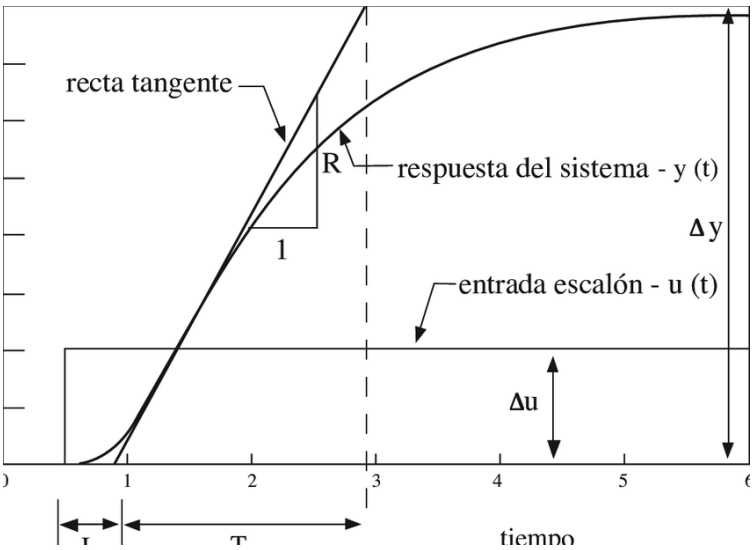

OBJETIVOS

- Implementar un sistema de medición de velocidad angular (RPM) para un motor DC, utilizando un encoder óptico conectado a la Raspberry Pi Pico.
- Controlar la velocidad de un motor DC mediante la generación de una señal PWM ajustable desde la Raspberry Pi Pico, implementando el sistema tanto en Arduino como en MicroPython.
- Caracterizar experimentalmente un motor DC usando la curva de reacción, relacionando el voltaje aplicado (controlado mediante PWM) con la velocidad obtenida, y analizar los resultados para encontrar un modelo de primer orden del motor.

DESCRIPCIÓN

En el diseño de sistemas embebidos aplicados al control de actuadores, es fundamental comprender cómo responden los motores DC a distintas señales de entrada. En particular, conocer la relación entre el voltaje aplicado y la velocidad del motor es clave para implementar sistemas de control efectivos.

La curva de reacción es una gráfica que relaciona la entrada aplicada al sistema (voltaje mediante PWM) con la salida obtenida (RPM). Se construye aplicando valores incrementales de entrada, midiendo la respuesta y graficando los resultados. Esta curva es fundamental para caracterizar el sistema, diseñar estrategias de control, y evaluar su desempeño.

En esta práctica, se utilizará una Raspberry Pi Pico con implementaciones independientes en Arduino (C/C++) y MicroPython. La señal PWM será utilizada para controlar el voltaje del motor DC, mientras que un encoder óptico medirá la velocidad en RPM. El objetivo final es obtener y analizar la curva experimental del motor.

PROCEDIMIENTO

1. **Preparación del hardware**

- Seleccionar un motor DC pequeño sin reductor de velocidad y un driver de potencia adecuado para suministrar y controlar la potencia del motor.
- Seleccionar un encoder óptico y la rueda dentada adecuada para el motor seleccionado.

- Diseñar e implementar el montaje físico para sujetar el motor y adecuar la rueda dentada requerida por el enconder óptico. Conectar el motor DC con el driver de potencia cuya señal PWM será controlada desde la Raspberry Pi Pico.
- Instalar encoder óptico en el eje del motor y verificar el correcto funcionamiento de este. Es decir que los pulsos se generen correctamente en todo el rango de velocidad del motor.

2. **Implementación del sistema de medición de RPM**

- Implementar un programa simple para la medición de RPM en la Raspberry pi pico, tanto en Arduino como en microPython.
    - Captura los pulsos del encoder ópticos con el MCU.
    - Convierte los pulsos a RPM, asegurando lecturas precisas.
- Verificar la correcta medición de las RPMs en todo el rango de velocidades del motor DC.
- Documentar el código de ambos programas en Doxygen.
- Comparar el rendimiento en Arduino vs el rendimiento en microPython.

3. **Implementación del sistema de control en lazo abierto del motor a través de señal PWM**

- Implementar un programa simple para generar la señal de PWM que controla la potencia suministrada al motor DC. Este programa debe implementarse tanto en Arduino como en microPython. La señal debe estar en capacidad de generar una señal cuadrada con porcentaje de ciclo de dureza variable entre el 0 y el 100% con una resolución mínima del 1%. Es decir, el cambio más pequeño soportado debe ser como mínimo del 1%.
- Verifica ajustes estables del PWM entre 0% y 100% y que estos ajustes efectivamente varíen la velocidad del motor.
- Encuentre el porcentaje de PWM más pequeño que hace que el motor empiece a girar.
- Documentar el código de ambos programas en Doxygen.
- Comparar el rendimiento en Arduino vs el rendimiento en microPython.

