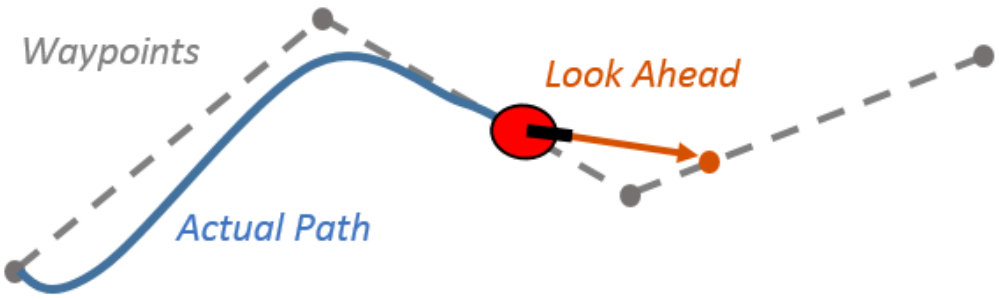


Figura 31. Ejemplo gráfico de la distancia de Look Ahead [9].

En valores de referencia introducidos existe un cambio con respecto a como se hacía hasta el momento, donde únicamente se introducían dos vectores con los tres o cuatro puntos de la trayectoria en cuestión. Con motivo de la introducción del *look-ahead distance* se decide aumentar la cantidad de valores que componen las trayectorias, para llevarlo a cabo se emplea una función de MATLAB a partir de la cual se obtienen puntos intermedios mediante interpolación. De esta forma, introduciendo únicamente los puntos finales de cada línea recta, se pueden definir todos los puntos intermedios que se deseen para aproximarse más a la solución deseada.

Al haber aumentado el número de variables y la forma en la que estas se introducen, se opta por elaborar un programa en MATLAB que se ejecute siempre de forma previa a la ejecución de las simulaciones, éste se muestra en el anexo y en él se encuentran todos los valores de las

constantes introducidas, así como el cálculo de los valores de la trayectoria mediante interpolación.

Por último, antes de definir el modo de funcionamiento del algoritmo de búsqueda falta por mencionar la última de sus variables de entrada que es el estado actual del vehículo o robot. Este estado está compuesto por un vector de tres variables, posición en X e Y y rotación angular θ, el cual se obtiene bien a partir de la cinemática directa como hasta ahora o con la utilización de un nuevo sensor que es la cámara cenital, cuyo funcionamiento aparece detallado más adelante en este mismo apartado.

Una vez definidas las entradas del *pure pursuit* se puede pasar a explicar cómo funciona, pudiendo encontrar su código en el anexo. Este algoritmo de búsqueda pura tiene como objetivo encontrar, para cada instante de tiempo T (periodo de funcionamiento), las velocidades lineal y angular necesarias para alcanzar el siguiente punto de referencia, el cual es definido por la distancia de *look-ahead*. A partir de los valores de referencia (posiciones de la trayectoria) y los de estado (posiciones del vehículo en cada instante), se calculan los diferenciales de posición en los ejes X e Y, que determinan la distancia que queda en ambos ejes hasta el siguiente punto de referencia. Con estos dos diferenciales se calcula la variable D, que define la distancia existente a dicho punto y la cual se compara con el *look-ahead* hasta que ésta es menor, momento en el que se actualizan los valores de velocidad para adaptarse a esa nueva referencia a alcanzar. Para la velocidad lineal es sencillo, ya que ésta debe ser constante y se iguala al valor que se introduce en el *pure pursuit*. En cambio, para el cálculo de la velocidad angular es necesario determinar previamente el valor de la variable Ka, la cual es función del ángulo de giro del robot en unidades de rad·m^{-1} y por último se multiplica dicho valor por la velocidad lineal.

4.3.4.1. Simulaciones con encoders

Con la inclusión del *pure pursuit* se consiguen llevar a cabo las primeras experimentaciones con trayectorias complejas. A continuación, se muestra el esquema utilizado, *Figura 32*, seguido de las dos primeras trayectorias que se completan de forma correcta:

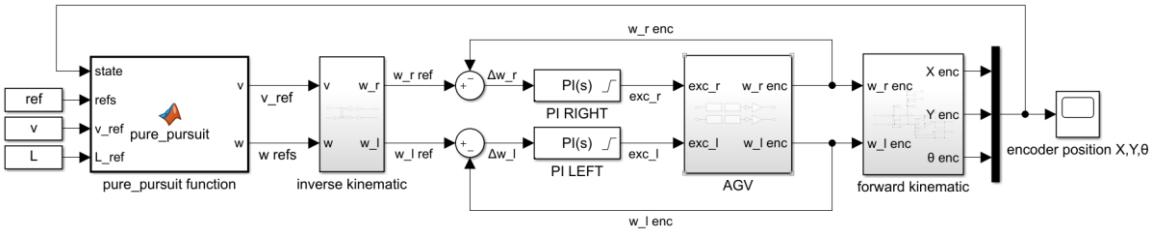


Figura 32. Esquema Simulink con sensado por encoders.

- Figura en forma de S: en esta primera simulación no se había definido aún la forma de detenerse del vehículo y, como se puede observar en las *Figura 33*, *Figura 34* y *Figura 35*, éste comienza a dar vueltas en torno al punto final. Pese a ello es una simulación de gran relevancia ya que es la primera que logra finalizarse correctamente.

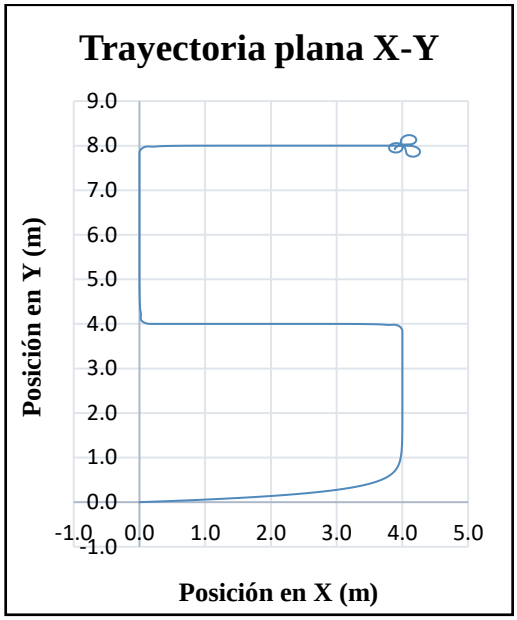


Figura 33. Trayectoria en "S" X-Y con encoders.

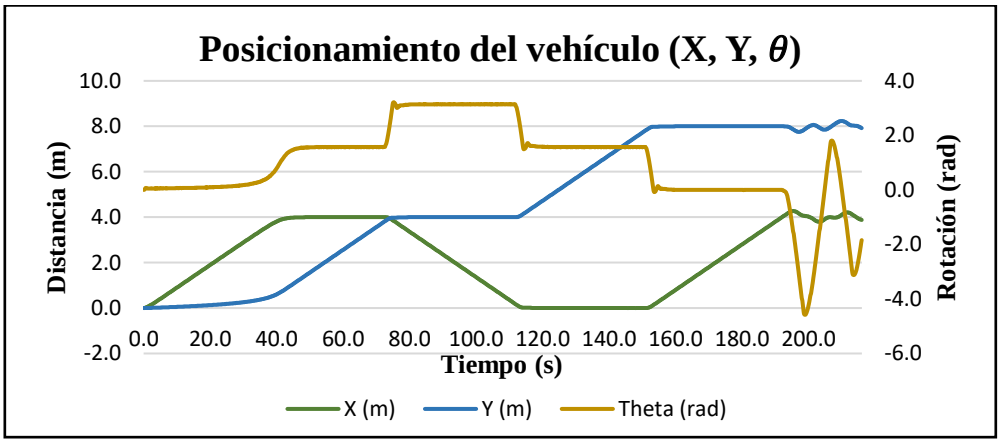


Figura 34. Posicionamiento del vehículo figura en "S" con encoders.

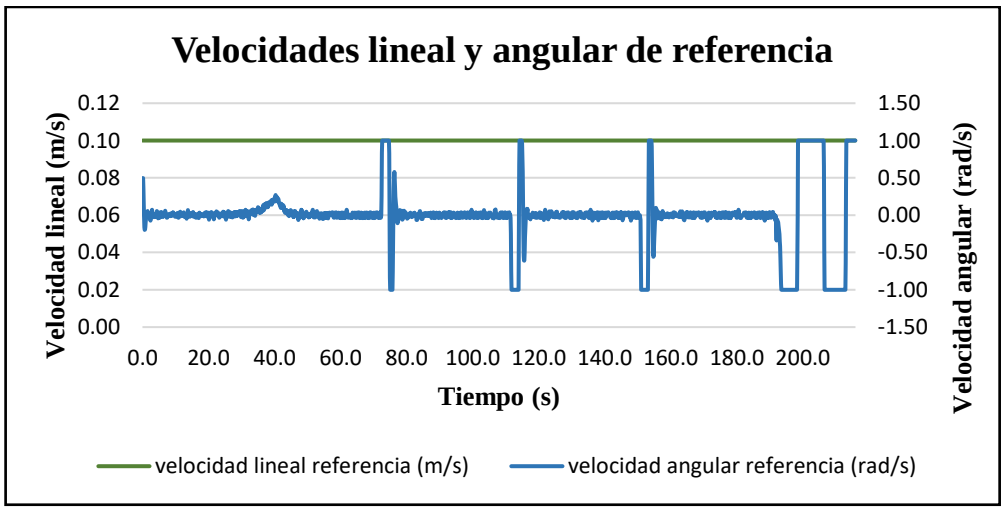


Figura 35. Velocidades lineal y angular de referencia figura en "S" con encoders.

- Figura cuadrada: esta simulación se lleva a cabo una vez programada la detención del robot al alcanzar el punto final. El motivo por el que se modifica la forma de la trayectoria es que se busca reducir el recorrido, limitando de esta forma el tiempo que tarda en ejecutarla, es por lo que a partir de este momento todas las simulaciones seguirán una trayectoria cuadrada. En la *Figura 36*, la *Figura 37* y la *Figura 38* se muestran los resultados obtenidos.

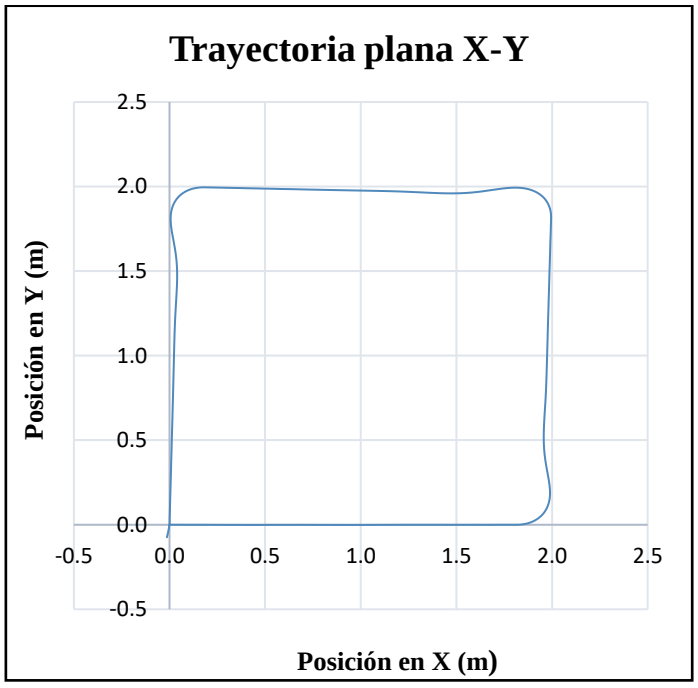


Figura 36. Trayectoria cuadrada X-Y con encoders.

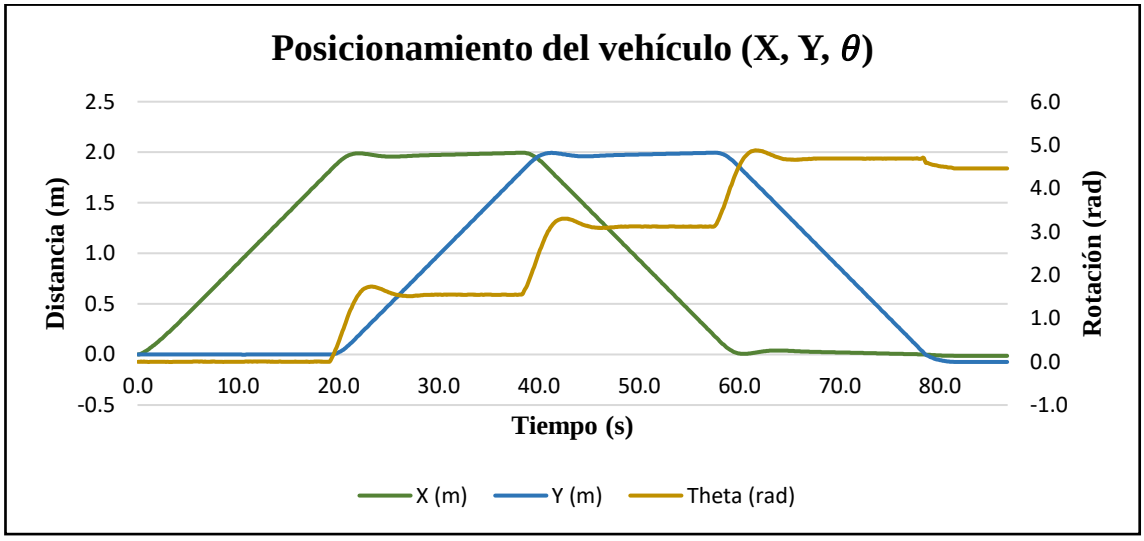


Figura 37. Posicionamiento del vehículo en figura cuadrada con encoders.

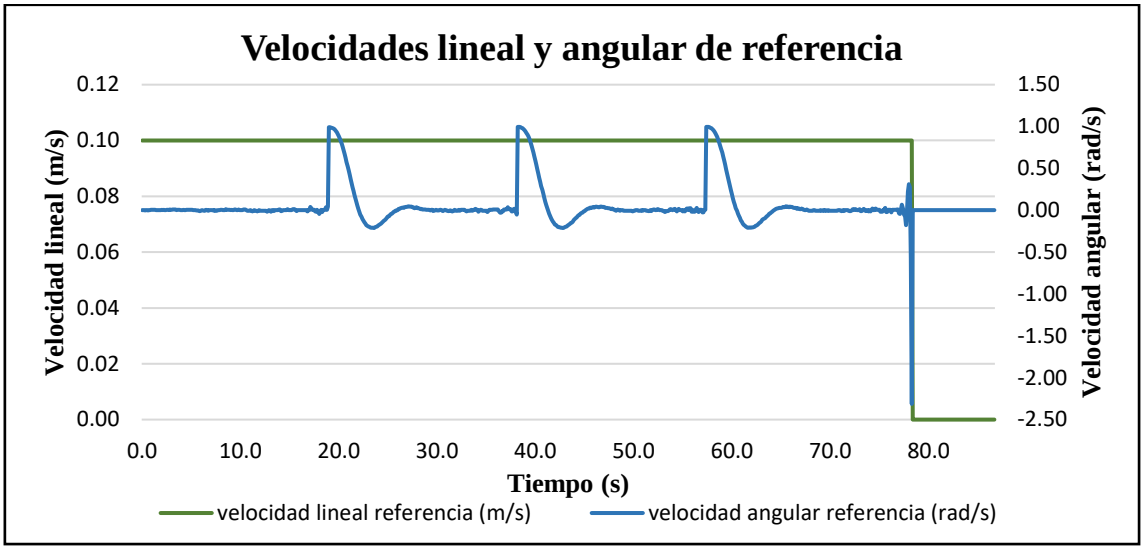


Figura 38. Velocidades lineal y angular de referencia figura cuadrada con encoders.

