Informe de Laboratorio #5

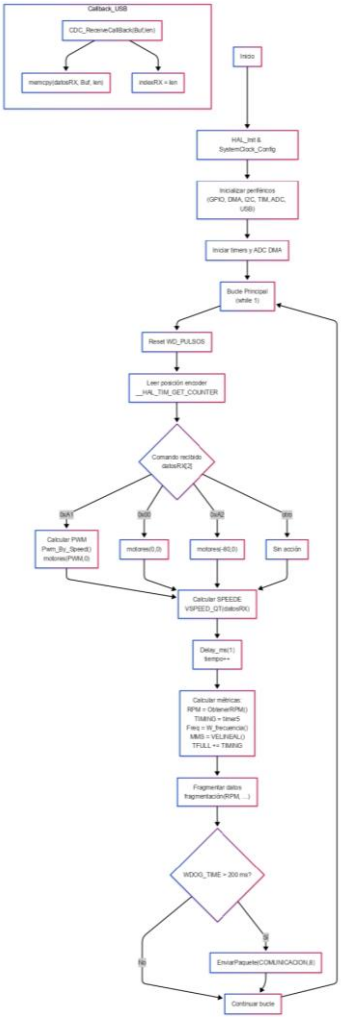
Universidad Sergio Arboleda

Estudiantes: Cristian Andrés Mora Cortes, Daniel Fernando Avila, Yeison Estiben Lara

Protocolo de Comunicación

Para mostrar cómo viajan los datos, el proceso es éste: primero la STM32 toma las RPM y el tiempo, calcula el checksum (haciendo XOR de los bytes de RPM, tiempo y 0x0D) y arma un paquete de 7 bytes [0x0F, tamaño, RPM_alto, RPM_bajo, tiempo, checksum, 0x09]. Luego envía ese paquete por USB-CDC a la app de Qt, que busca el byte de inicio 0x0F, lee cada campo según el tamaño, comprueba que el checksum sea correcto y verifica el byte de fin 0x09 antes de extraer y mostrar los datos en pantalla.

Diagrama de Flujo



Materiales

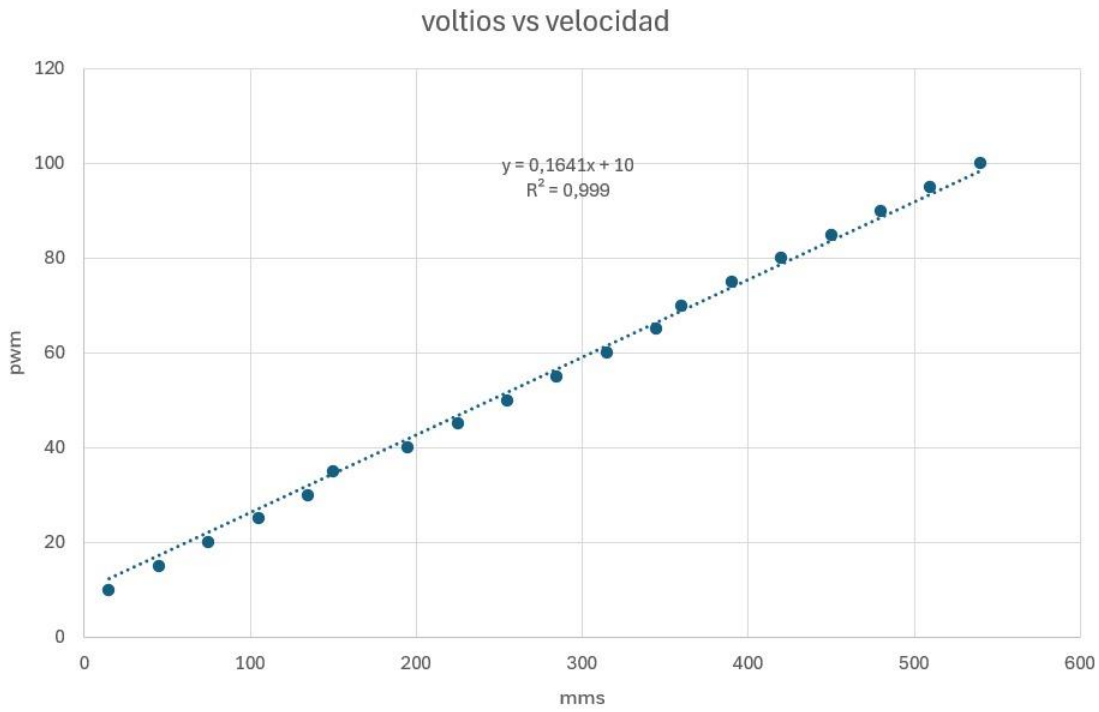
- **Microcontrolador STM32F411CEU (programado vía ST-LINK V2)**
- **Puente H TB6612FNG (manejo de potencia del motor)**
- **Jumpers**
- **Regulador de voltaje MP1584 (alimentación estable)**
- **Estructura del motor.**
- **Motor DC con encoder de cuadratura acoplado a una rueda con 10 canicas (simulando inercia)**
- **Cables y protoboard**

Funcionamiento

La práctica consistió en montar el motor con su encoder, calibrado a 1440 pulsos por vuelta, y aplicar un filtro digital FIR en la STM32 para suavizar la señal de pulsos antes de calcular variables de control. Cada 200 ms el firmware mide Δpulsos en un intervalo de 10 ms para obtener RPM, convierte esas RPM a frecuencia angular (rad/s) y luego a velocidad lineal (mm/s) mediante relaciones matemáticas integradas en funciones C. A partir de la velocidad deseada, la función Pwm_By_Speed usa la ecuación experimental PWM% = 0.1641·|RPM|+7.7 (limitada a ±100 %) para generar el ciclo de trabajo adecuado y controlar la dirección según el signo.

