TRABAJO FINAL DE GRADO

Control basado en red inalámbrica de un robot móvil con técnicas multifrecuenciales

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: Enrique Vicente Lledó Molina

Tutor: Julián José Salt Llobregat

Valencia, Junio 2018

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería del Diseño

Resumen

En el presente proyecto se desarrollará el control basado en red de un robot móvil de bajo coste mediante técnicas multifrecuencia. El robot móvil, se ha elaborado para ser una plataforma de laboratorio para el grupo de investigación CO_3 del *Departamento de Ingeniería de Sistemas y Automática* con fines tanto formativos como de investigación. Mediante esta plataforma se desean comprobar los desarrollos teóricos, tanto de este proyecto como de futuras investigaciones.

Se trata de un robot móvil diferencial apoyado con dos ruedas locas encargadas de mantener la estabilidad. El bucle de control se dividirá en dos partes. La primera, en local mediante un microcontrolador del fabricante Arduino, será la encargada de calcular las acciones de control, aplicarlas y tomar las medidas de velocidad a partir de los encoders de los motores. La segunda, de forma remota, se encargará de calcular las referencias necesarias para realizar el seguimiento de una trayectoria determinada a partir de las velocidades antes obtenidas.

La comunicación entre ambas partes del control se realizará a través de una red de comunicación compartida, empleando para ello módulos de radiofrecuencia.

Además del seguimiento de trayectorias, se realizará un análisis energético para determinar como varía el consumo en función de los diferentes controles implementados.

PALABRAS CLAVE: Control basado en red, eficiencia, radiofrecuencia, retardo, multifrecuencia, seguimiento de trayectorias, consumo energético, robot móvil de bajo coste.

Abstract

In this project the network-based control of a low-cost mobile robot will be developed using multi-rate techniques. The mobile robot has been developed to be a laboratory platform for the research group CO_3 (*Department of Systems Engineering and Automation*) for both training and research purposes. Through this platform you want to check the theoretical developments, both of this project and future research.

It is a mobile differential robot supported with two ball casters in charge of maintaining stability. The control loop will be divided into two parts. The first, locally using a microcontroller manufactured by Arduino, will be responsible for calculating, applying control actions and take speed measurements from the encoders of the engines. The second, remotely, will be responsible for calculating the necessary references to track a given trajectory from the speeds previously obtained.

The communication between both parts of the control will be made through a shared communication network, using radio frequency modules for this purpose.

In addition to tracking trajectories, an energy analysis will be carried out to determine how the consumption varies according to the different controls implemented.

KEYWORDS: Network-based control, efficiency, radiofrequency, delay, multirate, path tracking, energy consumption, low cost mobile robot.

Resum

En aquest projecte es desenvoluparà el control basat en xarxa d´un robot mòbil de baix cost amb tècniques de *multifreqüència*. El robot mòbil, s´ha elaborat per a ser una plataforma de laboratori per al grup de recerca CO_3 del *Departament d´Enginyeria de Sistemes i Automàtica* amb fins tant formatius com d´investigació. Mitjançant aquesta plataforma es volen comprovar els desenvolupaments teòrics, tant d´aquest projecte com de futures investigacions.

Es tracta d´un robot mòbil diferencial recolzat amb dues rodes boges encarregades de mantenir l´estabilitat. El bucle de control es dividirà en dues parts. La primera, en local mitjançant un microcontrolador del fabricant Arduino, serà l´encarregada de calcular les accions de control, aplicarles i prendre les mesures de velocitat a partir dels encoders dels motors. La segona, de forma remota, s´encarregarà de calcular les referències necessàries per a realitzar el seguiment d´una trajectòria determinada a partir de les velocitats abans obtingudes.

La comunicació entre les dues parts del control es realitzarà a través d´una xarxa de comunicació compartida, emprant mòduls de radiofreqüència.

A més del seguiment de trajectòries, es realitzarà una anàlisi energètic per determinar com varia el consum en funció dels diferents controls implementats.

PARAULES CLAU: Control basat en xarxa, eficiència, radiofreqüència, retard, multifreqüència, seguiment de trajectòries, consum energètic, robot mòbil de baix cost.

Agradecimientos

En primer lugar, mi más sincero agradecimiento a mi tutor, Julián José Salt Llobregat por su ayuda y por darme la oportunidad de participar en este proyecto de colaboración.

En segundo lugar, y no por ello menos importante, a mi compañero y sobre todo amigo, Sergio Benavent Nácher, por su capacidad para darle la vuelta a los problemas y contagiarme con su optimismo.

A mi familia, por estar siempre ahí, en los buenos y no tan buenos momentos. Sin su apoyo, nada de esto hubiera sido posible.

Por último, no quiero olvidarme mi segunda familia. El Colegio Mayor San Juan de Ribera y todos los que integran la institución. Mi etapa universitaria no hubiera sido lo mismo sin ellos.

Índice general

Índice de figuras

Índice de tablas

Parte I

Introducción

1. Introducción

1.1. Objetivo

El objetivo del presente trabajo es realizar el control basado en red inalámbrica de un sistema mediante técnicas multifrecuencia y analizar las diferencias, desde el punto de vista de prestaciones, que tiene este tipo de control respecto al convencional. Por otro lado, también se ha querido analizar si desde el punto de vista enérgetico, las técnicas multifrecuencia ofrecen ventajas significativas respecto a las técnicas clásicas de control.

1.2. Motivación

Este proyecto surge en un contexto en el que se combinan dos realidades. Por un lado, cada día los sistemas de control automático tienen más peso. Esto ocurre porque estos permiten reducir los posibles fallos y actuar más rápidamente frente a una situación imprevista. Es decir, dan la posibilidad de trabajar de una forma más eficiente. Por el otro lado, es una realidad que a día de hoy prácticamente todo el mundo lleva consigo de forma habitual un móvil, una tablet o un ordenador personal; elementos que se han convertido en una herramienta necesaria para realizar todo tipo de tareas. De esta combinación surge de forma casi inmediata la idea buscar la posibilidad de controlar nuestros procesos desde cualquier lugar con tan sólo encender nuestro ordenador portátil o desbloquear nuestro dispositivo móvil.

Sin embargo, realizar un control de forma remota introduce en el propio problema del control complicaciones adicionales como retrasos variables en la señal o pérdidas de datos. Por este motivo, resulta realmente interesante analizar las diversas opciones que existen actualmente para afrontar dicho problema y comparar los resultados no solo desde el punto de vista del control, sino también desde el estudio de la eficiencia energética. Aunque *a priori* este punto de vista no parece importante para determinar si un control es válido o no, cuando el proceso que se desea controlar es un sistema móvil o que simplemente funciona haciendo uso de baterías, el consumo energético resulta un aspecto clave ya que influye de manera directa en la autonomía del mismo.

Añadir que, si bien resulta muy interesante el hecho de emplear un dispositivo Android/iOS para controlar el proceso en cuestión, el objeto del presente trabajo no es estrictamente ese. Lo que se pretende demostrar es la validez de las técnicas de control "multi-rate" que dan la posibilidad de controlar sistemas con tiempos de muestreo tales que las técnicas convencionales de control no puedan ofrecer una buena respuesta. Por este motivo, el proyecto se ha realizado comunicando el sistema a controlar (en este caso un robot móvil de bajo coste) con un ordenador personal situado en una ubicación remota.

Además, la realización del proyecto ha permitido, por un lado, poner en práctica y por tanto afianzar los conocimientos adquiridos durante el grado y, por otro lado, ampliar el conocimiento en ramas del control, como el control multifrecuencia, que si bien han sido mencionadas en diferentes las asignaturas cursadas, no han sido estudiadas en profundidad.

Por último, añadir que este proyecto se ha realizado en el marco de una beca de colaboración con el grupo *CO3 (Comunicaciones y Control por Computador)* del *Instituto de Investigación ai2 (Automática e Informática Industrial)* de la Universidad Politécnica de Valencia lo que ha permitido tener un contacto directo con un grupo de investigación y por tanto familiarizarse con la forma de trabajar en un ambiente de tales características.

1.3. Estructura del trabajo

El presente documento se divide en 5 partes diferenciadas que se van a explicar brevemente.

La primera de ellas, la **introducción**, incluirá tanto los objetivos fundamentales que se desean lograr con el proyecto, como las razones que han motivado la realización del proyecto y la estructura del mismo.

En segundo lugar, se encuentra el **marco teórico**. En él, se hará una introducción teórica, explicando con más o menos detalle, en función de la complejidad, todos aquellos conceptos y/o métodos necesarios para la realización del proyecto. Se comentarán conceptos comunes como el análisis de sistemas continuos y el diseño de un bucle de control tanto continuo como discreto. También se introducirán aspectos más específicos del proyecto como el control por red, la radiocomunicación y las técnicas de control *multi-rate*.

En tercer lugar, se describirán todas las **herramientas** empleadas en la realización del proyecto, desde los componentes del robot hasta los *software* empleados para la realización de los algoritmos.

La cuarta parte es la encargada de recoger todo el desarrollo práctico realizado. Su contenido se puede dividir en: modelado del sistema, diseño de controladores continuos, discretización y obtención del controlador *multi-rate*, análisis de los retardos e implementación del predictor de Smith y algoritmo de seguimiento de trayectorias.

La quinta, **resultados**, será dónde se mostrarán y analizarán los resultados de las diferentes pruebas realizadas. Esta parte puede subdividirse a su vez en tres partes diferenciadas: el seguimiento de trayectorias, el análisis del consumo energético y las conclusiones del proyecto y posibles líneas de trabajo futuro.

A continuación, se mostrará el **presupuesto** del proyecto incluyendo gastos tanto de personal como de material y licencias de software.

Por último, se adjuntarán como **anexos** aquellos scripts, códigos o diagramas que han sido necesarios para la realización del presente proyecto.

Parte II

Marco Teórico

2. Control automático

2.1. Teoría clásica de control

Para afrontar un problema de regulación automática mediante la teoría clásica de control es importante tener claro como se definen, desde esta perspectiva, los elementos con los que se va a trabajar y cuál es el método a seguir.

Un sistema, desde el punto de vista clásico, se define como un conjunto de componentes interrelacionados que reaccionan frente a variaciones en las variables consideradas de entrada dando lugar a variaciones en las variables de salida. Es decir, la perspectiva clásica representa los sistemas desde un punto de vista externo mediante una función de transferencia donde los términos del numerador corresponden a los ceros o sumideros del sistema mientras que los del denominador son los polos o fuentes del mismo. Tal y como se puede observar en la ecuación (2.1), la función de transferencia es la relación entre la variable manipulada (entrada) y la variable controlada (salida). Este tipo de representación solo es válida para sistemas lineales e invariantes con el tiempo y no tiene en cuenta los cambios ocurridos en otras variables internas del sistema. De la conjunción del concepto de sistema y del objetivo del control, se puede definir un sistema de control como aquel que consigue una configuración tal que la respuesta obtenida es la deseada [1].

$$G(s) = \frac{Y(s)}{R(s)} = \frac{\prod_{j=0}^{m}(s+z_j)}{\prod_{i=0}^{n}(s+p_i)} \tag{2.1}$$

Así, el éxito de la solución de control obtenida depende esencialmente del conocimiento de la función de transferencia del sistema. Para obtener dicha relación entrada-salida es necesario obtener un modelo matemático, ya sea a partir de las ecuaciones de balance del sistema o mediante métodos experimentales, que sirva para conocer su comportamiento frente a las diferentes excitaciones posibles.

La forma más inmediata de realizar un control es la que se conoce como bucle abierto (figura 2.1). En este tipo de control la salida no se compara con la referencia por lo que para cada salida, se aplica una entrada determinada y por tanto, la precisión del sistema dependerá en gran medida de la calibración previa del sistema.

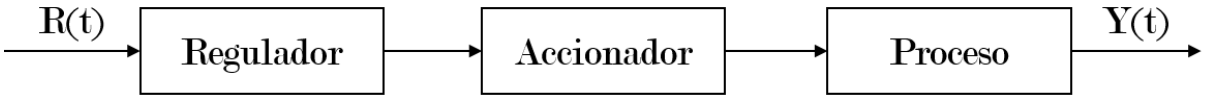


Figura 2.1: Diagrama de bloques genérico para control en bucle abierto

Sin embargo, en la mayoría de ocasiones, este tipo de control no permite conseguir los resultados esperados por dos motivos principalmente. El primero de ellos es que no es posible asumir total aislamiento entre un proceso y su entorno, es decir, cabe la posibilidad de que aparezcan perturbaciones externas que modifiquen el comportamiento del sistema. El segundo radica en la obtención del modelo matemático que se va a emplear para realizar el control. Si la respuesta de dicho modelo se comporta de forma similar, pero no exacta, a la del sistema real, la respuesta del sistema no será la esperada sino que presentará cambios más o menos significativos en función de la exactitud del modelo[2].

Ante esta situación, se opta por realizar un control en bucle cerrado (figura 2.2) cuya principal finalidad es minimizar la señal de error llegando, si es posible, a anularla. Esto es, la señal de entrada al regulador ya no es la referencia a seguir sino la diferencia entre dicho valor y la salida. De este modo, cuanto mayor sea el error mayores serán las acciones aplicadas para rectificar el comportamiento del sistema y, cuando la señal de salida alcance la referencia, las acciones de control serán mínimas.

El empleo de sistemas de control en lazo cerrado permite hacer al sistema más robusto frente a las perturbaciones y variaciones internas. Sin embargo, la estabilidad del mismo ya no queda asegurada por lo que será uno de los aspectos clave a tener en cuenta al realizar el diseño.

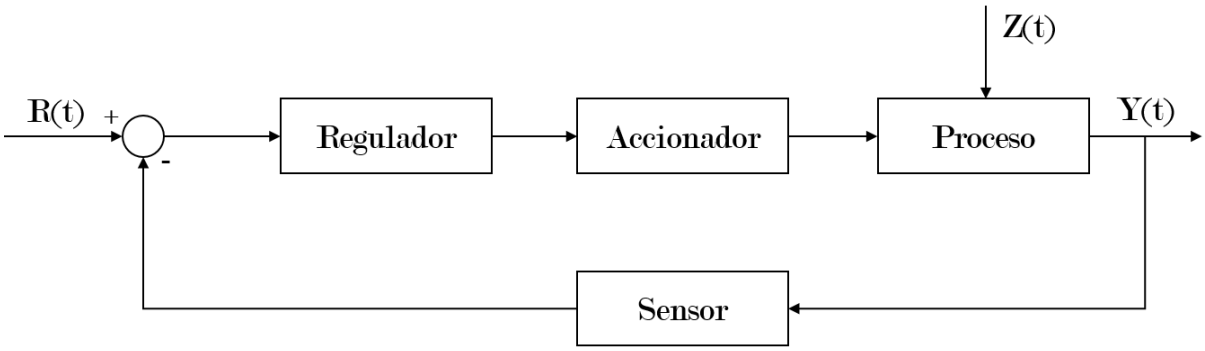


Figura 2.2: Diagrama de bloques genérico para control en bucle cerrado

Este tipo de sistemas controlados en bucle cerrado, se pueden clasificar según el resultado que se desee obtener del siguiente modo:

- **Servosistemas:** Lazos de control en los que la finalidad es que la señal de salida {Y(t)} siga fielmente la señal de referencia {R(t)} y sus variaciones. Es decir, se pretende seguir una señal que varía en el tiempo. Para ello se introduce en el bucle un regulador que se encargará de mejorar las características, tanto dinámicas como estáticas, de la respuesta del sistema.
- **Sistemas de regulación:** Lazos de control cuya finalidad es que la señal de salida {Y(t)} se estabilice en un valor deseado. En este caso la señal de referencia {R(t)} es constante o varía de forma muy lenta. En los sistemas de regulación también se introduce un regulador cuyos objetivos son conseguir que la salida alcance el valor y rectificar el efecto de las posibles perturbaciones {Z(t)} que modifiquen el comportamiento del sistema.

Así, al enfrentarse a un problema de control es necesario tomar decisiones respecto a dos aspectos fundamentales. En primer lugar, la estructura de control a implementar es una decisión esencial. Si bien el lazo de control típico es el mostrado anteriormente en la figura 2.2, existen estructuras complementarias que mejoran la respuesta del sistema ante determinadas situaciones. Hablamos de estructuras como el control en cascada o la prealimentación en caso de existencia de perturbaciones o el predictor de Smith en caso de que el sistema presente retardos significativos, es decir, retardos de un orden de magnitud similar o mayor al de las constantes de tiempo del sistema.

En segundo lugar, es importante decidir qué tipo de controlador se desea implementar en el bucle de control. Para tomar esta decisión es necesario tener en cuenta aspectos como las especificaciones que se desean conseguir, las acciones de control máximas que se pueden aplicar, los tiempos de respuesta deseados y la robustez frente a cambios bruscos y/o perturbaciones que se desea obtener.

Los reguladores habitualmente más implementados son:

- **Reguladores On/Off:** Empleado en sistemas en los que no se necesita un valor concreto sino un rango.
- **Reguladores por Cancelación:** Válido para aquellos casos en los que el modelo es muy bueno. En caso de no aplicarse sobre un buen modelo del proceso puede dar lugar a una mala cancelación y respuestas diferentes de las esperadas pudiendo llegar a inestabilizar el sistema.
- **Reguladores Tiempo Mínimo/Finito:** Cuando es muy relevante seguir las referencias en el mínimo tiempo posible y se dispone de un actuador para el cual elevadas acciones de control no suponen un problema.
- **Reguladores PID:** Son los más empleados por su buena respuesta dinámica y su fácil diseño. Permite combinar las acciones proporcional, integral y derivada tal y como se desee para obtener el regulador que mejor se adapte a las especificaciones de diseño.

2.2. Teoría moderna de control

La teoría clásica de control explicada a lo largo del apartado 2.1 es válida, como ya se ha mencionado anteriormente, para sistemas SISO (*Single Input Single Output*), lineales e invariantes en el tiempo. Sin embargo, la realidad es que para sistemas complejos esta situación no se da por lo que ha sido necesaria una nueva perspectiva de control que permita realizar el análisis y el diseño de los sistemas de regulación.

La base de esta nueva forma de plantear el control de un sistema es que ya no solo se tiene en cuenta la relación entrada - salida sino también las variables internas o de estado del proceso. Así, "*el estado de un sistema dinámico se define como el conjunto de variables más pequeño que determinan el estado del sistema dinámico*". Es decir, conociendo el valor de dichas variables en el instante inicial y las entradas a partir de ese momento, es posible conocer el comportamiento del sistema en cualquier instante de tiempo [2].

La representación del sistema desde esta perspectiva moderna se basa en la descripción del mismo en términos de n ecuaciones diferenciales de primer orden, que se combinan en un sistema matricial de primer orden compuesto por las ecuaciones (2.2) y (2.3) donde **A** se denomina matriz de estado, **B** matriz de entrada, **C** matriz de salida y **D** matriz de transmisión directa.

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \tag{2.2}$$

$$y(t) = C(t)x(t) + D(t)u(t) \tag{2.3}$$

En la ecuación (2.2), llamada ecuación de estados, se describe el comportamiento de las variables de estado en función de su propio valor en instantes anteriores y de las entradas que se introducen en el sistema. Mientras que el comportamiento de las variables de salida en función de los valores de los estados y las entradas quedará descrito por la ecuación (2.3).

Así la representación en diagrama de bloques de un sistema resultante de trasladar las ecuaciones quedará tal y como muestra la figura 2.3.

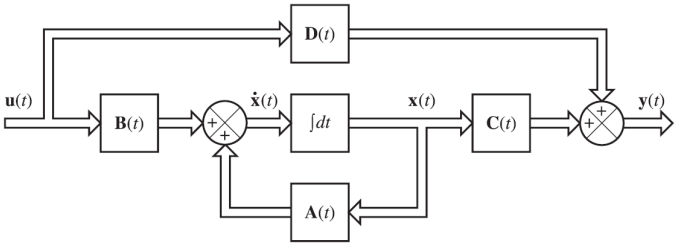


Figura 2.3: Diagrama de bloques genérico de un sistema en el espacio de estados

Mencionar que si bien la teoría moderna de control emplea la representación en espacio de estados para describir los sistemas, es posible obtener a partir de ella la matriz de transferencias mediante la cual definía un sistema la teoría clásica aplicando la relación mostrada en la ecuación (2.4). Sin embargo, al tratar sobre sistemas MIMO (*multiple input - multiple output*) el resultado obtenido no será una función de transferencia si no una matriz de funciones de transferencia tal y como muestra la ecuación (2.5) donde cada término M_{ij} es la función de transferencia que relaciona la salida **i** con la entrada **j**.

$$G(s) = C(sI - A)^{-1}B + D \tag{2.4}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & \dots & M_{1m} \\ M_{21} & M_{22} & \dots & M_{2m} \\ \vdots & & \ddots & \vdots \\ M_{n1} & M_{n2} & \dots & M_{nm} \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} \tag{2.5}$$

Destacar que $|sI - A|$ es igual al polinomio característico de la función de transferencia G(s) por lo que los valores propios de A se corresponden con los polos de G(s).

Por otro lado, para pasar de la función de transferencia a una representación de estados la solución no es única siendo las representaciones más empleadas las siguientes:

- **Canónica de Controlabilidad:** Garantiza que el modelo resultante es totalmente controlable. Esto es debido a que la representación del sistema es tal que el control entra a una cadena de integradores puede modificar todos y cada uno de los estados.
- **Canónica de Observabilidad:** Garantiza que el modelo resultante es totalmente observable. Esto es debido a que la representación del sistema es tal que la salida proviene de una cadena de integradores viéndose afectada por todos y cada uno de los estados.
- **Jordan:** Se caracteriza porque la matriz de estados (A) es una matriz diagonal formada por los valores propios del polinomio característico del sistema.

3. Análisis de los sistemas continuos

Realizar el diseño del regulador y seleccionar la estructura y/o estructuras complementarias del bucle de control dependerá en gran manera del sistema que se desee controlar y de las condiciones en que este vaya a trabajar. Por este motivo, para realizar el diseño del lazo de control es necesario obtener previamente un modelo matemático que nos permita conocer la dinámica propia del sistema dándonos la posibilidad de añadir un nuevo elemento cuya dinámica permita que el conjunto tenga el comportamiento deseado.

En este caso, se abordará el problema de control de los motores desde la perspectiva clásica por lo que el modelo del sistema que se obtendrá será la función de transferencia (apartado 11). Es importante recordar que para afrontar el diseño del control desde la teoría clásica es necesario asumir que el sistema con el que se va a trabajar es invariante en el tiempo y lineal. Esto es, aunque el sistema real no cumpla estos dos requisitos, se asumirá que lo hace y por tanto el modelo obtenido será una aproximación del proceso cuya precisión será mayor cuanto menores sean las variaciones entorno al punto de trabajo para el cual se ha obtenido la función de transferencia.

3.1. Estabilidad de un sistema

Uno de los aspectos clave a analizar de un sistema es su estabilidad absoluta. Se dice que un sistema es estable si ante una entrada acotada, su salida también lo es. Un sistema de control estable se caracteriza porque ante una determinada entrada, si no existen perturbaciones externas, la salida se mantiene en el mismo estado. En el caso contrario, un sistema inestable es aquel que ante una excitación diverge de su estado de equilibrio. Por último, si se cumple que el sistema queda oscilando de forma indefinida entorno al valor final se dirá que el sistema es críticamente estable [2].

La estabilidad de un sistema es una condición intrínseca del proceso, es decir, viene determinada por la dinámica del sistema ya que depende de las raíces de la ecuación característica del sistema. Desde la perspectiva clásica, esta característica se puede analizar a través del denominador de la función de transferencia debido a que este está formado por los polos del sistema ($p = \sigma + j\omega$). La condición que se debe cumplir para que un sistema sea estable es que la parte real de todos los polos del sistema sea negativa.

Por otro lado, es importante destacar que los polos del sistema determinan su estabilidad en bucle abierto. Un herramienta muy útil para conocer la estabilidad de un sistema en bucle cerrado, entre otras cosas, es el lugar de las raíces. Este, permite conocer la evolución de los polos y por tanto de la respuesta del sistema en bucle cerrado a partir de la función de transferencia en bucle abierto.

3.2. Comportamiento dinámico de un sistema continuo y lineal

Un sistema de control implica un almacenamiento de energía. Esto significa que al someter un sistema a una entrada, la salida no variará de forma inmediata a su nueva posición de equilibrio sino que necesitará un tiempo para evolucionar hasta su nuevo estado estacionario. Este intervalo de evolución de la respuesta del sistema se conoce como régimen transitorio. Por tanto la respuesta del sistema se puede escribir como

$$y(t) = y_{tr}(t) + y_{ss}(t)$$

donde el primer término corresponde con la respuesta transitoria del sistema y el segundo con la correspondiente al régimen permanente (*steady state*).

Así, en función de las características de la respuesta ante una entrada tipo escalón es posible hacer una primera clasificación de los sistemas en sobreamortiguados, subamortiguados y marginalmente estables u oscilatorios puros.

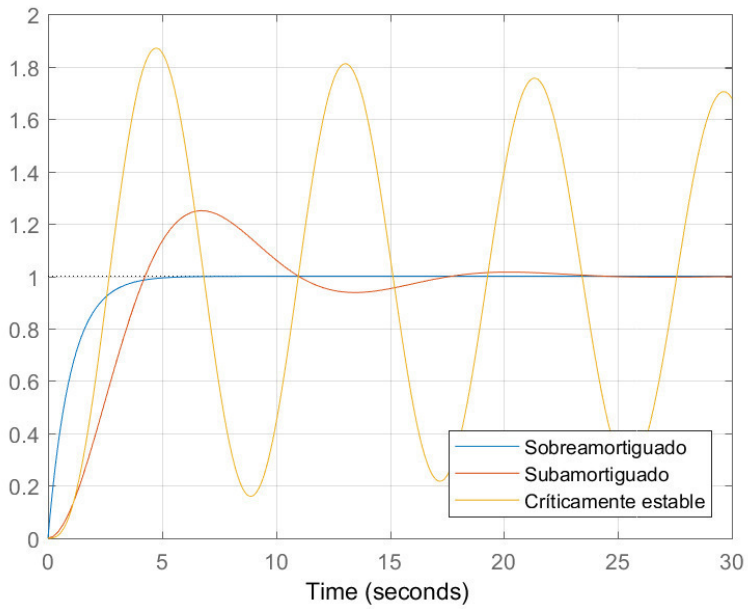


Figura 3.1: Tipos de respuesta frente escalón

Se dice que un sistema es **sobreamortiguado** cuando en ningún momento del transitorio se sobrepasa el valor final, es decir; el sistema no presenta ninguna oscilación. Esto sucede cuando todos los polos del sistema son reales ya que la respuesta en este caso es de tipo exponencial y por tanto no presenta oscilaciones.

$$\mathcal{L}^{-1}\left\{\frac{1}{s+a}\right\} = e^{-at} \tag{3.1}$$

Por el contrario, si el sistema sobrepasa (para luego volver) el valor en que se va a estabilizar en el régimen estacionario, el sistema que se está tratando es de tipo **subamortiguado**. Sucede cuando el sistema incluye un par de polos complejo conjugados. Ahora, la respuesta introduce un término correspondiente a un seno por lo que se comporta de forma oscilatoria aunque la parte exponencial va limitando progresivamente su amplitud.

$$\mathcal{L}^{-1}\left\{\frac{1}{(s+\sigma^2)+\omega^2}\right\} = e^{-at}sin(\omega t) \tag{3.2}$$

Un caso especial es el que se encuentra al límite entre la estabilidad y la inestabilidad. Es aquel en que los polos se encuentran sobre el eje imaginario, es decir, su parte real es nula. Se define un sistema de estas características como críticamente estable u oscilatorio puro por la respuesta que presenta.

$$\mathcal{L}^{-1}\left\{\frac{1}{(s+\sigma^2)+\omega^2}\right\}\Bigg|_{\sigma\to 0} = e^{-at}sin(\omega t)\big|_{\sigma\to 0} = sin(\omega t) \tag{3.3}$$

En resumen, un sistema es estable si su transitorio tiene duración finita y todos sus polos tienen la parte real negativa. En términos de estabilidad relativa, un sistema será más estable que otro cuanto más a la izquierda tenga sus polos, es decir, cuanto menor sea la parte real de sus polos.

Además del comportamiento dinámico del sistema, también es importante analizar el comportamiento estático una vez alcanzado el régimen permanente. Esto es, una vez finalizado el transitorio el sistema puede haber alcanzado la referencia de entrada o no. En caso de no haberla alcanzado se dirá que el sistema presenta **error en estado estacionario** que se podrá clasificar según las referencias que no se consiga seguir del siguiente modo:

- **Error de posición:** Error en el seguimiento de referencias de tipo escalón.
- **Error de velocidad:** Error en el seguimiento de referencias de tipo rampa.
- **Error de aceleración:** Error en el seguimiento de referencias de tipo parábola.

3.3. Sistemas continuos característicos

Habitualmente, al modelar un sistema físico se suele recurrir a funciones de transferencia de ordenes bajos ya que en la mayoría de los casos representan con suficiente precisión el comportamiento del sistema. Tan solo se recurre a sistemas de mayor orden cuando se desea o necesita una mayor exactitud en el modelo. Por este motivo, se va a profundizar en las respuestas de los sistemas de primer y segundo orden.

3.3.1. Sistemas de primer orden

El caso más sencillo de sistema con un solo polo es el conocido como integrador y se caracteriza por tener dicho polo en el origen y un numerador constante. Es un caso especial puesto que se comporta tratando de "integrar" alguno de los elementos del sistema [3].

Sin embargo, no es habitual que se produzca una acumulación instantánea, por lo que en la mayoría de casos los sistemas presentan otro término que suaviza la respuesta. Así la expresión del sistema sería de la forma:

$$a_1 \frac{dy(t)}{dt} + a_0\ y(t) = b_0\ u(t) \tag{3.4}$$

Para obtener la función de transferencia de un sistema genérico de primer orden tan solo queda transformar la ecuación (3.4) al dominio de Laplace y obtener el cociente entre la salida y la entrada.

$$(a_1 s + a_0)Y(s) = b_0\ U(s) \tag{3.5}$$

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_0}{a_1 s + a_0} \tag{3.6}$$

De forma normalizada se suele expresar en función de dos parámetros. La ganancia estática (K_{est}), que es la que representará la relación entrada - salida una vez finalizado el transitorio y la constante de tiempo (τ), que dará una idea de la rapidez del sistema. La expresión normalizada quedará del siguiente modo:

$$G(s) = \frac{Y(s)}{U(s)} = \frac{K_{est}}{1 + \tau s} \tag{3.7}$$

siendo la ganancia estática $K_{est} = \frac{b_0}{a_0}$ y la constante de tiempo $\tau = \frac{a_1}{a_0}$

Este tipo de sistemas suelen estudiarse analizando la respuesta ante una entrada de tipo escalón. La respuesta de estos sistemas para una entrada en escalón de amplitud A será de la siguiente forma:

$$Y(s) = G(s) \cdot U(s) = \frac{K_{est}}{1 + \tau s} \cdot \frac{A}{s} \Rightarrow y(t) = AK(1 + e^{\frac{-t}{\tau}}) \tag{3.8}$$

De esta expresión se puede deducir que la respuesta tendrá dos componentes. La primera de ellas, correspondiente al régimen estacionario, será de la forma $y(t) = AK$ y la segunda de ellas determinará el comportamiento durante el transitorio siendo su expresión $y(t) = AKe^{\frac{-t}{\tau}}$.

Del análisis de la expresión de la respuesta del sistema, podemos extraer que la constante de tiempo del sistema es aquel instante de tiempo en que la señal de salida ha alcanzado el 63.2 % de su valor final.

$$y(\tau) = AK(1 + e^{\frac{-\tau}{\tau}}) = AK(1 - 0.368) = \mathbf{0.632}AK \tag{3.9}$$

Con un análisis similar, podemos determinar el valor aproximado del tiempo de establecimiento o tiempo en que tarda el sistema en alcanzar el régimen permanente. Como la respuesta del sistema es asintótica se suele asumir que se ha alcanzado régimen estacionario al haber superado un porcentaje del valor final. Los criterios más empleados son:

- **Criterio del 95 %:** Se da en aproximadamente un tiempo de 3τ
  $y(3\tau) = AK(1 + e^{\frac{-3\tau}{\tau}}) = AK(1 - 0.05) = \mathbf{0.95}AK$
- **Criterio del 98 %:** Se da en aproximadamente un tiempo de 4τ
  $y(4\tau) = AK(1 + e^{\frac{-4\tau}{\tau}}) = AK(1 - 0.02) = \mathbf{0.98}AK$

A modo de conclusión, la figura 3.2 muestra las principales características de la respuesta de un sistema de primer orden.

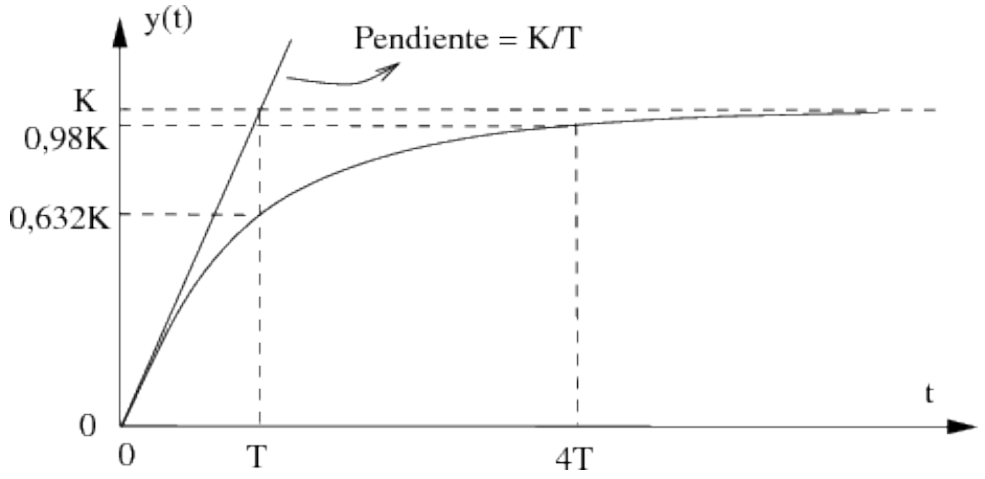


Figura 3.2: Respuesta de un sistema de primer orden

3.3.2. Sistemas de segundo orden

El análisis de los sistemas de segundo orden se realizará de forma análoga a la realizada para los sistemas de primer orden. En este caso, la función de transferencia del sistema procede de una ecuación diferencial de segundo orden.

$$\frac{d^2y(t)}{dt} + a_1\frac{dy(t)}{dt} + a_0\ y(t) = b_0\ u(t) \tag{3.10}$$

$$(s^2 + a_1 s + a_0)Y(s) = b_0\ U(s) \tag{3.11}$$

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_0}{s^2 + a_1 s + a_0} \tag{3.12}$$

De forma normalizada se suele expresar la función de transferencia en relación a una serie de parámetros característicos de las respuestas de este tipo de sistemas.

$$G(s) = \frac{Y(s)}{U(s)} = \frac{K\omega_n^2}{s^2 + 2\xi\omega_n + \omega_n^2} \tag{3.13}$$

siendo $\omega_n = \sqrt{a_0}$ la pulsación natural del sistema, $K = \frac{b_0}{a_0}$ la ganancia y $\xi = \frac{a_1}{2\sqrt{a_0}}$ el coeficiente de amortiguamiento.

Resolviendo la ecuación característica del sistema (denominador de la función de transferencia) se obtiene que las raíces serán de la forma:

$$s = \omega_n(-\xi \pm \sqrt{\xi^2 - 1}) \tag{3.14}$$

En función del valor del coeficiente de amortiguamiento, las polos del sistema serán diferentes y por tanto la respuesta del sistema también lo será.

- $\xi > \mathbf{1}$: Se tiene un sistema cuyos polos son reales y diferentes. La respuesta del sistema será sobreamortiguada.
- $\xi = \mathbf{1}$: Se tiene un sistema cuyos polos son dobles y situados en $s = -\omega_n$. La respuesta del sistema será sobreamortiguada.
- $\mathbf{0} < \xi < \mathbf{1}$: Se tiene un sistema cuyos polos son una pareja complejo conjugada de la forma $s = -\xi\omega_n \pm \sqrt{1 - \xi^2}\ \omega_n j$. La respuesta del sistema será subamortiguada.

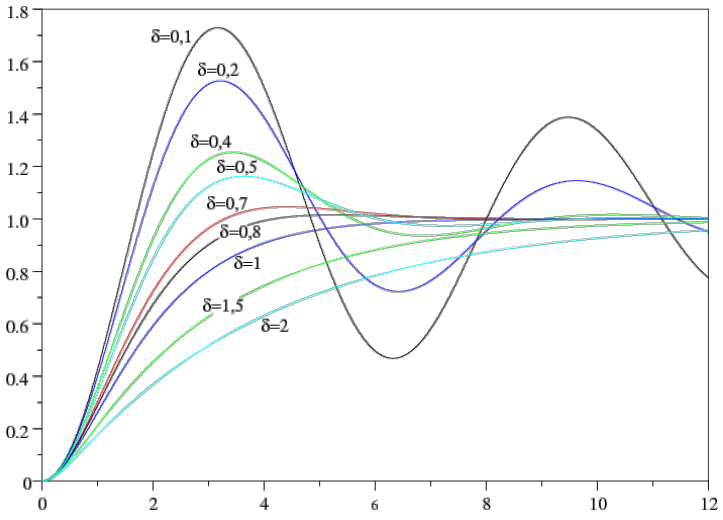


Figura 3.3: Respuesta de un sistema de segundo orden

Igual que en el caso de los sistemas de primer orden, la situación de los polos puede ayudar a conocer las características del sistema objeto de estudio.

En el caso de que el sistema esté formado por dos polos reales, se tendrá que la respuesta obtenida será análoga a la obtenida al disponer dos subsistemas de primer orden en serie. No se alcanzará el régimen permanente hasta que lo haga el subsistema más lento, es decir, el polo lento ralentizará la respuesta transitoria.

En el caso de la existencia de polos complejos, es necesario introducir dos parámetros necesarios para caracterizar la respuesta del sistema. Estos son la sobreoscilación porcentual y el tiempo de pico.

La sobreoscilación porcentual es una medida de amplitud que indica la relación entre el valor máximo de la salida y su valor en el régimen permanente.

$$\delta(\,\%) = \frac{y_{max} - y(\infty)}{y(\infty)} \cdot 100 \tag{3.15}$$

Así, el tiempo de pico se define como el instante temporal en que la señal alcanza su valor máximo. Obteniendo la expresión de dicho valor máximo de salida, se puede obtener fácilmente la relación entre el tiempo de pico y la sobreoscilación porcentual.

$$t_p = \frac{\pi}{\omega_p} = \frac{\pi}{\omega_n\sqrt{1-\xi^2}} \tag{3.16}$$

$$y_{max} = AK(1 + e^{-\frac{\xi\pi}{\sqrt{1-\xi^2}}}) \tag{3.17}$$

$$\delta(\,\%) = e^{-\frac{\xi\pi}{\sqrt{1-\xi^2}}} \cdot 100 = e^{-\sigma t_p} \tag{3.18}$$

El tiempo de establecimiento, de forma análoga a los sistemas de primer orden, dependerá por un lado del criterio escogido para la determinación del régimen estacionario y por el otro lado del valor de la "constante de tiempo" de este tipo de sistemas directamente relacionada con la parte real de los polos complejo conjugados.

$$t_{est}(95\,\%) = \frac{\pi}{\sigma} \qquad t_{est}(98\,\%) = \frac{4}{\sigma} \tag{3.19}$$

siendo $\sigma = \xi\omega_n$

Añadir que las relaciones entre variables empleadas en las ecuaciones (3.16),(3.17) y (3.18) se obtienen a partir de la caracterización geométrica de los polos en el plano S. Destacar que en la figura 3.4 solo se representa la parte superior del eje imaginario puesto que la posición de los polos será simétrica respecto al eje real.

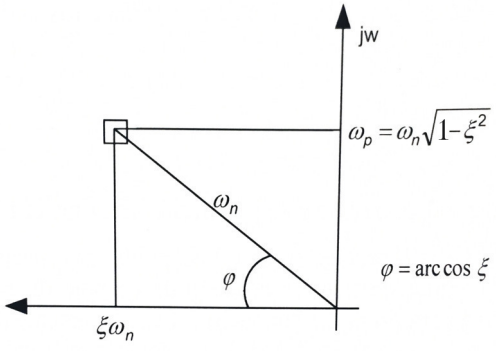


Figura 3.4: Caracterización polos en el plano S

4. Diseño del bucle de control

El objetivo fundamental del control automático es, como ya se ha mencionado en apartados anteriores, modificar la dinámica propia de un sistema para obtener una nueva dinámica tal que la respuesta del conjunto sea la deseada. Una vez obtenido el modelo matemático del proceso, esto es, las ecuaciones que rigen la dinámica del mismo, el siguiente paso es modificarla y transformarla en la deseada. Sin embargo, rara vez es posible modificar el comportamiento de un sistema físico. Por este motivo, será necesario diseñar un elemento externo de control que atendiendo a las especificaciones de diseño tenga una dinámica tal que consiga que la respuesta del sistema global sea la deseada.

Tal y como se ha mencionado en el apartado 2.1 la estructura de control a implementar puede ser bien en lazo abierto o cerrado. En este caso, como se desea seguir una referencia de velocidad, se empleará un control en bucle cerrado mediante un servosistema.

4.1. Obtención del regulador

Existe gran variedad de reguladores en función de los objetivos que se deseen conseguir. Desde controladores con acciones de control de tipo todo/nada hasta reguladores de tiempo mínimo o tiempo finito que permitan seguir la referencia en el mínimo número de instantes de tiempo.

En este caso, se va a emplear un regulador de tipo PID por ser con diferencia el más empleado a nivel industrial y por su fácil dieño. Este tipo de controlador combina las tres acciones básicas de control: acción proporcional (acción de presente), integral (de pasado) y derivativa (de futuro)

$$U(t) = K_P \cdot e(t) + K_I \cdot \int e(t)dt + K_D \cdot \frac{de(t)}{dt} \tag{4.1}$$

$$\mathbf{G_{PID}(s)} = \frac{\mathbf{U(s)}}{\mathbf{E(s)}} = \mathbf{K_P} + \mathbf{K_I} \cdot \frac{\mathbf{1}}{\mathbf{s}} + \mathbf{K_D} \cdot \mathbf{s} \tag{4.2}$$

El proceso de diseño del regulador consistirá en determinar que cantidad de cada una de las tres acciones de control básicas es necesario añadir al sistema para que su respuesta cumpla con las especificaciones de diseño. Para ello se variarán las constantes $K_P,\ K_D\ y\ K_I$.

Todo regulador PID ha de incluir una parte proporcional ya que su finalidad es eliminar el error y esto no es posible con las otras dos acciones básicas de control por separado. Un controlador tipo integral solo sería capaz de eliminar el error una vez acumulado y uno derivativo únicamente sería capaz de hacer ese error constante. Por este motivo, es necesario analizar en función de las características dinámicas y estáticas (figura 4.1) del sistema que combinación de las 3 acciones de control son necesarias siendo las opciones: P, PI, PD y PID.

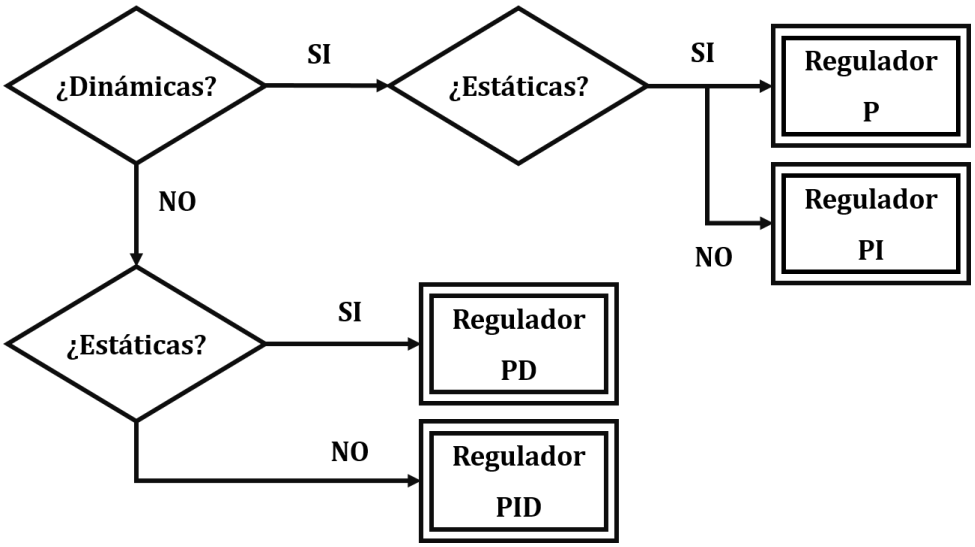


Figura 4.1: Diseño de un regulador tipo PID

4.1.1. Diseño por cancelación

El método más sencillo para la obtención de un regulador tipo PID es el de cancelación. Este consiste en colocar los polos y ceros del controlador para anular total o parcialmente la dinámica propia del sistema y, posteriormente, ajustar la ganancia proporcional para desplazar los polos del sistema por el lugar de las raíces y obtener la respuesta deseada.

La forma más habitual de trabajar es cancelar los polos más lentos del sistema. De esa forma añadiendo acción integral podemos eliminar uno de los polos lentos y con la acción derivada el segundo. Un ejemplo de diseño sería el siguiente:

$$G_P = \frac{1}{(s+a_0)(s+a_1)} \Rightarrow \mathbf{G_R} = \mathbf{K_P}\frac{(\mathbf{s}+\mathbf{a_0})(\mathbf{s}+\mathbf{a_1})}{\mathbf{s}}$$

Sin embargo, este método no se suele emplear porque para que el diseño del controlador sea válido es necesario que la cancelación de polos sea exacta y en caso contrario se pueden dar respuestas diferentes a las deseadas e incluso la inestabilidad.

Conseguir esa cancelación exacta es complejo principalmente por que al identificar la dinámica del sistema se suele obtener un modelo aproximado que introduce algo de error en el sistema. Esa aproximación hace que los polos y ceros del sistema no sean exactos y por tanto dificulta la obtención de un buen control mediante este método.

4.1.2. Diseño mediante el lugar de las raíces

El diseño de un regulador PID mediante el lugar de las raíces necesita, por un lado, la función de transferencia del sistema en bucle abierto y su trazado del lugar de las raíces y por otro lado, las especificaciones dinámicas y estáticas que se desean conseguir.

Las especificaciones dinámicas se refieren a las características de la respuesta en el régimen transitorio, mientras que las estáticas tienen aplicación en el permanente.

El primer paso para el diseño de un PID mediante este método es el trazado de la zona de especificaciones en el lugar de las raíces. Se trata de dividir el plano complejo en las zonas de puntos que cumplen dichas especificaciones dinámicas (tiempo de establecimiento, sobreoscilación porcentual...) y las que no. Una vez realizado, se debe seleccionar el punto de diseño. Si bien cualquier punto de la zona que cumple especificaciones es válido a priori, es posible que para alguno de ellos la solución no sea válida por las características físicas del sistema.

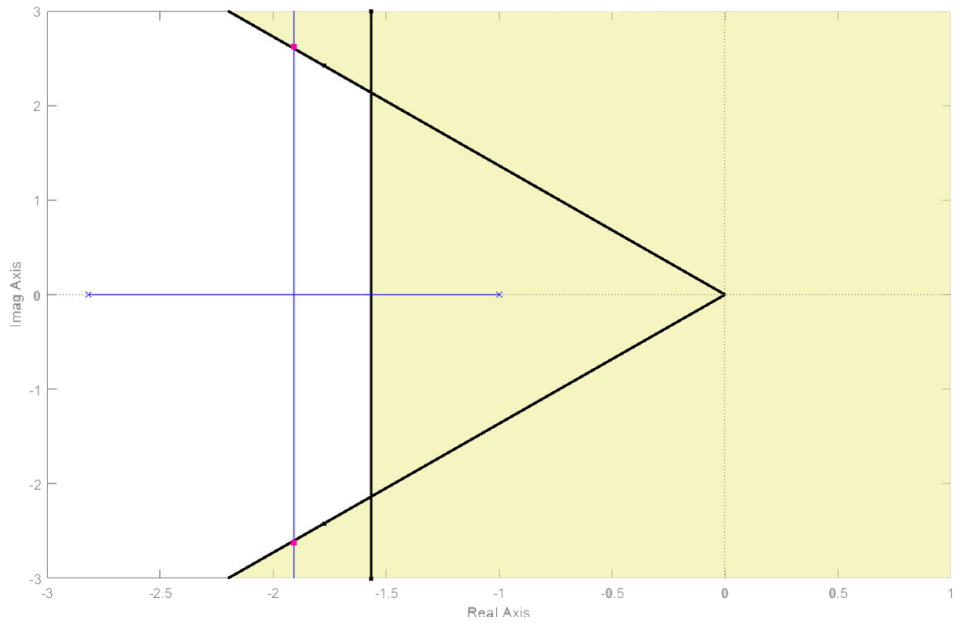


Figura 4.2: Diseño mediante el lugar de las raíces

Por último, se añaden los polos y ceros del regulador de forma que al modificar el lugar de las raíces del sistema, el resultante pase por el punto de diseño donde se quiere trabajar. Para ello se emplearán las siguientes reglas:

- **Criterio del argumento:** Sirve para determinar donde se deben colocar los ceros del regulador de forma que el punto de diseño pertenezca al lugar de las raíces.

$$\sum \theta_{z_i} - \sum \theta_{p_j} = (2k+1) \cdot 180,\ k = 0, 1, 2 \dots$$

- **Criterio del módulo:** Sirve para determinar cuál debe ser la ganancia del sistema para que los polos se encuentren en un punto determinado del lugar de las raíces.

$$K_{LDR} = \frac{\prod d_{z_i}}{\prod d_{p_j}}$$

4.2. El Predictor de Smith

La presencia de retardos de transporte en el bucle de control conlleva una pérdida de efectividad del controlador dando lugar a respuestas diferentes de las deseadas e incluso la inestabilidad. La estructura propuesta por Smith en los años 60 fue y sigue siendo la solución que se emplea habitualmente para solucionar este tipo de problemas.

La gran ventaja que ofrece el uso del Predictor de Smith es la posibilidad de realizar el diseño del lazo de control sobre el modelo del sistema sin tener en cuenta el retardo y posteriormente crear una estructura complementaria al bucle primario que permita eliminar la influencia de dicho retardo.

Es importante recordar que un retardo temporal en el dominio de Laplace se modela como una función exponencial de la forma e^{-Ts} donde T es la magnitud del retardo.

