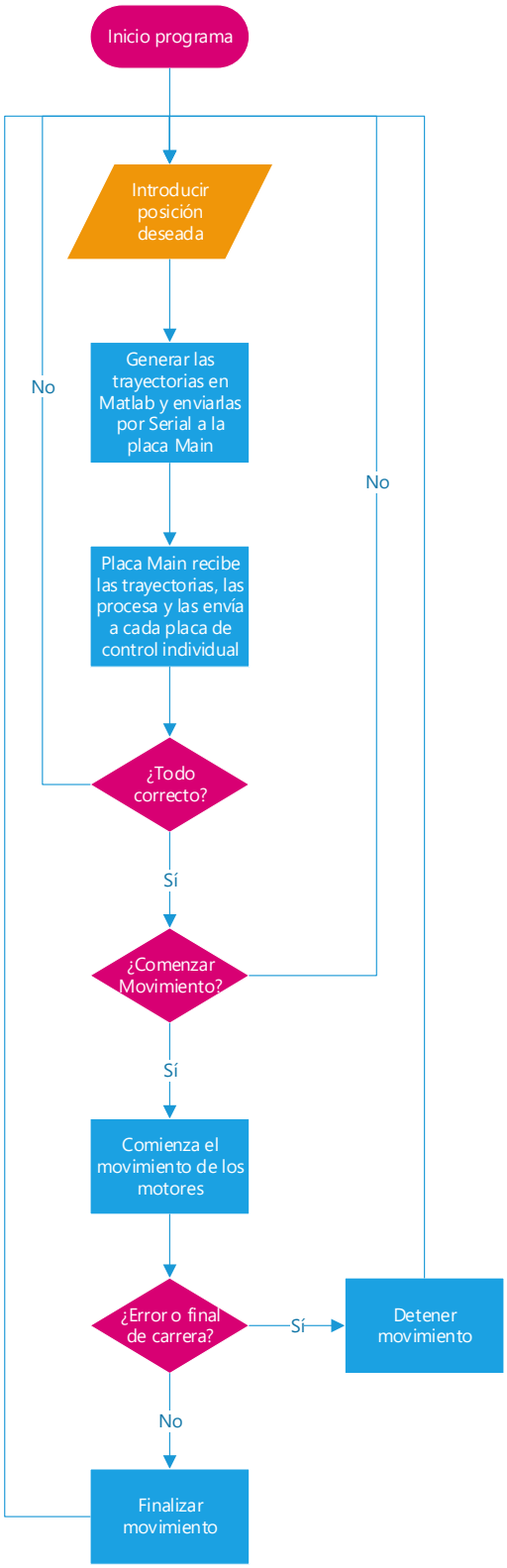
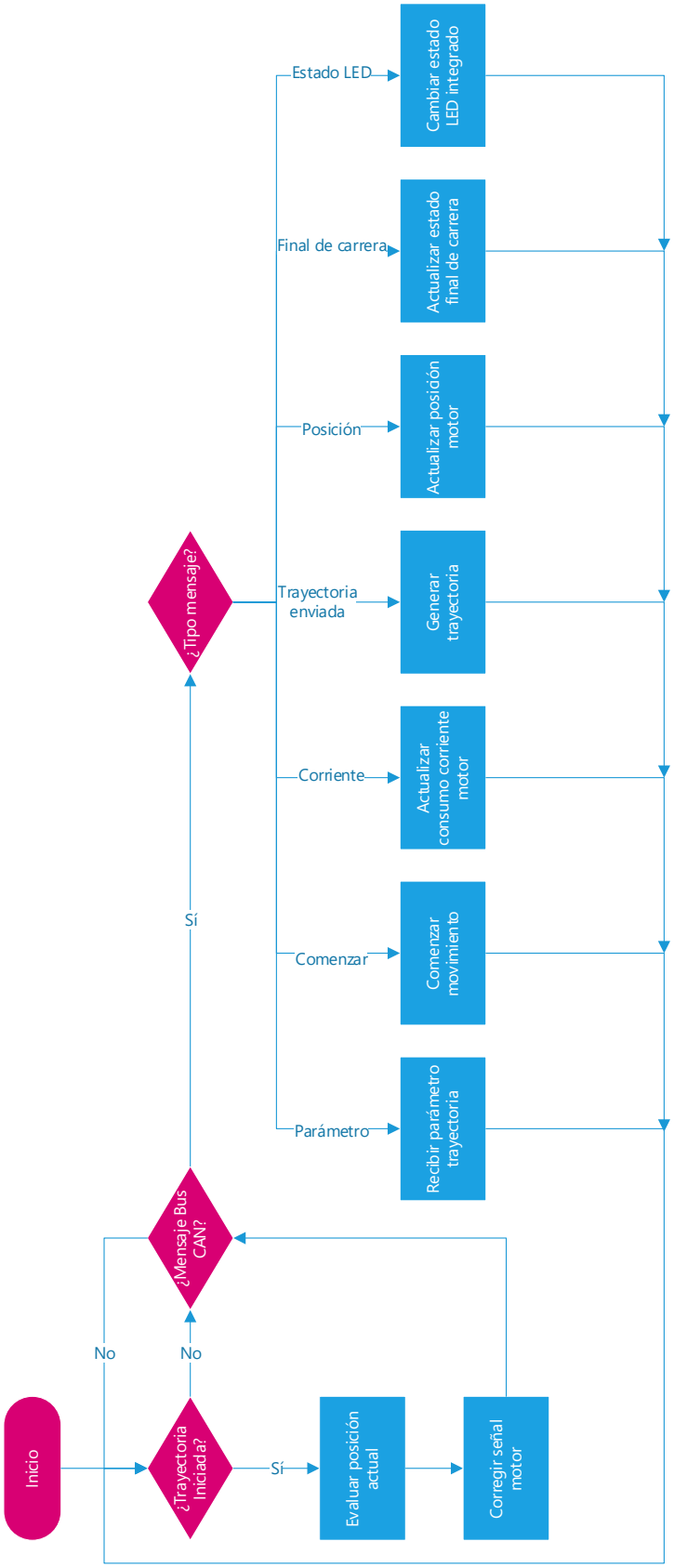
Esquema de control general

