INFORME DE LABORATORIO 3

Autores: *Angie Paola Jaramillo Ortega, Juan Manuel Rivera Florez*

Laboratorio de Electrónica Digital 3
Departamento de Ingeniería Electrónica y de Telecomunicaciones
Universidad de Antioquia

0.1. Introducción

Esta práctica tiene como objetivo caracterizar un motor DC mediante el análisis de su respuesta ante señales PWM y la medición de su velocidad angular usando un encoder óptico. Se implementó un sistema de adquisición con Raspberry Pi Pico, empleando estrategias de polling, interrupciones y su combinación para comparar su desempeño. La curva de reacción del motor se obtuvo aplicando escalones de PWM y registrando la velocidad en RPM.

0.2. Implementación

Configuración del Sistema

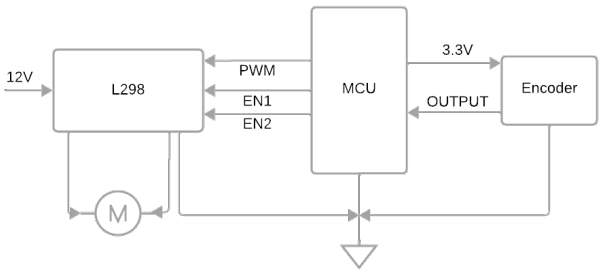

Figura **0-1**: Diagrama de bloques de Hardware

Se utilizó un driver L298N para controlar el motor DC desde la Raspberry Pi Pico mediante señales PWM. La señal PWM se conectó al pin ENABLE del driver, y las salidas del puente H se dirigieron al motor con polarización fija para giro en un sentido. El sistema se alimentó con 12V para el motor y USB para la Raspberry. El encoder óptico fue alimentado con 3.3V desde la Pico y su salida se conectó a un pin digital para el conteo de pulsos generados por un disco de 20 ranuras.

0.2.1. Sistema de medición de RPM

Polling

La lectura de pulsos se realizó por polling, evaluando el cambio de estado del pin asociado al encoder. Los RPM se calcularon cada 500 ms con la fórmula:

$$RPM = \frac{\text{Pulsos}}{\text{PPV}} \cdot \frac{60000}{\text{Tiempo de muestreo(ms)}} \tag{0-1}$$

Este cálculo fue implementado dentro del bucle principal, sin usar interrupciones ni temporizadores, donde 'Pulsos' se refiere a la cantidad de pulsos contados en el intervalo de tiempo y 'PPV' a los pulsos por vuelta.

Interrupciones

Se utilizó una interrupción en el pin seleccionado para el encoder. Cada vez que ocurre un flanco de subida, se llama a una función tipo callback, la cual incrementa en una unidad una va-

riable tipo contador. Posteriormente, se empleó un temporizador (timer) para calcular las revoluciones por minuto (RPM). Dado que el encoder genera únicamente 20 pulsos por revolución, se configuró el temporizador con un período de 100 ms. Esto se debe a que, si el intervalo es menor, el contador del encoder no alcanza a incrementarse adecuadamente, lo que resulta en mediciones erróneas.

Polling + interrupciones

Se utilizó una interrupción por flanco de subida para contar los pulsos. Cada 500 ms se leyó el contador y se reinició. El cálculo de RPM se realizó en el bucle principal con la misma fórmula que en la versión por polling, pero usando como fuente de datos el contador incrementado por la ISR.

0.2.2. Sistema de control en lazo abierto del motor

Polling

En esta implementación se controló el motor en lazo abierto aumentando el ciclo de trabajo PWM cada 2 segundos. Se utilizó la función get_absolute_time() para medir el tiempo transcurrido entre cada escalón. El sistema incrementó el valor del duty cycle desde 0 % hasta 100 %, y termina una vez alacanzado el 100 %. No se implementó medición de RPM, ya que el objetivo era verificar la variación de velocidad en el motor.

Interrupciones

En este código se utilizaron interrupciones para el encoder y dos temporizadores (timers). El primer temporizador, con un período de 100 ms, se empleó para el cálculo de las revoluciones por minuto (RPM). El segundo temporizador, con un período de 3 segundos, se utilizó para modificar el ciclo de trabajo (duty cycle) de la señal PWM, permitiendo así observar los cambios en la velocidad del motor.

Polling + interrupciones

Este código aplica PWM al motor en incrementos de 20 % cada 5 segundos. Se usa solo una interrupción para contar los pulsos del encoder.

0.2.3. Sistema de captura de la curva de reacción

Polling

Se implementó una rutina que mide la respuesta del motor ante variaciones escalonadas de PWM cada 2 segundos. El sistema toma muestras de RPM cada 50 ms, almacenando los valores en un buffer junto con el tiempo y el duty cycle correspondiente.

