E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Análisis y programación de un controlador comercial para motores brushless

Grado en Ingeniería
en Tecnologías Industriales

Trabajo Fin de Grado

Autor: Rubén Otazu Bara

Director: Eugenio Gubia Villabona

Pamplona

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Agradecimientos

Me gustaría agradecer el apoyo tanto a mi tutor en la empresa Noxon, Iker Apalategi, como al director del proyecto, Eugenio Gubia por la por el apoyo técnico y la colaboración desinteresada en el presente proyecto.

También agradecer el apoyo incondicional de mi familia.

Abstract

This project aims to control a brushless motor through a commercial driver, called VESC, to control the speed of movement of a vehicle on where cameras are mounted.

In this project it has been sought that the speed of the camera is stable and generates sufficient torque within a range of operation of electric speed from 0 to 8000 rpm. The solution currently available does not allow reaching speeds below 200 rpm in the optimal torque conditions, which is an impediment for this type of application. For this, the viability of using an alternative control technique based on position references has been studied. In order to implement this strategy, it has been necessary to modify the VESC firmware, which, in turn, has required understanding how it currently operates and how it is programmed.

Finally, the desired objective has been achieved and has been validated through the tracking of some speed profiles.

Resumen

Este trabajo fin de estudios tiene por objeto el control de un motor brushless a través de un variador comercial, denominado VESC, para controlar la velocidad de desplazamiento de un vehículo sobre el que se montan cámaras de grabado y fotografía.

En este proyecto se ha buscado que la velocidad de la cámara sea estable y genere par suficiente dentro de un rango de operaciónde velocidad eléctrica de 0 a 8000 rpm. La solución actualmente disponible no permite alcanzar velocidades por debajo de 200 rpm en las condiciones óptimas de par, lo que supone un impedimento para este tipo de aplicación. Para ello, se ha estudiado la viabilidad de utilizar una técnica de control alternativa basada en referencias de posición. Con objeto de implantar esta estrategia, ha sido necesario modificar el firmware del VESC, lo que, a su vez, ha requerido comprender cómo opera actualmente y cómo se programa.

Finalmente, se ha conseguido el objetivo deseado y se ha validado a través del seguimiento de unos perfiles de velocidad.

Palabras clave: Motor brushless, VESC, Control de posición, Thread, PID.

Índice

1. Introducción y objetivos 1
   1.1 Introducción 1
   1.2 Objetivos 2
2. Descripción del sistema 4
   2.1 Motor brushless y encoder magnético 5
      2.1.1 Motor brushless 5
      2.1.2 Encoder magnético 7
   2.2 Controlador VESC 8
      2.2.1 Hardware 8
      2.2.2 Firmware 10
   2.3 Microcontrolador STM32F103T6 14
   2.4 Mando de control remoto y módulo inalámbrico 15
3. Control de velocidad por medio de referencias de posición 16
   3.1 Introducción a la programación con threads 16
   3.2 Generación de movimiento 18
      3.2.1 Generación a paso constante 22
      3.2.2 Generación a frecuencia de actualización constante 25
      3.2.3 Función generadora de movimiento 27
      3.2.4 Tratamiento de las consignas de velocidad 29
      3.2.5 Problema del bloqueo externo del motor 29
      3.2.6 Uso de los dos modos de establecimiento de la velocidad 34
      3.2.7 Funcionamiento bajo rozamiento 38
4. Configuración del PID de posición 41
   4.1 Ajuste del proporcional, k_p 41
   4.2 Ajuste del derivativo, k_d 42
   4.3 Ajuste del integrador, k_i 43
5. Validación 46
6. Conclusiones y líneas futuras 50
   6.1 Conclusiones 50
   6.2. Líneas futuras 51
7. Referencias y bibliografía 52
   7.1. Referencias 52
   7.2 Bibliografía 52

1. Introducción y objetivos

1.1 Introducción

La empresa Creativitytech desarrolla productos electrónicos de la marca propia Noxon para mejorar la experiencia audiovisual de grabado y fotografía. Los usuarios de estos aparatos van desde empresas de realidad virtual hasta aficionados a la fotografía. Se trata de vehículos sobre los que va montada la cámara y de manera remota, a través de un mando, el usuario selecciona los parámetros del movimiento que se ejecuta mientras la cámara graba o saca fotos. Estos aparatos pueden trabajar a tiempo real, donde los parámetros del movimiento son modificables en todo momento o en modo automático, donde una vez elegidos estos no se pueden modificar hasta el final del recorrido.

Figura 1: Slider & Dolly, producto de la marca Noxon.

Un ejemplo claro es el vehículo *Slider & Dolly* de la figura 1. Como se puede observar, a través del mando se dirige la dolly, el vehículo con ruedas donde va montada la cámara que circula por encima del slider, el conjunto de cable de sujección con las barras horizontales y los soportes.

Hasta ahora, la mayoría de estos productos de la marca Noxon han utilizado sistemas basados en motores paso a paso. Únicamente la *High Speed Cablecam* y la *Mantis*, que son productos de la marca Noxon, utilizan un motor brushless, que es con el que se trabajará en el presente trabajo. La *High Speed Cablecam* (Figura 2) consiste en un soporte móvil para cámaras formado por tres poleas y una estructura de acero que va sujetada a una cuerda, sobre el cual se desplaza. Este movimiento se realiza en ambos sentidos. La *High Speed Cablecam*, como su propio nombre indica se trata de un vehículo que puede trabajar a altas velocidades, superando ampliamente los 30km/h y

es capaz de mover hasta 20kg de carga. Ambos requerimientos llevan al uso de un motor brushless en lugar del paso a paso del resto de los productos.

Figura 2: High Speed Cablecam, producto de la marca Noxon.

Por otro lado está la *Mantis*, un producto aún en desarrollo que también hace uso de un motor brushless. Se trata de un coche eléctrico de medio metro de largo dirigido de manera inalámbrica sobre el que va montado una cámara de fotos o de vídeo. En este caso, el motor brushless servirá para mover las ruedas traseras y generar el desplazamiento del vehículo. La *Mantis* alcanzará los 10km/h y tendrá una capacidad de carga de 10 kg.

Ambos productos tienen usos similares: la realización de time-lapse, técnica de fotografía que permite captar sucesos inapreciables para el ojo humano sacando una gran cantidad de fotografías en un lapso de tiempo muy corto, y grabaciones a todo tipo de velocidad con recorridos marcados en tiempo real o preestablecidos y que se puedan memorizar y reproducir posteriormente. Para todo ello, la precisión y la capacidad de trabajar en un amplio rango de velocidades son factores primordiales a la hora de controlar el motor brushless. Tanto la posición como la velocidad deben ser controladas de manera precisa para que la experiencia de grabación y y de fotografía sea la más adecuada posible a las preferencias del usuario. Está previsto también trabajar en planos inclinados y, dada la gran carga que se debe soportar será necesario generar un par grande en esta situación. Queda por tanto marcada la necesidad de conseguir un par suficientemente grande en todo el rango de velocidades incluyendo la situación en la que el vehículo queda parado, donde debe contrarrestar su peso mediante el par generado a velocidad de referencia 0.

Los motores brushless utilizados son controlados por una controlador VESC, descrito en apartados posteriores, que ejecutan un firmware modificable.

1.2 Objetivos

Este proyecto tratará de realizar un control de la velocidad del motor brushless de la *High Speed Cablecam* y de la *Mantis* de una manera muy precisa. De esta manera, se pretenderá que las velocidades sean fluidas y continuas y proporcionen el par adecuado. Además, se espera que abarque un amplio rango de valores de velocidad, respondiendo de manera precisa tanto a movimientos rápidos como lentos. La velocidad máxima que

se ha marcado son las 8000 rpm eléctricas. A partir de momento, siempre que se haga referencia a la velocidad se hará en unidades de rpm eléctricas.

El objetivo estará primero en analizar y programar el firmware de la controladora VESC del motor brushless. No se tratará, por tanto, únicamente de crear nuevo código sino también de modificar el existente. El segundo objetivo será ajustar un PID de posición necesario para la generación del movimiento. El último objetivo será la validación del sistema a través del seguimiento de unos perfiles de velocidad.

La programación del firmware se realizará en el entorno ChibiOS/RT y en lenguaje C/C++. Además, se utilizará la aplicación VESCTOOL que servirá para cargar el firmware modificado al controlador VESC del motor, para sacar gráficas de posición y velocidad a lo largo del tiempo y para modificar los parámetros del PID.

2. Descripción del sistema

El sistema utilizado para realizar el control de motor brushless resulta de la unión de varios elementos: motor brushless, controlador VESC, microcontrolador STM32F103T6, un módulo inalámbrico y un mando de control remoto. Está organizado de la siguiente manera. El mando de control comunica los parámetros del movimiento elegidos por el usuario al microcontrolador STM32F103T6 a través del módulo inalámbrico. Este microcontrolador procesa esas directivas y prepara y envía las funciones correspondientes que son procesables por parte de la controladora VESC. Este envío se realiza a través del puerto serie, de tal forma que el puerto rx de lectura está conectado al tx de transmisión del otro y viceversa. El VESC, que está alimentado a 18.5V, recibe la información desde el puerto serie y, tras procesarla, genera las correspondientes corrientes en las fases del motor. De esta manera se genera el movimiento esperado por el usuario. El sentido de la información será el inverso cuando se quiera obtener información acerca del motor, por ejemplo, la posición o la velocidad a la que gira. En la Figura 3 podemos observar la cadena de información del primer caso.

Figura 3: Cadena de información del sistema en el caso de envío de directivas a través del mando.

El esquema de conexiones de cada uno de los elementos se refleja en la Figura 4.

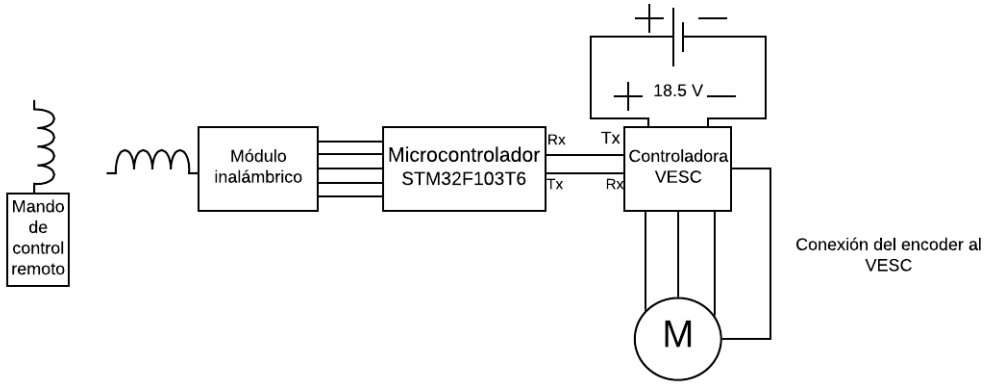

Figura 4: Conexiones del sistema.

2.1 Motor brushless y encoder magnético

2.1.1 Motor brushless

2.1.1 Partes y funcionamiento

Un motor brushless está compuesto por dos partes: el estátor bobinado y el rotor formado por imanes permanentes. Esta es la característica principal del motor brushless ya que, dado que es capaz de generar el campo magnético en el rotor mediante imanes, no precisa de escobillas, como su propio nombre indica. El bobinado del estátor está dividido en 3 partes independientes que constituyen cada una de las fases del motor. Cada fase tiene una separación de 120º en el espacio con la siguiente. La eliminación de las escobillas aporta robustez y bajo mantenimiento. Además, estos motores ejercen un par superior a los motores paso a paso, lo que resulta interesante para la aplicación que nos ocupa.

En la figura 5, se puede observar la imagen de un motor brushless en la que se aprecian los imanes del rotor y las bobinas del estátor. Este tipo de motores brushless en los que el rotor móvil se encuentra en el exterior y el estátor fijo en el interior se conocen como outrunner.

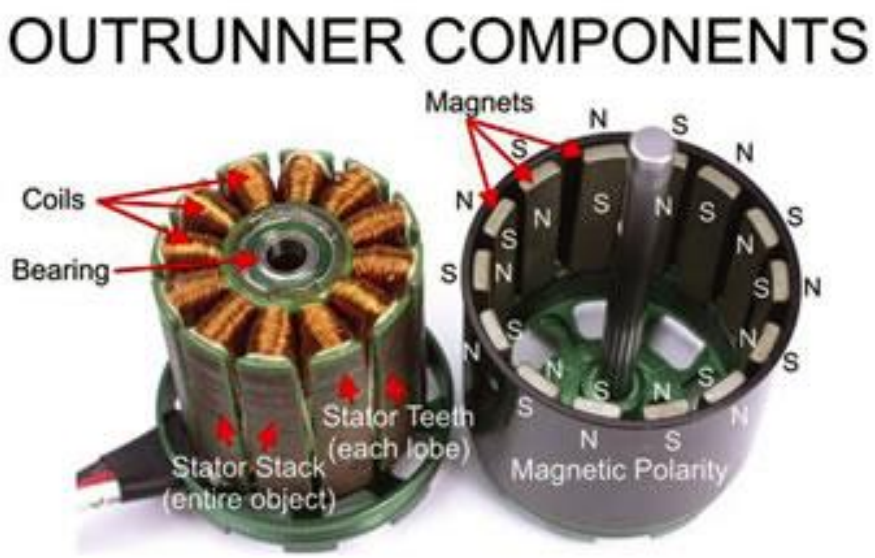

Figura 5: Imagen con las partes de un motor brushless.

El funcionamiento de este motor comienza con la excitación de una fase del motor. La bobina energizada generará un campo magnético. El rotor girará en busca de orientar la dirección de su campo en la misma que la del estátor. Así, se producirá el movimiento de giro del motor que se podrá mantener si se energiza la siguiente fase dejando de alimentar la anterior. La alimentación de estas bobinas se realiza a través del inversor de la figura 10 y el control de este inversor viene ya implementado en la tarjeta VESC. En el equipo que nos ocupa la posición del rotor se obtiene a través de un encoder magnético. Así, la secuencia de disparo del inversor será controlada por la tarjeta VESC.

En la figura 6 se muestra la conexión de el inversor con las 3 fases del motor (u,v,w).

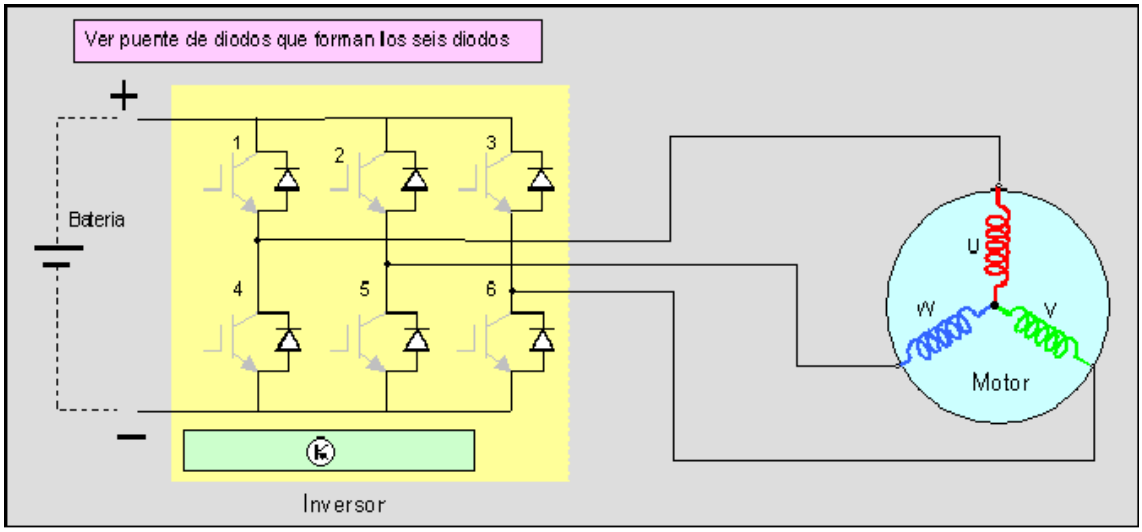

Ilustración 6: Esquema del motor brushless conectado al inversor.

La secuencia de disparo de los transistores del inversor generará un campo magnético rotatorio a una velocidad ω_e que provocará que el rotor gire a una velocidad mecánica ω_m. La relación entre ambas variables se refleja en la ecuación 1. Donde *p* corresponde al número de pares de polos del motor.

$$\omega_m = \frac{\omega_e}{p}$$

Ecuación 1.

2.1.1.3 Motor brushless utilizado

El motor brushless utilizado se corresponde con el modelo 6374 (Figura 7) y tiene las siguientes características

- 7 pares de polos
- 4400 W de potencia.
- 100 A de corriente máxima
- 192 kv: Valor del coeficiente (rpm/v) que nos indica la relación velocidad tensión de alimentación

Figura 7: Motor brushless 6374.

Como se ha mencionado anteriormente el uso de ese tipo de motor viene determinado por la necesidad de una potencia superior a los motores paso a paso del resto de productos, que contaban con un máximo de 300W. Además, unido al motor se coloca un encoder magnético absoluto que se encargará de medir su posición.