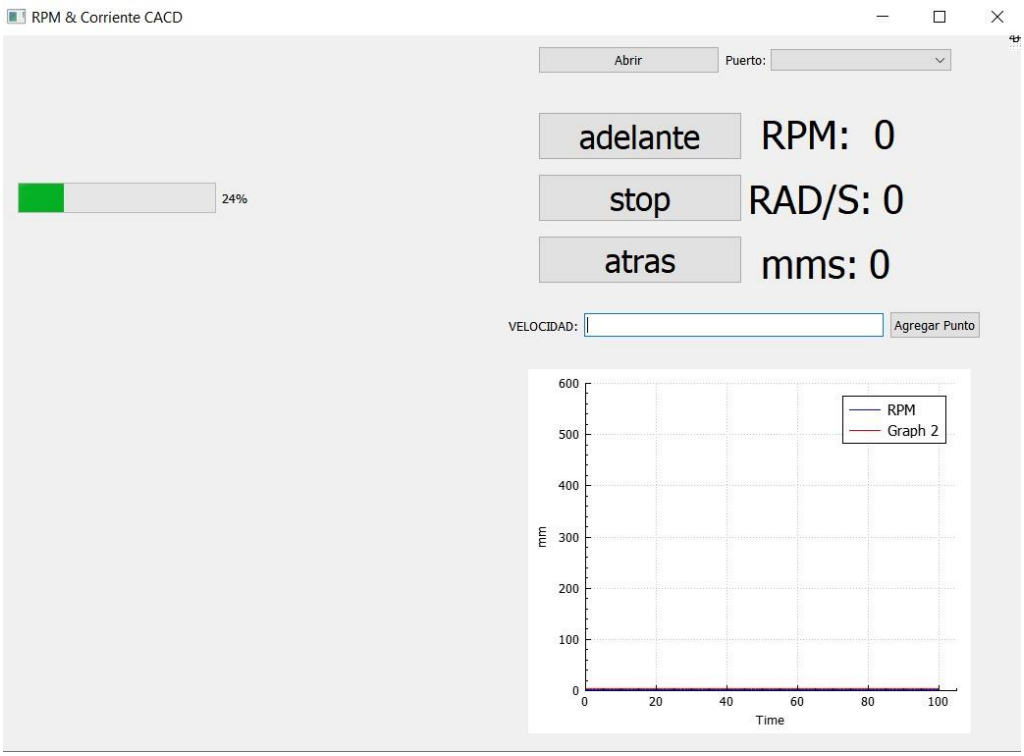
Desarrollo y Análisis en Excel

Los datos de RPM y mm/s se volcaron a un archivo CSV para su análisis en Excel. Allí se limpió la señal de atípicos, se trazaron curvas de RPM vs tiempo y mm/s vs tiempo, y se generaron gráficos de mm/s frente a RPM para valores de 0 a 100 RPM en pasos de 10. Mediante ajuste lineal se obtuvo la pendiente y la ordenada al origen, y se derivó la función inversa que luego se implementó en el firmware para convertir directamente mm/s deseados en valores de PWM.



Integración con Qt

En Qt Creator se configuró QSerialPort (115 200 baud, 8N1) en modo lectura/escritura. Un slot conectado a botones “adelante”, “stop” y “atrás” envía un byte de comando al STM32, y un lector de trama valida inicio, CRC y fin para extraer RPM, tiempo, rad/s y mm/s. Estos valores se muestran en labels y en tiempo real en un QCustomPlot de dos curvas deslizantes. La aplicación ofrece además guardar los datos en un archivo de texto para análisis posterior.



Funciones en el Microcontrolador

El firmware agrupa la lógica en funciones modulares: calcularCRC (XOR acumulativo),

serializarPaquete (arma la trama de 7 bytes), ObtenerRPM (cuenta pulsos en ventana de 10 ms), W_frecuencia y VELINEAL (convierten RPM a rad/s y mm/s), y Pwm_By_Speed (genera PWM según la recta experimental y el signo). Estas rutinas cierran el lazo de control embebido entre la lectura del encoder y el accionamiento del motor.

Registro y Almacenamiento

La interfaz Qt incluye la opción “Guardar” que vuelca tiempo y mm/s a un archivo de texto plano. Esto permitió un análisis detallado en Excel y la generación de reportes de desempeño, estabilidad y linealidad del sistema.

Problemas y Soluciones

Durante el proyecto se identificaron variables locales inaccesibles en la capa gráfica, resueltas declarándolas globales en widget.h; la complejidad inicial de QCustomPlot se superó simplificando la serie de datos; y el manejo de signo para inversión de sentido requirió ajustar la función Pwm_By_Speed() para asegurar que valores negativos de velocidad invirtieran correctamente la dirección.

Conclusiones

El uso de un filtro FIR de bajo orden junto con captura por timer/interrupción en el STM32 eliminó eficientemente el ruido del encoder sin sobrecargar el micro. La conversión matemática directa de pulsos a RPM y mm/s y la implementación de la función inversa de la curva PWM vs velocidad garantizaron un control preciso. La integración con Qt vía USB-CDC ofreció visualización y registro en tiempo real, cerrando un sistema embebido robusto, preciso y fácilmente extensible.