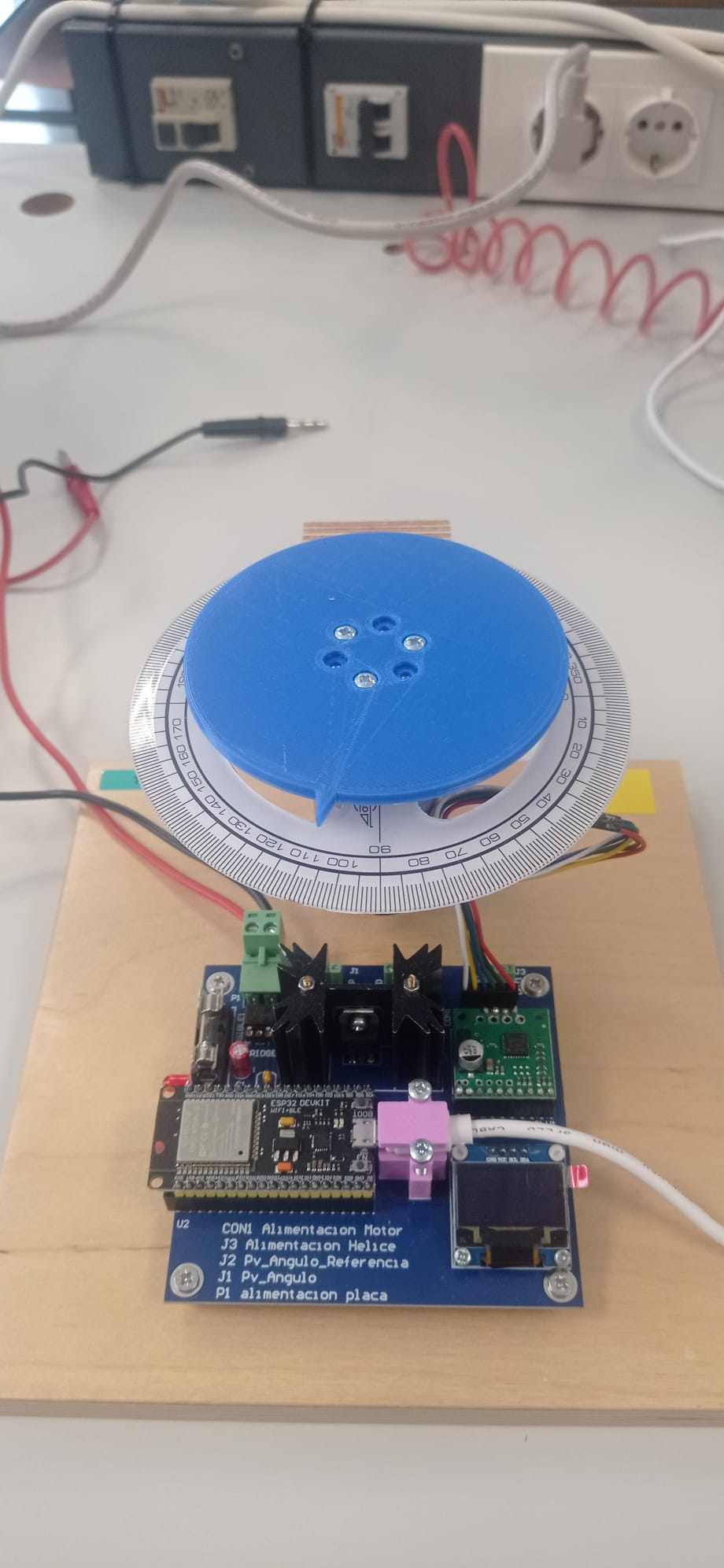
Control de velocidad y posición de un motor



**Daniel Burruchaga Sola**

3° GTI

<https://github.com/danielei9/Control_PID>

**ÍNDICE**

1. Introducción.
2. Tareas a Realizar en el informe
3. Implementación del algoritmo básico de control PID para el control de velocidad del motor.
4. Modificación del sistema para que realice el control de posición angular del motor.
5. Implementación del algoritmo de anti “windup” condicional.
6. Implementación del algoritmo de anti “windup” con recálculo.
7. Implementación del algoritmo de anti zona muerta.

**Introducción**

En esta práctica vamos a implementar diferentes algoritmos de control en el micro-controlador ESP32 para realizar el control de la velocidad de rotación y ángulo de giro de un motor de DC.

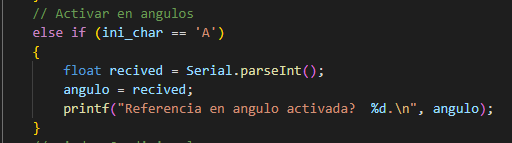
Tareas a realizar

1. Implementación del algoritmo básico de control PID para el control de velocidad del motor. 
2. Modificación del sistema para que realice el control de posición angular del motor. 
3. Implementación del algoritmo de anti “windup” condicional. 
4. Implementación del algoritmo de anti “windup” con recálculo. 
5. Implementación del algoritmo de anti zona muerta.
6. Implementación del filtro derivativo.

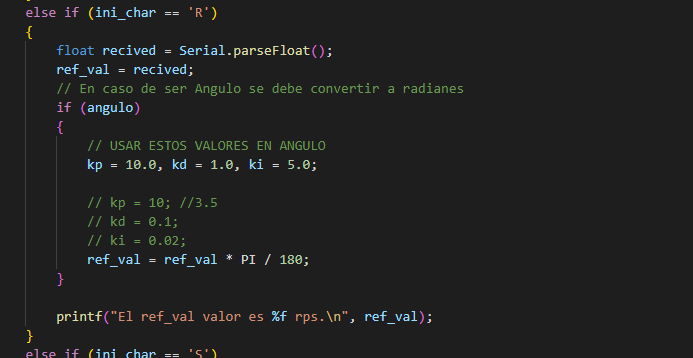
1 - Implementación del algoritmo básico de control PID para el control de velocidad del motor.

**Obtención de parámetros:**

El primer paso, necesario es decidir si vamos a usar como referencia ángulos para el control de posición o si usaremos el control de velocidad. Podemos seleccionar el uso escribiendo por la terminal A1 -> para activar ángulos, A0 -> para seleccionar velocidad , de forma dinámica.