4.3.4.2. Simulaciones con cámara

Tras llevar a cabo las simulaciones empleando los encoders, se aprecia un problema en la toma de datos. Al estar situados ambos sensores en los servomotores de ambas ruedas, cuando existen deslizamientos o bloqueos el vehículo cambia su orientación, pero no son capaces de captarlo. Esto da lugar a problemas en cuanto al posicionamiento, pese a que gráficamente parece que las trayectorias son perfectas, los puntos por los que circula el vehículo pueden no coincidir en función del rozamiento entre otros factores.

Por este motivo se decide introducir un nuevo sensor que es la cámara cenital, el cual da tanto los valores de posición en X e Y como la desviación angular θ. Con este dispositivo que proporciona correctamente la posición en todo instante se solucionan las irregularidades que existen entre el funcionamiento que debería tener el dispositivo (lo que se grafica) y lo que ocurre realmente. El esquema de este nuevo modelo es el que se muestra en la *Figura* 39.

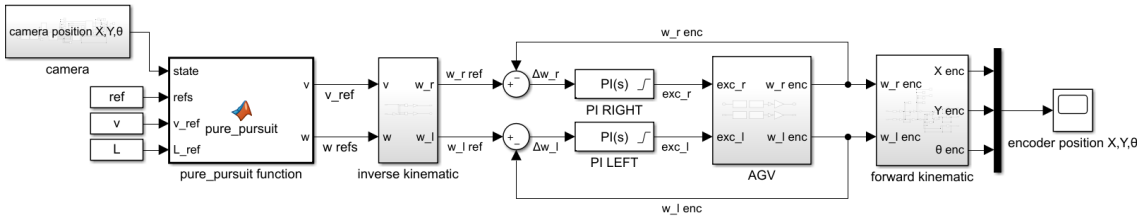


Figura 39. Esquema Simulink con sensado por cámara.

El simulink correspondiente a la cámara es el que se muestra en la *Figura* 40. En esta imagen se observan los bloques encargados de solicitar la información a la cámara y de recibirla, los cuales utilizan un protocolo UDP, el cual es escogido ya que no existen pérdidas relevantes como se ha demostrado anteriormente.

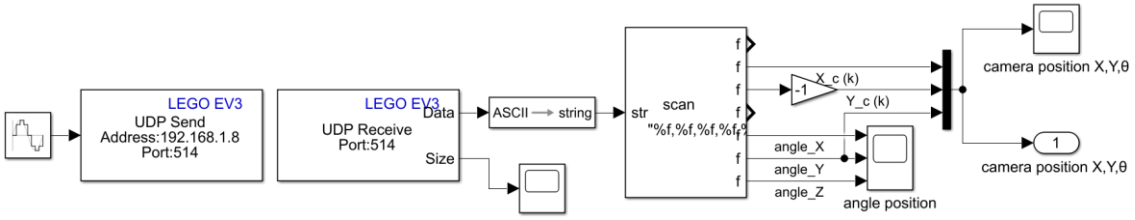


Figura 40. Subesquema Simulink bloque cámara cenital.

Gracias a este nuevo sistema de toma de información se logran obtener los resultados mostrados en la *Figura 41*, la *Figura 42* y la *Figura 43* donde, como se puede ver a simple vista, no coincide lo obtenido con los encoders a lo captado por la cámara, lo cual es debido a esas diferencias en el rozamiento ya definidas.

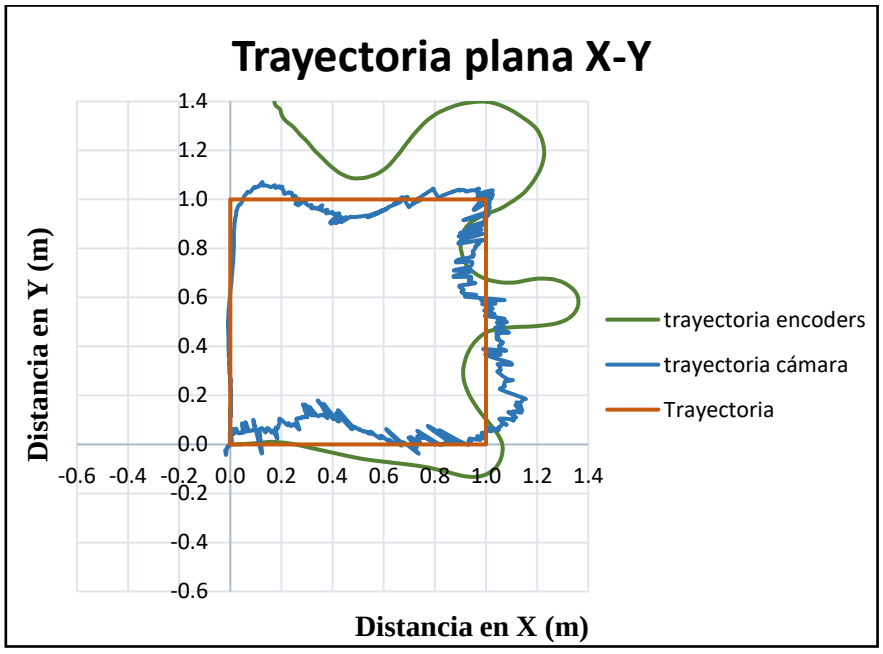


Figura 41. Trayectoria X-Y con cámara cenital.

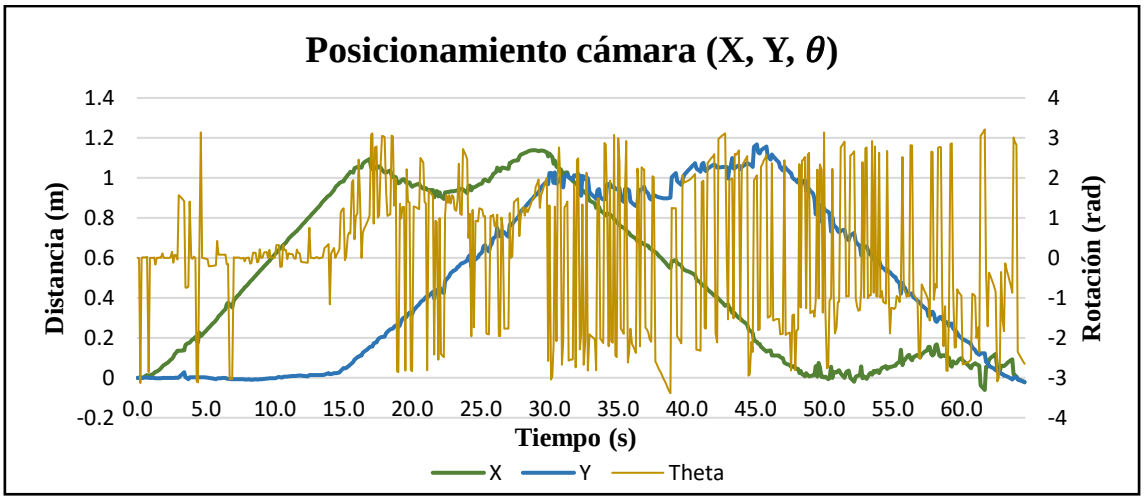


Figura 42. Posicionamiento en simulación con cámara cenital.

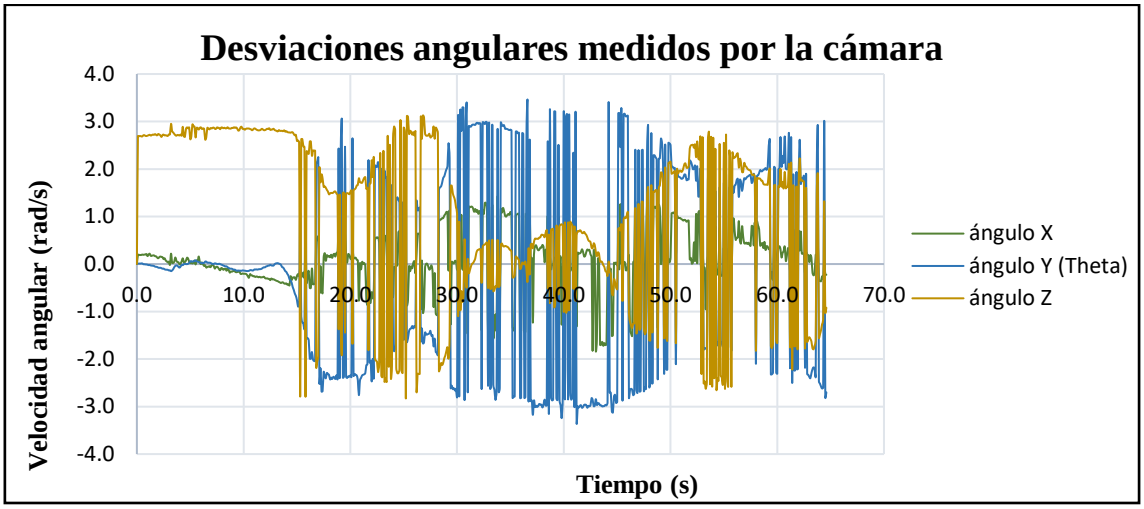


Figura 43. Desviaciones angulares en simulación con cámara cenital.

Como se puede ver en estas simulaciones, en las que se emplea la cámara como dispositivo para definir el estado del vehículo en cada instante, también se dan lugar ciertos problemas. A diferencia de lo que ocurría con los encoders, el vehículo siempre consigue llegar al punto final de la trayectoria, y por lo tanto no se ve afectado por esas situaciones indeseadas. Pese a esta mejora, existen dos inconvenientes que hacen que este sensor no sea idóneo, que son el problema en la medición de ángulos y los retrasos en el envío de información. A continuación, se define por qué ocurren ambos problemas, así como algunas de sus características.

En cuanto al problema en la medición de ángulos, tal y como se muestra *Figura* 43 donde se grafican los tres ángulos tomados por el programa de la cámara, se observa que ninguno de ellos posee unos valores acordes con el recorrido propuesto. Dentro de los tres ángulos que se muestran, el correcto debería ser el segundo (ángulo θ), pero al no proporcionar unos valores adecuados se trata de elaborar una función a partir de la cual, mediante diferenciales de posición entre cada instante de tiempo k y k-1, se trata de obtener el ángulo θ correcto. De esta forma, se pretende que no sea necesario utilizar el valor angular medido directamente por el sensor, utilizando un esquema como el que se muestra en la *Figura 44*.

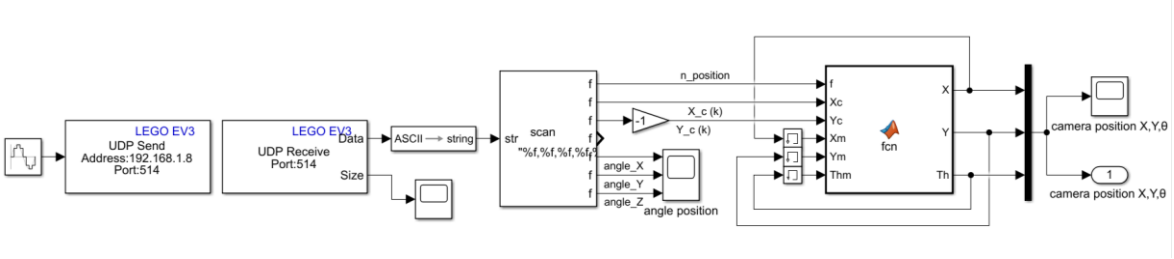


Figura 44. Subesquema Simulink cámara cenital con ángulo calculado.

Al llevar a cabo un segundo grupo de experimentaciones empleando la cámara, pero recalculando el ángulo θ se llegan a alcanzar mejores resultados, pero el nuevo método sigue sin proporcionar un valor de desviación angular óptimo.

El segundo de los problemas mencionados se debe a la pérdida de información por parte de la cámara, la cual envía un paquete de datos cada periodo T como se le solicita, pero si en ese instante no detecta el TAG (elemento que capta la cámara) colocado sobre el vehículo renvía la

posición pasada, lo que trae consigo una pérdida de precisión que provoca desvíos e irregularidades. Esto se observa en la *Figura 45* perteneciente a la simulación ya mencionada, donde se muestra el número de periodos en los que la cámara ha detectado (paquete completo recibido) o no el TAG (paquete incompleto) con respecto al total.

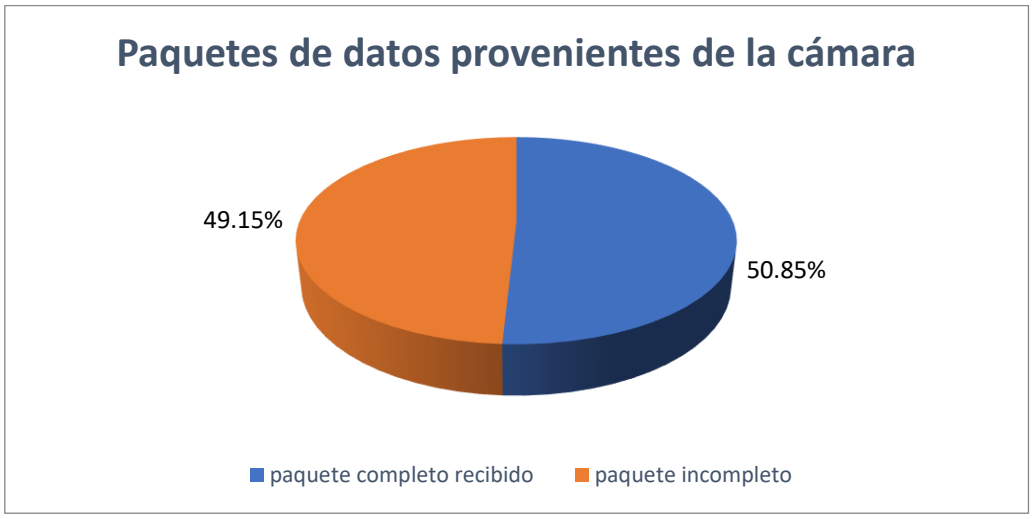


Figura 45. Pérdidas de información al recibir de la cámara.

Por este conjunto de problemas que carecen de solución directa, se decide modificar la forma de obtener la posición y la orientación, pasándose a utilizar el sistema de balizas junto con un sensor de orientación IMU para la captación de informaciones de estado más precisas.

4.3.4.3. Simulaciones con balizas

En este tercer grupo de experimentaciones se utiliza el sistema de balizas, descrito en el tercer apartado de este mismo documento, para la obtención de los valores de posición en los ejes X e Y. Este conjunto formado por cuatro balizas fijas y una baliza móvil, que ha de colocarse sobre el vehículo, basan su funcionamiento en ultrasonidos y transmiten la información a una terminal de radio que se conecta directamente a un ordenador para recibir la información correspondiente. Junto al sistema de balizas se utiliza un sensor de orientación IMU BNO055 AHRS, a partir del cual se obtiene la información de desviación angular θ.

Con la inclusión de este nuevo sistema de sensado, la cámara cenital no se requiere dentro del esquema de funcionamiento, por lo que queda desplazada a una posición de comprobación para corroborar el correcto funcionamiento de las simulaciones.

