DEPARTAMENTO DE ELECTRÓNICA Y AUTOMÁTICA

FACULTAD DE INGENIERÍA – UNIVERSIDAD NACIONAL DE SAN JUAN

Informe de Practica Nº 1

LAZO DE CONTROL IMPLEMENTADO EN UN MICROCONTROLADOR

Asignatura: Sistemas Para Control

Ingeniería Electrónica

Autores:

Fabaro, Verónica – Registro 25434

Dominguez, Cristian – Registro 20406

2º Semestre

Año 2024

1. Introducción.

El presente trabajo práctico tiene como finalidad la implementación de un lazo de control de posición para un motor de corriente continua utilizando un controlador PID. Para ello, se empleará un sensor resistivo tipo potenciómetro para medir la posición angular del motor, mientras que con un microcontrolador Arduino UNO se procesarán las señales y se calculará la acción de control necesaria para ajustar la posición según la referencia impuesta. La señal de referencia será generada mediante un teléfono celular utilizando la app HyperIMU, lo que permitirá una comunicación inalámbrica con el sistema.

El objetivo principal es desarrollar un sistema de control capaz de mantener un error de posición menor a un grado en estado estacionario. La implementación del software de control, la calibración del controlador PID y la integración del sistema con Matlab para el análisis y simulación de resultados serán elementos clave para evaluar el desempeño del lazo de control.

2. Contenido.

2.1 Actividades a desarrollar

A continuación, se detallan las actividades a desarrollar, que comprenden desde la instalación de software necesario hasta la implementación de un lazo de control PID en un microcontrolador

2.1.1Instalación del entorno de desarrollo Arduino

El primer paso consiste en descargar e instalar el software Arduino desde la página oficial. Este entorno de desarrollo será utilizado para programar y cargar el código en la placa Arduino UNO, que será la encargada de controlar el motor DC.

2.1.2. Configuración de la App HyperIMU en un teléfono Android

Se procederá a descargar e instalar la aplicación HyperIMU desde Google Play en un teléfono Android. Esta app será utilizada para generar la señal de referencia para el control del motor, obtenida de los sensores disponibles en el teléfono.

2.1.3. Instalación y configuración de Matlab con SASTOOLS

A continuación, se instalará el software Matlab junto con la librería SASTOOLS, proporcionada por la cátedra. Esta librería será utilizada para modelar y simular el sistema de control del motor DC, así como para visualizar y analizar los resultados obtenidos en tiempo real desde la aplicación HyperIMU.

2.1.4. Implementación de un lazo de control PID en el microcontrolador

El objetivo principal de esta etapa es implementar un lazo de control PID en el microcontrolador para controlar la posición angular del motor DC, que estará acoplado a una rueda de plástico. El controlador PID incluirá características como: Parte proporcional, integral y derivativa, Control de pendiente y saturación de la acción de control, y Antiwindup para prevenir la acumulación excesiva en la componente integral. La referencia para la posición del motor será obtenida desde los sensores del teléfono celular configurado previamente.

2.1.5. Calibración del controlador PID

Se procederá a la calibración del controlador PID con el objetivo de lograr un error de control en estado estacionario menor a 1 grado para una referencia constante. Para ello, se ajustarán los parámetros del controlador.

2.1.6. Análisis de señales y simulación en Simulink

Finalmente, se deberán graficar y analizar las señales de referencia, salida, y acción de control en sus respectivas unidades físicas. Utilizando el modelo de Simulink proporcionado por la cátedra, se compararán los resultados obtenidos en la simulación con los valores observados durante la implementación física del sistema, evaluando la efectividad del lazo de control.

2.2 Implementación de la planta

Para la implementación física de la planta se siguió el esquema proporcionado por la cátedra, que consistió en el montaje de un motor DC acoplado a un potenciómetro, que permitía medir la posición angular del eje del motor. La precisión y estabilidad del control fueron gestionadas a través del microcontrolador Arduino UNO implementado.