Si el retardo que aparece en el sistema es de magnitud mucho menor que las constantes de tiempo del proceso se optará por realizar el desarrolllo en serie de Taylor (ecuación 4.3) o de Padè (ecuación 4.4) de la función exponencial y aproximar el retardo por una función de transferencia con ceros y/o polos que se añadirá a la función de transferencia del sistema para realizar de nuevo el diseño del controlador.

$$e^{-Ts} = \frac{1}{e^{Ts}} = \frac{1}{1 + Ts + \frac{T^2 s^2}{2} + \dots} \approx \frac{1}{1 + Ts} \tag{4.3}$$

$$e^{-Ts} = \frac{e^{-Ts/2}}{e^{Ts/2}} = \frac{1 - \frac{Ts}{2} + \frac{(\frac{Ts}{2})^2}{2} + \dots}{1 + \frac{Ts}{2} + \frac{(\frac{Ts}{2})^2}{2} + \dots} \approx \frac{1 - \frac{Ts}{2}}{1 + \frac{Ts}{2}} \tag{4.4}$$

Sin embargo hay casos en los que el retardo es significativo, es decir, del mismo orden de magnitud que las constantes de tiempo del sistema. En estas ocasiones se implementará un predictor de Smith.

La idea de este tipo de estructura es conseguir la respuesta que se obtendría en el lazo de control sin retardo, pero retardada. Este planteamiento queda representado en la figura 4.3. Si fuera posible acceder a la señal Y'(s), el problema estaría solucionado ya que se podría realimentar la salida para la cuál se ha diseñado el bucle de control. Destacar que el objetivo de esta estructura no es eliminar el retardo en la respuesta puesto que este es intrínseco al sistema y por tanto no se puede eliminar.

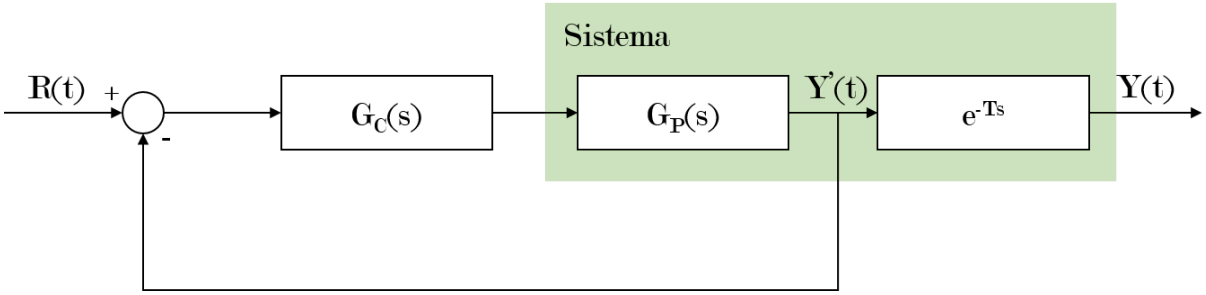


Figura 4.3: Propuesta ideal de tratamiento del bucle de control con retardo.

Como realimentar la salida antes de que se vea afectada por el retardo no es posible, la siguiente opción es realimentar la salida del controlador con el modelo del proceso ($G'_P(s)$) de forma que pueda predecir la salida y por tanto el regulador calcular las siguientes acciones de control. Pero tal y como muestra la figura 4.4, la salida queda sin realimentar por lo que el sistema está en bucle abierto y es vulnerable a perturbaciones.

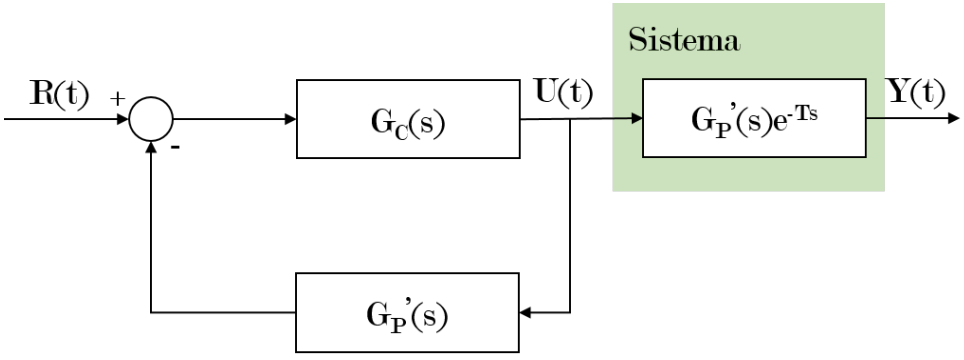


Figura 4.4: Propuesta real inicial de tratamiento del bucle de control con retardo.

Para solucionar este problema se realimentará la acción de control a través del modelo del sistema incluyendo el retardo. Tal y como muestra la siguiente imagen, el bucle de control estimará las salidas tanto sin retardo como con él y calculará las acciones de control correspondientes. Mencionar que en los diagramas de bloques, las magnitudes y/o funciones de transferencia con superíndice prima serán aquellas que sean modelos o estimaciones.

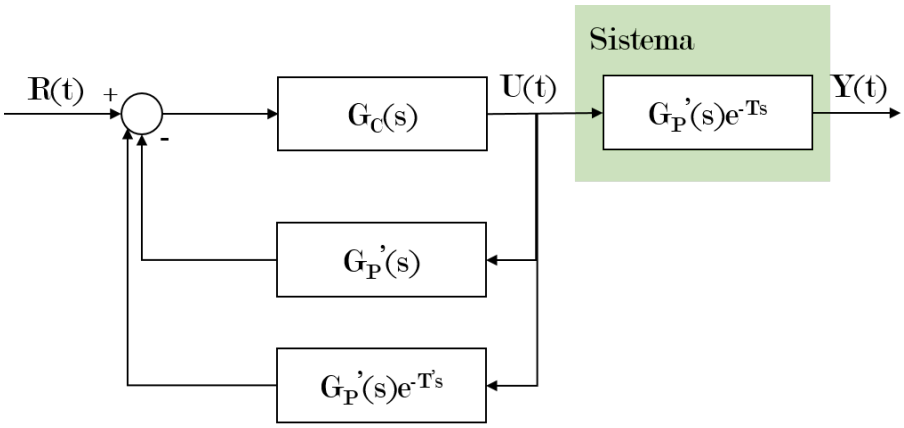


Figura 4.5: Propuesta real de tratamiento del bucle de control con retardo.

Operando mediante el álgebra de bloques el lazo de control queda como el de la figura 4.6, donde el nuevo regulador (sobre fondo azul) incluye el regulador diseñado previamente para el sistema sin retardo y la estructura del predictor de Smith. La nueva ecuación del regulador será:

$$G'_R(s) = \frac{G_R(s)}{1 + G_P(s)G_R(s)(1 - e^{-Ts})} \tag{4.5}$$

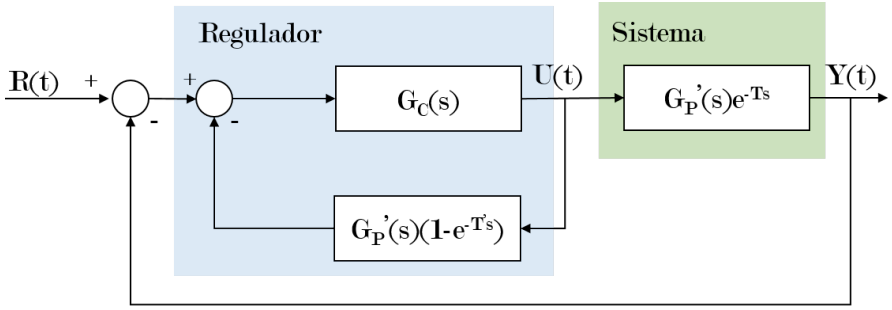


Figura 4.6: Bucle de control con predictor de Smith

Para comprobar el correcto funcionamiento de esta estructura, solo es necesario obtener la nueva función de transferencia del bucle completo.

$$M_{Con\_retardo} = \frac{G_R(s)G_P(s)e^{-Ts}}{1 + G_R(s)G_P(s) - G_R(s)G'_P(s)e^{-T's} + G_R(s)G_P(s)e^{-Ts}} \tag{4.6}$$

En el caso de que las estimaciones, tanto del retardo como del modelo, sean muy precisas se podría concluir que el nuevo bucle de control se comporta tal y como el inicial pero con un retardo.

$$M_{con\_retardo} = \frac{G_R(s)G_P(s)e^{-Ts}}{1 + G_R(s)G_P(s)} = M_{sin\_retardo} \cdot e^{-Ts} \tag{4.7}$$

5. Control por computador

El control en tiempo continuo realizado mediante un montaje analógico permite trabajar sobre las señales de una forma casi instantánea, pero es un tipo de implementación difícilmente adaptable a cambios en las condiciones de trabajo o en el proceso a controlar.

En los últimos años se ha incrementado el uso de sistemas digitales y, por tanto, del control en tiempo discreto, principalmente por su flexibilidad y por la capacidad para realizar toma de decisiones.

5.1. Control discreto

Un sistema discreto es aquel que, no es continuo ni en magnitud ni en tiempo. Es decir, solo existe valor para sus señales en diversos instantes de tiempo. Normalmente, el tiempo entre muestras sigue una ley periódica regular de forma que existen muestras para el instante inicial y para aquellos instantes que difieran del primero en un número entero de lo que se conoce como periodo de muestreo (tiempo entre muestras).

$$\exists\ f(t)\ \forall t = t_0 + K \cdot T_M;\ K \in \mathbb{N}$$

Un lazo de control por computador puede considerarse como un sistema híbrido que combina una parte continua (el sistema real) y una discreta (el algoritmo de control).

5.1.1. La transformada en Z

La transformada de Laplace permitía el paso de las ecuaciones diferenciales a ecuaciones algebraicas para su fácil resolución. Ahora, de forma análoga, para la resolución de las ecuaciones en diferencias que modelan los sistemas en tiempo discreto se introduce la transformada en Z cuya definición es:

$$\mathcal{Z}[\{y(k)\}] \triangleq Y(z) = \sum_{k=0}^{\infty} y(k) z^{-k} \tag{5.1}$$

De donde se obtiene la ecuación:

$$Y(z) = y_0 + y_1 z^{-1} + y_2 z^{-2} + \ldots + y_k z^{-k} \tag{5.2}$$

Así, al aparecer la variable z en una expresión elevada a un exponente negativo se puede interpretar como procedente de una secuencia retardada tantos intervalos como el exponente indique. Por este motivo, se suele llamar operador retardo a la expresión z^{-1}. De igual forma, si el exponente es positivo indicará que se trata de un término adelantado [4].

Las propiedades más importantes de la transformada en $\mathcal{Z}$ son las siguientes:

- Carácter lineal de la transformación:

$$\mathcal{Z}\{af_1(k) \pm bf_2(k)\} = a\mathcal{Z}\{f_1(k)\} \pm b\mathcal{Z}\{f_2(k)\} \; para \; todo \; a, b \in \mathbb{R} \tag{5.3}$$

- Traslación temporal:

$$\mathcal{Z}\{f(k-N)\} = z^{-N}F(z) \tag{5.4}$$

$$\mathcal{Z}\{f(k+N)\} = z^{N}\left[F(z) - \sum_{k=0}^{N-1} f(kT)z^{-k}\right] \tag{5.5}$$

  siendo $F(z) = \mathcal{Z}\{f(k)\}$

- Teorema del valor inicial:

$$\lim_{k\to 0} f(k) = \lim_{z\to\infty} F(z) \tag{5.6}$$

- Teorema del valor final:

$$\lim_{k\to\infty} f(k) = \lim_{z\to 1}(1 - z^{-1})F(z) \tag{5.7}$$

5.1.2. Representación de sistemas discretos

En el caso continuo, se podía expresar la salida como una función temporal. De forma parecida, se puede expresar la salida como una función discreta cuyo valor en cada instante se calcula a partir del valor de la entrada en el dicho instante y los valores de entrada y salida en los instantes anteriores.

$$y_k = f(u_k, u_{k-1}, u_{k-2}, \ldots, u_{k-m}, y_{k-1}, y_{k-2}, \ldots, y_{k-n}) \tag{5.8}$$

Esta función, para sistemas lineales e invariantes con el tiempo se puede expresar como:

$$\{y_k\} = \sum_{j=o}^{m} b_j\{u_{k-j} + \sum_{i=o}^{n} a_j\{y_{k-i}\} \tag{5.9}$$

Aplicando el operador retardo (z^{-d}) a la ecuación (5.9) se obtiene la expresión de la secuencia temporal en el dominio de las $\mathcal{Z}$. Una vez obtenida dicha expresión, el paso a la función de transferencia discreta es inmmediato.

$$Y(z) = b_0U(z) + b_1z^{-1}U(z) + \ldots + b_mz^{-m}U(z) + a_1z^{-1}Y(z) + \ldots + a_nz^{-n}Y(z) \tag{5.10}$$

$$G(z) = \frac{Y(z)}{U(z)} = \frac{b_0 + b_1z^{-1} + b_2z^{-2} + \ldots + b_mz^{-m}}{1 - (a_1z^{-1} + a_2z^{-2} + \ldots a_nz^{-n})} \tag{5.11}$$

A partir de la expresión de la función de transferencia del sistema se pueden definir los siguientes términos:

- **Orden del sistema:** Es el grado del denominador de la función de transferencia, es decir: Orden(G) = n
- **Polos del sistema:** Son las raíces del polinomio que forma el denominador de la función de transferencia.
- **Ceros del sistema:** Son las raíces del polinomio que forma el numerador de la función de transferencia.

5.2. Control por computador

Ante el problema de controlar un sistema continuo mediante un computador, es evidente la necesidad de adquirir dichas señales para que el procesador opere sobre ellas y genere unas acciones de control en la salida que sirvan como entrada a dicho sistema continuo. Sin embargo, un computador no puede funcionar de forma continua puesto que trabaja realizando diversas tareas de forma secuencial y empleando para ello un tiempo que denominaremos tiempo de ciclo del programa.

Dada la imposibilidad de trabajar de forma continua será imprescindible adquirir la señal de entrada pero de forma discreta. Para este propósito se empleará un convertidor analógico/digital encargado de obtener el valor de la señal para cada instante de tiempo, esperando entre la toma de dos valores consecutivos un tiempo constante y conocido (periodo de muestreo). De este modo, la señal sobre la cuál se va a trabajar ya no será continua sino una secuencia de valores.

Del mismo modo, en la salida se dispondrá de una secuencia de valores de acción de control a partir de los cuales se deberá construir una señal continua que alimente al proceso continuo. En este caso, el encargado de realizar esta tarea será el convertidor digital/analógico que en función de la estrategia de reconstrucción que se desee emplear generará una señal continua u otra. La forma más habitual de reconstrucción es la conocida como ZOH (*Zero Order Hold*). Este retenedor de primer orden se encargará de mantener constante el valor del último instante de tiempo hasta la llegada del siguiente quedando una señal como la de la figura 5.1.

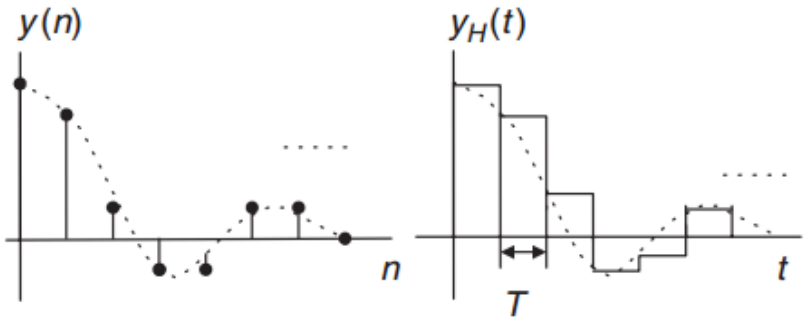


Figura 5.1: Señal reconstruida mediante un *Zero Order Hold*

Con estas premisas, el esquema básico de un sistema controlado mediante un computador quedará tal y como muestra la figura 5.2 donde la salida es continua pero tanto las referencias como las acciones de control calculadas son variables discretas.

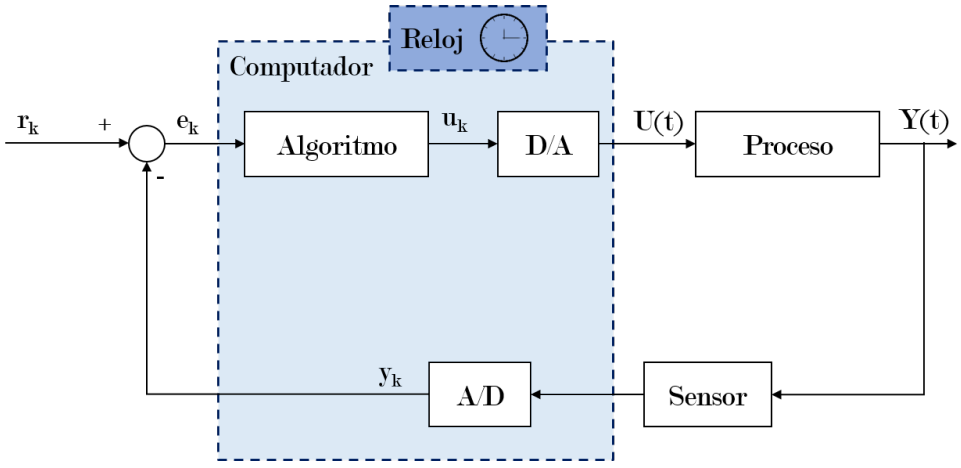


Figura 5.2: Diagrama de bloques típico de un lazo de control por computador

En el proceso de realizar una conversión analógico-digital intervienen principalmente dos funciones. La primera, la cuantificación, es el proceso de transformación de una señal continua o analógica en un conjunto de estados discretos. Está directamente relacionada con el número de bits o tamaño de palabra con el que se va a representar dicha magnitud, siendo posible para una variable digital de n bits representar 2^n estados discretos.

Se denomina resolución de un sistema al mínimo cambio en la entrada que produce un cambio apreciable en la salida. En el caso de un convertidor analógico, esta se corresponde con el valor del bit menos significativo (***L**ess-**S**ignificative-**B**it*) y es el resultado de dividir el rango máximo de la señal de entrada entre el número de estados posibles o bits de la palabra digital.

$$\textbf{LSB} = \frac{Rango\ V_{Ent}}{2^n} \tag{5.12}$$

Por este motivo, el proceso de cuantificar digitalmente una señal analógica implica necesariamente la asunción de un error relacionado directamente con la resolución. Dicho error variará entre 0 y $\pm\frac{LSB}{2}$ y por tanto podrá hacerse tan pequeño como se desee aumentando el número de bits empleados para representar el valor de dicha señal. Sin embargo, no podrá anularse puesto que en la práctica existe un número máximo de bits y por tanto siempre existirá una incertidumbre conocida como ruido de cuantificación[5].

La segunda función implicada en la conversión analógico-digital de una señal continua es la codificación y consiste en la asignación de una palabra o código digital a cada uno de los estados discretos cuantificados.

5.2.1. Teorema del muestreo

Un factor importante para realizar un control en tiempo discreto es determinar el tiempo de muestreo adecuado para cada caso concreto. Para poder hacer la selección correcta será necesario tener en cuenta dos factores.

En primer lugar, deberá cumplirse el teorema de Shannon-Nyquist que intenta establecer el límite de frecuencia de muestreo que permite asegurar para qué valores de la misma se puede reconstruir perfectamente la señal. Este dice que para una señal limitada en banda de frecuencia, la frecuencia mínima de muestreo debe ser al menos el doble que la frecuencia máxima de la señal.

$$\omega_s \geq 2 \cdot \omega_{max} \tag{5.13}$$

Esta condición viene impuesta por las características frecuenciales del sistema. El muestreo de una señal se traduce en una serie de repeticiones de la banda base centradas en los múltiplos enteros de la frecuencia de muestreo ω_s. Tal y como se muestra en la figura 5.3 si se cumple con la condición del teorema de muestreo (ecuación 5.13) no se producirá solapamiento entre las diferentes bandas de frecuencias y por tanto con un filtro paso bajo ideal se podría reconstruir perfectamente la señal original.

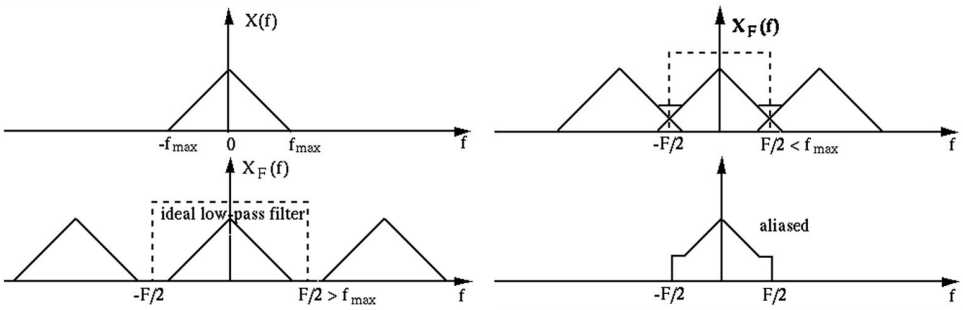


Figura 5.3: Teorema del muestreo

En el caso de darse dicho solapamiento, conocido como *aliasing*, no existirán suficientes muestras para reconstruir la señal y la salida podrá ser tanto una señal totalmente distinta de la esperada como una señal que aparentemente coincide con la esperada pero puede tener diferente amplitud, frecuencia, etc.

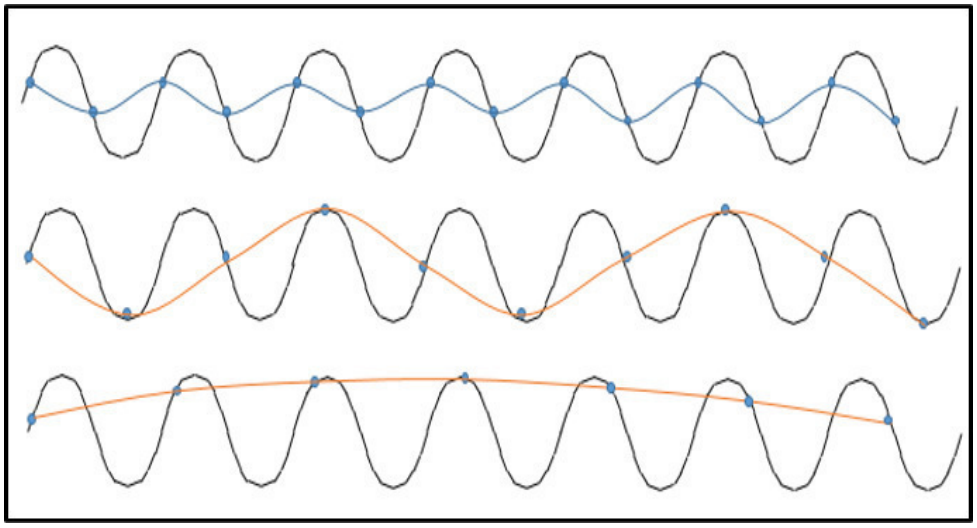


Figura 5.4: Ejemplo de aliasing en el muestreo de una señal

La segunda condición viene impuesta por el hecho de realizar el control mediante un computador. El periodo de muestreo mínimo debe ser mayor que el tiempo que tarda el computador en realizar un ciclo completo de programa para poder cumplir con el tiempo real.

5.3. Sistemas de control basados en red

Las redes de comunicación son muy utilizadas en el ámbito del control para transferir datos puesto que permite centralizar la toma de decisiones. Tradicionalmente, se han empleado protocolos de comunicación propios como *CAN* o *PROFIBUS* pero, por su elevado coste y poca flexibilidad, están dejando paso en la actualidad a protocolos más extendidos como ethernet, bluetooth o radiofrecuencia.

Los sistemas de control que incluyen redes de comunicación se pueden clasificar en sistemas de control complejos y sistemas de control remoto.

Los **sistemas de control complejos** están compuestos por una estructura de subsistemas conectados a un controlador. La conexión física de los elementos en la red es de por si complicada por el hecho de que se trata con estructuras de gran complejidad, cosa que empeora cuanto mayor es el número de subsistemas y la distancia física entre los elementos a conectar.

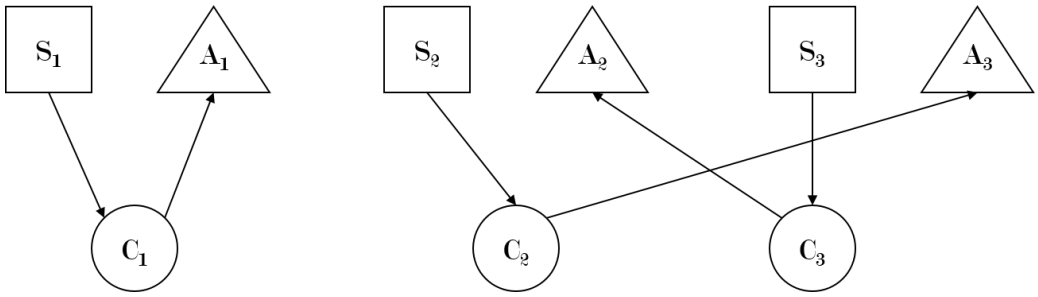


Figura 5.5: Conexión directa

Habitualmente, para simplificar las conexiones se suele optar por implementar una red de comunicación compartida entre los diferentes subsistemas. De este modo, además de facilitar la instalación y su posterior mantenimiento, se abre la posibilidad a transmitir información entre los diferentes bucles de control.

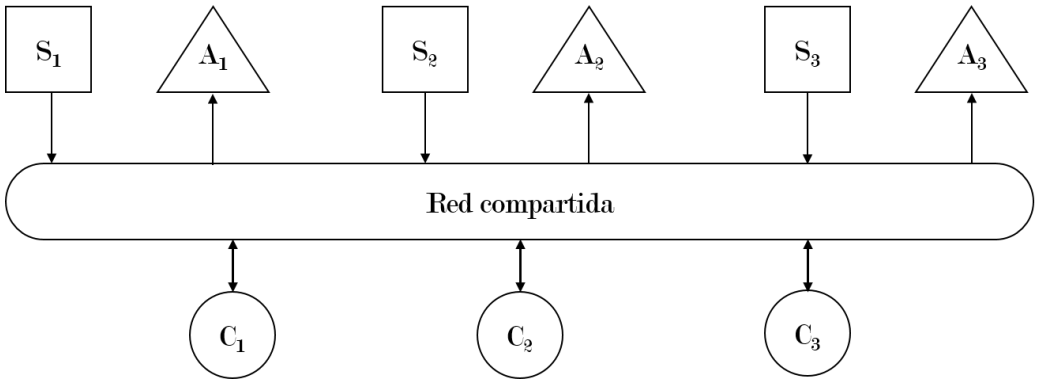


Figura 5.6: Conexión mediante red compartida

Por otro lado, los **sistemas de control remoto** son cada vez más empleados en el ámbito industrial gracias a la comodidad y seguridad que ofrece a los operarios que pueden actuar sobre los sistemas sin necesidad de desplazarse al lugar en cuestión. La principal novedad en estos sistemas es que combinan elementos locales y remotos. Los elementos locales son aquellos que se encuentran físicamente donde se va a realizar el proceso industrial y normalmente son el proceso y el actuador. El controlador es un elemento que puede estar tanto en local como en remoto según el tipo de planteamiento de control que se emplee.

El control en red se puede plantear de dos formas. En primer lugar, mediante una estructura jerárquica dónde cada subsistema incluye su propio controlador pero esta subordinado a un controlador central que establece el punto de funcionamiento que cada subsistema ha de satisfacer [6].

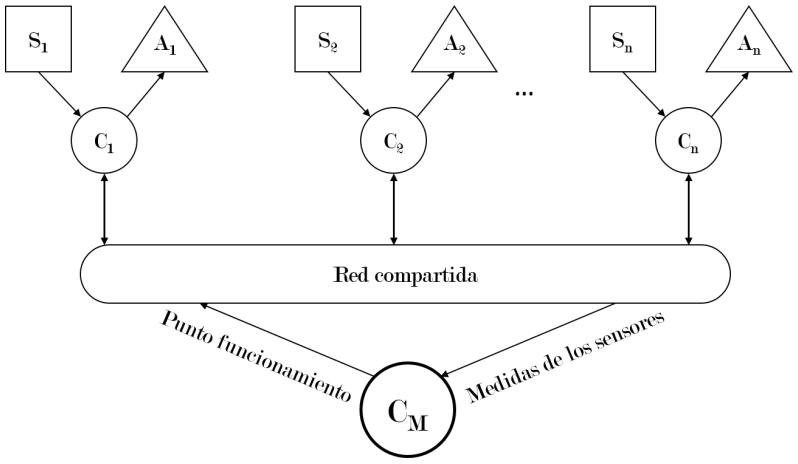


Figura 5.7: Conexión en red mediante estructura jerárquica

La otra forma de realizar el control el red es conocido como estructura directa. En este caso, los sensores y actuadores adscritos a cada planta se encuentran en local mientras que solo hay un controlado central en remoto.

En este caso, los pasos que debe seguir el controlador central para cada uno de los subsistemas son los siguientes:

1. Lectura de los sensores desde la red.
2. Cálculo de las acciones de control.
3. Envío de las acciones de control para los actuadores a través de la red.

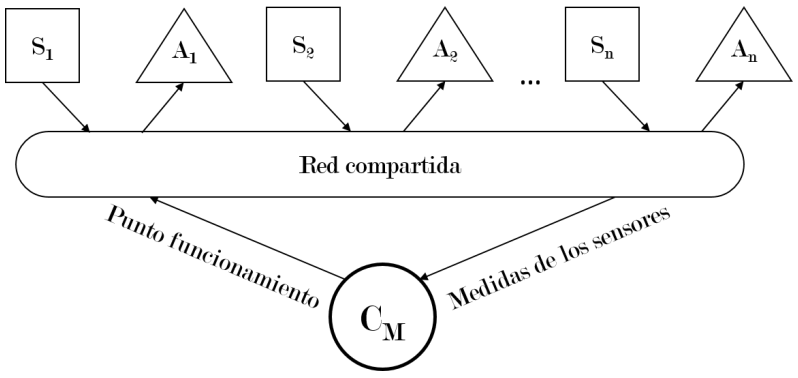


Figura 5.8: Conexión en red mediante estructura directa

5.3.1. Retrasos

Uno de los problemas más habituales en los sistemas de control en red son los retrasos de transporte que junto a la pérdida de datos durante la transmisión pueden llegar a desestabilizar el sistema. Dichos retardos, descritos en la figura 5.9 se pueden clasificar básicamente en tres tipos.

En primer lugar, el retardo introducido por la red desde que el sensor toma una medida hasta que esta le llega al controlador. Este retardo se denomina retardo sensor-controlador ($\tau_{\mathbf{sc}}$).

El siguiente tipo de retraso es el conocido como retardo computacional ($\tau_{\mathbf{c}}$) y es aquel relacionado con el tiempo que emplea el controlador para realizar los cálculos pertinentes y obtener las acciones de control que se deben aplicar a los actuadores.

Por último, el retardo controlador-actuador ($\tau_{\mathbf{ca}}$) es el tiempo que transcurre entre el instante en que el controlador envía las acciones de control hasta el momento en que son aplicadas por los actuadores. Este retardo, al igual que el primero, está causado directamente por el uso de una red.

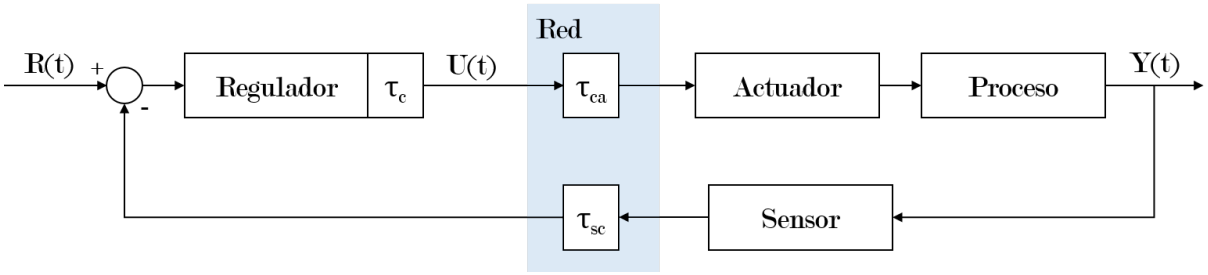


Figura 5.9: Retardos habituales en un lazo de control basado en red

De este modo el retardo total del sistema será la suma de los tres retardos arriba mencionados y será importante conocer su valor para poder realizar la implementación del predictor de Smith (apartado 4.2).

El retardo computacional es variable e inevitable pero puede ser reducido variando la arquitectura del controlador y sobretodo la eficiencia del código empleado. Sin embargo, los retardos sensor-controlador y controlador-actuador, como ya se ha destacado, están directamente relacionados con el protocolo de red empleado y no son fácilmente reducibles. Además cabe destacar que, habitualmente, la magnitud del retardo computacional es mucho menor que las de los retardos generados por la red por lo que el primero puede no tenerse en cuenta.

De no tenerse en cuenta el efecto de los retardos, el comportamiento del lazo de control puede verse comprometido siendo lo más habitual un empeoramiento del transitorio (aparición de grandes oscilaciones y estabilización más lenta) siendo posible llegar a casos en los que el sistema pueda volverse inestable.

6. Control multifrecuencial

6.1. Introducción

Un sistema multifrecuencia es aquel que contiene diversas variables muestreadas y actualizadas a diferentes frecuencias. No obstante, para simplificar el tratamiento, se puede asumir que el muestreo es periódico. Es decir, que aunque cada variable sea muestreada y/o actualizada a una frecuencia diferente, existe un periodo global T_0 para el cual el funcionamiento del sistema es cíclico.

En los sistemas de control básicos, frecuentemente, se suele asumir un patrón de muestreo perfectamente uniforme y regular. Sin embargo, en las aplicaciones prácticas no tiene porque ser así pudiendo no ser un patrón síncrono de forma natural o directamente modificado voluntariamente para mejorar la respuesta del sistema.

Por sus características, los sistemas multifrecuencia pueden ser útiles en un gran rango de aplicaciones industriales donde el tiempo de muestreo quede restringido por las condiciones del lazo de control (retardos de transporte, tiempos de ciclo del controlador digital...) a un valor mayor del establecido por el teorema del tiempo de muestreo y por tanto impida cumplir con el tiempo real o realizar un buen control mediante las técnicas de control convencionales. Así la finalidad de este tipo de lazos de control será obtener un comportamiento similar al de los controladores clásicos en aquellas situaciones donde estos no funcionen correctamente.

En resumen, un sistema multifrecuencial constará esencialmente de una parte rápida y otra parte lenta. La parte lenta trabajará con el tiempo de muestreo con el que se tomen las medidas de los sensores (NT) mientras que la parte rápida del controlador será la encargada de aplicar las acciones de control N veces más rápido que se realizan las lecturas (T). La resolución de un problema con técnicas multifrecuenciales tiene dos posibles planteamientos. El primero de ellos, fija el periodo de muestreo con el que se va a hacer la lectura de datos (NT) y varía la frecuencia con la que se van a aplicar las acciones de control hasta encontrar una respuesta válida.

Por el otro lado, el método que se va a emplear en este proyecto, trata de fijar el tiempo de muestreo (T) con el que se aplican las acciones de control y se va disminuyendo el número de muestras tomadas para ver la respuesta del sistema. Este método permite, para un sistema con un tiempo de muestreo adecuado para aplicación de acciones de control, disminuir la frecuencia con la que se toman muestras sin perder demasiadas prestaciones mientras que el primer caso permite conseguir un buen control para un periodo de muestreo de lectura de sensores malo. [7]

6.2. Base teórica

Como se ha mencionado en la introducción de este apartado, los sistemas multifrecuencia son aquellos donde existen señales muestreadas a dos o más frecuencias diferentes. La figura 6.1 pretende facilitar la aproximación al concepto de sistema multifrecuencia y a la notación que se va a emplear.

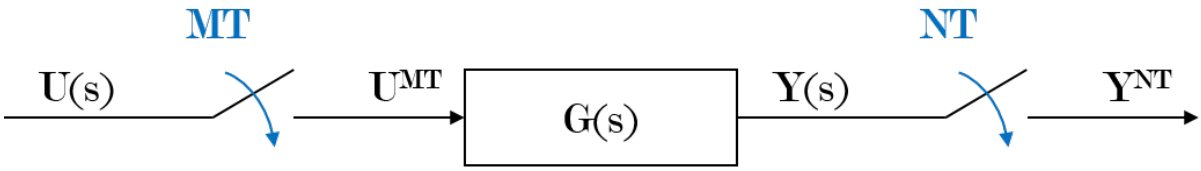


Figura 6.1: Esquema inicial de un sistema multifrecuencia

La notación empleada para la descripción de los diferentes sistemas y señales, tal y como muestra la figura, indicará en forma de superíndice el periodo de muestreo que se ha empleado en la discretización. De este modo, la señal Y^T indicaría que proviene de la transformada en Z de la secuencia y(kT) proviniente del muestreo de la señal continua y(t) con un periodo T.

$$\mathcal{Z}[\{y(k)\}] \triangleq Y(z) = \sum_{k=0}^{\infty} y(k) z^{-k} \tag{6.1}$$

Del mismo modo, observando la figura 6.1, se puede ver que se produce la obtención de una señal discreta (Y) muestreada a NT a partir de otra señal discreta (U) muestreada a MT.

$$Y^{NT} = [G(s)U^{MT}]^{NT} = G^{NT}[U^{MT}]^{NT} \tag{6.2}$$

donde G^{NT} representa la función de transferencia del proceso continuo discretizada con un periodo NT normalmente mediante un retenedor de orden cero (ZOH):

$$G^{NT} = \mathcal{Z}\left[\frac{1 - e^{-NTs}}{s} G(s)\right] \tag{6.3}$$

Este caso sencillo, sirve para mostrar que en los problemas con sistemas multifrecuencia, las transformaciones de frecuencia de muestreo para adecuar las señales a los diferentes bloques (elementos del sistema) son muy habituales. Una forma sencilla de trabajar con estos sistemas es emplear el máximo común divisor (mcd) y el mínimo común múltiplo (mcm) de los diferentes periodos de muestreo implicados. De este modo, cada periodo de muestreo va a ser repetido un número entero de veces en un tiempo resultado de realizar el mcm de los periodos de muestreo del sistema (T_0). De igual forma, existirá un periodo base (mcd) T_B tal que $T_0 = P \cdot T_B$ siendo P un entero mayor que la unidad [8].

Para poder realizar las transformaciones de las señales para los diferentes tiempos de muestreo es necesario introducir dos operadores básicos:

Por un lado, el operador "expand" sirve para crear una secuencia muestreada a T a partir de una secuencia muestreada a NT. Realiza esta operación rellenando con 0 los valores de la secuencia para los instantes de tiempo para los cuales no se dispone de información.

$$[Y^{NT}(z_N)]^{NT} \triangleq \overline{Y}^T(z^N) \triangleq \sum_{k=0}^{\infty} \overline{y}(kT) z^{-kN} \begin{cases} \overline{Y}^T = y(kT); \forall k = \lambda N \\ \overline{Y}^T = 0; \forall k \neq \lambda N \end{cases} \lambda \in \mathcal{Z}^+ \tag{6.4}$$

La siguiente figura muestra de forma gráfica el funcionamiento de este operador.

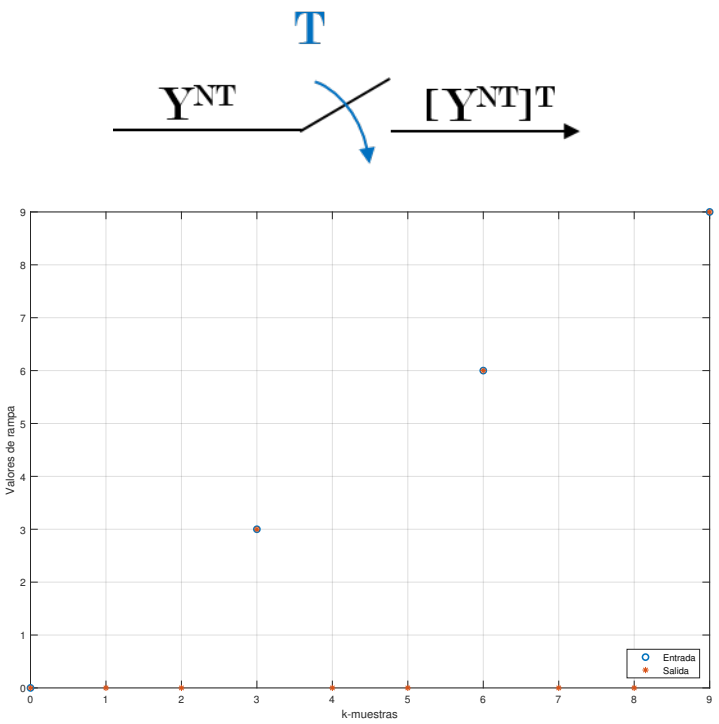


Figura 6.2: Operador expand para el caso N=3

Por el otro lado, el operador "*skip*" hace justo el proceso inverso. Permite crear una secuencia muestrada a NT a partir de una muestreada a T. Para ello simplemente elimina aquellas muestras que no coincidan con los instantes de muestreo a NT.

$$[Y^{NT}(z_N)]^{NT} \triangleq \breve{Y}^T(z^N) \triangleq \sum_{k=0}^{\infty} \overline{y}(kT) z^{-kN} = Y^{NT}(z_N) \tag{6.5}$$

De forma gráfica, el funcionamiento de este operador es el siguiente:

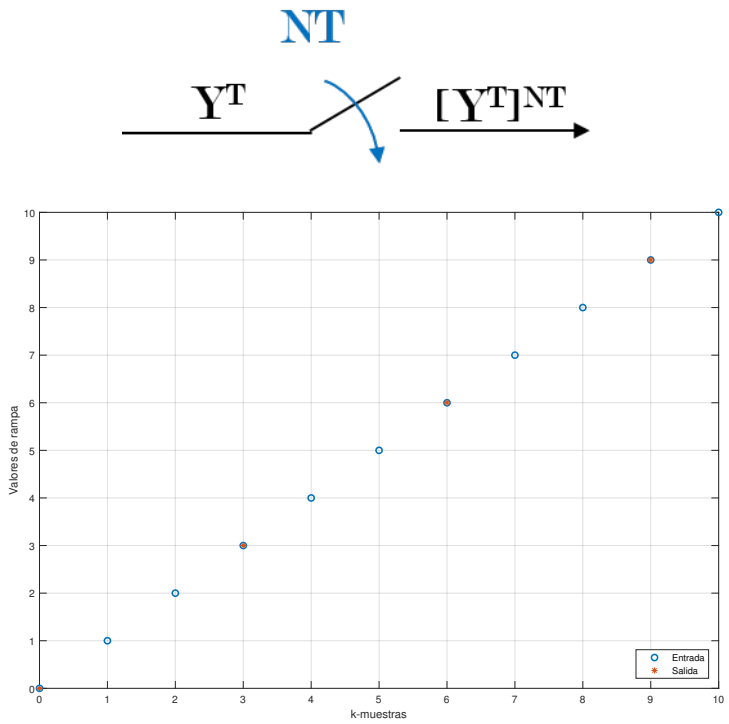


Figura 6.3: Operador expand para el caso N=3

Para simplificar notación a partir de este momento se indicará con z aquellas secuencias referidas a T y con z^N aquellas referidas a NT.

Partiendo del ejemplo de la figura 6.1, es posible obtener una función de transferencia discreta para cualquier par de señales entrada-salida. De este modo, si analizamos la parte rápida del proceso (parte muestreada a T) obtenemos el modelo discreto de muestreo rápido (FSDT):

$$\frac{Y^T(z)}{U^T(z)} = G^T(z) = [H_T G_p(s)]^T = \frac{B^T(z)}{A^T(z)} = \frac{\sum_{i=1}^{n} b_{i,T}\ z^{-i}}{1 + \sum_{i=1}^{n} a_{i,T}\ z^{-i}} \tag{6.6}$$

donde $B^T(z), A^T(z)$ son polinomios en zetas negativas y representan secuencias de n elementos siendo $a_{0,T} = 1$ y $b_{0,T} = 0$. Cabe destacar que los operadores explicados anteriormente (*expand* y *skip*) también son aplicables en zetas con índice negativo.

Los polos de la función de transferencia rápida (FSDT) se expresan como $\alpha_{i,T}$.

$$A^T(z) = \prod_{i=1}^{n}(z - \alpha_{i,T}) \tag{6.7}$$

De forma análoga se puede obtener la función de transferencia correspondiente a la parte lenta del sistema (SSDT), es decir, de la parte muestreada a NT.

$$\frac{Y^{NT}(z^N)}{U^{NT}(z^N)} = G^{NT}(z^N) = [H_{NT}G_p(s)]^{NT} = \frac{B^{NT}(z_N)}{A^{NT}(z_N)} = \frac{\sum_{i=1}^{n} b_{i,NT}\ z^{-iN}}{1 + \sum_{i=1}^{n} a_{i,NT}\ z^{-iN}} \tag{6.8}$$

donde $B^{NT}(z_N), A^{NT}(z_N)$ son polinomios en z_N^{-1}.

En este caso los polos se expresarán como $\alpha_{i,NT}$ siendo

$$A^{NT}(z^N) = \prod_{i=1}^{n}(z^N - \alpha_{i,NT}) \tag{6.9}$$

Para continuar con el desarrollo teórico será necesario tomar dos asunciones técnicas.

a. Si α es un polo de $G^T(z)$, entonces $\alpha e^{2\pi kj/N}, j = 1 \ldots (N-1)$ no lo es.

   Esta condición es necesaria para evitar el *aliasing*.

b. Todos los polos son diferentes.

   Esta condición permite simplificar la notación.

Llegados a este punto, se puede obtener fácilmente la siguiente expresión:

$$W_A^T(z) = \frac{\prod_{i=1}^{n}(z^N - \alpha_{i,NT})}{\prod_{i=1}^{n}(z - \alpha_{i,T})} = \frac{[A^{NT}(z^N)]^T}{A^T(z)} = \frac{\overline{A}^T(z^N)}{A^T(z)} = \prod_{i=1}^{n} z^{N-1} + \alpha_{i,T}z^{N-2} + \ldots + \alpha_{i,T}^{N-1} \tag{6.10}$$

El modelo correspondiente a la parte rápida también puede ser expresado como:

$$\frac{Y^T(z)}{U^T(z)} = G^T(z) = \frac{B^T(z)}{A^T(z)} = \frac{B^T(z)W_A^T(z)}{A^T(z)W_A^T(z)} = \frac{\widetilde{B}^T(z)}{[A^{NT}(z^N)]^T} \tag{6.11}$$

donde $G^T(z) \equiv \mathcal{Z}_{\mathcal{T}}[H_T(s)G_P(s)]$ siendo H_T un retenedor de orden cero (ZOH).

A partir de esta última ecuación es inmediato obtener la siguiente expresión:

$$\widetilde{B}^T U^T = [A^{NT}]^T Y^T \tag{6.12}$$

Si se aplica el operador *skip* a una secuencia muestreada a T, se obtiene una secuencia remuestreada a NT de la siguiente forma:

$$\left[\widetilde{B}^T U^T\right]^{NT} = \left[[A^{NT}]^T Y^T\right]^{NT};\ \left[\widetilde{B}^T U^T\right]^{NT} = \left[[A^{NT}]^T\right]^{NT}\left[Y^T\right]^{NT} \tag{6.13}$$

$$\left[\widetilde{\mathbf{B}}^{\mathbf{T}}\mathbf{U}^{\mathbf{T}}\right]^{\mathbf{NT}} = \left[A^{NT}\right]\left[Y^T\right]^{NT} = \mathbf{A}^{\mathbf{NT}}\mathbf{Y}^{\mathbf{NT}}; \tag{6.14}$$

En la expresión (6.14) no es posible aislar los elementos del término $[\widetilde{B}^T U^T]^{NT}$ por las características no conmutativas del operador *skip*. Por este motivo, no es viable una función de transferencia que relacione una entrada rápida y una salida lenta. Sin embargo, existe la posibilidad de expresar una señal rápida como la suma de N señales lentas, separar los términos y resolver el problema.

Por otro lado, el proceso inverso sí es factible. Es posible obtener una salida rápida a partir de una entrada lenta empleando un retenedor de orden cero pero bifrecuencial ($DRZOH$), definido como:

$$[H_{NT}(s)]^T = \frac{U_O^T}{[U^{NT}]^T} = [\frac{1-e^{-NTs}}{s}]^T = \frac{1-z^{-N}}{1-z^{-1}} = [W_R^{-1}]^T \tag{6.15}$$

donde la salida queda definida como U_O^T.

De este modo, es posible obtener la función de transferencia del proceso junto al retenedor bifrecuencia:

$$Y^T = [H_{NT}G_P(s)U_{NT}]^T = [H_{NT}G_P(s)]^T[U^{NT}]^T \tag{6.16}$$

A partir de esta expresión y lo obtenido en la ecuación (6.15) se obtiene que

$$[H_{NT}G_P(s)]^T = [W_R^{-1}]^T[H_T G_P(s)]^T \tag{6.17}$$

Por lo que un operador bifrecuencia en tiempo discreto (DRDT) se define como:

$$G^{T,NT} \equiv \frac{Y^T}{[U^{NT}]^T} = [W_R^{-1}]^T G^T = [W_R^{-1}]^T \frac{B^T}{A^T} = [W_R^{-1}]^T \frac{B^T W_A^T}{A^T W_A^T} = \frac{[W_R^{-1}]^T \widetilde{B}^T}{[A^{NT}]^T} \tag{6.18}$$

donde $G^{T,NT}$ describe la función de transferencia de una salida rápida muestreada a T respecto una entrada lenta muestreada a NT. Con la misma notación, el operador $DRZOH$ se puede expresar como $H^{T,NT}$. De este modo, se verifica que:

$$\left[\frac{Y^T}{[U^{NT}]^T}\right]^{NT} = \frac{[Y^T]^{NT}}{[[U^{NT}]^T]^{NT}} = \frac{[Y^T]^{NT}}{U^{NT}} = \frac{\left[[W_R^{-1}]^T \widetilde{B}^T\right]^{NT}}{[[A^{NT}]^T]^{NT}} = \frac{B^{NT}}{A^{NT}} \tag{6.19}$$

Al afrontar el problema de modelar un sistema *multi-rate*, generalmente se emplea un método conocido como *discrete-lifting*. Mediante este procedimiento, cada señal estará referida al mínimo común múltiplo de los periodos del sistemas multifrecuencia.

En una aplicación general de multifrecuencia donde se asuma mínimo común múltiplo (caso de *lifting*), una señal Y^T será modelada del siguiente modo:

$$Y_l(z^N) = \begin{bmatrix} y_{l,0}(z^N) \\ y_{l,1}(z^N) \\ \vdots \\ y_{l,N-1}(z^N) \end{bmatrix} = \begin{bmatrix} y_0 + y_N z^{-N} + \dots \\ y_1 + y_{N+1} z^{-N} + \dots \\ \vdots \\ y_{N-1} + y_{2N-1} z^{-N} + \dots \end{bmatrix} \tag{6.20}$$

Es decir, cada secuencia $y_l, i(z^N)$ se obtiene de expandir los N elementos previamente eliminados por el operador *skip* de la señal original muestreada a T. Así es posible expresar dicha señal como:

$$Y(z) = (1 \quad z^{-1} \quad z^{-2} \quad \dots \quad z^{-(N-1)}) \cdot Y_l(z^N) \tag{6.21}$$

ya que

$$y(z) = \sum_{k=0}^{\infty} y_k z^{-k} = (y_0 + y_N z^{-N} + \dots) + z^{-1}(y_1 + y_{N+1} z^{-N} + \dots) + \dots + z^{-(N-1)}(y_{N-1} + y_{2N-1} z^{-N} + \dots) \tag{6.22}$$

6.2.1. Exposición del problema

Como se ha mencionado en el apartado 6.1, la estrategia de control multifrecuencia cobra sentido en problemas de control donde la implementación de un control monofrecuencia no sea viable por restricciones en el tiempo de muestreo mínimo, normalmente impuestas por los sensores o las redes compartidas de comunicación.

