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

---

Revision #3
Created 2026-05-03 04:10:48 UTC by Mesh America Admin
Updated 2026-06-10 03:26:57 UTC by Mesh America Admin