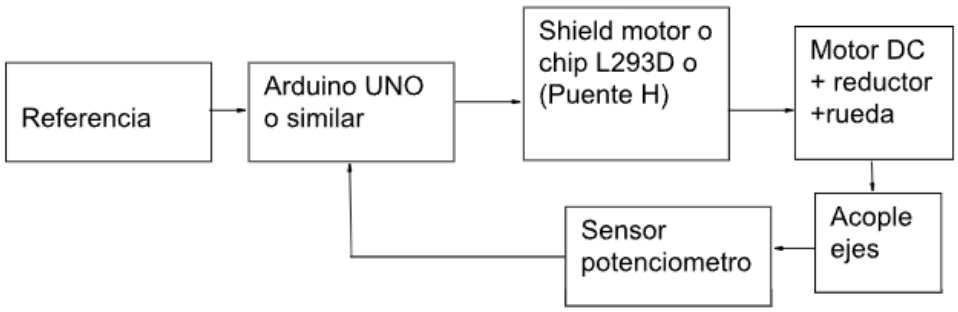

Fig.: 1 Diagrama de Bloques

A continuación, se detallan los componentes utilizados para armar el sistema:

- **Arduino UNO**: fue programado para ejecutar el controlador PID. Su función principal fue calcular la acción de control necesaria para ajustar la posición del motor, basándose en la diferencia entre la señal de referencia recibida desde HyperIMU y la posición actual medida por el potenciómetro.
- **Shield motor L293D**: Este componente actúa como un puente H dual que permite controlar tanto la velocidad como la dirección del motor mediante señales PWM enviadas desde el Arduino. El L293D maneja corrientes más altas que las que el microcontrolador puede suministrar, garantizando que el motor reciba la energía necesaria para operar correctamente. Además, su diseño permite invertir la dirección del motor.

En la Fig. 2 se pueden observar las placas Arduino y L293D. El diseño de ambas facilita una conexión rápida, ya que al montar una sobre la otra, como se muestra en la Fig. xx, se garantiza la correcta interconexión entre ellas. Este ensamblaje permite que el sistema esté listo para funcionar de inmediato.

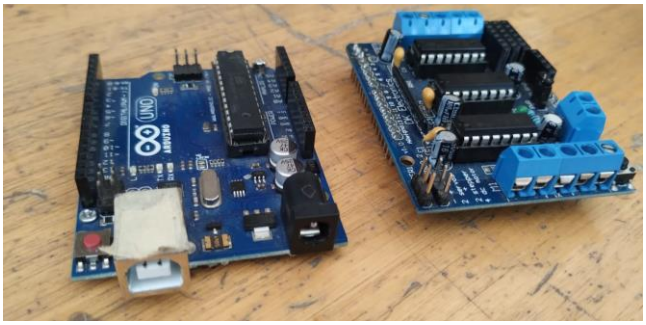

Fig.: 2 Placa Arduino UNO Y Placa Shield L293D utilizadas

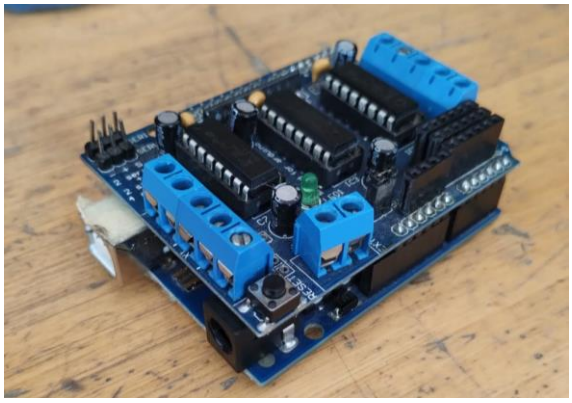

Fig.: 3 Montaje de placas Arduino Uno y Placa Shield L293D

- **Potenciómetro**: El potenciómetro fue acoplado al eje del motor, actuando como un sensor de posición. La lectura de este sensor fue clave para la retroalimentación del lazo de control.
- **Motor DC con Reductor**: El motor utilizado era un motor de corriente continua con un reductor de velocidad para mejorar la precisión del control de la posición angular. El reductor permitía tener un mayor control sobre el ángulo de giro del eje y una respuesta más precisa al aplicar el controlador PID.

