Informe laboratorio Entrega Final

Daniel Avila

Yeison Lara

Cristian Mora

¿Que se usó?

- Robot móvil ensamblado con sus motores y sensores
- Cable delgado para conexión UART
- Programador STLink
- Teleplot o Qt para visualización
- **FreeRTOS** para multitarea (CMSIS-OS).

¿Que se realizó?

Al encender el sistema, en **main** se inicializaron todos los periféricos esenciales: PWM de motores (TIM4), contadores de encoders (TIM2, TIM3), temporizadores de muestreo (TIM9, TIM11), ADC en modo DMA para sensores IR, y UART1 a 115200 bps. Después de resetear la odometría y configurar los filtros EMA, se inicializaron por separado los controladores **PID lineal** y **angular** (`PID_Init_Linear(0.03f,0.01f)` y `PID_Init_Angular(...)`).

Posteriormente se creó y arrancó el **scheduler de FreeRTOS** con tres tareas:

1. **TAREA_ODOMETRIA**
   Cada 5 ms lee pulsos con `leer_pulsos()`, corrige sobreflow y actualiza la pose con `actualizar_odometria()`. Luego mide la reflectancia de cada sensor IR aplicando un filtro exponencial (EMA), calcula la diferencia LED-apagado/LED-encendido, la mapea logarítmicamente a una distancia (`mapLog` + `CONV_Y_TRANS`) y almacena en el array `DISTANCIAS[4]`.
2. **TAREA_MOVIMIENTO**
   Ejecuta la máquina de estados `STATE_MACHINE_RUN()`, que cicla entre los estados:
   - **ESPERA** (pequeño delay) → **LEER** (detección de obstáculos izquierda/derecha) →
   - **FRENTE**, donde corre un perfil trapezoidal perpetuo (`trapecio_perpetuo`) usando la distancia frontal `DISTANCIAS[0]` como tope, calcula en cada iteración el error de velocidad y aplica PID lineal (`calc_vel_pid`) para generar las señales de control de cada motor, y envía esas señales con `motores()`.

    - **GIRO_DERECHA** / **GIRO_IZQUIERDA**, donde emplea `trapecio_Angular` para girar ±2.5 rad, usando PID angular si fuera necesario, y vuelve a ESPERA al finalizar.
3. **TAREA_LED_PERM**
   Parpadea un LED cada 50 ms como indicador de vida.