Así, la idea escogida para solucionar el problema es alcanzar una respuesta similar a la que se conseguiría con un control *single-rate* a T pero tomando medidas N veces más lento. En términos generales, se necesita un controlador que con una entrada lenta genere una salida rápida. La figura 6.4 muestra un esquema general de un controlador bifrecuencial (T,NT).

El funcionamiento del controlador en este lazo es procesar el error a frecuencia lenta E^{NT} y generar N acciones de control rápidas (U^T) dentro de cada metaperiodo (NT). Solo se ha considerado los casos en que la relación entre los periodos de muestreo de entrada y salida es entera.

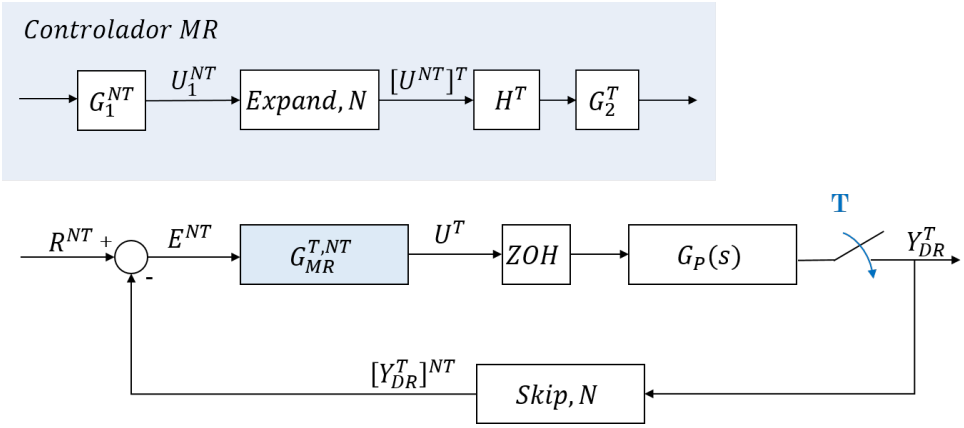


Figura 6.4: Bucle de control típico Bifrecuencia

A partir de la figura, aplicando las propiedades de los operadores *skip-expand* se puede obtener la expresión de la salida del controlador.

$$U^T = G_2^T[U_1^{NT}]^T = G_2^T[G_1^{NT}]^T[E^{NT}]^T = G_2^T[G_1^{NT}]^T\left[R^N T - [y_{DR}^T]^{NT}\right]^T \tag{6.23}$$

y por tanto:

$$y_{DR}^T = G_P^T H^T G_2^T[G_1^{NT}]^T\left[R^N T - [y_{DR}^T]^{NT}\right]^T \tag{6.24}$$

Pero como $y_{DR}^T \neq \left[[y_{DR}^T]^{NT}\right]^T$ no se puede expresar la salida de forma cerrada. Sin embargo, si el modelado se considera a frecuencia lenta si es posible obtener esta expresión.

$$\begin{aligned}[y_{DR}^T]^{NT} &= \left[G_P^T H^T G_2^T[G_1^{NT}]^T\left[R^N T - [y_{DR}^T]^{NT}\right]^T\right]^{NT} = \\ &= \left[G_P^T H^T G_2^T\right]^{NT} G_1^{NT}\left[R^{NT} - [y_{DR}^T]^N T\right]\end{aligned} \tag{6.25}$$

$$\frac{[y_{DR}^T]^{NT}}{R^{NT}} = \frac{\left[G_P^T H^T G_2^T\right]^{NT} G_1^{NT}}{1 + \left[G_P^T H^T G_2^T\right]^{NT} G_1^{NT}} \tag{6.26}$$

Y finalmente:

$$\frac{y_{DR}^T}{[R^{NT}]^T} = \frac{G_P^T H^T G_2^T[G_1^{NT}]^T}{1 + G_P^T H^T G_2^T[G_1^{NT}]^T} \tag{6.27}$$

Una vez obtenido el modelo, es necesario afrontar el diseño del controlador. El objetivo es conseguir que la salida rápida ($[y_{DR}^T]^{NT}$) se comporte como la salida del bucle de control monofrecuencia ($\overline{Y}^T$). Es decir, $\left[M^T R^T\right]$ debe coincidir con $M^{NT} R^{NT}$.

Así, considerando M(s) como modelo de referencia para el proceso y M^{NT} y M^T sus equivalentes discretizados a NT y T respectivamente mediante un retenedor de orden cero, el controlador bifrecuencia quedará del siguiente modo:

- Parte rápida del controlador:

$$G_2^T(z) = \frac{M^T(z)}{G_P^T(z)} \tag{6.28}$$

- Parte lenta del controlador:

$$G_1^{NT}(z) = \frac{1}{1 - M^{NT}(z_N)} \tag{6.29}$$

- Conversor de frecuencia:

$$H^T(z) = \frac{R^T(z)}{[R^{NT}]^T(z)} \tag{6.30}$$

Si el proceso es de fase no mínima, la cancelación de polos y ceros inestables debe evitarse puesto que una mala cancelación debida a errores de modelado puede llevar al sistema a la inestabilidad. En estos caso, la parte rápida del sistema puede calcularse como:

$$G_2^T(z) = \frac{M^T(z)}{G^T(z)} = \frac{G_R^T(z)}{1 + G_R^T(z)G^T(z)} \tag{6.31}$$

Esta forma de calcular la parte rápida del controlador, además, permite eliminar el rizado en la salida.

6.2.2. Procesos con retardos temporales

En el diseño de los controladores multifrecuencia no se ha tenido en cuenta la posible existencia de retardos. En el caso de la presencia de estos, una solución es realizar una adaptación del predictor de Smith (introducido en el apartado 4.2) a este tipo de sistemas.

En un lazo de control bifrecuencia (T y NT) el retardo podrá expresarse de forma genérica como $d = L \times NT + J \times T + m$ donde L y J son las multiplicidades integrales del retardo respecto a cada periodo de muestreo.

Optar por una adaptación del predictor de Smith en este tipo de sistemas (figura 6.5) se traduce en aplicar las técnicas clásicas con la novedad del álgebra que se acaba de introducir en los apartados anteriores.

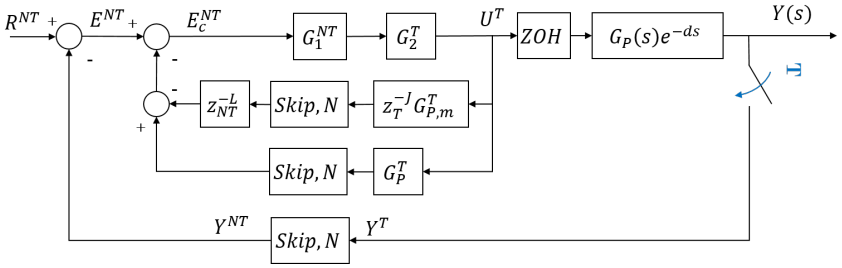


Figura 6.5: Esquema de un bucle bifrecuencia con predictor de Smith

7. Radiofrecuencia

Como ya se ha ido mencionando a lo largo del trabajo, el objetivo del proyecto es realizar un control multifrecuencia mediante una comunicación por radiofrecuencia. Se ha escogido este protocolo por las ventajas que ofrece frente a otros tipos de comunicación similares como el Bluetooth®. La radiocomunicación, permite transmitir a gran distancia, ya sea en espacios interiores como exteriores. Además, a diferencia de otros protocolos que se encuentran activos continuamente para mantener la conexión, solo tiene un consumo significativo cuando se esta realizando la transmisión o recepción de datos cosa que, para sistemas móviles alimentados mediante baterías, supone una gran ventaja puesto que permite aumentar su autonomía. Para comprender bien como funciona la comunicación mediante radiofrecuencia será necesario introducir varios conceptos.

En primer lugar, se entienden por telecomunicaciones aquellas técnicas mediante las cuales se transfiere información desde un punto inicial a otro remoto. La principal clasificación de estas se realiza en función del canal o medio por donde se transmiten los datos. De este moto se pueden distinguir comunicaciones guiadas como la fibra óptica y las comunicaciones no guiadas como la radiocomunicación y, más concretamente, la radiofrecuencia.

Se habla de radiocomunicación si el envío se realiza mediante ondas radioeléctricas. Es decir, se emplean campos electromagnéticos para realizar los envíos a la mayor velocidad posible. En este tipo de sistemas, la transmisión de la información se realiza superponiendo la información que se desea enviar a una onda electromagnética que se denomina portadora mediante un proceso conocido como modulación.

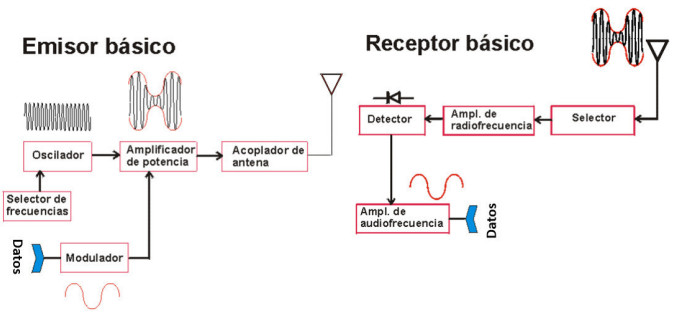


Figura 7.1: Esquema básico radiocomunicación

La modulación funciona de tal modo que realiza una serie de modificaciones en los parámetros (amplitud y fase) de la onda portadora, que posteriormente serán detectadas por el receptor que a partir de estas reconstruirá la información transmitida. Dichas modificaciones convertirán la señal portadora (un tono) en una señal resultado de la superposición de una serie de tonos. Así, se definirá el ancho de banda de la señal como la porción del espectro electromagnético que ocupa ese conjunto de tonos. Cuanto más información se desee transmitir mayor será el ancho de banda que ocupará dicha señal y por tanto, menor el número de comunicaciones simultáneas que se puedan establecer[9].

Para realizar una buena radiocomunicación es importante diseñar las modificaciones de la onda portadora para que ocupen el mínimo espectro posible y conocer las pérdidas y distorsiones que se pueden introducir en el sistemas como consecuencia de la frecuencia portadora escogida y el medio donde se va a propagar la señal. Además, los receptores deben ser selectivos, esto es, ser capaces de seleccionar un ancho de banda concreto para recibir exclusivamente las señales de interés.

La radiofrecuencia es un tipo de radiocomunicación que se caracteriza por trabajar con señales comprendidas entre 3Hz y 300GHz. Esto es, la parte menos energética del espectro electromagnético.

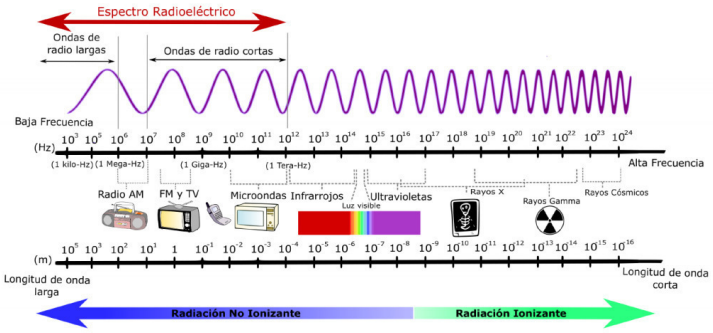


Figura 7.2: Espectro electromagnético

En función de la frecuencia empleada para la comunicación se puede realizar una clasificación de las ondas de radiofrecuencia en bandas tal y como muestra la figura 7.3. En este caso, se emitirá a 434 MHz por lo que se trabajará con señales UHF (*Ultra-High-Frecuency*).

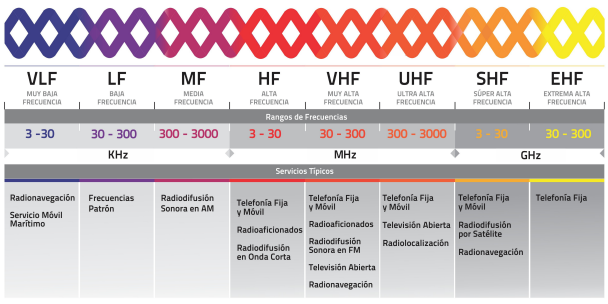


Figura 7.3: Bandas en función de frecuencia y longitud de onda

En función del número de canales y frecuencias que emplea la red para trabajar existen diferentes configuraciones.

- **Red Simplex (1 Canal y 1 Frecuencia):**

  - Se da cuando la frecuencia de recepción (Rx) y la de transmisión (Tx) es la misma.

  - Permite comunicación entre todos los equipos siempre que estén en la zona de cobertura.

- **Red Simplex (1 Canal y 2 Frecuencias):**

  - Permite comunicar la base con el resto de los equipos que no podrán comunicarse entre ellos.

- **Redes con repetidores:**

  - Se emplean para tener mayor alcance. Esto es, cobertura a mayor distancia. Hay diferentes configuraciones:

    Modo Duplex

    Modo Semiduplex

    Modo Semiduplex radioenlazado

    Modo doble cruzado

En este caso, se empleará una red *half-duplex* con 1 canal y 1 frecuencia para realizar la comunicación. Para ello se dispondrá de un par de antenas de radiofrecuencia que permitirán establecer una comunicación bidireccional. Más adelante (apartado 8.3), se tratará de describir más detalladamente las antenas que se han empleado, cómo se ha realizado la conexión y qué parámetros se han empleado para su configuración.

Parte III

Herramientas

8. Robot móvil de bajo coste

Si bien el objetivo del proyecto inicialmente era realizar un control multifrecuencia sobre un sistema real, se ha considerado que la mejor forma de probar el correcto funcionamiento era realizar el control de un robot móvil de forma que siga determinadas trayectorias por diversos motivos.

En la actualidad, han cobrado gran importancia los sistemas de navegación autónoma donde un robot móvil de desplaza libremente, evitando obstáculos en caso de haberlos, para llegar a determinados puntos y realizar la actividad que proceda.

En este caso, se ha implementado un control en remoto mediante una red de comunicación por radiofrecuencia para introducir en el sistema retrasos de transporte que impidan realizar el control mediante las técnicas clásicas, siendo necesario el uso de técnicas multifrecuenciales.

Aunque existen diferentes dispositivos comerciales perfectamente aplicables al proyecto, se ha decidido desarrollar una plataforma de laboratorio específica para realizar el estudio con la idea de conseguir un dispositivo que se ajuste mejor a las necesidades del proyecto. Por este motivo, se ha diseñado un robot móvil diferencial. A lo largo de los siguientes apartados, se ampliará información sobre las diferentes partes del robot móvil que se ha desarrollado para validar el control implementado.

8.1. Arduino DUE

La parte más importante del sistema es el computador que se va a emplear para controlar el sistema móvil. Por la gran comunidad de usuarios que lo emplean, pero sobretodo por su fácil programación se ha decidido emplear una *board* de Arduino para el proyecto.

En este caso se ha empleado una placa de desarrollo Arduino Due. Esta dispone de un microcontrolador de 32-bit Atmel SAM3X8E ARM Cortex-M3. Además tiene 54 entradas/salidas digitales (12 de las cuales se puede emplear como salidas PWM), 12 entradas analógicas, 4 puertos serie, 2 convertidores digital-analógicos y conexión USB.

Figura 8.1: Placa de desarrollo Arduino Due

Si bien esta placa de desarrollo está pensada para proyectos más complejos, ha sido la escogida porque ofrece la posibilidad de emplear la plataforma de laboratorio para futuros proyectos sin necesidad de desmontar los elementos actuales gracias a la gran disponibilidad de pines y puertos de comunicación.

8.2. Motores

Para el accionamiento de las dos ruedas se ha decidido emplear dos motores MG37D de la marca Pololu. Estos motores, se alimentan con 12V de corriente continua e integran una reductora 70:1. Además disponen de un encoder con una resolución de 64 cuentas por vuelta que, teniendo en cuenta la caja de cambios, se traduce en 4480 cuentas por revolución. Es decir, se dispone de una resolución de $^{360}/_{4480} = 0.08^{\text{o}}$[10].

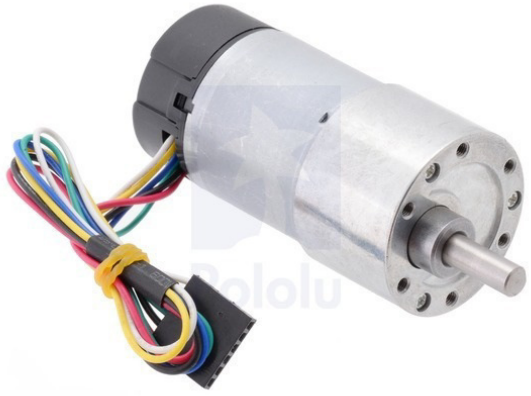


Figura 8.2: Motor MG37D de Pololu ®

La tensión nominal del motor es de 12V, sin embargo, este es capaz de trabajar con tensiones tanto menores como mayores. En el caso de ser menores, la velocidad de giro cada vez será menor. Por el contrario, si se le aplican tensiones superiores a la nominal, el motor trabajará en malas condiciones y en consecuencia reducirá su vida útil.

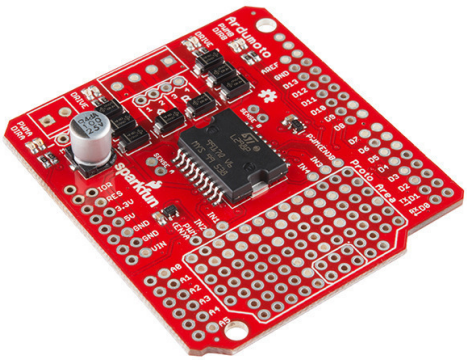


Figura 8.3: Motor Driver Shield de SparkFun Ardumoto

Si bien la tensión no es un factor determinante, la corriente sí lo es. Para poder aplicar una corriente constante se ha empleado un driver para arduino. Concretamente se ha empleado el *Motor Driver Shield* fabricado por SparkFun porque sirve para controlar dos motores y ofrece una corriente de 2A [11]. Además, se alimenta con la misma tensión que la placa arduino lo que facilita el montaje. Este módulo permite controlar fácilmente tanto la velocidad como la dirección de giro de ambos motores.

Como el fabricante no dispone de ruedas para ejes de 6mm como el del motor empleado, se ha optado por comprar ruedas para ejes de 3mm y un casquillo para anclarlas al eje del motor.

Figura 8.4: Rueda eje para 3mm y casquillo adaptador

8.3. Módulo Radiofrecuencia

Para la comunicación mediante radiofrecuencia se ha empleado el módulo APC220 de APPcon technologies por ser una solución de bajo consumo y fácil integración en el proyecto. Además, posee capacidad para transmitir datos con un alcance de hasta 1 km.

Figura 8.5: Módulo APC220 de APPcon Technologies

En el dispositivo móvil, este módulo se conecta al puerto serie de la placa Arduino. Así, para realizar los envíos y recepciones de datos tan solo es necesario escribir o leer dicho puerto serie.

En el ordenador, el módulo se conecta mediante un adaptador USB y en este caso, para leer y escribir es necesario trabajar sobre el puerto COMx que le asigne el PC.

Las principales características de este módulo de radiofrecuencia, extraídas de sus hojas de características [12], son:

- **Tensión de alimentación:** 3.3 - 5 V
- **Corriente:** 25 - 35 mA
- **Interfaz:** UART/TTL
- **Baud rate:** 1200 - 19200 bps
- **Buffer de recepción:** 256 bytes
- **Frecuencia de trabajo1:** 431 - 478 MHz

Para poder establecer la comunicación entre dos antenas, es necesario configurar sus parámetros de la forma adecuada. El propio fabricante ofrece un software (*RF-Magic*) que permite realizar dicha configuración de una forma muy sencilla.

El primer parámetro a configurar, tal y como muestra la figura 8.6, será la frecuencia de la señal portadora que se empleará para transferir los datos, siendo la escogida 434 MHz.

A continuación, se configurará el *baud-rate* y la potencia de transmisión. Es decir, la cantidad de información transmitida por segundo y la energía que se aportara para que la información llegue a más o menos distancia. En este caso, se ha escogido un *baud-rate* de 19200 bps y una potencia de 9 puesto que son los valores máximos que admite el módulo de radiofrecuencia.

Por último, será necesario configurar los identificadores de la red (*NET ID* y *NODE ID*). Estos valores en principio son libres, siempre que se configuren las antenas con el mismo *NET ID* y diferente *NODE ID*.

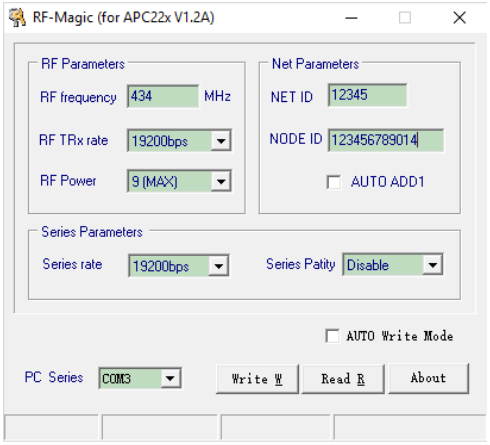


Figura 8.6: Software de configuración del módulo APC220

8.4. Bateria

Para la aplicación se ha elegido una batería de 3 celdas Litio-Polímero del fabricante Yuntong. Se ha escogido esta batería, de 11.1 V de tensión nominal, porque su capacidad es de 2600 mAh lo que nos permite tener una gran autonomía ya que el consumo de nuestro sistema completo (placa arduino, antena de radiofrecuencia y motores) es bastante bajo.

Figura 8.7: Bateria Li-Po 11.1V 2600mAh 15c

8.5. Diseño de la carcasa

La carcasa ha sido diseñada en 3D mediante el software *Inventor* de Autodesk ®. Se ha empleado este programa para el diseño porque tiene una interfaz bastante intuitiva que hace del diseño un proceso relativamente sencillo.

El carcasa consta de tres piezas diseñadas por separado que servirán de soporte para todos los elementos del sistema (batería, motores, antena...).

La primera pieza, será la carcasa propiamente dicha. En ella se encuentran los orificios para sujetar los bastidores de los motores (*L-Bracket Pair for 37D for MG*) y la antena. También tiene los orificios por donde saldrán los ejes de los motores. Como se ha diseñado un robot diferencial, con el fin de estabilizar el robot y así mantener su horizontalidad se han incluido en el diseño dos ruedas locas (*Pololu ball caster with 3/4"metal ball*).

Figura 8.8: Carcasa

La segunda pieza, figura 8.9, ha sido diseñada para soportar la batería. Esta, esta diseñada para encajar con la pieza 3 de tal forma que la batería no pueda moverse. Además, se encajará sobre la carcasa para evitar el posible desplazamiento en el interior de esta.

Figura 8.9: Soporte batería

Por último, la figura 8.11, muestra la pieza que se encargará de sujetar, mediante tres tornillos, la placa Arduino.

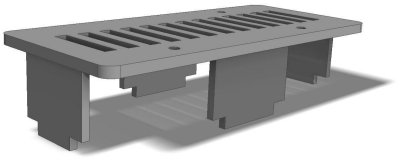


Figura 8.10: Soporte placa Arduino Due

Las tres piezas han sido impresas en material plástico ABS con la impresora 3D del instituto de automática e informática industrial (ai2) de la Universitat Politécnica de València. El resultado final es el siguiente:

Figura 8.11: Ensamblaje del robot

9. CompactRIO

El compactRIO es un controlador embebido de National Instruments que combina un procesador que ejecuta NI Linux Real-Time, un FPGA programable y diferentes módulos de entrada y salida, tanto analógicas como digitales. Se caracteriza por su robustez y confiabilidad, características certificadas con los estándares industriales.

Figura 9.1: CompactRIO de National Instruments

Su programación se realiza de forma sencilla a través del software LabVIEW desarrollado por el mismo fabricante, lo que le permite interactuar de forma remota a través de la red con otros elementos de la misma.

Su utilidad para este proyecto reside en que al incorporar módulos de entrada/salida analógicos permite registrar con alta precisión valores de intensidad y tensión, almacenarlos en un fichero y enviarlos a través de la red para su análisis. En este caso para realizar la medida de corriente y tensión se han empleado los módulos NI 9227 y NI 9201 respectivamente.

Figura 9.2: Módulos de medida de corriente y tensión CompactRIO

10. Entorno de programación

En la realización del proyecto, hay tareas como la programación del microcontrolador que tan solo se pueden hacer mediante el uso de programas informáticos. Por contra, el diseño de los reguladores o la construcción de la carcasa podría realizarse perfectamente sin ayuda de ningún *software*. Sin embargo, se ha optado por realizar dichas tareas mediante diferentes programas específicos por varios motivos.

La principal razón es la flexibilidad y versatilidad que ofrecen las técnicas CAD (*Computer Aided Design* y CAM (*Computer Aided Manufacturing*). El empleo de estas técnicas permite exportar el trabajo realizado a otras plataformas, la fácil resolución de errores de cálculo o incluso la simulación del correcto funcionamiento del sistema diseñado.

Concretamente en los sistemas de control, las técnicas CADCS (*Computer Aided Desing of Control Systems*) ofrecen la posibilidad de expresar los conceptos técnicos o realizar los cálculos necesarios de forma bastante intuitiva mediante su interfaz gráfica de usuario. Además permiten obtener gráficas, ajustes, animaciones y simulaciones que permitan analizar el funcionamiento del sistema facilitando así la rectificación de los posibles errores.

Los diferentes *software* que se han empleado para la ejecución del proyecto son:

- **NI LabVIEW 2016 32-bits**:

  Se ha empleado este entorno de programación para implementar la parte del lazo de control que se realiza en el ordenador. Esta se corresponderá con el cálculo de las referencias de las ruedas a partir de las lecturas anteriores y la trayectoria que se desee seguir. Se ha escogido el software de National Instruments sobre todo por su versatilidad.

  LabVIEW, a diferencia de otros entornos, emplea un lenguaje de programación totalmente gráfico dónde el orden en que se ejecutan las diferentes tareas viene determinado por las lineas de flujo de programa. Ser un entorno gráfico no solo resulta interesante por simplificar el análisis y diseño de los programas, sino que su gran utilidad es la posibilidad de exportar ese código a diferentes plataformas sin necesidad de emplear los lenguajes propios de dicho dispositivo.

- **Arduino IDE 1.8.5 :**

  Se ha empleado para la programación de la placa arduino encargada de realizar la comunicación con el ordenador y el control de las ruedas del robot móvil. Gracias a la gran comunidad Arduino, existen gran número de librerías que facilitan la configuración y uso de los diferentes *shields* o placas de expansión.

- **Matlab 2017b:**

  Se ha empleado para realizar las tareas de identificación de la planta de los motores, el cálculo de los reguladores y su simulación y el análisis de los datos obtenidos. Gracias a su *toolbox* Simulink ® este software es una herramienta perfecta para el análisis y simulación de sistemas de control.

- **Autodesk Inventor:**

  Se ha empleado para el diseño de la carcasa y las piezas de soporte del robot móvil. Se ha optado por realizar el diseño mediante una herramienta CAD porque además de simplicar la tarea de diseñar, permite realizar pruebas de ensamblaje para eliminar posibles errores. Además ofrece la posibilidad de, o bien obtener fácilmente los bocetos 2D del objeto para su fabricación manual o exportar el diseño para su posterior impresión en 3D.

Parte IV

Desarrollo práctico

11. Caracterización del motor

El primer paso para realización del control de velocidad de los motores es obtener el modelo mátematico de estos sobre el cual se trabajará para establecer la estructura de control más adecuada y diseñar el regulador que permita cumplir con las especificaciones de diseño.

11.1. Curva característica entrada - salida

En primer lugar, se ha decidido obtener la relación entre la entrada y la salida del sistema. Para ello, se ha empleado como referencia una señal triangular (ecuación (11.1)) que barre todos los posibles valores de acción de control (desde $-100\,\%$ hasta $+100\,\%$). Esta comprobación nos permite caracterizar de forma cualitativa el comportamiento dinámico del sistema.

$$\left.\begin{aligned} y = -10 + 0.02x \quad &, \; si \; 0 < x \leq 1000 \\ y = 10 - 0.02(x - 1000) \; &, \; si \; 1000 < x \leq 2000 \end{aligned}\right\} \tag{11.1}$$

donde ± 10 se corresponde con el $\pm 100\,\%$ de la acción de control máxima y 0.02 es la pendiente con la que se ha deseado que se aumente la acción de control de forma que las variaciones sean suficientemente lentas para que el motor responda ante dichos cambios.

Si el sistema tuviera un comportamiento lineal, la salida ante esta entrada sería otra triangular con diferente pendiente siendo el cociente de ambas pendientes la ganancia que introduce el sistema. Sin embargo, observando la salida obtenida (figura 11.1) es evidente que el motor tiene un comportamiento claramente no lineal.

De forma cualitativa, se puede ver que la respuesta del sistema varía en función de la entrada de forma que al aumentar la acción de control introducida por encima del $60\,\%$ - $70\,\%$ la ganancia del sistema disminuye. Además, se observa la presencia de una zona muerta ya que para valores bajos de entrada, la salida es nula.

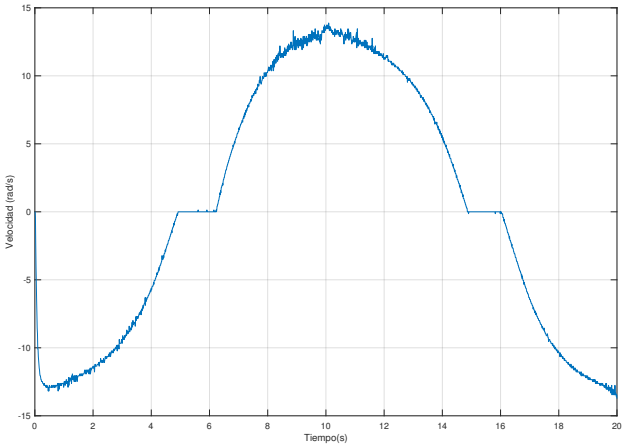


Figura 11.1: Curva característica entrada - salida del motor MG37D

11.1.1. Caracterización zona muerta

Para determinar el valor exacto de la zona muerta se va a someter al sistema, igual que en el apartado anterior, a una entrada de tipo rampa (ecuación (11.1)) donde los valores de entrada variarán entre el -100 % y el 100 % de la acción de control máxima.

Sin embargo, en este caso, en lugar de la variación temporal de la salida se visualizará la señal de salida en función de la señal de entrada (figura 11.2). De este modo, observando para qué valores de entrada la salida es nula, se puede determinar la amplitud concreta de la zona muerta. En el caso de los motores que se van a emplear para el proyecto esta zona muerta va desde -2.4 V hasta 2.44 V, que corresponde con aproximadamente el ± 25 % de la acción de control máxima.

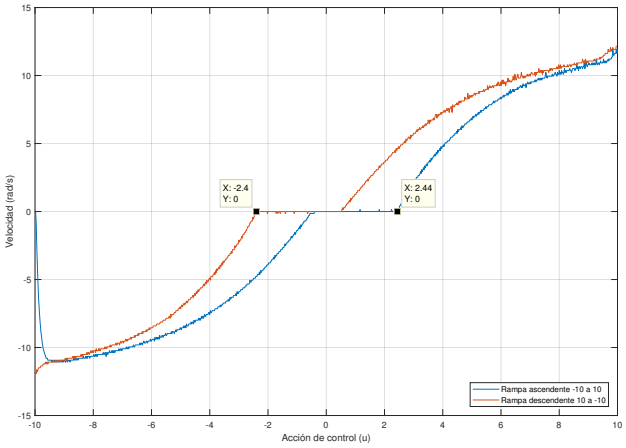


Figura 11.2: Zona muerta del motor MG37D

11.2. Obtención función de transferencia del motor

Para realizar la identificación de la función de transferencia del motor se ha analizado la respuesta del mismo frente a diferentes entradas de tipo escalón. Se ha decidido así porque la magnitud que se va a considerar como salida del sistema es la velocidad angular del motor, y su comportamiento frente a la entrada es modelable mediante un sistema de primer orden (11.2).

$$G_{vel} = \frac{K_{est}}{1 + \tau s} \tag{11.2}$$

Así, el método de caracterización que se ha empleado, tal y como se puede observar en la figura 11.3, consiste en obtener los parámetros característicos de un sistema de primer orden (K y τ) atendiendo a las características de la respuesta de este tipo de sistemas dadas en el apartado 3.3.1.

- $K \rightarrow$ Cociente entre la amplitud de la entrada empleada como referencia y el valor final en que se estabiliza la salida.
- $\tau \rightarrow$ Valor de tiempo (t) cuando la señal de salida alcanza el 63 % de su valor final.

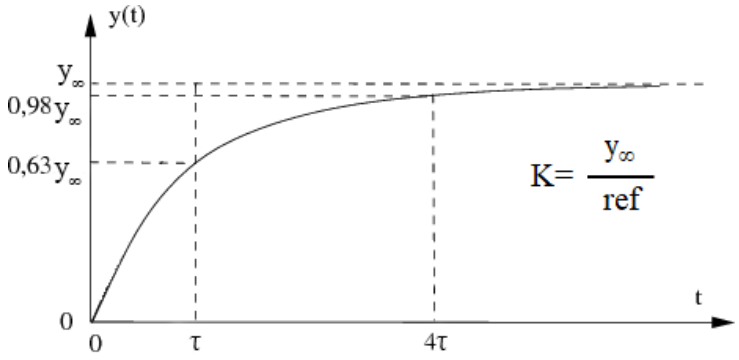


Figura 11.3: Método de caracterización de sistemas de orden 1

Debido a la no linealidad de los motores, se ha decidido realizar la identificación para un rango de limitado de acciones de control. Este rango de valores de referencia se ha situado entre el 30 % y el 50 % de la acción de control máxima aplicable por el *driver* de los motores, porque para valores menores los motores entran en la zona muerta y, para mayores, la velocidad de giro es demasiado elevada para el seguimiento de trayectorias y, además, debido a la no linealidad, la ganancia del sistema es bastante menor. Tal y como se observa en la figura 11.4 se ha realizado dicha prueba para ambos sentidos de giro, aplicando para ello acciones de control positivas y negativas.

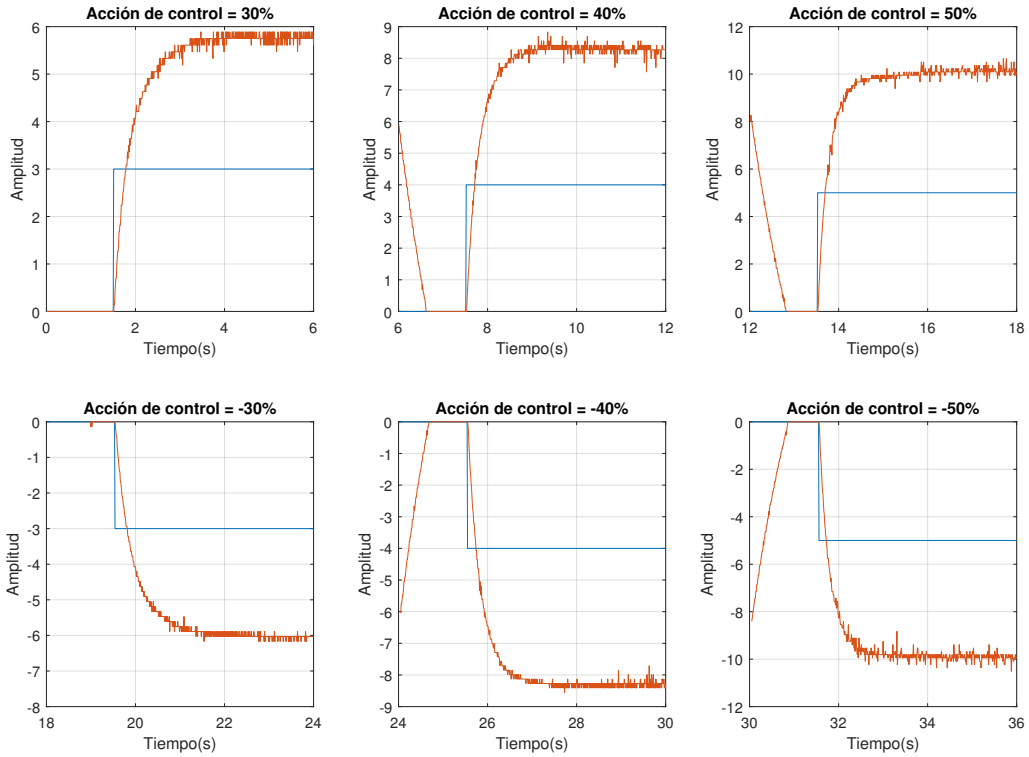


Figura 11.4: Respuesta del motor MG37D frente a diferentes entradas

Una vez obtenidas las señales de salida para cada una de las referencias, se han calculado los valores de ganancia estática y constante de tiempo en cada caso y se ha empleado como valor del modelo la media de ellos tal y como se muestra en las tablas 11.1 y 11.2.

Páram	**U= 30 %**	**U= 40 %**	**U= 50 %**
$\mathbf{K_{est}}$	1.87	2.03	2.07
τ	0.38	0.28	0.27

Páram	**U=-30 %**	**U=-40 %**	**U=-50 %**
$\mathbf{K_{est}}$	2.01	2.07	1.96
τ	0.41	0.32	0.24

Tabla 11.1: Valores obtenidos para los parámetros del modelo

$\mathbf{K_{est}}$	1.995
τ	0.3475

Tabla 11.2: Parámetros del modelo

Una vez obtenidos los valores de los parámetros, tan solo queda sustituirlos en la función de transferencia del sistema (11.2) para obtener el modelo de velocidad del motor (11.3).

$$G_{vel} = \frac{1.995}{1 + 0.3475s} = \frac{5.74}{s + 2.87} \tag{11.3}$$

Aunque en nuestro caso no es necesario, si se desea obtener el modelo de posición, únicamente habrá que añadir un integrador al modelo de velocidad (11.4).

$$G_{pos} = \frac{1.995}{s(1 + 0.3475s)} = \frac{5.74}{s(s + 2.87)} \tag{11.4}$$

Para comprobar que el modelo es válido, se ha sometido tanto el sistema real como el modelo a las mismas entradas y se han comparado las salidas, viéndose en la figura 11.5 que ambas dinámicas son muy similares. Por tanto, el modelo es válido y se puede emplear para realizar el control de los motores.

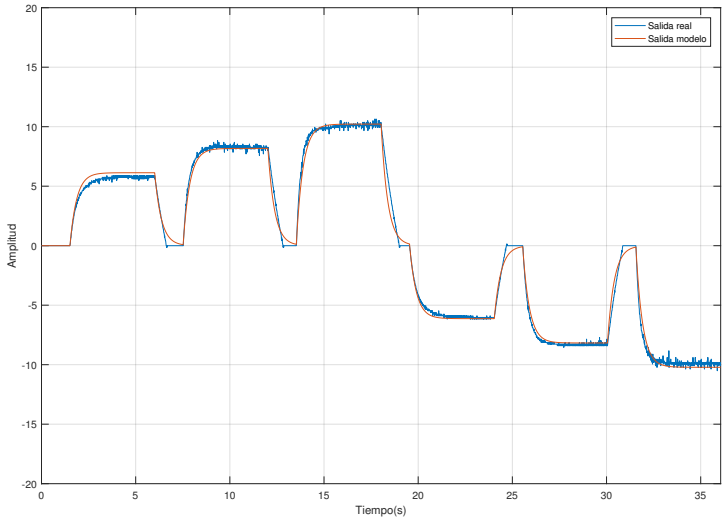


Figura 11.5: Comparación respuesta del sistema real con la del modelo teórico

12. Diseño del regulador

Una vez obtenida la planta del sistema a controlar, el siguiente paso es diseñar el controlador que permita que el sistema cumpla con las especificaciones. Si bien el regulador que se desea implementar es de tipo discreto y bifrecuencia, para obtener este, se ha diseñado previamente un controlador que, en tiempo continuo, permita conseguir la respuesta dinámica esperada.

12.1. Diseño del regulador continuo

Antes de comenzar a diseñar un controlador para el sistema, es necesario conocer las características de la de respuesta del sistema y ver cuáles son válidas y cuáles es necesario modificar. Generalmente, las características que se suelen analizar son el tiempo de establecimiento y la sobreoscilación porcentual para caracterizar el transitorio y el error de posición para caracterizar el régimen permanente.

Para calcular el tiempo de establecimiento se ha empleado el criterio del 98 %, esto es, el tiempo que tarda la salida en alcanzar el 98 % del valor final en el régimen permanente.

$$t_{est} = 4 \cdot \tau = 4 \cdot 0.3475 = 1.39s \tag{12.1}$$

Como se observa en la figura 11.4, para ninguna de las entradas existe ningún valor de la salida que sobrepase su valor final. Es decir, la sobreoscilación del sistema es nula tal y como corresponde con los sistemas de primer orden como este.

$$\delta(\,\%) = 0 \tag{12.2}$$

Por último, queda analizar el error de posición del sistema, es decir, la relación entre el valor final de la señal en el régimen permanente y el valor de la referencia que se desea seguir. Este, se puede calcular a partir de la siguiente expresión:

$$e_p = \lim_{s\to 0} s\ E(s) = \lim_{s\to 0} s \frac{1}{1+G(s)H(s)} \frac{1}{s} = \frac{1}{1+\lim_{s\to 0} G(s)H(s)} = \frac{1}{1+K_P} \tag{12.3}$$

siendo K_P el producto de la ganancia estática del sistema por la del sensor.

$$K_P = \lim_{s \to 0} G(s)H(s) = \frac{5.74}{2.87} \cdot 1 = 1.995 \tag{12.4}$$

y por tanto:

$$e_P = \frac{1}{1 + K_P} = 0.334 \tag{12.5}$$

Es decir, existe un error de posición del 33,4 %.

Analizando los resultados obtenidos, se ha decidido que la respuesta del sistema es demasiado lenta y que además no se consigue alcanzar el valor de referencia con un error de posición bastante elevado. Por este motivo, se ha decidido fijar las especificaciones de diseño en los siguientes valores:

- **Especificaciones dinámicas**

  Sobreoscilación porcentual: $\delta(\,\%) \leq 10\,\%$

  Tiempo de establecimiento: $t_{est} \leq 1s$

- **Especificaciones estáticas**

  Error de posición: $e_p = 0$

Para realizar el diseño del controlador que permita cumplir con las especificaciones se ha utilizado el método del lugar de las raíces, empleando para ello la herramienta *sisotool* de MATLAB. Se ha considerado así debido a que, como se ha visto en el apartado 11, el motor tiene un comportamiento muy no lineal y por tanto el modelo que se ha obtenido no es exacto. Esta inexactitud en el modelo, en el caso de realizar el diseño por cancelación, puede dar lugar a respuestas diferentes de las esperadas e incluso a la inestabilidad.

En primer lugar, se obtendrá el lugar de las raíces del sistema y se marcarán sobre él las zonas que permitan cumplir con las especificaciones (figura 12.1). Como se puede observar, se podría cumplir con las especificaciones dinámicas con un regulador de tipo proporcional.

Sin embargo, en este caso, se ha optado por implementar un regulador de tipo PI puesto que, al ser el modelo de velocidad del motor de tipo 0, es necesario aplicar acción integral para conseguir anular el error en régimen permanente. La acción derivada en este caso, no se ha considerado ya que sin ella es posible cumplir con las especificaciones.

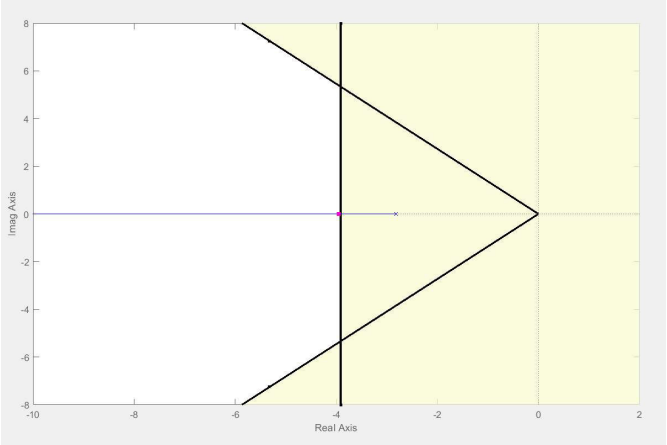


Figura 12.1: Especificaciones de diseño en el lugar de las raíces

El cero del integrador, de forma habitual, se coloca de tal modo que tenga una constante de tiempo 10 veces mayor que la del polo más lento del sistema. Sin embargo, tal y como se observa en la figura 12.2 , esa configuración da lugar en este caso a un transitorio muy brusco al principio pero muy lento en estabilizarse.

$$\sigma_{z_I} = \sigma_p/10 \tag{12.6}$$

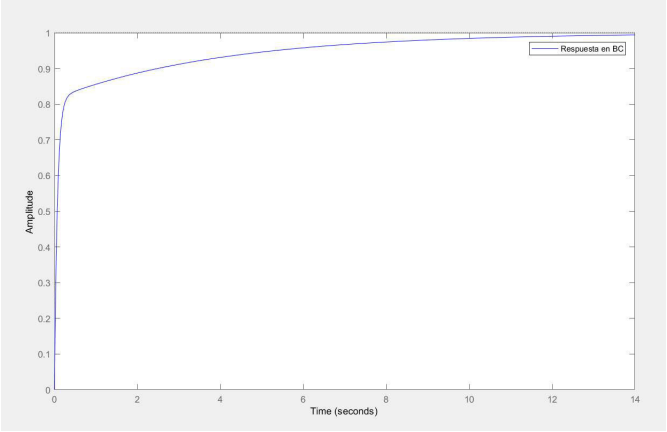


Figura 12.2: Respuesta con el diseño inicial del regulador PI

Por este motivo, se ha decidido desplazar el cero del regulador hacia la izquierda de forma que, en caso de no cancelarse perfectamente con el polo del integrador influya en menor grado en la respuesta del sistema. Además, tal y como se ha observa en la figura 12.3, no se ha escogido como punto de diseño el que marca el límite de la zona de especificaciones. Esto es porque al no cancelarse perfectamente la pareja polo cero del integrador, esta ralentiza la respuesta y da lugar a un tiempo de establecimiento mayor del deseado. Este problema se soluciona fácilmente aumentado la ganancia proporcional.

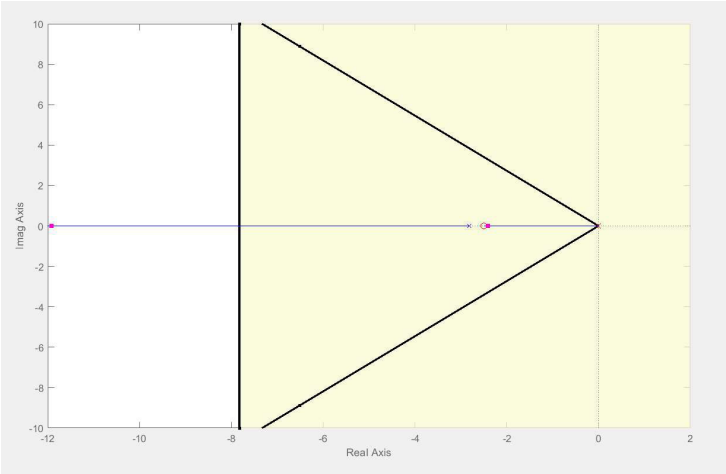


Figura 12.3: Diseño del regulador en el lugar de las raíces

Así, con las condiciones arriba mencionadas, se ha obtenido el siguiente regulador PI cuya respuesta, figura 12.4, cumple con las condiciones de diseño.

$$G_R(s) = \frac{2(s + 2.5)}{s} \tag{12.7}$$

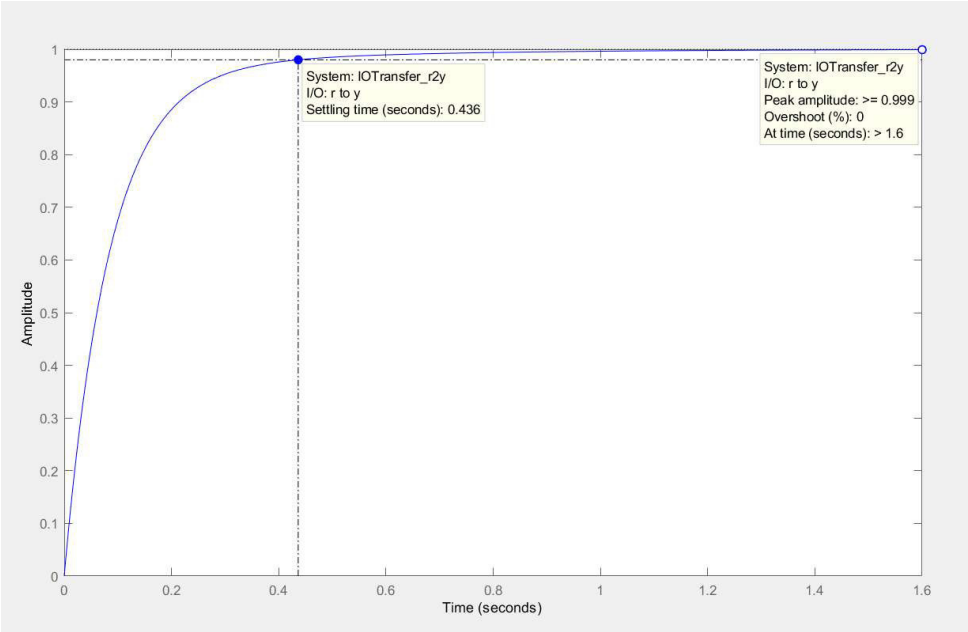


Figura 12.4: Respuesta con el diseño final del regulador PI

12.2. Diseño del regulador discreto multifrecuencia

Una vez realizado el diseño del controlador continuo y determinado su correcto funcionamiento, es necesario adaptar dicho controlador al entorno donde se va a utilizar. En este caso, el control se realizará en un microcontrolador por lo que será necesario realizar la discretización del regulador para obtener las ecuaciones en diferencia con las que trabajar el algoritmo de control digital.

La transformada en Z, como se ha explicado anteriormente, es la herramienta empleada para la discretización de sistemas y reguladores debido a que facilita en gran manera la resolución de las ecuaciones en diferencia. Sin embargo, en realidad no se aplica la transformada en Z a la respuesta del sistema continuo sino a la secuencia que surge del muestrear a T dicha respuesta.

Como se ha destacado en el apartado 5.2.1, para conseguir un buen control debe respetarse el teorema del muestreo que aplicado a este caso, significaría que el tiempo de muestreo escogido debería ser al menos la mitad de la constante de tiempo más lenta del sistema. Es decir:

$$T_m \leq \frac{\tau}{2} \rightarrow T_{m(max)} = \frac{0.3475}{2} = 0.1737 \ s \tag{12.8}$$

De no cumplirse el teorema del muestreo, la respuesta obtenida no cumplirá las especificaciones de diseño empeorando su comportamiento dinámico hasta alcanzar la inestabilidad.