El siguiente esquema representa de manera muy simplificada el comportamiento del robot completo, fusionando la parte generada en Matlab con el procesamiento y la función de las diferentes placas de control.



Esquema interno placa drivers



La función principal de la placa que integra los drivers es cerrar el bucle de control del motor, pero también integra funciones secundarias como comunicar el estado del motor mediante bus CAN y otras funciones que se detallan en el esquema.

Su actuación empieza cuando comienza a recibir los parámetros necesarios para generar la trayectoria del motor asociado a dicha placa. Cuando todos los parámetros son recibidos, el programa se encarga de generar la trayectoria y queda listo para recibir la señal de inicio de movimiento. Una vez recibida dicha señal, se inicia automáticamente el movimiento siguiendo la trayectoria. La placa se encarga de comprobar si está en la posición correcta en todo momento, y en caso contrario corrige la desviación.

Para conseguir que el motor siga la trayectoria deseada, es necesario identificar el comportamiento del motor y programar un bucle de control adecuado a las necesidades.

Identificación del sistema

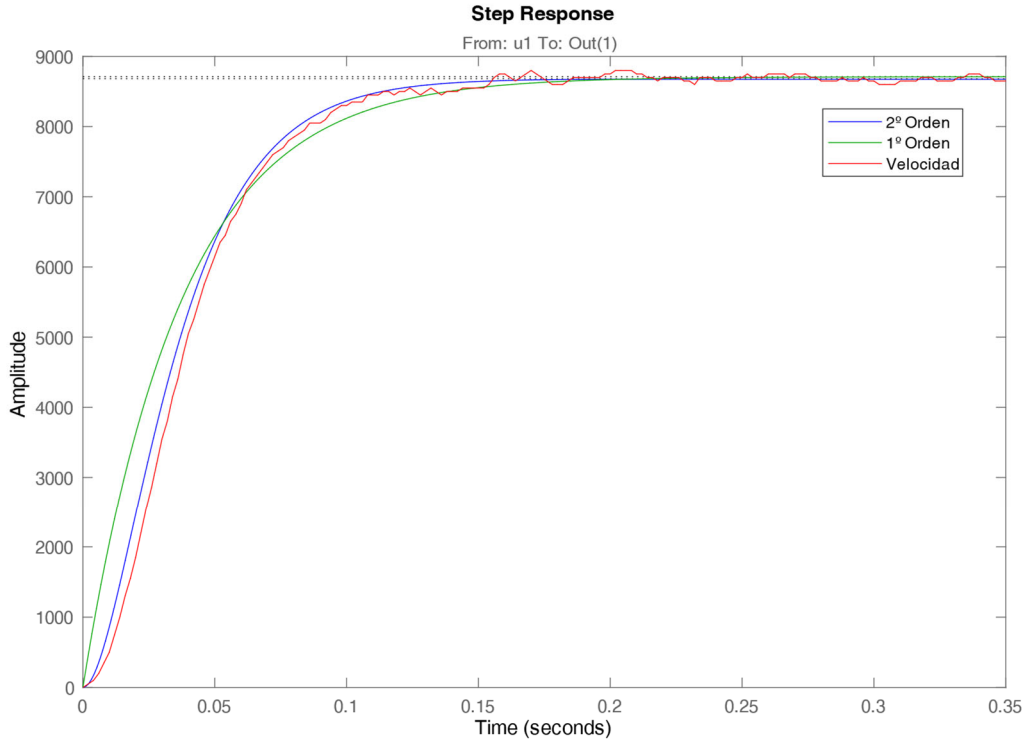
Para poder realizar un control preciso de los motores se procedió a su identificación en primera instancia. La identificación del motor permite conocer su comportamiento ante variaciones en la entrada y con ello poder ajustar la salida acorde con las necesidades.