2.1.2 Encoder magnético

Para le medición de la posición se usa un encoder magnético, el AS5047P. Como se explica en su datasheet [1], este sensor está formado por un imán circular situado de manera que gira solidario y concéntrico al motor. Como se puede observar en la figura 8, los polos están situados de tal forma que la dirección del flujo variará con la rotación del motor. Varios sensores hall situados en posiciones que les permitan captar las variaciones en la dirección del flujo convierten la componente de flujo vertical a su superficie en voltaje. Estas señales son filtradas por el módulo AFE y convertidas en señales digitales a través del módulo analógico-digital. A partir de estas señales digitales, el módulo CORDIC es capaz de computar el ángulo y la magnitud del flujo magnético. Esta información es enviada al VESC a través del módulo de comunicaciones SPI, con una resolución de 14 bits. Todos los módulos mencionados están presentes en el diagrama de bloques de la figura 9.

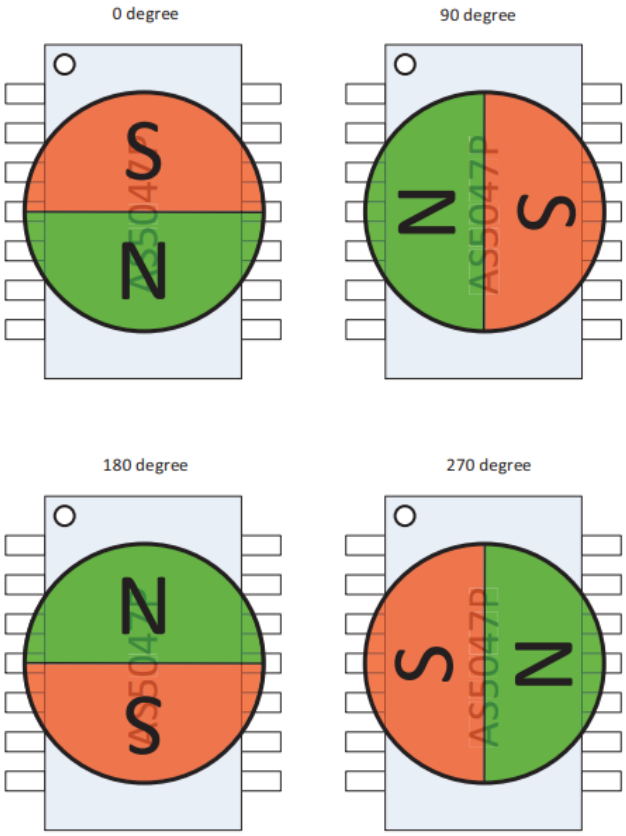

Figura 8: Posición de los polos del sensor magnético en determinados ángulos de rotación.

[1] ams AG, *AS5047P "High-Resolution Position Sensor Datasheet"*, 2016

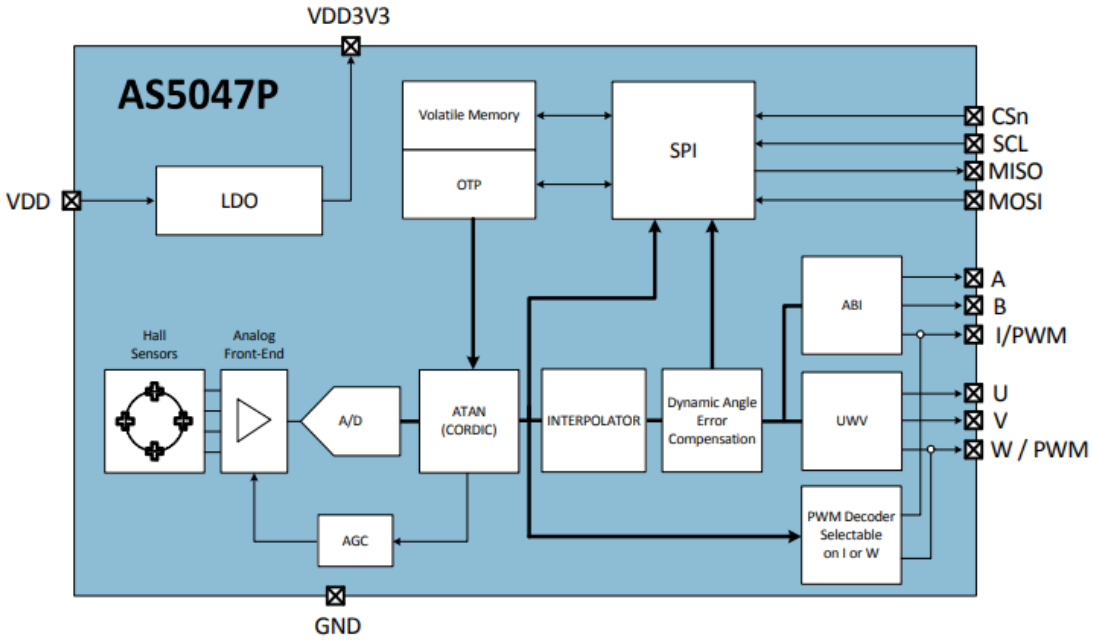

Figura 9: Diagrama de bloques del sensor de posición AS5047P.

El modelo del encoder magnético utlizado es el que cuenta con una sensibilidad de 0.1º, tiene una resolución de 14 bits y es capaz de medir velocidades de hasta 28000 rpm.

2.2 Controlador VESC

Las siglas VESC hacen referencia a Vedder, por el apellido del desarrollador, Electronical Speed Controller. Dado que es el elemento con el que más se va a trabajar en el presente proyecto, se realizará un análisis tanto de su hardware como del firmware que ejecuta su microcontrolador, el STM32F4. La distribución de los elementos se puede observar en las figura 10.

2.2.1 Hardware

La controladora VESC cuenta con una gran cantidad de elementos que contribuyen a su correcto funcionamiento. Por ello, en este apartado se analizarán los más importantes y aquellos que resultan esenciales para el sistema utilizado.

- Transistores IRFS7530: Tres parejas de transistores, cada una de las cuales controla una fase del motor. Junto con la batería forman el esquema típico de un inversor, como se puede observar en la figura 11. La evacuación del calor se realiza a la propia PCB, no existiendo radiadores.
- Microcontrolador STM32F4 que ejecuta el firmware explicado en el siguiente apartado. Cuenta con una memoria flash de 1 Megabyte y 168MHz de frecuencia máxima.
- Entradas de alimentación IN^+ e IN^- .
- Puerto HALL/Encoder encargado de recibir de uno de estos dos elementos la información referente a la posición del motor.

- Puerto PWR_COMM con los pines Tx y Rx importantes en la comunicación serie.

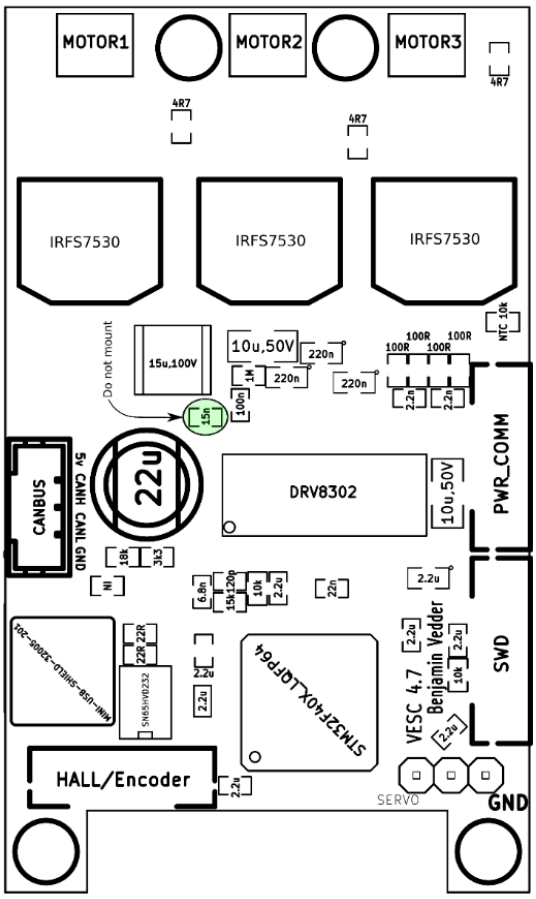

Figura 10: Cara superior del VESC.

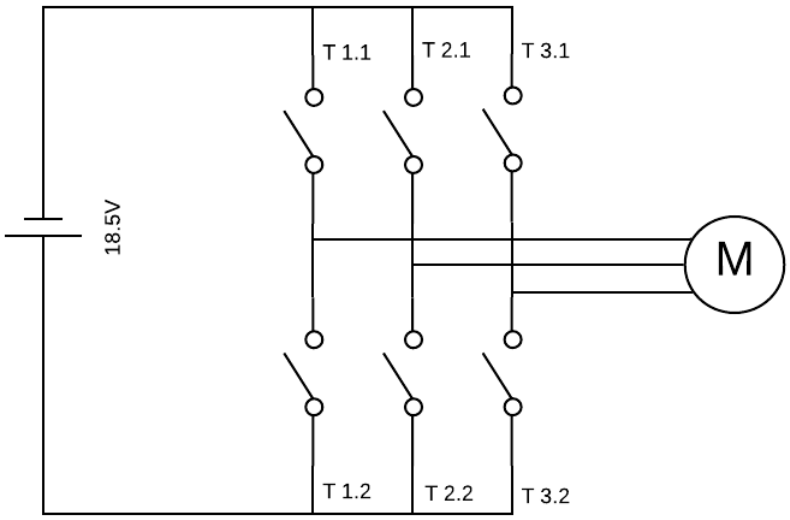

Figura 11: Inversr del VESC. La letra T hace referencia a cada transistor IRFS7530.

2.2.2 Firmware

El tratamiento y modificación del firmware va a ser el principal objeto del presente proyecto. Por ello, conviene analizar como funciona el mismo y mencionar que partes son las más importantes.

El firmware está dividido en aplicaciones. Estas aplicaciones no son más que estructuración del código y se ejecutará únicamente la parte del código correspondiente a la aplicación seleccionada. Como se puede observar en la figura 12 el main, programa principal y que se ejecuta tras el encendido del VESC, selecciona que aplicación se va a utilizar y únicamente se ejecutará el código correspondiente a esa aplicación. En esta situación el código del resto de aplicaciones no es accesible salvo que se genere un cambio de aplicación desde el main. Además, independientemente de la aplicación seleccionada, se puede acceder toda clase de funciones de alto nivel, que van desde control de la posición (SET_POS), corriente o velocidad hasta obtención de parámetros del motor como el nivel de la batería o la temperatura a la que se encuentra. Estas funciones, a su vez, controlan otras de bajo nivel como es el establecimiento del valor de la corriente que circulará en el motor (RUN_PID_CONTROL).

Tenemos 4 aplicaciones principales: la UARTCOMM, implementa la comunicación serie, la ADC, la conversión analógico-digital, la PPM, la generación de señales de modulación por posición de pulso y la CUSTOM se trata de una aplicación en blanco, sin código, preparada para ser programada por cualquier usuario del VESC.

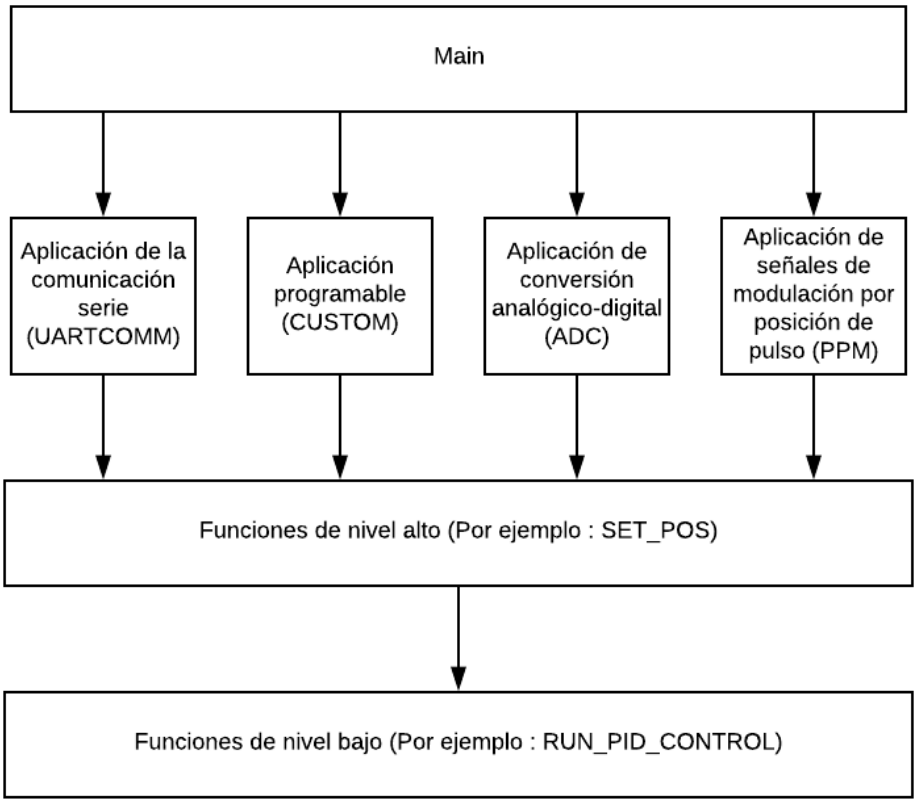

Figura 12: Estructuración del código del firmware del VESC.

Dado que solo se puede ejecutar una aplicación en cada momento y el cambio de una a otra produce una parada en el funcionamiento del VESC se decide trabajar únicamente con una aplicación. El proyecto se basará en el código de la aplicación uartcomm, que implementa el uso del puerto serie. Este código se copiará en el espacio en blanco de la aplicación CUSTOM. A partir de ese código y mediante modificaciones y creación de nuevo código se pretenderá conseguir el objetivo. Es decir, se modificará la aplicación CUSTOM que ejecutará el código que ya posee la aplicación que implementa la comunicación por el puerto serie pero además, se añadirá código que busque conseguir el objetivo deseado. Por tanto, será la aplicación uartcomm y su código la que se analice.

Esta aplicación es la encargada de implementar la comunicación serie a través de los pines rx y tx. De esta forma, recibe los mensajes, los procesa y genera una respuesta si es necesario. Para describir el funcionamiento de esta comunicación es conveniente explicar el protocolo de la misma, es decir, que forma tienen los mensajes y que se pretende transmitir con cada parte del mensaje enviado.

2.2.2.1 Protocolo de comunicación

El protocolo de comunicación serie define como debe ser el mensaje. En este caso, la trama de datos que se intercambia debe estar formado por las siguientes partes:

- Un byte de inicio de valor 2 para paquetes cortos (payload inferior a 256 bytes) y 3 para largos (payload superior a 256 bytes).
- Uno o dos bytes especificando la largura concreta de los datos de carga útil (payload).
- El payload del mensaje, la trama de datos útil. El primero de sus bytes es el indentificador del paquete, que especificará el tipo de dato o función a la que se hará referencia.
- Dos bytes de detección de errores a través del código CRC.
- Un byte que indica el final de la transmisión de valor constante 3.

2.2.2.2 Lectura y procesamiento del mensaje

Una vez se detecta la llegada del mensaje por el puerto serie se realiza la lectura del mismo. Como en toda comunicación serie el envío y recepción de mensajes se hace bit a bit alojando conjuntos de 8 bits (1 byte) en cada uno de los valores de un array llamado rx_data[a]. El procesado se realizará leyendo cada uno de estos bytes conforme indica el protocolo explicado previamente.

Si el valor del primer byte es 2 significa que el tamaño del payload es inferior a 256 bytes y que, por tanto, la variable que especifica su longitud puede ocupar un byte únicamente. Si el primer byte es de valor 3 significa que el tamaño del payload es superior a 256 y entonces la variable que especifica su longitud debe ocupar dos bytes.

En el primer caso se salta el primer byte de la pareja que define el tamaño del payload y se lee únicamente el segundo, alojándolo en una variable llamada payloadlength. En el segundo de los casos se leen los dos bytes y esa información se aloja en payloadlength.

Después se procede a la lectura del payload. Se lee byte a byte y se lleva la cuenta hasta que se llegue a leer tantos bytes como hay definidos en payloadlength. Todos estos bytes se alojan en el array de bytes payload.

Luego, se leen los dos bytes siguientes de detección de errores y se alojan en la variable crchigh el primero y en la crclow el segundo. Tanto crchigh como crclow son variables correspondientes al código de errores CRC. Por último, se lee el byte de finalización de la comunicación y si tiene valor 3, indicando que ha llegado el final de la transmisión satisfactoriamente, se realiza la detección de errores conforme al código CRC. Si es satisfactoria se confirma la llegada del paquete.

