Desafío Técnico Eolian Auto Solar

Sistema de Adquisición de Datos (Flutter)

Contexto del Problema

En el auto solar, el bus de comunicación CAN es el sistema nervioso que conecta todos los componentes electrónicos. El computador central necesita un software robusto y eficiente para **leer** los mensajes de este bus (ej. voltaje de la batería, velocidad de las ruedas), **mostrarlos** en tiempo real para diagnóstico y **guardarlos** para un análisis post-carrera.

Tu misión es diseñar la arquitectura de este sistema y crear un prototipo funcional de una de sus partes clave.

1. Requisitos del Sistema a Diseñar

1. **Lector CAN (CAN Reader):** Un componente responsable de interactuar con la red CAN y decodificar los mensajes entrantes.
2. **Motor de Datos (Data Engine):** Lógica que procesa los datos decodificados. Debe mantener el estado actual de las variables (ej. la última temperatura leída).
3. **Interfaz de Usuario (UI):** Un componente que muestra los datos actuales en la consola de manera clara y legible.
4. **Registrador (Logger):** Un módulo que guarda un historial de todos los datos recibidos.

2. Entregables

2.1. Diagrama de Arquitectura (UML)

- Debes crear un **Diagrama de Componentes** que muestre los módulos principales del sistema, sus responsabilidades y cómo se interconectan.
- Opcionalmente, puedes incluir un **Diagrama de Secuencia** que ilustre el flujo de un mensaje CAN a través de tu sistema.
- **Importante:** El objetivo del UML es mostrar tu capacidad de diseño arquitectónico, no detallar cada clase o función.

2.2. Prototipo en Flutter

- Debes desarrollar una aplicación simple en Flutter que reciba datos **simulados** del bus CAN y los muestre en la pantalla.
- **Simulación del Bus CAN:** No necesitas hardware real. Debes crear una clase o función en Dart que simule la llegada de tramas CAN, preferiblemente usando un 'Stream' que emita nuevos datos periódicamente(puedes emitir periódicamente el mismo mensaje).

```
class CANFrame {
  final int id;
  final List<int> data;

  CANFrame(this.id, this.data);
}

// Funcion que simula el stream de datos
Stream<CANFrame> getCANDatastream() async* {
  // Logica para emitir nuevos frames periodicamente
}
```

- La aplicación debe escuchar este 'Stream' (ej. con un 'StreamBuilder'), procesar los datos y mostrar los valores en la UI (ej. en una 'ListView').

Plantilla de Avance Semanal

Semana #: [1, 2 o 3]	**Fecha:** [dd/mm/aaaa]
1. Actividades Realizadas:	*Ej: Investigué sobre patrones de diseño para sistemas embebidos. Comencé el diseño del Diagrama de Componentes. Creé la estructura base del proyecto en C++ con CMake.*
2. Descubrimientos y Aprendizajes Clave:	*Ej: Aprendí que usar un patrón de observador podría ser útil para notificar a la UI y al Logger sin acoplarlos fuertemente. Descubrí la librería...*
3. Bloqueos o Dificultades Encontradas:	*Ej: Tuve dudas sobre cómo estructurar la simulación del bus CAN para que fuera realista. Me costó decidir el mejor formato para el archivo de log.*
4. Plan para la Próxima Semana:	*Ej: Finalizar el diagrama UML. Implementar la función de simulación CAN y la lógica de decodificación. Comenzar con la impresión en consola.*