Para la identificación se procedió a conectar un motor a una de las placas de control. Dicha placa se alimentó a la tensión de funcionamiento y se mandó un escalón al motor. El escalón mandado corresponde a cambiar la alimentación de entrada del motor desde 0V a 24V, lo cual se consigue mandando una orden lógica de PWM completo (5V) desde el controlador.

Desde el entorno de programación, el PWM completo corresponde a “analogWrite(Value);” siendo “Value=255”, por lo que se tomará “Value” como valor de entrada al sistema y la velocidad del motor como valor de salida.

Con estos datos de entrada, se obtiene la salida representada en rojo en la siguiente figura. La respuesta corresponde a la velocidad, cuyas unidades son pulsos por segundo. A su vez se representan el sistema de primer y segundo orden identificados, y como se puede apreciar, la aproximación de segundo orden es mucho más exacta, por lo que el motor quedará definido por la siguiente función de transferencia.

$$\boldsymbol{G(s) = \frac{9590}{s^2 + 108.5s + 2817}}$$



Bucle de control

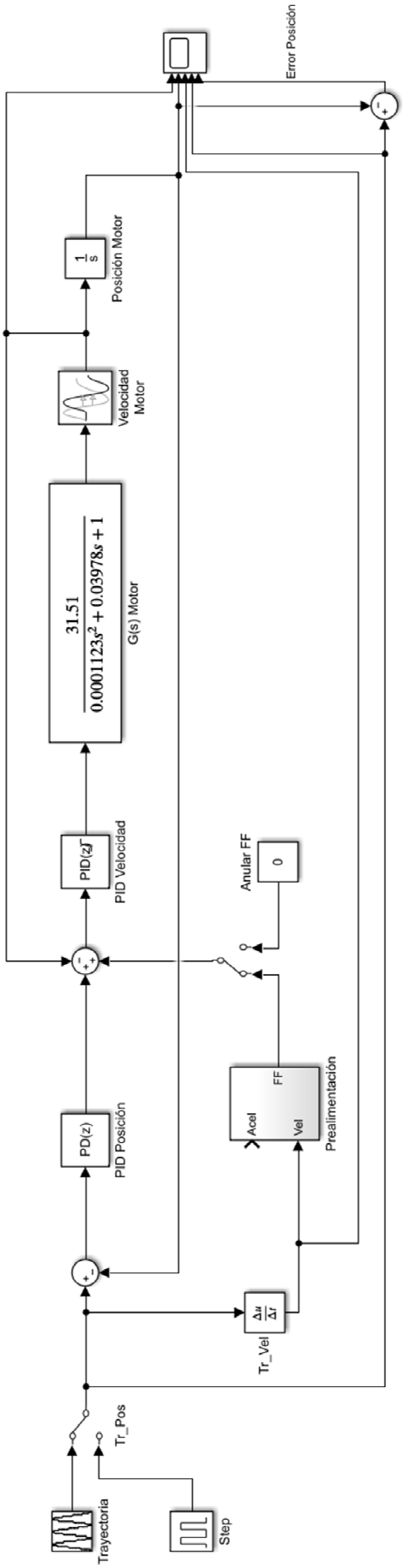
Una vez identificado el motor se puede proceder a diseñar un bucle de control para controlar de forma precisa el movimiento de éste. Para ello es necesario tener en cuenta de qué realimentaciones se dispone, la entrada introducida al sistema y qué objetivo se desea obtener.