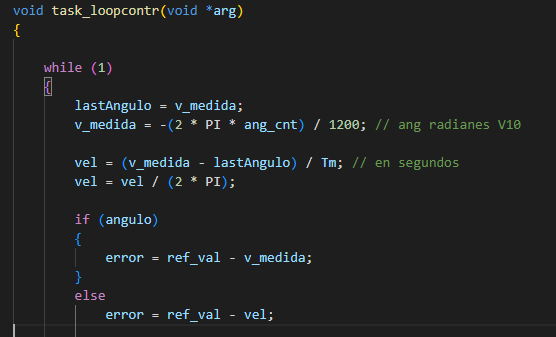
Una vez realizado esto, debemos de introducir la Referencia deseada, para ello introducimos a través de la terminal serial. RXX – Siendo XX la referencia deseada en velocidad o ángulos dependiendo de la selección del paso anterior.



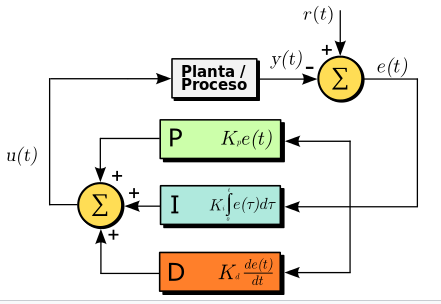
Ahora ya podemos iniciar / parar introduciendo en terminal S1/S0, respectivamente.

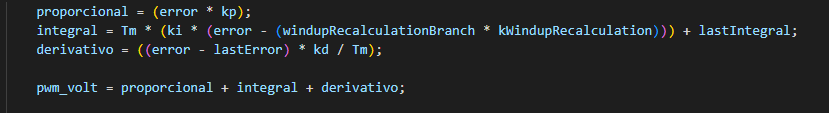
Para nuestro controlador PID es necesario obtener la referencia, y el error.

Procedemos a calcular el error:



Ahora procedemos a realizar el cálculo del algoritmo PID:



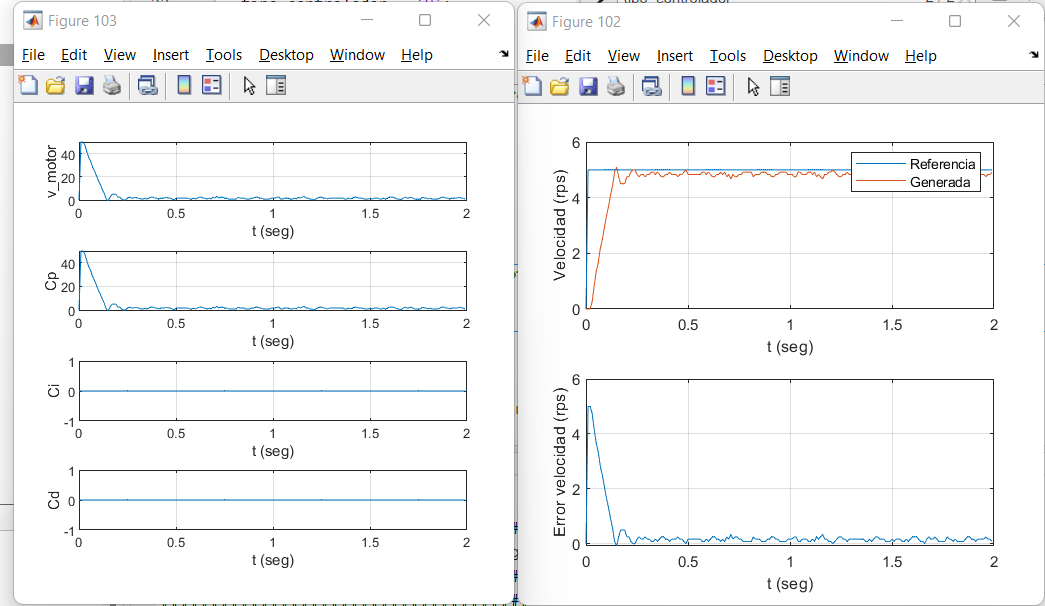
****

El cual es un sumatorio de la parte proporcional, integral y derivativa. Permitiendo así ser un mecanismo de control que a través de un lazo de retroalimentación permite regular la velocidad o posición en este caso.

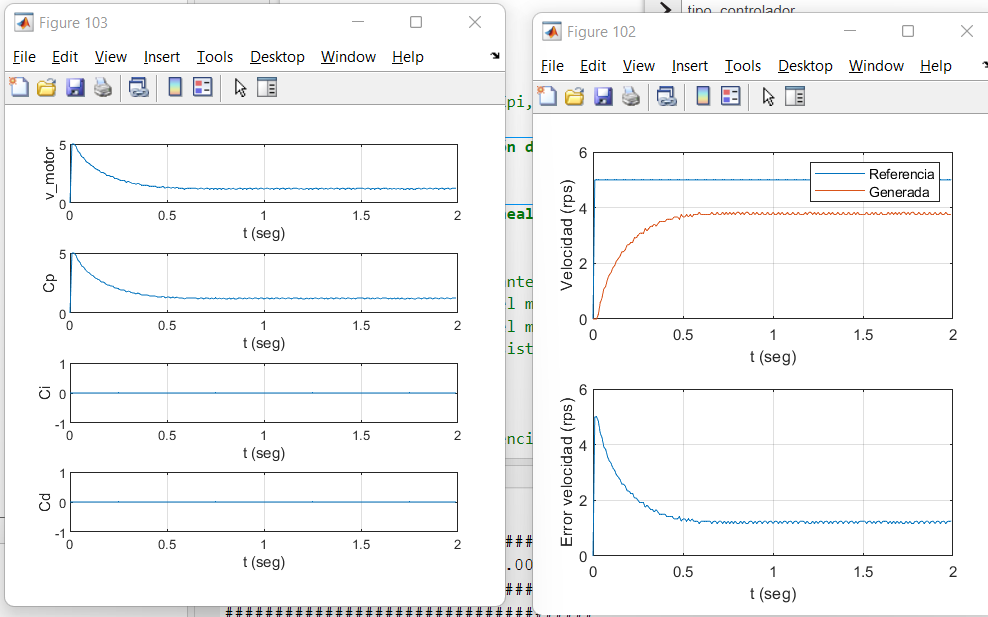
La parte proporcional consiste en el producto entre la señal de error y la constante proporcional para lograr que el error en estado estacionario se aproxime a cero, pero al no disponer de un integrador comprobamos que no es posible.