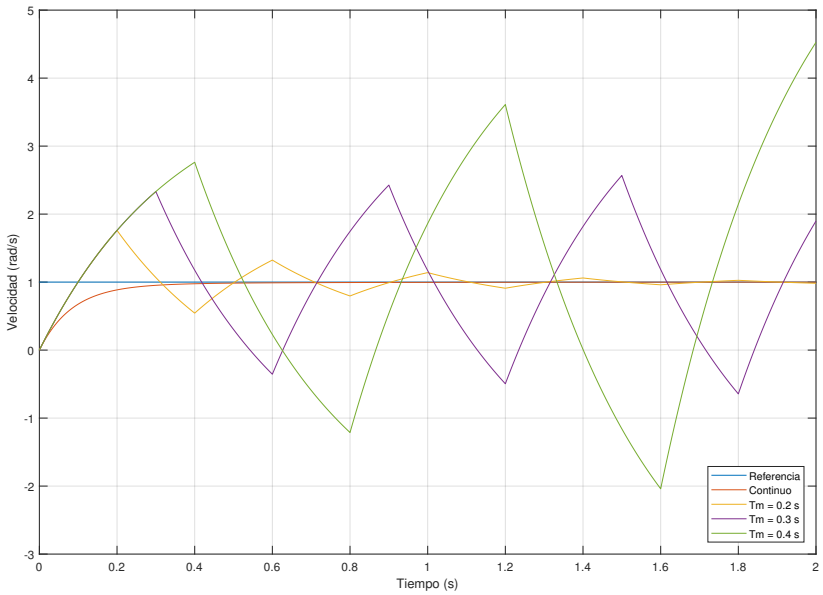


Figura 12.5: Respuesta para el proceso discretizado con periodos de muestreo superiores al máximo

Se ha probado a obtener el regulador discreto para Tm = 150 ms y de este modo poder analizar su respuesta. Para facilitar la discretización del regulador PID lo expresaremos dejando aislados sus parámetros Kc, Td y Ti.

$$G_R(s) = K_c \left(1 + T_d \cdot s + \frac{1}{T_i \cdot s}\right) \tag{12.9}$$

$$G_R(s) = 2\frac{(s+2.5)}{s} = 2\left(1 + 0 \cdot s + \frac{2.5}{s}\right) \tag{12.10}$$

donde $K_c = 2$, $T_d = 0$ y $T_i = 1/2.5$.

Así, a partir de los parámetros del regulador y el tiempo de muestreo escogido (150 ms) se puede obtener rápidamente el controlador discreto correspondiente:

$$G_R(z) = \frac{U(z)}{E(z)} = K_c \frac{1 - \left(1 - \frac{T_m}{T_i}\right) z^{-1}}{1 - z^{-1}} = K_c \frac{z - \left(1 - \frac{T_m}{T_i}\right)}{z - 1} \tag{12.11}$$

$$\mathbf{G_R(z) = 2\left(\frac{z - 0.625}{z - 1}\right)} \tag{12.12}$$

Analizando la respuesta obtenida mediante este regulador, se puede observar que su comportamiento desde el punto de vista de las especificaciones estáticas es bueno puesto que consigue anular el error de posición. Sin embargo, la respuesta dinámica del sistema empeora respecto a la continua por haber empleado un tiempo de muestreo cercano al límite.

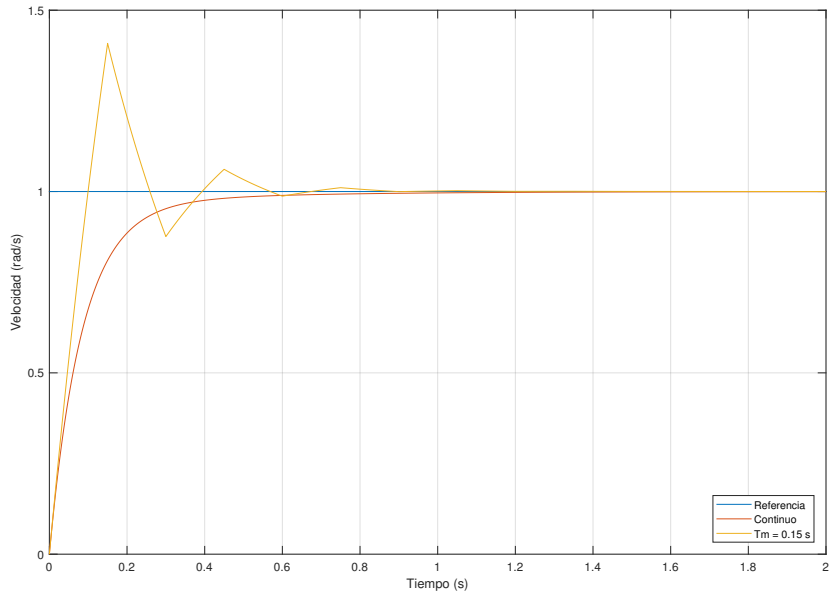


Figura 12.6: Respuesta para el proceso discretizado con Tm = 150 ms

Se ha decidido pues, discretizar el regulador a diferentes periodos de muestreo para ver a partir de que valor se cumplen las especificaciones de diseño y por tanto la respuesta del regulador discreto se puede considerar válida. Es evidente que a menor tiempo de muestreo (mayor número de muestras disponibles) el comportamiento del proceso será más similar al continuo y por tanto se ajustará mejor al diseño original.

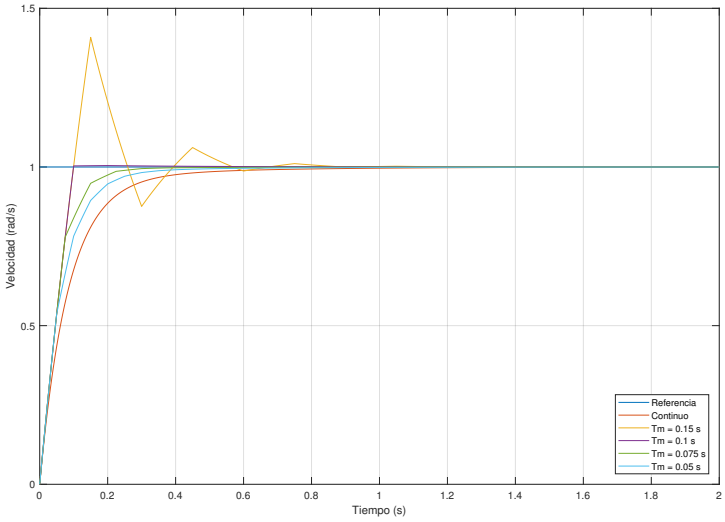


Figura 12.7: Respuesta para el proceso discretizado con diferentes periodos de muestreo

- **Tm =150 ms** $\rightarrow G_R(z) = 2 \cdot \frac{z-0.625}{z-1}$
- **Tm =100 ms** $\rightarrow G_R(z) = 2 \cdot \frac{z-0.75}{z-1}$
- **Tm =75 ms** $\rightarrow G_R(z) = 2 \cdot \frac{z-0.8125}{z-1}$
- **Tm =50 ms** $\rightarrow G_R(z) = 2 \cdot \frac{z-0.875}{z-1}$

Sin embargo, por las características del sistema de control, no será posible disminuir el tiempo de muestreo tanto como se desee. Los retardos de transporte, introducidos en el sistema por la red inalámbrica, aumentan notablemente el tiempo de ciclo del sistema (tiempo en que tarda un dato en dar una vuelta completa) y limitan por tanto el tiempo de muestreo máximo quedando este doblemente condicionado (ecuaciones (12.8) y (12.13))

$$Tm \geq T_{ciclo} \approx 80ms \tag{12.13}$$

Por estos motivos, finalmente se ha decidido escoger un periodo de muestreo de 100 ms puesto que cumple las dos condiciones y su respuesta respeta las especificaciones de diseño. Si bien es verdad que con un periodo de muestreo por encima de 50 ms, aplicando las técnicas de multifrecuencia se podría realizar el control, se ha decidido fijar el tiempo de muestreo a 100 ms para poder realizar comparaciones entre un control realizado mediante las técnicas clásicas y los diferentes controles multifrecuencia diseñados. De este modo, el regulador discreto obtenido será:

$$G_R(z) = \frac{U(z)}{E(z)} = 2 \cdot \frac{z - 0.75}{z - 1} \tag{12.14}$$

siendo su ecuación en diferencias

$$\begin{aligned} U(s) &= U(s)z^{-1} + 2E(z) - 1.5E(z)z^{-1} \\ \mathbf{u(k)} &= \mathbf{u(k-1)} + \mathbf{2e(k)} - \mathbf{1.5e(k-1)} \end{aligned} \tag{12.15}$$

Aunque en el apartado 6.2 ya se ha explicado con detalle cual es la estructura de un controlador multifrecuencia y cómo se debe obtener cada uno de los bloques implicados, se hará un breve recordatorio para facilitar el seguimiento del proceso de diseño.

Cabe destacar que el objetivo del control multifrecuencia es obtener la misma respuesta que se obtendría con un bucle de control muestreado a T pero tomando dichas muestras a NT siendo N = 2,3,4... De este modo, el diagrama de bloques del sistema será similar al de un servosistema pero incluirá señales y sistemas que trabajan a diferentes periodos de muestreo.

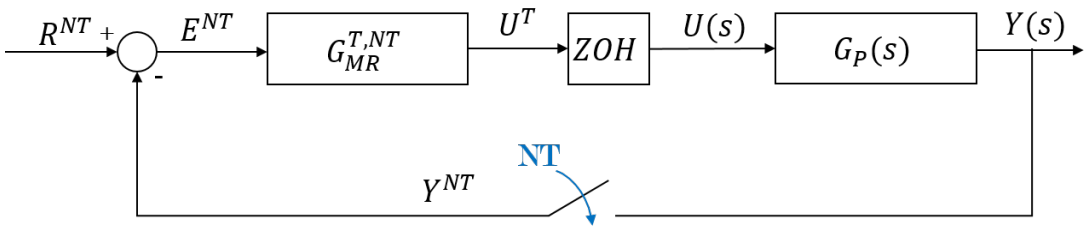


Figura 12.8: Diagrama de bloques de un lazo de control multifrecuencia

En esta imagen, se muestra de forma clara que las lecturas de la salida se realizan a un tiempo de muestreo lento (NT) mientras que el cálculo de las acciones de control se realizan de forma rápida (T) y luego son transformadas en señales continuas mediante un retenedor de orden cero.

Así, el bloque encargado del control tiene como entrada señales muestreadas a NT y como salida acciones de control muestreadas a T. Esto es posible porque el regulador multifrecuencia, tal y como muestra la figura 12.9, esta formado por dos reguladores, uno lento (G_1^{NT}) y otro rápido (G_2^T), y dos operadores (*expand* y H^T) encargados de remuestrear la señal al periodo de muestreo adecuado para el siguiente bloque.

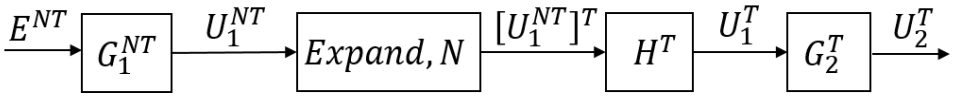


Figura 12.9: Esquema interno de un regulador multifrecuencia

El siguiente paso es calcular los bloques del regulador multifrencuencia a partir del regulador discreto monofrecuencia calculado anteriormente siguiendo las indicaciones del apartado 6.2 donde se definen:

$$G_2^T(z) = \frac{M^T(z)}{G_P^T(z)} \tag{12.16}$$

$$G_1^{NT}(z) = \frac{1}{1 - M^{NT}(z_N)} \tag{12.17}$$

donde $M^XT(z_X)$ es la función de transferencia del bucle cerrado del proceso muestreado cada XT.

Para empezar con la obtención de los reguladores multifrecuencia que se emplearán para el proyecto, el primer paso será obtener las funciones de transferencia tanto del proceso como del bucle cerrado muestreadas cada T, en este caso 0.01 s.

$$M^T(z) = \frac{0.6759z - 0.5265}{z^2 - 1.089z + 0.2386} \tag{12.18}$$

$$G_P^T(z) = \frac{0.5017}{z - 0.7545} \tag{12.19}$$

A continuación se calculará la parte rápida del regulador puesto que, al no depender de N, será la misma para todos los casos.

$$\begin{aligned} G_2^T(z) = \frac{U_2^T}{U_1^T} =& \frac{1.347z^2 - 2.066z + 0.7917}{z^2 - 1.089z + 0.2386} = \\ =& \frac{1.347 - 2.066z^{-1} + 0.7917z^{-2}}{1 - 1.089z^{-1} + 0.2386z^{-2}} \end{aligned} \tag{12.20}$$

A partir de esta expresión, se puede obtener la ecuación en diferencias del regulador para introducirla en el algoritmo de control.

$$\begin{aligned} u_2(k) = 1.089\ u_2(k-1) - 0.2386\ u_2(k-2) + 1.347\ u_1(k) - \\ - 2.066\ u_1(k-1) + 1.132\ u_1(k-2) \end{aligned} \tag{12.21}$$

Por último, será necesario calcular para las diferentes N la parte rápida del regulador.

◦ **Para N=2**

Bucle cerrado muestreado a NT

$$M^{2T} = \frac{0.8856z - 0.5378}{z^2 - 0.7091z + 0.05691} \tag{12.22}$$

Parte rápida del regulador

$$G_1^{2T} = \frac{1 - 0.7091z^{-1} + 0.05691z^{-2}}{1 - 1.595z^{-1} + 0.5947z^{-2}} \tag{12.23}$$

Ecuación en diferencias

$$\begin{aligned} u_1(k) = 1.595\ u_1(k-1) - 0.5947\ u_1(k-2) + \\ + e(k) - 0.7091\ e(k-1) + 0.05691\ e(k-2) \end{aligned} \tag{12.24}$$

◦ **Para N=3**

Bucle cerrado muestreado a NT

$$M^{3T} = \frac{0.9527z - 0.4516}{z^2 - 0.5125z + 0.01358} \tag{12.25}$$

Parte rápida del regulador

$$G_1^{3T} = \frac{1 - 0.5125z^{-1} + 0.01358z^{-2}}{1 - 1.465z^{-1} + 0.4652z^{-2}} \tag{12.26}$$

Ecuación en diferencias

$$\begin{aligned} u_1(k) = 1.465\ u_1(k-1) - 0.4652\ u_1(k-2) + \\ + e(k) - 0.5125\ e(k-1) + 0.01358\ e(k-2) \end{aligned} \tag{12.27}$$

◦ **Para N=4**

Bucle cerrado muestreado a NT

$$M^{4T} = \frac{0.9758z - 0.3615}{z^2 - 0.389z + 0.003239} \tag{12.28}$$

Parte rápida del regulador

$$G_1^{4T} = \frac{1 - 0.389z^{-1} + 0.003239z^{-2}}{1 - 1.365z^{-1} + 0.3648z^{-2}} \tag{12.29}$$

Ecuación en diferencias

$$\begin{aligned} u_1(k) = 1.365\ u_1(k-1) - 0.3648\ u_1(k-2) + \\ + e(k) - 0.389\ e(k-1) + 0.003239\ e(k-2) \end{aligned} \tag{12.30}$$

- **Para N=5**

  Bucle cerrado muestreado a NT

$$M^{5T} = \frac{0.9849z - 0.2855}{z^2 - 0.3014z + 0.0007728} \tag{12.31}$$

  Parte rápida del regulador

$$G_1^{5T} = \frac{1 - 0.3014z^{-1} + 0.0007728z^{-2}}{1 - 1.286z^{-1} + 0.2863z^{-2}} \tag{12.32}$$

  Ecuación en diferencias

$$\begin{aligned} u_1(k) = 1.286\ u_1(k-1) - 0.2863\ u_1(k-2) + \\ + e(k) - 0.3014\ e(k-1) + 0.0007728\ e(k-2) \end{aligned} \tag{12.33}$$

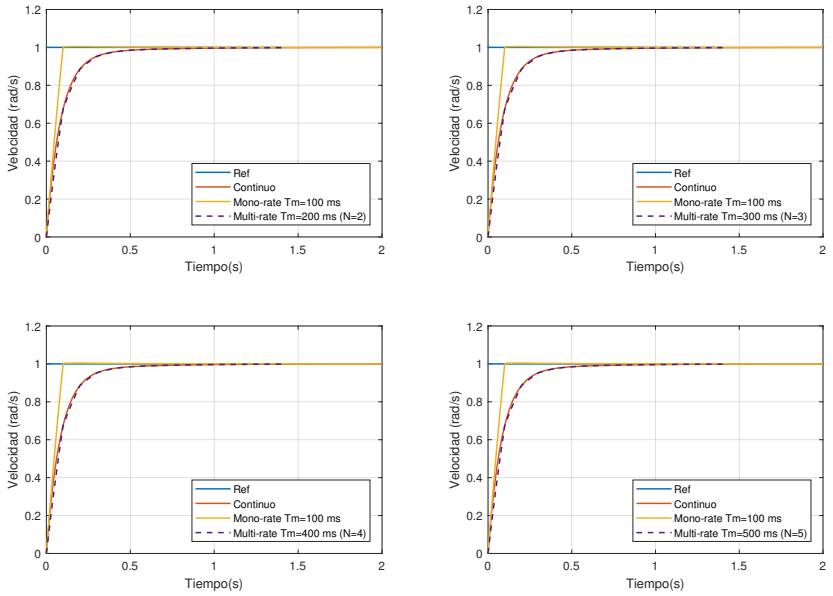


Figura 12.10: Respuesta de los diferentes reguladores *multi-rate*

Analizando las simulaciones realizadas se puede concluir que al variar la N, es decir, hacer menos lecturas de los sensores, la respuesta del sistema no pierde prestaciones. Es importante destacar que, como se trata de un control basado en el modelo, las respuestas de los reguladores multifrecuencia son muy similares a la del regulador continuo. Esto no sucede así con el regulador monofrecuencia porque al realizar la discretización se pierden prestaciones.

Aunque las respuestas obtenidas son buenas, antes de trasladar los reguladores calculados al sistema real para comprobar su funcionamiento se ha decidido calcular la parte rápida del regulador tal y como se indicaba en la expresión (6.31).

Se ha querido trabajar con este nuevo regulador porque el obtenido en la ecuación (12.20), tal y como se explicaba en el apartado 6.2, podía introducir rizado en la respuesta cosa que no es conveniente para este proyecto puesto que finalmente se va a tratar de realizar el seguimiento de diferentes trayectorias.

Así, el nuevo controlador rápido será:

$$G_2^T(z) = \frac{G_R^T(z)}{1 + G_R^T(z)G_P^T(z)} = \frac{2 - 3.009z^{-1} + 1.132z^{-2}}{1 - 0.7511z^{-1} + 0.001921z^{-2}} \tag{12.34}$$

y su ecuación en diferencias:

$$\begin{aligned} u_2(k) = 0.7511\ u_2(k-1) - 0.001921\ u_2(k-2) + 2\ u_1(k) - \\ - 3.009\ u_1(k-1) + 1.132\ u_1(k-2) \end{aligned} \tag{12.35}$$

Tal y como se observa en la figura, la respuesta de este sistema también es válida para todas las N que se han analizado. Sin embargo, por la forma en que se ha calculado la parte rápida del regulador, al contrario que en el caso anterior, la respuesta es más similar a la del regulador monofrecuencia discretizado que a la del continuo.

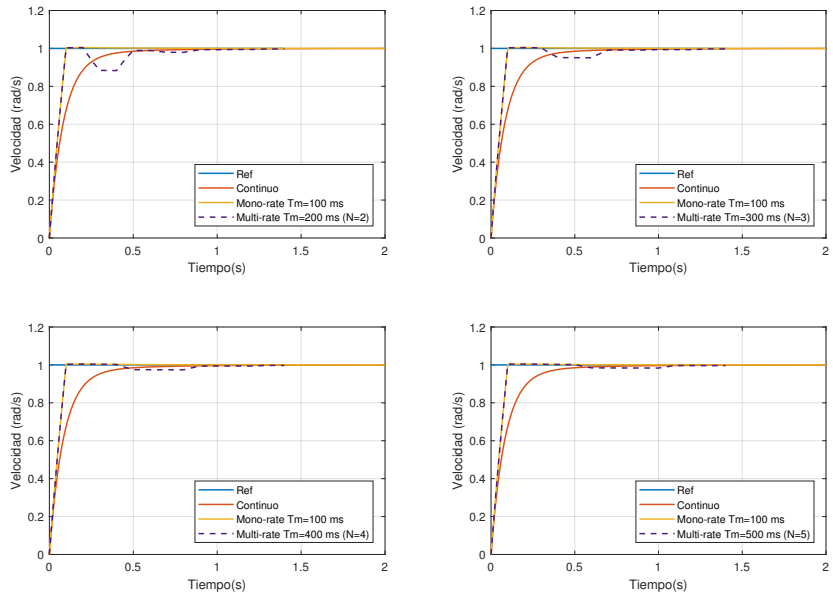


Figura 12.11: Respuesta de los diferentes reguladores *multi-rate* y G_2^T sin ripple

13. Control remoto y retardos

A lo largo del trabajo se ha mencionado, en reiteradas ocasiones, que la necesidad de emplear un control multifrecuencia derivaba de la imposibilidad de realizar un control clásico con el tiempo de muestreo impuesto por las características del sistema.

Como también se ha mencionado anteriormente, se pretende controlar el sistema de forma remota, empleando para ello una red inalámbrica. La finalidad de realizar un control de este estilo radica en el amplio número de posibilidades que se abren para todo tipo de aplicaciones. El control remoto permite desde un seguimiento de una ruta por parte de un vehículo no tripulado con la posibilidad realizar cambios en la trayectoria en tiempo real hasta el control de cualquier proceso o actividad industrial desde una sala de control o cualquier lugar físicamente alejado del propio proceso.

De este modo, la limitación del tiempo de muestreo mínimo será esencialmente consecuencia directa de realizar el control de forma remota. Es decir, será la red compartida la que introducirá en el sistema los retardos que en la mayoría de los casos será necesario tratar para no comprometer el funcionamiento del sistema.

13.1. Estructura del lazo de control

En este caso, se empleará una comunicación mediante radiofrecuencia en la que intervendrán esencialmente dos elementos. El primero de ellos será el microcontrolados encargado de controlar el robot móvil. Este será el elemento local del sistema y tendrá como tareas principales enviar las lecturas de sensores, recibir las referencias y calcular las acciones de control oportunas para el correcto funcionamiento del sistema.

Por el otro lado, se encuentra el ordenador que, actuando de forma remota, será el encargado de recibir las lecturas de los sensores y en función de estas y de la trayectoria que se desee seguir, obtener las referencias que debe seguir el robot.

De este modo, la red transmitirá las lecturas de velocidad angular de cada una de las ruedas del robot al ordenador remoto y las referencias de velocidad en sentido contrario.

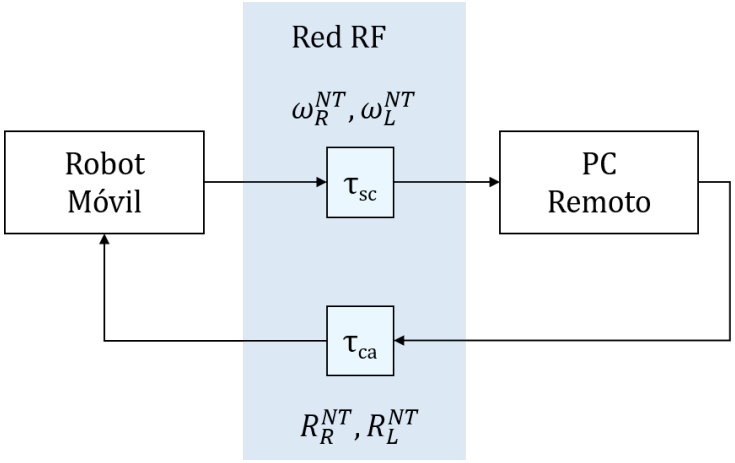


Figura 13.1: Estructura del lazo de control remoto

A efectos prácticos, el lazo de control se puede expresar tal y como muestra la figura 13.2. Se observa claramente que la comunicación afecta al cálculo de las referencias siendo la diferencia de tiempos entre el envío de las velocidades y la recepción de referencias el parámetro conocido como *round-trip*.

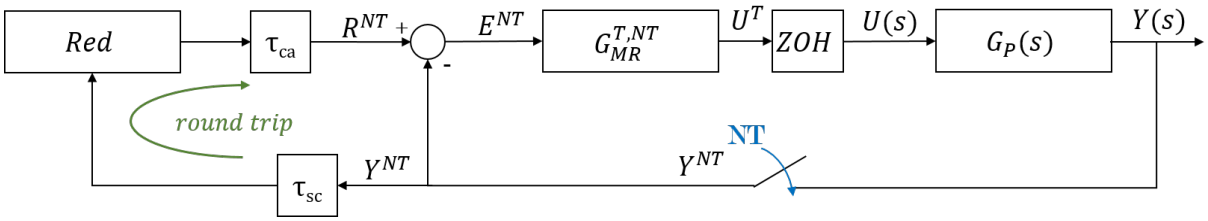


Figura 13.2: Diagrama de bloques del sistema incluyendo la red de comunicación

13.2. Determinación del retardo

Para poder implementar un predictor de Smith y eliminar la influencia de los retardos en el bucle de control, es necesario conocer cuál es la magnitud de estos. En la mayoría de ocasiones el retardo es variable y será necesario medirlo en cada iteración del bucle de control para emplearlo en el predictor de Smith. Sin embargo, en este caso, debido a la estructura con la que se ha configurado el sistema, el retardo es constante y conocido lo que permitirá una sencilla implementación del predictor.

Para entender mejor el funcionamiento del sistema y determinar el valor del retardo se ha elaborado un diagrama temporal que representa cronológicamente las acciones realizadas por los dos elementos del sistema (robot y ordenador remoto) para un control multifrecuencia de periodo rápido T y periodo lento NT siendo N=2.

Como se observa la figura 13.3, en el instante inicial, el robot enviará las medidas de los sensores a través de la red. A continuación, como no dispone de referencias, las considerará nulas y por tanto, la primera acción de control será nula también. Una vez calculada y aplicada dicha acción de control, el sistema se quedará a la espera de aplicar la siguiente acción de control cuando se alcance un tiempo T. Por el otro lado, el ordenador se encuentra a la espera de recibir lecturas desde el instante inicial y en el momento las reciba calculará las referencias y las enviará tan rápido como pueda aunque el robot no las leerá hasta llegado el instante NT.

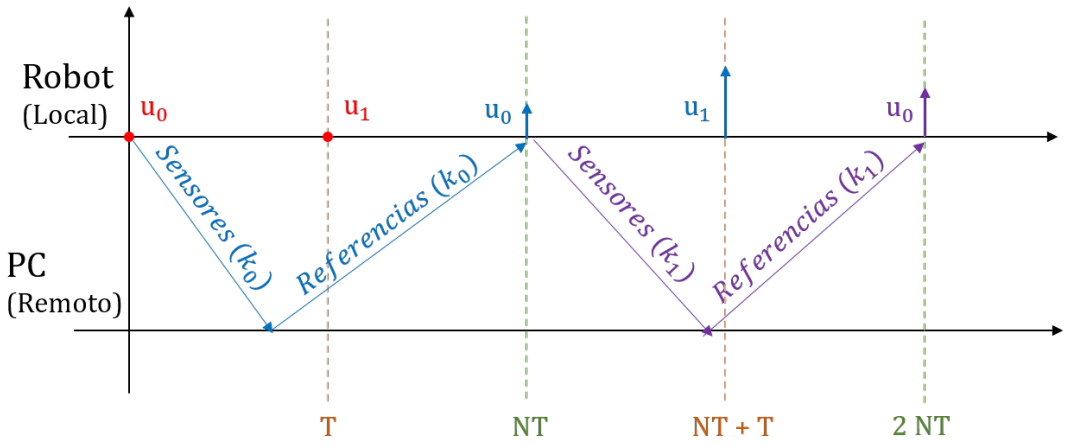


Figura 13.3: Diagrama temporal de funcionamiento del sistema

Una vez enviadas las referencias, el ordenador quedará en espera hasta que se cumpla el primer metaperiodo (NT)), momento en que volverá a leer el puerto serie para obtener las nuevas medidas de los sensores. Por su parte, el robot, tras aplicar la segunda acción de control en el instante T se quedará a la espera hasta completar el primer metaperiodo, momento en el que enviará las medidas de los sensores, leerá las referencias a seguir y calculará y aplicará la primera acción de control del segundo metaperiodo. A partir de este momento, el funcionamiento es exactamente igual que en la primera iteración.

De este modo, se observa de forma clara que las referencias calculadas para el valor de los sensores del primer metaperiodo son empleadas para el cálculo de las acciones de control del segundo, y a su vez las del segundo serán empleadas en el tercero, por lo que se puede concluir que el retardo del sistema es constante e igual a un metaperiodo. El sistema funcionará de igual modo para cualquier valor de N.

13.2.1. Implementación del predictor de Smith

Determinado ya el retardo introducido en el sistema por la red compartida, es el momento de diseñar el predictor de Smith encargado de compensarlo, evitando así el mal funcionamiento del sistema. Si recordamos la estructura del predictor (figura 13.4), este añadía dos nuevas ramas al lazo de control. La primera de ellas era una realimentación positiva de la salida del modelo del sistema incluyendo el retardo y la segunda, una realimentación negativa del modelo sin retardo. De este modo, la efectividad de esta estructura queda supeditada en gran manera a la precisión con la que se haya obtenido el modelo del sistema y el valor de dicho retardo de transporte.

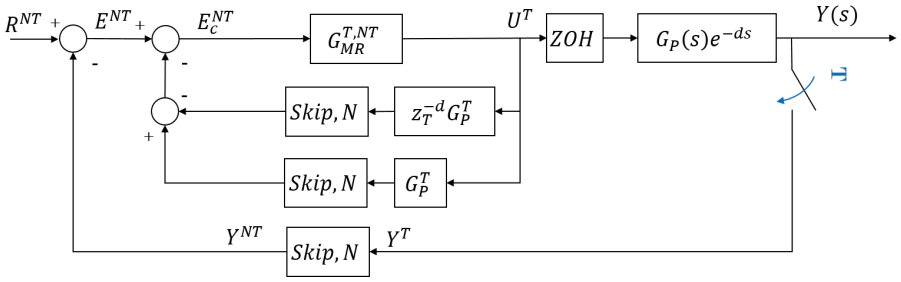


Figura 13.4: Bucle de control con predictor de Smith

La primera rama será la encargada de intentar eliminar el efecto del retardo de transporte en la respuesta del sistema. Con este fin, se obtendrá la salida del modelo del sistema con retardo ($G_P^T z^{-d}$) al aplicar las diferentes acciones de control calculadas. Esta señal de salida, se sumará con la salida global del sistema para anular su efecto.

$$G_P^T z^{-d} = \mathcal{Z}[ZOH \cdot G_P(s)e^{-ds}] = z^{-d}(1-z^{-1})\mathcal{Z}\left[\frac{G_P(s)}{s}\right] \tag{13.1}$$

Por el otro lado, la realimentación negativa, permitirá obtener la salida estimada del sistema sin retardo para cada una de las mencionadas acciones de control. De este modo, tras la cancelación de la salida real con la salida estimada del sistema con retardo, se calculará el error como la diferencia entre la referencia y la salida ahora estimada mediante el modelo de sistema sin retardo (G_P^T) lo que permitirá que el cálculo de las acciones de control no se vea influenciado por el retardo de transporte.

$$G_P^T = \mathcal{Z}[ZOH \cdot G_P(s)] = (1-z^{-1})\mathcal{Z}\left[\frac{G_P(s)}{s}\right] \tag{13.2}$$

En ambas ramas, será necesario incluir el operador *skip* para transformar la señal discreta rápida (muestreada a T) en una señal muestreada a NT.

Para comprobar el funcionamiento de esta estructura se ha obtenido por simulación la respuesta del proceso con y sin retardo. Además se ha simulado la salida del sistema en caso de incluir el retardo en el sistema e implementar el predictor de Smith.

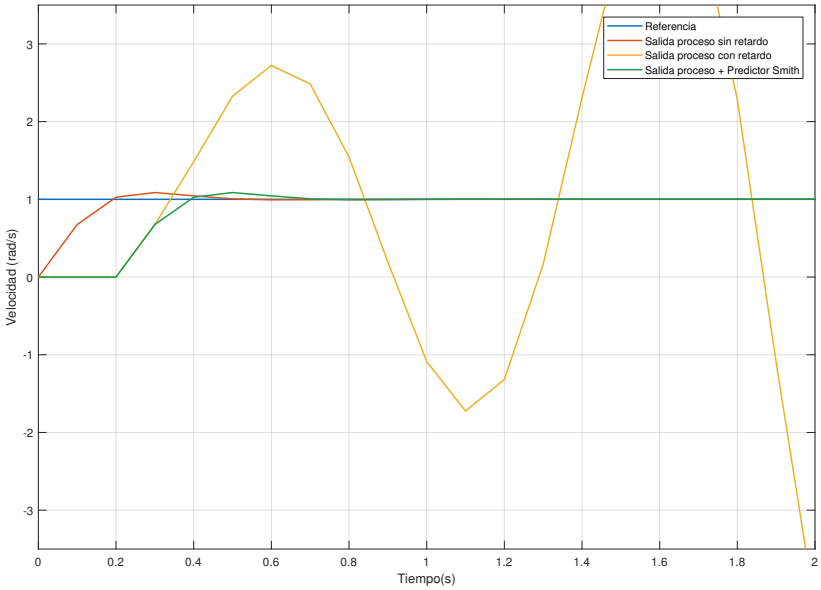


Figura 13.5: Comparación de las respuestas con y sin retardo

Como ya se ha explicado en apartados anteriores, la aparición de un retardo en el lazo de control se traduce en una pérdida de prestaciones en el control y por tanto un empeoramiento de la respuesta. En este caso, tal y como se observa en la figura 13.5, el retardo de transporte es tan significativo que es capaz de inestabilizar el sistema. Sin embargo, con la implementación del predictor de Smith, es posible anular el efecto de dicho retardo en el cálculo de las acciones de control y obtener la misma salida que se obtendría en el sistema sin retardo pero desplazada en el tiempo.

Para finalizar, una vez comprobada la efectividad del predictor de Smith, es necesario obtener las ecuaciones en diferencias que rigen su comportamiento para implementarlas en el algoritmo del microcontrolador.

En primer lugar, se obtendrá la ecuación en diferencias correspondiente a la rama de realimentción negativa (proceso sin retardo).

$$G_P^T = (1 - z^{-1}) \cdot \mathcal{Z}\left[\frac{G_P(s)}{s}\right] = (1 - z^{-1}) \cdot \frac{0.5017 z^{-1}}{1 - 0.7545 z^{-1}} \tag{13.3}$$

$$y_1(k) = 0.7545 y_1(k-1) + 0.5017 u_2(k-1) \tag{13.4}$$

A continuación, se va a obtener la ecuación en diferencias de la rama que incluye el modelo del proceso con el retardo (NT).

$$G_P^T z^{-N} = (1 - z^{-1}) \cdot \mathcal{Z}\left[\frac{G_P(s)e^{-Ns}}{s}\right] = (1 - z^{-1}) \cdot \frac{0.5017z^{-1}}{1 - 0.7545z^{-1}} z^{-n} \tag{13.5}$$

$$y_2(k) = 0.7545y_2(k-1) + 0.5017u_2(k-N-1) \tag{13.6}$$

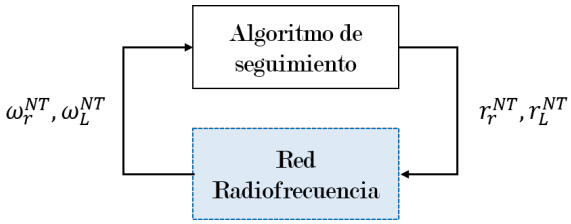
Ahora, tras la implementación del predictor de Smith, el error empleado para calcular las diferentes acciones de control será obtenido de la siguiente manera:

$$e(k) = r(k) - y(k) + y_2(k) - y_1(k) \tag{13.7}$$

siendo $r(k)$ la referencia, $y(k)$ la salida real del sistema, $y_1(k)$ la salida estimada con el modelo del proceso sin retardo y $y_2(k)$ la salida estimada con el modelo del proceso incluyendo retardo.

14. Seguimiento de trayectorias

Llegados a este punto, ya se han obtenido todos los elementos necesarios para implementar en el algoritmo de control del robot de forma que este sea capaz de seguir fielmente unas referencias de velocidad recibidas a través de la red de radiofrecuencia. Así pues, el siguiente paso es explicar cuál será el algoritmo que se empleará para calcular dichas referencias a partir de las velocidad de giro de las ruedas medidas.



El algoritmo de persecución pura, a diferencia de otros algoritmos, necesita conocer de forma previa que trayectoria se desea seguir para funcionar correctamente [13]. De esta forma puede calcular las referencias a partir de la situación del robot y de los puntos donde se desee llegar. En este caso, para realizar las pruebas se ha decidido emplear dos trayectorias diferentes, una flor y un cuadrado.

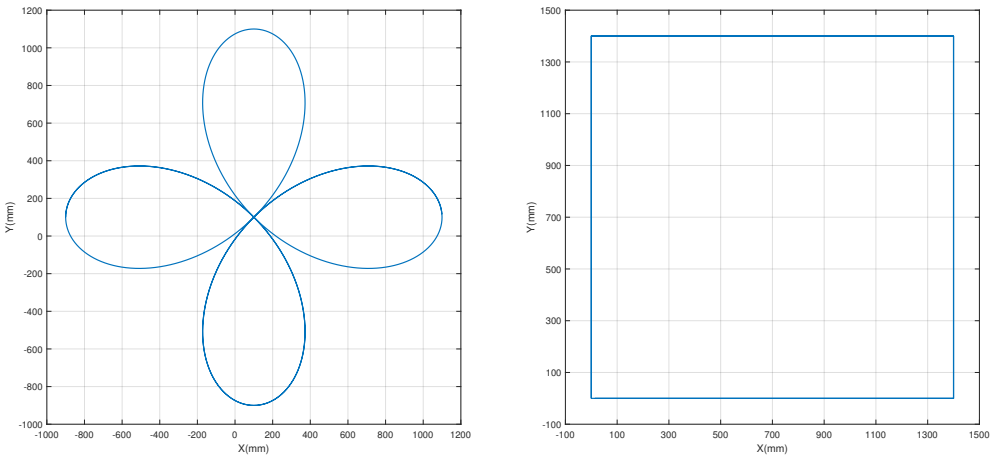


Figura 14.1: Trayectorias empleadas para el análisis del funcionamiento del sistema

El motivo de emplear la flor es que al ser una trayectoria totalmente curva, las velocidades no cambian de forma brusca pero tampoco se mantienen constantes lo que permite analizar si el control es capaz de seguir rápidamente diferentes referencias.

Por otro lado, emplear una trayectoria cuadrada, significa realizar cambios bruscos en las referencias que permitirán analizar la pérdida de prestaciones en la respuesta transitoria del sistema a medida que se aumente el periodo de muestreo con el que se obtienen las lecturas de los sensores.

14.1. Algoritmo "Pure Pursuit"

Antes de centrarnos en el propio algoritmo, se van a introducir una serie de conceptos necesarios para la comprensión del mismo.

La navegación a estima (*Dead reckoning*) es un método empleado para determinar la posición actual de un vehículo a partir de la posición previa y la velocidad que se ha llevado en un periodo de tiempo dado. La manera más sencilla de implementar este método es la conocida como odometría. En este caso, se calculará el desplazamiento a partir de las velocidades de los motores medidas mediante unos encoders.

Por otro lado, la cinemática directa, permite resolver el problema de determinar que posición es posible alcanzar a partir de las velocidades de las ruedas. De este modo, combinando odometría y cinemática directa en el algoritmo, será posible conocer la posición actual del robot y calcular las velocidades necesarias para ir desde dicha posición hasta la deseada.

El objetivo del algoritmo *Pure Pursuit* es calcular la curvatura con la que debe moverse el vehículo para desplazarse de su posición actual a la deseada [14]. En la figura 14.2 se muestra un diagrama explicativo del algoritmo.

Para una velocidad constante (V), en cada instante se calcula el punto de la trayectoria $(x_{obj,m}, y_{obj,m})$ más próximo a la posición del robot (x_R, y_R) y se elige el punto objetivo (x_{obj}, y_{obj}) situado a una distancia fija.

El primer paso pues, será obtener la posición y orientación del robot a partir de las velocidades de las ruedas. Para ello será necesario seguir el siguiente desarrollo:

$$\begin{aligned} v_r &= \omega_r r \\ v_l &= \omega_l r \end{aligned} \tag{14.1}$$

donde r es el radio de las ruedas.

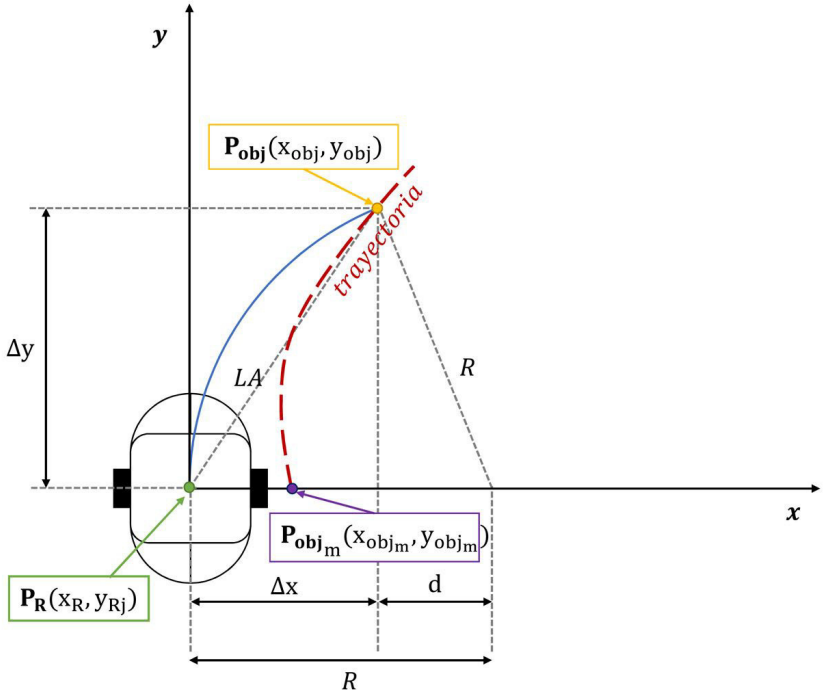


Figura 14.2: Diagrama explicativo del algoritmo *Pure-Pursuit*

Una vez obtenida la velocidad de cada una de las ruedas, las velocidades del robot se obtendrán del siguiente modo:

$$\begin{bmatrix} v_R \\ \omega_R \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2b} & \frac{-1}{2b} \end{bmatrix} \begin{bmatrix} v_r \\ v_l \end{bmatrix} \tag{14.2}$$

Por último, la posición y orientación del robot se calcularan como:

$$\begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta}_R \end{bmatrix} = \begin{bmatrix} x_R \\ y_R \\ \theta_R \end{bmatrix} + \begin{bmatrix} v_R T_s cos(\theta_R) \\ v_R T_s sin(\theta_R) \\ \theta_R + \omega_R T_s \end{bmatrix} \tag{14.3}$$

Una vez obtenida la posición del robot partir del esquema de la figura 14.2, mediante el teorema de pitágoras, se puede obtener la relación:

$$\Delta x^2 + \Delta y^2 = LA^2 \tag{14.4}$$

Y de forma sencilla se deduce que:

$$\Delta x + d = R \rightarrow d = R - \Delta x \tag{14.5}$$

Ahora sustituyendo la expresión (14.5) en la ecuación (14.4) es posible calcular la curvatura:

$$(R - \Delta x)^2 + \Delta y^2 = R^2 \tag{14.6}$$

$$R^2 - 2R\Delta x + \Delta x^2 + \Delta y^2 = R^2 \tag{14.7}$$

$$2R\Delta x = LA^2 \rightarrow R = \frac{LA^2}{2\Delta x} \tag{14.8}$$

$$\gamma = \frac{\mathbf{1}}{\mathbf{R}} = \frac{\mathbf{-2\Delta x}}{\mathbf{LA^2}} \tag{14.9}$$

siendo LA la distancia entre la posición actual del robot y la posición objetivo, calculada como:

$$LA = \sqrt{(x_{obj} - x_R)^2 + (y_{obj} - y_R)^2} \tag{14.10}$$

y el desplazamiento en x:

$$\Delta x = (x_{obj} - x_R)cos\theta + (y_{obj} - y_R)sin\theta \tag{14.11}$$

Por último, será necesario calcular las velocidades de referencia lineales y a partir de ellas las angulares

$$\begin{aligned} v_{ref,r} &= v_R \left(1 + \frac{b}{2}\gamma\right) \\ v_{ref,l} &= v_R \left(1 - \frac{b}{2}\gamma\right) \end{aligned} \tag{14.12}$$

$$\begin{bmatrix} \omega_{ref,r} \\ \omega_{ref,l} \end{bmatrix} = R \begin{bmatrix} v_{ref,r} \\ v_{ref,l} \end{bmatrix} \tag{14.13}$$

Así, las referencias a seguir dependerán de la distancia mínima que se desee que exista entre el punto objetivo y el actual, distancia denominada de *Look-Ahead*, y de la velocidad de referencia media a la que se desea que se mueva el robot.

15. Implementación de los algoritmos

Para finalizar con el desarrollo práctico se va a explicar cual es funcionamiento de los programas empleados para realizar el control remoto del robot móvil.

15.1. Algoritmo del computador remoto

El programa elaborado para realizar las tareas correspondientes al ordenador remoto se ha diseñado mediante el *software* LabVIEW ® y sus principales tareas, serán la recepción de las velocidades de los motores, el cálculo de las referencias pertinentes y su envío a través de la red.

Para su mejor comprensión se ha realizado un diagrama de flujo que ilustra las diferentes tareas que se realizan desde el computador de forma secuencial.

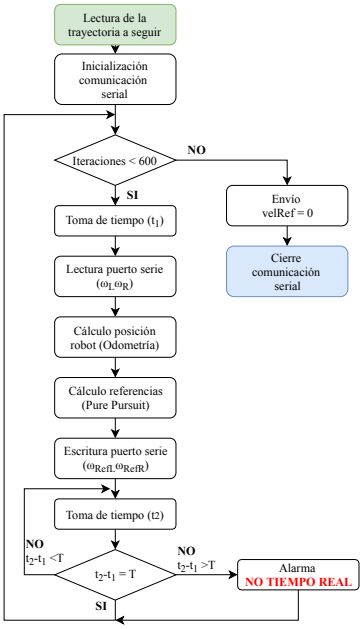


Figura 15.1: Diagrama de flujo del programa correspondiente al ordenador Remoto

15.2. Algoritmo del robot móvil

Como ya se ha mencionado anteriormente, el robot estará controlado en local mediante una placa Arduino DUE y es por esto que el programa se ha elaborado mediante la interfaz de desarrollo propia de Arduino.

Las tareas que debe realizar este programa, tal y como muestra la figura 15.2, serán la recepción de las referencias de velocidad angular a seguir, el cálculo de las acciones de control adecuadas de acuerdo al regulador multifrecuencia, su aplicación a los motores y el envío de las lecturas de los sensores a través de la red.

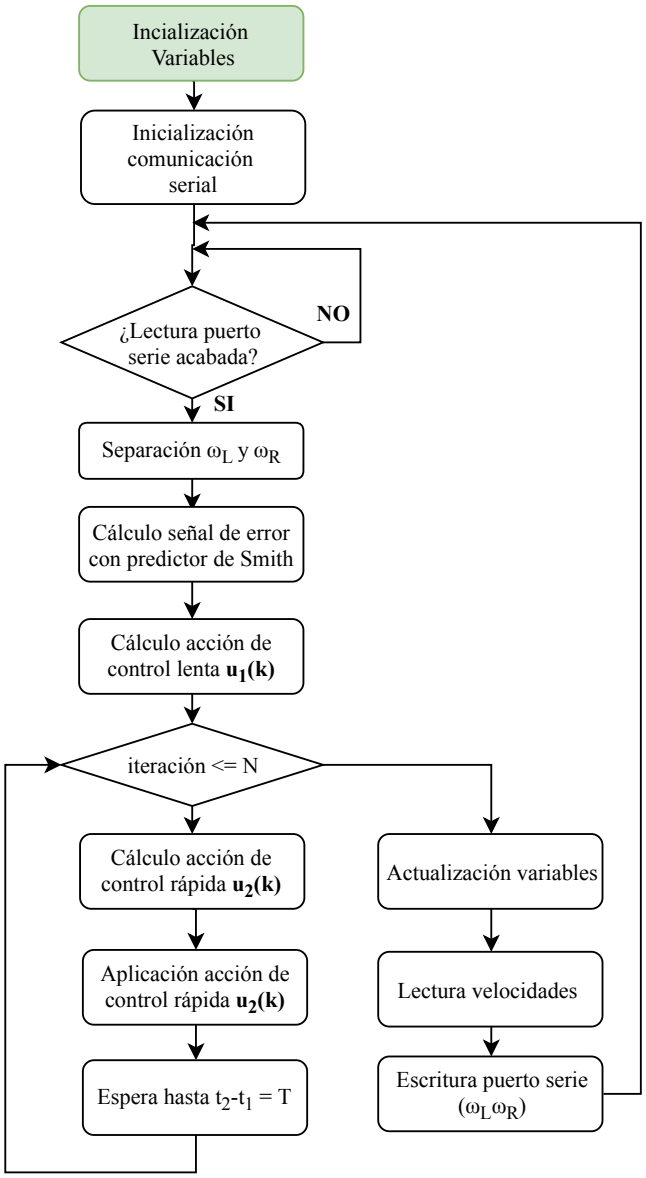


Figura 15.2: Diagrama de flujo del programa correspondiente al control local

Atendiendo a las ecuaciones en diferencias de los reguladores obtenidas en el apartado 12.2, el algoritmo necesita emplear los valores de las diferentes señales en instantes anteriores. Sin embargo, al tratarse de un regulador multifrecuencia, la determinación del valor de dichas señales no es tan inmediata como *a priori* podría parecer.

Como se observa en la figura, para el cálculo de la primera acción de control rápida de un metaperiodo (instante k), el valor de la acción de control lenta correspondiente a dos instantes atrás (k-2) corresponde con el empleado dos periodos rápidos (T) atrás y no el correspondiente a dos metaperiodos (NT). Esto es consecuencia directa de emplear el operador *expand* junto al retenedor H_T.