Partiendo de la entrada, se introducirá una trayectoria completa en posición del motor. Dicha trayectoria estará previamente limitada en velocidad y aceleración acorde con el desempeño del motor, por lo que en condiciones normales el motor será capaz de seguirla.

Como realimentación se cuenta con un encoder incremental por lo que es posible conocer la posición exacta del motor. El procesador a su vez es capaz de calcular la velocidad del motor a partir de la variación de pulsos por unidad de tiempo, pero al tener pulsos finitos, la medición rápida de la velocidad disminuye la precisión en la medida.

La salida del sistema será la velocidad del motor, ya que es la variable que se puede regular, sin embargo, el objetivo último será garantizar la posición según la trayectoria, por lo que se deberá ajustar la velocidad para que esto suceda.

Con todo esto en mente, se procede a diseñar un sistema de control que permita una alta flexibilidad y minimice el error de posición durante una trayectoria. El sistema propuesto se muestra a continuación.

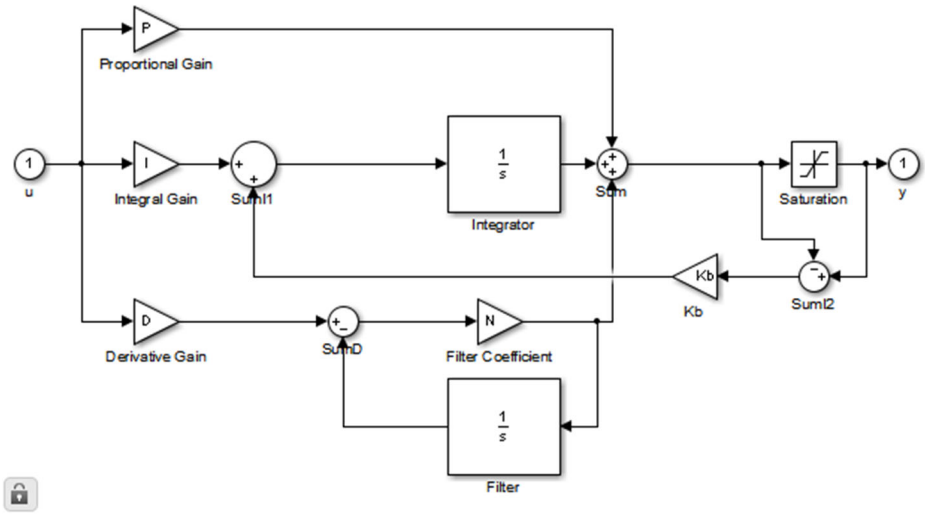

Error Posición
Posición Motor
Velocidad Motor
G(s) Motor
PID Velocidad
Anular FF
0
PID Posición
FF
Acel
Vel
Prealimentación
Tr_Vel
Tr_Pos
Trayectoria
Step

Como se puede observar, el sistema cuenta con dos reguladores, uno de velocidad y otro de posición. El regulador de posición tiene como entrada directamente la trayectoria enviada y se realimenta con la posición exacta en pulsos de encoder. Por otro lado, el regulador de velocidad tiene como entrada la salida del regulador de posición sumada a una prealimentación de velocidad que se obtiene directamente a partir de la derivada de la trayectoria. Dicha prealimentación se volverá extremadamente importante a la hora de seguir una referencia variable, no tanto en la respuesta ante escalón. La realimentación del regulador se obtiene indirectamente a partir de la variación de los pulsos de encoder, por lo que, al ser limitada la cantidad de pulsos, la medida de velocidad puede tener cierta incertidumbre. La salida del regulador de velocidad entra directamente a la función de transferencia del motor, de la cual se obtiene la velocidad de salida.

En el esquema desarrollado en Simulink es posible intercambiar la entrada entre una trayectoria generada en Matlab o un Step para realizar ajustes en los controladores. A su vez, los valores se representan en un Scope lo cual permite ajustar el sistema de manera sencilla. A continuación, se explicará cómo se procedió para ajustar cada controlador.

- **Control de velocidad**

El esquema básico de un regulador PID en Matlab es similar al que se muestra a continuación:



En él se pueden apreciar la forma de actuación de los valores P, I y D, así como el filtro derivativo y el filtro anti-windup.

Para el regulador de velocidad es necesario tener en cuenta que, pese a que la entrada tenga un valor muy elevado, la salida siempre estará acotada ya que la señal más elevada en valor absoluto que se le puede enviar al driver del motor (G(s)) es “255”, por lo que la salida entrará en saturación para valores mayores a 255 e inferiores a -255.