Implementando control P en matlab :

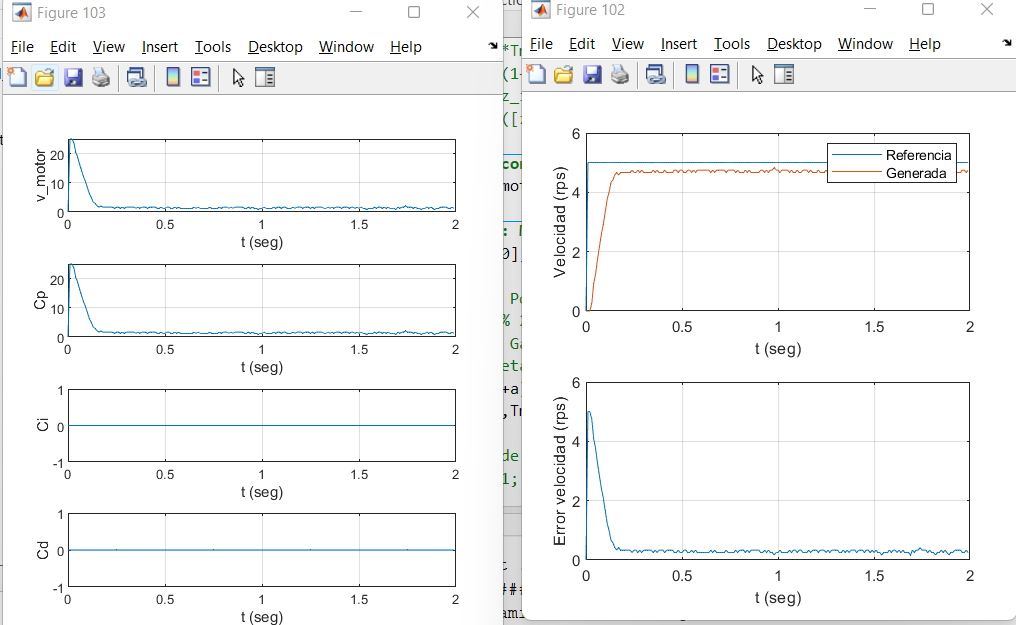
**K = 10**



**K = 1**



**k= 5** Nos quedamos con esta debido a que la señal no tiene sobreamortiguamiento. Y está ajustada a la referencia salvo el error en estado estacionario que desaparecerá con el integrador.

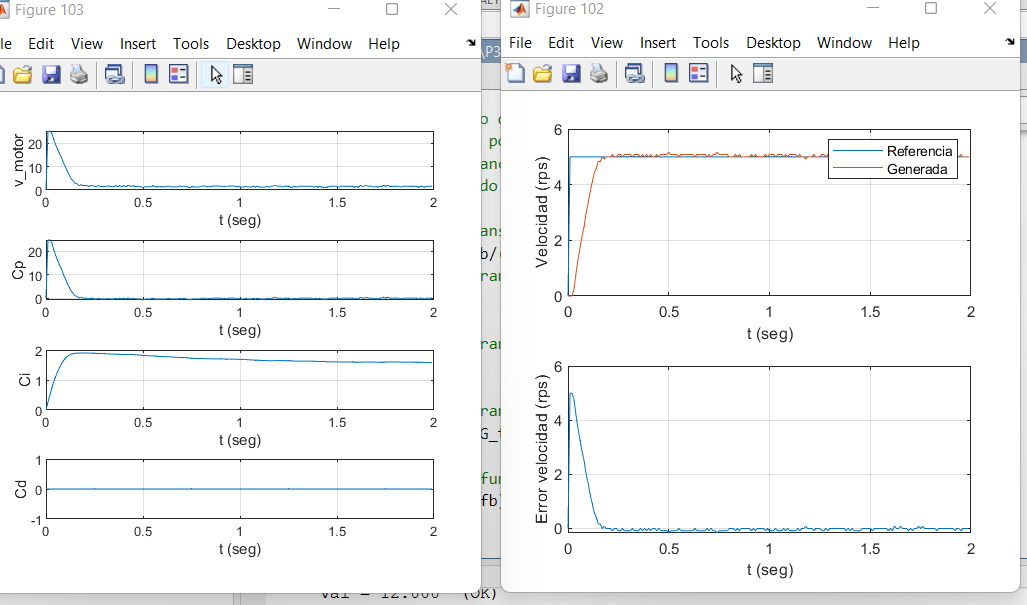


Una vez situado nuestro control proporcional, el siguiente paso será añadir un controlador integral al proporcional, obteniendo un controlador PI:

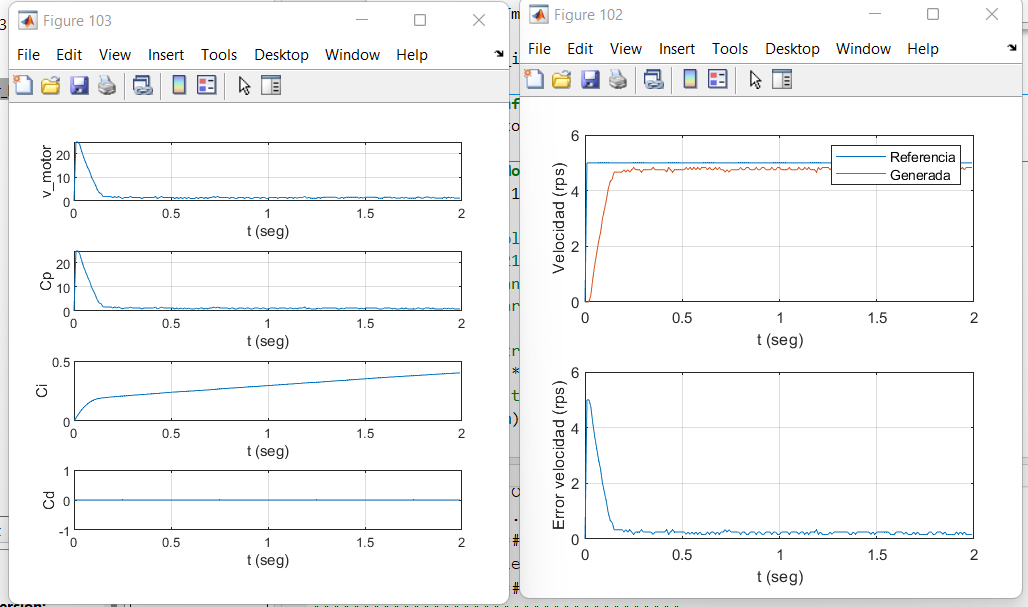
El modo de control Integral tiene como propósito disminuir y eliminar el error en estado estacionario, provocado por perturbaciones exteriores y los cuales no pueden ser corregidos por el control proporcional.