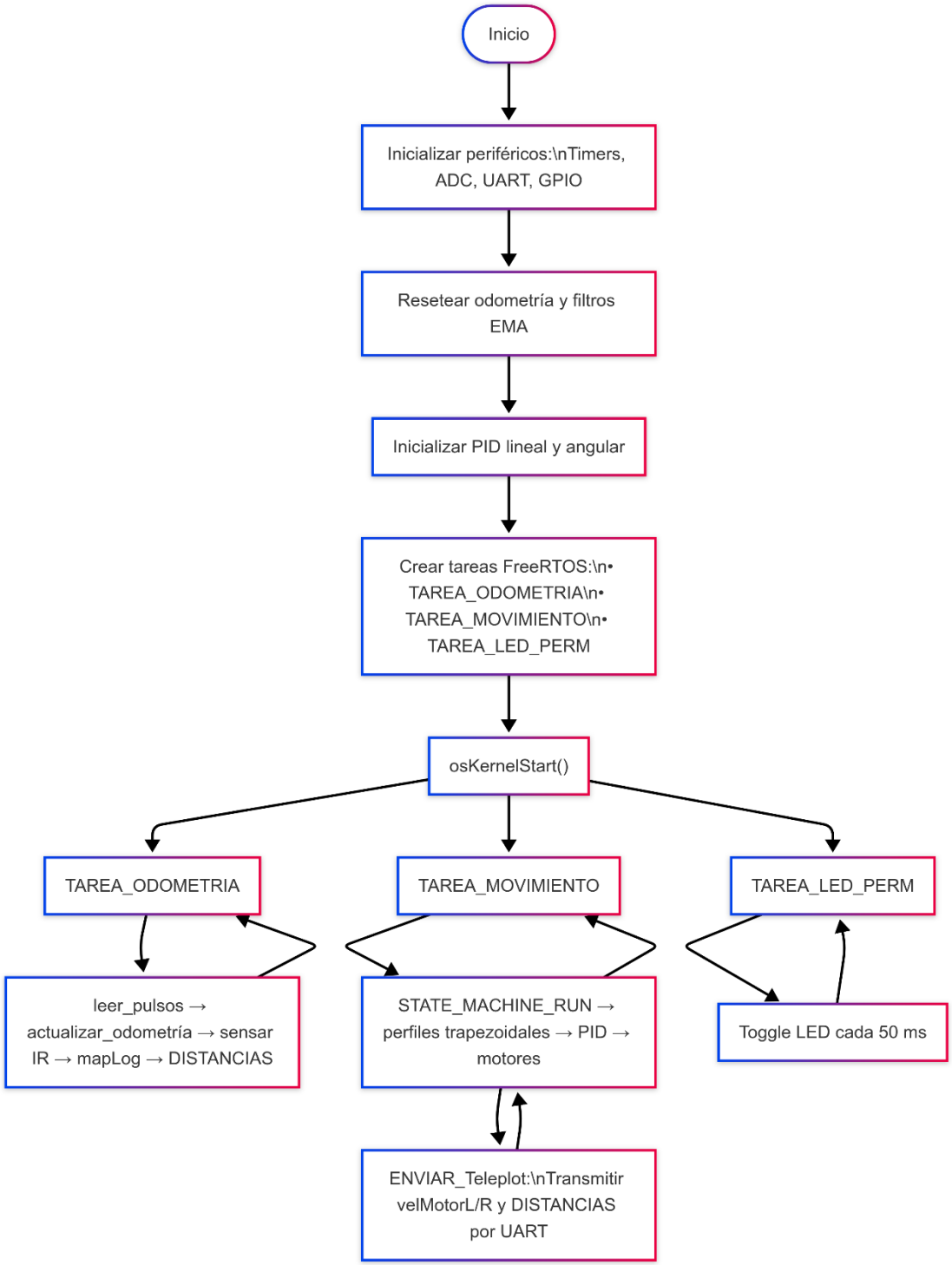
En paralelo, desde dentro de la tarea de movimiento se llama a `ENVIAR_Teleplot()` enviando por UART1 las velocidades de motor y las cuatro distancias IR en formato texto, para que **Teleplot** muestre gráficas en tiempo real de:

- Velocidad deseada vs. velocidad real (izquierda/derecha).
- Distancias medidas por los cuatro sensores IR.

¿Qué sucedió?

Cuando iniciamos, el robot arrancó sin problema y en cuanto la tarea de movimiento se activó, avanzó siguiendo el perfil trapezoidal que habíamos configurado. Cada 5 ms la tarea de odometría actualizaba la posición y calculaba distancias con los sensores IR; cuando detectó un obstáculo a la izquierda o derecha, la máquina de estados cambió al estado de giro correspondiente, y el robot giró de forma suave, sin brusquedades, gracias al PID angular. Luego regresó al avance frontal automáticamente. Mientras tanto, Teleplot iba graficando en tiempo real cómo la velocidad real de cada motor seguía casi exactamente la referencia deseada y cómo las distancias IR se actualizaban al momento. Todo el ciclo de avance–detección–giro–espera se repitió sin errores ni cuelgues, y al completar cada maniobra el sistema volvió a modo espera sin necesidad de reinicios manuales.

Diagrama de flujo:



Conclusión:

En este laboratorio se validó la viabilidad de un enfoque **multitarea** basado en RTOS para control de robots móviles con sensores IR y encoders. El uso de perfiles trapezoidales perpetuos, combinado con un **control PID modular** (PID lineal y angular independientes), permitió lograr movimientos suaves y precisos, tanto en avance como en giros. La lógica de detección de obstáculos integró sensorización, mapeo logarítmico y una máquina de estados clara y escalable. Por último, la visualización con Teleplot aportó una capa de análisis en tiempo real que facilitó la calibración de parámetros y la depuración. El diseño resultante es **modular**, **reutilizable** y **robusto**, apto para extensiones futuras (nuevas tareas, algoritmos de rastreo de línea, Flood Fill, etc.) sin comprometer el rendimiento.