Para ajustar el regulador se procede a realimentarlo con la señal correspondiente e inyectar directamente un escalón. En primera instancia se utiliza la herramienta PID Tuning de Matlab intentando minimizar el tiempo de establecimiento controlando la sobreoscilación. Finalmente se procede a ajustar manualmente de manera más precisa para intentar mejorar la respuesta.

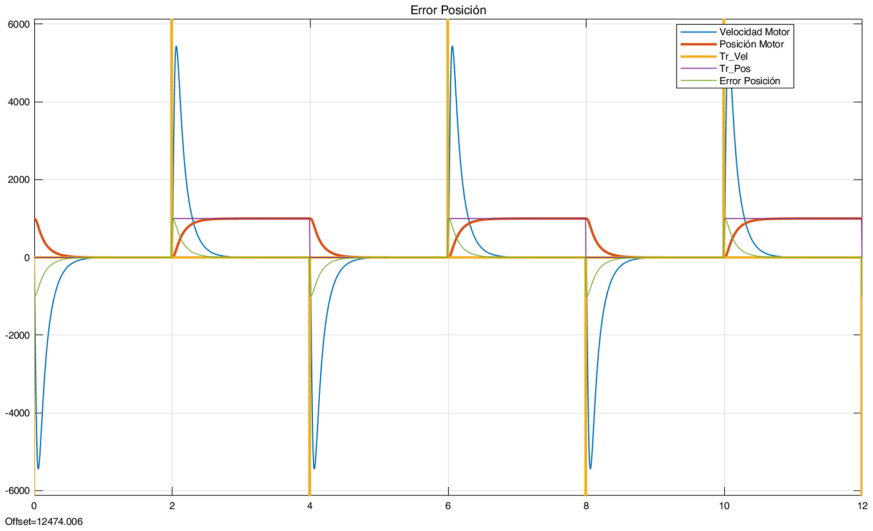
Mediante este procedimiento quedan ajustados los valores P, I y D.

- **Control de posición**

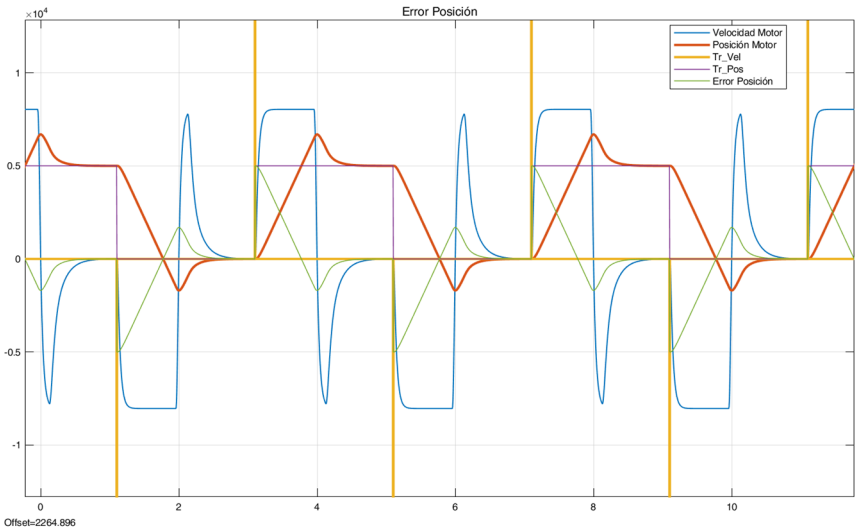
Con el regulador anterior se consigue llevar la velocidad del motor a la deseada. Si el control fuera perfecto, por sí solo el regulador de velocidad podría garantizar la posición también, pero el sistema real se aleja del ideal. Para garantizar la posición se añade un nuevo regulador mucho más sencillo que el anterior. Para ajustarlo se realiza el mismo procedimiento que con el regulador de velocidad, llegando a la conclusión que la influencia de I y de D son despreciables, por lo que se ajusta un regulador tipo P. En este caso no existe saturación y, por lo tanto, no es necesario filtro anti-windup.