4. Implementar sistema de captura de la curva de reacción

- Implementar un programa simple para capturar la curva de reacción de un motor DC tanto en Arduino como en microPython
- La curva de reacción permite visualizar el transiente de velocidad del motor producido por el cambio repentino en la señal de PWM (escalón). Para observar adecuadamente las características dinámicas del motor como el tiempo muerto es necesario muestrear la velocidad del motor lo suficientemente rápido. Para efectos de esta práctica usaremos una frecuencia mínima de 250 Hz. Esto quiere decir que cada 4 ms debo medir la velocidad del motor.
- El proceso de caracterización de la curva de reacción consiste entonces en variar el PWM en escalones de un porcentaje constante de PWM (20% por ejemplo) entre 0 y 100%. La variación se realiza primero de forma ascendente y luego de forma decreciente. El tiempo entre escalones de PWM debe ser superior a por lo menos 10 veces el tiempo de establecimiento del motor. Para motores de juguetería sin carga, el tiempo de establecimiento esta entre los 50 ms y los 150 ms aproximadamente. Esto nos dice que un tiempo razonable entre pasos de PWM puede estar alrededor de los 2 segundos.
- En paralelo a las variaciones en la señal de PWM el sistema debe almacenar en un buffer en memoria, una marca de tiempo respecto al tiempo de inicio del proceso de caracterización, el PWM y las RPM del motor medidas en ese instante de tiempo.
- Al finalizar la secuencia de escalones, el MCU debe enviar la información almacenada en el buffer al PC por medio de la interfaz UART o por el USB. Los datos capturados se envían en formato CSV.
- Documentar el código de ambos programas en Doxygen.
- Comparar el rendimiento en Arduino vs el rendimiento en microPython.

5. **Implementación del sistema de caracterización de un motor DC**

- Implementar el sistema de caracterización del motor DC de tal forma que permita medir en todo instante la velocidad en RPM del motor y el PWM aplicado al mismo, adicionalmente este sistema permite capturar la curva de reacción del motor. Para controlar el sistema es necesario desarrollar una interfaz de comandos a través una terminal. Este sistema debe implementarse tanto en Arduino como en microPython.
- Implemente una interfaz serial con dos comandos:
    1. START <valor>: Inicia automáticamente la captura de la curva experimental, almacenando temporalmente los datos (MARCA-TEMPORAL, PWM y RPM) en un buffer en memoria. Los datos se envían al PC al finalizar la secuencia de escalones ascendentes y descendentes. El parámetro valor indica la magnitud del cambio del PWM durante la secuencia de captura. Tenga en cuenta que valor no es un divisor exacto de 100, el valor máximo del PWM no puede superar 100.
    2. PWM <valor>: Cuando el sistema no este efectuando ninguna captura, es posible ajustar manualmente el ciclo de dureza del PWM. El parámetro valor indica el ciclo de dureza que debe generar el MCU para aplicar al motor. Mientras el sistema está en este modo de ejecución (NO CAPTURA), el MCU envía continuamente al PC los valores actuales de PWM y RPM con una frecuencia de 2Hz.
- Documentar el código de ambos programas en Doxygen.
- Comparar el rendimiento en Arduino vs el rendimiento en microPython.

6. **Encuentra el modelo analítico del motor DC**

- Modela el motor como un sistema lineal de primer orden con retardo. Para esto puedes usar múltiples capturas de la curva de reacción del motor para diferentes magnitudes de variación.
- Compara la respuesta modelo del motor encontrado y la respuesta del motor real en las curvas de reacción aplicando el mismo patrón de cambios en el PWM al modelo analítico.
- Grafique los resultados de la comparación.

7. Reporte en formato IEEE

- Escriba un reporte de los pasos anteriores. Adicionalmente realice una descripción del hardware y software implementados para el sistema de caracterización del motor DC. Acompañe su informe de diagramas de bloque para el hardware y diagramas de flujo o seudocódigo para el software. Describa los problemas (En caso de existir) de su implementación y la posible causa de estos. No olvide finalizar el reporte con las principales conclusiones de los resultados obtenidos.
- Suba el reporte y el código fuente (4 proyectos) en un solo archivo comprimido a la plataforma del curso antes de la media noche de lunes 21 de abril de 2025. Marque el archivo con el numero de la práctica y el primer apellido de cada integrante del equipo de laboratorio.

EVALUACIÓN

- Funcionamiento 40%
    - Medición de RPMs (4% Arduino y 4% microPython)
    - Generación PWM Motor (4% Arduino y 4% microPython)
    - Captura curva de reacción (4% Arduino y 4% microPython)
    - Sistema caracterización del motor DC completo (8% Arduino y 8% microPython)
- Sustentación 30%
    - Dominio del código del sistema y la lógica del mismo (5% Arduino y 5% microPython)
    - Identifica las diferencias y limitaciones de cada ambiente de desarrollo (5%)
    - Identifica el origen de los problemas y las acciones de mejora que deberían implementarse para mejorar el rendimiento (10%)
    - Entiende el procedimiento para modelar el motor DC (10%)
- Documentación 10%
    - Documentación Arduino 5%
    - Documentación microPython 5%

- Reporte 20%
    - Redacción, Ortografía y calidad del texto (4%)
    - Diagrama de bloques y explicación (2%)
    - Diagrama de flujo y explicación (2%)
    - Reporte de observaciones de los pasos del procedimiento (8%)
    - Conclusiones (4%)