La medición de pulsos se realiza mediante polling, detectando flancos en el pin del encoder. El programa genera una curva completa de subida y bajada de PWM, y al finalizar, imprime los datos en formato CSV para su posterior análisis.

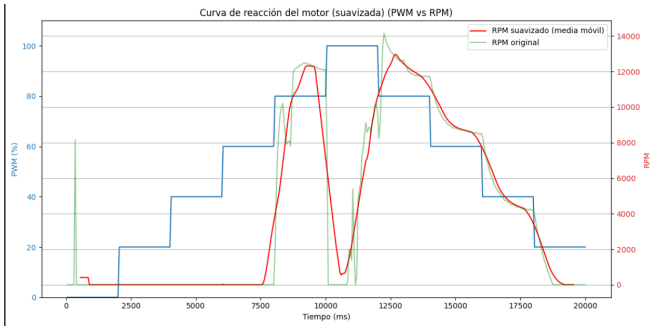

Figura **0-2**: Curva de reacción por polling de un motor sin reductor

Inicialmente, se implementó la estrategia de polling utilizando un motor DC sin reductor, con el objetivo de capturar la curva de reacción del

sistema. Sin embargo, al analizar los datos obtenidos, se observaron lecturas con caídas inesperadas en la velocidad angular en niveles altos de PWM. Estos comportamientos sugerían posibles errores en la captura de los pulsos del encoder, ya que el polling, al ser secuencial, podría no estar respondiendo adecuadamente a los eventos de alta frecuencia generados por un motor que gira a gran velocidad.

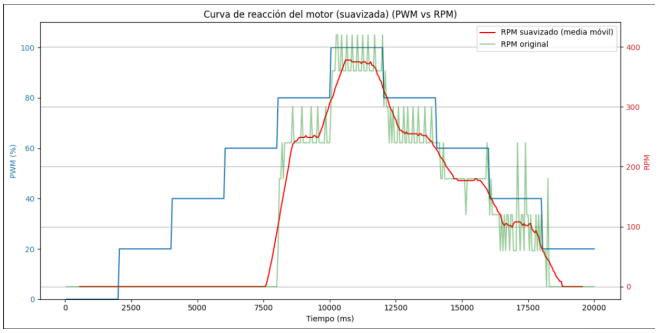

Figura **0-3**: Curva de un motorreductor por polling

Para descartar fallos en el código y verificar si el problema estaba relacionado con la velocidad del motor, se repitió la prueba utilizando un motor con reductor, el cual opera a menores RPM. Los resultados obtenidos fueron significativamente más estables, mostrando una curva de reacción suave y con forma típica de un sistema de primer orden con retardo. Aunque persistió cierto nivel de ruido, las mediciones suavizadas confirmaron que la implementación era funcional.

Interrupciones

En este código se busca variar el ciclo de trabajo (duty cycle) de la señal PWM en incrementos del 20 %, primero de forma ascendente y luego descendente. El objetivo es registrar la variación de las RPM del motor en función de los cambios en el duty cycle.

Se utilizó una interrupción para el encoder, así como dos temporizadores. El primer temporizador, con un período de 100 ms, se empleó para calcular las RPM. El segundo temporizador, configurado con un período de 3 segundos, se utilizó para modificar gradualmente el duty cycle.

Después de configurar los temporizadores e interrupciones, el código entra en un bucle while en el que permanece a la espera de que el segundo temporizador realice las modificaciones del duty cycle, permitiendo así observar y registrar los cambios en la velocidad del motor.

Adicionalmente, se creó un buffer en el cual se almacenan los datos correspondientes al tiempo, el duty cycle y las RPM. El almacenamiento en el buffer se realiza dentro del callback del primer temporizador (utilizado para el cálculo de las RPM).

Una vez finalizado el ciclo while, es decir, tras haber variado el duty cycle desde 0 % hasta 100 % y luego de regreso a 0 %, los datos almacenados en el buffer se envían a través del puerto serial. Finalmente, se desactivan los temporizadores y se restablece la referencia de velocidad a 0.

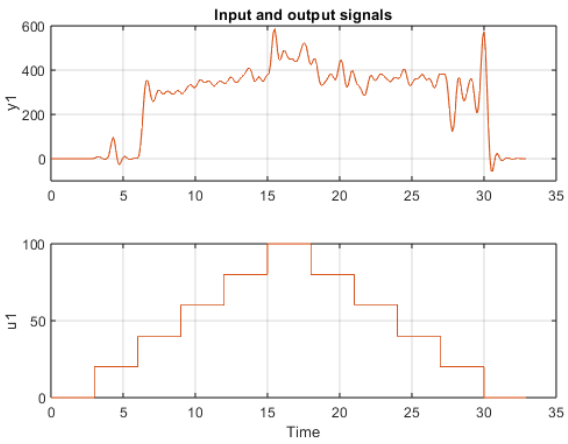

Figura **0-4**: Curva de reacción por interrupciones de un motor.