PI K= 5

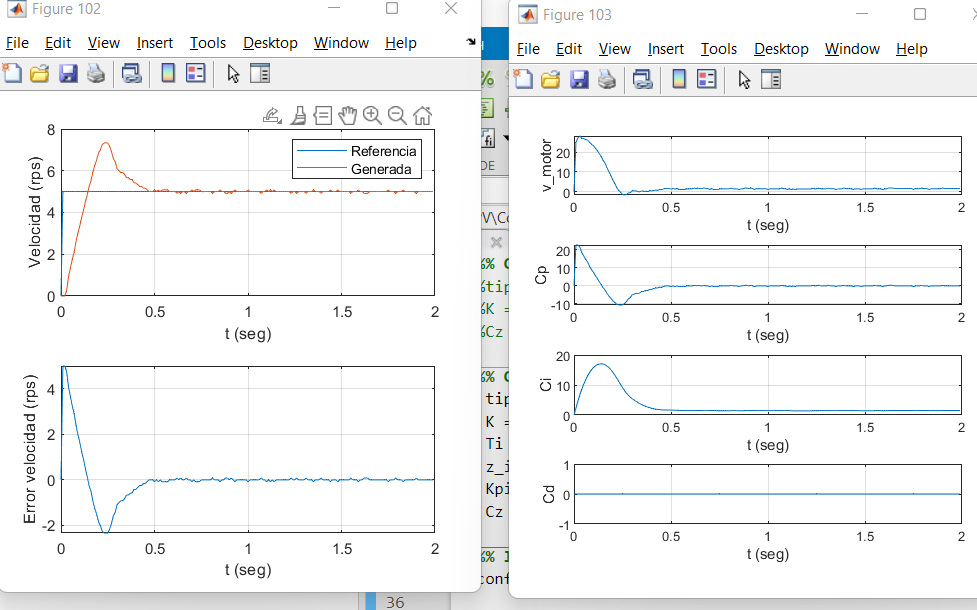
TI = 100 \* TM



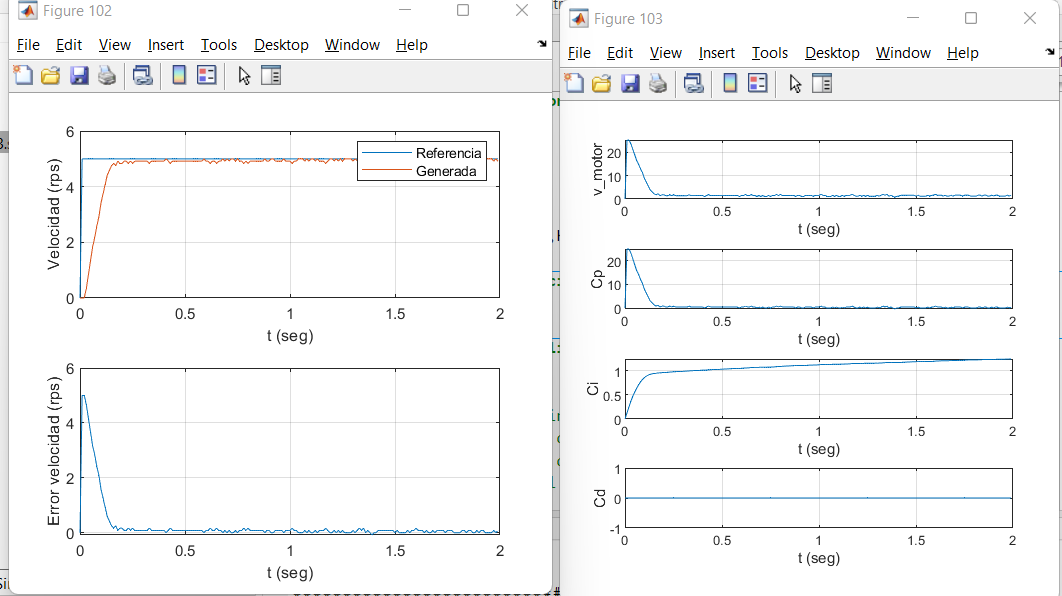
PI K = 5 TI = 1000\*tm



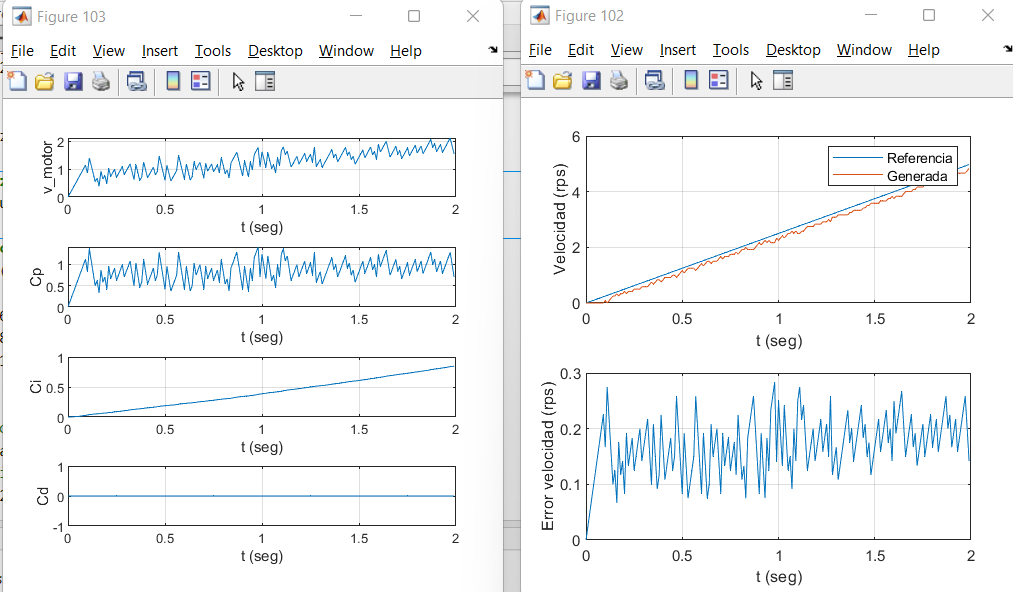
PI K = 5 TI = 10\*tm



PI K = 5 TI = 200 \* Tm

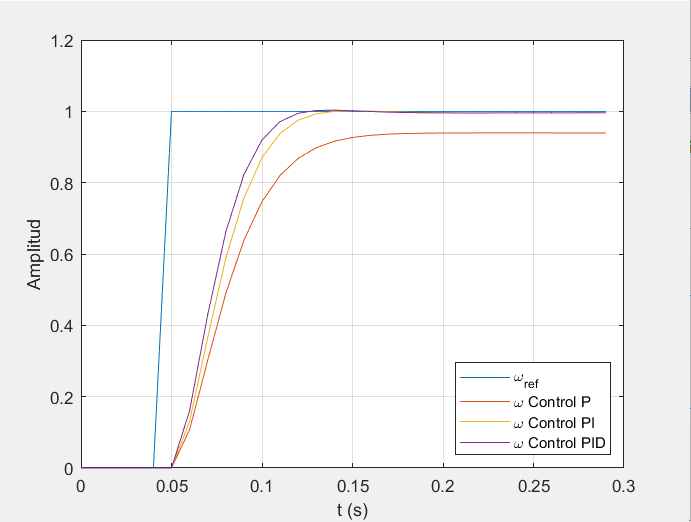


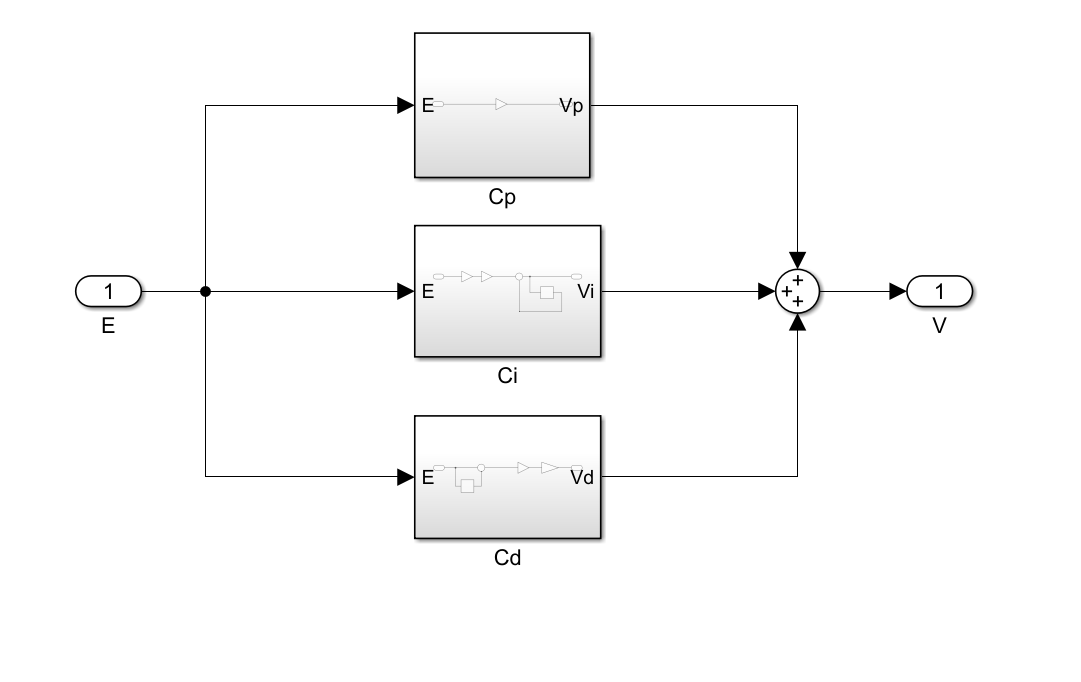
PI RAMPA PI K = 5 TI = 10\*tm



