IoT Integration Guides

Step-by-step guides for integrating LoRa mesh with sensors, home automation, and asset tracking systems.

- Environmental Monitoring with Meshtastic
- Home Assistant and Node-RED Integration
- Remote Asset Tracking

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.

Home Assistant and Node-RED Integration

Integration Architecture

The standard integration path from a Meshtastic mesh to Home Assistant or Node-RED is:

1. **Meshtastic MQTT gateway** - a mesh node with WiFi (or a dedicated gateway device) publishes all received mesh packets to an MQTT broker as JSON
2. **MQTT broker (Mosquitto)** - runs on the local network (often on the same machine as Home Assistant); receives all packets from the gateway
3. **Home Assistant or Node-RED** - subscribes to the MQTT topics and processes the incoming data

Home Assistant Setup via HACS

The Meshtastic integration for Home Assistant is community-maintained (not an official HA core integration) and is available via HACS by adding `github.com/meshtastic/home-assistant` as a custom repository (Integration category), then installing it:

1. Install HACS in Home Assistant if not already present
2. In HACS, add the Meshtastic integration's GitHub URL (`github.com/meshtastic/home-assistant`) as a custom repository: HACS > Integrations > three-dot menu > Custom repositories, category "Integration". (It is not in the default HACS catalog, so a plain search will not find it until the repo is added.)
3. After adding the repo, install the Meshtastic integration from HACS
4. Add the integration in Home Assistant Settings → Integrations → Add Integration → Meshtastic (see the integration's README for the exact config-flow steps)
5. Configure with your MQTT broker address, port, and topic prefix. The default root topic is `msh` (the full path is `msh/REGION/...`); see the integration's config options.
6. Nodes appear as Home Assistant devices with individual sensors for position, battery level, and telemetry values

What You Can Do in Home Assistant

- **Geofencing automations:** trigger actions when a node's GPS position enters or leaves a defined zone - useful for "person arriving home" automations using a mesh device as a low-cost tracker
- **Telemetry alerts:** send a notification when a remote temperature sensor exceeds a threshold (e.g., greenhouse overheating, freezer failure)
- **Battery low notifications:** alert when a field node's battery level drops below a set percentage
- **Dashboard visualization:** display node positions on a map card alongside other Home Assistant entities

Node-RED Alternative

Node-RED provides more flexibility for complex processing pipelines than the Home Assistant integration:

- The simplest, most portable approach is to use Node-RED's standard **MQTT-in** node plus a **JSON** node to consume the gateway's raw MQTT JSON from the broker. A community Meshtastic contrib node for direct serial/TCP connection may also exist - verify the exact package name on flows.nodered.org before relying on it.
- Build custom routing logic: filter messages by node ID, apply transformations, forward to multiple destinations simultaneously
- Suitable for multi-destination forwarding - e.g., send alerts to both Home Assistant and a Telegram bot at the same time

Example Node-RED Flow

A typical data logging flow:

1. **MQTT In node** - subscribes to the Meshtastic MQTT topic
2. **JSON parse node** - parses the raw MQTT payload
3. **Switch node** - filters for a specific node ID (e.g., only process packets from your sensor node)
4. **Function node** - extracts the desired telemetry fields (snake_case under `payload`, e.g. `payload.temperature`) and formats a row

5. **Google Sheets node** - appends the row to a Google Sheet for long-term logging

MeshCore and Home Assistant

MeshCore does not have a native MQTT gateway equivalent to Meshtastic's built-in WiFi gateway. A real community Home Assistant integration does exist (`github.com/meshcore-dev/meshcore-ha`, HA domain `meshcore`, built on the meshcore-py library). Integration options include:

- **meshcore-ha integration:** the community-maintained `meshcore-ha` integration (domain `meshcore`) connects to a MeshCore node via the meshcore-py library and exposes it to Home Assistant
- **Serial/USB bridge script:** connect a MeshCore node to a Raspberry Pi or other Linux machine via USB, use the MeshCore Python library (meshcore-py) to read incoming data, and publish it to an MQTT broker with a custom script
- This approach requires more custom development compared to the Meshtastic HACS integration

Privacy Considerations

Position and telemetry data from your nodes is visible to *anyone* who has access to your mesh channel. By default, Meshtastic devices transmit on the public LongFast channel, meaning nearby nodes operated by others can receive your telemetry.

- Configure a private channel with a unique name and PSK for nodes you don't want public network participants to observe
- Consider whether you want GPS position data from home-based sensor nodes to be visible on public MQTT bridges. The public default Meshtastic broker is `mqtt.meshtastic.org` (verify the current host against Meshtastic docs; as of 2026-06-08).
- Disable MQTT uplink on nodes with sensitive data if you use the public broker

Remote Asset Tracking

Use Case Overview

LoRa mesh networking provides a low-cost option for tracking assets in areas with no cellular coverage: vehicles, heavy equipment, livestock, shipping containers, or packages moving through rural or remote areas. Unlike cellular asset trackers that require a SIM and data plan for *every* asset, LoRa-based trackers report over your own mesh, avoiding per-asset SIM costs. Note that this does not eliminate recurring cost entirely: to deliver positions off the local mesh to a server you still need at least one gateway with internet backhaul, which may itself be a cellular/fixed connection with its own monthly cost. The savings come from not needing a SIM per tracked asset, not from eliminating connectivity costs altogether.