Polling + interrupciones

Este código genera una curva de reacción escalonada, aplicando PWM creciente y luego decreciente. Cada 50 ms se mide la velocidad mediante el contador de pulsos capturados por interrupción. Las muestras se almacenan en un buffer y al finalizar se imprimen en formato CSV. Esta curva es clave para observar el comportamiento dinámico del motor.

Al implementar la estrategia de polling + interrupciones, se observó una mejora considerable en la calidad de la curva de reacción del motor. La gráfica muestra una respuesta mucho más estable, con un comportamiento suave y continuo tanto en el ascenso como en el descenso del PWM. La señal de RPM suavizada sigue una dinámica coherente con un sistema de primer orden, donde se evidencian claramente los tiempos de subida y bajada, así como los tiempos de establecimiento en cada escalón de entrada. El uso de interrupciones permitió capturar con mayor precisión los pulsos del encoder, reduciendo las pérdidas de eventos que afectaban la estrategia de polling puro, especialmente en las zonas de mayor velocidad.

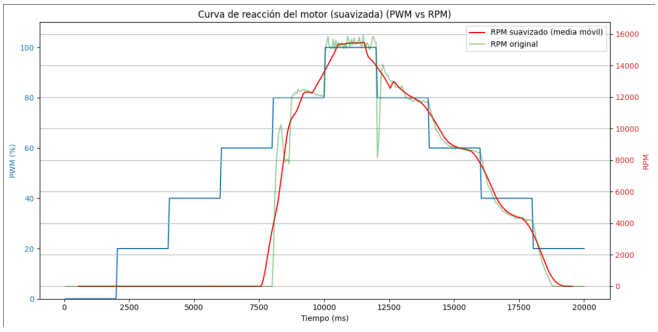

Figura **0-5**: Curva de reacción por polling + interupciones de un motor

0.2.4. Sistema de caracterización de un motor DC

Polling

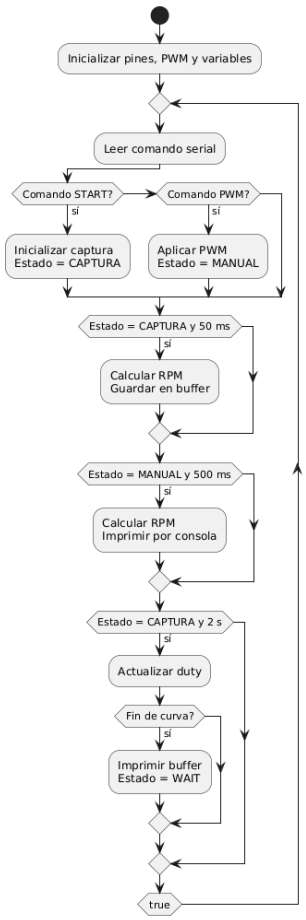

Figura **0-6**: Flujo código de caracterización en polling

Esta sección amplía los códigos anteriores, añadiendo una interfaz serial que permite controlar el sistema mediante los comandos START 'valor' y PWM 'valor'. El sistema opera completamente bajo polling, y permite seleccionar entre modo manual y automático de forma interactiva.

Interrupciones

Este código es el resultado de las secciones anteriores, con la integración de una interfaz básica para el usuario. Se añadió la opción de seleccionar entre dos modos de operación: Start (modo de prueba con variación automática del duty cycle) y PWM (modo manual).

En el modo Start, el código ejecuta una rutina de variación automática del duty cycle, registrando las RPM en cada paso. En el modo PWM, el usuario puede ingresar directamente el valor deseado del duty cycle o un valor relacionado con la velocidad a la que desea que opere el motor.

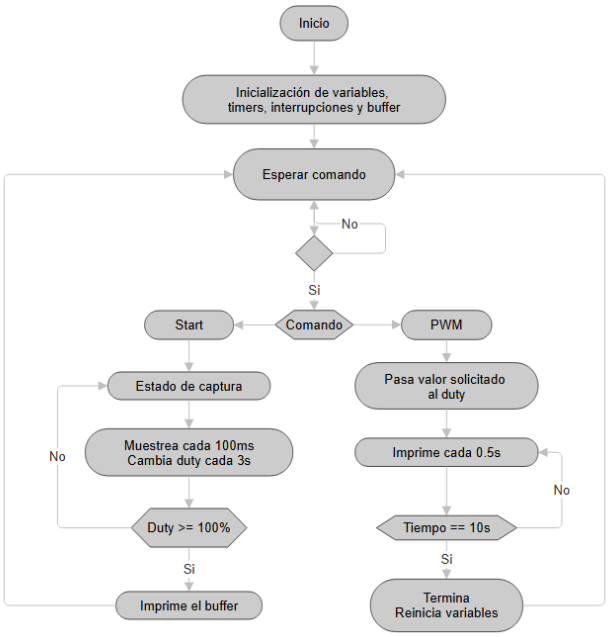

Figura **0-7**: Curva de reacción por interrupciones de un motor.