Una vez obtenido el control PI, podemos añadir el control Derivativo, obteniendo el PID completo:

La acción derivativa se manifiesta cuando hay un cambio en el valor absoluto del error; (si el error es constante, solamente actúan los modos proporcional e integral).





VP z = Kp E(z)

VP n = Kp e [n]

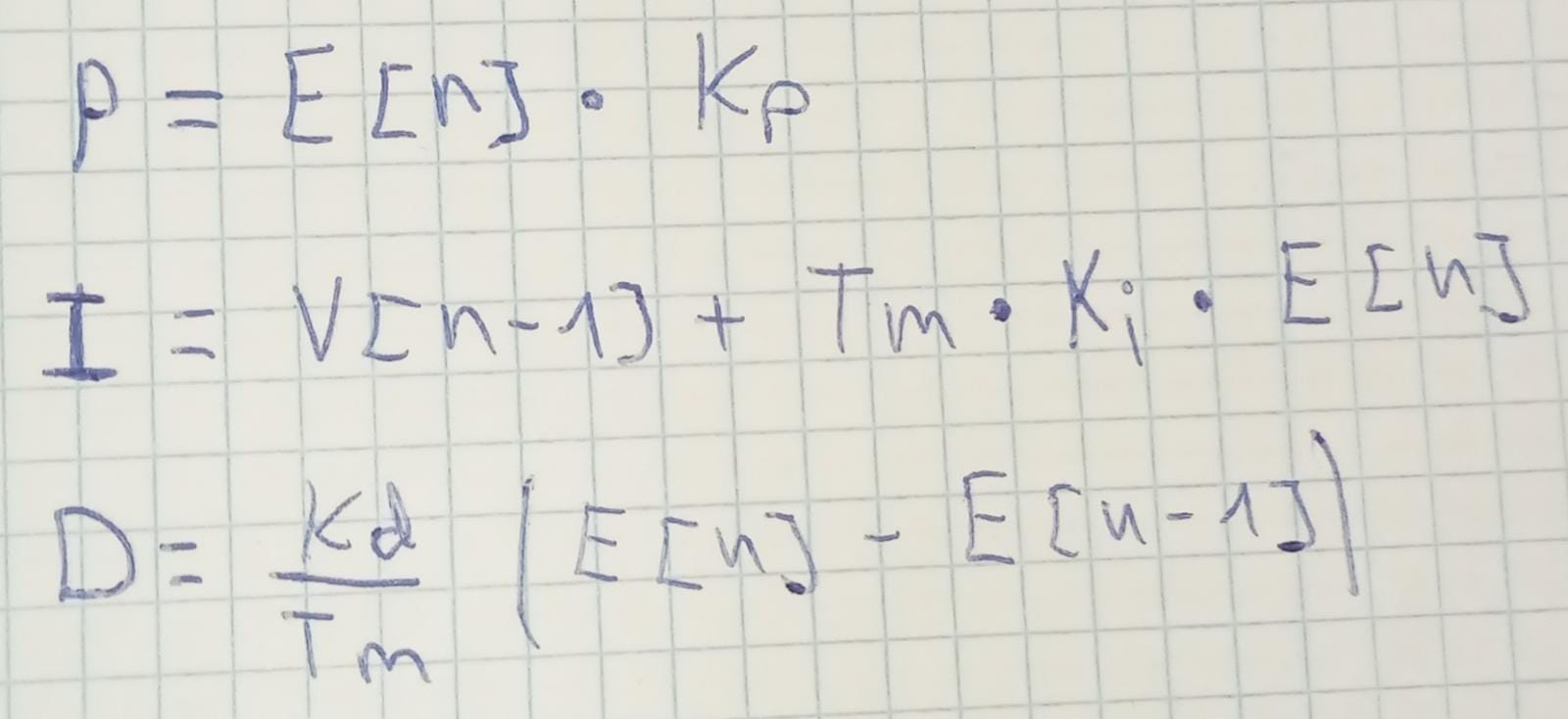
VD z= Kd/ ™ (E z - Ez z^-1)