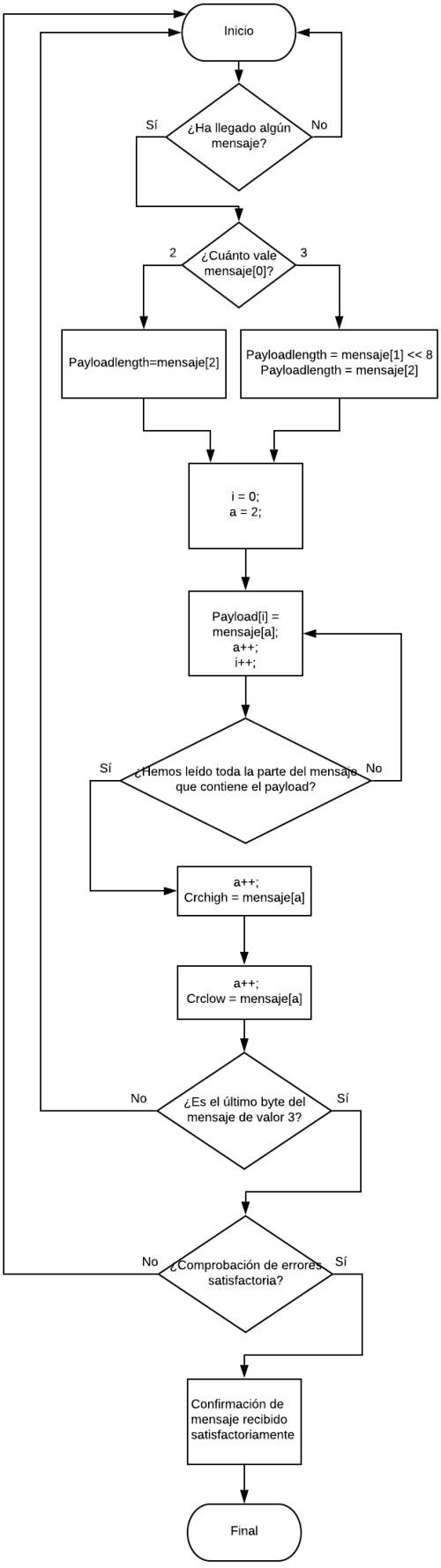

Figura 13: Diagrama de lectura y procesado de los mensajes.

2.2.2.3 Administración de la información del mensaje

Una vez leído y procesado el mensaje, es necesario gestionar toda esa información que se aloja en el array payload y que constituye la información útil del mensaje. Esta gestión se realiza a partir del primer byte de este array, llamado identificador de paquete, que indicará que acciones se deben ejecutar en función de su valor. Se elegirán conforme a la lista de la figura 14, de manera que el primer comando se seleccionará cuando ese primer byte valga 0, el segundo cuando valga 1, y así sucesivamente.

De la lista de la figura 14 destacan las acciones COMM_SET_RPM y COMM_SET_POS que sirven para ejecutar control de velocidad y de posición respectivamente a través de un PID. Su funcionamiento y el uso dado en el sistema se explicará mas adelante.

Se puede acceder, por tanto, a cada una de estas funciones a través de mensajes por el puerto serie.

```
COMM_FW_VERSION = 0,
COMM_JUMP_TO_BOOTLOADER,
COMM_ERASE_NEW_APP,
COMM_WRITE_NEW_APP_DATA,
COMM_GET_VALUES,
COMM_SET_DUTY,
COMM_SET_CURRENT,
COMM_SET_CURRENT_BRAKE,
COMM_SET_RPM,
COMM_SET_POS,
COMM_SET_HANDBRAKE,
COMM_SET_DETECT,
COMM_SET_SERVO_POS,
COMM_SET_MCCONF,
COMM_GET_MCCONF,
COMM_GET_MCCONF_DEFAULT,
COMM_SET_APPCONF,
COMM_GET_APPCONF,
COMM_GET_APPCONF_DEFAULT,
COMM_SAMPLE_PRINT,
COMM_TERMINAL_CMD,
COMM_PRINT,
COMM_ROTOR_POSITION,
COMM_EXPERIMENT_SAMPLE,
COMM_DETECT_MOTOR_PARAM,
COMM_DETECT_MOTOR_R_L,
COMM_DETECT_MOTOR_FLUX_LINKAGE,
COMM_DETECT_ENCODER,
COMM_DETECT_HALL_FOC,
COMM_REBOOT,
COMM_ALIVE,
COMM_GET_DECODED_PPM,
COMM_GET_DECODED_ADC,
COMM_GET_DECODED_CHUK,
COMM_FORWARD_CAN,
COMM_SET_CHUCK_DATA,
COMM_CUSTOM_APP_DATA,
COMM_NRF_START_PAIRING,
```

Figura 14: Lista de acciones seleccionables a través del puerto serie.

2.3 Microcontrolador STM32F103T6

El microcontrolador es el encargado de enviar las directivas procedentes del módulo inalámbrico al VESC. Posee una memoria flash de 120 Kilobytes y una frecuencia máxima de 75MHz. Cabe destacar la necesidad de este micro pese a la existencia del STM32F4 del VESC debido a la necesidad de acceso a todos los pines, acción que no es posible realizar en el entorno hardware del VESC. Los pines más destacables de este microcontrolador que se pueden observar en la figura 15 son los siguientes:

- Vcc y GND de alimentación
- Pines 25 (rx1) y 26 (tx1) importantes en la comuncación serie con el VESC
- Pines 4, 5, 6 y 13 encargados de la conexión de la tarjeta SD de grabado de los movimientos.

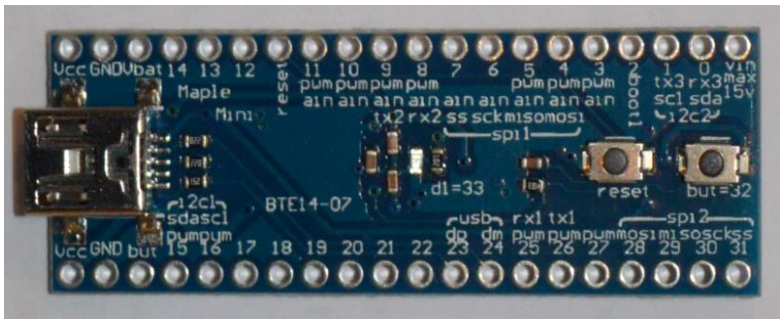

Figura 15 : Microcontrolador STM32F103T6.

2.4 Mando de control remoto y módulo inalámbrico

La comunicación inalámbrica se implementa a través de un mando de control remoto (figura 17) y un módulo inalámbrico (figura 16) conectado al STM32F10T6. El estudio, análisis o mejora de la misma no está contemplado en este proyecto. Por tanto, simplemente se hace uso de ella sin entrar en más detalle.

Figura 17: Mando de control remoto.

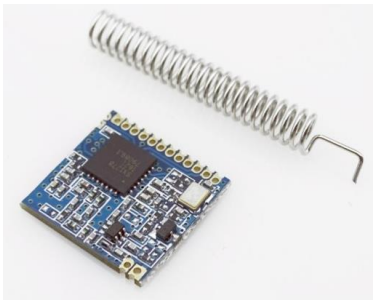

Figura 16: Módulo inalámbrico.

3. Control de velocidad por medio de referencias de posición

3.1 Introducción a la programación con threads

Como se explica en el manual del ChibiOS[2]. Un thread es una parte del código que se ejecuta de manera independiente y con un espacio propio de memoria. Con independiente se quiere dar a entender que los threads no dependen unos de otros y que se puede apagar uno de ellos o varios y el resto seguir funcionando correctamente. En caso de sistemas multithreading, el microcontrolador es capaz de ejecutarlos a la vez, en paralelo. En este proyecto, el microcontrlador del VESC tan solo es capaz de ejecutar un thread. Por ello, en el caso de coexistan varios threads en la misma aplicación deberá saltar de uno a otro cuando convenga. El criterio para realizar este salto se realiza en base a la prioridad que se da a los threads. De esta manera, primero se ejecutan los threads de mayor prioridad y cuando no haya ninguno de estos esperando se pasará al siguiente nivel de prioridad y así progresivamente. No obstante, siempre que haya un thread de prioridad superior listo para ejecutarse ese será el siguiente en hacerlo.

Los threads sirven para tener la posibilidad de realizar la programación de acciones como si fuera la única parte de código que la CPU ejecuta. De esta manera, la programación se vuelve mucho más sencilla y se pueden realizar acciones que de otra manera serían imposibles. Un ejemplo claro podría ser un robot industrial que recoge objetos de una cinta transportadora a la vez que realiza una perforación con láser en otros provenientes de otra cinta. Se podría programar un thread para cada una de las acciones sin tener en cuenta la existencia de la otra y, generando un salto de una a otra acción con una frecuencia suficientemente alta, daría esa impresión de estar ejecutándose las dos tareas a la vez.

En este proyecto se hará uso de los threads con todas las ventajas que ello conlleva. Sin embargo, hay que tener en cuenta la limitación de que, al no tratarse de un microcontrolador multithreading cada thread no puede estar ejecutándose un tiempo infinito, sino que es necesario parar uno para ejecutar los demás. Por ello, gana importancia el tiempo de espera en la atención a cada thread, es decir, el tiempo que tarda el microcontrolador desde que paramos el thread hasta que está listo para ejecutarlo de nuevo.

Todo esto se puede entender mejor a través de la figura 18 donde se observa de forma gráfica como se gestionan 3 threads en el caso de un sistema que no sea multithreading.

[2] Giovanni Di Sirio, “Chibios Full Reference Manual”, Salerno, Italia, 2016.

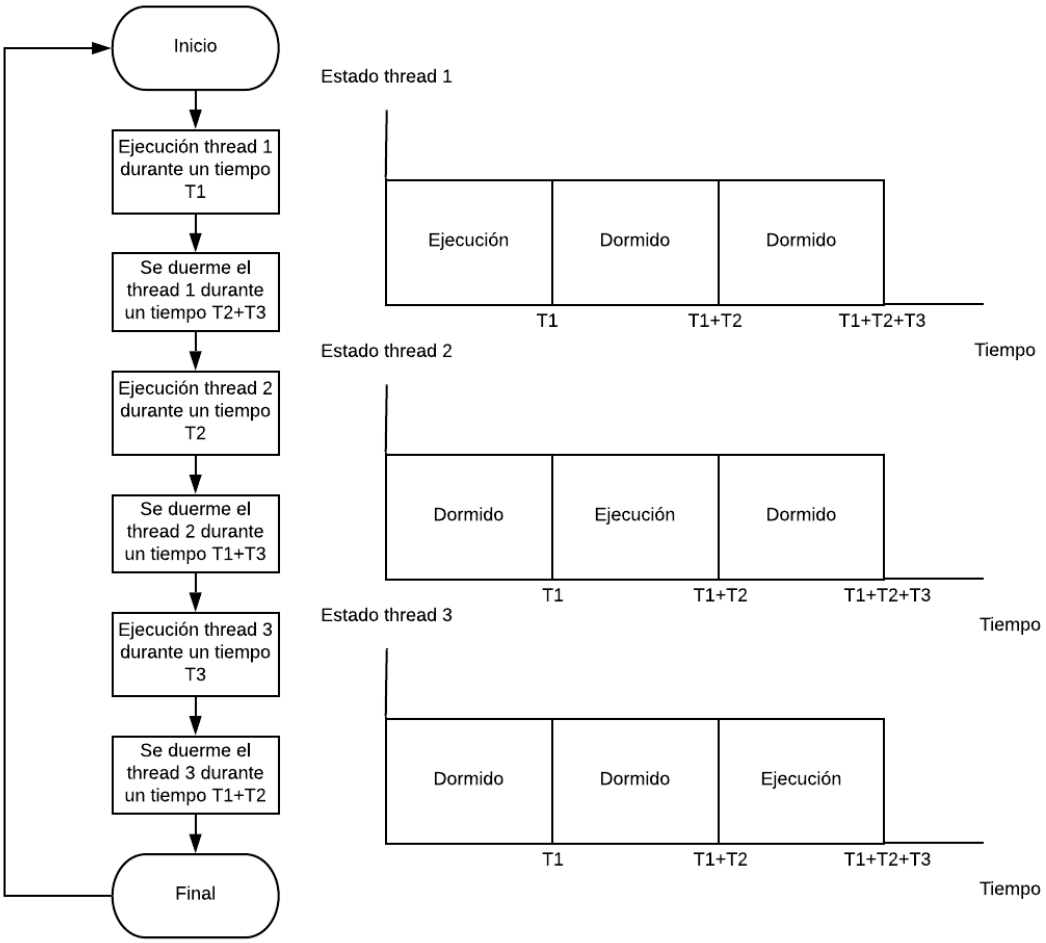

Figura 18: Gestión de 3 threads en el caso no multithreading.

3.2 Generación de movimiento

En el propio firmware del VESC ya existe una directiva capaz de generar una velocidad a partir de una consigna en el motor a través de un PID de velocidad. También existe una directiva capaz de controlar la posición del motor a través de un PID de posición. Ahora que se ha hecho referencia al PID conviene explicar en que consiste este sistema de control.

Un PID, cuyo esquema general se refleja en la figura 19, es un lazo cerrado de control utilizado en gran número de aplicaciones de la industria. El funcionamiento consiste en recibir la salida de una determinada planta o proceso y compararla con la referencia deseada, hallando el error. Este error se trata mediante una acción proporcional, una derivativa y una integradora cumpliendo cada una de ellas las siguientes funciones:

- Proporcional: Genera una salida resultado de multiplicar el error por una constante. Contribuye a la rapidez de respuesta. El coeficiente de esta acción es k_p.
- Integrador: Integra los errores a lo largo del tiempo por lo que disminuye el error en régimen permanente. El coeficiente de esta acción es k_i.
- Derivativo: Genera una salida proporcional a la derivada del error. Este elemento es el encargado de conocer la velocidad a la que la salida se acerca a la referencia y decelerar antes de que se sobrepase. El coeficiente de esta acción es k_d.

Las acciones calculadas por de cada uno de los elementos se suman y esta suma es la que llega a la planta provocando la evolución de su salida.

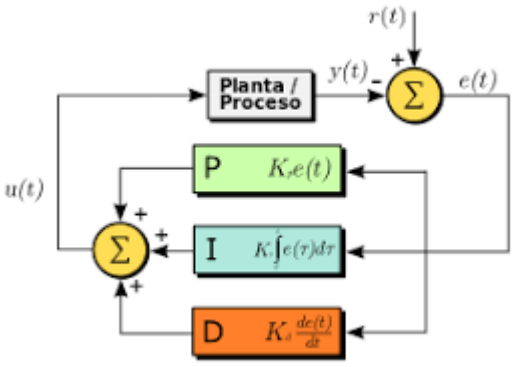

Figura 19: Esquema general de un control PID.

El diagrama de control del PID de velocidad queda reflejado en la figura 20. Su funcionamiento está basado en la generación de un error resultado de la diferencia entre la velocidad real y la velocidad referencia que se quiere alcanzar. Este error se trata mediante las mencionadas acciones proporcional, integral y derivativa que sumadas generan un par a la salida del PID. Este par es aplicado al motor y generará una aceleración inversamente proporcional a la inercia del motor(I). Esta aceleración integrada en el tiempo resultará ser la velocidad real del motor medida a través de un sensor y utilizada para calcular el nuevo error.

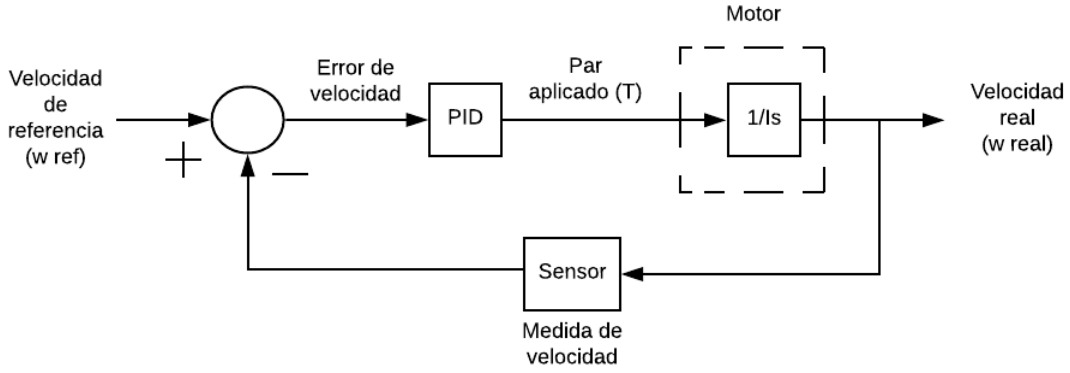

Figura 20: Diagrama del PID de velocidad.

Trabajando con este lazo de control de velocidad se encuentran bastantes limitaciones. Se encuentra la dificultad de ajustar unos coeficientes del PID de manera que funcionen suave a altas velocidades y sean capaces de generar corrientes suficientemente altas como para generar el par de arranque necesario o el demandado para ascender una pendiente a velocidades bajas en las que los errores son relativamente bajos. Para ello, haría falta una k_p tan alta que perderíamos la suavidad a altas velocidades. Con una k_p ajustada para que errores tan pequeños como los que podemos tener a bajas velocidades provoquen par suficiente para arrancar a velocidades bajas, cambios grandes de velocidad provocarán cambios de par muy bruscos, perdiendo suavidad.

