Home Assistant & Grafana Integration

Home Assistant & Grafana Integration

Once your Meshtastic node is publishing to an MQTT broker, that data stream can feed home automation and monitoring platforms.

Home Assistant Integration

Home Assistant has a built-in MQTT integration that can subscribe to Meshtastic topics.

1. Navigate to **Settings → Devices & Services**
2. Click **Add Integration** and search for **MQTT**
3. Configure with broker address: `mqtt.meshtastic.org`. The public broker is **not** anonymous - supply username `meshdev` and password `large4cats`. It also enforces restrictions (zero-hop, location-precision and topic limits), so for full data a self-hosted broker is recommended (see below).
4. Subscribe to your node's topic tree, e.g., `msh/US/2/json/LongFast/#`. Note the `/json/` topic only carries data when `mqtt.json_enabled` is true on the gateway (and JSON output is not available on nRF52 gateways); the public broker also filters JSON output.
5. Create template sensors to parse position, telemetry, and message payloads from the JSON. The fields live under `value_json.payload.*` in snake_case (e.g. `value_json.payload.temperature`, `value_json.payload.relative_humidity`, `value_json.payload.text`).

With location data flowing into Home Assistant, you can create dashboards showing node positions, battery levels, and last-seen times, or trigger automations when specific nodes report in.

Grafana Integration

Grafana is commonly used for time-series visualization of mesh telemetry (signal strength, battery voltage, node uptime).

- Community projects exist that expose Meshtastic MQTT data as Prometheus metrics. Because Prometheus scrapes HTTP endpoints rather than MQTT directly, such a project typically bridges MQTT to a metrics endpoint that Prometheus then scrapes.
- Search GitHub for **meshtastic prometheus exporter** for current community projects.
- A typical stack: Meshtastic node → MQTT broker → Node-RED or custom MQTT-to-metrics bridge → Prometheus → Grafana.

Self-Hosted MQTT Broker

For production or privacy-sensitive deployments, run your own MQTT broker (Mosquitto is the standard choice) rather than relying on the public broker. This also enables TLS with your own certificates and username/password authentication.

```
# Mosquitto install (Debian/Ubuntu)
sudo apt install mosquitto mosquitto-clients
# Mosquitto 2.0+ defaults allow_anonymous to false and binds localhost-only,
# so add a listener and auth before remote clients can connect, e.g. in
# /etc/mosquitto/conf.d/local.conf:
#   listener 1883 0.0.0.0
#   password_file /etc/mosquitto/passwd   (with: mosquitto_passwd -c /etc/mosquitto/passwd
<user>)
#   allow_anonymous false
# Configure TLS (listener 8883 with cafile/certfile/keyfile) and ACLs as needed.
```

Revision #4
Created 2026-05-03 03:00:14 UTC by Mesh America Admin
Updated 2026-06-10 05:44:36 UTC by Mesh America Admin