Para el montaje del sistema, se utilizó una base impresa en 3D de 13x13 cm, diseñada específicamente con el objetivo de mantener la alineación precisa de los componentes del proyecto, mostrada en las Fig. 4 y Fig. 5. Sobre esta base, se levantó una columna de 9,5 cm de altura donde se colocó el motor, orientado con la rueda hacia arriba. En el eje opuesto, se añadieron acoples reductores para conectar el motor con el potenciómetro, que estaba fijado a la base, permitiendo una correcta transmisión del movimiento y la medición precisa de la posición angular. Los acoples utilizados son de tipo cilíndrico uno metálico y un par adicional mediante impresión 3D para adecuarlos al motor y al potenciómetro.

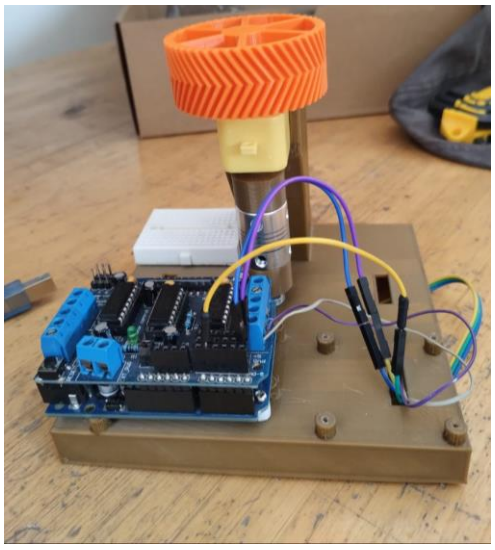

Fig.: 4 Vista Frontal de la planta

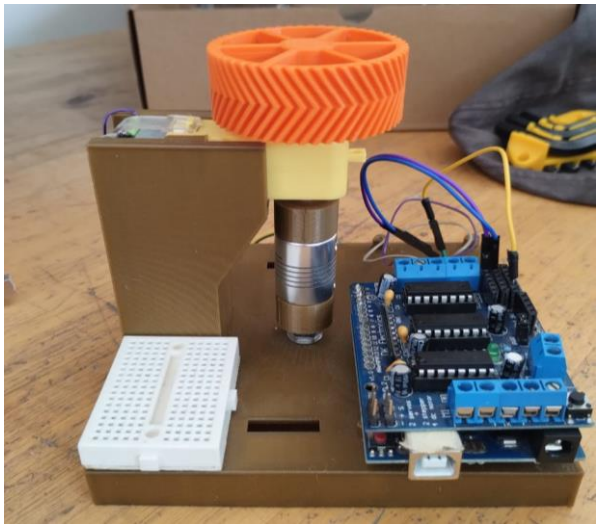

Fig.: 5 Vista lateral de la planta

2.3 Comunicación

2.3.1 HyperIMU

Los sensores IMU son dispositivos que miden la aceleración lineal, la velocidad angular y, en algunos casos, el campo magnético, lo que permite registrar datos de movimiento y orientación del dispositivo.

Para la conexión con la planta lo primero que se realizó fue vincular la computadora con la aplicación HyperIMU, la cual facilita la extracción de estos datos del teléfono. El teléfono configurado como un punto de acceso, permitió que la notebook se conectara a la red local generada por el teléfono.

Para completar la configuración, se obtuvo la dirección IP de la computadora desde la terminal de línea de comandos usando el comando ipconfig.