Se podría pensar que la solución es realizar un ajuste dinámico de los coeficientes, de manera que a velocidades bajas actúen unos y a altas otros. Esta podría ser una solución pero probablemente la complejidad requerida para el desarrollo de ese sistema y la dificultad de encontrar unos coeficientes que garanticen suavidad a altas velocidades a la vez que una corriente suficientemente alta a bajas velocidades hagan de esta una solución poco recomendable. Los coeficientes que generan el par necesitado a bajas velocidades probablemente den como resultado un lazo inestable.

Por último, existe el problema del mantenimiento de una posición estable cuando se trabaja en plano inclinado. El PID de velocidad cuando esté parado a una velocidad de referencia de 0 rpm generará par nulo a causa del error cero. En esta situación en plano inclinado, la fuerza del peso no encontrará ninguna oposición y acelerará el vehículo. Esta aceleración generará una velocidad que cuando sea suficientemente grande provocará un error (la velocidad de referencia sigue siendo 0 rpm) que provoca que comience a generarse un par que contrarreste el peso. Este equilibrio en el que la suma de fuerzas sobre el vehículo es igual a 0 se dará a una velocidad determinada a la que se mantendrá indefinidamente si ninguna fuerza externa actúa sobre el. Por todas estas limitaciones se decide explorar otro método para el control de la velocidad. Los coeficientes del PID de velocidad quedan ajustados en los valores para los que Noxon calibraron un correcto funcionamiento a altas velocidades (k_p = 0,03 , k_d= 0,005, k_i = 0,0002).

Habitualmente lo que se suele hacer es implementar un control en cascada, como el que se refleja en la figura 21. Sin embargo, este lazo de control no está implementado en el firmware del VESC. Por tanto, se buscará una solución a través de un lazo de control que esté ya implementado dado que seguramente resulte más sencillo que desarrollar por programa uno nuevo. Este es el caso del PID de posición.

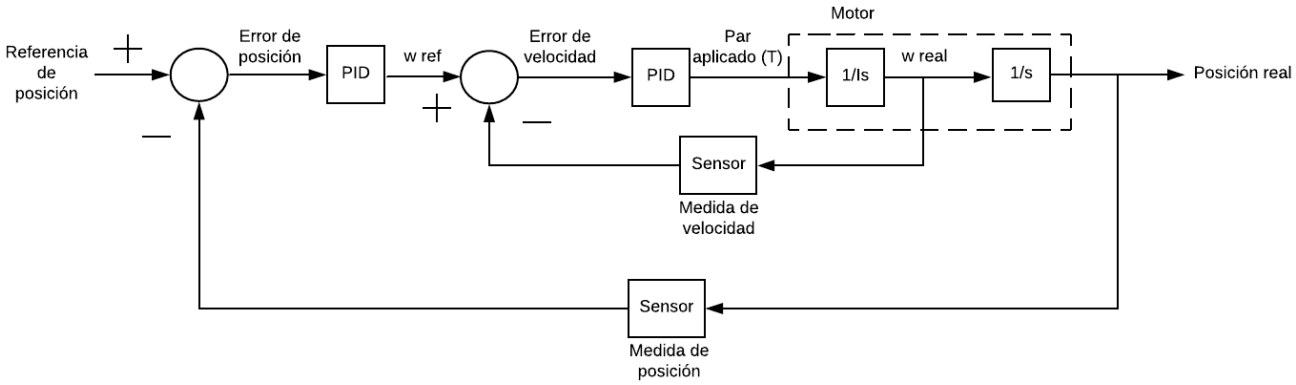

Figura 21: Diagrama de control de velocidad en cascada.

Se propone, por tanto, utilizar un lazo PID de control de posición, como el que se muestra en la figura 22 para generar el movimiento. El lazo de control PID de la posición está basado en la generación de un error resultado de la diferencia entre la posición real y la referencia de posición. Este error es tratado por las acciones proporcional, integral y derivativa de manera que sumadas generan un par a la salida que es aplicado al motor. Este par produce una aceleración inversamente proporcional a la inercia del motor. Esta aceleración integrada en el tiempo da como resultado la velocidad real del motor. A su vez, esta velocidad integrada de nuevo en el tiempo proporciona la posición del motor. Esta posición del motor es medida y utilizada para calcular el nuevo error.

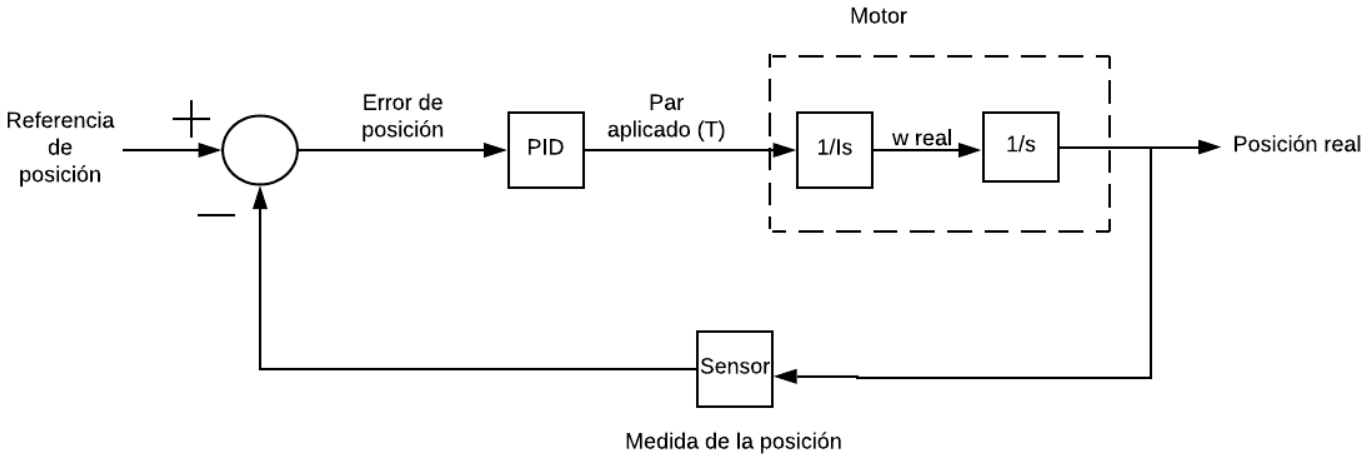

Figura 22: Diagrama del control PID de la posición.

El giro del motor generado a través de este PID de posición se conseguirá aumentando la referencia de posición un pequeño ángulo (paso) cada vez que se actualice la posición a través de una directiva nueva con una frecuencia lo suficientemente alta como para generar un movimiento continuo. De esta manera, se recibirá una consigna de velocidad a la que se adaptarán los parámetros de esa actualización del paso (largura de paso o frecuencia de actualización) para alcanzarla y mantenerla constante.

Este sistema solventa las limitaciones que tenía el trabajar con el PID de velocidad. Cuando se trabaja con el PID de posición para generar velocidades se conseguirá par incluso a las velocidades mas bajas. Esto ocurre porque el error y por tanto el par, ya no son directamente proporcionales a la velocidad como se puede observar en el diagrama de bloques de la figura 22. Ahora, el par es directamente proporcional a la diferencia entre la posición del motor real y la referencia de posición. Esta diferencia puede ser muy grande incluso a velocidades muy bajas donde el error de posición se acumulará hasta que sea suficiente como para generar el par de arranque y movimiento a velocidades bajas. El par se mantendrá constante a velocidades bajas siempre que el error de posición permanezca constante. La velocidad será una sucesión de estados de equilibrio, esto es, puntos en los que se produce un par suficiente como para llevar el motor de la posición real a la posición de consigna sean cuales sean las condiciones de velocidad a la que se produce el movimiento.

La generación del movimiento irá implementada en un thread con este único propósito. A la hora de plantear la solución encontramos dos posibles caminos, explicados a continuación.

3.2.1 Generación a paso constante

Este método consistiría en fijar un ángulo de paso constante y en hacer variar la frecuencia de actualización de la posición, es decir, el tiempo entre actualizaciones. Este tiempo de actualización es el tiempo que se esperará entre un envío de una directiva de posición y la siguiente a la que se habrá sumado el paso constante. Este tiempo vendría definido por la ecuación 2.

$$Tiempo\ espera\ (\mu s) = \frac{Paso \times 60 \times 7\ (Número\ de\ pares\ polos\ del\ motor) \times 10^6}{Velocidad\ de\ referencia\ (rpm) \times 360}$$

Ecuación 2.

Para favorecer el entendimiento de este modo de trabajo se dibuja la gráfica de la figura 23. En ella se pueden observar dos tramos de velocidades claramente diferenciadas por la pendiente. En el primer tramo (50 rpm), la pendiente es mucho menor que en el segundo (1100 rpm). También se puede observar como la distancia entre puntos (tiempo) es mucho mayor en el primer tramo que en el segundo. Además, la variación en posición del motor para cada punto (paso) permanece constante.

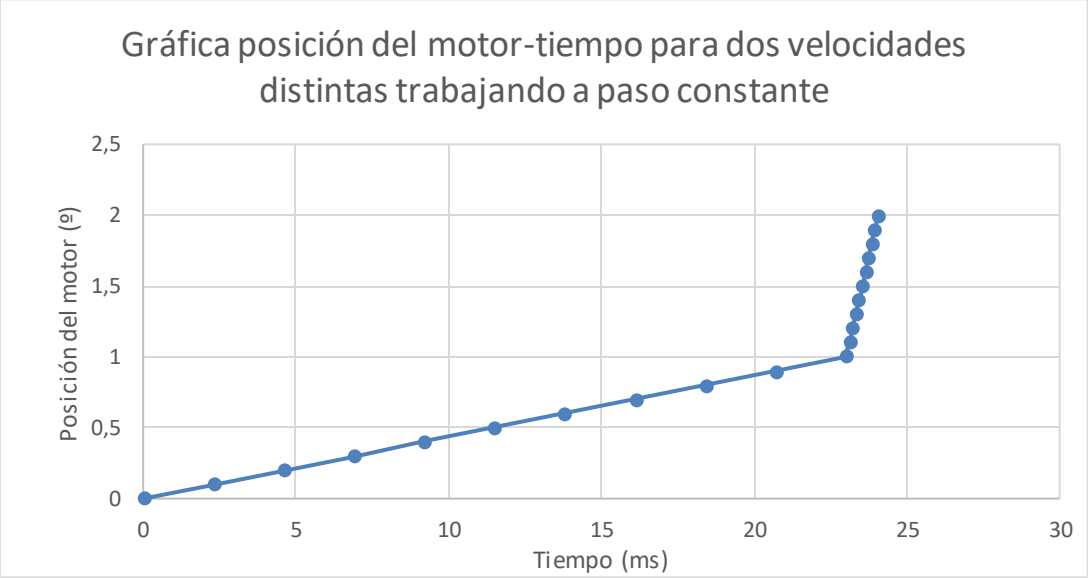

Figura 23: Gráfica de la posición del motor a lo largo del tiempo para velocidades generadas a paso constante. Dos tramos: a 50 rpm de 0 a 23 ms y a 1100 de 23 a 24 ms.

En este punto, comienza a ganar importancia el tiempo de atención al thread que limitará la frecuencia con la que podemos actualizar las referencias de posición. Esto significa que aunque se establezca en este thread un tiempo de espera muy pequeño si la ejecución del resto de threads ocupan un tiempo superior, será este tiempo el que marque la frecuencia de actualización y, por tanto, la velocidad.

Estableciendo en 0.1º el paso, que es la sensibilidad mínima del encoder se procede a realizar pruebas a diferentes velocidades. Los resultados quedan reflejados en la tabla 1.

Consigna de velocidad (rpm)	Velocidad real (rpm) y error con respecto a la consigna		Tiempo de ejecución teórico (µs)	Tiempo de ejecución efectivo(µs)
50	48	4.00%	2333.33	2400
75	73	2.67%	1555.56	1600
100	97	3.00%	1166.67	1200
150	146	2.67%	777.78	800
200	194	3.00%	583.33	600
250	234	6.40%	466.67	500
300	293	2.33%	388.89	400
400	390	2.50%	291.67	300
500	390	22.00%	233.33	300
600	583	2.83%	194.44	200
700	583	16.71%	166.67	200
800	583	27.13%	145.83	200
900	583	35.22%	129.63	200
1000	583	41.70%	116.67	200
1100	583	47.00%	106.06	200
1200	1165	2.92%	97.22	100
1500	1165	22.33%	77.78	100
2000	1165	41.75%	58.33	100
4000	1165	70.88%	29.17	100

Tabla 1: Muestreo de errores en la velocidad para el modo a paso constante.

Primero hay que destacar que dada la precisión de la gráfica a partir de la que se obtienen los datos no tiene sentido utilizar decimales ni en esa columna ni en la de tiempo de espera efectivo calculado a partir de ella.

Se pueden sacar varias conclusiones de los datos obtenidos reflejados en la tabla 1.

Se observa que la función que se utiliza para mandar al thread a dormir una determinada cantidad de microsegundos trunca el valor en las centenas de microsegundo. Es decir, tan solo es capaz de dormir el thread múltiplos de 100 µs. Esto significa que solo se pueden conseguir tiempos de ejecución múltiplos de 100 µs. Por ello, tan solo son alcanzables ciertas cantidades discretas de velocidades. Esta se trata de una limitación determinante, ya que para este sistema se busca poder acceder a cualquier tipo de velocidad dentro del rango establecido.

Además, observamos que no se puede disminuir el tiempo de ejecución por debajo de los 100 µs. Esto significa que con este método las velocidades superiores a 1165 rpm son inalcanzables con el paso que se ha fijado.

Por último, se tiene otra limitación. Existe una posible solución al útlimo problema comentado reescalando el paso. Fijando en 8000 rpm la velocidad máxima, en la que el tiempo de espera tiene que ser el mínimo posible (100 µs), se calcula el paso a fijar despejando en la ecuación superior. El resultado es que el paso constante debe ser como mínimo 0.686º. Llevando el caso al otro extremo con este valor y estableciedo una velocidad mínima de 50 rpm, se obtiene que en este caso el tiempo de espera será de

16007µs, unos 16ms. Se trata de una frecuencia de actualización excesivamente pequeña, que no explota totalmente las posibilidades del VESC. Dado que la importancia de este método recae sobre todo en funcionar lo mejor posible a bajas velocidades se decide explorar el siguiente método.

3.2.2 Generación a frecuencia de actualización constante

El segundo consistiría en fijar la frecuencia de actualización y en hacer variar la largura del paso. Este ángulo vendrá definido por la ecuación 3.

$$Paso\ (º) = \frac{Velocidad\ de\ referencia\ (rpm) \times Tiempo\ espera\ (\mu s)\ \times 360}{60\ \times\ 10^6\ \times 7\ (Número\ de\ pares\ de\ polos\ del\ motor)}$$

Ecuación 3.

Para favorecer el entendimiento de este modo de trabajo se dibuja la gráfica de la figura 24. En ella se pueden observar dos tramos de velocidades claramente diferenciadas por la pendiente como en el método anterior. Aquí también en el primer tramo (50 rpm), la pendiente es mucho menor que en el segundo (1100 rpm). Sin embargo, se puede observar como la distancia entre puntos (tiempo) es constante mientras que la variación en posición del motor para cada punto (paso) cambia de un tramo a otro siendo más pequeña en el primero que en el segundo.

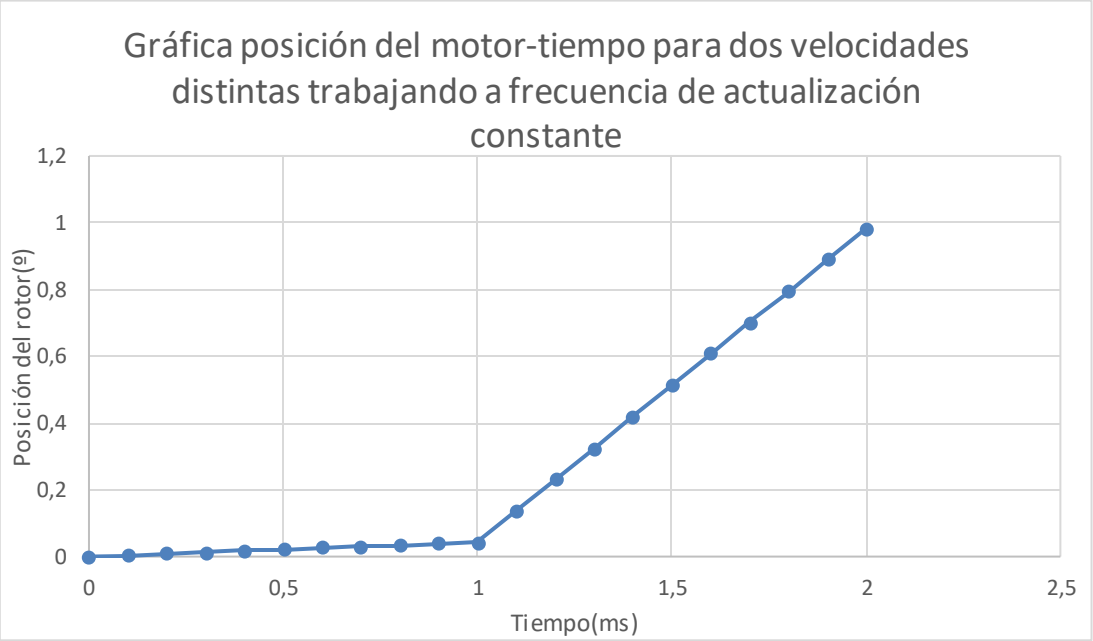

Figura 24: Gráfica de la posición del motor a lo largo del tiempo para velocidades generadas a frecuencia de actualización constante. Dos tramos: a 50 rpm de 0 a 23 ms y a 1100 de 23 a 24 ms.

