**EnviroPulse V1.1 Architecture Overview**

This document describes the finalized architecture of EnviroPulse V1.1 for both gateway and node systems. The layout is modular, event-driven, and designed for long-term scalability and educational clarity.

**📡 Gateway Architecture (V1.1)**

**🔁 Message Flow**

1. **RAK2287 Listener**
   * gateway\_listener.py: Listens for incoming LoRa packets.
   * gateway\_receive\_over\_lora.py: Receives and formats raw byte streams.
2. **Decoding Pipeline**
   * gateway\_decoder.py: Parses message bytes using protocol.py.
   * gateway\_sanity\_check.py: Validates structure and content.
3. **Dispatch and Routing**
   * gateway\_dispatch.py: Main router for all decoded events.
   * gateway\_ack\_manager.py: Handles ACK/Retry/Fail logic with gateway\_unique\_id\_manager.py.
4. **Message Builders**
   * gateway\_inbound\_message\_builder.py: Constructs incoming messages for dispatch.
   * gateway\_lora\_message\_builder.py: Builds outbound LoRa messages.
   * gateway\_server\_inbound\_message\_builder.py: Routes inbound server-side commands (e.g. change\_mode).
5. **Transmission**
   * gateway\_send\_over\_lora.py: Sends messages via RAK2287.
   * gateway\_transmitter.py: Master transmitter module.
6. **Registry Logic**
   * gateway\_registries\_manager.py: Loads and holds registry states.
   * gateway\_registries\_update\_executor.py: Applies registry updates.
   * gateway\_registries\_accessor.py: Provides lookup tools.
   * gateway\_unique\_id\_manager.py: Tracks message UIDs.
7. **System Utilities**
   * time\_sync\_manager.py: Handles time sync events.
   * timekeeper.py: Provides internal time tracking.
   * gateway\_logger.py: Logs all critical events.
8. **Server Communication**
   * push\_to\_server.py: Forwards validated packets to server.
   * server\_alert.py: Receives server-triggered commands.

**🎯 Node Architecture (V1.1)**

**🧠 Event Flow**

1. **Sensor Data Chain**
   * node\_sht31\_driver.py, node\_bmp390\_driver.py, node\_gps\_driver.py: Collects raw sensor data.
   * node\_hardware\_manager.py: Synchronizes all hardware modules.
   * node\_weather\_sampler.py, node\_telemetry.py: Prepares data for encoding.
2. **BirdNET Integration**
   * birdnet\_soft.py: Processes audio.
   * node\_confidence\_manager.py: Classifies detection confidence.
3. **Constructors and Builders**
   * node\_sensor\_message\_builder.py: Packages sampled data.
   * node\_constructor.py: Combines base and sensor events.
   * node\_inductor\_message\_builder.py: Forms full outbound messages.
4. **Outbound Transmission**
   * node\_release\_over\_lora.py: Handles outbound LoRa handoff.
   * send\_over\_lora.py: Sends message via RAK3272.
5. **Inbound Reception**
   * node\_message\_decoder.py: Decodes inbound LoRa.
   * node\_sanity\_check.py: Confirms integrity.
   * node\_ask\_manager.py: Handles retry and UID confirmation.
6. **Initialization**
   * initial\_launch.py: Handles startup handshake.
   * node\_time.py, node\_time\_mode\_manager.py: Maintains time state.
   * node\_unique\_id\_manager.py: Tracks local UIDs.
   * node\_registries\_manager.py: Handles local copies of configs.
7. **Logging**
   * node\_logger.py: Writes events to onboard logs.

**🧰 Shared Infrastructure**

* protocol.py: Defines the binary format for all event types.
* MAPS/: Contains all translation maps (event type, confidence scale, etc).
* PROTOCOL/: Houses event\_sentence\_structure.json.

**🗂 Message Types (Selected)**

* Avis\_Lite, Avis\_TDOA, Telemetry, Weather
* Startup, ACK, Fail, Change\_Mode, Recalibrate, Initial\_Launch

**🧭 Design Goals**

* Modular and readable by undergraduate developers.
* All components loosely coupled and protocol-driven.
* Robust UID-based retry logic and end-to-end traceability.
* Supports low-power LoRa node design with centralized gateway intelligence.

*This README reflects the official V1.1 diagram for gateway and node components. Use this as the reference point for all further development until V1.2.*