# 🚍 Vehicle Simulator Telemetry Pipeline

This simulator models vehicles with physics-accurate movement, navigates them along real routes, and transmits live telemetry packets to a backend server.

## 🔄 End-to-End Flow

1. **EngineBlock (engine\_block.py)**
   * Loads a speed model (e.g., kinematic\_speed) via sim\_speed\_model.
   * Produces cumulative physics data: velocity, distance, acceleration, time.
   * Writes diagnostics into the **EngineBuffer**.
2. **Navigator (navigator.py)**
   * Reads cumulative distance from EngineBuffer.
   * Maps vehicle position along a polyline route (GeoJSON).
   * Produces raw telemetry entries: lat, lon, bearing, speed, distance, time.
   * Writes results into the **TelemetryBuffer**.
3. **FleetDispatcher (fleet\_dispatcher.py)**
   * Runs in its own thread.
   * Drains TelemetryBuffer entries and wraps them into strict **TelemetryPacket** objects (schema defined in packet.py).
   * Ensures outbound data conforms to fleet/server standards.
   * Pushes packets into the **RxTxBuffer** for transmission.
4. **GPSDevice (device.py)**
   * Owns the WebSocket transmitter.
   * Continuously drains packets from RxTxBuffer.
   * Encodes them with PacketCodec and sends them to the server.
   * Represents the “radio” link from vehicle → depot/server.
5. **VehiclesDepot (vehicles\_depot.py)**
   * High-level orchestrator.
   * Lifecycle:
     + [INFO] FleetDispatcher ONSITE
     + [INFO] Depot OPERATIONAL...
     + Navigator boards, GPSDevice turns ON, Engine starts.
     + At shutdown: Engine stops, GPSDevice OFF, Navigator disembarks, FleetDispatcher OFFSITE, Depot UNOPERATIONAL.

## ✅ Verification

* **Engine Data** validated with physics checks (distance increment matches velocity × tick).
* **Telemetry Data** verified along real route geometry with consistent lat/lon increments and headings.
* **Server Logs** confirm receipt of schema-compliant packets, aligned with physics and route progression.

## Example Server Log

{  
 "deviceId": "ZR1001",  
 "route": "1",  
 "vehicleReg": "ZR1001",  
 "driverId": "sim-ZR1001",  
 "driverName": { "first": "Sim", "last": "ZR1001" },  
 "timestamp": "2025-09-01T21:53:40.175803+00:00",  
 "lat": 13.326506,  
 "lon": -59.615599,  
 "speed": 24.0,  
 "heading": 216.8  
}

This gives anyone cloning the repo a **clear mental model**: physics → navigation → dispatch → radio → server.