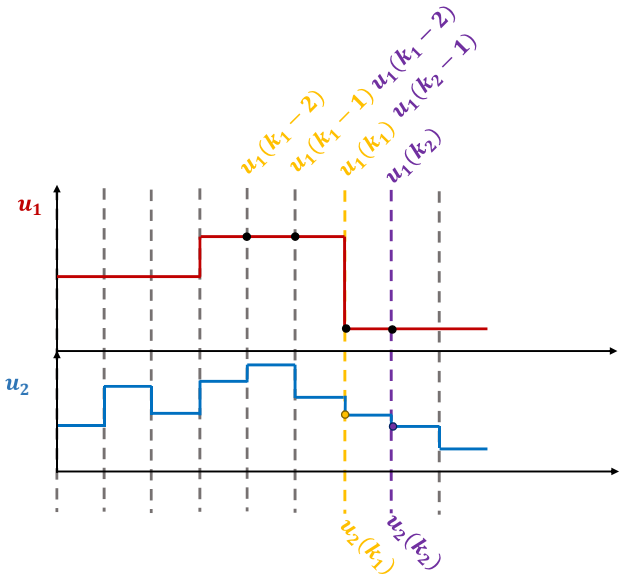


Figura 15.3: Diagrama temporal para el cálculo de acciones de control

Parte V

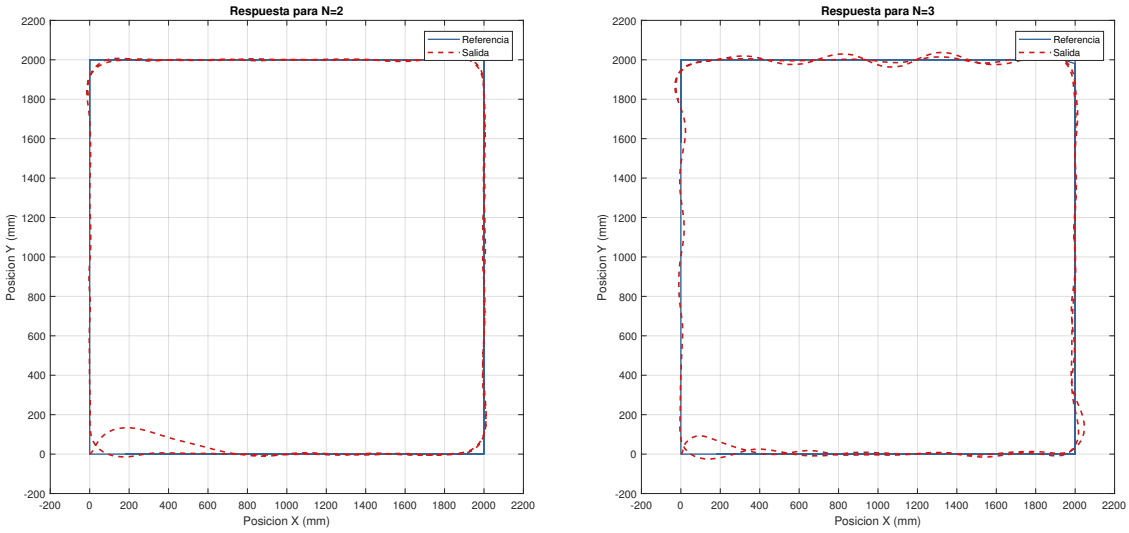
Resultados

16. Trayectorias

Una vez realizados todos los cálculos pertinentes e implementados los algoritmos correspondientes tanto al control local como remoto del robot, es el momento de comprobar con el sistema real el correcto funcionamiento de los controladores diseñados y validados por simulación. Para ello se va querido realizar el seguimiento de diversas trayectorias. Además, para analizar la robustez del sistema se han realizado las mismas pruebas en condiciones diferentes.

16.1. Seguimiento de trayectorias con las ruedas en el aire

En primer lugar, se han realizado las pruebas con las ruedas en el aire para determinar el correcto funcionamiento del sistema antes de introducir en el perturbaciones derivadas del rozamiento y/o deslizamiento de las ruedas sobre el suelo. Es decir, de este modo se pretende comprobar que los algoritmos de control y seguimiento de trayectorias están bien implementados sin introducir otro tipo de incertidumbre.



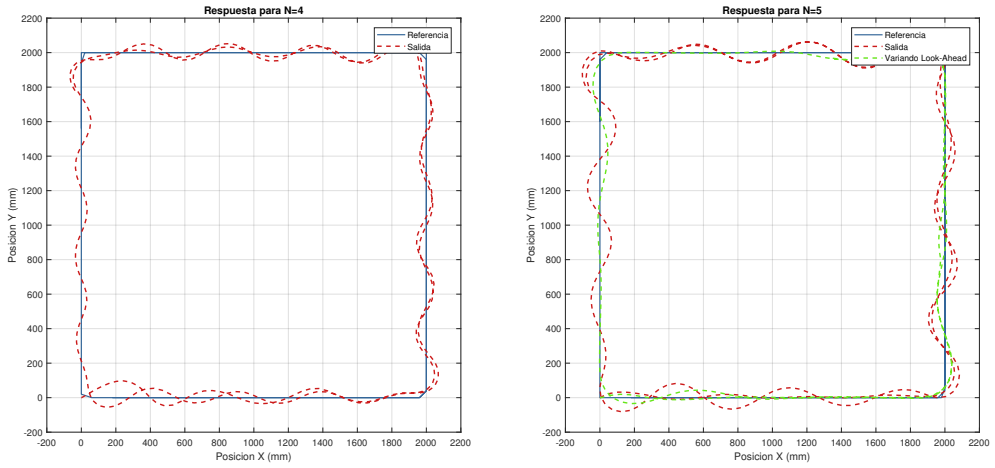


Figura 16.1: Resultados seguimiento de trayectoria cuadrada con las ruedas al aire para diferentes N

De forma cualitativa, observando la figura se puede determinar que el seguimiento de la trayectoria es bueno para las diferentes N. Se puede ver de forma clara que al aumentar el valor dicho parámetro y por tanto obtener menor número de medidas, la respuesta del robot empeora. Esta pérdida de prestaciones, a diferencia de lo que ocurrirá en los siguientes apartados, no es debida a la aparición de distintas perturbaciones comprometen la respuesta del sistema.

Así, la pérdida de prestaciones al realizar el seguimiento de la trayectoria es consecuencia directa los posibles errores de modelado y del algoritmo de seguimiento empleado. Este, tal y como se ha desarollado en el apartado 14.1, se configura de acuerdo a dos parámetros, la velocidad de referencia entorno a la cuál se realizarán variaciones para seguir la trayectoria y la distancia mínima entre el punto actual del robot y el objetivo. Sin embargo, al variar NT, varía el número de accesos al algoritmo. Es decir, al pasar más tiempo entre dos cálculos de referencias consecutivas, las posiciones objetivo están más separadas entre si. Por este motivo, a mayor NT es necesario disminuir la distancia de *look-ahead* con el fin de evitar ese distanciamiento y mejorar la respuesta.

Esta última idea, se puede ver de forma visual en la gráfica correspondiente a N=5 de la figura 16.1. Como se puede observar, los resultados obtenidos inicialmente (sin variar el *look-ahead*), ponen de manifiesto un empeoramiento en la respuesta del sistema, mientras que al variar dicha distancia la respuesta vuelve a mejorar.

Con la idea de apoyar de forma cuantitativa las ideas arriba desarrolladas se ha decidido emplear como indicadores de bondad el error integral absoluto (IEA) y el error integral cuadrático medio (IEC).

$$IEA = \sum_{i=0}^{n} |e(i)| = \sum_{i=0}^{n} \left| \sqrt{[x_{ref}(i) - x_r(i)]^2 + [y_{ref}(i) - y_r(i)]^2} \right| \tag{16.1}$$

$$IEC = \frac{1}{n}\sum_{i=0}^{n} e(i)^2 = \frac{1}{n}\sum_{i=0}^{n}\left(\sqrt{[x_{ref}(i) - x_r(i)]^2 + [y_{ref}(i) - y_r(i)]^2}\right)^2 \tag{16.2}$$

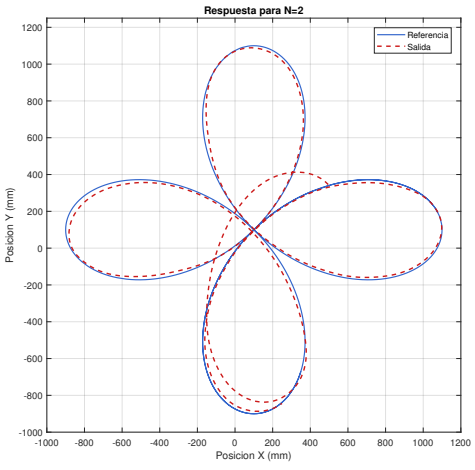
Para que los resultados sean comparables y por tanto, útiles para el análisis, se han calculado dichos índices para un número fijo de muestras. En este caso se han empleado 125 muestras.

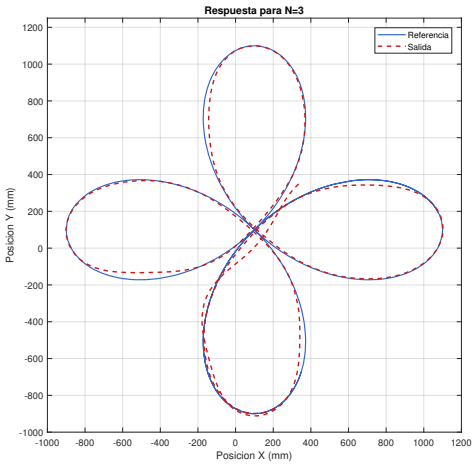
ÍNDICE	**N=2**	**N=3**	**N=4**	**N=5**	**N=5 (ΔLA)**
IEA (mm)	$1.8202{\cdot}10^3$	$2.5473{\cdot}10^3$	$4.3799{\cdot}10^3$	$4.9748{\cdot}10^3$	$2.0260{\cdot}10^3$
IEC (mm^2)	$0.3391{\cdot}10^3$	$0.5509{\cdot}10^3$	$1.5675{\cdot}10^3$	$2.5632{\cdot}10^3$	$0.48340{\cdot}10^3$

Tabla 16.1: Índices de error para el seguimiento de una trayectoria cuadrada con las ruedas en el aire

Los valores de los índices obtenidos corroboran lo descrito anteriormente. Conforme se aumenta el periodo de muestreo con el que se toman las lecturas de sensores (N), el error también lo hace. Esto es consecuencia directa del desajuste existente entre el periodo de muestreo y la distancia (LA) definida en el algoritmo de persecución pura. Es por esto que, con un buen ajuste de dicha distancia, la respuesta puede mejorar hasta comportarse de forma similar que las obtenidas para los menores tiempo de muestreo.

Ahora, se van a presentar los resultados de realizar las mismas pruebas para el seguimiento de una trayectoria de tipo flor.





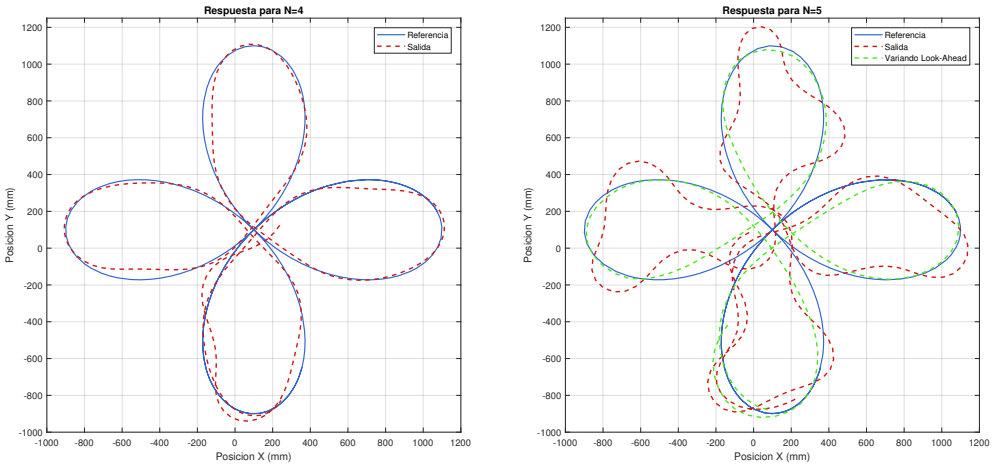


Figura 16.2: Resultados seguimiento de trayectoria tipo flor con las ruedas al aire para diferentes N

Como se observa, los resultados son muy similares a los obtenidos para la trayectoria cuadrada. Al aumentar el valor de N, la respuesta empeora pudiéndose solucionar reajustando el algoritmo de seguimiento. Destacar que tal y como muestra la tabla 16.2, el error acumulado durante 125 muestras es mayor en este caso. Esto es debido a que en el caso de la trayectoria cuadrada las referencias de velocidad eran constantes excepto en los giros por lo que la respuesta del sistema al finalizar el transitorio se estabilizaba en ese valor de referencia. Ahora, en la flor, las referencias cambian constantemente y, por tanto, la salida no llega a estabilizarse por lo que, aún siendo buena la respuesta, el error acumulado es mayor.

ÍNDICE	**N=2**	**N=3**	**N=4**	**N=4 (ΔLA)**	**N=5**	**N=5 (ΔLA)**
IEA (mm)	$0.5978 \cdot 10^4$	$1.1925 \cdot 10^4$	$1.4521 \cdot 10^4$	$0.7072 \cdot 10^4$	$2.1209 \cdot 10^4$	$0.8073 \cdot 10^4$
IEC (mm^2)	$0.3456 \cdot 10^4$	$0.9545 \cdot 10^4$	$1.8307 \cdot 10^4$	$0.8846 \cdot 10^4$	$3.3269 \cdot 10^4$	$0.8796 \cdot 10^4$

Tabla 16.2: Índices de error para el seguimiento de una trayectoria tipo flor con las ruedas en el aire

16.2. Seguimiento de trayectorias en el suelo

Una vez comprobado que el diseño del bucle de control funciona correctamente, se realizarán las mismas pruebas dejando el robot en el suelo. De este modo, será posible analizar cuanto empeora la respuesta del sistema al introducir perturbaciones derivadas del deslizamiento y/o fricción de las ruedas en el suelo.

Igual que en el caso anterior, empezaremos a analizar los resultados del seguimiento de una trayectoria cuadrada.

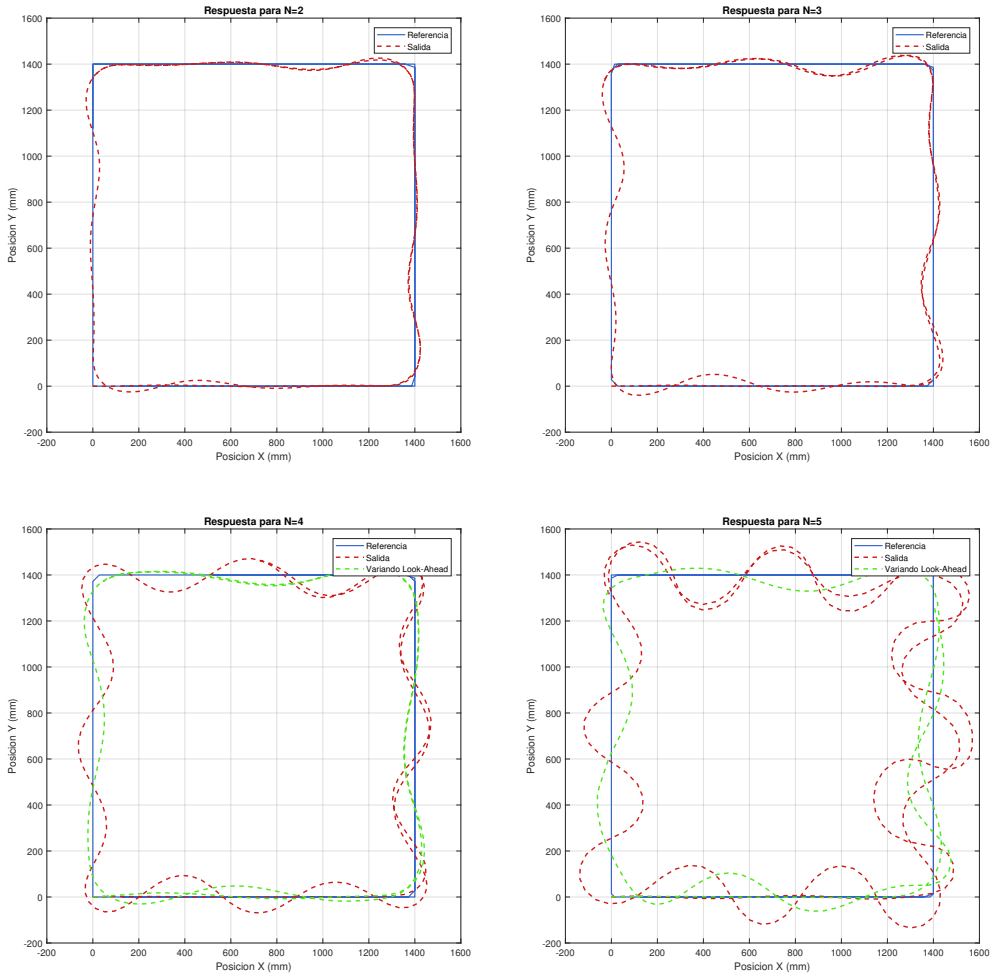


Figura 16.3: Resultados seguimiento de trayectoria cuadrada para diferentes N

ÍNDICE	N=2	N=3	N=4	N=4 (ΔLA)	N=5	N=5 (ΔLA)
IEA (mm)	$0.2053 \cdot 10^4$	$0.3798 \cdot 10^4$	$0.6671 \cdot 10^4$	$0.3123 \cdot 10^4$	$1.2337 \cdot 10^4$	$1.0224 \cdot 10^4$
IEC (mm^2)	$0.0479 \cdot 10^4$	$0.1173 \cdot 10^4$	$0.3792 \cdot 10^4$	$0.08311 \cdot 10^4$	$1.2876 \cdot 10^4$	$0.7797 \cdot 10^4$

Tabla 16.3: Índices de error para el seguimiento de una trayectoria cuadrada

Los resultados, igual que en el caso anterior (apartado 16.1), muestran que a medida que se aumentan el periodo de muestreo lento, la respuesta del sistema empeora. Sin embargo, a diferencia de lo que ocurría con las ruedas en el aire, ahora, modificando la distancia de look ahead es posible mejorar la respuesta, pero no hasta el punto de ser igual de buena que con tiempos de muestreo menores. Esto es debido a que aunque se elimine el error debido al desajuste del algoritmo, lo cierto es que se toman menos lecturas por lo que el sistema se vuelve más vulnerable a las perturbaciones o errores de modelado derivados del contacto existente entre las ruedas y el suelo.

Si se realizan las mismas pruebas para el caso de la trayectoria tipo flor, tal y como muestran tanto la figura 16.4 como la tabla 16.4, los resultados son muy similares a los de la trayectoria cuadrada. De nuevo, el error acumulado será mayor por ser una trayectoria en la que las referencias nunca se estabilizan. Sin embargo, como se puede observar, al ser menores las variaciones de velocidad entorno a la referencia del algoritmo, el transitorio también lo es.

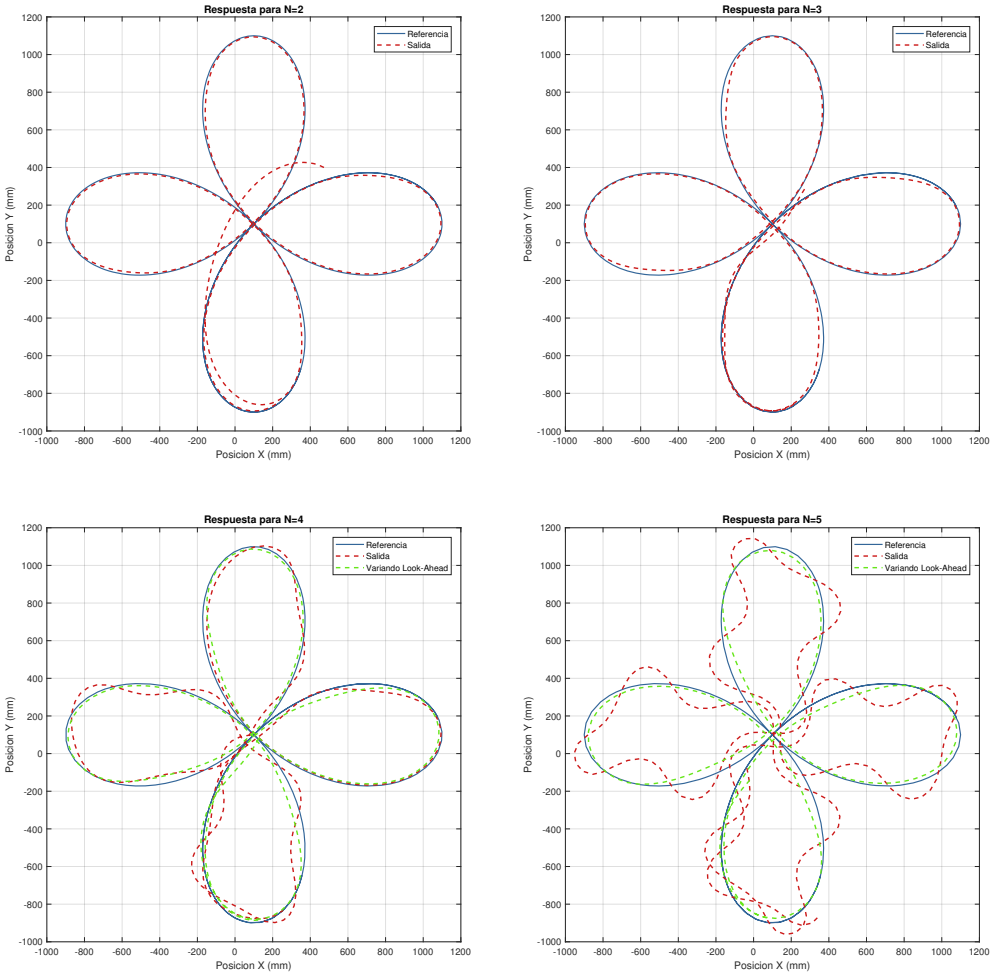


Figura 16.4: Resultados seguimiento de trayectoria tipo flor para diferentes N

ÍNDICE	N=2	N=3	N=4	N=5	N=5 (ΔLA)
IEA (mm)	$0.9341 \cdot 10^4$	$1.0307 \cdot 10^4$	$2.0773 \cdot 10^4$	$2.1608 \cdot 10^4$	$1.6005 \cdot 10^4$
IEC (mm^2)	$0.7637 \cdot 10^4$	$1.8849 \cdot 10^4$	$2.7818 \cdot 10^4$	$3.6312 \cdot 10^4$	$2.6370 \cdot 10^4$

Tabla 16.4: Índices de error para el seguimiento de una trayectoria cuadrada

16.2.1. Seguimiento de trayectorias aumentando la distancia

Al aumentar la distancia entre dos dispositivos comunicados mediante una red inlámbrica, aumenta el tiempo que tarda en llegar una señal y también el número de muestras que se pierden. Es por este motivo que se ha decidido analizar la influencia de este factor en la respuesta del sistema. Es decir, se han realizado las mismas pruebas, pero en este caso, llevando al robot a una ubicación alejada del ordenador remoto.

Destacar que tan solo se han realizado las pruebas para la trayectoría cuadrada puesto que en las pruebas anteriores se ha visto que el cambio de trayectoría no suponía un cambio significativo en la respuesta del sistema.

Así, los resultados obtenidos han sido:

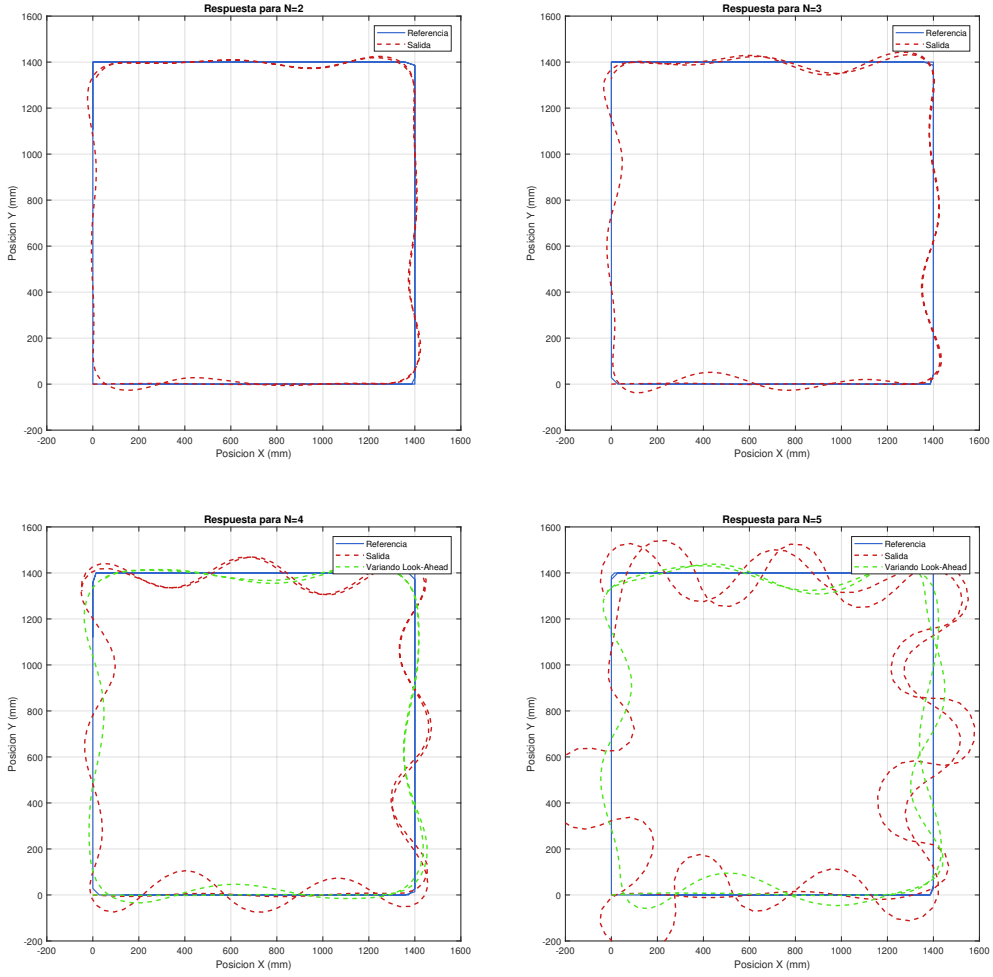


Figura 16.5: Resultados seguimiento de trayectoria cuadrada para diferentes N aumentando la distancia robot-ordenador

ÍNDICE	N=2	N=3	N=4	N=4 (ΔLA)	N=5	N=5 (ΔLA)
IEA (mm)	$0.2945 \cdot 10^4$	$0.3301 \cdot 10^4$	$0.6197 \cdot 10^4$	$0.3054 \cdot 10^4$	$1.5264 \cdot 10^4$	$1.0025 \cdot 10^4$
IEC (mm^2)	$0.0626 \cdot 10^4$	$0.0908 \cdot 10^4$	$0.3389 \cdot 10^4$	$0.0814 \cdot 10^4$	$2.0294 \cdot 10^4$	$0.7262 \cdot 10^4$

Tabla 16.5: Índices de error para el seguimiento de una trayectoria cuadrada aumentando la distancia robot-ordenador

Se puede determinar así, que la respuesta del sistema es buena aunque se aumente la distancia entre el robot y el controlador remoto puesto que las respuestas en esta situación no empeoran significativamente respecto a las obtenidas en el apartado 16.2.

17. Análisis energético

Una de las ventajas de la radiocomunicación, tal y como se ha mencionado anteriormente, es que no necesita mantener la comunicación activa constantemente. Así, a diferencia de otros protocolos como el *Bluetooth*® donde receptor y emisor necesitan estar continuamente emparejados, al comunicarse mediante radiofrecuencia tan solo se consume energía al enviar y/o recibir datos.

Por otro lado, el objetivo principal del proyecto era realizar un control *multi-rate* basado en red que permitiera controlar de forma remota un sistema que por sus características y las de la red no podría controlarse con las técnicas convencionales. Para ello, se han realizado pruebas para diferentes N, de forma que el periodo de muestreo con el que se toman lecturas de los sensores queda definido como NT. Es decir, a mayor N, menor número de muestras se envían al controlador remoto y menor número de referencias se reciben para realizar el control.

Combinando estas dos ideas, es inmediato pensar que al realizar menor número de envíos y recepciones de datos también se reducirá el consumo derivado de ellos. Por tanto, se producirá un aumento en la autonomía del robot, aspecto esencial para todo tipo de sistemas móviles. Por este motivo, se ha decidido realizar un análisis energético para las diferentes N.

Para ello, se obtendrán los valores tanto de tensión como de intensidad a la entrada del controlador mediante el *compactRIO* y se calculará a partir de ellos la potencia consumida. Destacar, que se va a realizar el análisis del consumo energético relacionado directamente con las comunicaciones puesto que al ser el consumo de los motores mucho mayor que el de las antenas empleadas, un análisis global no permitiría observar los cambios de consumo derivados de realizar mayor o menor número de envíos.

En la figura 17.2 se muestran los consumos debidos a la comunicación y el consumo base de la placa Arduino DUE. En las gráficas, se puede observar de forma cualitativa que el consumo del sistema se encuentra entorno a 0.8 W durante la espera y la recepción de datos mientras que se producen picos de 0.92 W durante la transmisión. También se puede observar claramente que, como se esperaba, a medida que se aumenta la N disminuyen los picos de consumo .

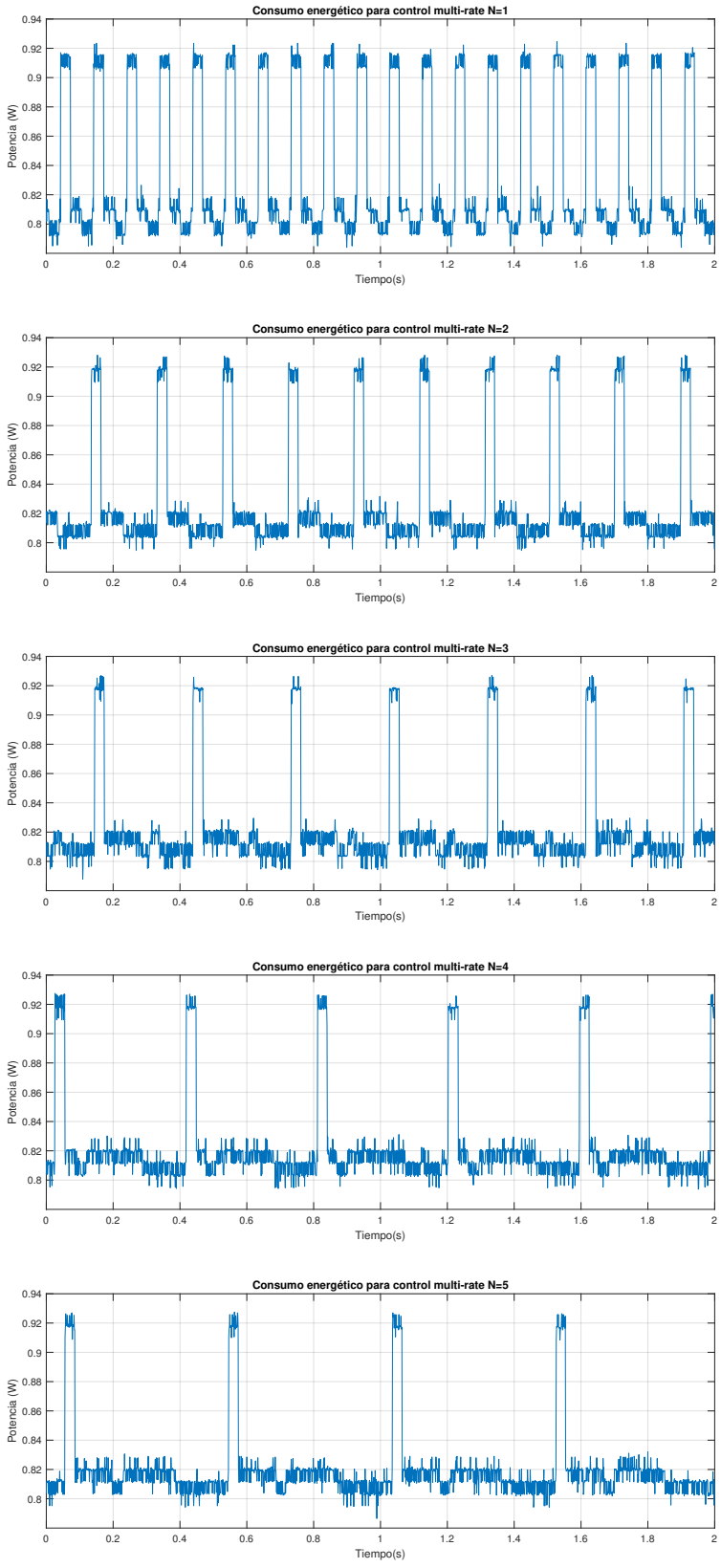


Figura 17.1: Potencia consumida para las comunicaciones con distinta N

Para analizar de forma cuantitativa se ha decidido obtener diversos indicadores de consumo. El primero de ellos será el consumo del sistema en Wh. También se obtendrá la autonomía de la batería del robot (2600mAh) en el caso de tan solo trabajar en comunicaciones y por último se obtendrá el porcentaje de consumo ahorrado para cada una de las diferentes N respecto al control monofrecuencia N=1.

Los resultados obtenidos son los siguientes:

	Consumo	**Autonomía**	**Ahorro respecto N=1**
N=1	0.83561 Wh	25.437 h	———
N=2	0,82781 Wh	25,769 h	0,9336 %
N=3	0,82228 Wh	25,934 h	1,5960 %
N=4	0,82096 Wh	25,969 h	1,7540 %
N=5	0,82035 Wh	25,983 h	1,8260 %

Tabla 17.1: Resultados de consumo para diferentes N

Como se puede observar, el consumo no disminuye notablemente al disminuir el número de envíos y por tanto, al no haber prácticamente ahorro energético, la autonomía de la batería no aumenta. Esto es debido a las características de consumo de las antenas empleadas. El consumo del módulo APC220 en espera sumado con el de la propia placa Arduino DUE es de aproximadamente 0.8 Wh mientras que los picos de consumo en transmisión suben hasta 0.9 Wh. Esto significa que las variaciones son de aproximadamente 0.1 W durante aproximadamente 10 milisegundos dentro de un periodo de más de 100 milisegundos donde hay un consumo de 0.8 W.

En conclusión, los picos de consumo debidos a los envíos son poco significativos respecto al consumo medio en espera por lo que variar el número de envíos no supone un importante ahorro energético.

Llegados a este punto, se ha optado por analizar si el ahorro energético sería significativo si, en lugar del módulo APC220 se hubiera empleado un módulo cuyo consumo en envío y recepción sea mucho mayor que en modo *sleep*. Para realizar esta prueba se han buscado las características técnicas de un módem de radiofrecuencia de alta potencia y se han escalado los valores de corriente obtenidos con el módulo APC220 para que coincidan con los que se habrían obtenido al emplear el nuevo módem. Concretamente, se ha empleado el XTend®-PKG-R RF-Modem por tener las siguientes características [15].

- **Corriente Recepción:** 110 mA
- **Corriente Envío:** $>$ 580 mA
- **Corriente *Sleep*:** 40 mA

Así, se han escalado los valores de corriente, medidos anteriormente para el módulo APC220, para conseguir que coincidan con los valores correspondientes al módem de alta potencia. Una vez realizado el preescalado, se ha realizado el mismo análisis de consumo y se han obtenido los siguientes resultados.

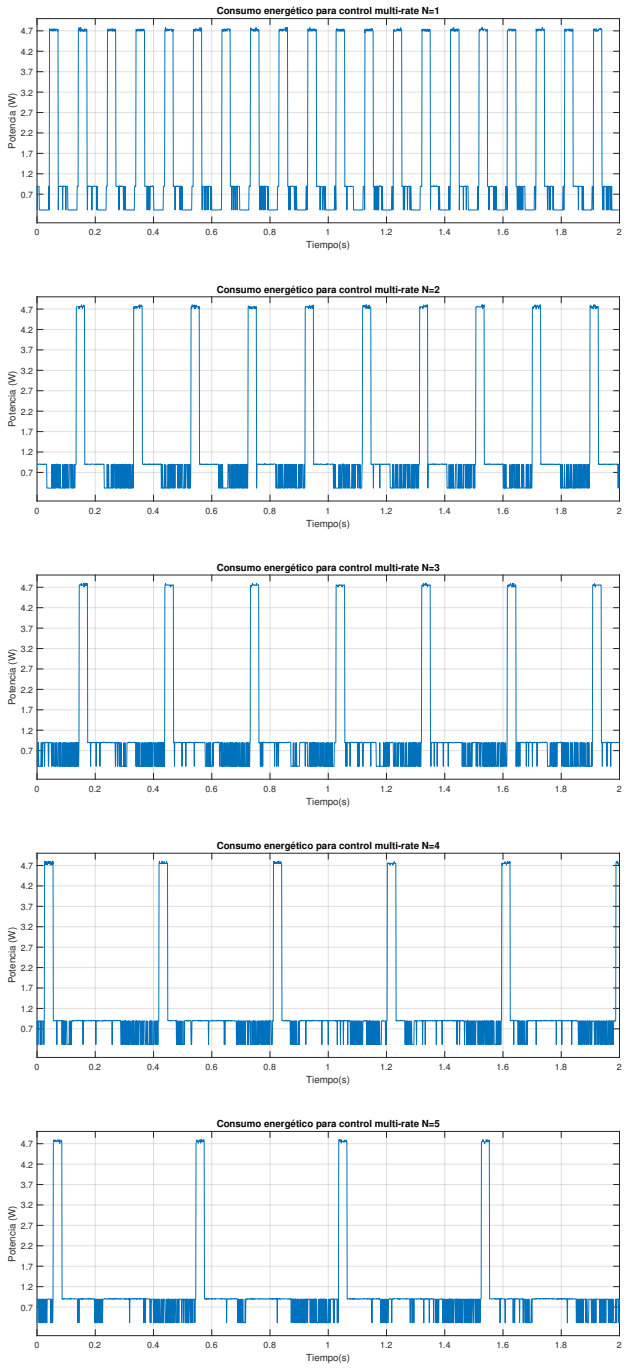


Figura 17.2: Potencia consumida en comunicaciones con el módem XTend®-PKG-R con distinta N

Analizando las gráficas, se puede observar que el consumo en espera está entorno a 0.8 W mientras que los consumos en recepción y tranmisión son de 1 W y 4.7 W respectivamente. Al aumentar la diferencia entre los picos y el consumo en *sleep-mode*, es de esperar que el ahorro energético derivado de disminuir el número de envíos sea significativo. Para comprobarlo, se han vuelto a obtener los indicadores de consumo empleados para el caso del módulo APC220.

	Consumo	**Autonomía**	**Ahorro respecto N=1**
N=1	1.8036 Wh	11.785 h	——
N=2	1.3054 Wh	16.341 h	27.62 %
N=3	1.1141 Wh	19.14 h	38.23 %
N=4	1.0559 Wh	20.192 h	41.46 %
N=5	1.0334 Wh	20.626 h	42.7 %

Tabla 17.2: Resultados de consumo para el módem XTend®-PKG-R con diferentes N

A partir de los resultados obtenidos, se puede determinar que empleando un módem de estas características realizar un menor número de envíos si resulta significativo. También es importante destacar que, el mayor salto en el ahorro energético se produce para N=2. A partir de ahí, conforme se disminuyen el número de envíos, va aumentando el porcentaje de ahorro hasta alcanzar el 40 % para N=4. A partir de ese momento, al continuar disminuyendo el número de envíos, el ahorro ya no es tan significativo como para los primeros casos.

18. Conclusiones

Tras el análisis de los datos obtenidos, se considera que el se ha alcanzado con éxito el objetivo planteado para el proyecto. Sin embargo, es importante destacar algunos aspectos claves que han sido determinantes para el correcto desarrollo del proyecto.

En primer lugar, se ha observado que aunque por simulación la respuesta del sistema controlado mediante multifrecuencia no perdía prestaciones a medida que se aumentaba el periodo lento (NT), en el sistema real, a partir de N=4 la respuesta empezaba a empeorar. Esto es debido al error cometido al realizar el modelado del sistema, ya que se ha obtenido un modelo lineal del motor entorno al punto de funcionamiento pero para las acciones de control mayores, que se alejan de dicho punto, la respuesta deja de comportarse de acuerdo al modelo. Además, otros aspectos como la pérdida de muestras en la comunicación por radiofrecuencia y el rozamiento y/o deslizamiento de las ruedas también influirán en el empeoramiento de la respuesta.

Mencionar también que, si bien la respuesta es buena para diferentes N, aumentar el valor de dicho parámetro significa obtener menor número de muestras y más separadas, es decir; mirar que está ocurriendo menos veces. Por este motivo, aunque al aumentar la N, la respuesta sigue siendo buena, el sistema se vuelve más vulnerable a la aparición de perturbaciones que, entre dos muestras consecutivas, puedan influir en el sistema sin que este sea capaz de eliminarlas al no recibir información de los sensores hasta cumplirse el metaperiodo completo.

En segundo lugar, destacar una idea ya mencionada en el apartado 16.1. Para el seguimiento de las trayectorias se ha empleado el algoritmo *pure pursuit*. Como ya se ha explicado, la idea de este algoritmo es calcular la curvatura que debe llevar el robot para ir de un punto al siguiente, ajustando para ello la distancia de *look-ahead* ($P_{actual} - P_{obj}$). Ahora bien, separar dos muestras en el tiempo y por tanto dos cálculos de referencias supone mantener dicha referencia más tiempo. Este hecho se puede traducir en que entre un cálculo de referencias y el siguiente, el robot haya pasado del punto objetivo y sea necesario volver atrás para recuperar la trayectoria. Por este motivo, para no perder prestaciones, será necesario ir aumentando dicha distancia (LA) a medida que se aumenta el tiempo de muestreo (NT).

Otro aspecto destacable de el algoritmo empleado para el seguimiento de las trayectorias es el hecho de que necesita conocer la trayectoria que se desea seguir de forma previa al control cinemático. Supone un aumento del coste computacional y por tanto del tiempo de ejecución pero permite asegurar que el robot pase por determinados puntos.

Respecto al seguimiento de trayectorias, se ha podido observar que, la respuesta obtenida con las ruedas en el aire y la obtenida al dejar el robot en el suelo son muy similares. En el segundo caso, a causa de la fricción con el suelo, el error cometido en el seguimiento es mayor pero aún así la respuesta del sistema es buena. También se ha podido concluir que aumentar la distancia entre robot y controlador remoto, dentro de los límites marcados por los módulos de comunicación, en este caso no supone un empeoramiento de la respuesta puesto que no aumenta notablemente el número de muestras perdidas. Sin embargo, para sistemas que se comuniquen a distancias mayores, es posible que se pierdan un mayor número de muestras y sea necesario implementar un estimador.

Por último, destacar un aspecto relevante del consumo energético del sistema. Es cierto, que al emplear técnicas multi-rate, ha sido posible disminuir el número de veces que se envían y/o reciben datos en el robot. A priori, esto supone un importante ahorro energético puesto que la radiofrecuencia tan solo consume al realizar dichos envíos y/o recepciones. Sin embargo, como se ha podido ver en el apartado 17, este ahorro tan solo es significativo en el caso de emplear un módem cuyas características sean tales que el consumo en envío y recepción sea significativamente superior al consumo del sistema (módem + controlador) en modo espera.

18.1. Líneas de trabajo futuro

Las líneas de trabajo que se podrían tomar a partir de este proyecto son básicamente tres. En primer lugar, se podría optar por la mejora de los resultados de la odometría para conseguir un mejor resultado en el seguimiento de trayectorias. Para ello se podría emplear, por ejemplo, un sistema de visión artificial.

Otra posibilidad sería optar por transmitir a mayor distancia y analizar la pérdida de prestaciones y el ahorro energético. Para ello, se podría emplear un módem de radiofrecuencia más potente o cambiar el protocolo de comunicación y emplear una red Ethernet, por ejemplo.

Por último, sería interesante trasladar la parte correspondiente al controlador remoto del ordenador a algún dispositivo móvil (tablet o móvil) que permita a un posible operario trabajar con el sistema cómodamente. Es decir se trataría de transcribir el programa realizado en LabVIEW a algún lenguaje válido para ser ejecutado en sistemas operativos como Android o iOS.

Parte VI

Presupuesto

19. Presupuesto

19.1. Introducción

A continuación se presenta el presupuesto estimado para el proyecto realizado. En él, se incluyen tanto los costes relacionados con los equipos y licencias software empleados como también los costes de personal e instalaciones.

En primer lugar se realizará el recuento de los recursos utilizados para la realización del proyecto. A continuación se mostrará el desglose de costes, especificando que porcentaje del coste global de los recursos se corresponde al proyecto. Es decir, se calculará que porcentaje de la vida útil del elemento en cuestión se ha empleado en el proyecto y que valor económico tiene.

En relación a los costes de personal e instalaciones, el proyecto se ha realizado en el marco de una beca de colaboración con la Universitat Politécnica de València. Por este motivo, se ha considerado como único gasto la retribución económica que tiene un becario tipo A. Las instalaciones, han supuesto un coste 0, puesto que el proyecto se ha realizado en el laboratorio de investigación del grupo CO3 en el edificio 5C de la UPV.

19.2. Estado de las mediciones

- **Hardware**

Descripción	Unidades
MSI GL62M 7REX	1
Arduino Due CPU de 32-bit Atmel SAM3X8E ARM Cortex M3	1
2 SOPORTES ALUMINIO PARA MOTORES 37D MM POLOLU	1
2 SOPORTE PARA EJE 6MM M3 POLOLU	1
2 POLOLU WHEEL 80X10MM PAIR - BLACK	1
70:1 METAL GEARMOTOR 37DX70L MM WITH 64 CPR ENCODER	2
ARDUINO SHIELD MOTOR 2A SPARKFUN ARDUMOTO	1
Bateria LI-PO 7.4V 2600mAh 15C	1
Antenas Radiofrecuencia	1
POLOLU BALL CASTER WITH 3/4"METAL BALL	2
IMPRESIÓN 3D DE LA CARCASA	1

Tabla 19.1: Lista de Hardware

- **Licencias Software**

  Tan solo se tendrán en cuenta aquellas licencias de pago. Por tanto, no aparecerán en la tabla aquellos *software* empleados en el proyecto que sean de licencia abierta.

Descripción	Unidades
Licencia Matlab R2017b	1
Licencia NI LabVIEW 16	1

Tabla 19.2: Lista de Software

- **Personal**

Descripción	Unidades
Becario de colaboración tipo A	1

Tabla 19.3: Lista de personal

- **Instalaciones**

Descripción	Unidades
Laboratorio de Investigación	1

Tabla 19.4: Lista de instalaciones

19.3. Desglose de costes

- **Hardware:**

  Destacar que se ha considerado como coste para el proyecto el precio completo de los elementos empleados para realizar el montaje del robot. Sin embargo, para el caso del ordenador, tan solo se ha considerado la parte de la vida útil que se ha empleado durante la realización del proyecto. Es decir, 8 meses de 60 meses de vida útil (5 años).

Descripción	Coste Unitario	Coste Total (IVA Incluido)	Coste Proyecto
MSI GL62M 7REX	866,95 €	1.049,01 €	139,87 €
Arduino Due CPU de 32-bit Atmel SAM3X8E ARM Cortex M3	39,75 €	48,10 €	48,10 €
2 SOPORTES ALUMINIO PARA MOTORES 37D MM POLOLU	6,55 €	7,93 €	7,93 €
2 SOPORTE PARA EJE 6MM M3 POLOLU	7,19 €	8,70 €	8,70 €
2 POLOLU WHEEL 80X10MM PAIR - BLACK	8,38 €	10,14 €	10,14 €
70:1 METAL GEARMOTOR 37DX70L MM WITH 64 CPR ENCODER	38,95 €	94,26 €	94,26 €
ARDUINO SHIELD MOTOR 2A SPARKFUN ARDUMOTO	22,88 €	27,68 €	27,68 €
Bateria LI-PO 7.4V 2600mAh 15C	16,53 €	20,00 €	20,00 €
Antenas Radiofrecuencia	46,28 €	56,00 €	56,00 €
POLOLU BALL CASTER WITH 3/4"METAL BALL	2,70 €	6,53 €	6,53 €
IMPRESIÓN 3D DE LA CARCASA	150,21 €	181,76 €	181,76 €
	TOTAL	1.510,11 €	**600,97 €**

Tabla 19.5: Costes Hardware

El coste total correspondiente al *hardware* es de SEISCIENTOS EUROS CON NOVEINTA Y SIETE CÉNTIMOS.

- **Licencias de Software:**

  Para el cálculo de las licencias software se ha tomado la parte del coste total correspondiente a las horas empleadas. Para ello se ha obtenido tanto el número de horas de trabajo en un año como las que se han empleado para realizar el proyecto.

  Horas de trabajo anuales =8 horas · 5 días · 4 semanas · 11 meses = 1760 h

  Horas de trabajo anuales =4 horas · 5 días · 4 semanas · meses = 640 h

  Coste Proyecto = Coste Total · 640/1760

Descripción	Coste Unitario	Coste Total (IVA Incluido)	Coste Proyecto
Licencia Matlab R2017b	661,16 €	800,00 €	290,91 €
Licencia NI LabVIEW 16	1.228,93 €	1.487,00 €	540,73 €
	TOTAL	2.287,00 €	**831,64 €**

Tabla 19.6: Costes Licencias Software

El coste total correspondiente a las licencias *software* es de OCHOCIENTOS TREINTAIÚN EUROS CON SESEINTA Y CUATRO CÉNTIMOS.

- **Personal:**

  Como ya se ha mencionado, como coste de personal, tan solo se tendrá en cuenta la retribución económica de un becario de colaboración tipo A.

Descripción	**Coste Total**
Becario de colaboración tipo A	2.000,00 €
TOTAL	**2.000,00 €**

Tabla 19.7: Costes Personal

El coste total correspondiente al personal es de DOS MIL EUROS.

- **Instalaciones:**

  Al trabajar en los laboratorios de la Universitat Politécnica de València, no hay coste de instalaciones.

  El coste total correspondiente a las instalaciones es de CERO EUROS.

19.4. Resumen del presupuesto

CONCEPTO	Coste
Costes totales Hardware	600,97 €
Costes totales Software	831,64 €
Costes totales Personal	2.000,00 €
Costes totales Instalaciones	0,00 €
TOTAL	**3.432,61 €**

Tabla 19.8: Coste total del proyecto

El coste total del proyecto es de TRES MIL CUATROCIENTOS TREINTA Y DOS EUROS CON SESEINTAIÚN CÉNTIMOS.

Bibliografía

[1] Richard C Dorf and Robert H Bishop. *Sistemas de control moderno*. 2005.

[2] Katsuhiko Ogata. *Ingeniería de Control Moderna*, volume 53. 5ta edition, 2013.

[3] Julián Salt, Ángel Cuenca, Vicente Casanova, and Antonio Correcher. *CONTROL AUTOMÁTICO Tiempo Continuo y Tiempo Discreto*. 2 edition, 2016.

[4] Julián Salt Ducajú. *Control basado en red de un robot móvil educacional*. PhD thesis, Universitat Politècnica de València, 2016.

[5] Katsuhiko Ogata. *Katsuhiko Ogata Sistemas de Control en Tiempo Discreto*. 2nd edition, 1996.

[6] Mo-Yuen Chow and Y Tipsuwan. Network-based control systems: a tutorial. *Industrial Electronics Society, 2001. IECON '01. The 27th Annual Conference of the IEEE*, **3**(C), 1593–1602 vol.3, 2001.