VD n = Kd/Tm (e[n] - e[n-1])

VI n = Ki \* Tm \* e[n] + Vi [n-1]

PID = VP + VI + VD

**Ecuación en diferencias**



**Breve conclusión**

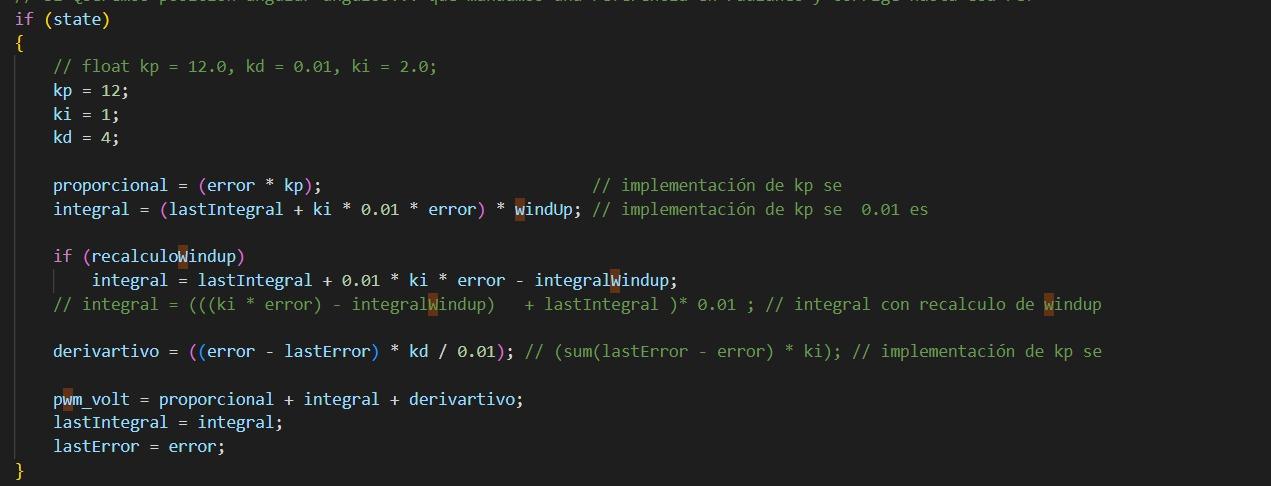
El algoritmo implementa un controlador PID.

El algoritmo de control incluye tres parámetros fundamentales: Ganancia proporcional (P), Integral (I) y Derivativo (D).

* El parámetro Proporcional (P) mide la diferencia entre el valor actual y referencia, aplica el cambio.
* El parámetro Integral (I) se refiere al tiempo que se toma para llevar a cabo acción correctiva. Mientras el valor sea más pequeño, es más rápida la corrección, pero se debe tener precaución, ya que reducir demasiado el valor puede causar inestabilidad.
* El parámetro Derivativo (D) emite una acción predictiva, es decir, prevé el error e inicia una acción oportuna.

2 - Modificación del sistema para que realice el control de posición angular del motor.

**Algoritmo**



**Valor de los parámetros Kp, Ki, Kd**

// VELOCIDAD

kp = 5.0, kd = 0.1, ki = 5.0;

// ANGULO

kp = 10.0, kd = 1.0, ki = 5.0;

**Breve explicación**

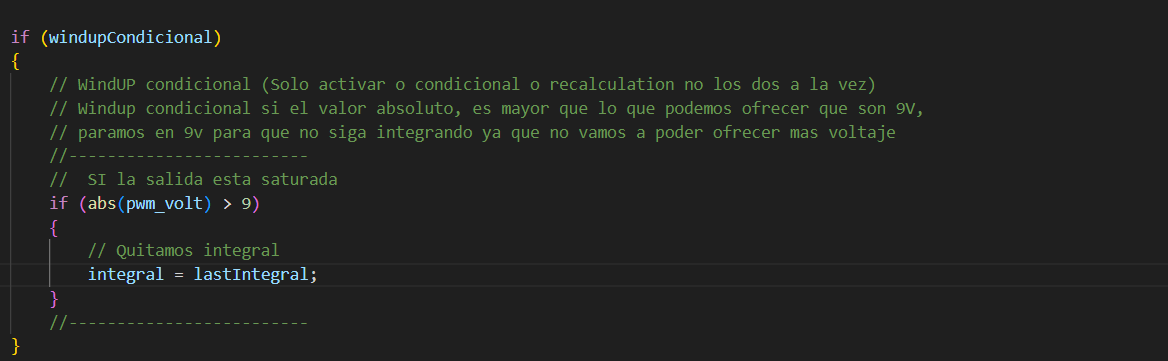
El algoritmo implementa un controlador PID VELOCIDAD KP = 12 KI=2.5 KD = 0.01

La función de transferencia del controlador PID es Cz(z)= K(1 + CI (z) + CD(z)), donde la componente integral es CI (z)= Tmz Ti(z−1) , siendo Ti la constante de tiempo de integración y Tm el periodo de muestreo del sistema, y la derivativa es CD(z)= Td(z−1) Tmz , siendo Td la constante de tiempo derivativa.