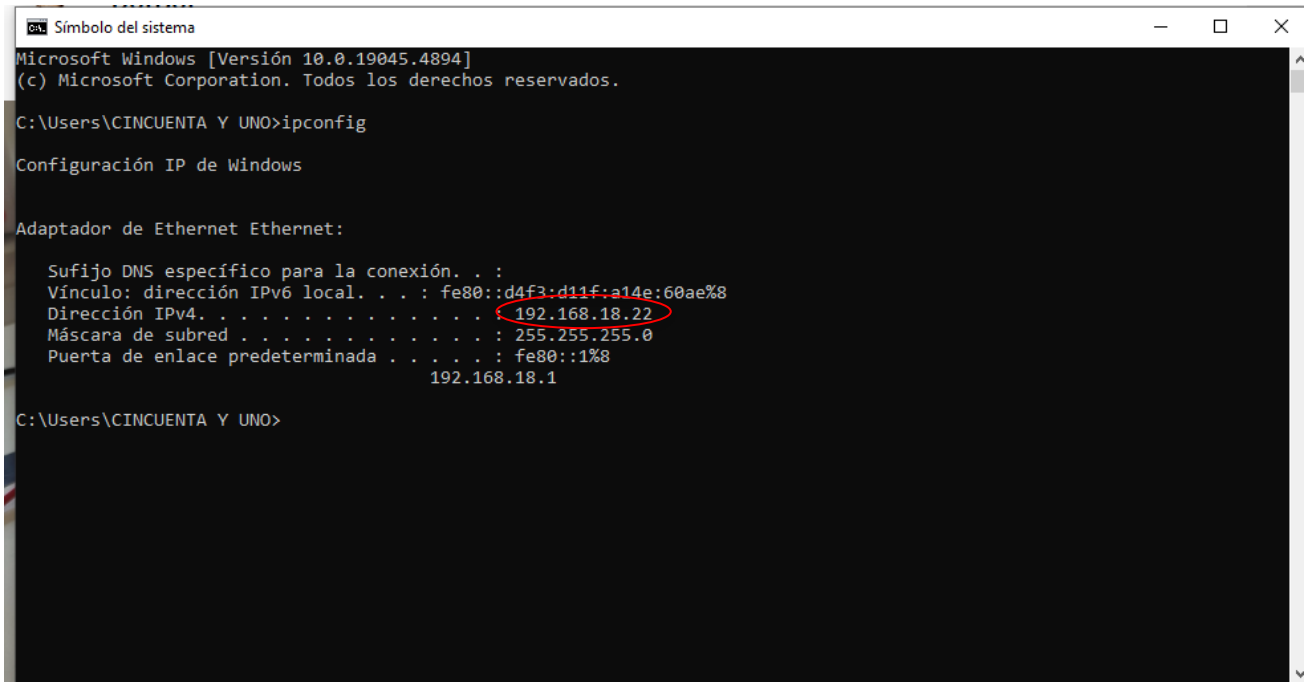

Fig.: 6 Comando ipconfig en terminal de línea de comandos de Windows

Para configurar la aplicación, se ingresa a la lista de sensores para seleccionar un sensor que se utilizará como referencia de entrada. Luego entre las configuraciones, se modificaron los parámetros del protocolo de comunicación, tiempo de muestreo, dirección IP del servidor, y número de puerto del servidor. Estos ajustes permitieron que la app se comunique correctamente con la aplicación de escritorio “HyperIMU.exe”, proporcionada por la cátedra.

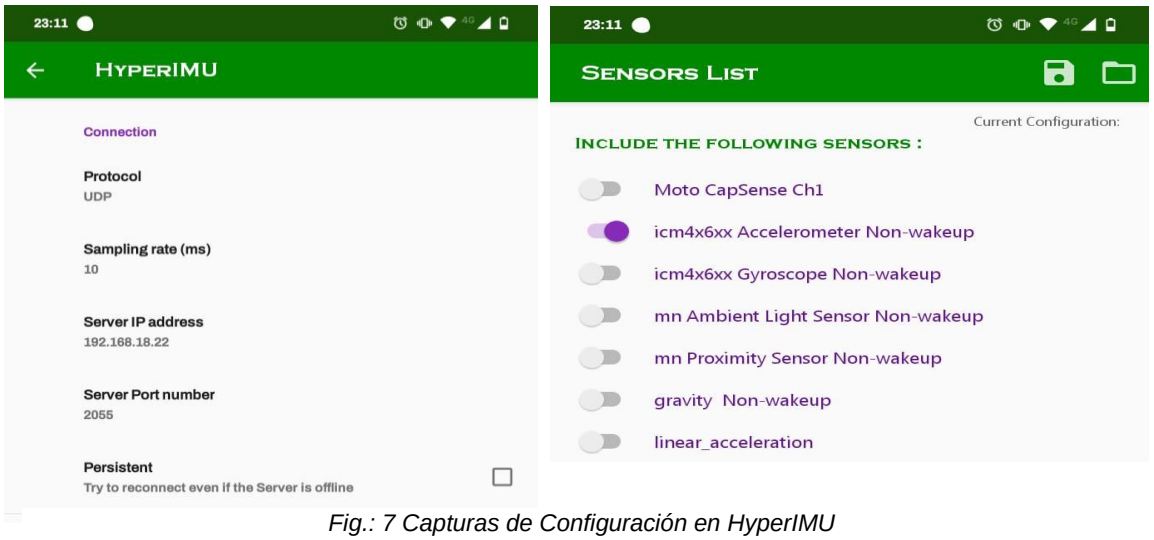

Fig.: 7 Capturas de Configuración en HyperIMU

2.3.2 Matlab

El diagrama en bloques diseñado en Simulink, con pertenecientes a la librería SASTOOLS, consistió en varios componentes que interactúan de forma integrada para simular y controlar la posición del motor DC. Estos bloques incluyen:

- **Adquisición de datos desde HyperIMU**: Se utilizó un bloque de comunicación que permitió la adquisición de los datos de referencia de posición desde la app **HyperIMU**. La app transmitió la información sensorial (posición, aceleración, etc.) a través de una red local. El bloque de Simulink se conectó a este flujo de datos en tiempo real, permitiendo alimentar al sistema con la señal de referencia generada por el sensor del teléfono.
- **Interfaz con Arduino**: Para establecer la comunicación entre el sistema y el hardware físico, se emplearon bloques de Simulink que permitieron la interacción bidireccional entre Arduino y Matlab, los cuales enviaron las señales de referencia obtenidas desde HyperIMU al Arduino UNO para controlar el motor DC.
- **Simulación del sistema**: Paralelamente a la adquisición de datos y el control del motor en tiempo real, se realizó una simulación del comportamiento del sistema, a través del bloque scope se pudo observar este comportamiento. Esto permitió verificar el desempeño del controlador PID bajo diferentes condiciones, y ajustar los parámetros de control para minimizar el error de posición y mejorar la respuesta dinámica del sistema.

En la Fig. xx se puede ver el diagrama de bloques utilizado en Simulink, donde se realizaron las conexiones del puerto serie correspondiente al Arduino. Se calibró el tiempo de muestreo en todos los bloques y se puso en marcha la simulación.

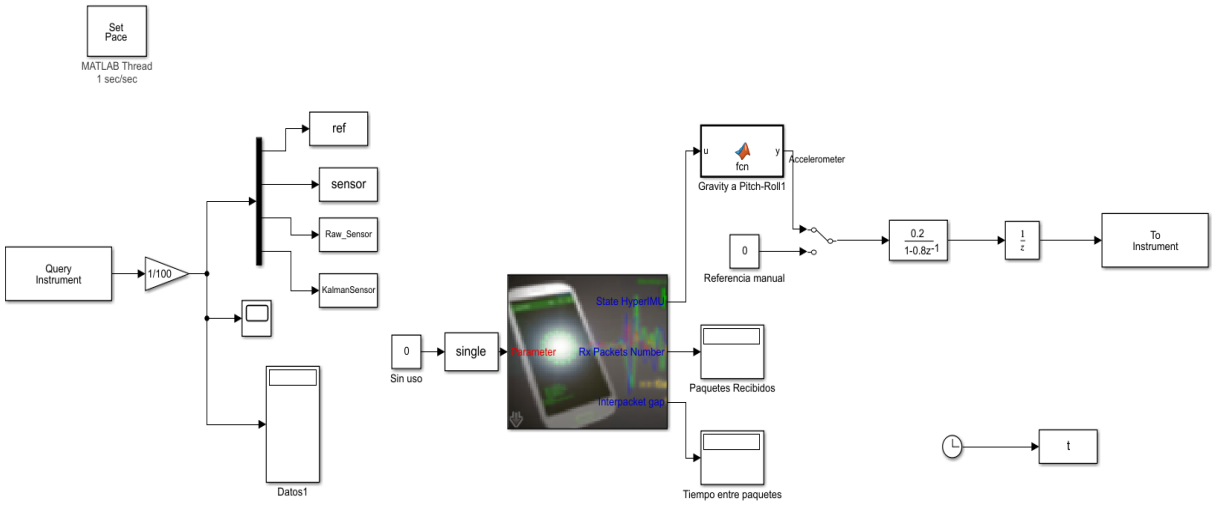

Fig.: 8 Diagrama de Bloques en Simulink

2.3.3 Arduino

Se utilizó una placa **Arduino UNO** como plataforma de control del motor de corriente continua. Arduino fue el microcontrolador encargado de ejecutar el algoritmo de control PID y manejar la interacción con los sensores y actuadores, permitiendo implementar el lazo de control físico que interactúa con el sistema de referencia proporcionado por la app HyperIMU.
El teléfono, configurado como punto de acceso, enviaba los datos de referencia de posición al Arduino por medio de una red local. Arduino recibió estos datos y los utilizó como la entrada al lazo de control para ajustar la posición del motor en tiempo real.
Entonces, el código fue ajustado para adaptarse a las características del motor y las exigencias de la práctica, lo que incluyó:

- **Lectura del potenciómetro**: Arduino obtuvo la posición actual del motor DC mediante un sensor resistivo tipo potenciómetro acoplado al eje del motor. Esta lectura se usó como retroalimentación en el lazo de control para calcular el error con respecto a la señal de referencia.

- **Cálculo de la acción de control**: El error calculado entre la posición de referencia y la posición actual del motor se introdujo en el algoritmo PID, que aplicó la acción proporcional, integral y derivativa. El resultado fue una señal de control que representaba el ajuste necesario para llevar el motor a la posición deseada.
- **Generación de señal PWM**: Arduino envió la señal de control al motor utilizando PWM para regular la velocidad del motor en función de la acción de control calculada.

En el apartado referencias de este informe se colocará un link al repositorio Git con el código modificado.

2.4 Ajustes y Mejoras en el Código

2.4.1 Modificaciones en los parámetros del motor

La selección del punto muerto para el motor DC fue un paso necesario en el diseño del sistema de control para evitar daños en el motor debido a la acción de control. Este punto se determinó conectando el motor a una fuente de 0 V y aumentando gradualmente la tensión hasta que el motor apenas comenzó a moverse, localizado aproximadamente en 1.2 V. Una vez establecido, este valor se aplicó en el código de Arduino, lo que permitió al controlador PID establecer un punto de referencia preciso para las operaciones de posicionamiento del motor. En la fig xx se representa la modificación realizada en el código.

```
#define SALIDA_MAX 3
#define TENSION_MAX 3.5
#define ZONA_MUERTA 1.2
#define TOLERANCIA 2.0
#define REF_MAX 90.0
#define REF_MIN -90.0
```

Fig.: 9 Valores Modificados en Programa Arduino

2.4.2 Modificación en constantes del PID

Un controlador PID (Proporcional-Integral-Derivativo) es un mecanismo de control utilizado en sistemas de automatización y control para mantener una variable en un valor deseado. Este tipo de controlador ajusta la salida del sistema basándose en tres parámetros:

- Proporcional (Kp): Responde a la diferencia entre el valor deseado y el valor actual (error). Un valor proporcional alto puede hacer que el sistema responda rápidamente, pero también puede causar oscilaciones si es demasiado alto.
- Integral (Ki): Considera la suma de los errores pasados. Ayuda a eliminar el error acumulado a lo largo del tiempo, asegurando que el sistema alcance el valor deseado sin desviaciones persistentes.
- Derivativo (Kd): Responde a la tasa de cambio del error. Ayuda a predecir la tendencia del error y a suavizar la respuesta del sistema, reduciendo las oscilaciones.

Para ajustar manualmente los parámetros del controlador PID, se establecieron valores de referencia manuales en 90, 0 y -90. Estos valores representan los puntos límites que el sistema debe alcanzar. Luego, con los valores de referencia establecidos, se procedió a ajustar los parámetros del controlador PID (Kp, Ki y Kd).

Se realizaron múltiples pruebas variando uno a uno los valores de Kp, Ki y Kd para observar cómo cada parámetro afectaba la respuesta del sistema. En la fig xx se observa la línea de código modificada en el programa correspondiente a las constantes del PID.

```
float Kp=0.5 , Ki=0.03, Kd=0.01;
```

Fig.: 10Modificación de Parámetros del PID

2.5 Resultados

Para cada combinación de parámetros, se registraron las respuestas del sistema. Esto incluyó la estabilidad, el tiempo de respuesta y la precisión con la que el sistema alcanzaba los valores de referencia. Se generaron imágenes y gráficos que muestran las respuestas del sistema bajo diferentes configuraciones de Kp, Ki y Kd.