[7] Raul K Mansano, Eduardo P Godoy, and Arthur J V Porto. The benefits of soft sensor and multi-rate control for the implementation of Wireless Networked Control Systems. *Sensors (Basel, Switzerland)*, **14**(12), 24441–61, 2014.

[8] Julián Salt, Ángel Cuenca, Francisco Palau, and Sebastián Dormido. A multirate control strategy to the slow sensors problem: An interactive simulation tool for controller assisted design. *Sensors (Switzerland)*, **14**(3), 4086–4110, 2014.

[9] Juan Jos and Murillo Fuentes. *Fundamentos de Radiación y Radiocomunicación*.

[10] Pololu. 70:1 Metal Gearmotor 37Dx70L mm with 64 CPR Encoder ← →. Technical Report 3, 2017.

[11] ST Microelectronics. L298 DUAL FULL-BRIDGE DRIVER -Datasheet. Technical report, 2012.

[12] APPCON TECHNOLOGIES. APC220 - RF Transceiver Module Datasheet. Technical report, 2008.

[13] Ángel Soriano Vigueras. Implementación de un algoritmo de persecución pura. Technical report.

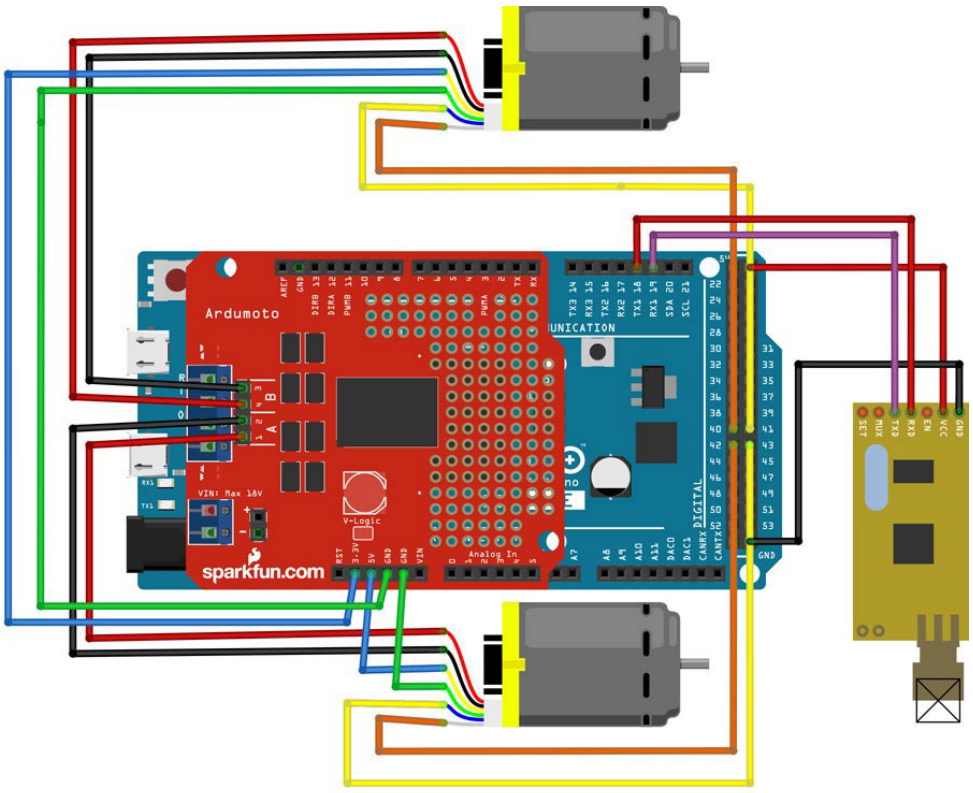
[14] M. Lundgren. Path tracking and obstacle avoidance for a miniature robot. *Umea University, Masterarbeit*, 2003.

[15] DIGI INTERNATIONAL. XTEND ® -PKG - Datasheet. Technical report.

Parte VII

ANEXOS

A. Diagrama de conexiones



B. Códigos Arduino

Código Arduino para Tm = 100 ms y N=1

```
/*
Control MultiRate para Gr = 2(s+2.5)/s
Tm = 100 ms
N = 1
*/

#define RT 13
#define CW  0
#define CCW 1

#define ApinDIR 2                 // A -> Motor Izquierda
#define ApinACC 3
#define BpinDIR 4                // B -> Motor Derecha
#define BpinACC 11

#define ApinENC_chA 42
#define ApinENC_chB 43
#define BpinENC_chA 40
#define BpinENC_chB 41

#define MINU -10.0
#define MAXU 10.0
#define MINW -15.0
#define MAXW 15.0
#define MAXZM 2.4
#define MINZM -2.4

long t1 = 0 ;
long t2 = 0 ;
long t3 = 0 ;
long t4 = 0 ;
int Tr = 100 ;                      // Periodo de muestreo RAPIDO milisegundos
int N = 1;                          // REPETICIONES
int T=N*Tr;                         // Metaperiodo

float Tm = T/1000.0 ;               // Periodo de muestreo segundos
float C2D = 360.0/4480.0 ;          // Conversion cuentas a deg
float C2RS = (2*3.14/4480.0)/Tm ;   // Conversion cuentas a rad/seg

//---INICIALIZACION DE VARIABLES----------
float accA = 0.0 ;
float accB = 0.0 ;
float velA= 0.0 ;
float posA = 0.0 ;
long quadA = 2241 ;
float velB= 0.0 ;
float posB = 0.0 ;
long quadB = 2241 ;
```

```
long enc_antA = 0 ;
long enc_actA = 0 ;
int enc_difA = 0 ;
long enc_antB = 0 ;
long enc_actB = 0 ;
int enc_difB= 0 ;
long enc_antA2;
long enc_antB2;

int velIzq=0;
int velDer=0;
int i;
int k=0;
int pos;
bool p_leido=0;
float refD=0;
float refI=0;
String buffIzq = "0";
String buffDer="0";
char buffb[64] = {0};
String buffOUT="2000";

float A_eck=0;
float A_ek=0;
float A_ek1=0;
float A_ek2=0;
float A_y1k=0;
float A_y2k=0;
float A_y1k1=0;
float A_y2k1=0;
float A_u1k=0;
float A_u1k1=0;
float A_u1k2=0;
float A_u2k=0;
float A_u2k1=0;
float A_u2k2=0;
float A_u2kN=0;
float B_eck=0;
float B_ek=0;
float B_ek1=0;
float B_ek2=0;
float B_y1k=0;
float B_y2k=0;
float B_y1k1=0;
float B_y2k1=0;
float B_u1k=0;
float B_u1k1=0;
float B_u1k2=0;
float B_u2k=0;
float B_u2k1=0;
float B_u2k2=0;
float B_u2kN=0;

float A_U1V[3]= {0};
float B_U1V[3]= {0};

void setup(){
  Serial.begin(19200);
  Serial1.begin(19200); // Establecemos la velocidad del puerto serie (Igual que APC220)
  pinMode(RT,OUTPUT) ;
  Serial1.setTimeout(T/2);

  attachInterrupt(ApinENC_chA,Aencoder_chA,CHANGE) ; //CONFIGURACION INTERRUPCIONES
      ENCODER
  attachInterrupt(ApinENC_chB,Aencoder_chB,CHANGE) ;
  attachInterrupt(BpinENC_chA,Bencoder_chA,CHANGE) ;
  attachInterrupt(BpinENC_chB,Bencoder_chB,CHANGE) ;

  analogReadResolution(8) ;
```

```
  analogWriteResolution(8) ;
  analogWrite(ApinACC,0) ;
  pinMode(ApinDIR,OUTPUT ) ;
  analogWrite(BpinACC,0) ;
  pinMode(BpinDIR,OUTPUT ) ;
  p_leido=0;
  delay(1000);
  i=0;
}

void loop(){
    while (Serial1.available() >=1) {  // Lectura del puerto
           Serial1.readBytes(buffb,8); // mientras haya informacion
           Serial1.flush();            // disponible
           p_leido=1;
           t4=millis();
           Serial.println(t4-t3);
     }

    if(Serial1.available()<0) p_leido=0;
    refD=0;
    refI=0;
    buffDer="";                        // Separacion refD y refI
    buffIzq="";
    buffOUT="";
     for(int k=0; k<4; k++){
       buffDer += buffb[k+4];
       buffIzq += buffb[k];
     }
    refI=buffIzq.toInt();
    refD=buffDer.toInt();
    refI=(refI-1000)/100-10;
    refD=(refD-1000)/100-10;
    if (p_leido==0){
        refD=0;
        refI=0;
    }

     //Calculo de error + Predictor Smith

     A_eck=refI-velA;
     B_eck=refD-velB;

     A_y1k=0.7545*A_y1k1+0.5017*A_u2k1;
     A_y2k=0.7545*A_y2k1+0.5017*A_u2kN;
     A_y1k1=A_y1k;
     A_y2k1=A_y2k;
     A_ek=A_eck+A_y2k-A_y1k;
     A_u2kN=A_u2k1;

     B_y1k=0.7547*B_y1k1+0.5017*B_u2k1;
     B_y2k=0.7545*B_y2k1+0.5017*B_u2kN;
     B_y1k1=B_y1k;
     B_y2k1=B_y2k;
     B_ek=B_eck+B_y2k-B_y1k;
     B_u2kN=B_u2k1;

    // Calculo accion de control
     A_u2k=A_u2k1+2*A_ek-1.5*A_ek1;
     A_u2k1=A_u2k;
     accA=A_u2k;

     B_u2k=B_u2k1+2*B_ek-1.5*B_ek1;
     B_u2k1=B_u2k;
     accB=B_u2k;

     // Aplicar accion de control
     if (accA>=MAXU) {accA=MAXU ;} ; if (accA<=MINU) {accA=MINU ;} //Saturar acciones de
          control
```

```
185                if (accB>=MAXU) {accB=MAXU ;} ; if (accB<=MINU) {accB=MINU ;}
186                if (accA>0) {
187                          digitalWrite(ApinDIR,CW) ;
188                 } else {digitalWrite(ApinDIR,CCW);}
189                 if (accB>0) {
190                          digitalWrite(BpinDIR,CW) ;
191                 } else {digitalWrite(BpinDIR,CCW) ;}
192
193               analogWrite(ApinACC,(byte)(abs(accA)*255.0/MAXU)) ;
194               analogWrite(BpinACC,(byte)(abs(accB)*255.0/MAXU)) ;
195
196    // Actualizar variables
197        A_u1k2=A_u1k1;
198        A_u1k1=A_u1k;
199        A_ek2=A_ek1;
200        A_ek1=A_ek;
201
202        B_u1k2=B_u1k1;
203        B_u1k1=B_u1k;
204        B_ek2=B_ek1;
205        B_ek1=B_ek;
206
207       // Lectura y calculo de velocidades
208       enc_actA=quadA;
209       enc_difA=enc_actA-enc_antA ;
210       posA=(enc_actA)*C2D ; //pos=((int)pos % 360)-180.0 ;
211       velA=(enc_actA-enc_antA)*C2RS ;
212       enc_antA2=enc_antA;
213       enc_antA=enc_actA ;
214
215       enc_actB=quadB ;
216       enc_difB=enc_actB-enc_antB ;
217       posB=(enc_actB)*C2D ; //pos=((int)pos % 360)-180.0 ;
218       velB=(enc_actB-enc_antB)*C2RS ;
219       enc_antB2=enc_antB;
220       enc_antB=enc_actB ;
221
222       // Envio velocidades
223       velIzq=(velB+10)*100+1000;
224       velDer=(velA+10)*100+1000;
225       buffOUT+=velDer;
226       buffOUT+=velIzq;
227       Serial1.print(buffOUT);
228       t3=millis();
229       t2=millis();
230
231       // Espera para cumplir tiempo real
232   if (t2-t1>T) {digitalWrite(RT,HIGH);
233                 // Serial1.print(" ");
234                  Serial.println(t2-t1);
235                  }
236   else {digitalWrite(RT,LOW) ;
237         Serial.println(t2-t1);
238        }
239   while (t2-t1<T) {t2=millis() ;}
240   t1=millis() ;
241 }
242
243 // Handlers de las interrupciones para los encoders
244
245 void Aencoder_chA (){if(digitalRead(ApinENC_chA)==digitalRead(ApinENC_chB)){ quadA--;}
        else{quadA++;} }
246 void Aencoder_chB (){if(digitalRead(ApinENC_chA)==digitalRead(ApinENC_chB)){ quadA++;}
        else{quadA--;} }
247
248 void Bencoder_chA (){if(digitalRead(BpinENC_chA)==digitalRead(BpinENC_chB)){ quadB--;}
        else{quadB++;} }
249 void Bencoder_chB (){if(digitalRead(BpinENC_chA)==digitalRead(BpinENC_chB)){ quadB++;}
        else{quadB--;} }
```

Código Arduino para Tm = 100 ms y N=2

```
/*
Control MultiRate para Gr = 2(s+2.5)/s
Tm = 100 ms
N = 2
*/

#define RT 13

#define CW  0
#define CCW 1

#define ApinDIR 2                 // A -> Motor Izquierda
#define ApinACC 3

#define BpinDIR 4                // B -> Motor Derecha
#define BpinACC 11

#define ApinENC_chA 42
#define ApinENC_chB 43
#define BpinENC_chA 40
#define BpinENC_chB 41

#define MINU -10.0
#define MAXU 10.0
#define MINW -15.0
#define MAXW 15.0

#define MAXZM 2.4
#define MINZM -2.4

long t1 = 0 ;
long t2 = 0 ;
long t3 = 0 ;
long t4 = 0 ;
int Tr = 100 ;                     // Periodo de muestreo RAPIDO milisegundos
int N = 2;                         // REPETICIONES
int T=N*Tr;                        // Periodo de muestreo LENTO milisegundos

float Tm = T/1000.0 ;               // Periodo de muestreo segundos
float C2D = 360.0/4480.0 ;          // Conversion cuentas a deg
float C2RS = (2*3.14/4480.0)/Tm ;   // Conversion cuentas a rad/seg

//---INICIALIZACION DE VARIABLES----------
float accA = 0.0 ;
float accB = 0.0 ;
float velA= 0.0 ;
float posA = 0.0 ;
long quadA = 2241 ;
float velB= 0.0 ;
float posB = 0.0 ;
long quadB = 2241 ;
long enc_antA = 0 ;
long enc_actA = 0 ;
int enc_difA = 0 ;
long enc_antB = 0 ;
long enc_actB = 0 ;
int enc_difB= 0 ;
long enc_antA2;
long enc_antB2;

int velIzq=0;
int velDer=0;
int i;
int k=0;
int pos;
bool p_leido=0;
```

```
67 float refD=0;
68 float refI=0;
69 String buffIzq = "0";
70 String buffDer="0";
71 char buffb[64] = {0};
72 String buffOUT="20002000";
73
74 float A_eck=0;
75 float A_ek=0;
76 float A_ek1=0;
77 float A_ek2=0;
78 float A_y1k=0;
79 float A_y2k=0;
80 float A_y1k1=0;
81 float A_y2k1=0;
82 float A_u1k=0;
83 float A_u1k1=0;
84 float A_u1k2=0;
85 float A_u2k=0;
86 float A_u2k1=0;
87 float A_u2k2=0;
88 float A_u2kN=0;
89
90 float B_eck=0;
91 float B_ek=0;
92 float B_ek1=0;
93 float B_ek2=0;
94 float B_y1k=0;
95 float B_y2k=0;
96 float B_y1k1=0;
97 float B_y2k1=0;
98 float B_u1k=0;
99 float B_u1k1=0;
100 float B_u1k2=0;
101 float B_u2k=0;
102 float B_u2k1=0;
103 float B_u2k2=0;
104 float B_u2kN=0;
105
106 float A_U1V[3]= {0};
107 float B_U1V[3]= {0};
108
109 void setup(){
110   Serial.begin(19200);
111   Serial1.begin(19200); // Establecemos la velocidad del puerto serie (Igual que APC220)
112   pinMode(RT,OUTPUT) ;
113   Serial1.setTimeout(T/2);
114
115   attachInterrupt(ApinENC_chA,Aencoder_chA,CHANGE) ; //CONFIGURACION INTERRUPCIONES
          ENCODER
116   attachInterrupt(ApinENC_chB,Aencoder_chB,CHANGE) ;
117   attachInterrupt(BpinENC_chA,Bencoder_chA,CHANGE) ;
118   attachInterrupt(BpinENC_chB,Bencoder_chB,CHANGE) ;
119
120   analogReadResolution(8) ;
121   analogWriteResolution(8) ;
122   analogWrite(ApinACC,0) ;
123   pinMode(ApinDIR,OUTPUT ) ;
124   analogWrite(BpinACC,0) ;
125   pinMode(BpinDIR,OUTPUT ) ;
126   p_leido=0;
127   delay(1000);
128   i=0;
129 }
130
131 void loop(){
132     while (Serial1.available() >=1) {  // Lectura del puerto
133             Serial1.readBytes(buffb,8); // mientras haya informacion
134             Serial1.flush();            // disponible
```

```
135           p_leido=1;
136           t4=millis();
137           Serial.println(t4-t3);
138       }
139
140     if(Serial1.available()<0) p_leido=0;
141     refD=0;
142     refI=0;
143     buffDer="";                          // Separacion refD y refI
144     buffIzq="";
145     buffOUT="";
146     for(int k=0; k<4; k++){
147       buffDer += buffb[k+4];
148       buffIzq += buffb[k];
149     }
150     refI=buffIzq.toInt();
151     refD=buffDer.toInt();
152     refI=(refI-1000)/100-10;
153     refD=(refD-1000)/100-10;
154     if (p_leido==0){
155       refD=0;
156       refI=0;
157       }
158
159     //Calculo de error + Predictor Smith
160
161       A_eck=refI-velA;
162       B_eck=refD-velB;
163
164       A_y1k=0.7545*A_y1k1+0.5017*A_u2k1;
165       A_y2k=0.7545*A_y2k1+0.5017*A_u2kN;
166       A_y1k1=A_y1k;
167       A_y2k1=A_y2k;
168       A_ek=A_eck+A_y2k-A_y1k;
169       A_u2kN=A_u2k;
170
171       B_y1k=0.7545*B_y1k1+0.5017*B_u2k1;
172       B_y2k=0.7545*B_y2k1+0.5017*B_u2kN;
173       B_y1k1=B_y1k;
174       B_y2k1=B_y2k;
175       B_ek=B_eck+B_y2k-B_y1k;
176       B_u2kN=B_u2k;
177
178 // -----------------CALCULO ACCIONES DE CONTROL-------------------------
179
180 // %%%%%%%%%%%%%%%%%%%%%%%%BUCLE RAPIDO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
181       A_u1k=1.595*A_u1k1-0.5947*A_u1k2+A_ek-0.7091*A_ek1+0.05691*A_ek2;
182       B_u1k=1.595*B_u1k1-0.5947*B_u1k2+B_ek-0.7091*B_ek1+0.05691*B_ek2;
183
184 // %%%%%%%%%%%%%%%%%%%%%%%%BUCLE LENTO  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
185
186       //A_u2k=1.089*A_u2k1 - 0.2386*A_u2k2 + 1.347*A_u1k - 2.066*A_u1k1 + 0.7917*A_u1k1;
              // GT2 = MT/GpT
187       A_u2k=0.7511*A_u2k1 - 0.001921*A_u2k2 + 2*A_u1k - 3.009*A_u1k1 + 1.132*A_u1k1; //
              GT2 = Grd/(1+Grd*GpT)
188       A_u2k2=A_u2k1;
189       A_u2k1=A_u2k;
190       accA=A_u2k;
191      //B_u2k=1.089*B_u2k1 - 0.2386*B_u2k2 + 1.347*B_u1k - 2.066*B_u1k1 + 0.7917*B_u1k1;
             // GT2 = MT/GpT
192       B_u2k=0.7511*B_u2k1 - 0.001921*B_u2k2 + 2*B_u1k - 3.009*B_u1k1 + 1.132*B_u1k1; //
              GT2 = Grd/(1+Grd*GpT)
193       B_u2k2=B_u2k1;
194       B_u2k1=B_u2k;
195       accB=B_u2k;
196
197       if (accA>=MAXU) {accA=MAXU ;} ; if (accA<=MINU) {accA=MINU ;} //Saturar acciones de
              control
198               if (accB>=MAXU) {accB=MAXU ;} ; if (accB<=MINU) {accB=MINU ;}
```

```
199               if (accA>0) {
200                         digitalWrite(ApinDIR,CW) ;
201                } else {digitalWrite(ApinDIR,CCW);}
202                if (accB>0) {
203                         digitalWrite(BpinDIR,CW) ;
204                } else {digitalWrite(BpinDIR,CCW) ;}
205
206              analogWrite(ApinACC,(byte)(abs(accA)*255.0/MAXU)) ; // Aplicar acciones de
                    control
207              analogWrite(BpinACC,(byte)(abs(accB)*255.0/MAXU)) ;
208       t2=millis();
209
210       if (t2-t1>Tr) {digitalWrite(RT,HIGH);  //Espera hasta alcanzar T
211                // Serial1.print(" ");
212                 Serial.println(t2-t1);
213                 }
214       else {digitalWrite(RT,LOW) ;
215            Serial.println(t2-t1);
216       }
217      //A_u2k=1.089*A_u2k1 - 0.2386*A_u2k2 + 1.347*A_u1k - 2.066*A_u1k + 0.7917*A_u1k1;
             // GT2 = MT/GpT
218      A_u2k=0.7511*A_u2k1 - 0.001921*A_u2k2 + 2*A_u1k - 3.009*A_u1k + 1.132*A_u1k1; //
             GT2 = Grd/(1+Grd*GpT)
219      A_u2k2=A_u2k1;
220      A_u2k1=A_u2k;
221      accA=A_u2k;
222     //B_u2k=1.089*B_u2k1 - 0.2386*B_u2k2 + 1.347*B_u1k - 2.066*B_u1k + 0.7917*B_u1k1; //
             GT2 = MT/GpT
223      B_u2k=0.7511*B_u2k1 - 0.001921*B_u2k2 + 2*B_u1k - 3.009*B_u1k + 1.132*B_u1k1; //
             GT2 = Grd/(1+Grd*GpT)
224      B_u2k2=B_u2k1;
225      B_u2k1=B_u2k;
226      accB=B_u2k;
227
228      if (accA>=MAXU) {accA=MAXU ;} ; if (accA<=MINU) {accA=MINU ;} //Saturar acciones de
              control
229              if (accB>=MAXU) {accB=MAXU ;} ; if (accB<=MINU) {accB=MINU ;}
230              if (accA>0) {
231                         digitalWrite(ApinDIR,CW) ;
232                } else {digitalWrite(ApinDIR,CCW);}
233                if (accB>0) {
234                         digitalWrite(BpinDIR,CW) ;
235                } else {digitalWrite(BpinDIR,CCW) ;}
236
237              analogWrite(ApinACC,(byte)(abs(accA)*255.0/MAXU)) ; // Aplicar acciones de
                    control
238              analogWrite(BpinACC,(byte)(abs(accB)*255.0/MAXU)) ;
239
240       // Actualizar variables
241       A_u1k2=A_u1k1;
242       A_u1k1=A_u1k;
243       A_ek2=A_ek1;
244       A_ek1=A_ek;
245
246       B_u1k2=B_u1k1;
247       B_u1k1=B_u1k;
248       B_ek2=B_ek1;
249       B_ek1=B_ek;
250
251      // Lectura y calculo de velocidades
252      enc_actA=quadA;
253      enc_difA=enc_actA-enc_antA ;
254      posA=(enc_actA)*C2D ; //pos=((int)pos % 360)-180.0 ;
255      velA=(enc_actA-enc_antA)*C2RS ;
256      enc_antA2=enc_antA;
257      enc_antA=enc_actA ;
258
259      enc_actB=quadB ;
260      enc_difB=enc_actB-enc_antB ;
```

```
261       posB=(enc_actB)*C2D ; //pos=((int)pos % 360)-180.0 ;
262       velB=(enc_actB-enc_antB)*C2RS ;
263       enc_antB2=enc_antB;
264       enc_antB=enc_actB ;
265
266       // Envio velocidades
267       velIzq=(velB+10)*100+1000;
268       velDer=(velA+10)*100+1000;
269       buffOUT+=velDer;
270       buffOUT+=velIzq;
271       Serial1.print(buffOUT);
272       t3=millis();
273       t2=millis();
274
275   if (t2-t1>T) {digitalWrite(RT,HIGH); // Espera hasta NT
276                 // Serial1.print(" ");
277                  Serial.println(t2-t1);
278                  }
279   else {digitalWrite(RT,LOW) ;
280        Serial.println(t2-t1);
281       }
282   while (t2-t1<T) {t2=millis() ;}
283   t1=millis() ;
284 }
285 // Handlers de las interrupciones para los encoders
286
287
288 void Aencoder_chA (){if(digitalRead(ApinENC_chA)==digitalRead(ApinENC_chB)){ quadA--;}
        else{quadA++;} }
289 void Aencoder_chB (){if(digitalRead(ApinENC_chA)==digitalRead(ApinENC_chB)){ quadA++;}
        else{quadA--;} }
290
291 void Bencoder_chA (){if(digitalRead(BpinENC_chA)==digitalRead(BpinENC_chB)){ quadB--;}
        else{quadB++;} }
292 void Bencoder_chB (){if(digitalRead(BpinENC_chA)==digitalRead(BpinENC_chB)){ quadB++;}
        else{quadB--;} }
```

Código Arduino para Tm = 100 ms y N=3

```
/*
Control MultiRate para Gr = 2(s+2.5)/s
Tm = 100 ms
N = 3
*/

#define RT 13

#define CW  0
#define CCW 1

#define ApinDIR 2                 // A -> Motor Izquierda
#define ApinACC 3

#define BpinDIR 4                // B -> Motor Derecha
#define BpinACC 11

#define ApinENC_chA 42
#define ApinENC_chB 43
#define BpinENC_chA 40
#define BpinENC_chB 41

#define MINU -10.0
#define MAXU 10.0
#define MINW -15.0
#define MAXW 15.0

#define MAXZM 2.4
#define MINZM -2.4

long t1 = 0 ;
long t2 = 0 ;
long t3 = 0 ;
long t4 = 0 ;
int Tr = 100 ;                      // Periodo de muestreo RAPIDO milisegundos
int N = 3;                          // REPETICIONES
int T=N*Tr;                         // Periodo de muestreo LENTO milisegundos

float Tm = T/1000.0 ;               // Periodo de muestreo segundos
float C2D = 360.0/4480.0 ;          // Conversion cuentas a deg
float C2RS = (2*3.14/4480.0)/Tm ;   // Conversion cuentas a rad/seg


//---INICIALIZACION DE VARIABLES----------
float accA = 0.0 ;
float accB = 0.0 ;
float velA= 0.0 ;
float posA = 0.0 ;
long quadA = 2241 ;
float velB= 0.0 ;
float posB = 0.0 ;
long quadB = 2241 ;
long enc_antA = 0 ;
long enc_actA = 0 ;
int enc_difA = 0 ;
long enc_antB = 0 ;
long enc_actB = 0 ;
int enc_difB= 0 ;
long enc_antA2;
long enc_antB2;

int velIzq=0;
int velDer=0;
int i;
int k=0;
int pos;
```

```
67 bool p_leido=0;
68 float refD=0;
69 float refI=0;
70 String buffIzq = "0";
71 String buffDer="0";
72 char buffb[64] = {0};
73 String buffOUT="2000";
74
75 float A_eck=0;
76 float A_ek=0;
77 float A_ek1=0;
78 float A_ek2=0;
79 float A_y1k=0;
80 float A_y2k=0;
81 float A_y1k1=0;
82 float A_y2k1=0;
83 float A_u1k=0;
84 float A_u1k1=0;
85 float A_u1k2=0;
86 float A_u2k=0;
87 float A_u2k1=0;
88 float A_u2k2=0;
89 float A_u2kN=0;
90
91 float B_eck=0;
92 float B_ek=0;
93 float B_ek1=0;
94 float B_ek2=0;
95 float B_y1k=0;
96 float B_y2k=0;
97 float B_y1k1=0;
98 float B_y2k1=0;
99 float B_u1k=0;
100 float B_u1k1=0;
101 float B_u1k2=0;
102 float B_u2k=0;
103 float B_u2k1=0;
104 float B_u2k2=0;
105 float B_u2kN=0;
106
107 float A_U1V[3]= {0};
108 float B_U1V[3]= {0};
109
110 void setup(){
111   Serial.begin(19200);
112   Serial1.begin(19200); // Establecemos la velocidad del puerto serie (Igual que APC220)
113   pinMode(RT,OUTPUT) ;
114   Serial1.setTimeout(T/2);
115
116   attachInterrupt(ApinENC_chA,Aencoder_chA,CHANGE) ; //CONFIGURACION INTERRUPCIONES
          ENCODER
117   attachInterrupt(ApinENC_chB,Aencoder_chB,CHANGE) ;
118   attachInterrupt(BpinENC_chA,Bencoder_chA,CHANGE) ;
119   attachInterrupt(BpinENC_chB,Bencoder_chB,CHANGE) ;
120
121   analogReadResolution(8) ;
122   analogWriteResolution(8) ;
123   analogWrite(ApinACC,0) ;
124   pinMode(ApinDIR,OUTPUT ) ;
125   analogWrite(BpinACC,0) ;
126   pinMode(BpinDIR,OUTPUT ) ;
127   p_leido=0;
128   delay(1000);
129   i=0;
130 }
131
132 void loop(){
133     while (Serial1.available() >=1) {  // Lectura del puerto
134             Serial1.readBytes(buffb,8); // mientras haya informacion
```

```
135           Serial1.flush();            // disponible
136           p_leido=1;
137           t4=millis();
138           Serial.println(t4-t3);
139       }
140
141       if(Serial1.available()<0) p_leido=0;
142       refD=0;
143       refI=0;
144       buffDer="";                     // Separacion refD y refI
145       buffIzq="";
146       buffOUT="";
147       for(int k=0; k<4; k++){
148         buffDer += buffb[k+4];
149         buffIzq += buffb[k];
150       }
151       refI=buffIzq.toInt();
152       refD=buffDer.toInt();
153       refI=(refI-1000)/100-10;
154       refD=(refD-1000)/100-10;
155       if (p_leido==0){
156         refD=0;
157         refI=0;
158       }
159
160     //Calculo de error + Predictor Smith
161
162       A_eck=refI-velA;
163       B_eck=refD-velB;
164
165       A_y1k=0.7545*A_y1k1+0.5017*A_u2k1;
166       A_y2k=0.7545*A_y2k1+0.5017*A_u2kN;
167       A_y1k1=A_y1k;
168       A_y2k1=A_y2k;
169       A_ek=A_eck+A_y2k-A_y1k;
170       A_u2kN=A_u2k;
171
172       B_y1k=0.7545*B_y1k1+0.5017*B_u2k1;
173       B_y2k=0.7545*B_y2k1+0.5017*B_u2kN;
174       B_y1k1=B_y1k;
175       B_y2k1=B_y2k;
176       B_ek=B_eck+B_y2k-B_y1k;
177       B_u2kN=B_u2k;
178
179
180 // -----------------CALCULO ACCIONES DE CONTROL-------------------------
181
182 // %%%%%%%%%%%%%%%%%%%%%%%%BUCLE RAPIDO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
183       A_u1k=1.465*A_u1k1-0.4652*A_u1k2+A_ek-0.5125*A_ek1+0.01358*A_ek2;
184       B_u1k=1.465*B_u1k1-0.4652*B_u1k2+B_ek-0.5125*B_ek1+0.01358*B_ek2;
185 // %%%%%%%%%%%%%%%%%%%%%%%%BUCLE LENTO  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
186
187       //A_u2k=1.089*A_u2k1 - 0.2386*A_u2k2 + 1.347*A_u1k - 2.066*A_u1k1 + 0.7917*A_u1k1;
              // GT2 = MT/GpT
188       A_u2k=0.7511*A_u2k1 - 0.001921*A_u2k2 + 2*A_u1k - 3.009*A_u1k1 + 1.132*A_u1k1; //
              GT2 = Grd/(1+Grd*GpT)
189       A_u2k2=A_u2k1;
190       A_u2k1=A_u2k;
191       accA=A_u2k;
192      //B_u2k=1.089*B_u2k1 - 0.2386*B_u2k2 + 1.347*B_u1k - 2.066*B_u1k1 + 0.7917*B_u1k1;
             // GT2 = MT/GpT
193       B_u2k=0.7511*B_u2k1 - 0.001921*B_u2k2 + 2*B_u1k - 3.009*B_u1k1 + 1.132*B_u1k1; //
              GT2 = Grd/(1+Grd*GpT)
194       B_u2k2=B_u2k1;
195       B_u2k1=B_u2k;
196       accB=B_u2k;
197       if (accA>=MAXU) {accA=MAXU ;} ; if (accA<=MINU) {accA=MINU ;} //Saturar acciones de
              control
198           if (accB>=MAXU) {accB=MAXU ;} ; if (accB<=MINU) {accB=MINU ;}
```

```
199               if (accA>0) {
200                         digitalWrite(ApinDIR,CW) ;
201                } else {digitalWrite(ApinDIR,CCW);}
202                if (accB>0) {
203                         digitalWrite(BpinDIR,CW) ;
204                } else {digitalWrite(BpinDIR,CCW) ;}
205
206              analogWrite(ApinACC,(byte)(abs(accA)*255.0/MAXU)) ; // Aplicar acciones de
                    control
207              analogWrite(BpinACC,(byte)(abs(accB)*255.0/MAXU)) ;
208       t2=millis();
209
210       if (t2-t1>Tr) {digitalWrite(RT,HIGH); //Espera hasta alcanzar T
211               // Serial1.print(" ");
212                Serial.println(t2-t1);
213                }
214       else {digitalWrite(RT,LOW) ;
215            Serial.println(t2-t1);
216       }
217      //A_u2k=1.089*A_u2k1 - 0.2386*A_u2k2 + 1.347*A_u1k - 2.066*A_u1k + 0.7917*A_u1k1;
             // GT2 = MT/GpT
218      A_u2k=0.7511*A_u2k1 - 0.001921*A_u2k2 + 2*A_u1k - 3.009*A_u1k + 1.132*A_u1k1; //
             GT2 = Grd/(1+Grd*GpT)
219      A_u2k2=A_u2k1;
220      A_u2k1=A_u2k;
221      accA=A_u2k;
222     //B_u2k=1.089*B_u2k1 - 0.2386*B_u2k2 + 1.347*B_u1k - 2.066*B_u1k + 0.7917*B_u1k1; //
             GT2 = MT/GpT
223      B_u2k=0.7511*B_u2k1 - 0.001921*B_u2k2 + 2*B_u1k - 3.009*B_u1k + 1.132*B_u1k1; //
             GT2 = Grd/(1+Grd*GpT)
224      B_u2k2=B_u2k1;
225      B_u2k1=B_u2k;
226      accB=B_u2k;
227
228      if (accA>=MAXU) {accA=MAXU ;} ; if (accA<=MINU) {accA=MINU ;} //Saturar acciones de
              control
229              if (accB>=MAXU) {accB=MAXU ;} ; if (accB<=MINU) {accB=MINU ;}
230              if (accA>0) {
231                         digitalWrite(ApinDIR,CW) ;
232                } else {digitalWrite(ApinDIR,CCW);}
233                if (accB>0) {
234                         digitalWrite(BpinDIR,CW) ;
235                } else {digitalWrite(BpinDIR,CCW) ;}
236
237              analogWrite(ApinACC,(byte)(abs(accA)*255.0/MAXU)) ; // Aplicar acciones de
                    control
238              analogWrite(BpinACC,(byte)(abs(accB)*255.0/MAXU)) ;
239
240       t2=millis();
241
242       if (t2-t1>Tr) {digitalWrite(RT,HIGH); //Espera hasta alcanzar T
243               // Serial1.print(" ");
244                Serial.println(t2-t1);
245                }
246       else {digitalWrite(RT,LOW) ;
247            Serial.println(t2-t1);
248       }
249      //A_u2k=1.089*A_u2k1 - 0.2386*A_u2k2 + 1.347*A_u1k - 2.066*A_u1k + 0.7917*A_u1k; //
             GT2 = MT/GpT
250      A_u2k=0.7511*A_u2k1 - 0.001921*A_u2k2 + 2*A_u1k - 3.009*A_u1k + 1.132*A_u1k; // GT2
             = Grd/(1+Grd*GpT)
251      A_u2k2=A_u2k1;
252      A_u2k1=A_u2k;
253      accA=A_u2k;
254     //B_u2k=1.089*B_u2k1 - 0.2386*B_u2k2 + 1.347*B_u1k - 2.066*B_u1k + 0.7917*B_u1k; //
            GT2 = MT/GpT
255      B_u2k=0.7511*B_u2k1 - 0.001921*B_u2k2 + 2*B_u1k - 3.009*B_u1k + 1.132*B_u1k; // GT2
             = Grd/(1+Grd*GpT)
256      B_u2k2=B_u2k1;
```

```
257       B_u2k1=B_u2k;
258       accB=B_u2k;
259
260       if (accA>=MAXU) {accA=MAXU ;} ; if (accA<=MINU) {accA=MINU ;} //Saturar acciones de
              control
261                 if (accB>=MAXU) {accB=MAXU ;} ; if (accB<=MINU) {accB=MINU ;}
262                 if (accA>0) {
263                           digitalWrite(ApinDIR,CW) ;
264                  } else {digitalWrite(ApinDIR,CCW);}
265                  if (accB>0) {
266                           digitalWrite(BpinDIR,CW) ;
267                  } else {digitalWrite(BpinDIR,CCW) ;}
268
269                analogWrite(ApinACC,(byte)(abs(accA)*255.0/MAXU)) ; // Aplicar acciones de
                      control
270                analogWrite(BpinACC,(byte)(abs(accB)*255.0/MAXU)) ;
271
272        // Actualizar variables
273        A_u1k2=A_u1k1;
274        A_u1k1=A_u1k;
275        A_ek2=A_ek1;
276        A_ek1=A_ek;
277
278        B_u1k2=B_u1k1;
279        B_u1k1=B_u1k;
280        B_ek2=B_ek1;
281        B_ek1=B_ek;
282
283       // Lectura y calculo de velocidades
284       enc_actA=quadA;
285       enc_difA=enc_actA-enc_antA ;
286       posA=(enc_actA)*C2D ; //pos=((int)pos % 360)-180.0 ;
287       velA=(enc_actA-enc_antA)*C2RS ;
288       enc_antA2=enc_antA;
289       enc_antA=enc_actA ;
290
291       enc_actB=quadB ;
292       enc_difB=enc_actB-enc_antB ;
293       posB=(enc_actB)*C2D ; //pos=((int)pos % 360)-180.0 ;
294       velB=(enc_actB-enc_antB)*C2RS ;
295       enc_antB2=enc_antB;
296       enc_antB=enc_actB ;
297
298        // Envio velocidades
299       velIzq=(velB+10)*100+1000;
300       velDer=(velA+10)*100+1000;
301       buffOUT+=velDer;
302       buffOUT+=velIzq;
303       Serial1.print(buffOUT);
304       t3=millis();
305       t2=millis();
306
307   if (t2-t1>T) {digitalWrite(RT,HIGH); // Espera hasta NT
308                 // Serial1.print(" ");
309                  Serial.println(t2-t1);
310                  }
311   else {digitalWrite(RT,LOW) ;
312        Serial.println(t2-t1);
313       }
314   while (t2-t1<T) {t2=millis() ;}
315   t1=millis() ;
316 }
317
318 // Handlers de las interrupciones para los encoders
319
320 void Aencoder_chA (){if(digitalRead(ApinENC_chA)==digitalRead(ApinENC_chB)){ quadA--;}
        else{quadA++;} }
321 void Aencoder_chB (){if(digitalRead(ApinENC_chA)==digitalRead(ApinENC_chB)){ quadA++;}
        else{quadA--;} }
```

```
322
323 void Bencoder_chA (){if(digitalRead(BpinENC_chA)==digitalRead(BpinENC_chB)){ quadB--;}
        else{quadB++;} }
324 void Bencoder_chB (){if(digitalRead(BpinENC_chA)==digitalRead(BpinENC_chB)){ quadB++;}
        else{quadB--;} }
```

Código Arduino para Tm = 100 ms y N=4

```
/*
Control MultiRate para Gr = 2(s+2.5)/s
Tm = 100 ms
N = 4
*/

#define RT 13

#define CW  0
#define CCW 1

#define ApinDIR 2                 // A -> Motor Izquierda
#define ApinACC 3

#define BpinDIR 4                // B -> Motor Derecha
#define BpinACC 11

#define ApinENC_chA 42
#define ApinENC_chB 43
#define BpinENC_chA 40
#define BpinENC_chB 41

#define MINU -10.0
#define MAXU 10.0
#define MINW -15.0
#define MAXW 15.0

#define MAXZM 2.4
#define MINZM -2.4

long t1 = 0 ;
long t2 = 0 ;
long t3 = 0 ;
long t4 = 0 ;
int Tr = 100 ;                      // Periodo de muestreo RAPIDO milisegundos
int N = 4;                          // REPETICIONES
int T=N*Tr;                         // Periodo de muestreo LENTO milisegundos

float Tm = T/1000.0 ;               // Periodo de muestreo segundos
float C2D = 360.0/4480.0 ;          // Conversion cuentas a deg
float C2RS = (2*3.14/4480.0)/Tm ;   // Conversion cuentas a rad/seg

//---INICIALIZACION DE VARIABLES----------
float accA = 0.0 ;
float accB = 0.0 ;
float velA= 0.0 ;
float posA = 0.0 ;
long quadA = 2241 ;
float velB= 0.0 ;
float posB = 0.0 ;
long quadB = 2241 ;
long enc_antA = 0 ;
long enc_actA = 0 ;
int enc_difA = 0 ;
long enc_antB = 0 ;
long enc_actB = 0 ;
int enc_difB= 0 ;
long enc_antA2;
long enc_antB2;

int velIzq=0;
int velDer=0;
int i;
int k=0;
int pos;
bool p_leido=0;
```

```
67 float refD=0;
68 float refI=0;
69 String buffIzq = "0";
70 String buffDer="0";
71 char buffb[64] = {0};
72 String buffOUT="2000";
73
74 float A_eck=0;
75 float A_ek=0;
76 float A_ek1=0;
77 float A_ek2=0;
78 float A_y1k=0;
79 float A_y2k=0;
80 float A_y1k1=0;
81 float A_y2k1=0;
82 float A_u1k=0;
83 float A_u1k1=0;
84 float A_u1k2=0;
85 float A_u2k=0;
86 float A_u2k1=0;
87 float A_u2k2=0;
88 float A_u2kN=0;
89
90 float B_eck=0;
91 float B_ek=0;
92 float B_ek1=0;
93 float B_ek2=0;
94 float B_y1k=0;
95 float B_y2k=0;
96 float B_y1k1=0;
97 float B_y2k1=0;
98 float B_u1k=0;
99 float B_u1k1=0;
100 float B_u1k2=0;
101 float B_u2k=0;
102 float B_u2k1=0;
103 float B_u2k2=0;
104 float B_u2kN=0;
105
106 float A_U1V[3]= {0};
107 float B_U1V[3]= {0};
108
109 void setup(){
110   Serial.begin(19200);
111   Serial1.begin(19200); // Establecemos la velocidad del puerto serie (Igual que APC220)
112   pinMode(RT,OUTPUT) ;
113   Serial1.setTimeout(T/2);
114
115   attachInterrupt(ApinENC_chA,Aencoder_chA,CHANGE) ;   //CONFIGURACION INTERRUPCIONES
          ENCODER
116   attachInterrupt(ApinENC_chB,Aencoder_chB,CHANGE) ;
117   attachInterrupt(BpinENC_chA,Bencoder_chA,CHANGE) ;
118   attachInterrupt(BpinENC_chB,Bencoder_chB,CHANGE) ;
119
120   analogReadResolution(8) ;
121   analogWriteResolution(8) ;
122   analogWrite(ApinACC,0) ;
123   pinMode(ApinDIR,OUTPUT ) ;
124   analogWrite(BpinACC,0) ;
125   pinMode(BpinDIR,OUTPUT ) ;
126   p_leido=0;
127   delay(1000);
128   i=0;
129 }
130
131 void loop(){
132     while (Serial1.available() >=1) {  // Lectura del puerto
133             Serial1.readBytes(buffb,8); // mientras haya informacion
134             Serial1.flush();            // disponible
```

```
135           p_leido=1;
136           t4=millis();
137           Serial.println(t4-t3);
138       }
139
140      if(Serial1.available()<0) p_leido=0;
141       //if (p_leido==0){
142         refD=0;
143         refI=0;
144       //}
145     //if(p_leido==1){
146           buffDer="";                   // Separacion refD y refI
147           buffIzq="";
148           buffOUT="";
149          for(int k=0; k<4; k++){
150            buffDer += buffb[k+4];
151            buffIzq += buffb[k];
152          }
153          refI=buffIzq.toInt();
154          refD=buffDer.toInt();
155          refI=(refI-1000)/100-10;
156          refD=(refD-1000)/100-10;
157          if (p_leido==0){
158            refD=0;
159           refI=0;
160          }
161
162        //Calculo de error + Predictor Smith
163       A_eck=refI-velA;
164       B_eck=refD-velB;
165
166       A_y1k=0.7545*A_y1k1+0.5017*A_u2k1;
167       A_y2k=0.7545*A_y2k1+0.5017*A_u2kN;
168       A_y1k1=A_y1k;
169       A_y2k1=A_y2k;
170       A_ek=A_eck+A_y2k-A_y1k;
171       A_u2kN=A_u2k;
172
173       B_y1k=0.7545*B_y1k1+0.5017*B_u2k1;
174       B_y2k=0.7545*B_y2k1+0.5017*B_u2kN;
175       B_y1k1=B_y1k;
176       B_y2k1=B_y2k;
177       B_ek=B_eck+B_y2k-B_y1k;
178       B_u2kN=B_u2k;
179
180   // -----------------CALCULO ACCIONES DE CONTROL-------------------------
181
182  // %%%%%%%%%%%%%%%%%%%%%%%%BUCLE RAPIDO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
183       A_u1k=1.365*A_u1k1-0.3648*A_u1k2+A_ek-0.389*A_ek1+0.00239*A_ek2;
184       B_u1k=1.365*B_u1k1-0.3648*B_u1k2+B_ek-0.389*B_ek1+0.00239*B_ek2;
185  // %%%%%%%%%%%%%%%%%%%%%%%%BUCLE LENTO  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
186
187       //A_u2k=1.089*A_u2k1 - 0.2386*A_u2k2 + 1.347*A_u1k - 2.066*A_u1k1 + 0.7917*A_u1k1;
              // GT2 = MT/GpT
188       A_u2k=0.7511*A_u2k1 - 0.001921*A_u2k2 + 2*A_u1k - 3.009*A_u1k1 + 1.132*A_u1k1; //
              GT2 = Grd/(1+Grd*GpT)
189       A_u2k2=A_u2k1;
190       A_u2k1=A_u2k;
191       accA=A_u2k;
192      //B_u2k=1.089*B_u2k1 - 0.2386*B_u2k2 + 1.347*B_u1k - 2.066*B_u1k1 + 0.7917*B_u1k1;
             // GT2 = MT/GpT
193       B_u2k=0.7511*B_u2k1 - 0.001921*B_u2k2 + 2*B_u1k - 3.009*B_u1k1 + 1.132*B_u1k1; //
              GT2 = Grd/(1+Grd*GpT)
194       B_u2k2=B_u2k1;
195       B_u2k1=B_u2k;
196       accB=B_u2k;
197       if (accA>=MAXU) {accA=MAXU ;} ; if (accA<=MINU) {accA=MINU ;} //Saturar acciones de
               control
198                 if (accB>=MAXU) {accB=MAXU ;} ; if (accB<=MINU) {accB=MINU ;}
```

```
199               if (accA>0) {
200                         digitalWrite(ApinDIR,CW) ;
201                } else {digitalWrite(ApinDIR,CCW);}
202                if (accB>0) {
203                         digitalWrite(BpinDIR,CW) ;
204                } else {digitalWrite(BpinDIR,CCW) ;}
205
206              analogWrite(ApinACC,(byte)(abs(accA)*255.0/MAXU)) ; // Aplicar acciones de
                    control
207              analogWrite(BpinACC,(byte)(abs(accB)*255.0/MAXU)) ;
208       t2=millis();
209
210       if (t2-t1>Tr) {digitalWrite(RT,HIGH); //Espera hasta alcanzar T
211               // Serial1.print(" ");
212                Serial.println(t2-t1);
213                }
214       else {digitalWrite(RT,LOW) ;
215            Serial.println(t2-t1);
216       }
217      //A_u2k=1.089*A_u2k1 - 0.2386*A_u2k2 + 1.347*A_u1k - 2.066*A_u1k + 0.7917*A_u1k1;
             // GT2 = MT/GpT
218      A_u2k=0.7511*A_u2k1 - 0.001921*A_u2k2 + 2*A_u1k - 3.009*A_u1k + 1.132*A_u1k1; //
             GT2 = Grd/(1+Grd*GpT)
219      A_u2k2=A_u2k1;
220      A_u2k1=A_u2k;
221      accA=A_u2k;
222     //B_u2k=1.089*B_u2k1 - 0.2386*B_u2k2 + 1.347*B_u1k - 2.066*B_u1k + 0.7917*B_u1k1; //
             GT2 = MT/GpT
223      B_u2k=0.7511*B_u2k1 - 0.001921*B_u2k2 + 2*B_u1k - 3.009*B_u1k + 1.132*B_u1k1; //
             GT2 = Grd/(1+Grd*GpT)
224      B_u2k2=B_u2k1;
225      B_u2k1=B_u2k;
226      accB=B_u2k;
227
228      if (accA>=MAXU) {accA=MAXU ;} ; if (accA<=MINU) {accA=MINU ;} //Saturar acciones de
              control
229              if (accB>=MAXU) {accB=MAXU ;} ; if (accB<=MINU) {accB=MINU ;}
230              if (accA>0) {
231                         digitalWrite(ApinDIR,CW) ;
232                } else {digitalWrite(ApinDIR,CCW);}
233                if (accB>0) {
234                         digitalWrite(BpinDIR,CW) ;
235                } else {digitalWrite(BpinDIR,CCW) ;}
236
237              analogWrite(ApinACC,(byte)(abs(accA)*255.0/MAXU)) ; // Aplicar acciones de
                    control
238              analogWrite(BpinACC,(byte)(abs(accB)*255.0/MAXU)) ;
239
240       t2=millis();
241
242       if (t2-t1>Tr) {digitalWrite(RT,HIGH); //Espera hasta alcanzar T
243               // Serial1.print(" ");
244                Serial.println(t2-t1);
245                }
246       else {digitalWrite(RT,LOW) ;
247            Serial.println(t2-t1);
248       }
249      //A_u2k=1.089*A_u2k1 - 0.2386*A_u2k2 + 1.347*A_u1k - 2.066*A_u1k + 0.7917*A_u1k; //
             GT2 = MT/GpT
250      A_u2k=0.7511*A_u2k1 - 0.001921*A_u2k2 + 2*A_u1k - 3.009*A_u1k + 1.132*A_u1k; // GT2
             = Grd/(1+Grd*GpT)
251      A_u2k2=A_u2k1;
252      A_u2k1=A_u2k;
253      accA=A_u2k;
254     //B_u2k=1.089*B_u2k1 - 0.2386*B_u2k2 + 1.347*B_u1k - 2.066*B_u1k + 0.7917*B_u1k; //
            GT2 = MT/GpT
255      B_u2k=0.7511*B_u2k1 - 0.001921*B_u2k2 + 2*B_u1k - 3.009*B_u1k + 1.132*B_u1k; // GT2
             = Grd/(1+Grd*GpT)
256      B_u2k2=B_u2k1;
```

```
257       B_u2k1=B_u2k;
258       accB=B_u2k;
259
260       if (accA>=MAXU) {accA=MAXU ;} ; if (accA<=MINU) {accA=MINU ;} //Saturar acciones de
              control
261                 if (accB>=MAXU) {accB=MAXU ;} ; if (accB<=MINU) {accB=MINU ;}
262                 if (accA>0) {
263                           digitalWrite(ApinDIR,CW) ;
264                  } else {digitalWrite(ApinDIR,CCW);}
265                  if (accB>0) {
266                           digitalWrite(BpinDIR,CW) ;
267                  } else {digitalWrite(BpinDIR,CCW) ;}
268
269                analogWrite(ApinACC,(byte)(abs(accA)*255.0/MAXU)) ; // Aplicar acciones de
                      control
270                analogWrite(BpinACC,(byte)(abs(accB)*255.0/MAXU)) ;
271        t2=millis();
272
273        if (t2-t1>Tr) {digitalWrite(RT,HIGH); //Espera hasta alcanzar T
274                 // Serial1.print(" ");
275                  Serial.println(t2-t1);
276                  }
277        else {digitalWrite(RT,LOW) ;
278             Serial.println(t2-t1);
279        }
280       //A_u2k=1.089*A_u2k1 - 0.2386*A_u2k2 + 1.347*A_u1k - 2.066*A_u1k + 0.7917*A_u1k; //
              GT2 = MT/GpT
281       A_u2k=0.7511*A_u2k1 - 0.001921*A_u2k2 + 2*A_u1k - 3.009*A_u1k + 1.132*A_u1k; // GT2
              = Grd/(1+Grd*GpT)
282       A_u2k2=A_u2k1;
283       A_u2k1=A_u2k;
284       accA=A_u2k;
285      //B_u2k=1.089*B_u2k1 - 0.2386*B_u2k2 + 1.347*B_u1k - 2.066*B_u1k + 0.7917*B_u1k; //
             GT2 = MT/GpT
286       B_u2k=0.7511*B_u2k1 - 0.001921*B_u2k2 + 2*B_u1k - 3.009*B_u1k + 1.132*B_u1k; // GT2
              = Grd/(1+Grd*GpT)
287       B_u2k2=B_u2k1;
288       B_u2k1=B_u2k;
289       accB=B_u2k;
290
291       if (accA>=MAXU) {accA=MAXU ;} ; if (accA<=MINU) {accA=MINU ;} //Saturar acciones de
              control
292                 if (accB>=MAXU) {accB=MAXU ;} ; if (accB<=MINU) {accB=MINU ;}
293                 if (accA>0) {
294                           digitalWrite(ApinDIR,CW) ;
295                  } else {digitalWrite(ApinDIR,CCW);}
296                  if (accB>0) {
297                           digitalWrite(BpinDIR,CW) ;
298                  } else {digitalWrite(BpinDIR,CCW) ;}
299
300                analogWrite(ApinACC,(byte)(abs(accA)*255.0/MAXU)) ; // Aplicar acciones de
                      control
301                analogWrite(BpinACC,(byte)(abs(accB)*255.0/MAXU)) ;
302
303         // Actualizar variables
304         A_u1k2=A_u1k1;
305        A_u1k1=A_u1k;
306        A_ek2=A_ek1;
307        A_ek1=A_ek;
308
309        B_u1k2=B_u1k1;
310        B_u1k1=B_u1k;
311        B_ek2=B_ek1;
312        B_ek1=B_ek;
313
314       // Lectura y calculo de velocidades
315       enc_actA=quadA;
316       enc_difA=enc_actA-enc_antA ;
317       posA=(enc_actA)*C2D ; //pos=((int)pos % 360)-180.0 ;
```

```
318       velA=(enc_actA-enc_antA)*C2RS ;
319       enc_antA2=enc_antA;
320       enc_antA=enc_actA ;
321 
322       enc_actB=quadB ;
323       enc_difB=enc_actB-enc_antB ;
324       posB=(enc_actB)*C2D ; //pos=((int)pos % 360)-180.0 ;
325       velB=(enc_actB-enc_antB)*C2RS ;
326       enc_antB2=enc_antB;
327       enc_antB=enc_actB ;
328 
329       // Envio velocidades
330       velIzq=(velB+10)*100+1000;
331       velDer=(velA+10)*100+1000;
332       buffOUT+=velDer;
333       buffOUT+=velIzq;
334       Serial1.print(buffOUT);
335       t3=millis();
336       t2=millis();
337 
338   if (t2-t1>T) {digitalWrite(RT,HIGH); // Espera hasta NT
339                 // Serial1.print(" ");
340                  Serial.println(t2-t1);
341                  }
342   else {digitalWrite(RT,LOW) ;
343         Serial.println(t2-t1);
344        }
345   while (t2-t1<T) {t2=millis() ;}
346   t1=millis() ;
347 }
348 
349 // Handlers de las interrupciones para los encoders
350 
351 void Aencoder_chA (){if(digitalRead(ApinENC_chA)==digitalRead(ApinENC_chB)){ quadA--;}
      else{quadA++;} }
352 void Aencoder_chB (){if(digitalRead(ApinENC_chA)==digitalRead(ApinENC_chB)){ quadA++;}
      else{quadA--;} }
353 
354 void Bencoder_chA (){if(digitalRead(BpinENC_chA)==digitalRead(BpinENC_chB)){ quadB--;}
      else{quadB++;} }
355 void Bencoder_chB (){if(digitalRead(BpinENC_chA)==digitalRead(BpinENC_chB)){ quadB++;}
      else{quadB--;} }
```