Para el uso de los nuevos dispositivos ya mencionados es necesario emplear una placa de desarrollo con un ESP32, la cual se encarga tanto de solicitar la lectura de datos como de transmitir la información correspondiente.

Todo este conjunto de nuevos elementos requiere de alimentación. Algunos de ellos, como es el caso de las balizas fija y móviles disponen de baterías y, por lo tanto, pueden funcionar sin conexión directa, aunque también admiten esta opción. En cambio, otros como la placa con el ESP32 requieren de un dispositivo que les aporte alimentación de forma continuada.

Finalmente, el esquema en Simulink del conjunto definido es el mostrado en la *Figura* 46, donde se observa como la cámara sirve únicamente para la toma de información y las balizas junto con el IMU la sustituyen en la obtención de la información de estado. El esquema del subconjunto que forman las balizas y el IMU es el mostrado en la *Figura 47*.

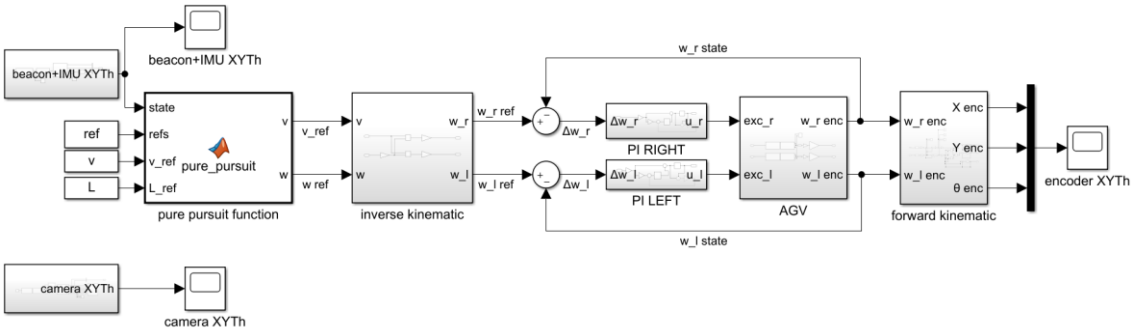


Figura 46. Esquema Simulink con sensado por balizas+IMU.

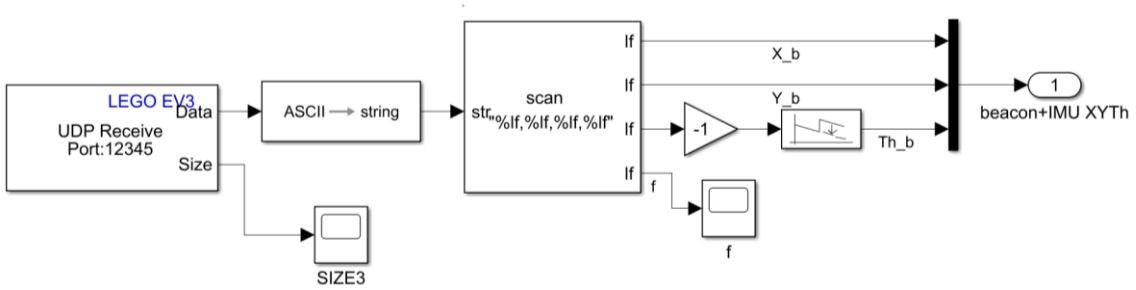


Figura 47. Subesquema Simulink de las balizas y el IMU.

Las representaciones gráficas correspondientes a las simulaciones efectuadas con este esquema de funcionamiento son: *Figura 48*, *Figura 49*, *Figura 50* y *Figura 51*.

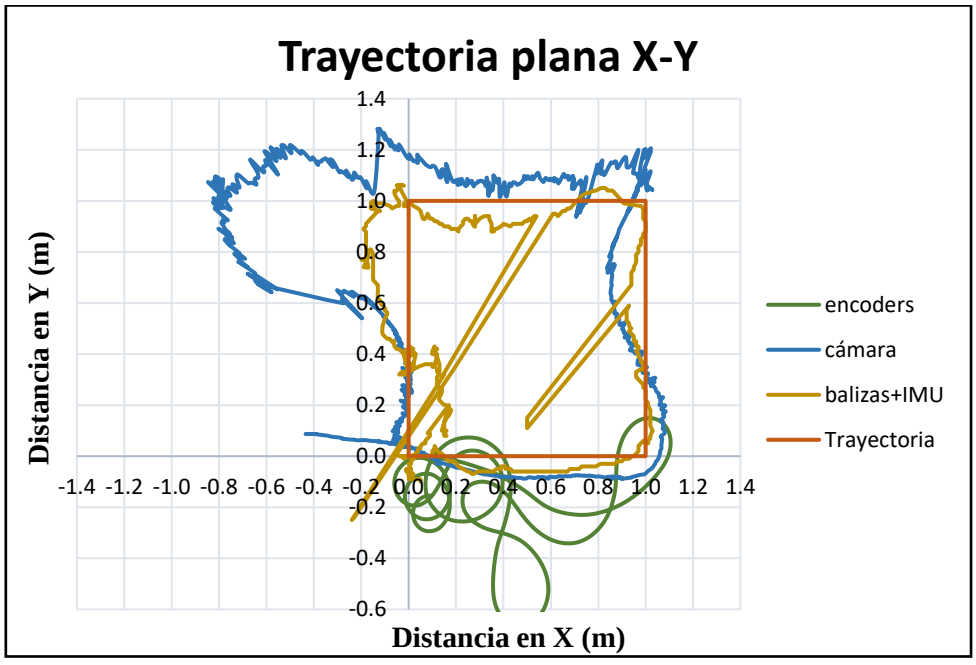


Figura 48. Posicionamiento X-Y con balizas+IMU.

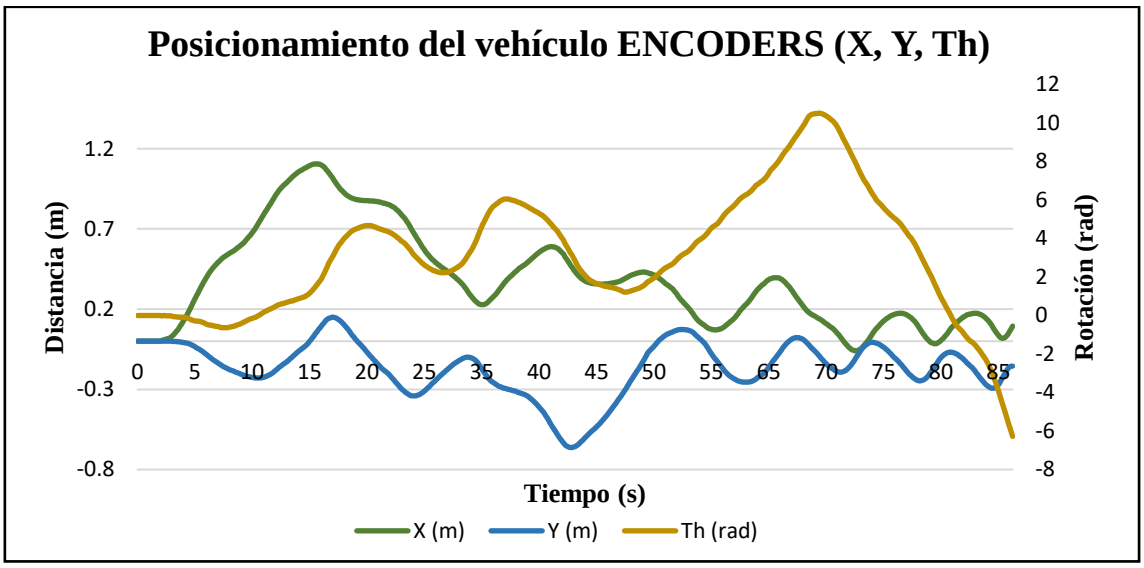


Figura 49. Posicionamiento a través de los encoders con balizas+IMU.

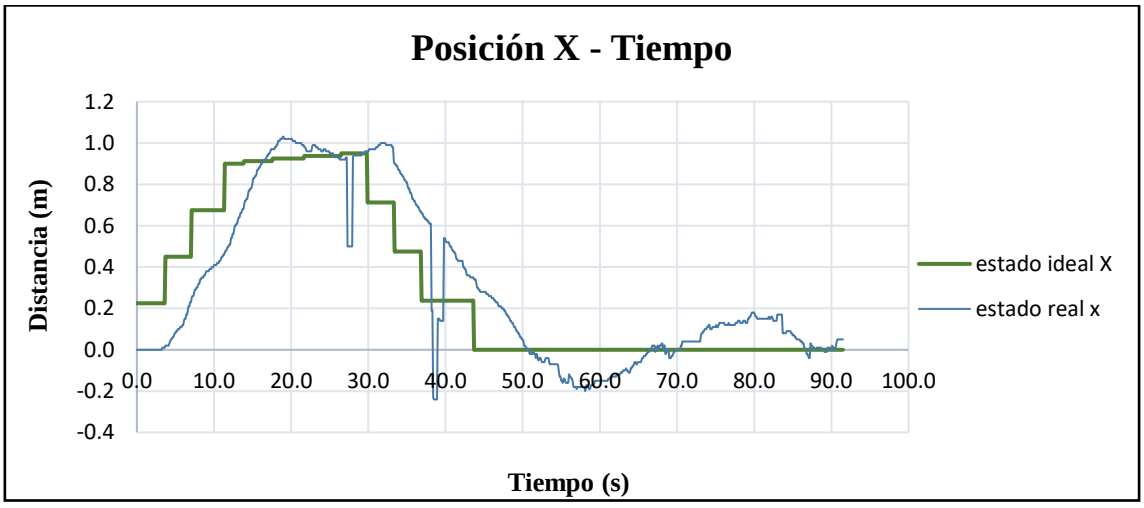


Figura 50. Seguimiento de la referencia en X-tiempo con balizas+IMU.

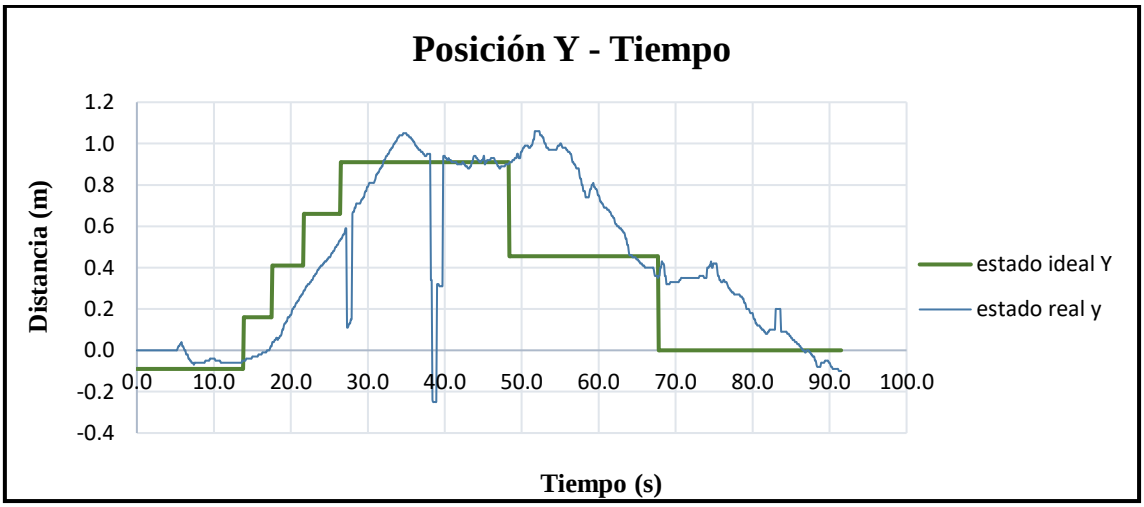


Figura 51. Seguimiento de la referencia en Y-tiempo con balizas+IMU.

Como se observa en la *Figura 48*, la cámara posee otro problema que no se ha mencionado aún, y es que cuando el vehículo se aproxima a los límites de captación de la misma la señal obtenida se ve afectada, produciendo irregularidades como las observadas en dicha gráfica.

4.3.5. EKF

El siguiente paso para mejorar aún más la respuesta obtenida consta de dos partes. La primera de ellas es dividir el programa en dos, uno que se ejecute en el ordenador para que asuma la mayor parte de la necesidad computacional y un segundo que esté cargado en el robot LEGO y únicamente se encargue de recibir la información de excitación y de enviar la sensada por los encoders. La segunda consiste en añadir un Filtro de Kalman Extendido (EKF) para mejorar la respuesta del sistema, ya que este actúa reduciendo los problemas que originan los diferentes ruidos del proceso y de la medición. El modelo matemático correspondiente al EKF se encuentra en el apartado 3.3 de este documento, [6] y [7].

En las siguientes figuras se muestran los modelos de Simulink que se emplean en las experimentaciones. En la *Figura 52* se encuentra el ejecutado en el ordenador, donde se puede observar como el bloque del EKF recibe los datos de posición del sistema de balizas y el IMU, la excitación que sale de ambos PI y la velocidad sensada por los encoders, generando como salidas las mismas variables de posición y velocidad, pero reduciendo el ruido. El subbloque de EKF se muestra en la *Figura 53* y sus respectivos programas de código están en el anexo. En la *Figura 54* se encuentra el modelo cargado en el robot LEGO, donde únicamente se recibe la señal de excitación de cada motor y se envía la velocidad medida en cada rueda.

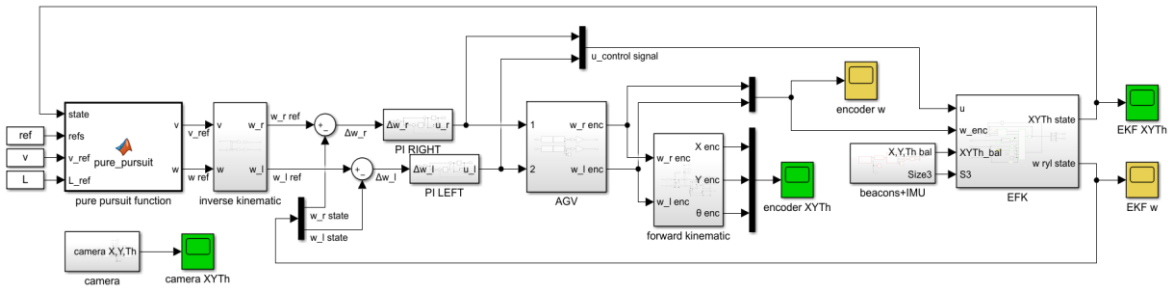


Figura 52. Esquema Simulink EKF ejecutado por el PC.