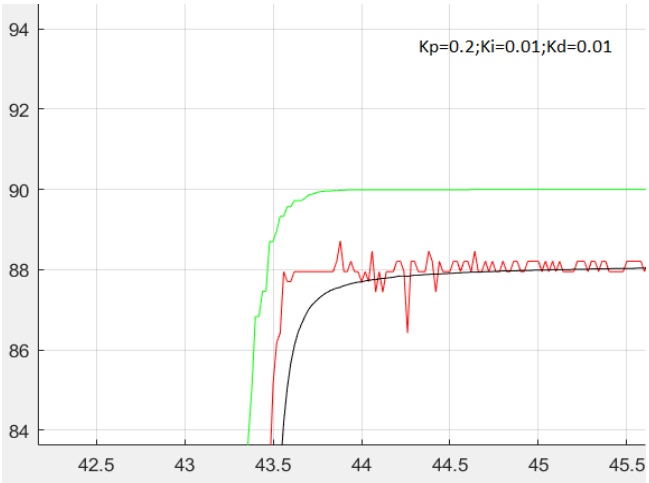

Fig.: 11Respuesta del sistema con Kp=0.2,Ki=0.01 y Kd=0.01.

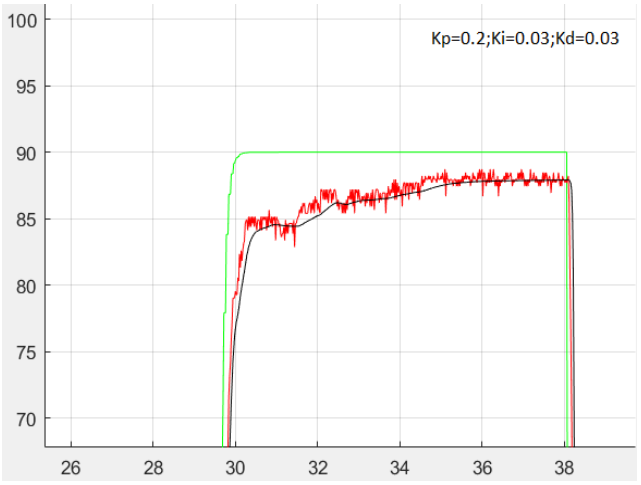

Fig.: 14 Respuesta del sistema con Kp=0.2,Ki=0.03 y Kd=0.03.

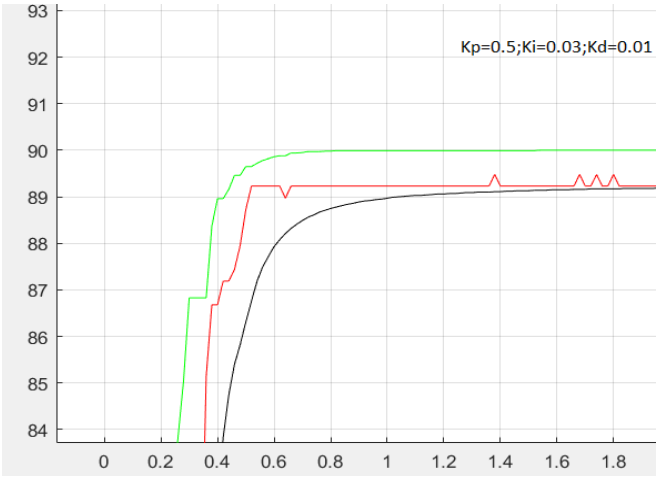

Fig.: 13 Respuesta del sistema con Kp=0.5,Ki=0.03 y Kd=0.01.

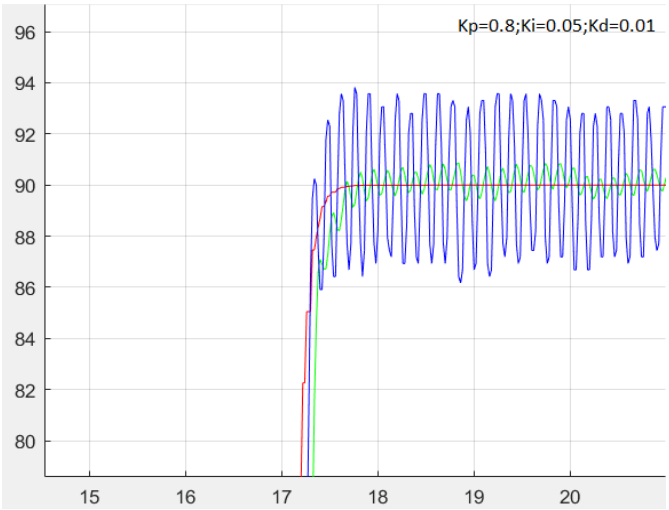

Fig.: 12 Respuesta del sistema con Kp=0.8,Ki=0.05 y Kd=0.01.