- **Ajuste anti-windup**

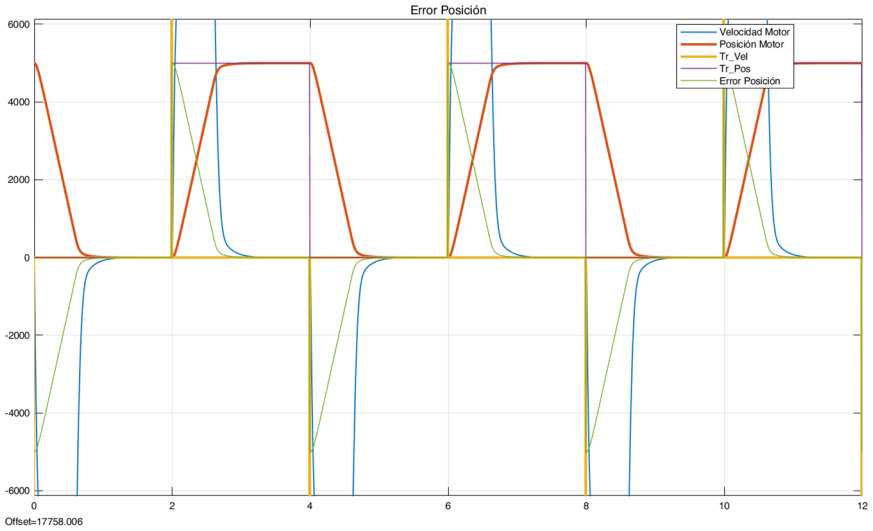
En este momento se procede a mostrar el comportamiento del sistema ante una entrada (Tr_Pos) tipo escalón de valor 1000 unidades (pulsos/s). Como se puede apreciar en la figura inferior, el motor adquiere rápidamente su posición sin sobreoscilar tal y como se desea.



Pero representándolo de nuevo ante un escalón de valor 5000 unidades, se puede ver como ahora se sobrepasa el valor deseado de posición, lo cual es debido a la parte integral que ha acumulado demasiado error ya que entrada al sistema del motor está saturada. Para solucionarlo se procede a ajustar el filtro anti-windup.



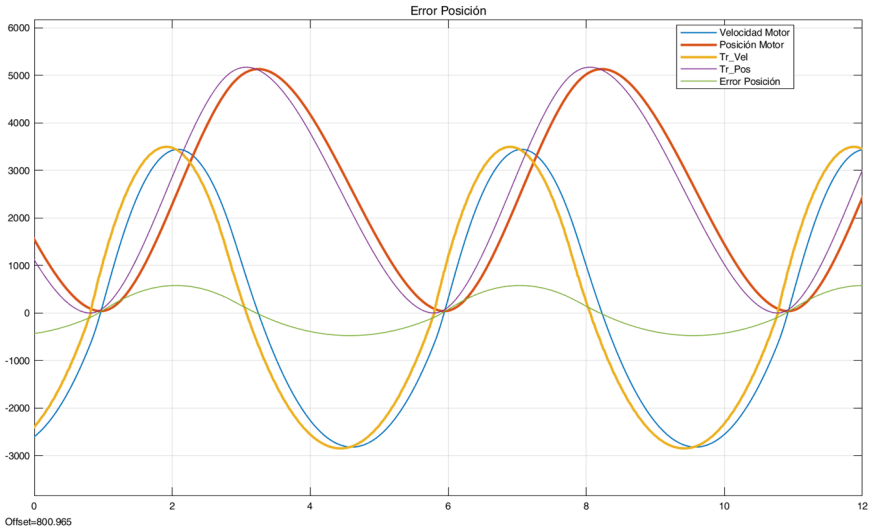
Ajustándolo experimentalmente, con la misma entrada que la imagen anterior, se consigue el siguiente resultado:



Con ello queda ajustado el valor Kb del filtro anti-windup.

- **Ajuste prealimentación**

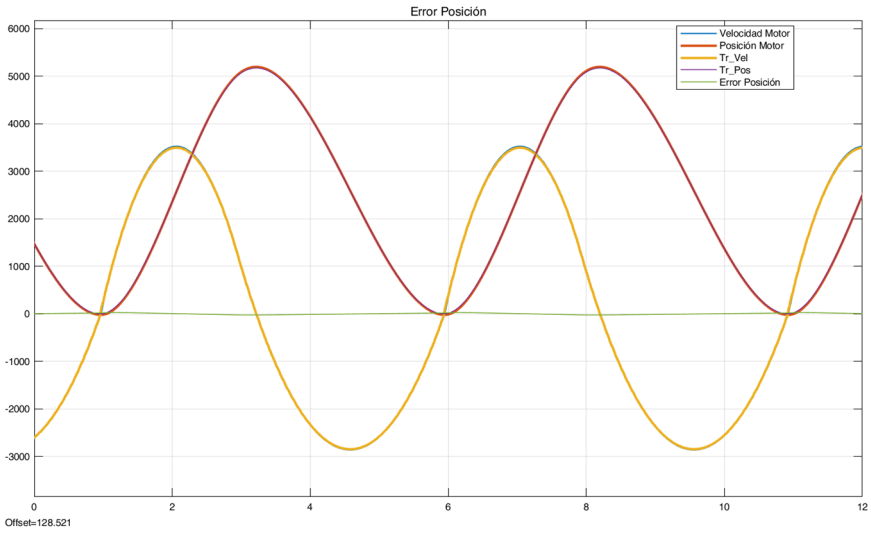
Como se ha podido observar el regulador ha quedado correctamente ajustado ante entrada escalón, pero la entrada real no será un escalón sino una trayectoria que será necesario seguir, por lo que se procede a simular el comportamiento actual del sistema ante una trayectoria a la entrada.