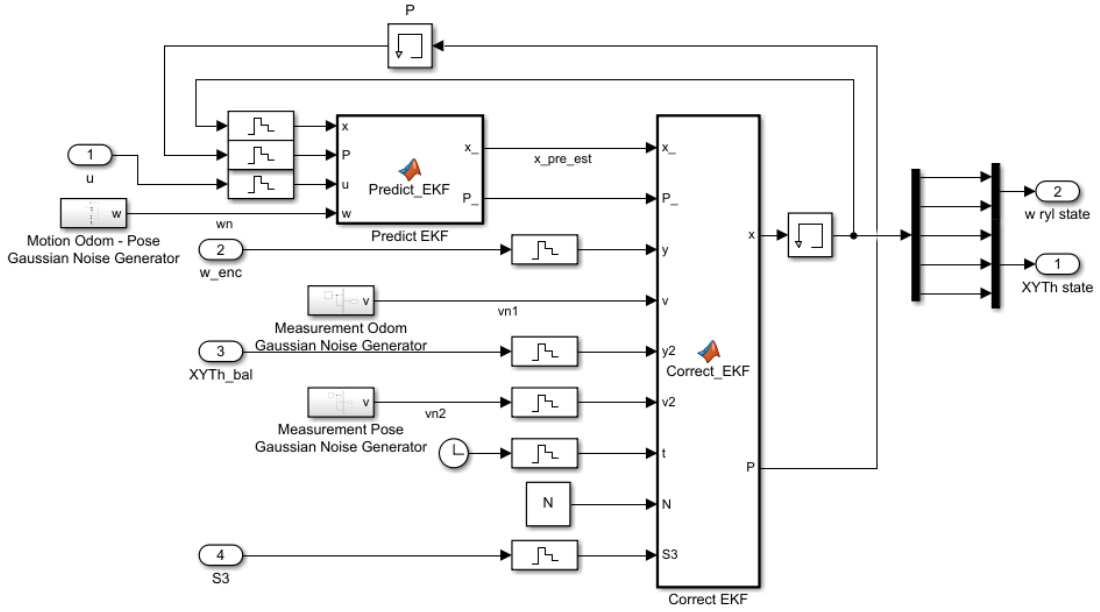


Figura 53. Subesquema Simulink bloque EKF.

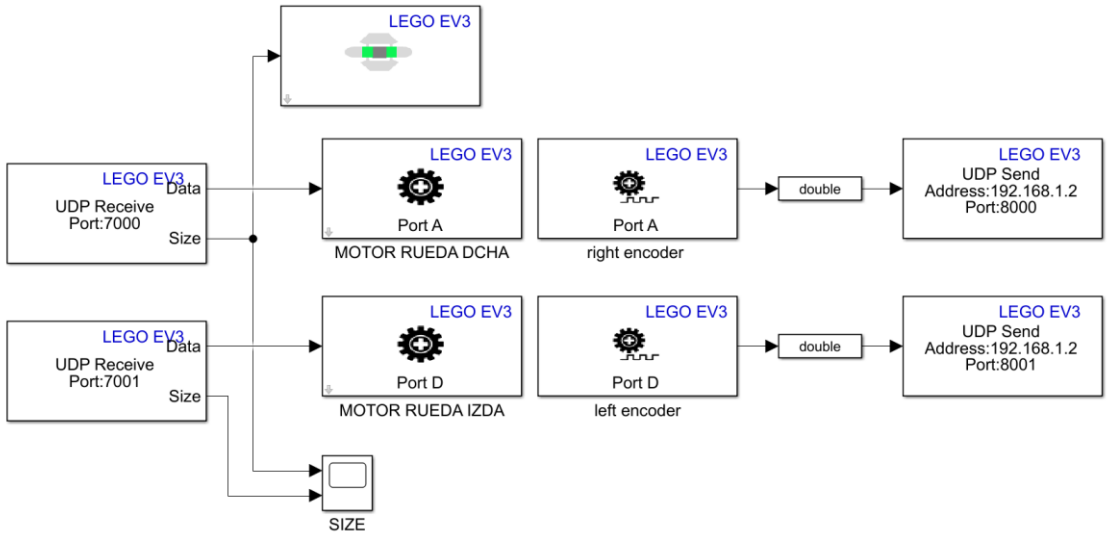


Figura 54. Esquema Simulink EKF ejecutado por el robot LEGO.

Una vez definidos ambos modelos, se llevan a cabo dos experimentaciones. La primera de ellas donde todo funciona a periodo rápido (T=0,1s), la cual se define como el caso nominal, ya que se trata del mejor de los resultados obtenido. La segunda simulación se ejecuta con todo funcionando a periodo lento (T=0,2s), donde se duplica el periodo para llevar a cabo ese ahorro energético y de consumo de red que se planteó en los objetivos del trabajo.

En primer lugar, se muestran los resultados del caso nominal, representando los datos obtenidos en las siguientes gráficas: *Figura 55*, *Figura 56*, *Figura 57*, *Figura 58* y *Figura 59*.

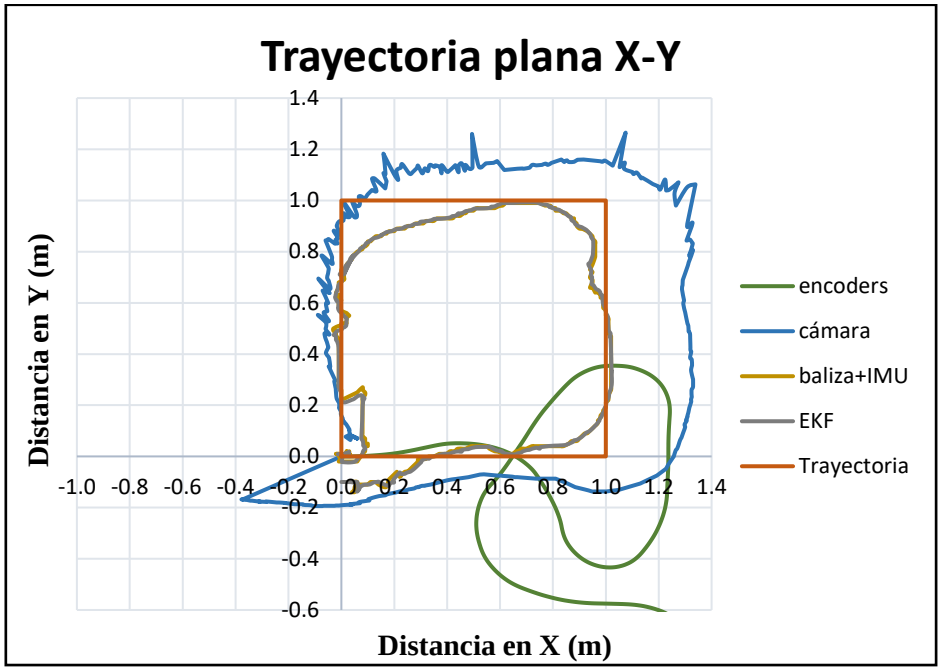


Figura 55. Trayectoria X-Y EKF con T=0,1s.

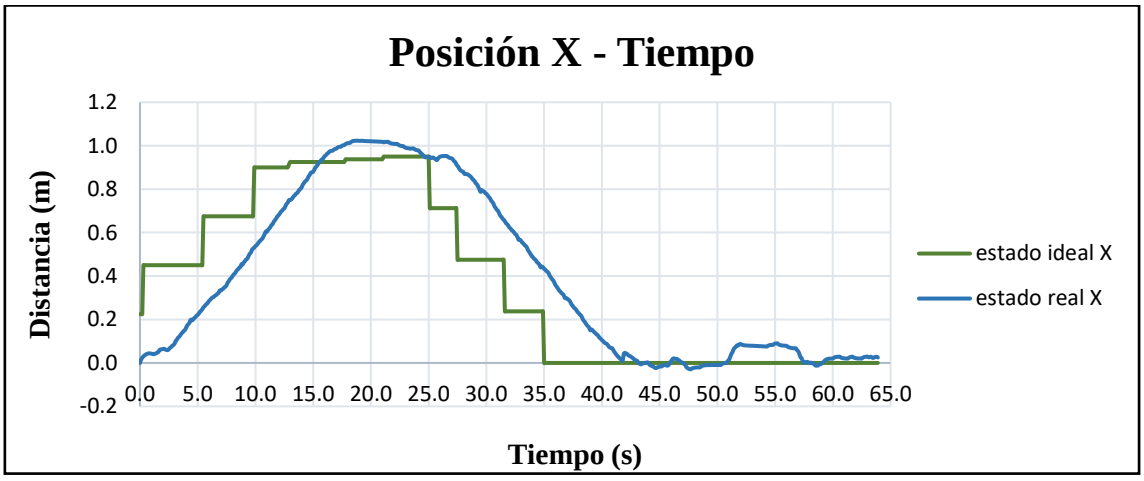


Figura 56. Seguimiento de la posición en X-tiempo con EKF y T=0,1s.

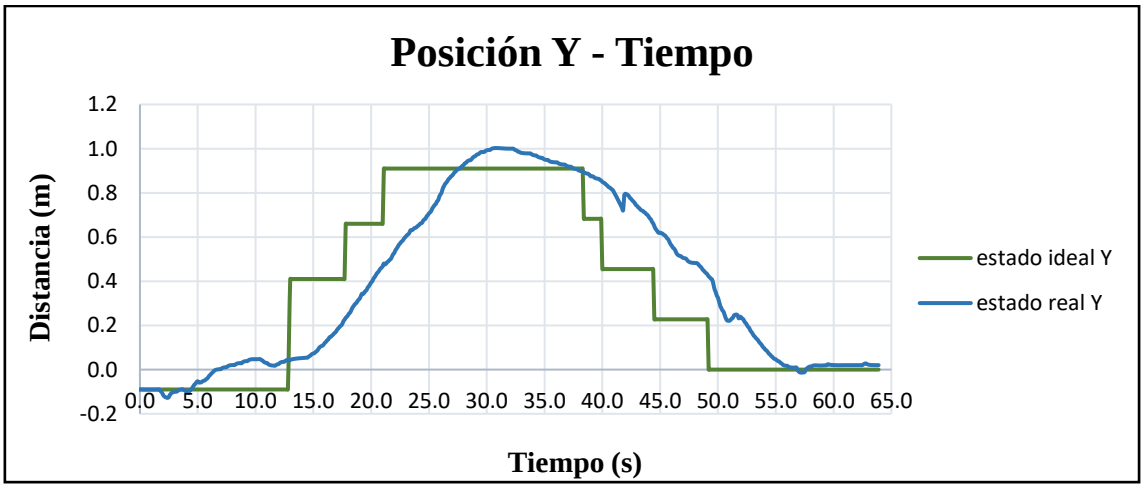


Figura 57. Seguimiento de la posición en Y-tiempo con EKF y T=0,1s.

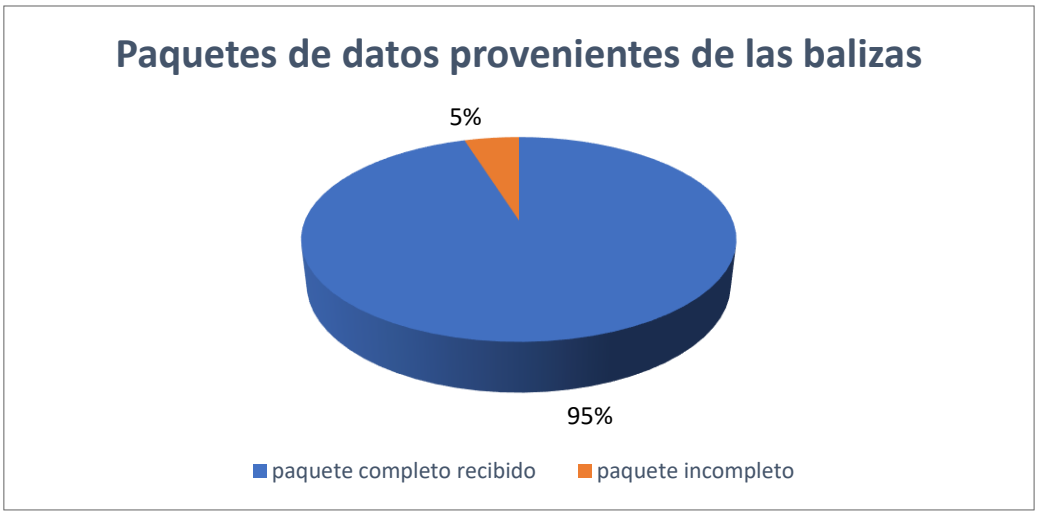


Figura 58. Pérdida de datos del sistema de balizas+IMU con T=0,1s.

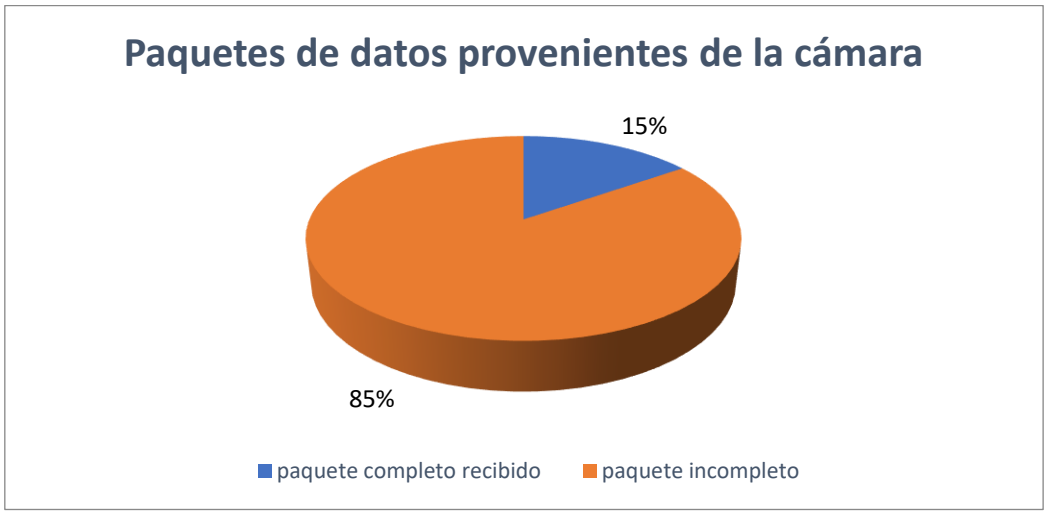


Figura 59. Pérdida de datos de la cámara cenital con T=0,1s.

Una vez comprobado lo que ocurre en el caso nominal donde la trayectoria seguida por el robot es correcta, se lleva a cabo la experimentación duplicando el periodo. También se añaden dos figuras, *Figura 58* y *Figura 59*, donde se muestran los paquetes de datos que llegan completos para los dispositivos de la cámara y las balizas junto con el IMU, de forma que se puedan comparar con los datos obtenidos al duplicar el periodo.

Los resultados extraídos con el doble de periodo se representan en las *Figura 60*, *Figura 61* y *Figura 62*.

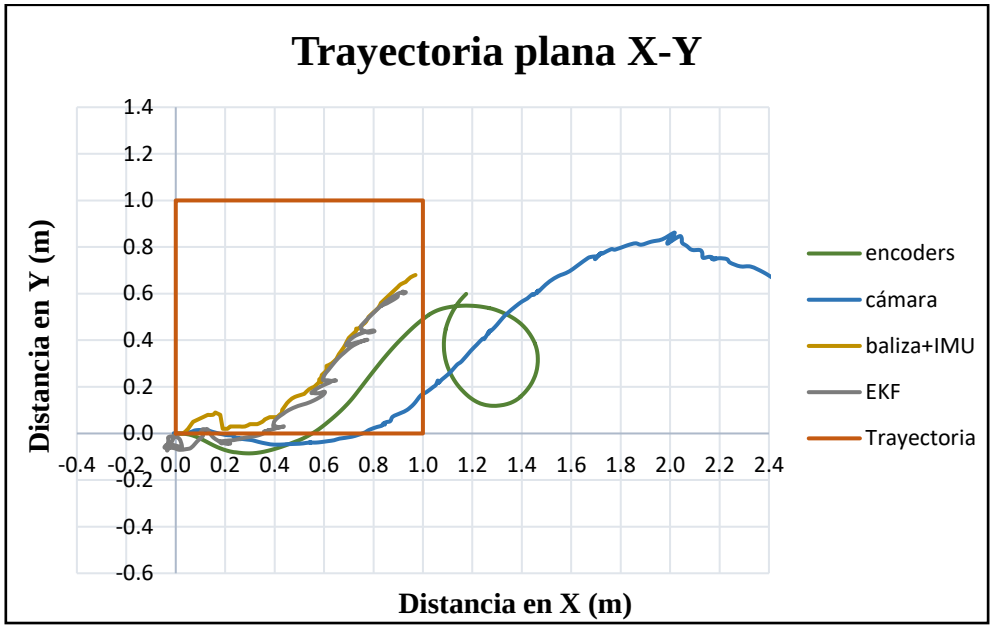


Figura 60. Trayectoria X-Y con EKF y T=0,2s.

