Environmental Monitoring with Meshtastic

Use Case Overview

LoRa mesh networking enables remote environmental monitoring in locations without cellular coverage or WiFi infrastructure. A solar-powered Meshtastic node with an attached sensor can transmit its natively-supported telemetry - temperature, humidity, barometric pressure, gas resistance (air-quality estimate), plus voltage and current - across the mesh to a gateway, which forwards it to a database or home automation system. Other sensor types (for example soil moisture) are **not** part of Meshtastic's native environment telemetry and require a separate microcontroller with a custom port number to publish onto the mesh.

Common applications include:

- Weather station on a remote property or off-grid cabin
- Soil moisture monitoring for irrigation management across a farm (requires a custom integration - see note above)
- Air quality monitoring at a remote site or community location
- Freezer/cold storage temperature monitoring with alerts

Supported Sensor Modules

Meshtastic's Telemetry module supports a range of **I2C digital** sensor types natively (supported sensors on the I2C bus are auto-detected at startup):

- **BME280 / BME680** - temperature, relative humidity, barometric pressure; BME680 also provides a gas resistance reading useful for air quality estimates
- **INA219 / INA260** - DC voltage and current monitoring; useful for monitoring solar panel output, battery charge state, or load current

Note: **MQ-series analog gas sensors** (MQ-2, MQ-135, etc.) are **not** natively supported by Meshtastic's Telemetry module, which reads I2C digital sensors only - there is no native ADC analog-sensor telemetry path. Using an MQ-series sensor requires a separate MCU/ADC and a

custom integration to publish the reading onto the mesh.

Hardware Setup: BME280 Wiring

The BME280 is a commonly used environmental sensor with Meshtastic. It connects via I2C. **I2C pins vary by board** - look up your exact board's default I2C (Wire) pins before wiring. As a starting point, on the classic ESP32 T-Beam the defaults are GPIO21 (SDA) / GPIO22 (SCL); the Heltec V3 (ESP32-S3) and nRF52 boards (e.g. RAK4631) use different I2C pins, so verify your specific board's pinout.

- **VCC** → 3.3V
- **GND** → GND
- **SDA** → board default SDA pin (GPIO21 on classic ESP32 T-Beam - verify your board)
- **SCL** → board default SCL pin (GPIO22 on classic ESP32 T-Beam - verify your board)

Caution - 3.3 V only: power the BME280 from the board's **3.3 V** rail, NOT 5 V. Most BME280/BME680 breakouts and the ESP32/nRF52 I2C pins are 3.3 V devices; applying 5 V, or feeding 5 V logic levels onto the 3.3 V SDA/SCL lines, can permanently damage the sensor or the MCU. If you use a 5 V-only sensor module, add a logic-level shifter on the I2C lines. (BME280 default I2C address is 0x76, or 0x77 if SDO is tied to VCC.)

After wiring, enable the sensor in the Meshtastic app or CLI: navigate to **Telemetry → Environment** and enable the module. The node will begin broadcasting environment telemetry on the mesh.

Reading Sensor Data

Telemetry data is accessible through multiple paths:

- **Meshtastic app (phone):** telemetry appears in the node info panel when you tap on a node - shows last-received temperature, humidity, pressure, and other available metrics
- **MQTT gateway:** a Meshtastic node with internet access forwards telemetry packets to an MQTT broker. **JSON output is only available on ESP32 gateways** - JSON MQTT is **not supported on the nRF52 platform** (RAK4631, etc.). Since this book's build pages center on the RAK4631 (nRF52), a RAK4631 cannot itself emit JSON to MQTT; use a separate ESP32 gateway, or parse the protobuf `ServiceEnvelope` instead. This enables integration with databases and dashboards.

Sample MQTT Telemetry JSON Structure

When a telemetry packet is forwarded via MQTT as JSON (ESP32 gateway only), the fields are **flat `snake_case` under a `payload` object**, with the envelope fields at the top level of the message. (Note: this is the JSON-MQTT format - it is *not* the protobuf `decoded.telemetry.environmentMetrics` camelCase structure; parsers built against that path will get nothing.)

- **from** (top-level, `value_json.from`) - the unique decimal node ID of the sending device
- **to** (top-level, `value_json.to`) - the destination node ID; a broadcast is `-1` (decimal of 0xFFFFFFFF)
- Other top-level envelope fields include **id**, **channel**, **type**, **sender**, and **timestamp**
- **payload** - the data object; for an environment telemetry packet it contains the flat snake_case keys:
  - **payload.temperature** - degrees Celsius (float)
  - **payload.relative_humidity** - percentage (float, 0 - 100)
  - **payload.barometric_pressure** - hPa (float)
  - **payload.gas_resistance** - BME680 only, correlates with VOC concentration (the firmware emits ohms; note the Home Assistant example labels this in MOhms, so reconcile units in your parser)
  - **payload.voltage** / **payload.battery_level** - when device-metrics telemetry is present

This JSON structure can be consumed directly by InfluxDB (via Telegraf or a Node-RED flow), Home Assistant MQTT sensors (key off `value_json.payload.*`), or any custom application that subscribes to the MQTT topic. Subscribe with a JSON-specific topic filter such as `msh/+/2/json/#` rather than a bare `msh/#` (which also catches binary `2/e/` topics the JSON parser cannot read).

Integration Targets

- **InfluxDB + Grafana:** store time-series telemetry and visualize on dashboards; well-suited for long-term environmental data logging
- **Home Assistant:** configure MQTT sensors using the official `value_json.payload.*` templates (e.g. `value_json.payload.temperature`, `value_json.payload.relative_humidity`); trigger automations on temperature or humidity thresholds
- **Node-RED:** flexible pipeline for transforming, filtering, and routing telemetry data to multiple destinations simultaneously

Deployment Considerations

- **Enclosure:** use a weatherproof IP65+ enclosure for outdoor deployments; mount the sensor in a vented radiation shield for accurate temperature readings - direct sunlight on the enclosure will give artificially high temperature readings
- **Solar power:** a small 1 - 5W solar panel with a LiPo battery can power most low-duty-cycle LoRa sensor nodes, and the low transmit duty cycle makes solar viable even in partial sun. Note that the common TP4056 module is a bare single-cell linear charger (not MPPT) with **no low-temperature charge cutoff** - it will happily charge a LiPo below 0 °C (32 °F), which damages the cell and risks a fire. For cold-climate outdoor use, add an NTC-based low-temp cutoff (or use a charger that supports it), and add an in-line fuse on the battery positive lead.
- **Sensor venting:** drill small holes or use a membrane vent in the enclosure so humidity and temperature readings reflect ambient conditions, not the trapped air inside the box

Power Impact of Environmental Sensors

The BME280 draws only a few microamps when sampling at low rates (≈3.6 µA at 1 Hz for humidity, pressure, and temperature) and well under 1 µA (≈0.1 µA) in sleep mode - negligible relative to the LoRa radio and microcontroller. At a 10-minute telemetry interval on a T-Beam or similar device, sensor power consumption has minimal impact on overall battery life. The dominant power draw remains the ESP32 or nRF52840 idle current and LoRa transmit bursts.

Revision #5
Created 2026-05-03 04:10:47 UTC by Mesh America Admin
Updated 2026-06-10 03:14:04 UTC by Mesh America Admin