Tal y como muestra la figura, la salida tiene un desfase bastante importante tanto en velocidad como en posición, provocando que el error sea elevado durante el movimiento. Este hecho es alarmante ya que la desviación con respecto a la trayectoria provocará desincronismos entre los motores y tensiones indebidas en los cables y estructura del robot.

Para minimizar notablemente este problema, se procede a prealimentar el regulador de velocidad con la velocidad teórica de la trayectoria de entrada en cada instante. Esto permitirá al sistema “anticiparse” al movimiento, aumentando la precisión de movimiento.

Realizando únicamente realimentación de velocidad se consigue el siguiente resultado:



Como se puede observar, el error de posición se ha reducido notablemente al igual que el desfase. En este momento, el sistema es capaz de seguir la trayectoria prácticamente de manera perfecta, por lo que se considera que ha quedado correctamente ajustado.

A continuación, se detallan las tablas con los valores de ajuste de cada regulador.

- **PID Velocidad**

Parámetro	Valor
P	0.04137
I	1.0842
D	0.0001
N	72.5163
UpperLimit	255
LowerLimit	-255
Kb	8
Sample Time	0.005 s

- **PID Posición**

Parámetro	Valor
P	5.9476
Sample Time	0.005 s

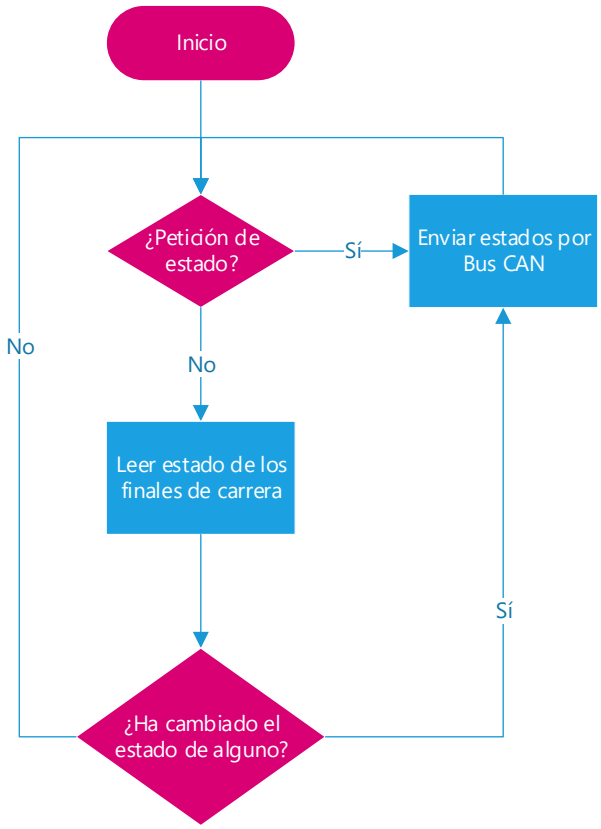
- **Prealimentación**