En este caso se fija un tiempo de espera de 100 µs, tiempo mínimo efectivo que la función que manda el thread a dormir permite. De esta manera, se consigue que sea la relación entre el paso variable y esos 100 µs los que marquen la velocidad.

Se realiza entonces una prueba a diferentes velocidades y se refleja el error con respecto a la consigna en la siguiente tabla 2.

Consigna de velocidad (rpm)	Velocidad real (rpm) y error con respecto a la consigna		Paso (º)	Paso efectivo (º)
50	50	0.00%	0.0043	0.0043
75	75	0.00%	0.0064	0.0064
100	99	1.00%	0.0086	0.0085
150	149	0.67%	0.0129	0.0128
200	198	1.00%	0.0171	0.0170
250	249	0.40%	0.0214	0.0213
300	298	0.67%	0.0257	0.0255
400	398	0.50%	0.0343	0.0341
500	497	0.60%	0.0429	0.0426
600	598	0.33%	0.0514	0.0513
700	699	0.14%	0.0600	0.0599
800	798	0.25%	0.0686	0.0684
900	897	0.33%	0.0771	0.0769
1000	996	0.40%	0.0857	0.0854
1100	1097	0.27%	0.0943	0.0940
1200	1197	0.25%	0.1029	0.1026
1500	1498	0.13%	0.1286	0.1284
2000	1995	0.25%	0.1714	0.1710
4000	3995	0.13%	0.3429	0.3424

Tabla 2: Muestreo de los errores de la velocidad para el modo a frecuencia de actualización constante.

En este método se observa que no existe limitación en la velocidad máxima ya que el paso puede alcanzar tanto valores muy pequeños como muy grandes. Además, son alcanzables todas las velocidades dentro del rango establecido porque la variable paso no cuenta con las limitaciones que si tenía la función de mandar a dormir el thread.

También cabe destacar el tratamiento de velocidades muy pequeñas. Para una velocidad mínima de 50 rpm se obtiene un paso de 0.0043º. Se trata de un paso realmente pequeño que el encoder no es capaz de detectar. Sin embargo, este paso se irá acumulando cada 100 µs hasta que el encoder por fin sea capaz de detectar esa diferencia entre la posición actual y la directiva de posición como para generar el cambio de posición, manteniendo la relación espacio/tiempo que proporciona la velocidad constante. Es decir, se necesita una variación de posición de por lo menos 0.1º para que el motor actúe y se irá acumulando el paso hasta que llegue a ser 0.043º x 3 cuando hayan pasado 100 x 3 µs. De esta manera, la velocidad conseguida se mantendrá gracias a esa acumulación de pasos.

Elegimos este segundo método porque no conlleva todas las limitaciones que sí tiene el primero.

3.2.3 Función generadora de movimiento

Con todo lo explicado anteriormente se pasa a describir el funcionamiento del thread que genera el movimiento a través de un diagrama de bloques (figura 25). Cabe destacar que, dado que se tiene un encoder absoluto, existe la necesidad de establecer un límite superior de 360º y uno inferior de 0º a la referencia de posición. Además, tras un reinicio o encendido del VESC es necesario igualar la referencia de posición a la posición actual del motor antes de sumar el paso para evitar movimientos bruscos. También hay que decir que no es necesario diferenciar cuando la velocidad es positiva o negativa ya que el propio paso definirá el signo de la velocidad y esa suma se convertirá en una resta en caso de velocidades por debajo de 0.

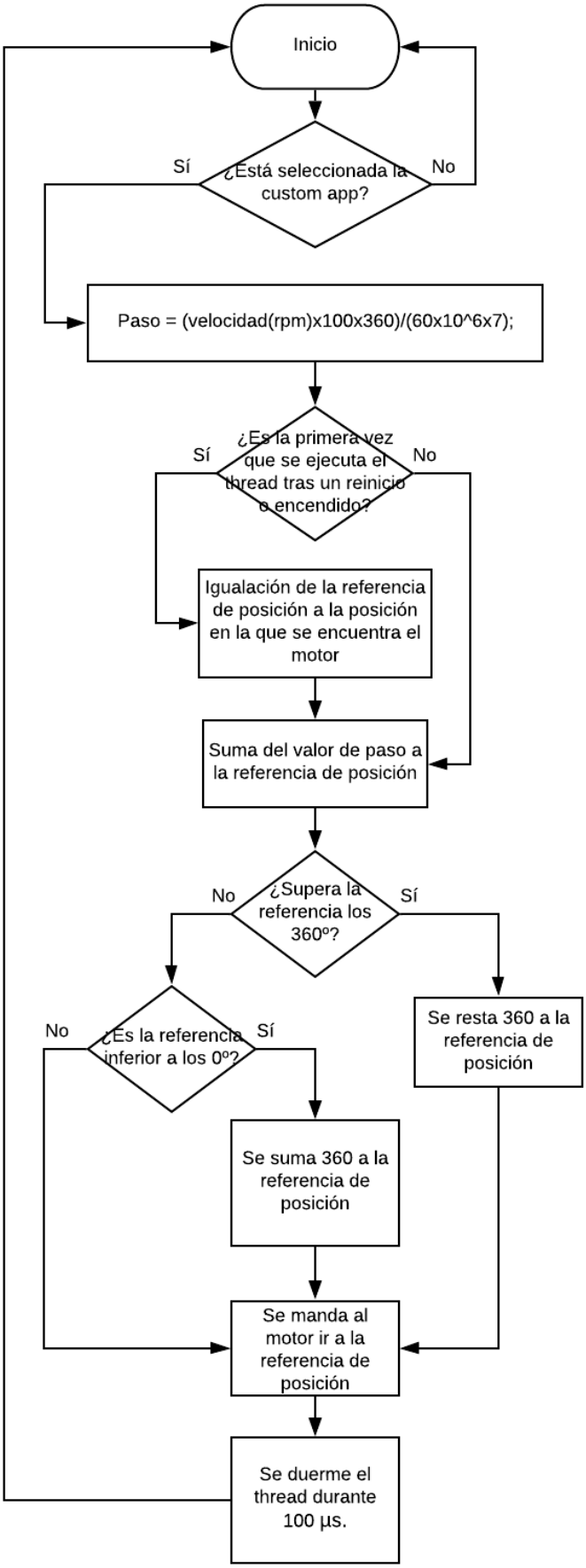

Figura 25: Diagrama de generación de movimiento.

3.2.4 Tratamiento de las consignas de velocidad

Por el puerto serie del VESC llegará un mensaje con la información de la velocidad a la que debe moverse el motor. En ese caso el valor del identificador del paquete es 8. Al llegar, es necesario comprobar si está activada la custom app o no. En caso afirmativo se tratará la velocidad mediante las referencias de posición y en caso contrario mediante el PID de velocidad.

3.2.5 Problema del bloqueo externo del motor

El método de control de movimiento implementado encuentra un problema cuando se bloquea externamente el motor. Si el motor se bloquea por un elemento externo, por ejemplo un obstáculo del camino, evitando su movimiento mientras la consigna de velocidad es diferente de cero, la referencia continuará aumentando estando en algunos instantes por delante de la posición real del motor (par en un sentido) y en otros por detrás (par en el otro sentido). De esta manera, cuando el motor se para externamente con una velocidad diferente de cero tratará de moverse en un sentido y, tras un tiempo determinado, en el otro. Estas dos situaciones quedan reflejadas en la figura 27. Este comportamiento no es nada recomendable para el sistema ya que no se quiere que cuando la *High Speed Cablecam* o la *Mantis* encuentren un obstáculo que les frene comiencen a moverse hacia adelante y hacia atrás.

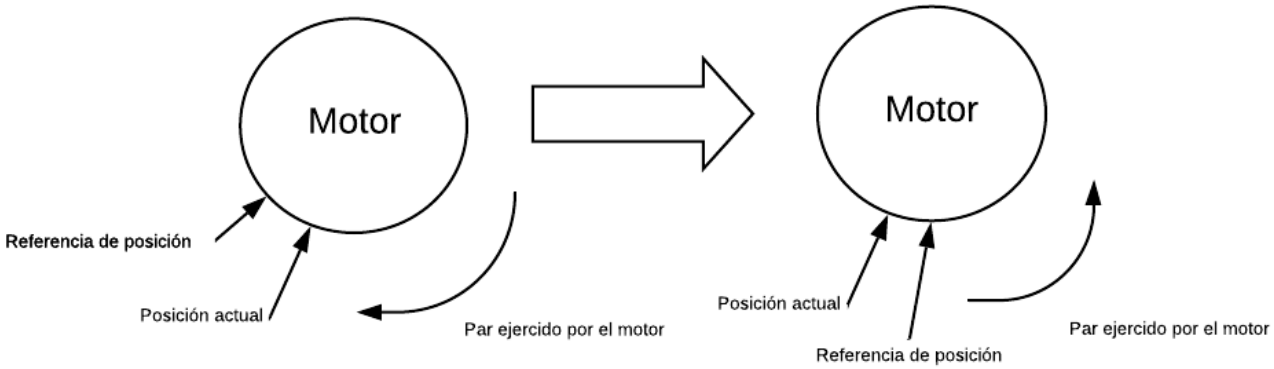

Figura 27: Imagen ilustrativa del problema del bloqueo externo del motor. Par y velocidad tienen el mismo sentido.

La solución consiste en medir la diferencia entre la posición actual del motor y la referencia de posición y evitar que la segunda siga aumentando cuando esta diferencia se acerque a la media vuelta. Por tanto, se hará una resta entre esas dos variables y se impondrá que cuando esta diferencia sea superior a los 170º se deje de sumar el paso a la referencia de posición. Sin embargo, se seguirá enviado directivas de actualización de posición para mantener el par del motor.

Aunque el planteamiento es sencillo, realizar una resta entre dos variables, hay que tener en cuenta algunos aspectos prácticos. Cuando la velocidad tenga un signo positivo el valor del ángulo de la referencia de posición será superior al de la posición actual y cuando sea negativo inferior. En ambos casos se buscará que la resta tenga un signo positivo. Es decir, en velocidades positivas la resta será referencia menos posición real y en negativas al revés. Sin embargo, al haber establecido unos límites superior e inferior en 360º y 0º respectivamente, existe la posibilidad de que la referencia de

posición rebase estos límites y la posición actual no. En este caso, la resta quedaría desvirtuada.

Un ejemplo de esta desvirtuación sería el siguiente. Se supone un movimiento en el que los pasos tienen un signo positivo de valor 0.5º y el valor de la referencia de posición siempre superará al de la posición del motor. La resta, que no debe superar los 170º, será igual a la referencia de posición menos posición del motor. Sin embargo, cuando por ejemplo el motor se encuentre en la posición de 359,7º la referencia de posición estará en 0.2º. Al realizar la resta, resulta el valor -359,5 y no 0.5º como debería ser.

La solución consiste en establecer 4 diferentes posibilidades o casos. Se referirán 1,2,3 y 4 en el orden en que se explica seguidamente. En el caso 1, el paso es positivo y la referencia de posición es de valor superior a la posición del motor. En el caso 2, el paso es positivo pero el valor de la referencia de posición es inferior al de la posición del motor. En el caso 3 el paso es negativo y la referencia de posición es inferior a la posición del motor. Por último, en el caso 4 el paso es negativo y la referencia de posición es superior a la posición del motor. En los casos 2 y 4, la referencia de posición ha saltado el límite y por tanto, nos dará un valor negativo erróneo. Para compensar esta desvirtuación o error, se sumará 360 grados a la variable de la resta. Se impondrá que cuando la resta, calculada de diferentes formas en cada uno de los casos, sea superior a 170º se deje de sumar el paso a la referencia de posición.

Para ilustrar este problema se pueden observar las siguientes tablas, la 3 para velocidades positivas y la 4 para negativas.

Tiempo (ms)	Posición real (º)	Referencia de posición(º)	Resta(º)	Caso
1	355	355,5	0,5	1
1,1	355,5	356	0,5	1
1,2	356	356,5	0,5	1
1,3	356,5	357	0,5	1
1,4	357	357,5	0,5	1
1,5	357,5	358	0,5	1
1,6	358	358,5	0,5	1
1,7	358,5	359	0,5	1
1,8	359	359,5	0,5	1
1,9	359,5	360	0,5	1
2	360	0,5	-359,5	2
2,1	0,5	1	0,5	1
2,2	1	1,5	0,5	1
2,3	1,5	2	0,5	1
2,4	2	2,5	0,5	1
2,5	2,5	3	0,5	1
2,6	3	3,5	0,5	1

Tabla 3: Comportamiento del valor de la resta al rebasar el límite de los 360º en velocidades positivas.

Tiempo (ms)	Posición real (º)	Referencia de posición(º)	Resta(º)	Caso
1	5	4,5	0,5	3
1,1	4,5	4	0,5	3
1,2	4	3,5	0,5	3
1,3	3,5	3	0,5	3
1,4	3	2,5	0,5	3
1,5	2,5	2	0,5	3
1,6	2	1,5	0,5	3
1,7	1,5	1	0,5	3
1,8	1	0,5	0,5	3
1,9	0,5	360	-359,5	4
2	360	359,5	0,5	3
2,1	359,5	359	0,5	3
2,2	359	358,5	0,5	3
2,3	358,5	358	0,5	3
2,4	358	357,5	0,5	3
2,5	357,5	357	0,5	3
2,6	357	356,5	0,5	3

Tabla 4: Comportamiento de la variable resta cuando se sobrepasa el límite de los 0º en el caso de velocidades negativas.

Todo lo mencionado anteriormente habla del caso en el que el par del motor acelera al elemento móvil. En esta situación el par contribuye a aumentar o mantener la velocidad. Sin embargo, existirán casos en los que el par motor deba frenar al vehículo. Un ejemplo claro es en el caso de descender una pendiente en el que la velocidad aumenta debido al peso de todo el elemento. En este caso el motor deberá generar un par que frene y anule este aumento de la velocidad manteniéndola constante.

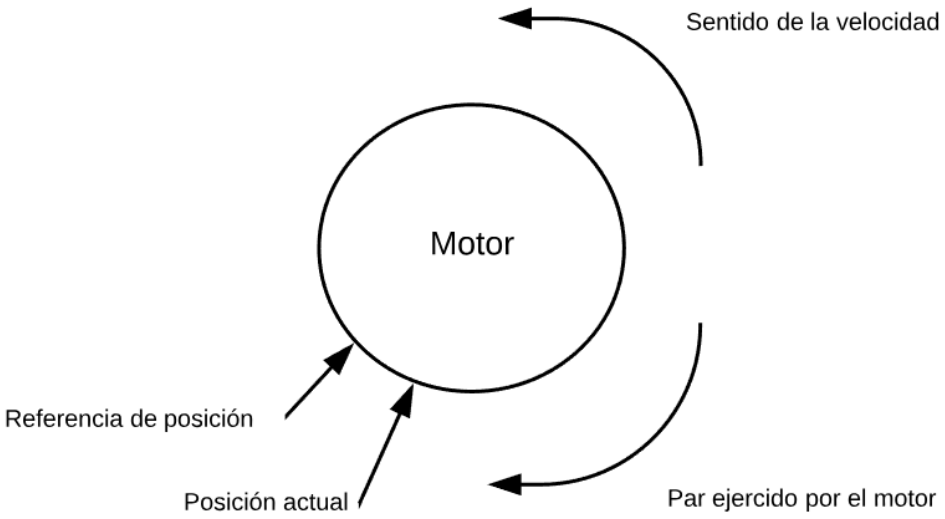

Figura 28 : Par, sentido de giro, referencia de posición y posición actual en el caso de par con sentido contrario a la velocidad.

Como se observa en la figura 28 en el caso de frenado la referencia de posición estará por detrás de la posición real en caso de pasos positivos y por delante en caso de pasos negativos. Se trata, por tanto, del caso justamente contrario al en el cual el par motor acelera el vehículo. En el caso de la frenada, la resta valdrá el ángulo complementario de lo que valdría en el caso de la aceleración ante la misma situación. Si no se hace ninguna modificación en los límites, cuando se produzca un descenso, la posición real del motor se separará del motor 170º, la referencia de posición dejará de sumar pasos y la velocidad dejará de seguirse con precisión. Por tanto, si queremos que tenga un comportamiento adecuado en descensos es necesario establecer un límite superior de valor de la resta para la cual los valores de paso comienzan a volverse a sumar de nuevo. Buscando un comportamiento simétrico, se elige el valor 190º para ese límite, complementario del límite inferior de 170º. Por todo ello se establece que cuando la resta tenga un valor superior a los 190º se continuará sumando paso a la variable de referencia.