Código Arduino para Tm = 100 ms y N=5

```
/*
Control MultiRate para Gr = 2(s+2.5)/s
Tm = 100 ms
N = 5
*/

#define RT 13

#define CW  0
#define CCW 1

#define ApinDIR 2                 // A -> Motor Izquierda
#define ApinACC 3

#define BpinDIR 4                // B -> Motor Derecha
#define BpinACC 11

#define ApinENC_chA 42
#define ApinENC_chB 43
#define BpinENC_chA 40
#define BpinENC_chB 41

#define MINU -10.0
#define MAXU 10.0
#define MINW -15.0
#define MAXW 15.0

#define MAXZM 2.4
#define MINZM -2.4

long t1 = 0 ;
long t2 = 0 ;
long t3 = 0 ;
long t4 = 0 ;
int Tr = 100 ;                      // Periodo de muestreo RAPIDO milisegundos
int N = 5;                           // REPETICIONES
int T=N*Tr;                          // Periodo de muestreo LENTO milisegundos

float Tm = T/1000.0 ;              // Periodo de muestreo segundos
float C2D = 360.0/4480.0 ;         // Conversion cuentas a deg
float C2RS = (2*3.14/4480.0)/Tm ;  // Conversion cuentas a rad/seg

//---INICIALIZACION DE VARIABLES----------
float accA = 0.0 ;
float accB = 0.0 ;
float velA= 0.0 ;
float posA = 0.0 ;
long quadA = 2241 ;
float velB= 0.0 ;
float posB = 0.0 ;
long quadB = 2241 ;
long enc_antA = 0 ;
long enc_actA = 0 ;
int enc_difA = 0 ;
long enc_antB = 0 ;
long enc_actB = 0 ;
int enc_difB= 0 ;
long enc_antA2;
long enc_antB2;

int velIzq=0;
int velDer=0;
int i;
int k=0;
int pos;
bool p_leido=0;
```

```
67 float refD=0;
68 float refI=0;
69 String buffIzq = "0";
70 String buffDer="0";
71 char buffb[64] = {0};
72 String buffOUT="2000";
73
74 float A_eck=0;
75 float A_ek=0;
76 float A_ek1=0;
77 float A_ek2=0;
78 float A_y1k=0;
79 float A_y2k=0;
80 float A_y1k1=0;
81 float A_y2k1=0;
82 float A_u1k=0;
83 float A_u1k1=0;
84 float A_u1k2=0;
85 float A_u2k=0;
86 float A_u2k1=0;
87 float A_u2k2=0;
88 float A_u2kN=0;
89
90 float B_eck=0;
91 float B_ek=0;
92 float B_ek1=0;
93 float B_ek2=0;
94 float B_y1k=0;
95 float B_y2k=0;
96 float B_y1k1=0;
97 float B_y2k1=0;
98 float B_u1k=0;
99 float B_u1k1=0;
100 float B_u1k2=0;
101 float B_u2k=0;
102 float B_u2k1=0;
103 float B_u2k2=0;
104 float B_u2kN=0;
105
106 float A_U1V[3]= {0};
107 float B_U1V[3]= {0};
108
109 void setup(){
110   Serial.begin(19200);
111   Serial1.begin(19200); // Establecemos la velocidad del puerto serie (Igual que APC220)
112   pinMode(RT,OUTPUT) ;
113   Serial1.setTimeout(T/2);
114
115   attachInterrupt(ApinENC_chA,Aencoder_chA,CHANGE) ; //CONFIGURACION INTERRUPCIONES
          ENCODER
116   attachInterrupt(ApinENC_chB,Aencoder_chB,CHANGE) ;
117   attachInterrupt(BpinENC_chA,Bencoder_chA,CHANGE) ;
118   attachInterrupt(BpinENC_chB,Bencoder_chB,CHANGE) ;
119
120   analogReadResolution(8) ;
121   analogWriteResolution(8) ;
122   analogWrite(ApinACC,0) ;
123   pinMode(ApinDIR,OUTPUT ) ;
124   analogWrite(BpinACC,0) ;
125   pinMode(BpinDIR,OUTPUT ) ;
126   p_leido=0;
127   delay(1000);
128   i=0;
129 }
130
131 void loop(){
132     while (Serial1.available() >=1) {  // Lectura del puerto
133             Serial1.readBytes(buffb,8); // mientras haya informacion
134             Serial1.flush();            // disponible
```

```
135           p_leido=1;
136           t4=millis();
137           Serial.println(t4-t3);
138       }
139
140     if(Serial1.available()<0) p_leido=0;
141     refD=0;
142     refI=0;
143     buffDer="";                          // Separacion refD y refI
144     buffIzq="";
145     buffOUT="";
146     for(int k=0; k<4; k++){
147       buffDer += buffb[k+4];
148       buffIzq += buffb[k];
149     }
150     refI=buffIzq.toInt();
151     refD=buffDer.toInt();
152     refI=(refI-1000)/100-10;
153     refD=(refD-1000)/100-10;
154     if (p_leido==0){
155       refD=0;
156       refI=0;
157     }
158
159     //Calculo de error + Predictor Smith
160
161       A_eck=refI-velA;
162       B_eck=refD-velB;
163
164       A_y1k=0.7545*A_y1k1+0.5017*A_u2k1;
165       A_y2k=0.7545*A_y2k1+0.5017*A_u2kN;
166       A_y1k1=A_y1k;
167       A_y2k1=A_y2k;
168       A_ek=A_eck+A_y2k-A_y1k;
169       A_u2kN=A_u2k;
170
171       B_y1k=0.7545*B_y1k1+0.5017*B_u2k1;
172       B_y2k=0.7545*B_y2k1+0.5017*B_u2kN;
173       B_y1k1=B_y1k;
174       B_y2k1=B_y2k;
175       B_ek=B_eck+B_y2k-B_y1k;
176       B_u2kN=B_u2k;
177
178 // -----------------CALCULO ACCIONES DE CONTROL-------------------------
179
180 // %%%%%%%%%%%%%%%%%%%%%%%%BUCLE RAPIDO %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
181       A_u1k=1.286*A_u1k1-0.2863*A_u1k2+A_ek-0.3014*A_ek1+0.0007728*A_ek2;
182       B_u1k=1.286*B_u1k1-0.2863*B_u1k2+B_ek-0.3014*B_ek1+0.0007728*B_ek2;
183 // %%%%%%%%%%%%%%%%%%%%%%%%BUCLE LENTO  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
184
185       //A_u2k=1.089*A_u2k1 - 0.2386*A_u2k2 + 1.347*A_u1k - 2.066*A_u1k1 + 0.7917*A_u1k1;
              // GT2 = MT/GpT
186       A_u2k=0.7511*A_u2k1 - 0.001921*A_u2k2 + 2*A_u1k - 3.009*A_u1k1 + 1.132*A_u1k1; //
              GT2 = Grd/(1+Grd*GpT)
187       A_u2k2=A_u2k1;
188       A_u2k1=A_u2k;
189       accA=A_u2k;
190      //B_u2k=1.089*B_u2k1 - 0.2386*B_u2k2 + 1.347*B_u1k - 2.066*B_u1k1 + 0.7917*B_u1k1;
             // GT2 = MT/GpT
191       B_u2k=0.7511*B_u2k1 - 0.001921*B_u2k2 + 2*B_u1k - 3.009*B_u1k1 + 1.132*B_u1k1; //
              GT2 = Grd/(1+Grd*GpT)
192       B_u2k2=B_u2k1;
193       B_u2k1=B_u2k;
194       accB=B_u2k;
195       if (accA>=MAXU) {accA=MAXU ;} ; if (accA<=MINU) {accA=MINU ;} //Saturar acciones de
              control
196               if (accB>=MAXU) {accB=MAXU ;} ; if (accB<=MINU) {accB=MINU ;}
197               if (accA>0) {
198                         digitalWrite(ApinDIR,CW) ;
```

```
199                 } else {digitalWrite(ApinDIR,CCW);}
200                 if (accB>0) {
201                           digitalWrite(BpinDIR,CW) ;
202                 } else {digitalWrite(BpinDIR,CCW) ;}
203
204               analogWrite(ApinACC,(byte)(abs(accA)*255.0/MAXU)) ; // Aplicar acciones de
                      control
205               analogWrite(BpinACC,(byte)(abs(accB)*255.0/MAXU)) ;
206        t2=millis();
207
208        if (t2-t1>Tr) {digitalWrite(RT,HIGH); //Espera hasta alcanzar T
209                 // Serial1.print(" ");
210                  Serial.println(t2-t1);
211                  }
212        else {digitalWrite(RT,LOW) ;
213              Serial.println(t2-t1);
214        }
215       //A_u2k=1.089*A_u2k1 - 0.2386*A_u2k2 + 1.347*A_u1k - 2.066*A_u1k + 0.7917*A_u1k1;
              // GT2 = MT/GpT
216       A_u2k=0.7511*A_u2k1 - 0.001921*A_u2k2 + 2*A_u1k - 3.009*A_u1k + 1.132*A_u1k1; //
              GT2 = Grd/(1+Grd*GpT)
217       A_u2k2=A_u2k1;
218       A_u2k1=A_u2k;
219       accA=A_u2k;
220      //B_u2k=1.089*B_u2k1 - 0.2386*B_u2k2 + 1.347*B_u1k - 2.066*B_u1k + 0.7917*B_u1k1; //
              GT2 = MT/GpT
221       B_u2k=0.7511*B_u2k1 - 0.001921*B_u2k2 + 2*B_u1k - 3.009*B_u1k + 1.132*B_u1k1; //
              GT2 = Grd/(1+Grd*GpT)
222       B_u2k2=B_u2k1;
223       B_u2k1=B_u2k;
224       accB=B_u2k;
225
226       if (accA>=MAXU) {accA=MAXU ;} ; if (accA<=MINU) {accA=MINU ;} //Saturar acciones de
               control
227                 if (accB>=MAXU) {accB=MAXU ;} ; if (accB<=MINU) {accB=MINU ;}
228                 if (accA>0) {
229                           digitalWrite(ApinDIR,CW) ;
230                  } else {digitalWrite(ApinDIR,CCW);}
231                  if (accB>0) {
232                           digitalWrite(BpinDIR,CW) ;
233                  } else {digitalWrite(BpinDIR,CCW) ;}
234
235               analogWrite(ApinACC,(byte)(abs(accA)*255.0/MAXU)) ; // Aplicar acciones de
                      control
236               analogWrite(BpinACC,(byte)(abs(accB)*255.0/MAXU)) ;
237
238        t2=millis();
239
240        if (t2-t1>Tr) {digitalWrite(RT,HIGH); //Espera hasta alcanzar T
241                 // Serial1.print(" ");
242                  Serial.println(t2-t1);
243                  }
244        else {digitalWrite(RT,LOW) ;
245              Serial.println(t2-t1);
246        }
247       //A_u2k=1.089*A_u2k1 - 0.2386*A_u2k2 + 1.347*A_u1k - 2.066*A_u1k + 0.7917*A_u1k; //
              GT2 = MT/GpT
248       A_u2k=0.7511*A_u2k1 - 0.001921*A_u2k2 + 2*A_u1k - 3.009*A_u1k + 1.132*A_u1k; // GT2
              = Grd/(1+Grd*GpT)
249       A_u2k2=A_u2k1;
250       A_u2k1=A_u2k;
251       accA=A_u2k;
252      //B_u2k=1.089*B_u2k1 - 0.2386*B_u2k2 + 1.347*B_u1k - 2.066*B_u1k + 0.7917*B_u1k; //
             GT2 = MT/GpT
253       B_u2k=0.7511*B_u2k1 - 0.001921*B_u2k2 + 2*B_u1k - 3.009*B_u1k + 1.132*B_u1k; // GT2
              = Grd/(1+Grd*GpT)
254       B_u2k2=B_u2k1;
255       B_u2k1=B_u2k;
256       accB=B_u2k;
```

```
257
258       if (accA>=MAXU) {accA=MAXU ;} ; if (accA<=MINU) {accA=MINU ;} //Saturar acciones de
              control
259                   if (accB>=MAXU) {accB=MAXU ;} ; if (accB<=MINU) {accB=MINU ;}
260                   if (accA>0) {
261                                 digitalWrite(ApinDIR,CW) ;
262                    } else {digitalWrite(ApinDIR,CCW);}
263                    if (accB>0) {
264                                 digitalWrite(BpinDIR,CW) ;
265                    } else {digitalWrite(BpinDIR,CCW) ;}
266
267                  analogWrite(ApinACC,(byte)(abs(accA)*255.0/MAXU)) ; // Aplicar acciones de
                        control
268                  analogWrite(BpinACC,(byte)(abs(accB)*255.0/MAXU)) ;
269        t2=millis();
270
271        if (t2-t1>Tr) {digitalWrite(RT,HIGH); //Espera hasta alcanzar T
272                 // Serial1.print(" ");
273                  Serial.println(t2-t1);
274                  }
275        else {digitalWrite(RT,LOW) ;
276              Serial.println(t2-t1);
277        }
278       //A_u2k=1.089*A_u2k1 - 0.2386*A_u2k2 + 1.347*A_u1k - 2.066*A_u1k + 0.7917*A_u1k; //
              GT2 = MT/GpT
279       A_u2k=0.7511*A_u2k1 - 0.001921*A_u2k2 + 2*A_u1k - 3.009*A_u1k + 1.132*A_u1k; // GT2
              = Grd/(1+Grd*GpT)
280       A_u2k2=A_u2k1;
281       A_u2k1=A_u2k;
282       accA=A_u2k;
283      //B_u2k=1.089*B_u2k1 - 0.2386*B_u2k2 + 1.347*B_u1k - 2.066*B_u1k + 0.7917*B_u1k; //
             GT2 = MT/GpT
284       B_u2k=0.7511*B_u2k1 - 0.001921*B_u2k2 + 2*B_u1k - 3.009*B_u1k + 1.132*B_u1k; // GT2
              = Grd/(1+Grd*GpT)
285       B_u2k2=B_u2k1;
286       B_u2k1=B_u2k;
287       accB=B_u2k;
288
289       if (accA>=MAXU) {accA=MAXU ;} ; if (accA<=MINU) {accA=MINU ;} //Saturar acciones de
              control
290                   if (accB>=MAXU) {accB=MAXU ;} ; if (accB<=MINU) {accB=MINU ;}
291                   if (accA>0) {
292                                 digitalWrite(ApinDIR,CW) ;
293                    } else {digitalWrite(ApinDIR,CCW);}
294                    if (accB>0) {
295                                 digitalWrite(BpinDIR,CW) ;
296                    } else {digitalWrite(BpinDIR,CCW) ;}
297
298                  analogWrite(ApinACC,(byte)(abs(accA)*255.0/MAXU)) ; // Aplicar acciones de
                        control
299                  analogWrite(BpinACC,(byte)(abs(accB)*255.0/MAXU)) ;
300
301        t2=millis();
302
303        if (t2-t1>Tr) {digitalWrite(RT,HIGH); //Espera hasta alcanzar T
304                 // Serial1.print(" ");
305                  Serial.println(t2-t1);
306                  }
307        else {digitalWrite(RT,LOW) ;
308              Serial.println(t2-t1);
309        }
310       //A_u2k=1.089*A_u2k1 - 0.2386*A_u2k2 + 1.347*A_u1k - 2.066*A_u1k + 0.7917*A_u1k; //
              GT2 = MT/GpT
311       A_u2k=0.7511*A_u2k1 - 0.001921*A_u2k2 + 2*A_u1k - 3.009*A_u1k + 1.132*A_u1k; // GT2
              = Grd/(1+Grd*GpT)
312       A_u2k2=A_u2k1;
313       A_u2k1=A_u2k;
314       accA=A_u2k;
```

```
315     //B_u2k=1.089*B_u2k1 - 0.2386*B_u2k2 + 1.347*B_u1k - 2.066*B_u1k + 0.7917*B_u1k; //
            GT2 = MT/GpT
316      B_u2k=0.7511*B_u2k1 - 0.001921*B_u2k2 + 2*B_u1k - 3.009*B_u1k + 1.132*B_u1k; // GT2
             = Grd/(1+Grd*GpT)
317      B_u2k2=B_u2k1;
318      B_u2k1=B_u2k;
319      accB=B_u2k;
320
321      if (accA>=MAXU) {accA=MAXU ;} ; if (accA<=MINU) {accA=MINU ;} //Saturar acciones de
             control
322           if (accB>=MAXU) {accB=MAXU ;} ; if (accB<=MINU) {accB=MINU ;}
323           if (accA>0) {
324                     digitalWrite(ApinDIR,CW) ;
325            } else {digitalWrite(ApinDIR,CCW);}
326            if (accB>0) {
327                     digitalWrite(BpinDIR,CW) ;
328            } else {digitalWrite(BpinDIR,CCW) ;}
329
330           analogWrite(ApinACC,(byte)(abs(accA)*255.0/MAXU)) ; // Aplicar acciones de
                 control
331           analogWrite(BpinACC,(byte)(abs(accB)*255.0/MAXU)) ;
332
333
334       // Actualizar variables  A_u1k2=A_u1k1;
335      A_u1k1=A_u1k;
336      A_ek2=A_ek1;
337      A_ek1=A_ek;
338
339      B_u1k2=B_u1k1;
340      B_u1k1=B_u1k;
341      B_ek2=B_ek1;
342      B_ek1=B_ek;
343
344      // Lectura y calculo de velocidades
345      enc_actA=quadA;
346      enc_difA=enc_actA-enc_antA ;
347      posA=(enc_actA)*C2D ; //pos=((int)pos % 360)-180.0 ;
348      velA=(enc_actA-enc_antA)*C2RS ;
349      enc_antA2=enc_antA;
350      enc_antA=enc_actA ;
351
352      enc_actB=quadB ;
353      enc_difB=enc_actB-enc_antB ;
354      posB=(enc_actB)*C2D ; //pos=((int)pos % 360)-180.0 ;
355      velB=(enc_actB-enc_antB)*C2RS ;
356      enc_antB2=enc_antB;
357      enc_antB=enc_actB;
358
359      // Envio velocidades
360      velIzq=(velB+10)*100+1000;
361      velDer=(velA+10)*100+1000;
362      buffOUT+=velDer;
363      buffOUT+=velIzq;
364      Serial1.print(buffOUT);
365      t3=millis();
366      t2=millis();
367
368   if (t2-t1>T) {digitalWrite(RT,HIGH);  // Espera hasta NT
369                 // Serial1.print(" ");
370                  Serial.println(t2-t1);
371                  }
372   else {digitalWrite(RT,LOW) ;
373         Serial.println(t2-t1);
374        }
375   while (t2-t1<T) {t2=millis() ;}
376   t1=millis() ;
377 }
378
379 // Handlers de las interrupciones para los encoders
```

```
380
381 void Aencoder_chA (){if(digitalRead(ApinENC_chA)==digitalRead(ApinENC_chB)){ quadA--;}
        else{quadA++;} }
382 void Aencoder_chB (){if(digitalRead(ApinENC_chA)==digitalRead(ApinENC_chB)){ quadA++;}
        else{quadA--;} }
383
384 void Bencoder_chA (){if(digitalRead(BpinENC_chA)==digitalRead(BpinENC_chB)){ quadB--;}
        else{quadB++;} }
385 void Bencoder_chB (){if(digitalRead(BpinENC_chA)==digitalRead(BpinENC_chB)){ quadB++;}
        else{quadB--;} }
```

C. Archivos de MATLAB

Script: Identificación de la planta

```
clear all;
close all;
clc;
salida = load ('salida.txt') % Carga de Datos
posk=salida(:,1);
velk=salida(:,2);
uk=salida(:,3);
tiempo=[0:0.01:36.06];
s=tf('s');

f1=figure('Units','centimeters','Position',[1 1 30 20]);
set(gcf,'papersize',[29 19]);

for i=1:6
    switch i    %% Limites donde cambia la accion de control
        case 1
            t_in=1;
            t_fin=601;
            acc=30;
            ejes=[0 6 0 6]
        case 2
            t_in=602;
            t_fin=1202;
            acc=40;
            ejes=[6 12 0 9]
        case 3
            t_in=1203;
            t_fin=1803;
            acc=50;
            ejes=[12 18 0 12]
        case 4
            t_in=1804;
            t_fin=2404;
            acc=-30;
            ejes=[18 24 -8 0]
        case 5
            t_in=2405;
            t_fin=3005;
            acc=-40;
            ejes=[24 30 -9 0]
        case 6
            t_in=3006;
            t_fin=3606;
            acc=-50;
            ejes=[30 36 -12 0]
    end
    subplot(2,3,i)
    plot(tiempo(t_in:t_fin), uk(t_in:t_fin)); % Grafica tiempo vs Acc
```

```
    hold on;
    plot(tiempo(t_in:t_fin), velk(t_in:t_fin)) % Grafica tiempo vs Vel
    grid on;
    axis(ejes)
    xlabel('Tiempo(s)');
    ylabel('Amplitud');
    title(['Accion de control = ' num2str(acc) '%'])
    VelMAX=abs(velk(t_fin));
    AccMAX=abs(uk(t_fin));
    Kest_it(i)=VelMAX/AccMAX % Obtencion ganancia estatica
    for j=t_in:t_fin
        if (abs(velk(j))-0.63*VelMAX<0.001)
            tau_it(i)=tiempo(j)-tiempo(t_in+150); %Obtencion constante
        end                                       % de tiempo
    end
end
Kest_it2=[Kest_it(1:2) Kest_it(4:5)]
Kest=mean(Kest_it2) %Ganancia estatica media
tau_it2=[tau_it(1:2) tau_it(4:5)]
tau=mean(tau_it2(:))    % Constante de tiempo media
Gpvel=Kest/(tau*s+1)    % Modelo de velocidad
Gppos=Kest/(tau*s^2+s) % Modelo de posicion

print('respMotVel', '-dpdf'); %Guardar grafica

f2=figure('Units','centimeters','Position',[1 1 30 20]);
set(gcf,'papersize',[29 19]);

ukSIM=[tiempo(1:3606)',uk(1:3606)]; % Grafica tiempo vs Acc
sim valida2.slx
plot(tiempo(1:3606),velk(1:3606)) % Grafica tiempo vs velocidad real
hold on;
plot(tiempo(1:3606), simout(1:3606)) % Grafica tiempo vs velocidad
grid;                                % simulacion
legend('Salida real','Salida modelo','Location', 'NorthEast')
axis([0 36.06 -20 20])
xlabel('Tiempo(s)');
ylabel('Amplitud');

print('modeloMotor', '-dpdf'); %Guardar grafica
%
```

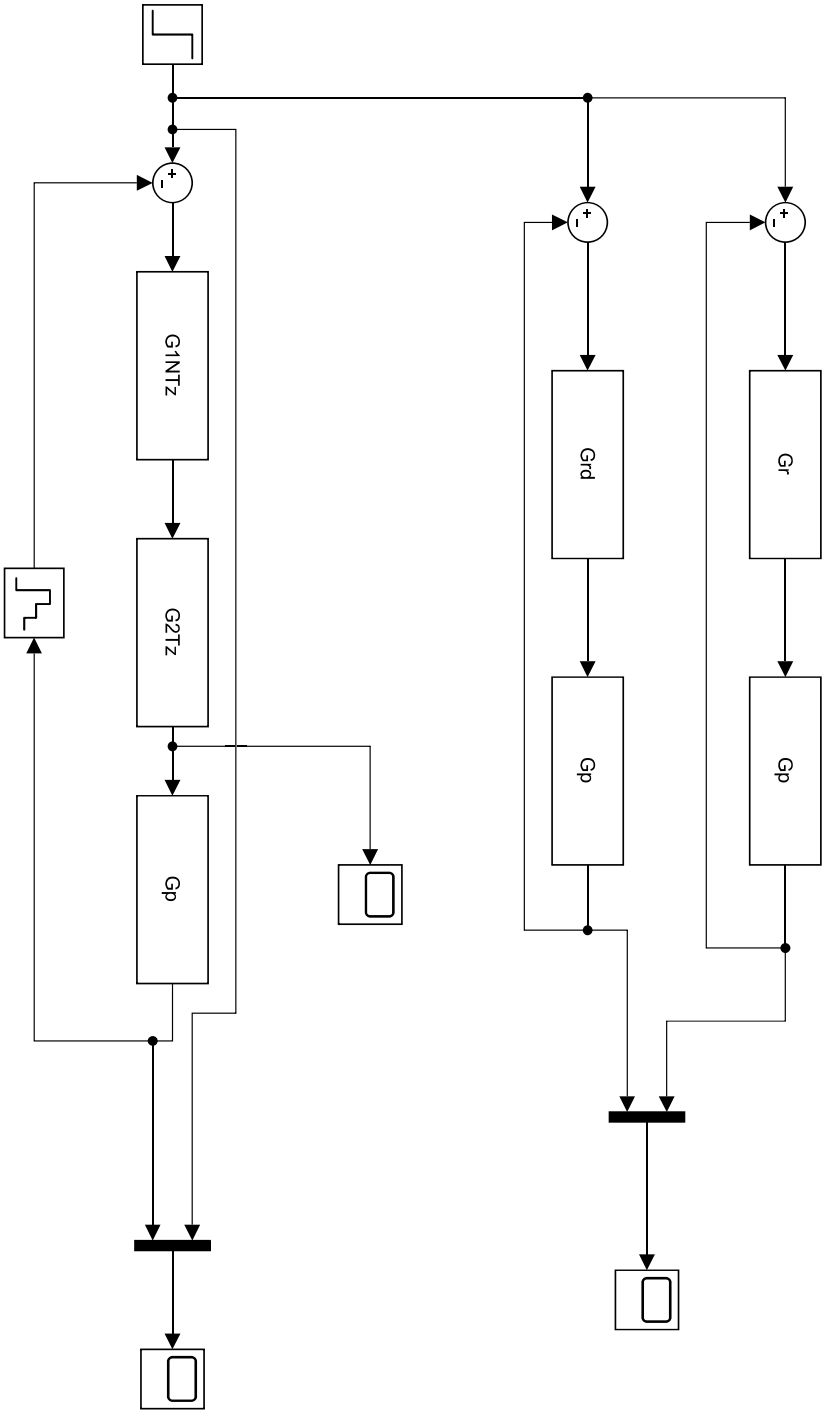
Script: Identificación zona muerta y curva entrada/salida

```
clear all;
salida = load ('salida.txt')
tiempo=0:0.01:20.01;
posk=salida(:,1);
velk=salida(:,2);
uk=salida(:,3);
% posk=floor(mod(posk,360))-180;
%% ZONA MUERTA VELOCIDAD
f1=figure('Units','centimeters','Position',[1 1 30 20]);
set(gcf,'papersize',[29 19]);

plot(uk(1:1001),velk(1:1001)); % De -10 a 10 voltios
hold on;
plot(uk(1002:2000),velk(1002:2000)); % De 10 a -10 voltios
grid;
legend('Rampa ascendente -10 a 10','Rampa descendente 10 a -10','Location','SouthEast');
xlabel('Accion de control (u)');
ylabel('Velocidad (rad/s)');
title('Caracterizacion zona muerta del motor')
print('zonaMuertaMotor','-dpdf');

%%% CURVA ENTRADA SALIDA
f2=figure('Units','centimeters','Position',[1 1 30 20]);
set(gcf,'papersize',[29 19]);
plot(tiempo(1:2001),uk);
hold on;
plot(tiempo(1:2001),velk);
grid;
legend('Referencia','Lectura velocidad','Location','SouthEast')
xlabel('Tiempo(s)')
ylabel('Velocidad (rad/s)')
%title('Curva entrada salida del motor')
print('curvaEntradaSalida','-dpdf')
%
```

Script: Obtención del regulador multifrecuencia

```
%% Control MR
clear all;
clc;
s=tf('s');
z=tf('z');
Tm=0.1;

Gp=5.757/(s + 2.817);
GpTz=c2d(Gp,Tm,'zoh'); %Discretizacion proceso
Gr = 1.3963*(s+2.82)/s;


MT=minreal(Gr*Gp/(1+Gp*Gr)); %Obtencion planta
MTz=c2d(MT,Tm,'zoh');        %Discretizacion planta


f1=figure(1);
multiplo=[1,2,3,5,7,10];
for i=1:3
    N=multiplo(i)
    MNTz=c2d(MT,N*Tm,'zoh');
    %G2Tz=minreal(MTz/GpTz);   %Parte rapida con ripple
    G2T=minreal(Gr/(1+Gr*Gp));
    G2Tz=c2d(G2T,Tm,'zoh');    %Parte rapida sin ripple
    G1NTz=minreal(1/(1-MNTz)); %Parte lenta

    sim('simMR');
    if(i==1)
        plot(tiempo, ref);
        hold on;
    end
    plot(tiempo,salida);
end
    legend('Ref','N=1', 'N=2', 'N=8' ,'N=12', 'N=16','N=20')
    title('Control MR para T = 0.075 s')
    grid;
```

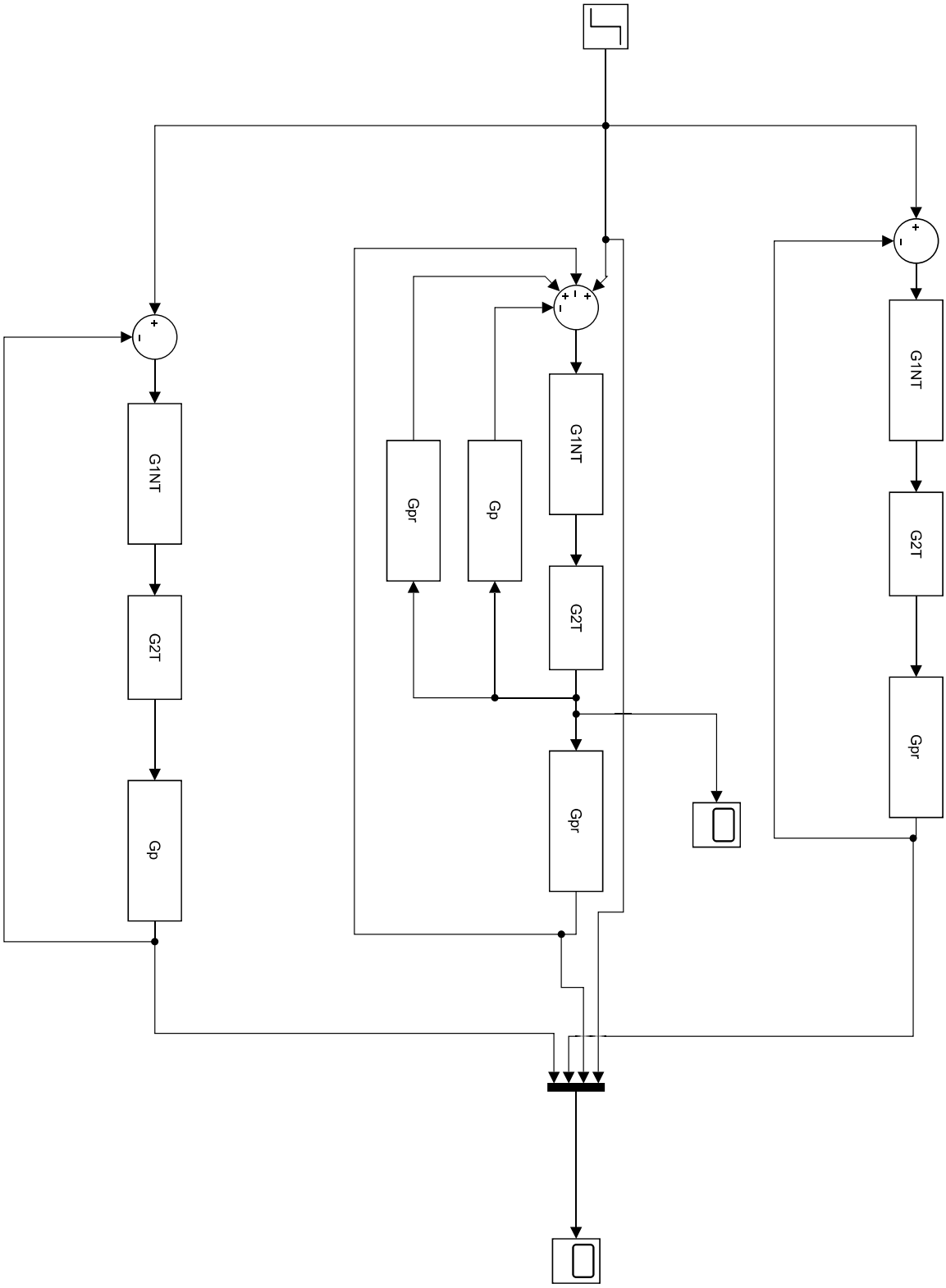
Simulink: Respuestas de los controladores



Script: Implementación del Predictor de Smith

```
s=tf('s');
Tm=0.100;
z=tf('z',Tm)
N=2;
d=N*Tm;
Gr=2*(s+2.5)/s
Grd=2*(z-0.75)/(z-1)
G1NT=(z^2-0.709*z+0.05691)/(z^2-1.595*z+0.5947) %Controlador LENTO
G2T=(1.347*z^2-2.066*z+0.7917)/(z^2-1.089*z+0.2386) %Controlador RAPIDO
Gp=5.757/(s+2.817); %Proceso sin retardo
Gpr=5.757/(s+2.817)*exp(-d*s); %Proceso con retardo

sim simMR.slx
tiempo=salida(:,1);
ref=salida(:,2);
salidaPred=salida(:,3);
salidaR=salida(:,4);
salidaSR=salida(:,5);

f1=figure('Units','centimeters','Position',[1 1 30 20]);
set(gcf,'papersize',[29 19]);
plot(tiempo,ref,'LineWidth',1.25);
hold on;
plot(tiempo,salidaSR,'LineWidth',1.25);
plot(tiempo,salidaR,'LineWidth',1.25);
plot(tiempo,salidaPred,'Color',[0.1,0.6,0.3],'LineWidth',1.25);
axis([0 2 -3.5 3.5])
grid;
xlabel('Tiempo(s)');
ylabel('Velocidad (rad/s)');
legend('Referencia','Salida proceso sin retardo','Salida proceso con retardo',...
    'Salida proceso + Predictor Smith');

print('salidaPredSmith', '-dpdf');
```

Simulink: Respuestas de los controladores



Script: Representación trayectoria realizada

```
clear all;
close all;
clc;
num=[2 3 4 5 52];
ind=[2 3 4 5 5];
color=[0.8 0.1 0.1];
for i=1:5

    ref=load(['ref100CuadN' num2str(num(i)) '.lvm']);   % Introducir nombre del fichero de la referencia
    tray=load(['tray100CuadN' num2str(num(i)) '.lvm']); % Introducir nombre del fichero de la trayectoria
    if (i<=4)
        f1=figure('Units','centimeters','Position',[1 1 20 20]);
        set(gcf,'papersize',[19 19])
        plot(ref(:,1),ref(:,2),'LineWidth',1.25,'Color',[0.2 0.4 0.6]);
    else color=[0.4 0.9 0.1];
    end
    hold on;
    plot(tray(:,1),tray(:,2),'--','LineWidth',1.5,'Color',color);
    titulo = ['Respuesta para N=' num2str(ind(i))'];
    leyenda = 'Salida';
    leyenda2= 'Variando Look-Ahead';
    if (i<=4)
        legend('Referencia', leyenda)
    end
    if (i>4)
        legend('Referencia', leyenda, leyenda2)
    end
    xlabel('Posicion X (mm)');
    ylabel('Posicion Y (mm)');
    title(titulo);
    axis([-200 2200 -200 2200], 'square') % Ajuste de los ejes
    grid on;
    set(gca,'Xtick',-200:200:2200) % Ajuste del Grid
    set(gca,'Ytick',-200:200:2200)
    nombre_foto=['cuadradoN' num2str(ind(i))];
    print( nombre_foto, '-dpdf');
end
```

Script: Obtención de los indicadores de error

```
clear all;
close all;
clc;
num=[2 3 4 5 52];
ind=[2 3 4 5 5];
Nmuestras=125;
for i=1:5
    n_ref=['ref100CuadN' num2str(num(i)) '.lvm'] ;   % Introducir nombre del fichero de la referencia
    n_tray=['tray100CuadN' num2str(num(i)) '.lvm'] ; % Introducir nombre del fichero de la trayectoria
    ref=load(n_ref);
    tray=load(n_tray);
    error=sqrt((ref(1:Nmuestras,1)-tray(1:Nmuestras,1)).^2+...
        (ref(1:Nmuestras,2)-tray(1:Nmuestras,2)).^2);
    IEA(i)=sum(abs(error))
    IEC(i)=sum(error.^2)/Nmuestras
end
hold off;
```

Script: Cálculo del consumo

```
%% Leer todos los ficheros
clear all;
clc;
for i=1:5
    for j=1:3
        switch j
            case 1
                numero='';
            case 2
                numero='1';
            case 3
                numero='2';
            case 4
                numero='3';
            case 5
                numero='4';
            case 6
                numero='5';
        end
        nombre =['consumoN' num2str(i) num2str(numero) '.m'];
        run(nombre); % Cargar datos
        tiempo=datos(1:7000,1);
        vR=datos(1:7000,2);
        iR=datos(1:7000,3);

        %Calculos modulo APC220
        pR=vR.*iR; %Potencia W
        pRW(j)=mean(pR(1:7000)); % Potencia media W

        %Calculos INTENSIDAD modem XTend-PKG-R
        for k=1:7000
             if iR(k)>= 0.102
                iModem(k,1) = iR(k)*580/112;
            end
            if (iR(k)>= 0.099 && iR(k)< 0.102)
                iModem(k,1) = iR(k)*110/100;
            end
            if (iR(k)< 0.099)
                iModem(k,1) = iR(k)*39/99;
            end
        end

        %Calculos POTENCIA modem XTend-PKG-R

        pModem=vR.*iModem; %Potencia W
        pModemW(j)=mean(pModem(1:7000)); % Potencia media W
    end
        pRAvg(i)=mean(pRW);
        pRAvgWh(i)=pRAvg(i);%*3600*1000;
        pModemAvg(i)=mean(pModemW);
        pModemAvgWh(i)=pModemAvg(i);%*3600*1000;
        vAvg=mean(vR);
        iRmAh =  pRAvgWh(i)/(vAvg)*1000;
        vida_batR(i)=2600/iRmAh;
        iModemmAh =  pModemAvgWh(i)/(vAvg)*1000;
        vida_batModem(i)=2600/iModemmAh;
        plotear; %Script para obtener las figuras

end
 resultados; %Script para sacar los datos por la
       %command window
```

Script: Plotear

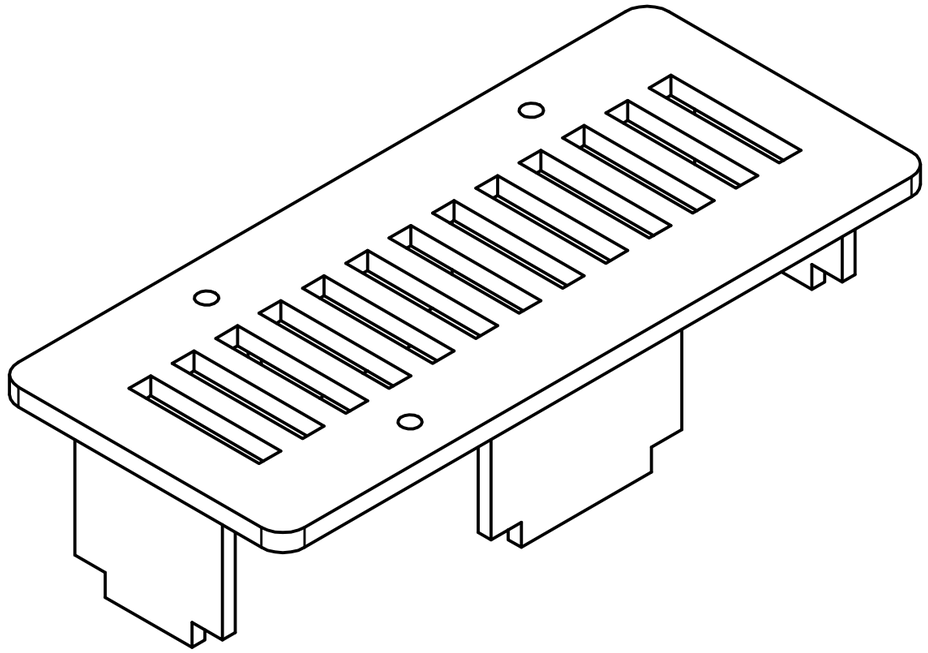
```
f1=figure('Units','centimeters','Position',[1 1 30 10]);
set(gcf,'papersize',[29 11])
plot(tiempo(1:2000)/1000,pR(1:2000));
titulo=['Consumo energetico para control multi-rate N=' num2str(i)];
title(titulo);
xlabel('Tiempo(s)');
ylabel('Potencia (W)');
grid;
set(gca,'Xtick',0:200:2000)
set(gca,'Ytick',0.7:0.02:1)
nombre_foto=['consumorealN' num2str(i)];
print( nombre_foto, '-dpdf');

f2=figure('Units','centimeters','Position',[1 1 30 10]);
set(gcf,'papersize',[29 11])
plot(tiempo(1:2000)/1000,pModem(1:2000)); %
titulo=['Consumo energetico para control multi-rate N=' num2str(i)];
title(titulo);
xlabel('Tiempo(s)');
ylabel('Potencia (W)');
grid;
set(gca,'Ytick',0.7:0.5:5)
nombre_foto=['consumomodemN' num2str(i)];
print( nombre_foto, '-dpdf');
```

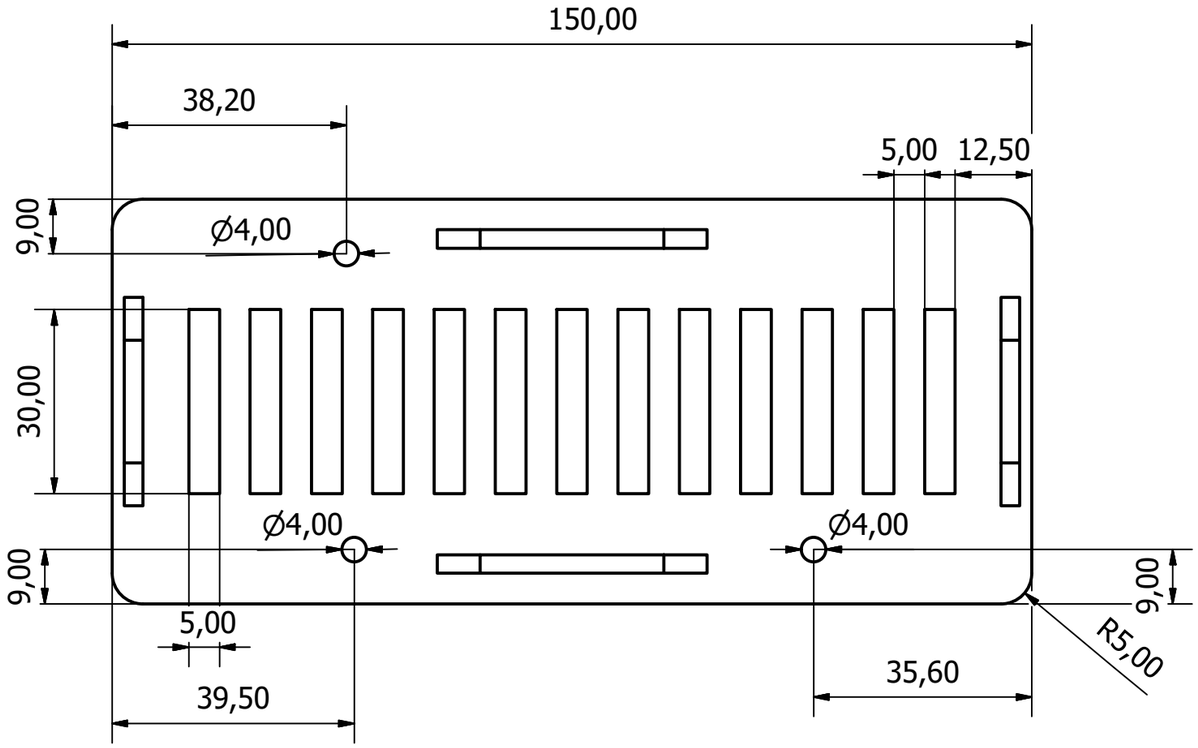
Script: Resultados

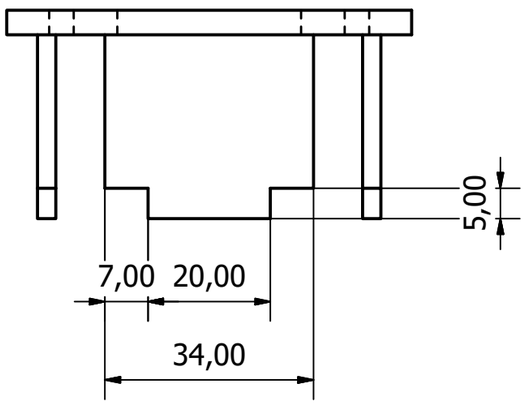
```
%% Calculo vida de la bateria
clc;
disp('------------------------');
disp('PARA EL MODULO RF APC220')
disp('------------------------');
for k=1:5
disp(['La vida de una bateria de 2600 mAh es de ' num2str(vida_batR(k),5),...
     ' h para N=' num2str(k)]);
 disp(['El consumo de potencia para N=' num2str(k) ' es de ' num2str(pRAvgWh(k)) ' Wh'])
 if i>=2
     ahorroR=(pRAvgWh(1)-pRAvgWh(k))/pRAvgWh(1)*100;
     disp(['El ahorro energetico entre realizar envios a N' num2str(k) ' respecto N1 es
         del '...
     num2str(ahorroR,4) ' %']);
 disp(' ');
 end
end
disp(' ')
disp('-----------------------------')
disp('PARA EL MODEM RF XTend-PKG-R')
disp('-----------------------------')
for k=1:5
disp(['La vida de una bateria de 2600 mAh es de ' num2str(vida_batModem(k),5),...
     ' h para N=' num2str(k)]);
 disp(['El consumo de potencia para N=' num2str(k) ' es de ' num2str(pModemAvgWh(k)) ' Wh
     '])
 if i>=2
     ahorroModem=(pModemAvgWh(1)-pModemAvgWh(k))/pModemAvgWh(1)*100;
     disp(['El ahorro energetico entre realizar envios a N' num2str(k) ' respecto N1 es
         del '...
     num2str(ahorroModem,4) ' %']);
 end
 disp(' ')
end
```

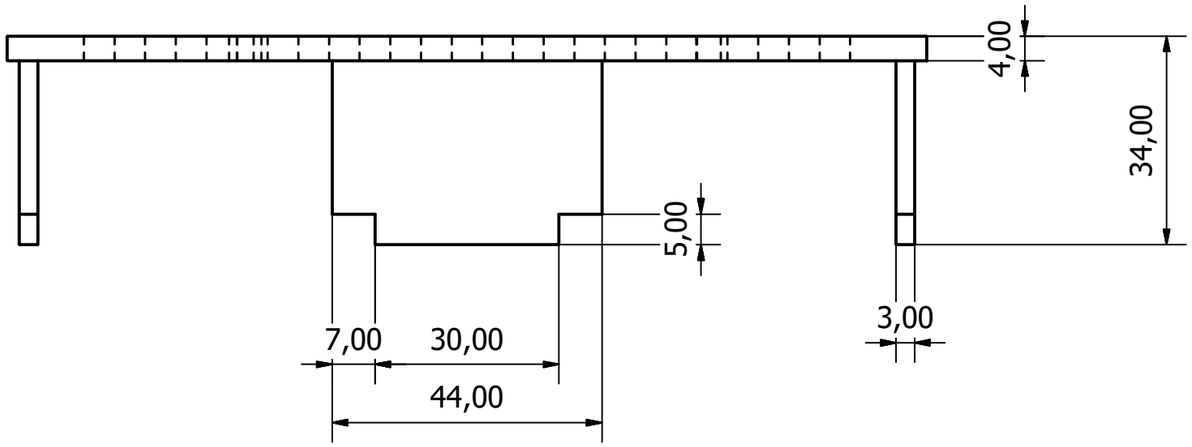
.