Parámetro	Valor
Kv	1
Ka	0

Esquema interno placa FdC

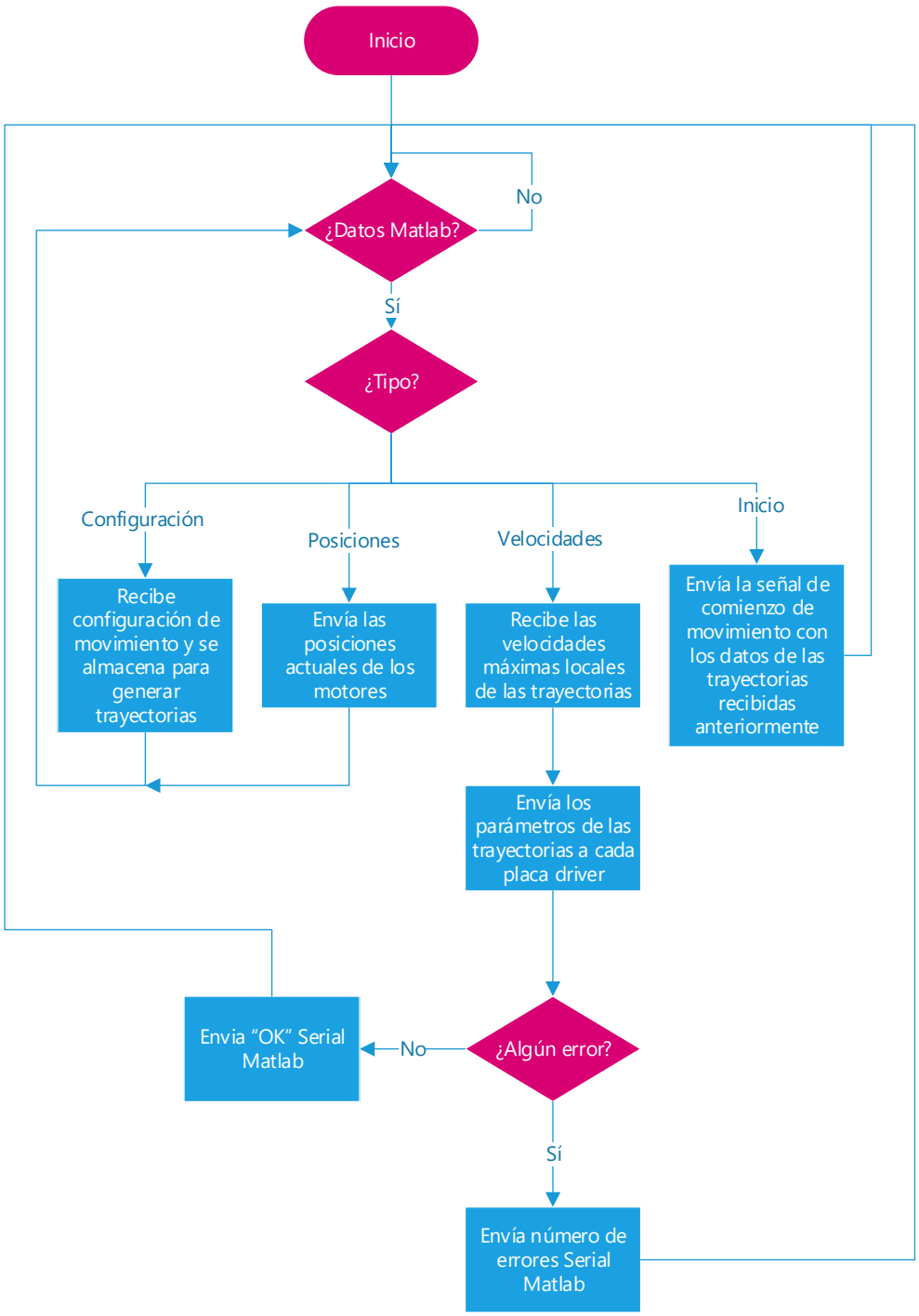
El esquema de la placa de finales de carrera es muy sencillo ya que su único cometido es leer de forma continua el estado de los finales de carrera y detectar cuando alguno ha cambiado de estado. En ese momento la placa debe actualizar y enviar el estado de todos los finales de carrera mediante bus CAN para que las placas correspondientes lo reciban.

A su vez la placa debe responder inmediatamente a las peticiones de estado de los finales de carrera enviando el estado de estos mediante bus CAN.



Esquema interno placa Main

El objetivo de la placa principal es crear una interfaz entre los datos recibidos por Serial desde Matlab y el bus CAN utilizado por todas las placas del robot. Dicha placa procesa las solicitudes de Matlab y actúa en consecuencia. A su vez también sirve como placa de depuración para detectar posibles errores de comunicación entre placas o entre Matlab y la propia placa.



Esquema Matlab

El esquema inferior representa la lógica que sigue la interfaz grafica de Matlab para procesar los datos de entrada y transformarlos en los movimientos deseados. La interfaz gráfica se explica en el apartado de “Programas y particularidades” y ayudara a entender el esquema abajo representado.