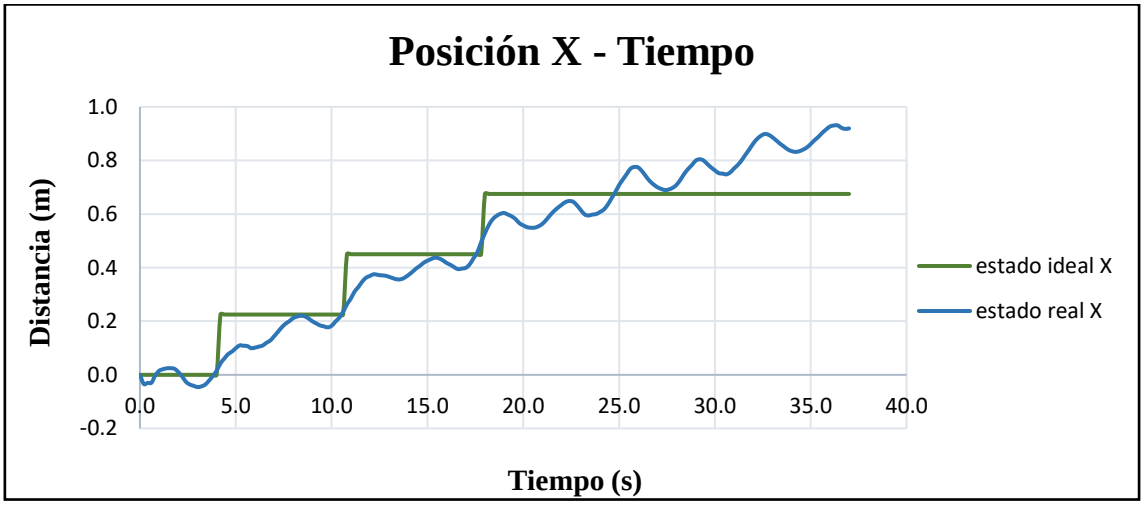


Figura 61. Seguimiento de la posición en X-tiempo con EKF y T=0,2s.

Figura 62. Seguimiento de la posición en Y-tiempo con EKF y T=0,2s.

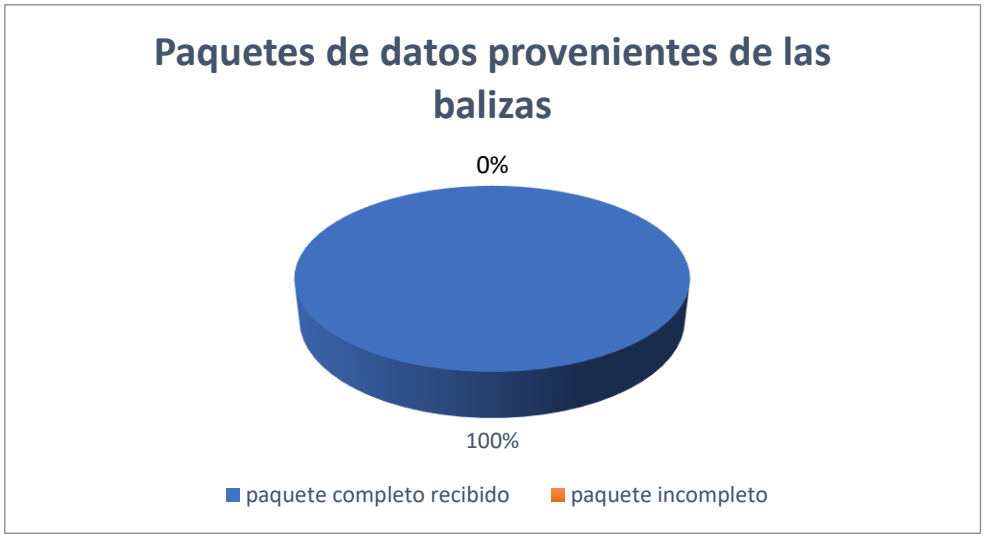


Figura 63. Pérdida de datos del sistema de balizas con T=0,2s.

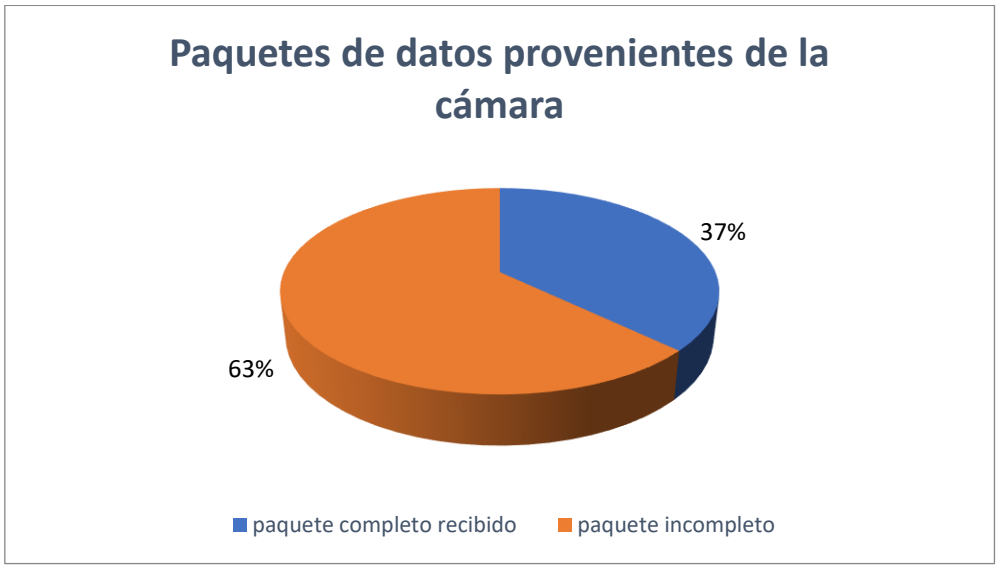


Figura 64. Pérdida de datos de la cámara cenital con T=0,2s.

Como se observa en este último caso, el resultado de duplicar el periodo conlleva una pérdida total de calidad en la respuesta obtenida, donde el robot es incapaz de seguir la trayectoria. También se comprueba como para este caso las pérdidas de información se reducen significativamente al solicitarse información cada más tiempo.

4.3.6. PI multirate

Este último paso consiste en la inclusión de un controlador PI que funcione a multifrecuencia, es decir, que por cada señal de medida que recibe a periodo lento 2T sea capaz de generar 2 acciones de control a periodo rápido T. De esta forma, se busca solucionar el problema que ocurría con el modelo pasado, cuando todo se llevaba a cabo a periodo lento y la respuesta obtenida no era la deseada. Con la inclusión de este PI se logra obtener una respuesta muy cercana al caso nominal, y ahorrando energía, ancho de banda y carga computacional, ya que todo funciona a periodo lento salvo el robot, el cual tiene una acción de control para cada periodo T.

El modelo necesario para llevar a cabo este último grupo de experimentaciones es el mostrado en la *Figura 65*, donde todo es idéntico al caso anterior salvo el subbloque PI *multirate*, cuyos componentes, entradas y salidas se encuentran en la *Figura 66*.

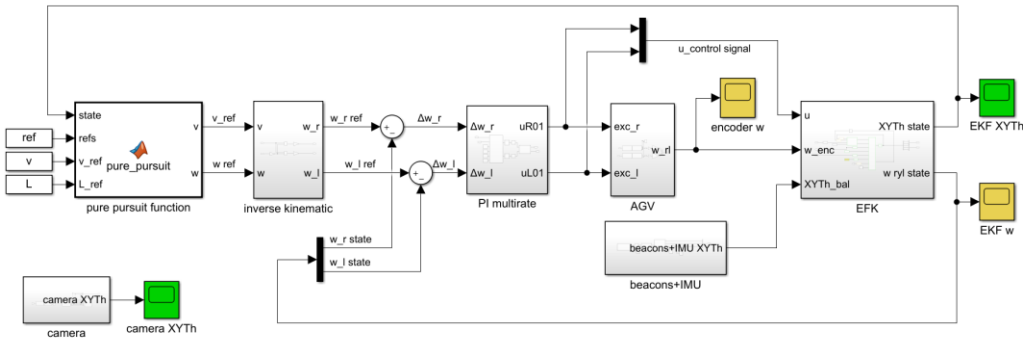


Figura 65. Esquema Simulink PI multirate.

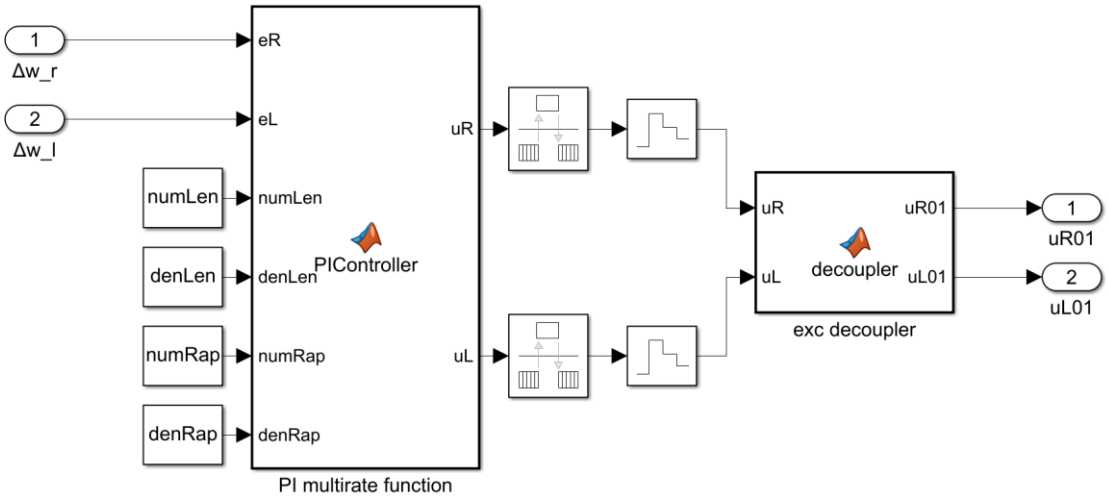


Figura 66. Subesquema Simulink PI multirate.

Como se aprecia en la *Figura 66*, los componentes principales del PI *multirate* son dos bloques de función de MATLAB, cuyos códigos se muestran en los anexos de este mismo documento junto con otro archivo de ejecución previa que sirve para inicializar las variables necesarias y generar las funciones de transferencia correspondientes. El primero de los bloques, "*PI multirate function*", se encarga de generar todas las acciones de control, mientras que el "*exc decoupler*" se encarga de ordenar esas acciones de control de forma que las rápidas se ejecuten en el siguiente periodo T y las lentas en el 2T.

De este modo, tomando como entradas las señales de error en velocidad para las ruedas izquierda y derecha a periodo lento (entradas del bloque), se computan unas acciones de control también a periodo lento (salidas del bloque) pero que son ejecutadas a periodo rápido por el robot LEGO.

En las siguientes gráficas, *Figura 67*, *Figura 68* y *Figura 69*, se muestran los resultados obtenidos de forma experimental al incluir este nuevo bloque:

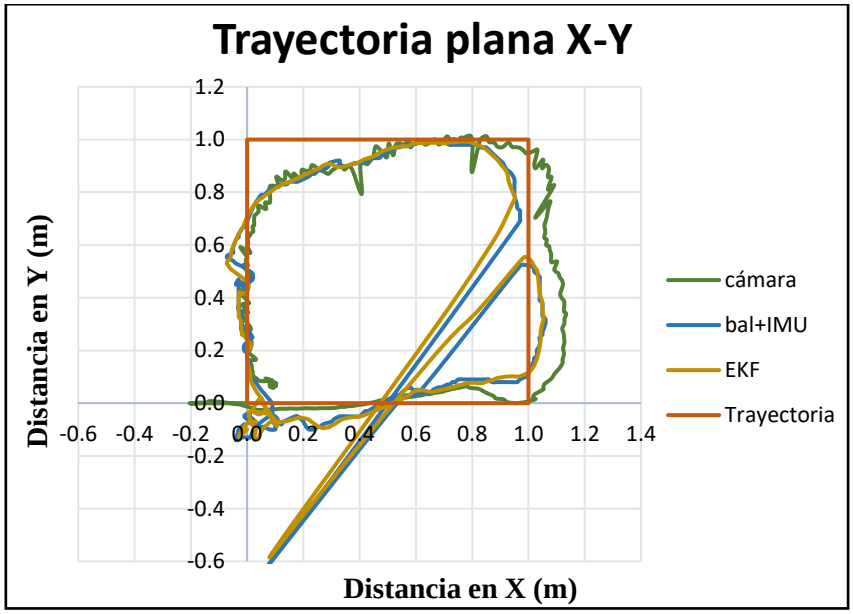


Figura 67. Trayectoria X-Y con PI multirate.

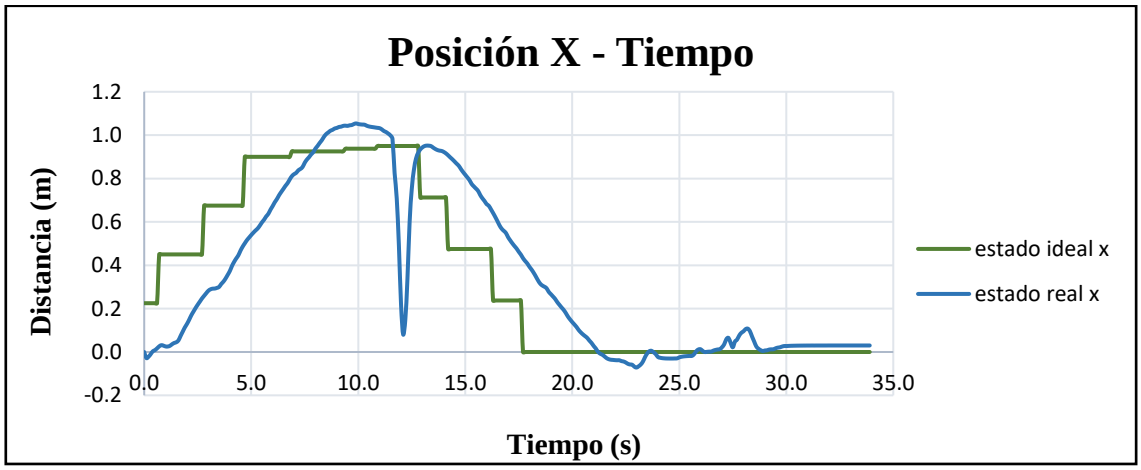


Figura 68. Seguimiento de la posición en X-tiempo con PI multirate.