D. Planos de la carcasa

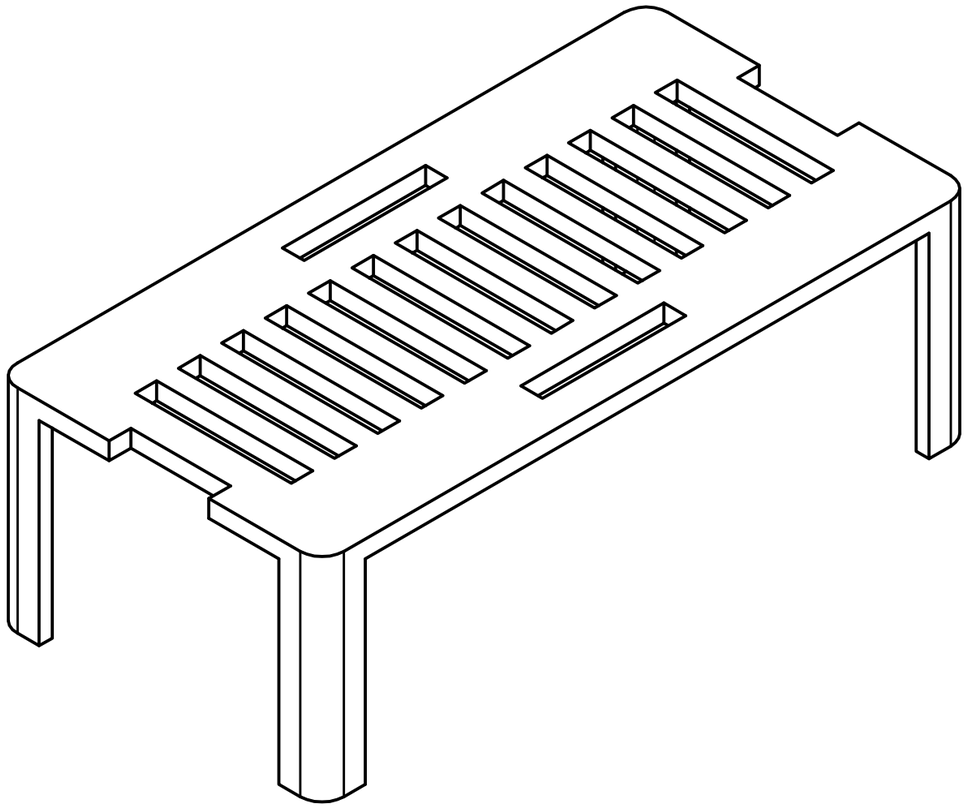


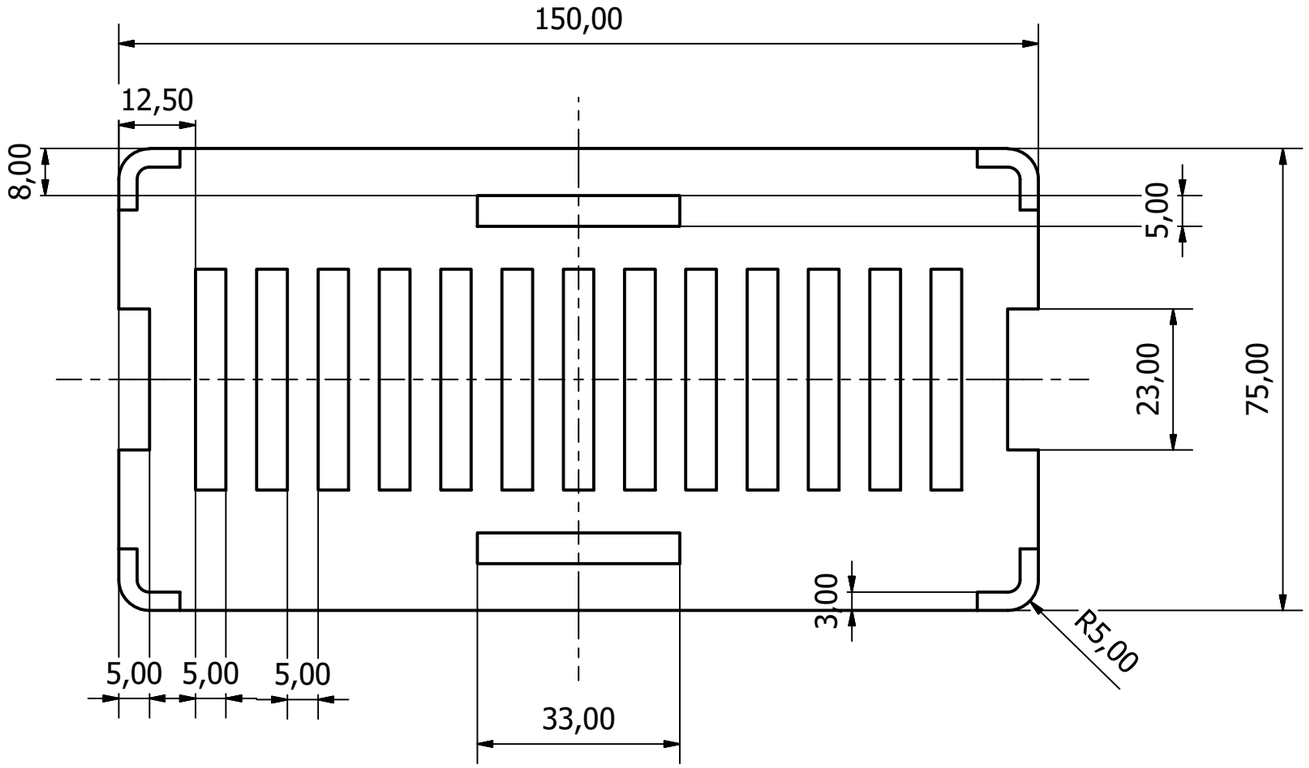


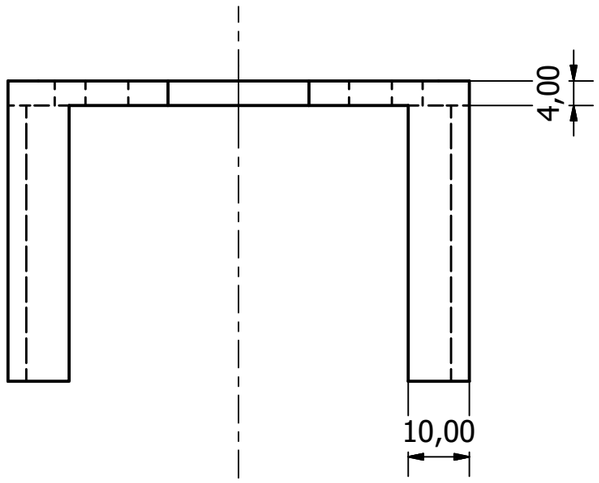


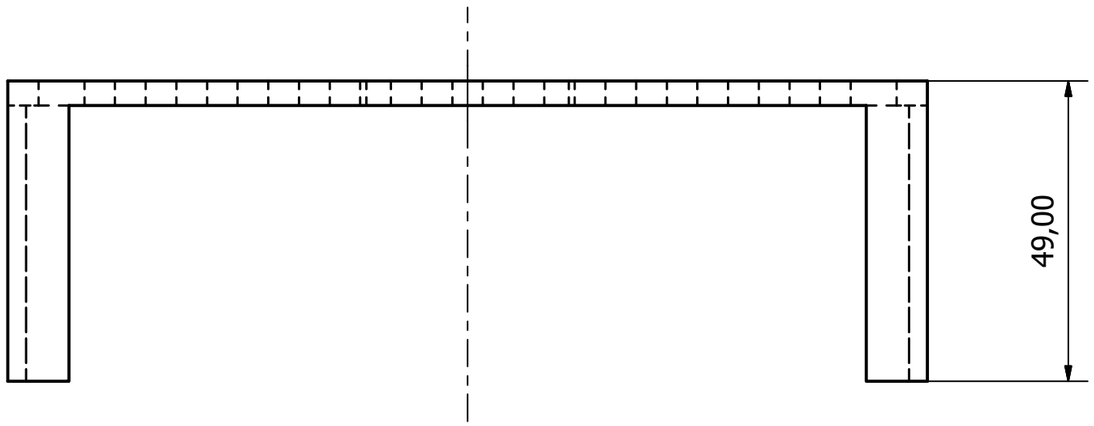


Diseño de Enrique Lledó	Descripción Pieza soporte para la placa Arduino Due	Fecha 25/06/18
UNIVERSITAT POLITÈCNICA DE VALÈNCIA	Proyecto: Diseño de un robot móvil de bajo coste	
	Pieza nº: 1	Escala 1:1 / Hoja 1 / 3

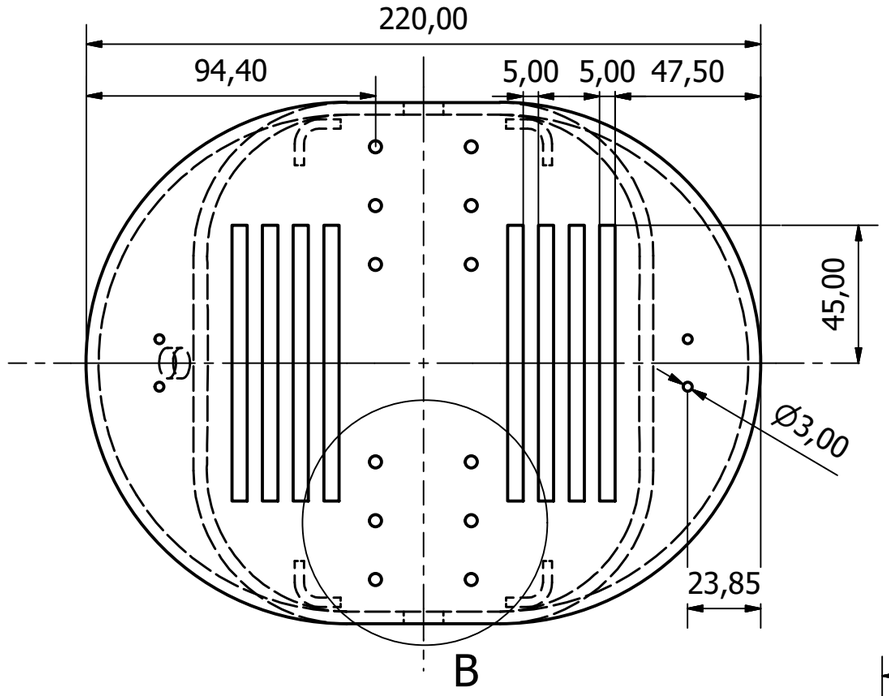




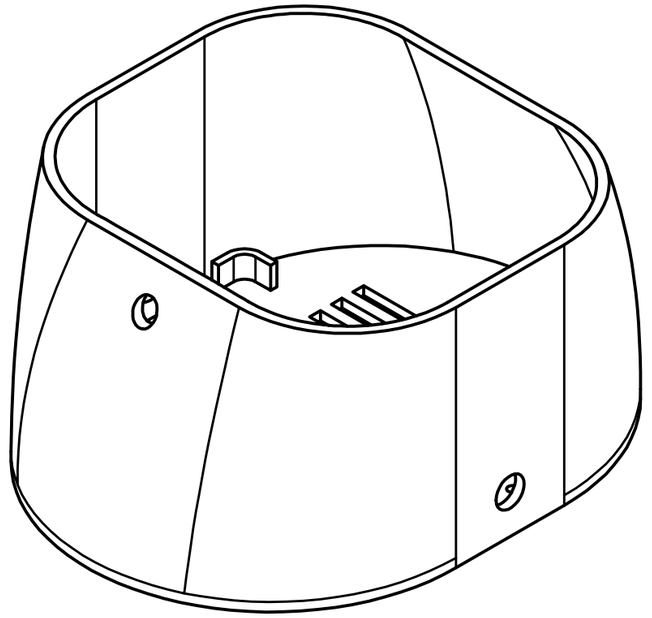


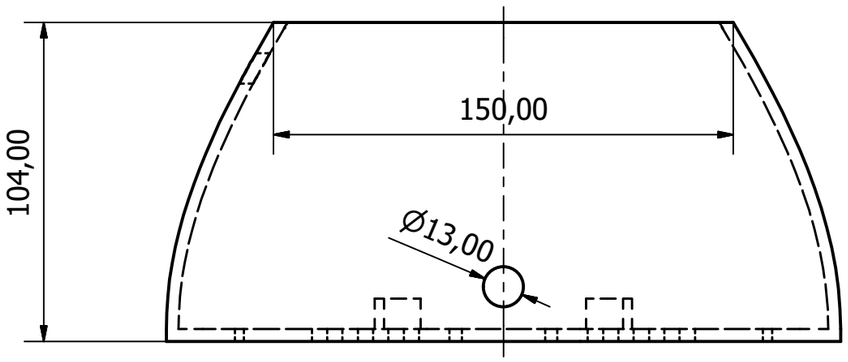


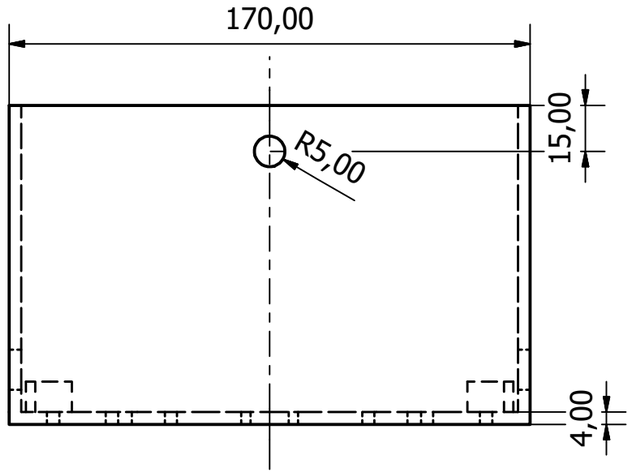
Diseño de Enrique Lledó	Descripción Pieza soporte para la batería	Fecha 25/06/18
UNIVERSITAT POLITÈCNICA DE VALÈNCIA	Proyecto: Diseño de un robot móvil de bajo coste	
	Pieza nº: 2	Escala 1:1
		Hoja 2 / 3

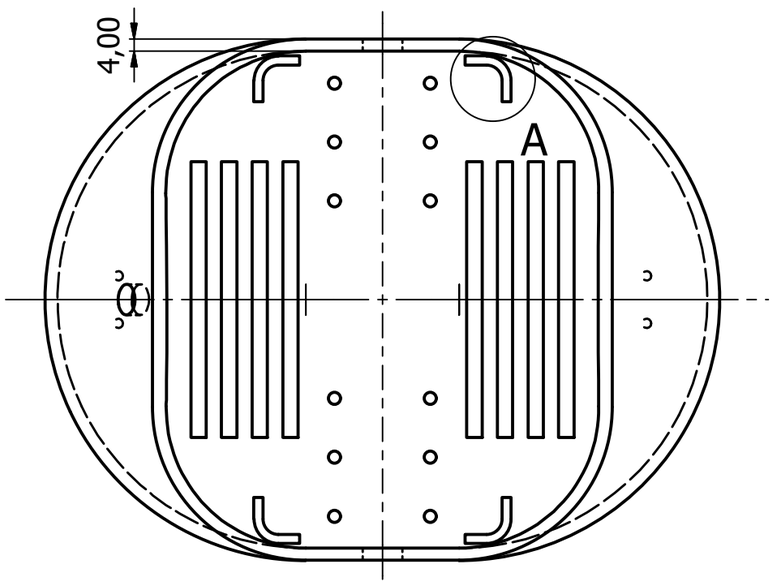


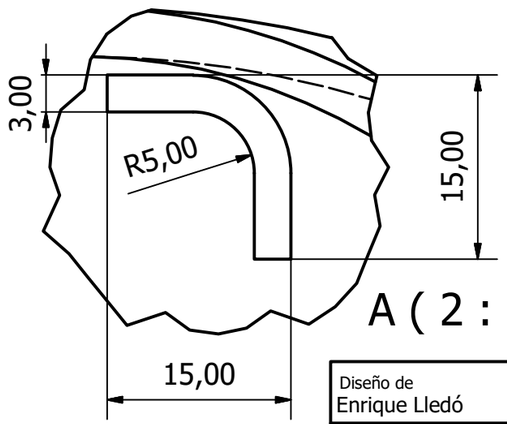
B





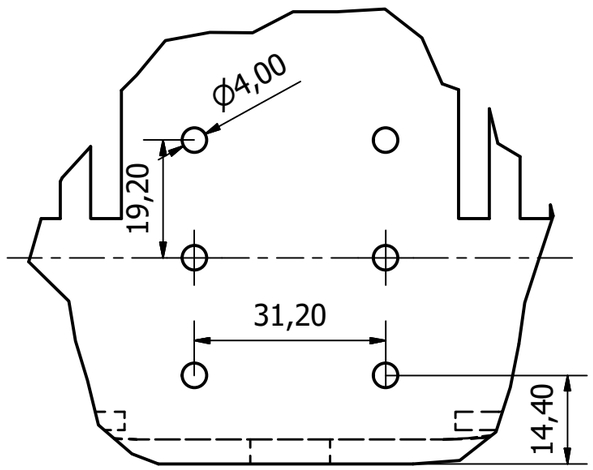






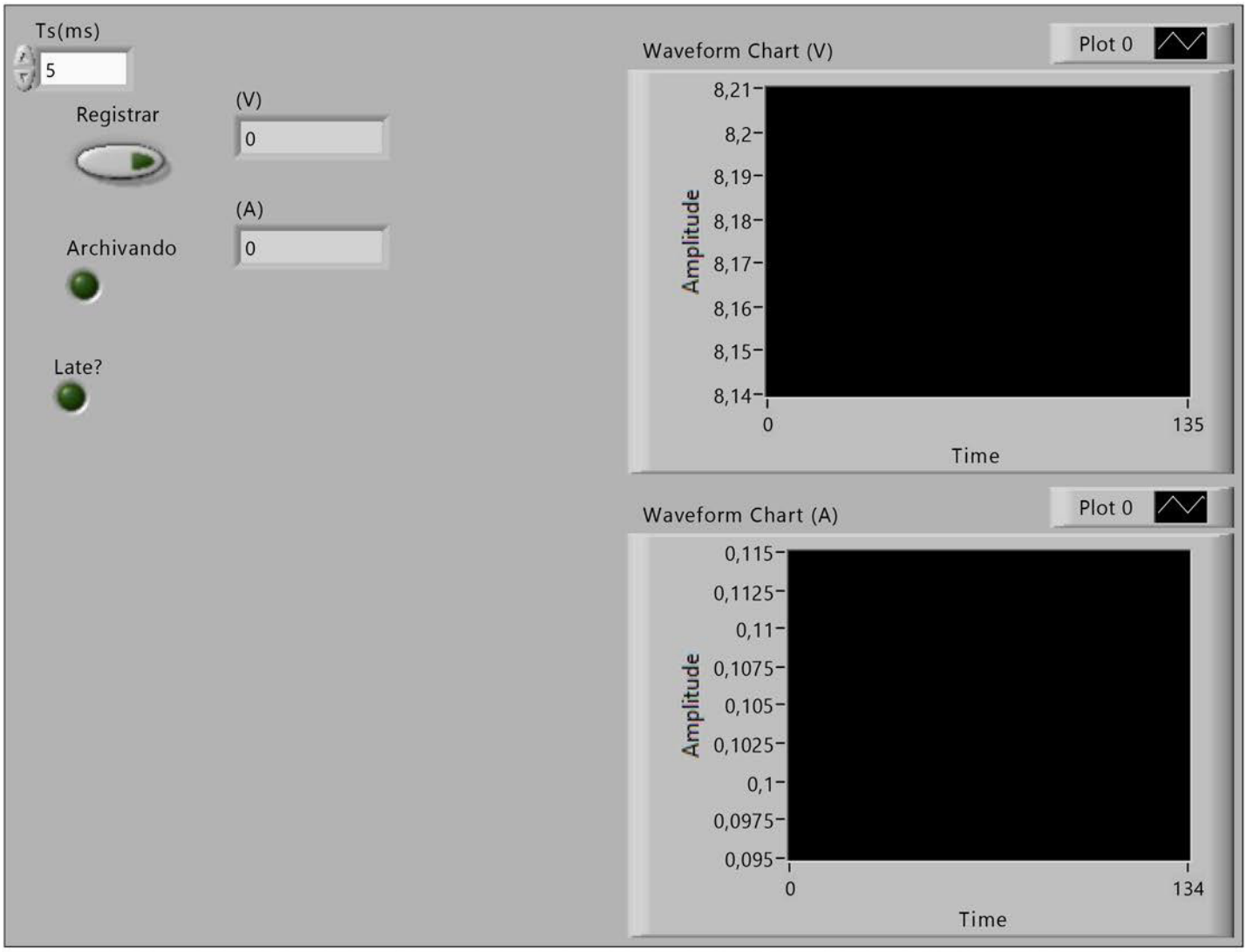
A (2 : 1)

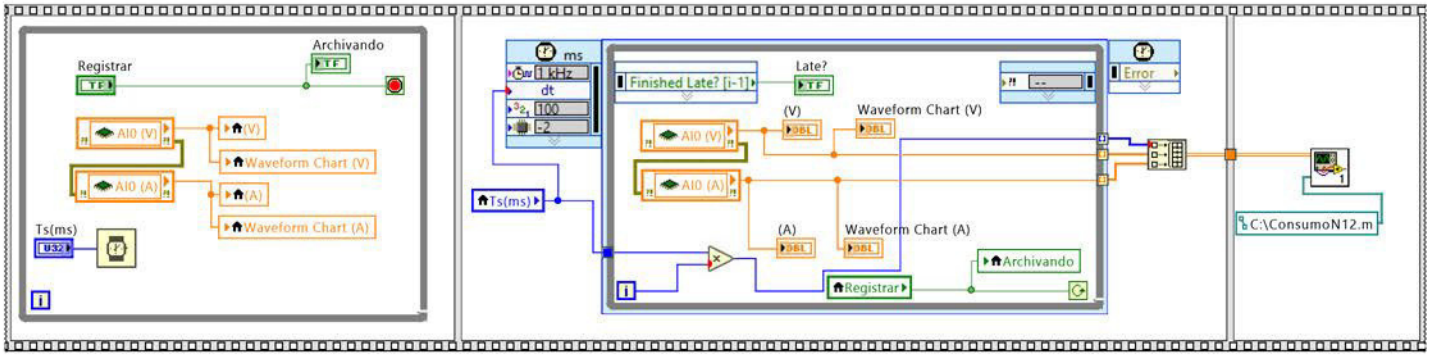
B (1 : 1)

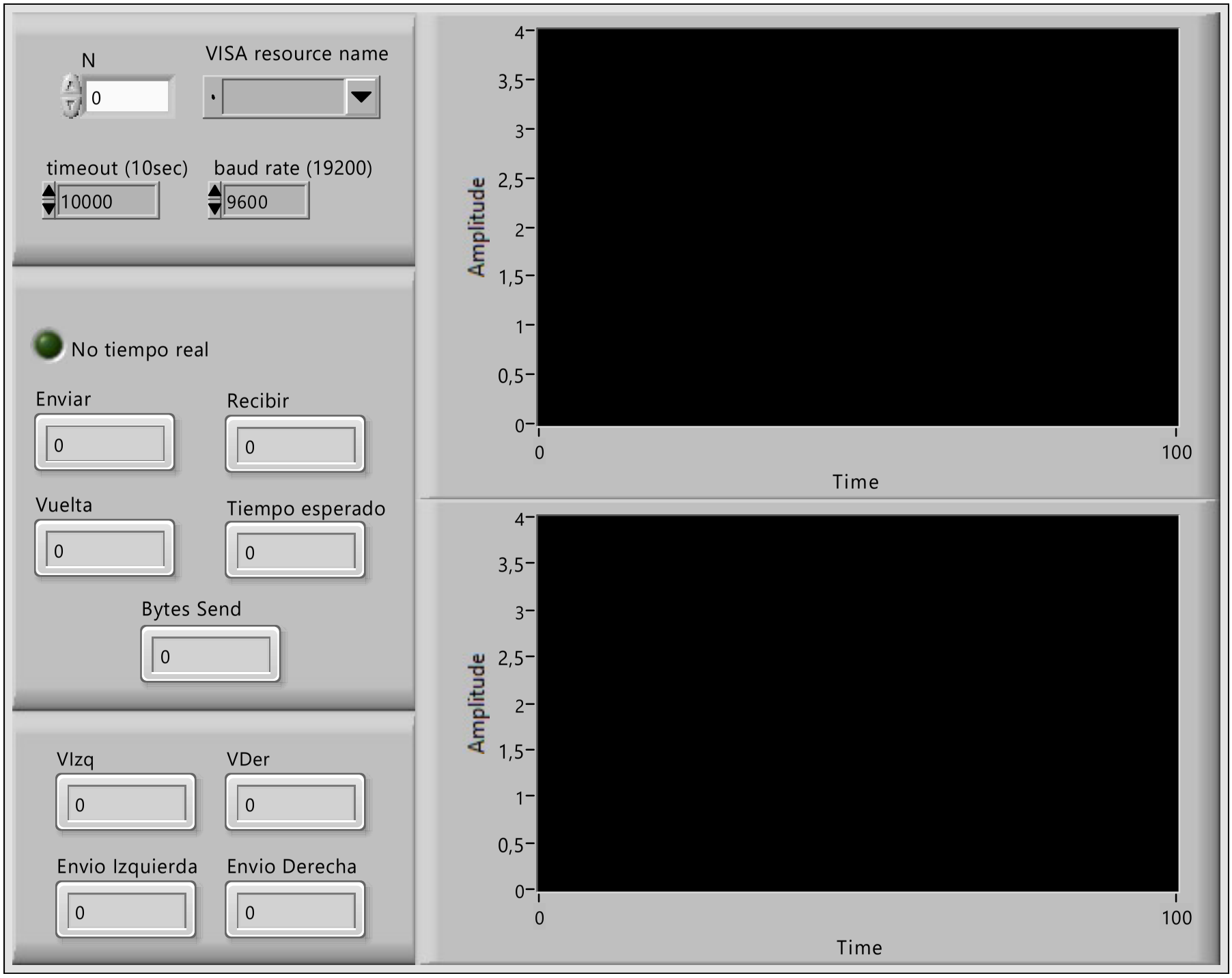


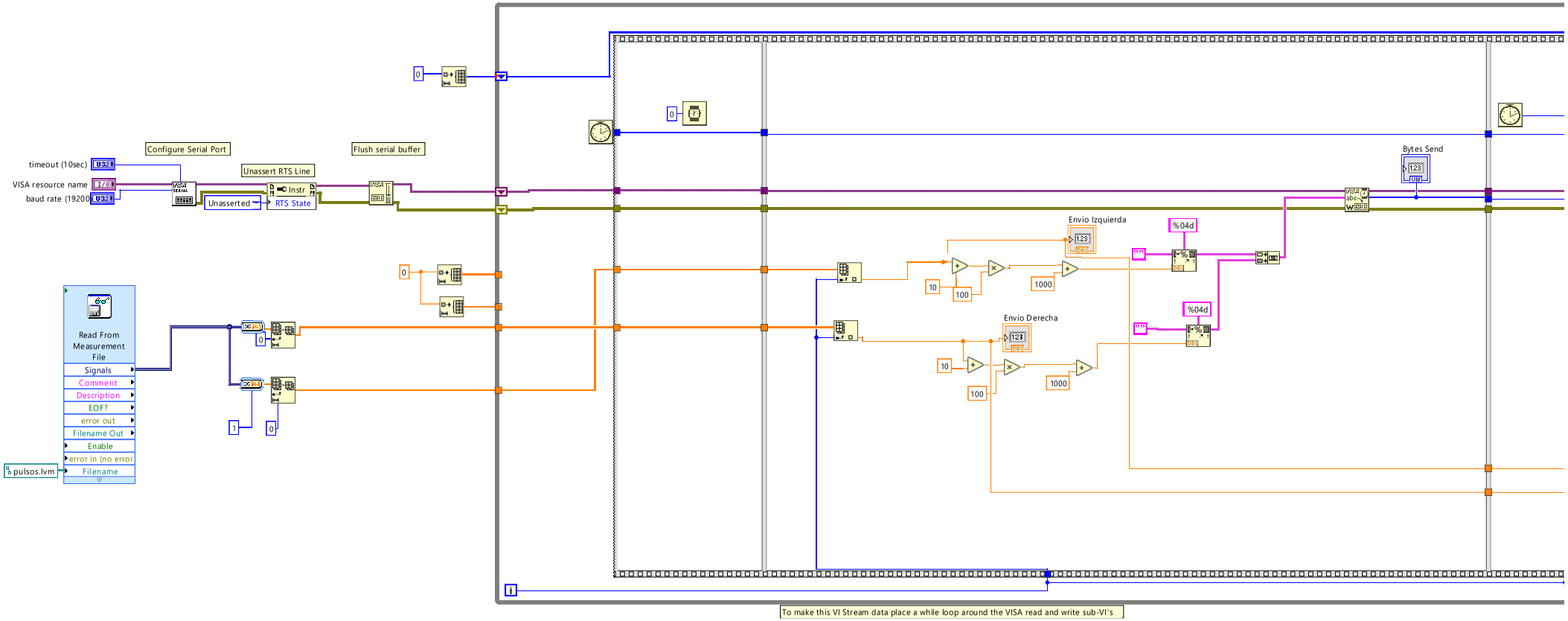
Diseño de Enrique Lledó	Descripción Carcasa externa del robot móvil	Fecha 25/06/18
UNIVERSITAT POLITÈCNICA DE VALÈNCIA	Proyecto: Diseño de un robot móvil de bajo coste	
	Pieza nº: 3	Escala 1:1 / Hoja 3 / 3

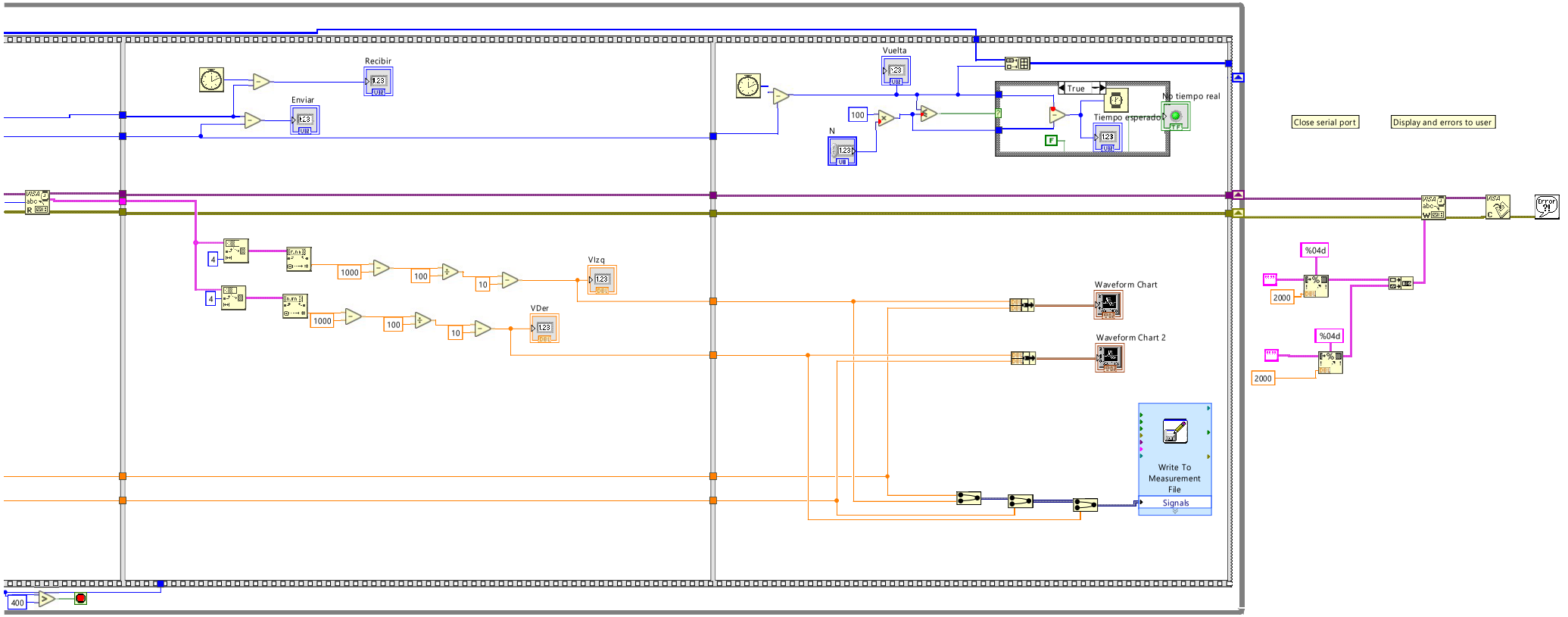
E. Programas en LabVIEW

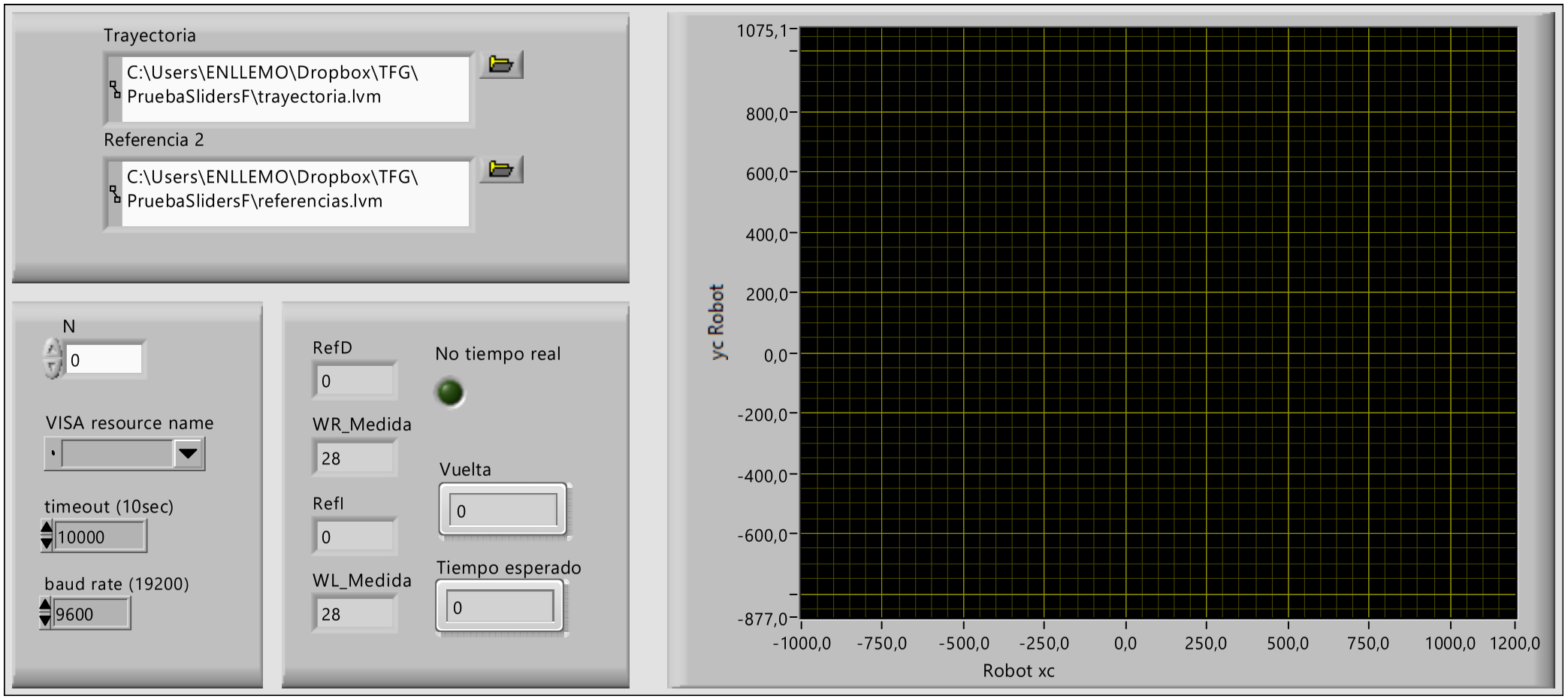

Ts(ms)
5
Registrar
(V)
0
(A)
0
Archivando
Late?
Waveform Chart (V)
Plot 0
8,21
8,2
8,19
8,18
8,17
8,16
8,15
8,14
Amplitude
0
135
Time
Waveform Chart (A)
Plot 0
0,115
0,1125
0,11
0,1075
0,105
0,1025
0,1
0,0975
0,095
Amplitude
0
134
Time

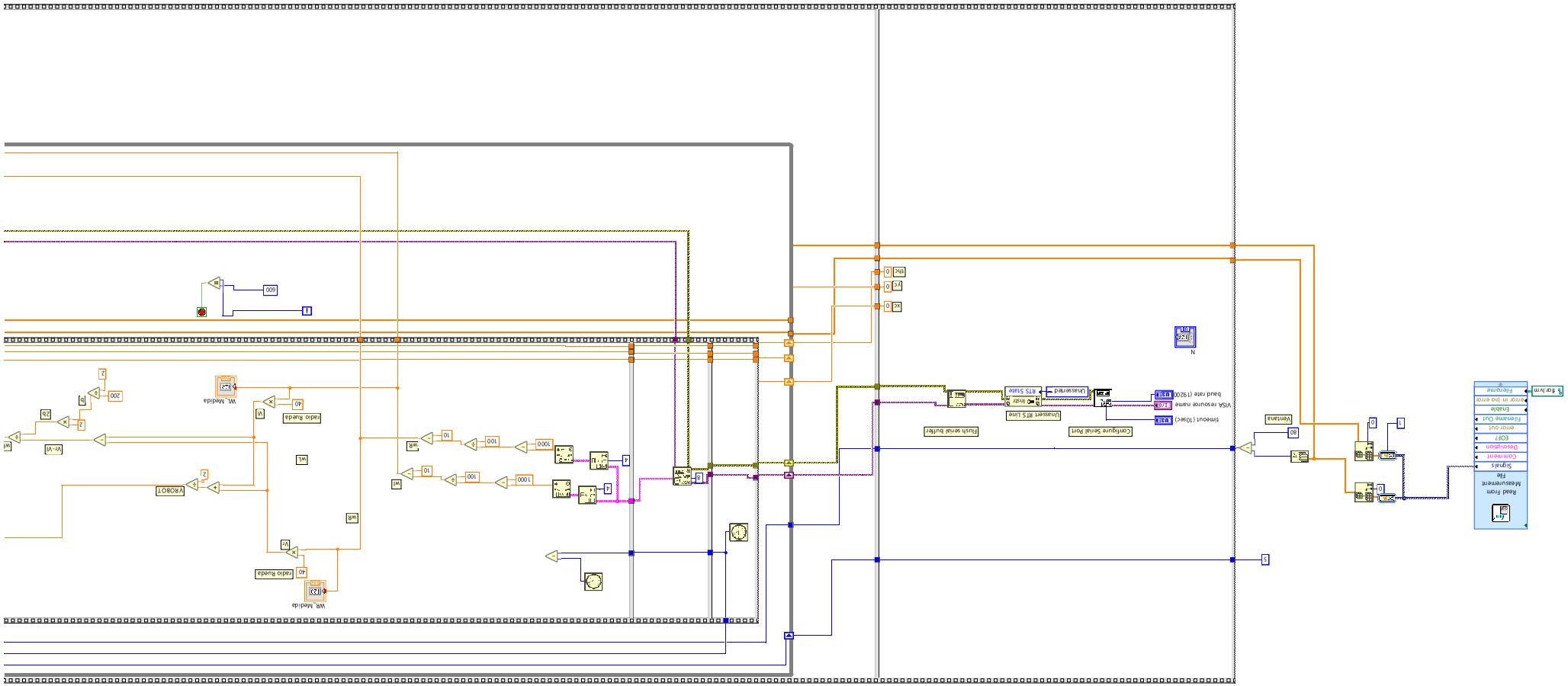

Registrar
Archivando
AI0 (V)
(V)
Waveform Chart (V)
AI0 (A)
(A)
Waveform Chart (A)
Ts(ms)
ms
1 kHz
dt
100
-2
Finished Late? [i-1]
Late?
Error
Ts(ms)
Archivando
Registrar
C:\ConsumoN12.m

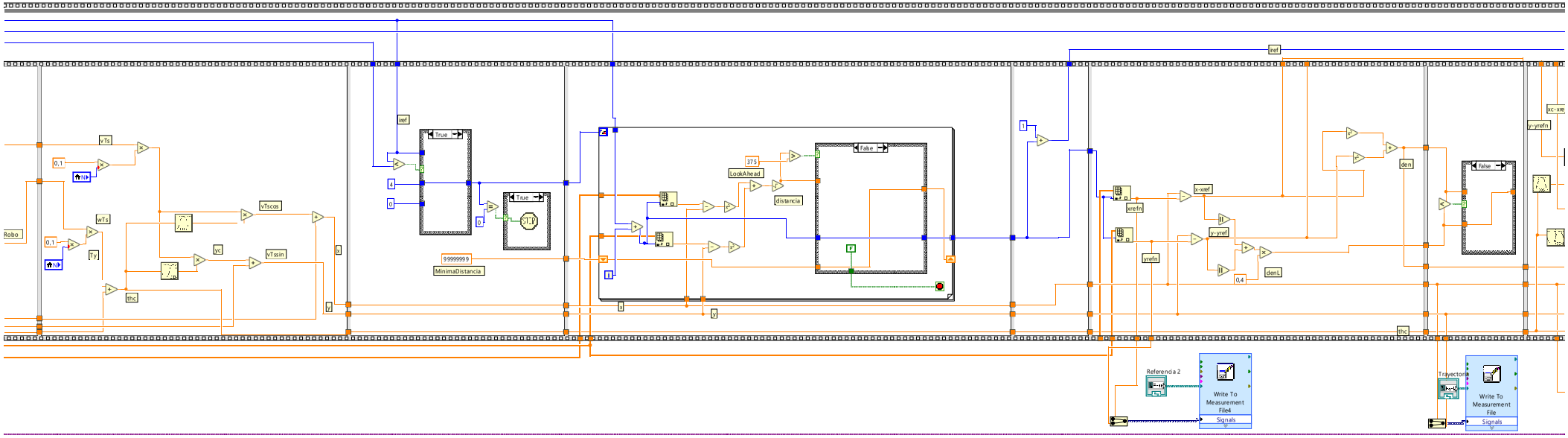

N
0
VISA resource name
timeout (10sec)
10000
baud rate (19200)
9600
No tiempo real
Enviar
0
Recibir
0
Vuelta
0
Tiempo esperado
0
Bytes Send
0
Vlzq
0
VDer
0
Envio Izquierda
0
Envio Derecha
0
Amplitude
4
3,5
3
2,5
2
1,5
1
0,5
0
0
100
Time
Amplitude
4
3,5
3
2,5
2
1,5
1
0,5
0
0
100
Time

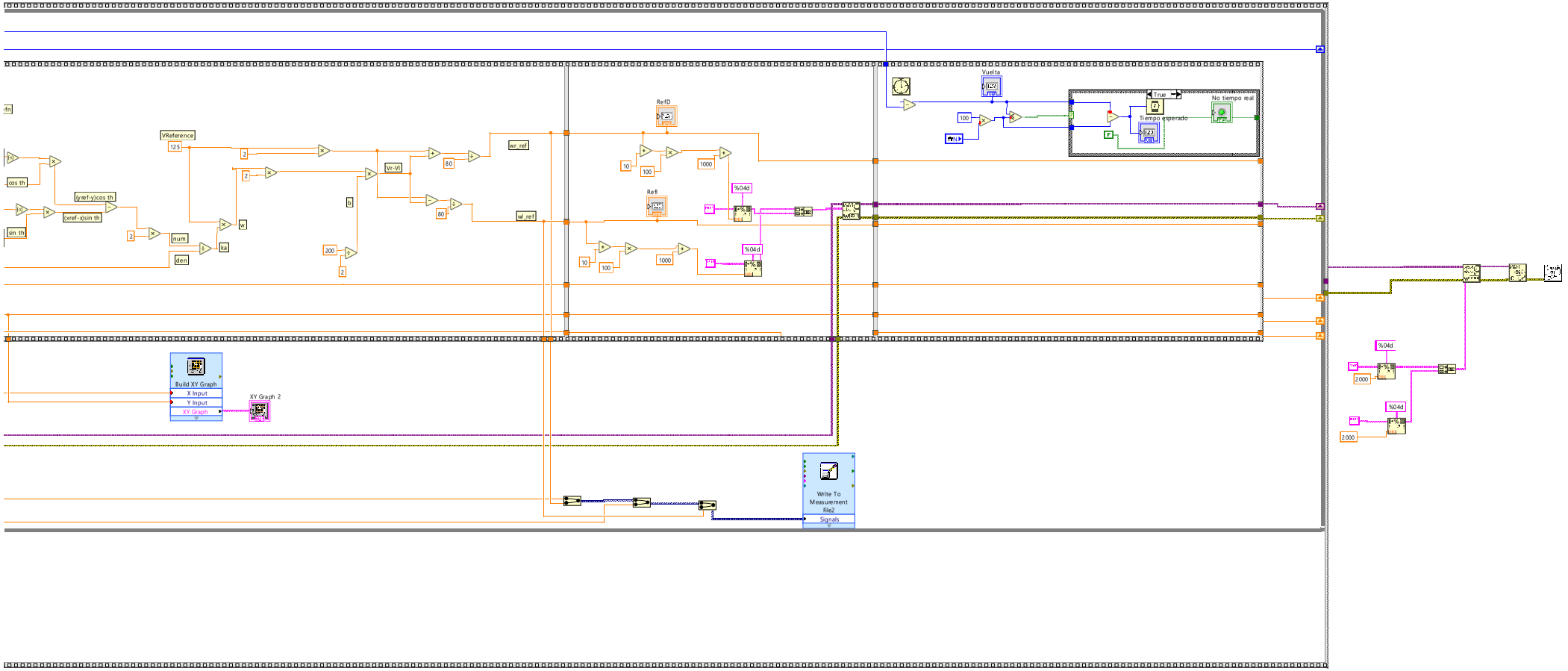

Configure Serial Port
timeout (10sec)
VISA resource name
baud rate (19200)
Unassert RTS Line
Unasserted
Instr
RTS State
Flush serial buffer
Read From Measurement File
Signals
Comment
Description
EOF?
error out
Filename Out
Enable
error in (no error
Filename
pulsos.lvm
Envio Izquierda
Envio Derecha
%04d
%04d
Bytes Send
10
100
1000
10
100
1000
To make this VI Stream data place a while loop around the VISA read and write sub-VI's


Recibir
Enviar
Vuelta
True
No tiempo real
Tiempo esperado
100
N
Close serial port
Display and errors to user
4
1000
100
10
VIzq
VDer
Waveform Chart
Waveform Chart 2
Write To Measurement File
Signals
%04d
2000
400


Trayectoria
C:\Users\ENLLEMO\Dropbox\TFG\
PruebaSlidersF\trayectoria.lvm
Referencia 2
C:\Users\ENLLEMO\Dropbox\TFG\
PruebaSlidersF\referencias.lvm
N
0
VISA resource name
timeout (10sec)
10000
baud rate (19200)
9600
RefD
0
WR_Medida
28
RefI
0
WL_Medida
28
No tiempo real
Vuelta
0
Tiempo esperado
0
yc Robot
1075,1
800,0
600,0
400,0
200,0
0,0
-200,0
-400,0
-600,0
-877,0
-1000,0
-750,0
-500,0
-250,0
0,0
250,0
500,0
750,0
1000,0
1200,0
Robot xc




vTs
0,1
wTs
Robo
0,1
Ty
vTscos
vTssin
yc
thc
ref
True
4
0
True
STOP
99999999
MinimaDistancia
375
LookAhead
distancia
False
F
1
xrefn
yrefn
x-xref
y-yref
0,4
denL
den
False
ref
xc-xre
y-yrefn
thc
Referencia 2
Write To Measurement File4
Signals
Trayectoria
Write To Measurement File
Signals


VReference
125
(y ref - y)cos th
(xref - x)sin th
num
den
wr_ref
wl_ref
Vr-Vl
RefD
Refl
Vuelta
True
Tiempo esperado
No tiempo real
%04d
Build XY Graph
X Input
Y Input
XY Graph
XY Graph 2
Write To Measurement File2
Signals
2000