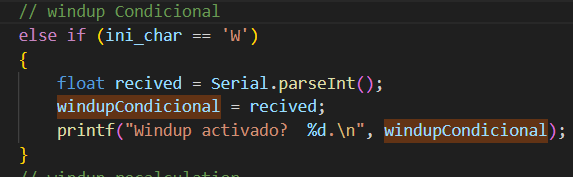
3 - Implementación del algoritmo de anti “windup” condicional.

El windup condicional en un controlador PID es una técnica utilizada para mitigar el efecto de saturación en la salida del controlador. La saturación ocurre cuando la salida del controlador PID alcanza su límite superior o inferior, lo que puede llevar a un comportamiento indeseable.

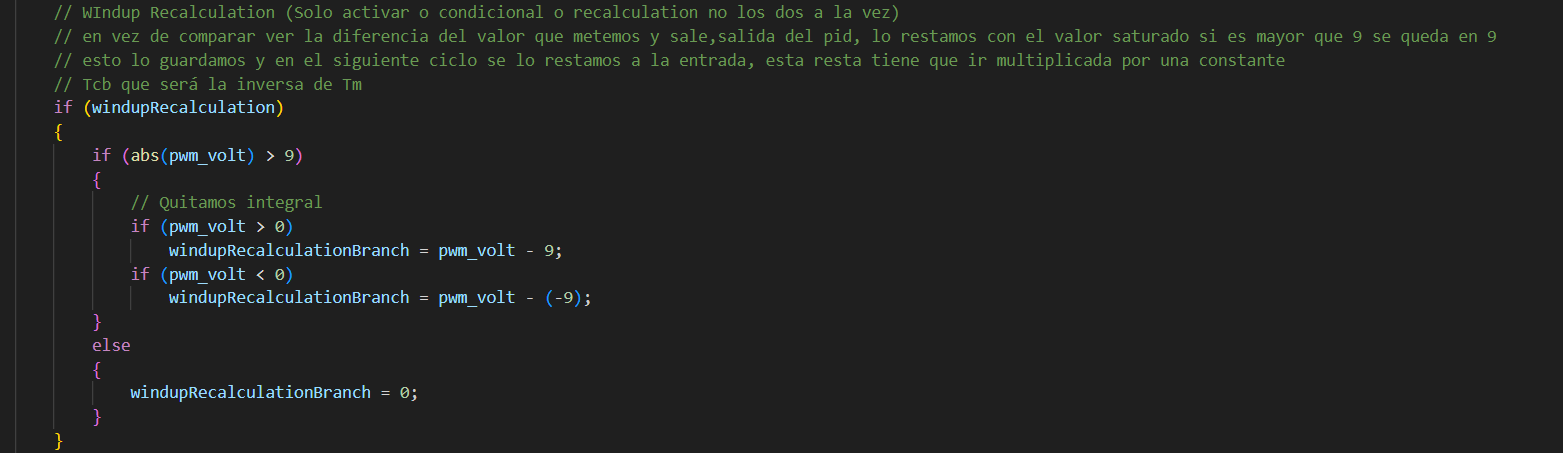
El windup condicional aborda este problema ajustando de manera dinámica los términos integral, del controlador PID cuando la salida está saturada.



Este cálculo de windup, se puede habilitar dinámicamente a través de la terminal serial ingresando W1 para habilitarlo o W0 para deshabilitar lo (mayúsculas).



3.1 - Implementación del algoritmo de anti “windup” con recálculo.