El problema del bloqueo externo del motor está resuelto también para el caso de la frenada por la simetría del propio sistema.

El algoritmo que resuelve el problema del bloqueo externo se muestra en el diagrama de bloques de la figura 29.

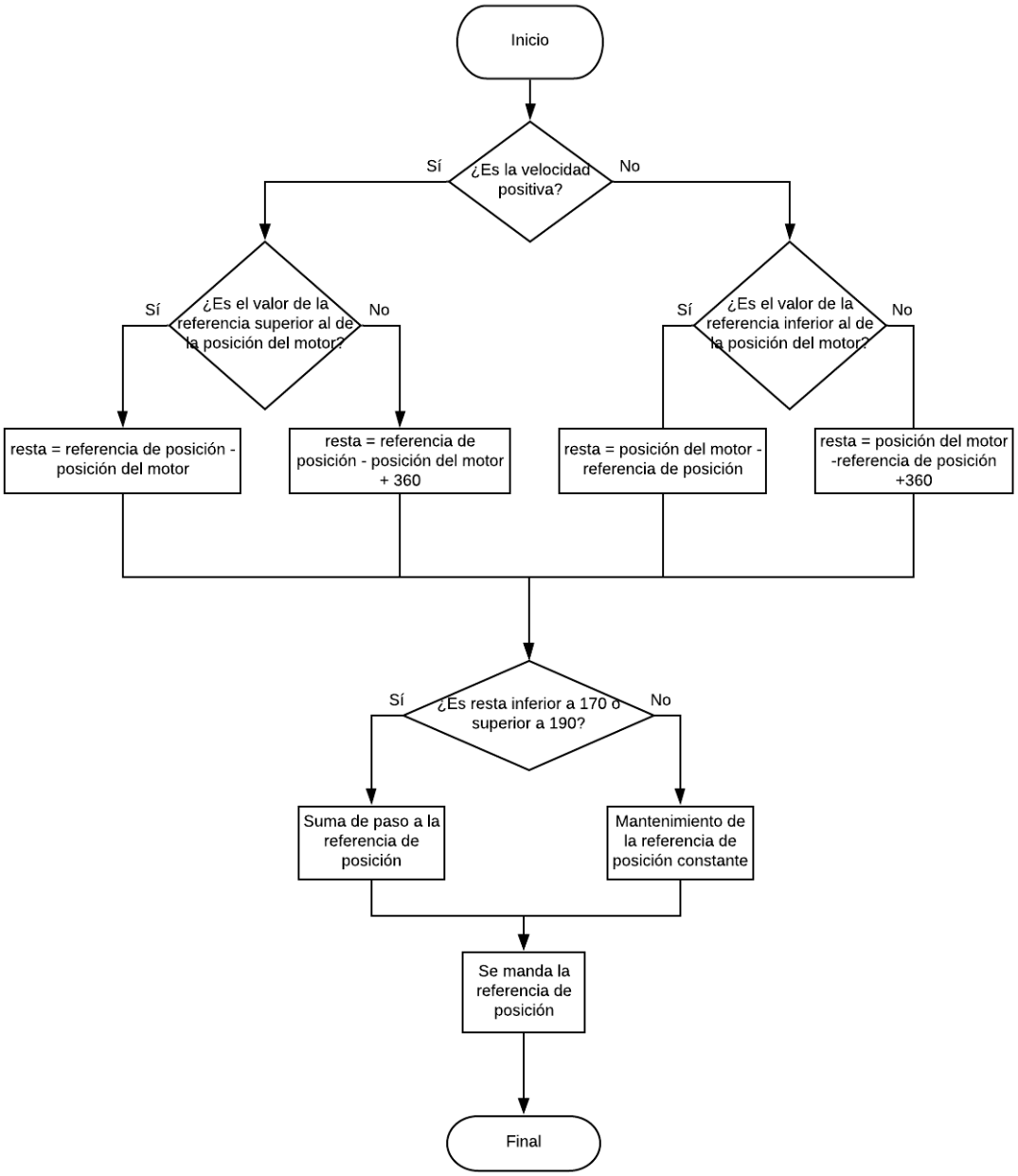

Figura 29: Diagrama de la solución para el bloqueo externo.

3.2.6 Uso de los dos modos de establecimiento de la velocidad

Tras comprobar que a velocidades bajas el funcionamiento en el modo de velocidad basada en posición es claramente mejor que el basado en el PID de velocidad, se procede a realizar el análisis en velocidades altas. Este análisis se hace para comprobar si conviene realizar un cambio de un modo a otro al llegar a una determinada velocidad. El análisis parte de las 1200 rpm porque por debajo de esta velocidad el modo basado en posición es claramente menos oscilante. Los resultados quedan reflejados en la tabla 5.

Velocidad (rpm)	Rango de oscilación - Modo basado en posición (rpm)		Rango de oscilación - Modo basado en el PID de velocidad (rpm)	
	Absoluto	Relativo	Absoluto	Relativo
1200	21	1,75%	52	4,33%
1400	22	1,57%	52	3,71%
1600	22	1,38%	54	3,38%
1800	34	1,89%	80	4,44%
2000	36	1,80%	100	5,00%
2200	38	1,73%	80	3,64%
2400	40	1,67%	66	2,75%
2600	42	1,62%	11	0,42%
2800	104	3,71%	11	0,39%
3000	105	3,50%	11	0,37%
4000	106	2,65%	52	1,30%
5000	83	1,66%	16	0,32%
6000	84	1,40%	18	0,30%

Tabla 5: Comparativa del rizado en la velocidad de los modos basado en el PID de posición y el modo basado en el PID de velocidad.

La primera conclusión que se saca a partir de los datos de la tabla es que el rango de oscilación no es directamente proporcional con la velocidad a la que corresponde

La segunda es que a por debajo de las 2500 rpm la oscilación es de menor rango en el modo basado en posición y por encima en el modo basado en el PID de velocidad. Por tanto, el límite de velocidad para un cambio de un modo a otro estará en 2500 rpm. Este límite se establece en valor absoluto. Es decir, que cambiará de posición a partir de las 2500 rpm gire el motor en un sentido o en el inverso.

El algoritmo que genera estos dos tramos se muestra en la figura 30.

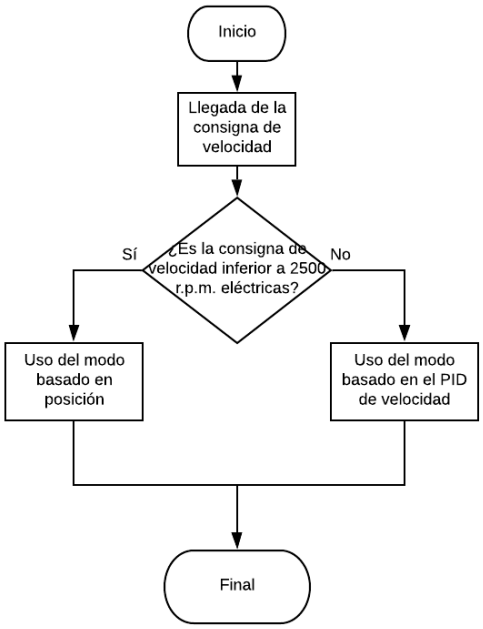

Figura 30: Diagrama de decisión del tipo de modo de establecimiento de la velocidad a usar.

Resulta destacable como el mismo sistema trabajando en modo basado en PID de posición y trabajando en modo basado en PID de velocidad generan resultados diferentes, no solo en la amplitud del rizado de la velocidad sino también la frecuencia del mismo. Este comportamiento se puede observar en las gráficas 31 y 32.

Sin entrar en mayor profundidad se plantea que este diferente comportamiento puede provenir de que se trata de dos lazos de control diferentes, explicados anteriormente. Como se observa en las gráficas 33 y 34, el lazo de control de posición cuenta con dos polos mientras que el de velocidad cuenta con uno.

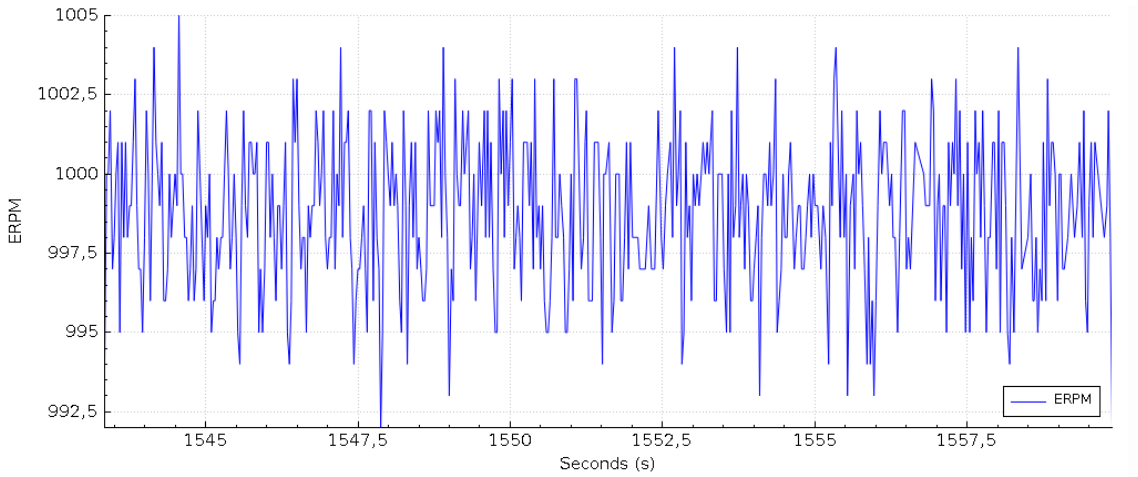

Figura 31: Gráfica de la velocidad a 1000 rpm en el modo por referencias de posición.

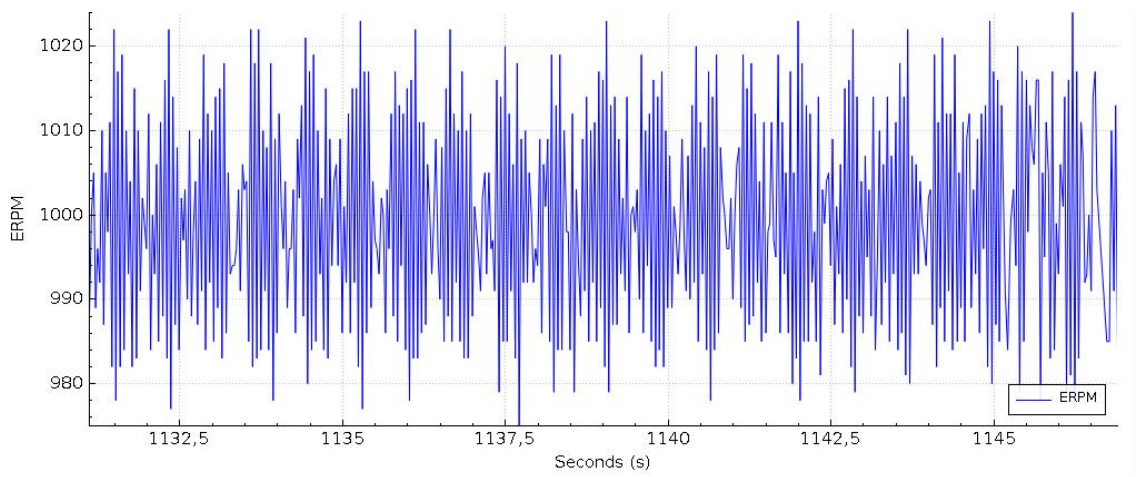

Figura 32: Gráfica de la velocidad a 1000 rpm en el modo mediante PID de velocidad.

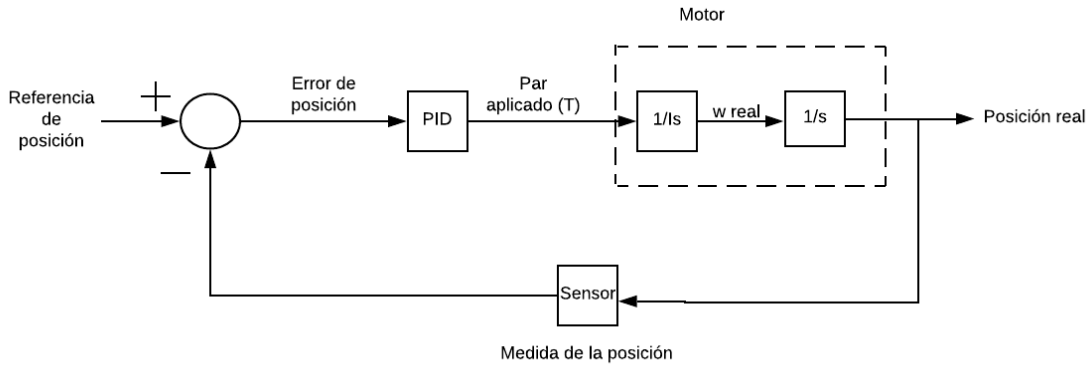

Figura 33: Lazo de control de posición.

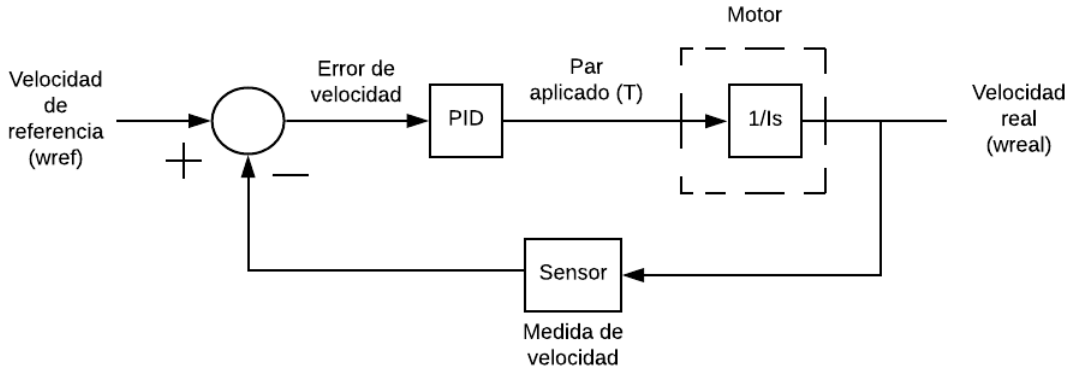

Figura 34 :Lazo de control de velocidad.

3.2.7 Funcionamiento bajo rozamiento

Dado que el sistema que va a utilizar el motor brushless va a trabajar, en el caso de la High Speed Cablecam, moviendo el peso de una cámara por encima de una cuerda conviene analizar la forma de respuesta cuando se le opone una fuerza de rozamiento. La forma de comprobarlo será simplemente someter a cierta fuerza de rozamiento. Se realizarán las pruebas con el motor parado, con motor girando a 2000 rpm (en el modo basado en posición) y a 3000 rpm (en el modo del PID de velocidad).

La primera prueba se realiza con el motor parado, con la consigna de 0 rpm, dentro del modo basado en posición. Los resultados se muestran en la figura 35.

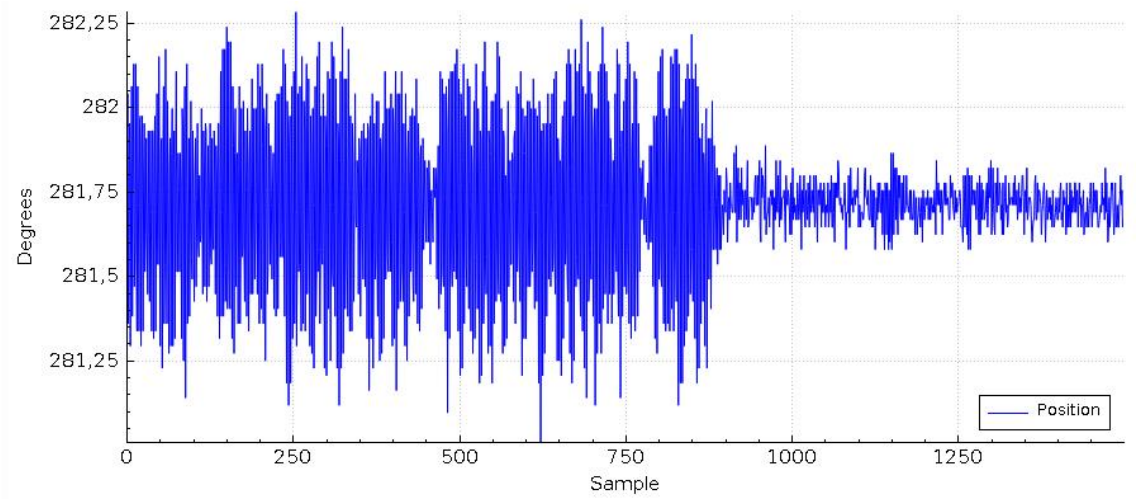

Figura 35: Gráfica de la respuesta de la posición del motor bajo rozamiento.