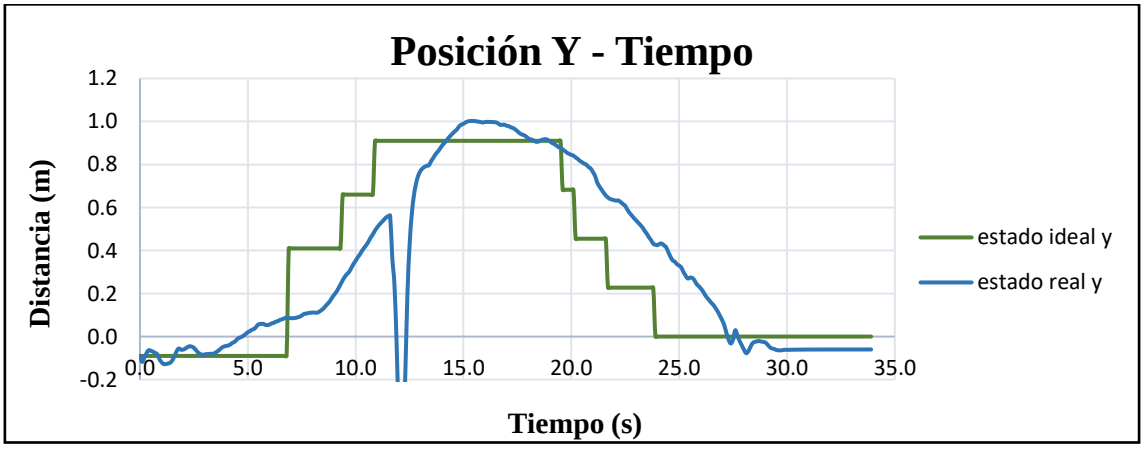


Figura 69. Seguimiento de la posición en Y-tiempo con PI multirate.

Tal y como muestran las anteriores representaciones gráficas, se obtiene una respuesta por parte del robot muy similar a la del caso nominal, todo ello habiendo mejorado en gran medida la cantidad de energía consumida y el consumo de ancho de banda de la red utilizada para el envío de información.

Es importante destacar la anomalía que sucede en torno al segundo 12 de simulación y que se observa en las tres representaciones anteriores. Este suceso que genera datos espurios es debido a un problema de localización puntual de las balizas que, como se aprecia en la figura, no afecta al resultado final, alcanzándose la solución deseada.

4.4. Análisis de resultados

Para facilitar el análisis del cumplimiento de los diferentes objetivos correspondiente al apartado de conclusiones, se decide llevar a cabo el cálculo de unos índices de coste que permitan ejecutar dicha comparación de modo cuantitativo [2] y [8].

A continuación, se muestra la definición de estos índices, explicando su funcionalidad y la forma de obtención de cada uno de ellos. La variable "l" indica el número de iteraciones que requerirá el AGV para alcanzar el punto final de la trayectoria.

- J1: se basa en ℓ_2-norm y su objetivo es determinar de forma numérica la precisión con la que el robot siguió la trayectoria que se le había solicitado. Su ecuación es la siguiente:

$$J_1 = \sum_{k=1}^{l} \min_{1 \le k\prime \le l} \sqrt{\{x_k^{NT} - \bar{x}_{k\prime}^{NT}\}^2 + \{y_k^{NT} - \bar{y}_{k\prime}^{NT}\}^2} \quad (27)$$

- J2: se basa en ℓ_∞-norm y su funcionalidad es conocer la diferencia máxima que se ha producido entre la ruta que debía seguir el robot y la que finalmente ha seguido. Su ecuación es la siguiente:

$$J_2 = \max_{1 \le k \le l} \left\{ \sum_{k=1}^{l} \min_{1 \le k\prime \le l} \sqrt{\{x_k^{NT} - \bar{x}_{k\prime}^{NT}\}^2 + \{y_k^{NT} - \bar{y}_{k\prime}^{NT}\}^2} \right\} \quad (28)$$

- J3: mide el tiempo total en segundos que ha tardado el robot en alcanzar la posición final. Su ecuación es la siguiente:

$$J_3 = lNT \quad (29)$$

Una vez definido cada uno de los coeficientes se pasan a comparar los resultados de cada una de las experimentaciones, los cuales aparecen en la *Tabla 8*.

Tabla 8. Comparación de los índices de coste.

	Pure pursuit	**EKF T (nominal)**	**EKF 2T**	**PI *multirate* (final)**
J_1 ( )	0,70308	0,00496	-	0,00813
J_2 ( )	1,70190	0,43908	∞	0,68301
J_3 (s)	80	55	∞	34

Es importante destacar el ligero aumento de los índices de costo en el caso del PI multirate con respecto al caso nominal, lo cual puede ser debido a los valores de referencia empleados ya que estos van dando saltos a medida que avanza el robot y su calidad también varía en función de los sensores, y no solo a una reducción de la precisión.

Para facilitar aún más la comparación entre los casos nominal (EKF todo a periodo T) y final (PI *multirate*), se adjunta la *Figura 70*, donde se observa la trayectoria de cada uno de los casos, así como la línea que idealmente debería seguir el robot.

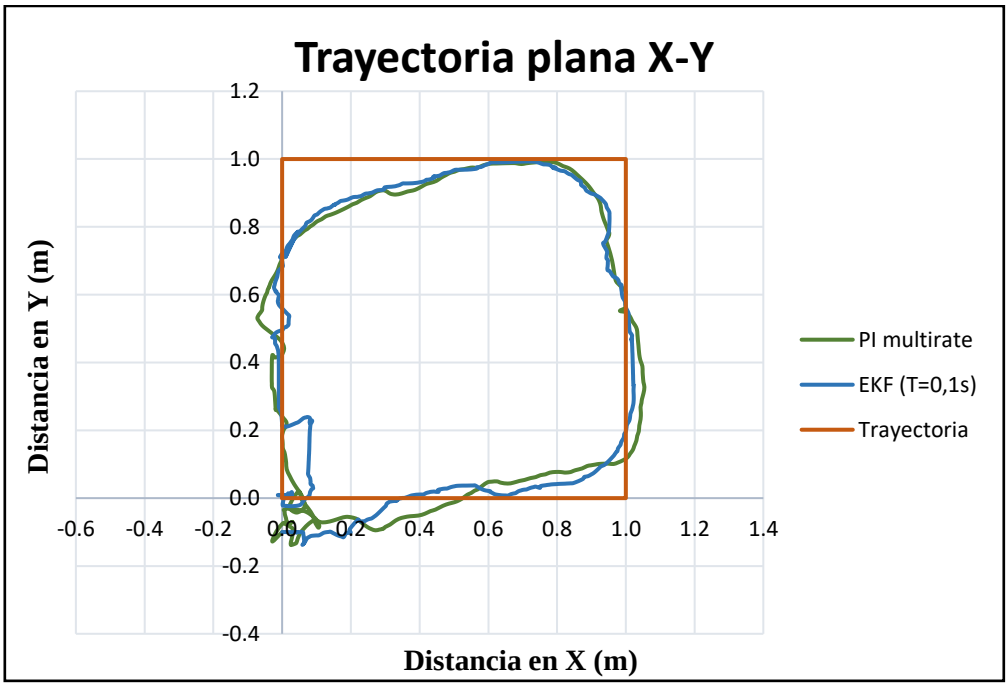


Figura 70. Comparación gráfica de la trayectoria final y la nominal.

Tanto en esta representación como en el cálculo de los índices de costes se han eliminado los valores espurios detectados en el caso del PI *multirate*.

Como se contempla con la comparación gráfica y de índices de coste, las diferencias de precisión y calidad de los casos nominal y final son mínimas.

5. CONCLUSIONES

Tal y como se muestra en la evolución de los apartados de desarrollo de este mismo documento, finalmente se logra alcanzar el objetivo principal planteado al comienzo del proyecto. Éste consiste en obtener datos experimentales en los que se muestre el control de un vehículo de dos ruedas que esté capacitado para ejecutar todo tipo de trayectorias, con el mayor ahorro computacional, energético y de ancho de banda de la red posible, todo ello sin perder calidad en el resultado obtenido.

No obstante, gracias a la información presentada en los análisis del apartado 4.4, es posible comprobar de forma cualitativa, *Figura 70*, y cuantitativa, *Tabla 8*, la validez de la solución obtenida, corroborando que la calidad del resultado final (PI *multirate*) es la adecuada en comparación con el caso nominal (EKF a periodo T=0,1s), todo ello acompañado del importante ahorro que se tiene en cuenta en todo instante.

Además del objetivo principal del proyecto se plantearon una serie de subobjetivos, los cuales necesariamente se deben alcanzar para verificar el correcto desarrollo del trabajo y así poder asegurar que las conclusiones extraídas son las adecuadas. A continuación, se lleva a cabo un análisis pormenorizado de cada uno de ellos:

- En cuanto a la obtención de un entorno de trabajo eficiente y adecuado, este es el primer punto abordado, en el cual se tuvieron en cuenta los diferentes programas a utilizar, los elementos o componentes necesarios para la ejecución de las simulaciones y las características físicas, principalmente cinemáticas, que han permitido llevar a cabo todas las simulaciones de forma óptima y eficiente.

- Con respecto a los modelos y programas de MATLAB y Simulink utilizados para las experimentaciones, se ha llevado a cabo un proceso iterativo de mejora continua en la solución obtenida, de forma que tras cada etapa se logra mejorar el resultado obtenido en la anterior, llegando a alcanzar la solución final del PI *multirate*, tal y como se ha descrito detalladamente en el apartado 4.3.

- Por último, se logran alcanzar unos resultados que se encuentren dentro de unos límites de estabilidad y precisión en la solución obtenida, sin olvidar en ningún instante el ahorro ya mencionado.

Finalmente, con el PI *multirate* se consigue una solución de elevada calidad que se asemeja en gran medida al caso nominal, reduciendo el gasto energético, computacional y de ancho de red a la mitad y estando siempre dentro de los límites de estabilidad establecidos como correctos para las simulaciones.

PRESUPUESTO

DISEÑO DE UN CONTROL BIFRECUENCIA EN RED PARA SEGUIMIENTO DE TRAYECTORIAS EN UN ROBOT DE DOS RUEDAS

Autor: Juan Antonio Román Ortega

Fecha: septiembre de 2022

ÍNDICE DEL PRESUPUESTO:

1. RECURSOS Y COSTES ASOCIADOS 57
1.1. Coste personal 58
1.2. Coste material 58
2. ACTIVIDADES 62
3. CRONOGRAMA 64
4. PRESUPUESTO 65
5. RESUMEN DEL PRESUPUESTO 67

ÍNDICE DE TABLAS DEL PRESUPUESTO:

Tabla 1. Características de los recursos empleados. 57
Tabla 2. Productos empleados y sus características. 60
Tabla 3. Resumen de actividades. 62
Tabla 4. Presupuesto (trabajo, duración y costes asociados). 65
Tabla 5. Resumen del presupuesto y coste final. 67

1. RECURSOS Y COSTES ASOCIADOS

Para poder llevar a cabo el cálculo presupuestario, el primer paso es recopilar información sobre los diferentes recursos utilizados a lo largo del proyecto, ya sean personales o materiales. A continuación, en la *Tabla 1*, se muestra un listado del conjunto de los recursos y algunas de sus principales características.

Tabla 1. Características de los recursos empleados.

Nombre del recurso	Tipo	Capacidad/ cantidad	Tasa/precio	Coste por uso	Coste Total
Trabajo personal	Trabajo	4,0 hr/día	17,34 €/hr	0,00 €	6.242,4 €
Formación	Trabajo	1,0 hr/día	31,33 €/hr	0,00 €	1.269,95 €
Ordenador ASUS	Material	1 unidad	111,69 €	0,00 €	111,69 €
Robot LEGO Mindstorms EV3	Material	1 unidad	33,06 €	0,00 €	33,06 €
Licencia MATLAB	Material	1 unidad	975,20 €	0,00 €	975,20 €
Licencia MSProject	Material	1 unidad	30,00 €	0,00 €	30,00 €
Cámara cenital	Material	1 unidad	32,99 €	0,00 €	32,99 €
Sistema balizas	Material	1 unidad	33,06 €	0,00 €	33,06 €
Placa ESP32	Material	1 unidad	9,29 €	0,00 €	9,29 €
Sensor IMU	Material	1 unidad	78,40 €	0,00 €	78,40 €
Espacio de trabajo	Material	1 unidad	250,00 €	0,00 €	250,00 €

En los dos siguientes apartados se llevan a cabo aclaraciones sobre los recursos definidos para hacer más sencilla su interpretación.

1.1. Coste personal

Como ya se ha visto en la tabla resumen, existen dos tipos de coste referidos al trabajo de personas, los cuales aparecen detallados a continuación. Las unidades en las que se mide este trabajo son en horas por cada día, siendo la remuneración en euros cada hora.

- Trabajo personal: consiste en el trabajo desarrollado por el ingeniero encargado del estudio, ejecución y redacción del proyecto. Para calcular el coste por hora de este recurso se ha tenido en cuenta el salario medio de un empleado de este tipo según el INE (instituto nacional de estadística) con datos del 2020 [10]. El salario bruto mensual medio es de 2774,21 €, para una jornada de 8 horas y 20 días al mes, por lo tanto, el correspondiente salario en € por cada hora de trabajo es el que se muestra en la *ecuación 1*.

$$salario_{personal} = \frac{2774{,}21\ €}{mes} \cdot \frac{mes}{20\ días} \cdot \frac{días}{8\ horas} = 17{,}34\ {}^{€}/_{hora} \quad (1)$$

- Formación: consiste en el trabajo de formación y apoyo aportado por otro personal al desarrollo del proyecto. Para calcular el coste por hora de este recurso se ha procedido de forma idéntica al caso anterior, pero esta vez escogiendo el rango más alto de esa misma sección que asciende a un total de 5013,05 €, calculando en la *ecuación 2* su valor en € por cada hora de trabajo.

$$salario_{formación} = \frac{5013{,}05\ €}{mes} \cdot \frac{mes}{20\ días} \cdot \frac{días}{8\ horas} = 31{,}33\ {}^{€}/_{hora} \quad (2)$$

1.2. Coste material

Dentro de los costes de material explicados a continuación se va a hacer referencia a tres clases o categorías claramente diferenciadas.

En la primera de ellas se engloban aquellos recursos materiales de valor relativamente elevado, por lo cual se decide efectuar una amortización sobre su valor global. Los recursos que concuerdan con estas características son los siguientes:

- Ordenador ASUS: este ordenador modelo GL752VW ha sido el empleado para el desarrollo del proyecto, tanto para la parte experimental con el uso de los programas como para llevar a cabo el análisis y la redacción. Al no ser ésta su única utilidad, se

decide calcular su valor durante los meses de duración del proyecto, escogiendo una amortización en 5 años (siendo el máximo de 10 según hacienda). Su valor económico es de 1.351,45 € y quitándole el 21% correspondiente al IVA (impuesto sobre el valor añadido) se queda en 1.116,90 €. El cálculo se lleva a cabo en la *ecuación 3* para los 6 meses de duración del proyecto.

$$amortización_{ordenador} = \frac{1116{,}90\ €}{5\ años \cdot \frac{12\ meses}{1\ año}} \cdot 6\ meses = \ 111{,}69\ € \quad (3)$$

- Robot LEGO Mindstorms EV3: este modelo de robot programable es el utilizado en todas las simulaciones llevadas a cabo. Su valor económico es de 400,00 € y quitándole el 21% correspondiente al IVA se queda en 330,58 €. El cálculo, efectuado en la *ecuación 4*, se lleva a cabo para los 2,5 meses de duración de la parte experimental y al igual que ocurre con el caso anterior se decide amortizar en un total de 5 años.

$$amortización_{robot\ LEGO} = \frac{330{,}58\ €}{5\ años \cdot \frac{12\ meses}{1\ año}} \cdot 6\ meses = \ 33{,}06\ € \quad (4)$$

- Sistema de balizas: este sistema de balizas es el utilizado para el control posicional en las simulaciones. Su valor económico es de 399,99 € y quitándole el 21% correspondiente al IVA se queda en 330,57 €. El cálculo se lleva a cabo en la *ecuación 5* para los 2,5 meses de duración de la parte experimental y se decide amortizar en un total de 5 años.

$$amortización_{balizas} = \frac{330{,}57\ €}{5\ años \cdot \frac{12\ meses}{1\ año}} \cdot 6\ meses = \ 33{,}06\ € \quad (5)$$

- Licencia MATLAB: este es el principal programa utilizado en el desarrollo del proyecto, dentro del cual se han utilizado varios módulos o productos, cada uno de los cuales posee su propio valor. En la *Tabla 2* se muestra el conjunto de productos utilizados y el precio del servicio por un año, así como su valor sin IVA.