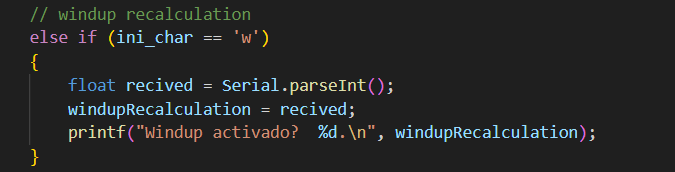
****

**Ecuación en diferencias**

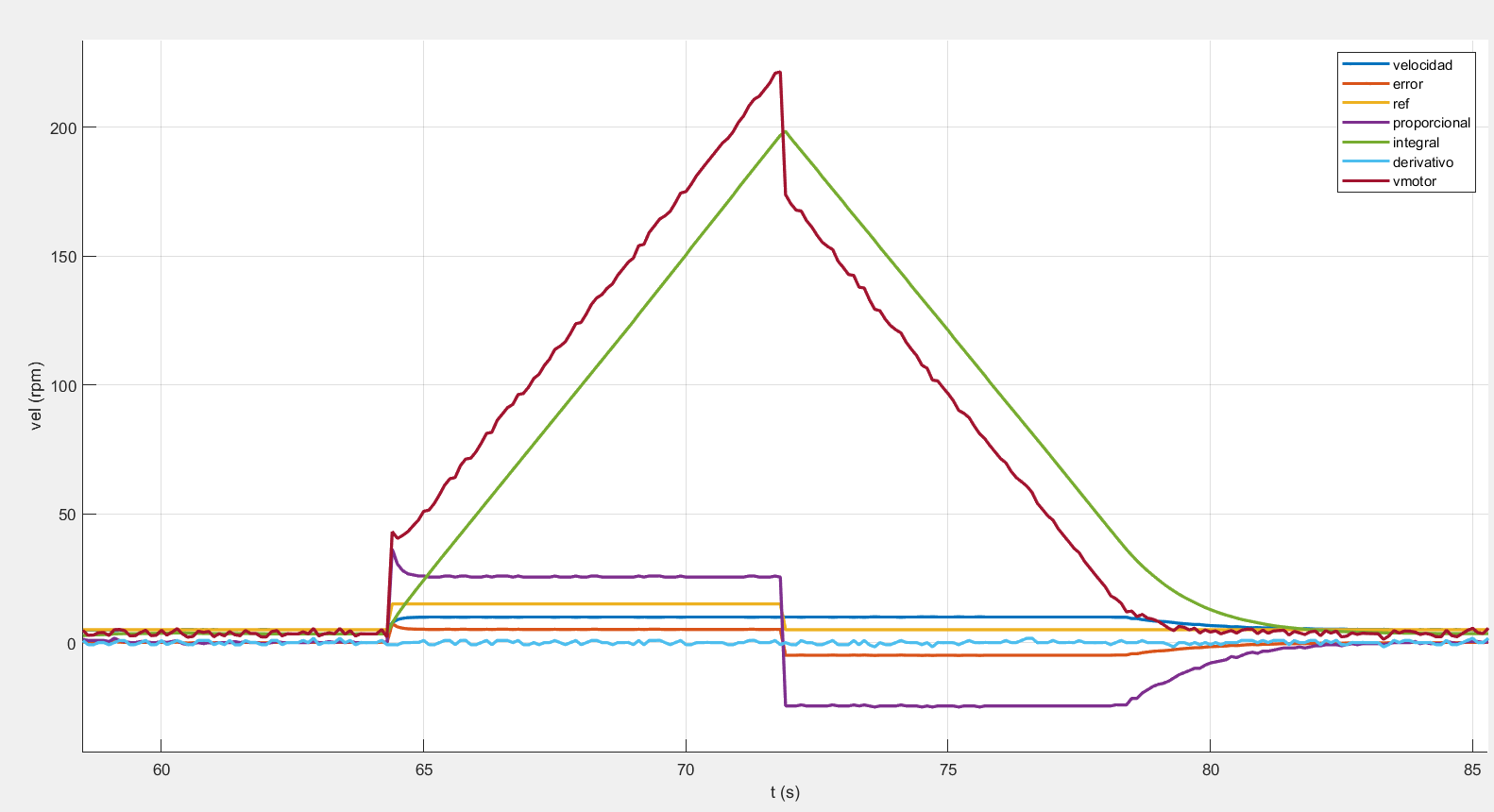


El algoritmo implementa el algoritmo de windup (back-calculation) con este algoritmo el integrador se incrementa (o decrementa) cuando se alcanza el límite de saturación mínimo (o máximo) de forma proporcional a la diferencia entre la salida del controlador y su valor saturado.

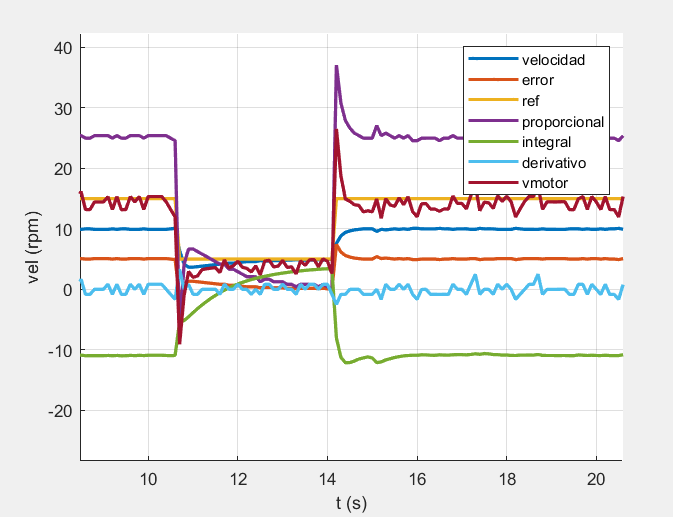
Se puede habilitar windup re-calculation mediante la terminal serial introduciendo w1 para habilitarlo o w0 para deshabilitar lo (minúsculas)



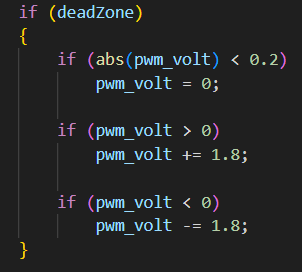
A continuación podemos ver como el apartado integral (sin windup), se acumula hasta que la salida deja de estar saturada.



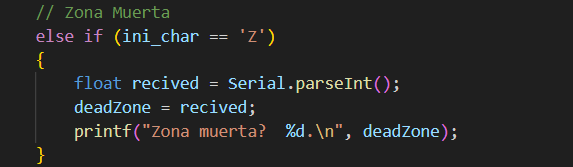
Y en el siguiente ejemplo (con windup) podemos visualizar como se detiene el acumulativo del apartado integral, debido a que tenemos activado el windup. Obteniendo un cambio mucho más rápido y preciso.



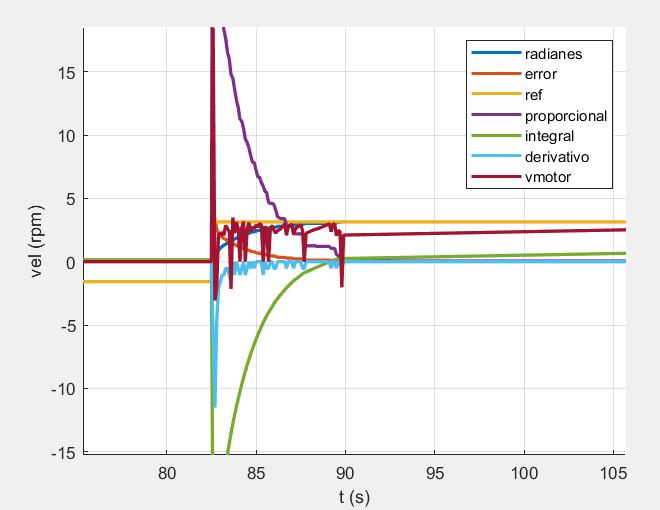
5 - Implementación del algoritmo de anti zona muerta.



Se puede habilitar anti zona muerta, mediante la terminal serial introduciendo Z1 para habilitarlo, Z0 para deshabilitar lo.



El algoritmo de anti-zona muerta en un controlador PID detecta la saturación de la salida y ajusta la acción integral para contrarrestar la zona muerta. Esto evita que el término integral se acumule en exceso durante la saturación y ayuda a mantener un control más preciso y estable.



**CONCLUSIÓN**

En la práctica final, hemos aprendido a aplicar los conocimientos adquiridos en la parte teórica de la asignatura para configurar un controlador PID y obtener una respuesta capaz de ser controlada. También hemos trabajado con los algoritmos anti-windup y anti zona muerta, los cuales nos han sido útiles para solucionar problemas que se daban en este caso específico. Particularmente nos ha servido para mejorar un proyecto de un robot velocista sigue líneas en un futuro.

[https://www.instagram.com/p/Bs0knd0Pnn5o/](https://www.instagram.com/p/Bs0kn0Pnn5o/)