En el segundo 780 se somete al motor a rozamiento. Como se puede observar el rozamiento afecta de manera muy positiva ya que reduce en gran medida las vibraciones producidas en vacío. El control es capaz de mantener la posición mucho más cerca de la posición de consigna.

En la segunda prueba (figura 36) se opone rozamiento al motor cuando gira a 2000 rpm, dentro del modo basado en posición.

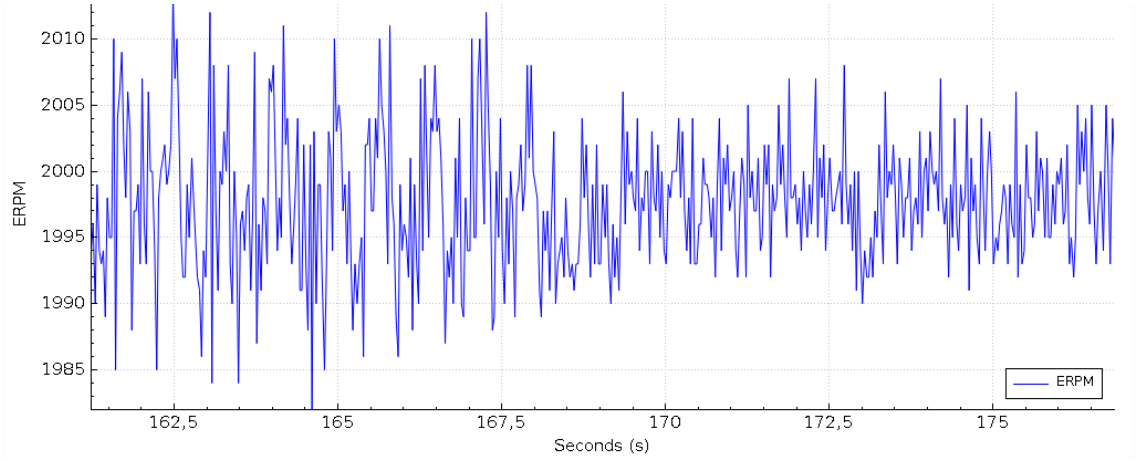

Figura 36: Gráfica de figura de la respuesta de la velocidad ante rozamiento en el modo basado en posición.

A partir del segundo 167,5 se somete al motor a rozamiento y se puede apreciar como en este caso, también se mejora el funcionamiento. El rango de velocidades en las que oscila (rizado) se vuelve más pequeño y, por tanto, se consigue un movimiento más fluido que en vacío.

La disminución del rizado, tanto en la posición cuando el motor se encuentra parado como en la velocidad cuando se encuentra en movimiento, es esperable y se puede explicar a partir del diagrama de bloques del lazo control PID de posición de la figura 37. En ese diagrama se puede observar como la acción del rozamiento que es proporcional a la velocidad está generando una atenuación del par. De esta manera, el par aplicado al motor en parado cuando está fijado en una posición constante será menor. Al ser este par menor, disminuirá la aceleración que está produciendo que la posición real esté continuamente sobrepasando la posición de consigna en un sentido y en otro. De esta manera quedará atenuada la vibración. Por otro lado, la disminución del rizado de la velocidad cuando se encuentra en movimiento tiene la misma explicación. La atenuación del par que genera el rozamiento es directamente proporcional a la velocidad. Por ello, cuando la velocidad aumenta también lo hace el par de rozamiento generando una disminución de la velocidad. En esta situación en el que la velocidad disminuye, la atenuación del par generada por la velocidad también lo hace. Así, el rozamiento provoca una estabilización de la velocidad en torno a la consigna.

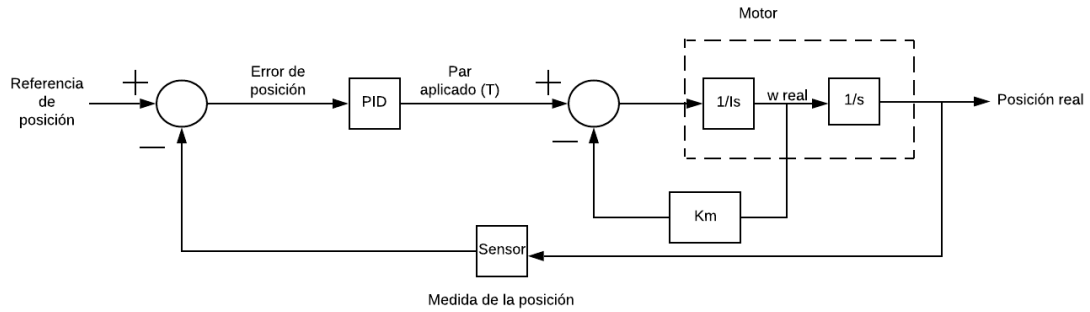

Figura 37: Diagrama del lazo de control PID con rozamiento.

En la tercera prueba (figura 38) se opone resistencia al motor cuando gira a 3000 rpm, fuera del modo basado en posición y controlado por el PID de velocidad.

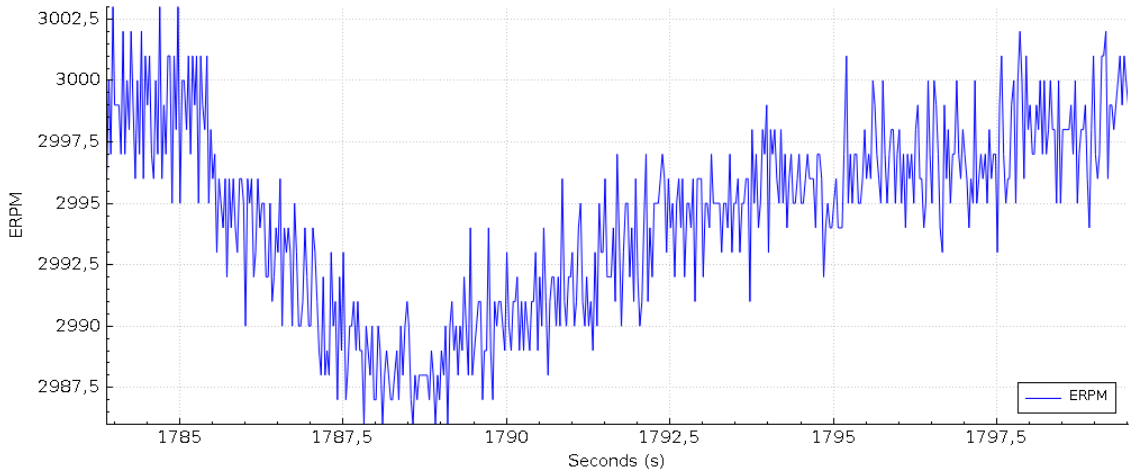

Figura 38: Gráfica de figura de la respuesta de la velocidad ante carga en modo basado en PID de velocidad.

Se introduce la fuerza de rozamiento en el segundo 1786. En este caso se observa como el rozamiento frena la velocidad en el primer momento. Después, a partir del segundo 1786 el PID aumenta el par generado por el motor contrarrestando la resistencia del rozamiento y recuperando la velocidad inicial. En cualquier caso, la desviación es en torno al 1% lo cual resulta despreciable. Sin embargo, sirve para destacar la diferencia en el comportamiento dinámico de este modo con respecto al basado en posición.

La conclusión que se puede sacar sobre la respuesta general del sistema desarrollado ante rozamiento es que en ningún caso empeora el funcionamiento. Es más, para el modo basado en posición mejora sensiblemente la fluidez de los movimientos y reduce en gran medida las vibraciones. Para el caso del PID de velocidad si que es cierto que en un primer momento la velocidad se desvía de la consigna, pero lo hace en un valor prácticamente despreciable. Además, el efecto del PID de velocidad, más en concreto la acción integral, corrige este error completamente.

4. Configuración del PID de posición

Ya se ha introducido en un apartado anterior en que consiste un PID. En este apartado se realizará el ajuste de los valores del PID de posición que se han presentado antes para conseguir un uso optimizado del modo basado en posición.

En la figura 39 se puede observar el esquema típico de un PID. La planta corresponde al motor brushless, la señal u(t) a las corrientes generadas por el VESC, y(t) la posición del motor medida desde el encoder y r(t) la referencia de posición calculada y actualizada cada 100 µs por el VESC.

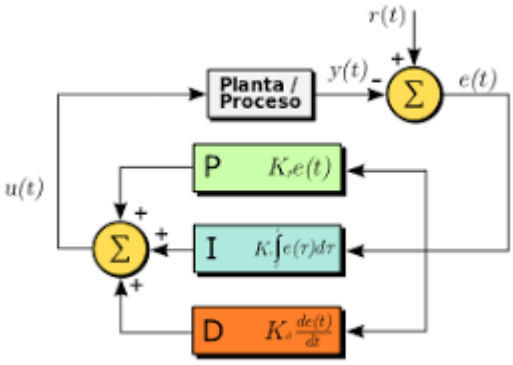

Figura 39: Esquema general de un control PID.

El proporcional deberá ser lo más alto posible ya que esto mejorará la rapidez del régimen transitorio. Además, el derivativo también tendrá que ser alto para neutralizar las potenciales inestabilidades debidas al alto valor del proporcional.

La finalidad fundamental del ajuste de este PID de posición es que el movimiento generado a partir de él sea lo más preciso y fluido posible. Por ello, el análisis se debe realizar utilizando el programa desarrollado de movimiento basado en referencias de posición y no solo introduciendo referencias de posición y viendo como actúa la salida.

4.1 Ajuste del proporcional, k_p

Se realizará un ajuste manual,es decir, se pondrán a cero k_p , k_d y k_i y se introducirán entradas escalón mientras se aumenta el proporcional hasta que se llegue a un valor en el que se haga oscilar la salida. El valor elegido para el k_p será la mitad de ese valor que provoca la respuesta oscilante. Este método utilizado para ajustar el coeficiente k_p del PID se conoce como método de Ziegler-Nichols[3].

[3] Michael A. Johnson, Mohammad Moradi, “Pid Control New Identification And Design Methods”, London, 2015.

Hay que tener en cuenta que este sistema no actúa de la misma manera ante entradas pequeñas que grandes. Ante un mismo valor de proporcional, cambios de referencia grandes generan oscilaciones cuando valores más pequeños no lo hacen. Por ello, conviene establecer el valor máximo de cambio de referencia que se usará, es decir, el paso máximo e introducirlo como cambio de referencia.

La velocidad máxima a la que se utiliza el modo basado en posición es 2500 rpm. Para calcular el paso máximo hay que hacerlo conforme a esa velocidad.

$$Paso\ máximo = \frac{Velocidad\ de\ referencia\ (rpm) \times Tiempo\ espera\ (\mu s) \times 360}{60\ \times\ 10^6\ \times 7\ (Número\ de\ pares\ de\ polos\ del\ motor)}$$
$$= \frac{2500 \times 100\ \times 360}{60\ \times\ 10^6\ \times 7\ (Número\ de\ pares\ de\ polos\ del\ motor)} = 0.21º$$

Ecuación 4

Introduciendo cambios en la referencia de 0.21º y aumentando el k_p hasta que oscile se obtiene que esta oscilación se produce cuando vale 0.3. De esta manera, el valor elegido para el k_p será 0.15.

4.2 Ajuste del derivativo, k_d

Una vez fijado el proporcional se procede a ajustar el derivativo. En este caso se actuará de una manera diferente y el análisis no se realizará en estático, introduciendo variaciones en la referencia, sino que se pondrá en marcha el modo basado en posición y variando el k_d se buscará el funcionamiento con menos rizado en la velocidad. La menor oscilación posible respecto a la velocidad de consigna será el criterio para elegir el valor de k_d. Los resultados se reflejan en la tabla 6.

	Velocidad (rpm)	Velocidad (rpm)	Velocidad (rpm)	Velocidad (rpm)	Vibración del motor de alta frecuencia. Apreciación audible
	50	100	1000	2000	
Valor del kd	Oscilación dominante a 20Hz (rpm)	Oscilación dominante a 20Hz (rpm)	Oscilación dominante a 20Hz (rpm)	Oscilación dominante a 20Hz (rpm)	
0,0007	15	20	17	15	No apreciable
0,0010	10	16	16	12	No apreciable
0,0012	10	12	13	12	No apreciable
0,0014	9	10	12	12	No apreciable
0,0016	9	10	11	10	No apreciable
0,0018	8	10	9	10	No apreciable
0,0020	8	9	9	10	No apreciable
0,0025	8	9	9	10	Comienza a aparecer
0,0030	8	9	9	10	Notablemente apreciable

Tabla 6: Resultados del rizado de la velocidad para diferentes valores del kd y diferentes velocidades.

Como se puede observar en la tabla para valores de k_d superiores a 0.002 el rango de oscilación no disminuye, pero sí que se nota un aumento de las vibraciones del motor. Por ello, nos quedamos con el valor más pequeño que produce menos vibraciones, el 0.002. Por debajo del rango estudiado en la tabla (valores de k_d inferiores a 0.007) el sistema es inestable y por encima de 0.003 las vibraciones producidas por el PD son excesivas.

4.3 Ajuste del integrador, k_i

En un principio puede parecer que el ajuste del integrador sea de vital importancia debido a la necesidad de que los pasos alcancen las posiciones de una manera muy precisa para ajustar lo máximo posible la velocidad a la deseada. Sin embargo, un error en régimen permanente constante no afecta a la precisión de la velocidad generada. En otras palabras no nos importa que las posiciones no se alcancen de forma precisa si ese error se mantiene constante en todo los puntos. De esta manera, se generará una recta de puntos de posición del motor que no coincidirá con la de referencia pero cuyas pendientes sean iguales. Esto es, generará la velocidad deseada.

Recopilando datos sobre las referencias de posición y las posiciones correspondientes alcanzadas con los k_p y k_d decididos anteriormente y sin integrador se genera la siguiente gráfica representada en la figura 40. En cada instante se dibuja la referencia de posición y el punto alcanzado a través de ella para una velocidad de 500 rpm.

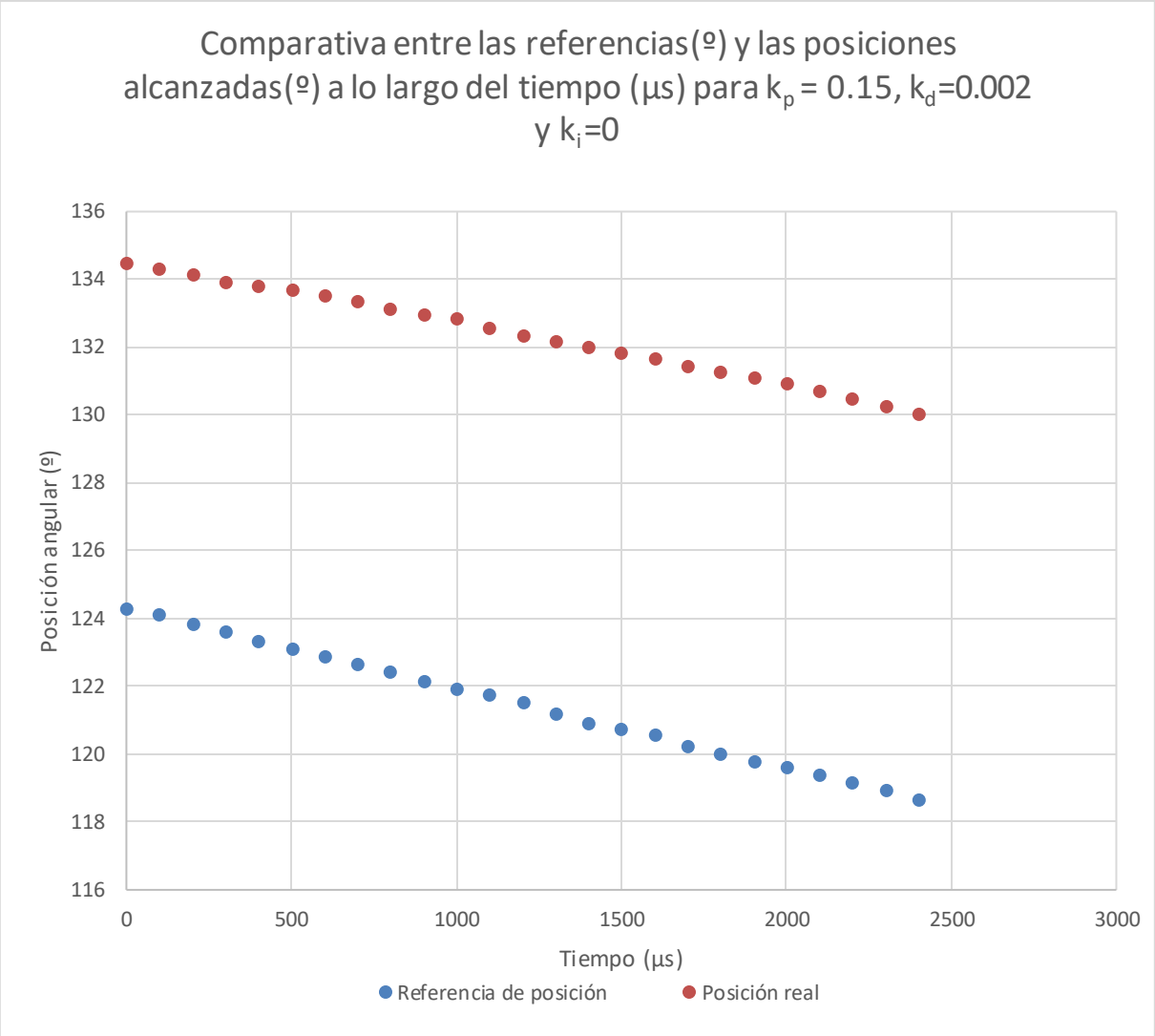

Figura 40: Comparativa entre referencias y posiciones alcanzadas para una acción integral nula.