Tabla 2. Productos empleados y sus características.

Productos	Periodo del servicio	Precio	Precio sin IVA
MATLAB	1 año	840,00 €	694,21 €
Simulink	1 año	1260,00 €	1041,32 €
Control System Toolbox	1 año	480,00 €	396,69 €
Instrument Control Toolbox	1 año	480,00 €	396,69 €
Optimization Toolbox	1 año	480,00 €	396,69 €
TOTAL	-	3540,00 €	2925,60 €

En este caso, como la duración de la licencia para cada producto es de un año, se calcula el valor imputable de los mismos para los 4 meses en los que el programa es necesario, tal y como se muestra en la *ecuación 6*.

$$amortización_{MATLAB} = \frac{2925{,}60\ €}{1\ año \cdot \frac{12\ meses}{1\ año}} \cdot 4\ meses = 975{,}20\ € \quad (6)$$

- Licencia MSProject: este programa es el utilizado para el cálculo del presupuesto y la organización del proyecto. Este tipo de licencia al igual que la de MATLAB es anual, pero permite obtenerla de forma mensual por un precio de 30 €, por lo cual no es necesario efectuar el cálculo de amortización para ella.

En el segundo grupo se encuentran esos objetos cuyo valor económico no es relativamente elevado y por ello no se les aplica amortización.

- Cámara cenital: con un valor económico de 32,99 €.
- Placa ESP32: con un valor económico de 9,29 €.
- Sensor IMU: con un valor económico de 78,40 €.

La última categoría de recursos es aquella en la que se encuentra el espacio de trabajo donde, a diferencia del resto, el coste va en función del número de días de utilización.

- Espacio de trabajo: este es el lugar en el que se llevan a cabo todas las simulaciones del proyecto y donde se instalan el resto de los recursos ya definidos. Para calcular su coste imputable al proyecto se realiza multiplicando un valor económico mensual por el número de meses que duran las simulaciones, un total de 2,5 meses, *ecuación 7*.

$$coste_{espacio\ de\ trabajo} = \frac{100{,}00\ €}{mes} \cdot 2{,}5\ meses = 250\ € \quad (7)$$

2. ACTIVIDADES

Este apartado sirve como introducción para comprender mejor el cálculo del presupuesto para el presente trabajo, haciendo un breve resumen del conjunto de las actividades que se han llevado a cabo en el transcurso de éste. Las actividades en las que se divide la ejecución del proyecto suman un total de diez, las cuales van numeradas del 1 al 10 por la ID (número de identificación de la actividad) y del 1.1 al 1.10 por la EDT (estructura de descomposición del trabajo), siendo la distinción importante en caso de que existiese diferencia jerárquica entre dos o más tareas.

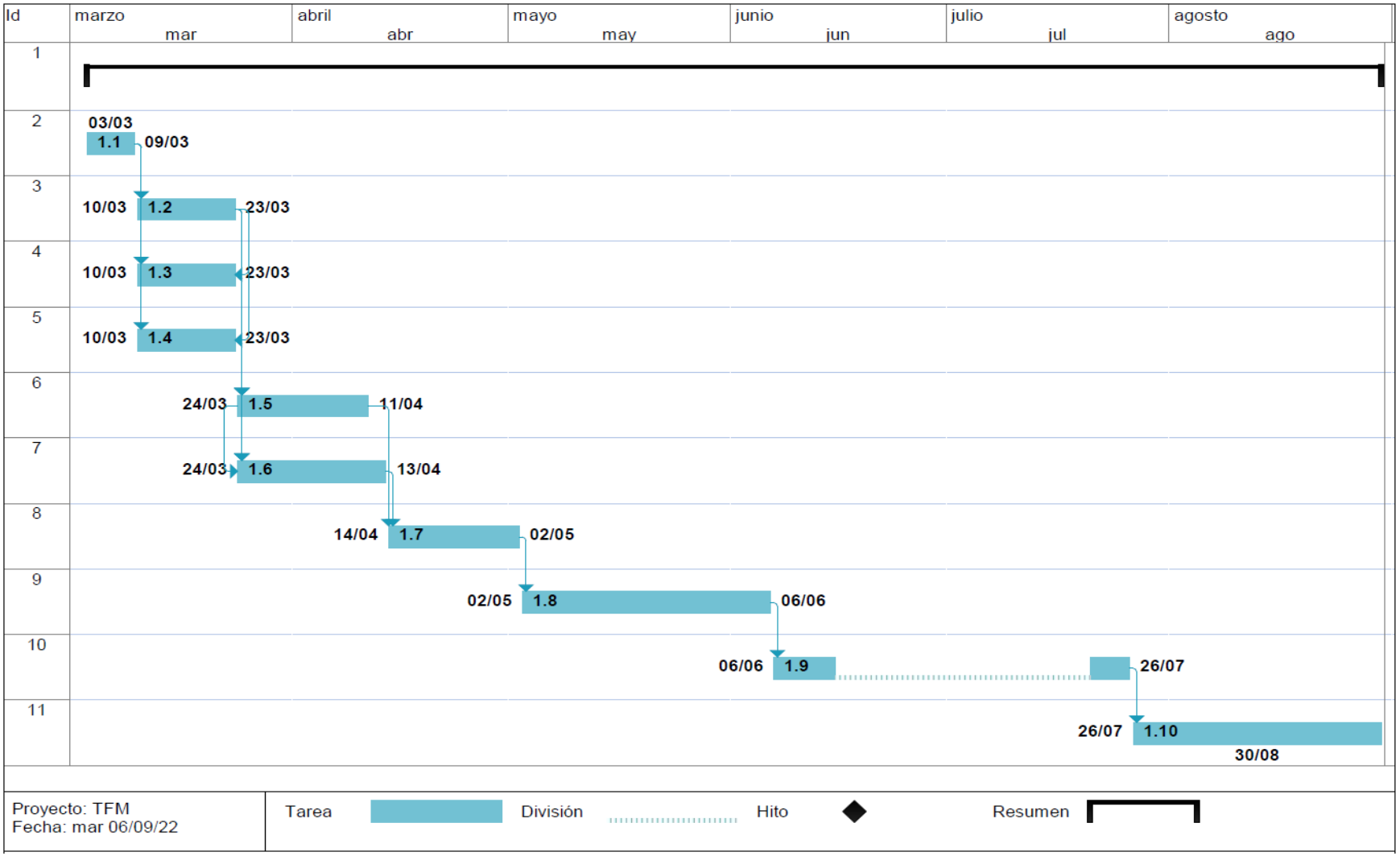
En la *Tabla 3* se presenta el listado de actividades junto con algunas de sus principales características como la duración o sus fechas de inicio y fin, además se añade una casilla donde se mencionan los recursos empleados, que han sido previamente definidos en el apartado anterior.

Tabla 3. Resumen de actividades.

ID	EDT	Nombre de tarea	Comienzo	Fin	Nombres de los recursos
	1	**Proyecto TFM**	**jue 03/03/22**	**mar 30/08/22**	
1	**1.1**	Análisis del problema a resolver	jue 03/03/22	mié 09/03/22	-Trabajo personal
2	**1.2**	Estudio de información bibliográfica	jue 10/03/22	mié 23/03/22	-Trabajo personal [50%]
3	**1.3**	Estudio de las técnicas de control necesarias	jue 10/03/22	mié 23/03/22	-Trabajo personal [25%] -Formación [50%]
4	**1.4**	Estudio de los programas necesarios	jue 10/03/22	mié 23/03/22	-Trabajo personal [25%] -Formación [50%]
5	**1.5**	Adquisición y preparación de los programas	jue 24/03/22	lun 11/04/22	-Trabajo personal [50%]
6	**1.6**	Montaje y preparación de los equipos	jue 24/03/22	mié 13/04/22	-Trabajo personal [50%]

ID	EDT	Nombre de tarea	Comienzo	Fin	Nombres de los recursos
7	**1.7**	Ejecución de los estudios previos	jue 14/04/22	lun 02/05/22	-Trabajo personal -Formación
8	**1.8**	Ejecución de las simulaciones	lun 02/05/22	lun 06/06/22	-Trabajo personal -Formación
9	**1.9**	Estudio y análisis de los resultados obtenidos	lun 06/06/22	mar 26/07/22	-Trabajo personal
10	**1.10**	Redacción del proyecto	mar 26/07/22	mar 30/08/22	-Trabajo personal [50%]

3. CRONOGRAMA



4. PRESUPUESTO

Para calcular el coste total del proyecto, así como para obtener el cronograma de la página anterior, se ha utilizado el MSProject, programa en el cual se introducen todos los datos definidos en los apartados de recursos con sus respectivos costes y actividades donde se determina el uso de cada recurso. De este programa se extrae la *Tabla 4*, donde se muestra el coste total de cada tarea, su duración y los recursos utilizados junto con su coste.

Tabla 4. Presupuesto (trabajo, duración y costes asociados).

ID	EDT	Nombre de tarea	Trabajo	Coste
	1	**Proyecto TFM**	**Tr. personal 360 hrs Formación 45 hrs**	**7.512,35 €**
		Ordenador ASUS	*1 ud*	*111,69 €*
		Robot LEGO Mindstorms EV3	*1 ud*	*33,06 €*
		Licencia MATLAB	*1 ud*	*975,20 €*
		Licencia MSProject	*1 ud*	*30,00 €*
		Cámara cenital	*1 ud*	*32,99 €*
		Sistema balizas	*1 ud*	*33,06 €*
		Placa ESP32	*1 ud*	*9,29 €*
		Sensor IMU	*1 ud*	*78,40 €*
		Espacio de trabajo	*1 ud*	*250,00 €*
1	**1.1**	Análisis del problema a resolver	20 hrs	346,80 €
		Trabajo personal	*20 hrs*	*346,80 €*
2	**1.2**	Estudio de información bibliográfica	20 hrs	346,80 €
		Trabajo personal	*20 hrs*	*346,80 €*
3	**1.3**	Estudio de las técnicas de control necesarias	15 hrs	260,10 €
		Trabajo personal	*10 hrs*	*173,40 €*
		Formación	*5 hrs*	*86,70 €*
4	**1.4**	Estudio de los programas necesarios	15 hrs	260,10 €
		Trabajo personal	*10 hrs*	*173,40 €*
		Formación	*5 hrs*	*86,70 €*

ID	EDT	Nombre de tarea	Trabajo	
5	**1.5**	Adquisición y preparación de los programas	26 hrs	450,84 €
		Trabajo personal	*26 hrs*	*450,84 €*
6	**1.6**	Montaje y preparación de los equipos	30 hrs	520,20 €
		Trabajo personal	*30 hrs*	*520,20 €*
7	**1.7**	Ejecución de los estudios previos	50 hrs	1.006,90 €
		Trabajo personal	*40 hrs*	*693,60 €*
		Formación	*10 hrs*	*313,30 €*
8	**1.8**	Ejecución de las simulaciones	125 hrs	2.517,25 €
		Trabajo personal	*100 hrs*	*1734,00 €*
		Formación	*25 hrs*	*783,25 €*
9	**1.9**	Estudio y análisis de los resultados obtenidos	54 hrs	936,36 €
		Trabajo personal	*54 hrs*	*936,36 €*
10	**1.10**	Redacción del proyecto	50 hrs	867,00 €
		Trabajo personal	*50 hrs*	*867,00 €*

5. RESUMEN DEL PRESUPUESTO

A partir de los datos del presupuesto extraído de MSProject se obtiene la *Tabla 5* a modo resumen, añadiendo el valor total del presupuesto con IVA.

Tabla 5. Resumen del presupuesto y coste final.

ID	EDT	Nombre de tarea	Coste Total
1	**1.1**	Análisis del problema a resolver	346,80 €
2	**1.2**	Estudio de información bibliográfica	346,80 €
3	**1.3**	Estudio de las técnicas de control necesarias	260,10 €
4	**1.4**	Estudio de los programas necesarios	260,10 €
5	**1.5**	Adquisición y preparación de los programas	450,84 €
6	**1.6**	Montaje y preparación de los equipos	520,20 €
7	**1.7**	Ejecución de los estudios previos	1.006,90 €
8	**1.8**	Ejecución de las simulaciones	2.517,25 €
9	**1.9**	Estudio y análisis de los resultados obtenidos	936,36 €
10	**1.10**	Redacción del proyecto	867,00 €
		COSTE TOTAL DEL TRABAJO	**7.512,35 €**
		OTROS COSTES	**1.553,69 €**
		Proyecto TFM (PRESUPUESTO SIN IVA)	**9.066,04 €**
		IVA (21%)	1.903,87 €
		Proyecto TFM (PRESUPUESTO CON IVA)	**10.969,91 €**

El presupuesto del proyecto asciende a un total de 10.969,91 € (diez mil novecientos sesenta y nueve coma noventa y un EUROS).