Según la gráfica, los valores de Kp=0.5, Ki=0.03, y Kd=0.01 proporcionaron la mejor respuesta, con un error menor a un grado. Esto indica que estos parámetros lograron un buen equilibrio entre rapidez y estabilidad, minimizando el sobreimpulso y las oscilaciones.

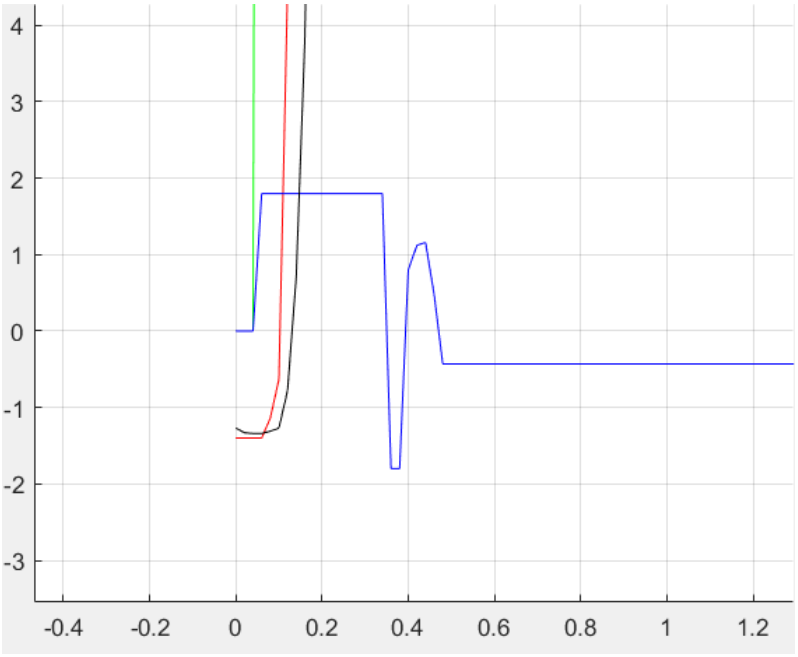

Fig.: 15 Respuesta del sistema al ingreso de referencias desde HyperIMU

La acción de control se refiere a cómo el controlador ajusta la salida para minimizar el error entre una variable de proceso medida y el punto de ajuste deseado. La acción de control se genera a partir de un punto muerto de 1.2 voltios, más una salida adicional de aproximadamente 0.5 voltios. Esto ocurre inicialmente porque, al establecer una referencia de posición, la acción de control actúa en el encendido del motor. Una vez encendido, el motor se estabiliza y la realimentación busca corregir el error de posición, por eso los valores dados en la gráfica.

3. Conclusiones.

La presente práctica permitió implementar un lazo de control PID para el control preciso de la posición de un motor DC, integrando tanto herramientas de software como hardware. A través de la utilización de Matlab y Simulink para la simulación, y Arduino para el control físico, se logró desarrollar un sistema robusto capaz de interactuar con sensores externos, en este caso, un teléfono móvil con la aplicación HyperIMU.

Las modificaciones al código original, que incluyeron la implementación de un anti-windup y el control de pendiente, fueron esenciales para optimizar el comportamiento del sistema, minimizando oscilaciones y asegurando una respuesta estable ante variaciones de la referencia. El enfoque modular y experimental de la práctica, con la integración de diferentes tecnologías, nos brindó una experiencia completa en el diseño y ajuste de sistemas de control en tiempo real.

Esta práctica reforzó los conceptos teóricos del control PID, y también demostró su aplicación práctica en el control de motores DC, destacando la importancia de la simulación previa y el ajuste fino en la implementación física.

4. Referencias.

- Link al repositorio Git:
  Practica1SPC