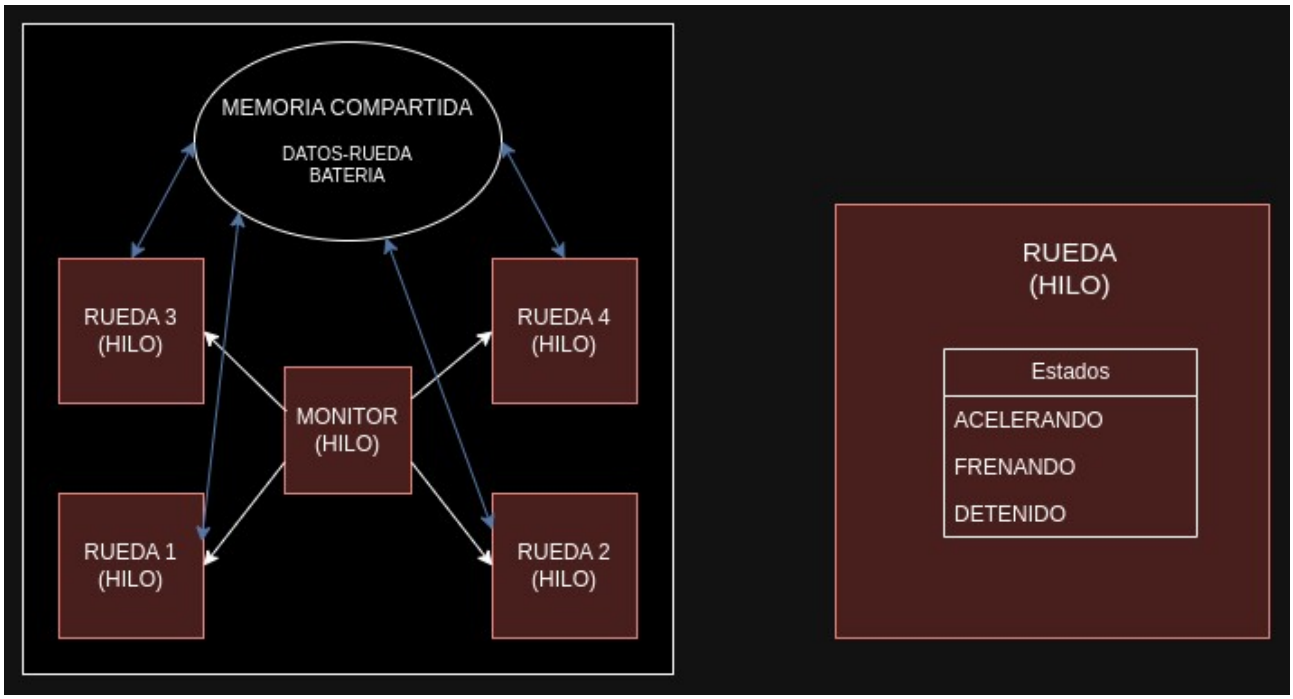
Proyecto final – sistemas operativos
Nombre: Adair Abrigo

Diagrama de componentes



Explicación del codigo

Aquí tienes un párrafo explicativo de aproximadamente 200 palabras sobre lo que hace el código:
Este código simula el funcionamiento de un vehículo eléctrico con cuatro ruedas. Utiliza programación multihilo y memoria compartida para modelar el comportamiento del vehículo en diferentes estados de conducción. Cada rueda es representada por un hilo independiente que puede acelerar, frenar y regenerar energía. La memoria compartida almacena el estado global del vehículo, incluyendo la carga total de la batería y el estado de cada rueda. El programa permite al usuario controlar el vehículo mediante comandos simples como acelerar, frenar o detener. Durante la aceleración, el vehículo consume energía de la batería, mientras que al frenar, regenera energía. El código también simula escenarios de daño, permitiendo que las ruedas se "revienten", lo que afecta el rendimiento general del vehículo. Se implementan mecanismos de sincronización como mutexes y variables de condición para gestionar el acceso a recursos compartidos, evitando condiciones de carrera. El programa monitorea constantemente el estado de conducción y actualiza la simulación en tiempo real, proporcionando una representación dinámica e interactiva de cómo funcionaría un vehículo eléctrico en diferentes condiciones de manejo. Cada hilo fue programado para sólo acceder al bloque de memoria compartido que le pretenece logrando así asincronia y seguridad casi completa
Explicación de las funciones principales

main(): Inicializa la memoria compartida y los hilos. Maneja la entrada del usuario para controlar el estado de conducción. Es el punto de entrada del programa y coordina todas las operaciones.
be_wheel(): Simula el comportamiento de cada rueda. Controla los ciclos de aceleración, frenado y carga según el estado del vehículo. Es ejecutada por cada hilo de rueda.

monitoring_state_drive(): Escucha constantemente las instrucciones del conductor. Actualiza el estado global de conducción (acelerar, frenar, parar) basado en la entrada del usuario.
acelerating(): Incrementa la velocidad del vehículo y reduce la carga de la batería. Simula el consumo de energía durante la aceleración.
charge_motor(): Regenera energía durante el frenado. Incrementa la carga de la batería y reduce la velocidad del vehículo.
damage_wheel(): Simula el daño a una rueda. Reduce el número de ruedas funcionales y ajusta el comportamiento del vehículo en consecuencia.
take_lines() y free_lines(): Gestionan el acceso a las líneas de carga. Evitan conflictos cuando múltiples ruedas intentan cargar simultáneamente.
Los semaforos que se utilizaron
shared_memory_mutex: Protege el acceso a la memoria compartida.
lines_mutex: Controla el acceso a las líneas de carga.
current_speed_mutex: Protege la variable global de velocidad actual.
Semáforos utilizados:

1. shared_memory_mutex: Protege el acceso a la memoria compartida.
2. lines_mutex: Controla el acceso a las líneas de carga.
3. current_speed_mutex: Protege la variable global de velocidad actual.

Estos semáforos se usan en las siguientes funciones:

- acelerating(): Usa current_speed_mutex y shared_memory_mutex.
- take_lines() y free_lines(): Utilizan lines_mutex.
- charge_motor(): Emplea current_speed_mutex.
- print_charge() y print_charge_after_break(): Usan shared_memory_mutex.