ANEXOS Y BIBLIOGRAFÍA

DISEÑO DE UN CONTROL BIFRECUENCIA EN RED PARA SEGUIMIENTO DE TRAYECTORIAS EN UN ROBOT DE DOS RUEDAS

Autor: Juan Antonio Román Ortega

Fecha: septiembre de 2022

ÍNDICE DE LOS ANEXOS Y LA BIBLIOGRAFÍA:

1. ANEXO 1: INICIALIZACIÓN E INTERPOLACIÓN 73
2. ANEXO 2: PURE PURSUIT 75
3. ANEXO 3: EKF 76
   3.1. Predicción 76
   3.2. Corrección 77
4. ANEXO 4: PI MULTIRATE 79
   4.1. Inicialización 79
   4.2. Función PI multirate 80
   4.3. Función de selección de estado 81
5. BIBLIOGRAFÍA 82

1. ANEXO 1: INICIALIZACIÓN E INTERPOLACIÓN

```
%Periodo de muestreo del encoder, etc. [s]
T = 0.1;

%Distancia de mirar hacia delante (look-ahead distance). [m]
L = 0.2;

%Posición inicial del robot. [m y rad]
X_0 = 1.1;
Y_0 = -0.45;
Th_0 = 0;

%Velocidad lineal del robot. [m/s]
v = 0.1;

%Velocidad angular máxima del robot. [rad/s]
w = 1;

%Geometría del robot

%(r = radio de las ruedas [m])
r = 0.028;
%(b = distancia entre el punto central del eje y una de las
ruedas/apoyos [m])
b = 0.07;

%Excitaciones máxima y mínima de envío a los motores del robot.
[]
exc_max = 100;
exc_min = -100;

%Parámetros del PID.
Kc = 3.6;
Ti = 0.2;
Td = 0;

%Cuadrado (interpolaciones):

%Control de posición con los encoders:
%p0 = [0 0];
%p1 = [1 0];
%p2 = [1 1];
%p3 = [0 1];
%p4 = [0 0];

%Control de posición con la cámara:
%p0 = [-0.63 0];
%p1 = [0.37 0];
%p2 = [0.37 1];
%p3 = [-0.63 1];
%p4 = [-0.63 0];
```

```
%Control de posición con las balizas:
p0 = [1.1 -0.45];
p1 = [2 -0.45];
p2 = [2.05 0.55];
p3 = [1.1 0.55];
p4 = [1.1 -0.36];

X1 = [0 1];
V1 = [p0;p1];
Xq1 = [0:0.25:1];
i1 = interp1(X1,V1,Xq1,'linear');

X2 = [0 1];
V2 = [p1;p2];
Xq2 = [0:0.25:1];
i2 = interp1(X2,V2,Xq2,'linear');

X3 = [0 1];
V3 = [p2;p3];
Xq3 = [0:0.25:1];
i3 = interp1(X3,V3,Xq3,'linear');

X4 = [0 1];
V4 = [p3;p4];
Xq4 = [0:0.25:1];
i4 = interp1(X4,V4,Xq4,'linear');

ref = [i1;i2;i3;i4];


%Error de detención. [m]
err = 0.1;
```

2. ANEXO 2: PURE PURSUIT

Partiendo como base para el desarrollo del siguiente proyecto: [8].

```
function [v, w]  = pure_pursuit(state, refs, v_ref, L_ref)

%Inicialización de variables
persistent idx finish end_idx
if isempty(idx)
    idx = 1;
    finish = 0;
    end_idx = length(refs);
end
v = 0;
w = 0;
ref = refs(idx,:);

%Comprobación de que se tienen todos los datos necesarios y se
calculan los diferenciales de posición y la distancia D
if length(state) == 3
    dx = ref(1) - state(1);
    dy = ref(2) - state(2);
    D = sqrt(dx*dx + dy*dy);
    while D < L_ref && finish == 0
        if idx < length(refs)
            idx = idx + 1;
            ref = refs(idx,:);
            dx = ref(1) - state(1);
            dy = ref(2) - state(2);
            D = sqrt(dx*dx + dy*dy);
        else
            break;
        end
    end

    %Salida del bucle while con o sin valores de velocidad
    if idx == end_idx && D <= L_ref/5
        finish = 1;
    end
    if finish == 0
         Ka = 2 * ( (dy * cos(state(3)) - dx * sin(state(3))) /
(dx^2 + dy^2) );
        v = v_ref;
        w = Ka * v_ref;
    end
end
```

3. ANEXO 3: EKF

Partiendo como base para el desarrollo del siguiente proyecto: [8].

3.1. Predicción

```
function [x_, P_] = Predict_EKF(x, P, u, w)
Q = diag([0.382 0.382 0.382 0.382 0.382]);
%Predicción
x_ = f_motionModel(x, u, w);
F = f_jac_x(x, u, w);
P_ = F * P * F' + Q ;
end

function x = f_motionModel(x, u, w)
%Modelo dinámico
x(1) = (1-(T/tau)) * x(1) + (K*T/tau) * u(1) + w(1);
x(2) = (1-(T/tau)) * x(2) + (K*T/tau) * u(2) + w(2);
%Modelo cinemático
v_ = (r/2) * (x(1) + x(2)) * T;
w_ = (r/(2*b)) * (x(1) - x(2)) * T;
x(5) = x(5) + w_ + w(5);
x(3) = x(3) + v_ * cos(x(5)) + w(3);
x(4) = x(4) + v_ * sin(x(5)) + w(4);
end

function F = f_jac_x(x, u, w)
z=f_motionModel(x, u, w);
n=numel(x);
m=numel(z);
F=zeros(m,n);
h=n*eps;
for k=1:n
    x1=complex(x);
    x1(k)=x1(k)+h*1i;
    F(:,k)=imag(f_motionModel(x1, u, w))/h;
end
end
```

3.2. Corrección

```
function [x, P] = Correct_EKF(x_, P_, y, v, y2, v2, t, N)
R = diag([0.382 0.382]);
RR = diag([0.382 0.382 0.5 0.5 0.9]);
xDim = size(x_,1);
I = eye(xDim);

persistent n
if isempty(n)
    n = 0;
end

m = mod(n, N);

if m == 0
t;
yy = [y; y2];
vv = [v; v2];
%Medición de no linealidades y linealización
y_ = h_measurementModel2(x_, vv);
H = h_jac_x2(x_, vv);
M = h_jac_v2(x_, vv);
P1 = P_ * H';                              %covarianza
K  = P1 * inv(H * P1 + (M * RR * M'));     %ganancia filtro Kalman
x  = x_ + K * (yy-y_);                     %estimación estado
%Matriz de covarianzas de estado
P  = K * RR * K' + (I - K * H) * P_ * (I - K * H)';
n = 0;
else

%Corrección
%Medición de las no linealidades y linealización
y_ = h_measurementModel(x_, v);
H = h_jac_x(x_, v);
M = h_jac_v(x_, v);
P1 = P_ * H';                              %covarianza
K  = P1 * inv(H * P1 + (M * R * M'));      %ganancia filtro Kalman
x  = x_ + K * (y-y_);                      %estimación estado
%Matriz de covarianzas de estado
P  = K * R * K' + (I - K * H) * P_ * (I - K * H)';
end

n = n + 1;
end

function y = h_measurementModel(x, v)
y = [x(1) + v(1); x(2) + v(2)];
end

function H = h_jac_x(x, v)
z=h_measurementModel(x, v);
n=numel(x);
m=numel(z);
```

```
H=zeros(m,n);
h=n*eps;
for k=1:n
    x1=complex(x);
    x1(k)=x1(k)+h*1i;
    H(:,k)=imag(h_measurementModel(x1, v))/h;
end
end

function M = h_jac_v(x, v)
z=h_measurementModel(x, v);
n=numel(v);
m=numel(z);
M=zeros(m,n);
h=n*eps;
x = complex(x);
for k=1:n
    v1=complex(v);
    v1(k)=v1(k)+h*1i;
    M(:,k)=imag(h_measurementModel(complex(x), v1))/h;
end
end

function y = h_measurementModel2(x, v)
y = [x(1) + v(1); x(2) + v(2); x(3) + v(3); x(4) + v(4); x(5) +
v(5);];
end

function H = h_jac_x2(x, v)
z=h_measurementModel2(x, v);
n=numel(x);
m=numel(z);
H=zeros(m,n);
h=n*eps;
for k=1:n
    x1=complex(x);
    x1(k)=x1(k)+h*1i;
    H(:,k)=imag(h_measurementModel2(x1, v))/h;
end
end

function M = h_jac_v2(x, v)
z=h_measurementModel2(x, v);
n=numel(v);
m=numel(z);
M=zeros(m,n);
h=n*eps;
x = complex(x);
for k=1:n
    v1=complex(v);
    v1(k)=v1(k)+h*1i;
    M(:,k)=imag(h_measurementModel2(complex(x), v1))/h;
end
end
```

4. ANEXO 4: PI MULTIRATE

4.1. Inicialización

Partiendo como base para el desarrollo del siguiente proyecto: [2].

```
periodSlow = 0.2;
Nbif = 2;

%Modelo del proceso
s=tf('s');
Gpm=(K/(tau*s+1));

% PID controller
Gr=minreal(Kp*(1+Td*s+1/(Ti*s)));

TRap=periodSlow/Nbif;
zr=tf('z',TRap);
TLen=periodSlow;
zl=tf('z',TLen);

q0Rap=Kp*(1+(Td/TRap));
q1Rap=-Kp*(1-TRap/Ti+2*Td/TRap);
q2Rap=Kp*Td/TRap;

q0Len=Kp*(1+(Td/TLen));
q1Len=-Kp*(1-TLen/Ti+2*Td/TLen);
q2Len=Kp*Td/TLen;

GrRap=minreal((q0Rap+q1Rap*zr^(-1)+q2Rap*zr^(-2))/(1-zr^(-1)));
GrLen=minreal((q0Len+q1Len*zl^(-1)+q2Len*zl^(-2))/(1-zl^(-1)));

%Controlador PI dual-rate
Mc=minreal((Gr*Gpm/(1+Gr*Gpm)));
Mdl=c2d(Mc,TLen,'zoh');            % slow-rate (L)
Gmrl=minreal(1/(1-Mdl));
Mdr=c2d(Mc,TRap,'zoh');            % fast-rate (R)
GP1dr=c2d(Gpm,TRap,'zoh');
Gmrr=minreal(Mdr/GP1dr);
Gmrr2=minreal(GrRap/(1+GrRap*GP1dr));

[numLen,denLen] = tfdata(Gmrl,'v');
[numRap,denRap] = tfdata(Gmrr,'v');
```

4.2. Función PI multirate

Partiendo como base para el desarrollo del siguiente proyecto: [2].

```
function [uR,uL] = PIController(eR, eL, numLen, denLen, numRap,
denRap)

persistent memR memL

if isempty(memR)
    memR=[0 0 0 0 0 0];
    memL=[0 0 0 0 0 0];
end

%Acciones de control
uRLen_1=memR(1);
uRLen_2=memR(2);
uRRap_1=memR(3);
uRRap_2=memR(4);
eR_1=memR(5);
eR_2=memR(6);

uLLen_1=memL(1);
uLLen_2=memL(2);
uLRap_1=memL(3);
uLRap_2=memL(4);
eL_1=memL(5);
eL_2=memL(6);

uR = zeros(2,1);
uL = zeros(2,1);

%CALCULOS R (rápidos):
uRLen=-denLen(2)*uRLen_1-denLen(3)*uRLen_2+
numLen(1)*eR+numLen(2)*eR_1+numLen(3)*eR_2;

uRRap=-denRap(2)*uRRap_1-denRap(3)*uRRap_2+
numRap(1)*uRLen+numRap(2)*uRLen_1+numRap(3)*uRLen_1;
uRRap_2=uRRap_1;
uRRap_1=uRRap;
uR(1,1)=uRRap;
uRRap=-denRap(2)*uRRap_1-
denRap(3)*uRRap_2+numRap(1)*uRLen+numRap(2)*uRLen +
numRap(3)*uRLen_1;
uRRap_2=uRRap_1;
uRRap_1=uRRap;
uR(2,1)=uRRap;

uRLen_2=uRLen_1;
uRLen_1=uRLen;
eR_2=eR_1;
eR_1=eR;
```

```
%CALCULOS L (lentos):
uLLen=-denLen(2)*uLLen_1denLen(3)*uLLen_2+
numLen(1)*eL+numLen(2)*eL_1+numLen(3)*eL_2;

uLRap=-denRap(2)*uLRap_1-denRap(3)*uLRap_2+
numRap(1)*uLLen+numRap(2)*uLLen_1+numRap(3)*uLLen_1;
uLRap_2=uLRap_1;
uLRap_1=uLRap;
uL(1,1)=uLRap;
uLRap=-denRap(2)*uLRap_1-
denRap(3)*uLRap_2+numRap(1)*uLLen+numRap(2)*uLLen +
numRap(3)*uLLen_1;
uLRap_2=uLRap_1;
uLRap_1=uLRap;
uL(2,1)=uLRap;

uLLen_2=uLLen_1;
uLLen_1=uLLen;
eL_2=eL_1;
eL_1=eL;

%Variables de memoria
memR=[uRLen_1 uRLen_2 uRRap_1 uRRap_2 eR_1 eR_2];
memL=[uLLen_1 uLLen_2 uLRap_1 uLRap_2 eL_1 eL_2];
```

4.3. Función de selección de estado

```
function [uR01, uL01] = decoupler(uR, uL)

persistent n

if isempty(n)
    n = 1;
end

%Separación de las acciones de control rápidas y lentas
if n == 1
    uR01 = uR(1,1);
    uL01 = uL(1,1);
    n = n + 1;
else
    uR01 = uR(2,1);
    uL01 = uL(2,1);
    n = 1;
end
```

5. BIBLIOGRAFÍA

[1] SAE International & ISO Copyright Office. (2021, abril). *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles* (J3016_202104). Recuperado el 20 de agosto de 2021, de https://www.sae.org/standards/content/j3016_202104

[2] Cuenca, Á., Zhan, W., Salt, J., Alcaina, J., Tang, C., & Tomizuka, M. (2019). A Remote Control Strategy for an Autonomous Vehicle with Slow Sensor Using Kalman Filtering and Dual-Rate Control. *Sensors, 19*(13), 2983. https://doi.org/10.3390/s19132983

[3] Cuenca, Á., Alcaina, J., Salt, J., Casanova, V., & Pizá, R. (2018). A packet-based dual-rate PID control strategy for a slow-rate sensing Networked Control System. *ISA Transactions, 76*, 155-166. https://doi.org/10.1016/j.isatra.2018.02.022

[4] Salt, J., Cuenca, Á., Casanova, V., & Correcher, A. (2020). *Control automático. Tiempo continuo y tiempo discreto.* (2.a ed.). Editorial Reverté.

[5] Sanz, W. E. (2016). *Cinemática de Robots Industriales.* Createspace Independent Publishing Platform.

[6] Simon, D. (2006). *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches* (1ª ed.). John Wiley & Sons.

[7] Bar-Shalom, Y., Li, R. X., & Kirubarajan, T. (2001). *Estimation with Applications to Tracking and Navigation* (1ª ed.). John Wiley & Sons.

[8] Carbonell, R., Cuenca, Á., Casanova, V., Pizá, R., & Salt Llobregat, J. J. (2021). Dual-Rate Extended Kalman Filter Based Path-Following Motion Control for an Unmanned Ground Vehicle: Realistic Simulation. *Sensors*, *21*(22), 7557. https://doi.org/10.3390/s21227557

[9] MathWorks. (2022). *Pure Pursuit Controller - MATLAB & Simulink - MathWorks España.* Recuperado el 20 de agosto de 2022, de https://es.mathworks.com/help/nav/ug/pure-pursuit-controller.html

[10] INE. *(*2022). *Salarios medios brutos mensuales del empleo principal.* Recuperado el 20 de agosto de 2022, de https://www.ine.es/dynt3/inebase/index.htm?padre=2581&capsel=2661

FIGURAS:

[11] *Robot LEGO Mindstorms.* (2022). Recuperado el 10 de agosto de 2022 https://www.robotix.es/es/lego-mindstorms-education-ev3

[12] *Cámara cenital.* (2022). Recuperado el 10 de agosto de 2022 https://i.ebayimg.com/images/g/N-kAAOSw891hXpsh/s-l1600.jpg

[13] *Sistema de balizas.* (2022). Recuperado el 10 de agosto de 2022 https://marvelmind.com/product/starter-set-hw-v4-9-nia/

[14] *Placa AZDelivery ESP32.* (2022). Recuperado el 10 de agosto de 2022 https://www.az-delivery.de/es/products/esp32-d1-r32-board

[15] *Sensor de orientación IMU.* (2022). Recuperado el 10 de agosto de 2022 https://alitools.io/es/showcase/imu-sensor-de-ruptura-imu-bno055-ahrs-acelerometro-sip-giroscopio-triaxial-magnetometro-geomagnetico-orientacion-absoluta-32809065847