Como se puede observar las posiciones reales del motor no se corresponden con las referencias que deberían seguir. En ese caso siempre existe un error constante de 11º. Sin embargo, ambas rectas tienen la misma pendiente, lo que indica que el seguimiento de la velocidad marcada por las referencias de posición es preciso. Por todo ello se puede concluir que el integrador no tiene influencia en la consecución de velocidades precisas y fluidas.

En algunas aplicaciones de la *High Speed Cablecam* y de la *Mantis* puede que sea necesario un control de la posición preciso. Dado que en la práctica ese error de 11º supone un error de 6 mm en el desplazamiento lineal sobre la cuerda, resulta claramente despreciable.

Además, parece probable que el régimen permanente tras cada directiva de posición no se alcance o se alcance durante un lapso de tiempo muy corto, que no permite a la acción integrar tener influencia en la salida. Esto es, el tiempo (τ) de estabilización del sistema será mayor, igual o ligeramente menor a los 100 µs que hay entre directivas de posición. No se ha profundizado más en el estudio de este aspecto ya que las

velocidades se consiguen igualmente y el error de posición trasladado al desplazamiento del vehículo por la cuerda es despreciable.

Por todo lo mencionado se fija la k_i en 0.

5. Validación

Para validar el sistema desarrollado se procede a introducir dos perfiles de velocidad diferentes como consigna y analizar la respuesta del motor brushless ante ellos. De esta manera, podremos comprobar si el sistema cumple el objetivo de generar velocidades fluidas y continuas en el rango establecido. Además, también servirá para ver como se comporta ante los cambios de velocidad, más en concreto en el paso del modo basado en posición al modo basado en el PID de velocidad. Las consignas de velocidad llegarán al VESC a través del puerto serie cada 10 ms. Por tanto, las pendientes de los perfiles de velocidad estarán formadas por escalones de 10 ms.

Primero se someterá al motor brushless al perfil de bajas velocidades reflejado en la figura 41 de forma cíclica y en el que no se saldrá del modo basado en posición. Se buscará validar el comportamiento y la precisión de las bajas velocidades en ambas direcciones.

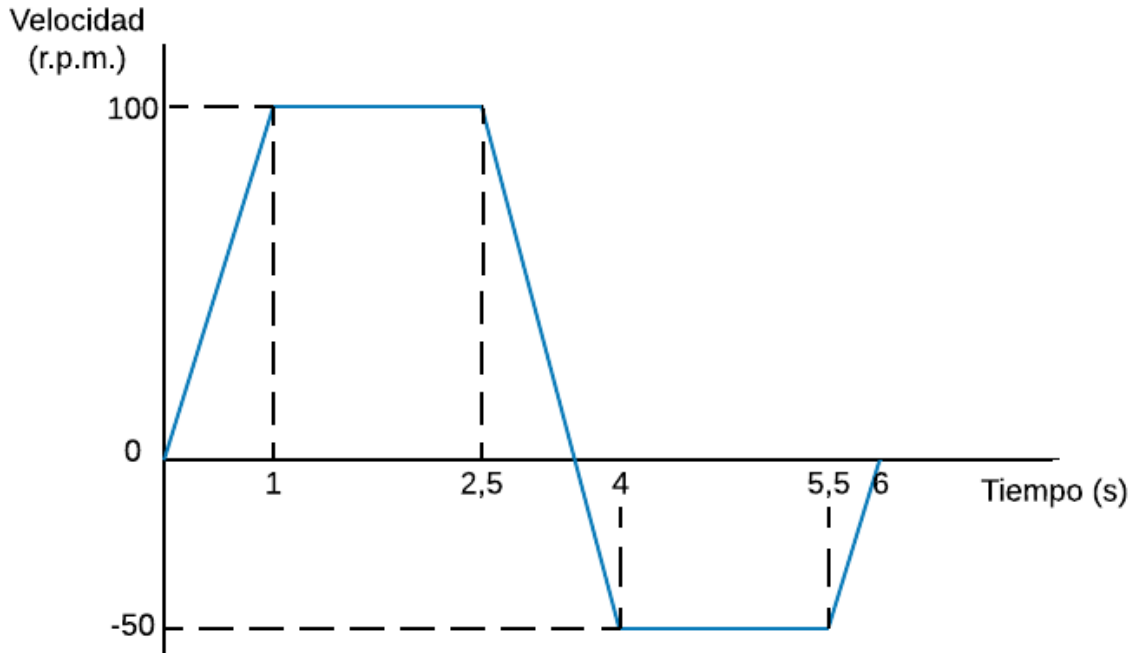

Figura 41: Perfil de velocidades a baja velocidad.

Los resultados del comportamiento del motor ante el perfil de la figura 42 se muestran en la figura 42.

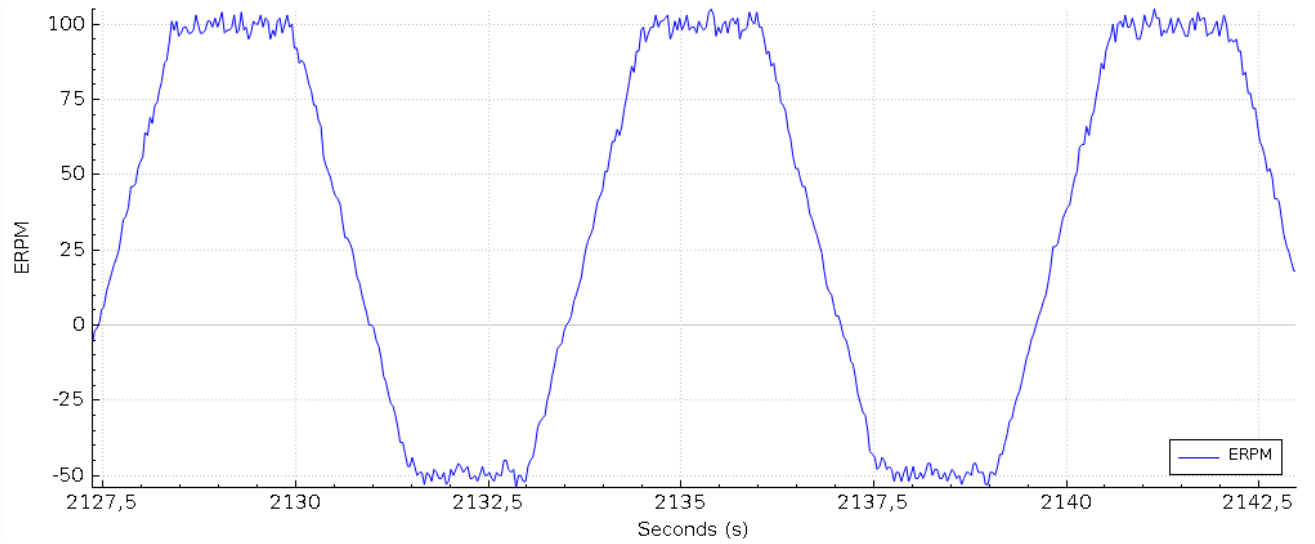

Figura 42: Gráfica de respuesta del motor ante el perfil de velocidades de la ilustración 40.

Se ve claramente como las velocidades son seguidas con total precisión tanto en un sentido en un sentido como en el otro ya están perfectamente centradas en los valores 100 y -50 rpm. Se puede apreciar el rizado a velocidad constante, que tiene en torno a 10 rpm de amplitud. El rizado en los tramos de aceleración proviene en parte de ese rizado a velocidad constante pero también de que los tramos de aceleración de los perfiles referencia, como se ha dicho antes, no son perfectamente rectos. A la vista de los resultados se puede decir que el sistema genera un comportamiento satisfactorio velocidades bajas.

El segundo perfil de velocidades al que se someterá el motor será el mostrado en la figura 43. En este caso se buscará validar el comportamiento en un rango de velocidad más amplio llegando a las 4000 rpm y saliendo del modo basado en posición. De esta manera se podrá estudiar si el cambio de un modo a otro tiene algún tipo de influencia.

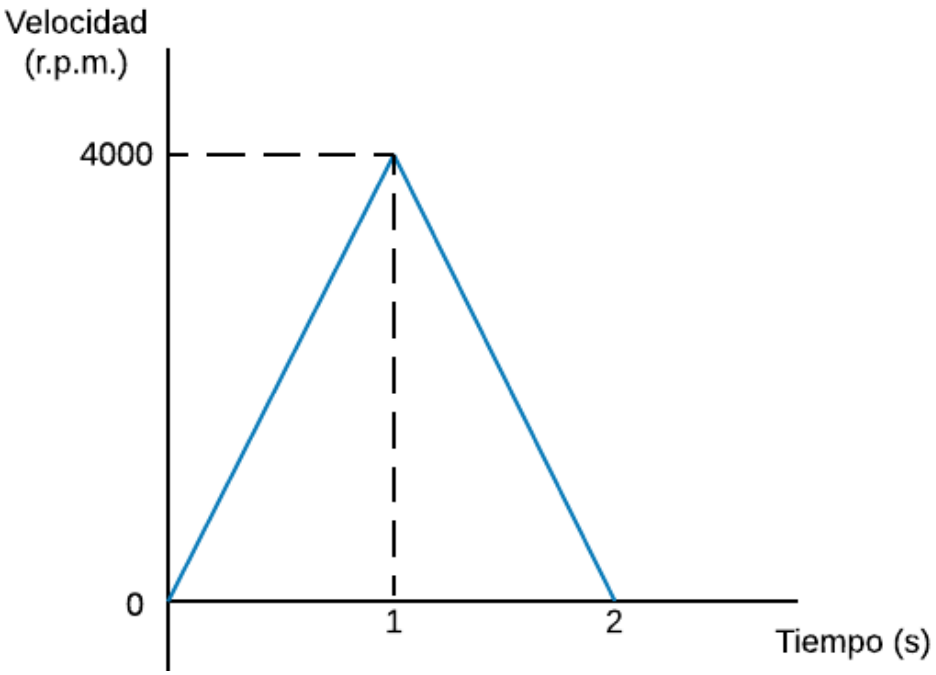

Figura 43: Perfil de velocidades para un rango hasta 4000 rpm.

La respuesta del sistema ante este perfil de velocidades referencia se muestra en la figura 44. Se puede observar cómo trabajando a velocidades altas las velocidades se alcanzan con igual precisión ya se comprueba que el pico de velocidad llega justo a las 4000 rpm. Se destaca que, dado que la gráfica se autoajusta a una escala mayor, las oscilaciones son prácticamente inapreciables. El efecto más resaltable de esta gráfica es el punto en el que pasa de el modo pasado en el control de posición al modo basado en el control de la velocidad. Se muestra más marcado en el cambio del modo basado en posición al modo basado en velocidad que a la inversa. El origen de estas pequeñas oscilaciones sería el hecho de que al pasar de un lazo de PID al otro el nuevo no tiene memoria respecto del estado anterior. Se trata de un cambio un tanto descontrolado en el que no se tienen referencias del pasado, lo que afecta a la acción integral ni errores calculados para el futuro, que afecta a la acción derivativa. En cualquier caso, resulta un comportamiento despreciable.

La validación y demostración de la consecución del par necesitado a velocidades bajas se realiza a través de la prueba en el vídeo adjuntado a esta memoria. En él se puede ver como el vehículo genera suficiente par como para subir una pendiente a velocidad baja.

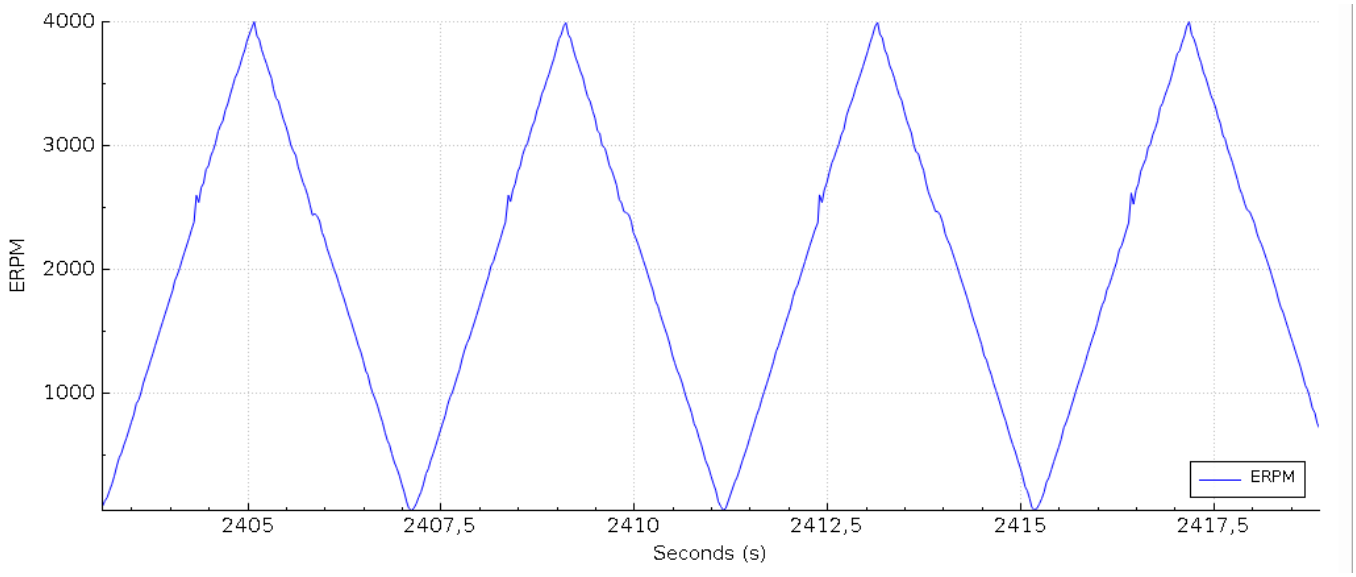

Figura 44: Gráfica de respuesta del motor ante el perfil de velocidades de la figura 42.

6. Conclusiones y líneas futuras

6.1 Conclusiones

La principal conclusión de este proyecto es que se ha podido desarrollar un sistema de velocidades basado en directivas de posición, controladas a través de un PID de control de posición, que funciona en bajas velocidades de una manera fluida y precisa y generando un par adecuado para la aplicación para la que se va a usar. Para implementar el sistema se ha visto que es mucho más conveniente realizar la generación de velocidad a una frecuencia de actualización de posiciones constante que a un paso constante.

A partir de esa velocidad se ha desarrollado un cambio de posición al modo basado en PID de velocidad, dadas las mejores prestaciones del mismo en ese rango. Además, se ha resuelto el problema del bloqueo externo del motor. Para desarrollar el sistema ha sido de vital importancia un minucioso análisis del firmware del VESC, centrándolo en la aplicación que implementa la comunicación serie y funciones de alto y bajo nivel.

También se concluye que es de vital importancia el correcto ajuste del PID en busca de esa fluidez y precisión del movimiento.

6.2. Líneas futuras

En lo referente a líneas futuras y tratando de conseguir una velocidad aún más fluida se podría buscar alguna forma de reducir esos 100 µs de tiempo de actualización interviniendo en las prioridades de los threads y estudiando las consecuencias. También un encoder con una mayor sensibilidad podría mejorar el sistema en este aspecto.

Por otro lado, también podría ser conveniente un estudio de la dinámica del lazo de control para entender mejor el funcionamiento de lo que se ha hecho hasta ahora. Para ello debería modelarse el motor y poder conseguir su inercia.

Por último, también podría desarrollarse por programa un sistema de lazo de control que pueda suponer algún tipo de mejora con respecto al implementado, quizá en un nivel más bajo.

7. Referencias y bibliografía

7.1. Referencias

[1] ams AG, *AS5047P "High-Resolution Position Sensor Datasheet",* 2016.

[2] Giovanni Di Sirio, *"Chibios Full Reference Manual",* Salerno, Italia, 2016.

Obtenido de:

http://chibios.org/dokuwiki/doku.php?id=chibios:documentation:start

[3] Michael A. Johnson, Mohammad Moradi, "*Pid Control New Identification And Design Methods",* London, 2015.

7.2 Bibliografía

Stephen J. Champman, "*Máquinas eléctricas",* Nueva York, USA, 1987.

J.R.Hendersonshot Jr.,T.J.E.Miller, "*Design of Brushless-Magnet Machines",* Florida, USA, 2010.