Polling + interrupciones

Este programa integra los anteriores con una interfaz por los comandos START 'valor' para iniciar una captura con escalones de PWM y PWM 'valor' para aplicar un valor fijo. La velocidad del motor se mide en RPM usando interrupciones para contar pulsos del encoder.Se almacenan los datos en un buffer que luego se imprime como tabla CSV cuando se está en modo captura.

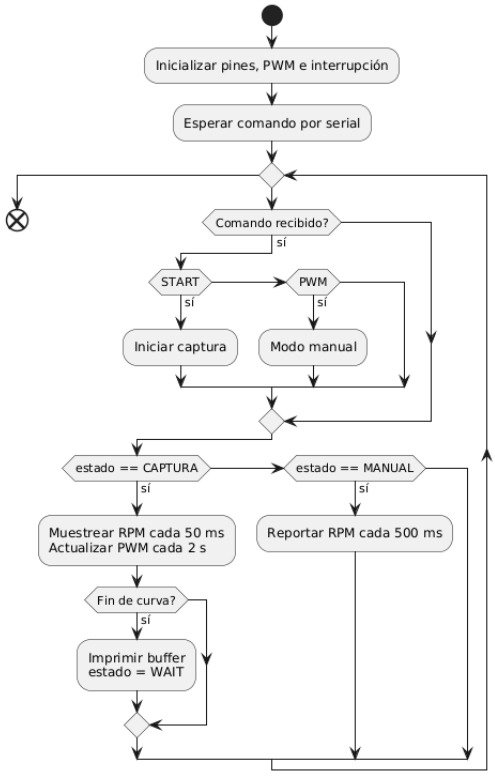

Figura **0-8**: Flujo código de caracterización en polling + interrupciones

0.3. Modelado del motor

Para el modelamiento del motor se utilizó el Toolbox de MATLAB System Identification, donde inicialmente se definieron la entrada y la salida del sistema: la señal PWM como entrada y las RPM como salida. Con este toolbox se obtuvo la función de transferencia del modelo, incluyendo su respectivo tiempo muerto.

$$G(s) = \frac{K}{\tau s + 1} e^{-t_d s} \tag{0-2}$$

El modelo resultante representa una aproxi-

mación imprecisa del comportamiento real del motor, debido principalmente a dos factores. En primer lugar, la función de transferencia identificada es de un solo polo, lo que limita su capacidad para replicar con precisión la dinámica de un sistema que podría tener dos polos o más. En segundo lugar, la naturaleza del sistema es inherentemente no lineal, por lo que al utilizar una función de transferencia, que representa un modelo lineal, se está realizando una simplificación del comportamiento real del sistema.

$$G(s) = \frac{126{,}84}{1 + 0{,}9174s} e^{-0{,}176s} \qquad (0\text{-}3)$$

0.4. Conclusiones

- El enfoque de polling consiste en verificar constantemente el estado de una variable o periférico dentro del ciclo principal del programa. Si bien es sencillo de implementar, tiene limitaciones importantes: consume recursos del procesador de forma ineficiente y no garantiza una respuesta precisa en aplicaciones donde se requiere capturar eventos en tiempo real. Es útil en sistemas simples o cuando los eventos ocurren a baja frecuencia.
- Las interrupciones permiten una respuesta inmediata ante eventos externos sin necesidad de monitoreo constante. En este proyecto, su uso fue fundamental para capturar los pulsos del encoder con precisión. Esta técnica mejora la eficiencia del sistema, ya que el microcontrolador puede realizar otras tareas mientras espera eventos, y solo reacciona cuando ocurren. Sin embargo, un mal manejo de interrupciones (por ejemplo, rutinas largas dentro del callback) puede afectar negativamente el rendimiento global.
- La combinación de polling e interrupciones resulta ser una estrategia eficiente y flexible. En este proyecto, se usaron interrupciones para eventos críticos como la captura de pulsos del encoder y polling para tareas menos sensibles al tiempo como el calculo de RPM y la actualización del PWM. Este enfoque balanceado permite mantener la precisión en la captura de eventos mientras se conserva la simplicidad y control del flujo del programa en otras partes del sistema.