How It Works

1. The asset carries a LoRa node configured to broadcast GPS position packets at regular intervals
2. Any mesh node or gateway within range receives the position packet and propagates it through the mesh via multi-hop routing
3. A gateway node with internet access (WiFi or cellular backhaul) forwards the position via MQTT to a server
4. The server stores the position in a database and displays it on a web map or dashboard

Meshtastic MQTT + Cloud Visualization

Position packets forwarded via Meshtastic's MQTT gateway can be ingested by a variety of tools:

- **Grafana map panel:** InfluxDB stores the time-series position data; Grafana's Geomap panel displays asset trails and current positions
- **Custom Leaflet application:** a lightweight web app using the Leaflet.js mapping library can display live asset positions from a database backend

- **Home Assistant:** device tracker entities can be updated from MQTT position packets, integrating asset positions into your existing HA setup

MeshCore Asset Tracking

MeshCore advertisement packets can **optionally** include the sending node's GPS position - this is not unconditional. The advert location mode controls it: **share** broadcasts the live GPS location, **prefs** advertises a location stored in preferences, and **none** (the default) includes no position in adverts. So position is only present in adverts when the node is configured to share it. Where positions are being advertised, a repeater that hears an advert and logs the contact (node ID and time) can make it possible to reconstruct asset movement through a coverage area by analyzing which repeaters logged contact with a given node ID and when. This is useful for:

- Understanding which parts of a property or trail system an asset has visited
- Detecting when an asset enters or leaves a coverage zone
- Post-event reconstruction of movement without requiring a centralized database during the event

Update Interval Trade-off

Position update frequency involves a trade-off between tracking resolution, battery life, and mesh traffic:

- **Every 1 minute:** good resolution for fast-moving vehicles, but significantly higher battery drain and substantial added mesh traffic. On a shared LoRa channel, 1-minute position broadcasts are aggressive - they can quickly eat into practical airtime/duty-cycle budgets on a busy mesh and crowd out other nodes. Meshtastic deliberately defaults position/telemetry to much longer intervals (30 minutes). Reserve sub-minute or 1-minute rates for a very small number of nodes or short tests, not a fleet.
- **Every 10 - 30 minutes:** appropriate for most slow-moving assets (livestock, equipment); much better battery life, and roughly in line with Meshtastic's 30-minute default
- **Motion-triggered:** the most power-efficient approach - configure the node to transmit on movement detected via accelerometer, and go silent when the asset is stationary. Ideal for parked equipment or containers that move infrequently.

Hardware Options

- **RAK4631 + RAK1910 GPS module:** excellent for asset tracking - the RAK4631's nRF52840 has very low sleep current, and the modular WisBlock system allows GPS to be

powered down when not sampling. Best-in-class for long-term battery-powered deployments.
- **T-Echo:** integrated GPS and ePaper screen for field checking the last known position; good for assets that someone will physically inspect periodically (e.g., a piece of equipment on a large property)
- **T-Beam:** has built-in GPS and is popular for vehicle tracking; higher power consumption than the RAK solution but easier to integrate with vehicle 12V power

Coverage Considerations

Asset tracking over LoRa mesh requires mesh coverage in the asset's operating area. Without a mesh node or gateway within range, the asset simply does not report until it re-enters coverage:

- For wide-area tracking across a large property, plan repeater placement to improve coverage of the operating area
- A gateway with cellular backhaul at the edge of coverage can ensure position data reaches a server even if the local mesh is isolated
- Assets that travel outside the mesh coverage area will have gaps in their track. The mesh does **not** fill these gaps: positions a node broadcasts while it is out of range are lost entirely - there is no retroactive delivery once it returns. The server can only ever display the last position that was received while the asset was in range, so store the last known position and timestamp to show "last seen at X location at Y time."

Regulatory and Privacy Considerations

Transmitting GPS coordinates of people or assets has legal and ethical implications. The points below are general guidance, not legal advice - consult local law for your situation:

- **Consent:** ensure anyone being tracked (employees, family members) is aware of and consents to the tracking. Covert GPS tracking of people may be illegal depending on your jurisdiction - laws vary widely by state and country, so consult local law and obtain consent before tracking any individual.
- **Employees:** employer tracking of company vehicles during work hours is generally permissible in the US but varies by jurisdiction; consult applicable state employment law (this is general guidance, not legal advice)
- **Livestock and equipment:** tracking your own property is generally straightforward; tracking assets on shared or public land may have additional considerations
- **Channel access:** as with all mesh data, position packets are visible to anyone with access to the same channel. Use a private channel with a unique PSK for asset tracking deployments to prevent third parties from observing your assets' movements.