

Meshtastic Module

Configuration Reference

Meshtastic's module configuration system enables advanced features beyond basic mesh communication. This page covers the most commonly needed modules and their key settings.

Range Test Module

Automates systematic coverage testing by sending timed packets that include position data.

```
# Enable and configure via CLI
meshtastic --set range_test.enabled true
meshtastic --set range_test.sender 60 # Send every 60 seconds
meshtastic --set range_test.save true # Save results to rangetest.csv on the device's internal
flash (ESP32 only)
```

The receiving node logs packets with GPS coordinates and signal data to its internal flash as `rangetest.csv` (ESP32 boards only; most boards have no SD card, so `range_test.save` has no effect on them). Download the file from the node's web interface at `meshtastic.local/rangetest.csv`, then import it into Google My Maps, uMap (OpenStreetMap), QGIS, or Google Earth to visualize coverage as a heatmap. **Note:** a range-test sender continuously floods airtime, so run it only briefly and on an isolated test channel to avoid disrupting your live mesh.

Telemetry Module

Controls how often nodes broadcast battery, environment, and air quality data.

```
meshtastic --set telemetry.device_update_interval 1800 # Battery/voltage every 30 min
meshtastic --set telemetry.environment_update_interval 900 # Temperature/humidity every 15 min
meshtastic --set telemetry.air_quality_enabled true # Enable if a particulate sensor (e.g.
PMSA003I) is attached
```

Telemetry data is visible in the app's node detail view. Excessive telemetry frequency contributes to network congestion in dense deployments - don't set intervals below 5 minutes for infrastructure nodes. **Note:** the Air Quality module requires a supported particulate sensor (e.g. PMSA003I). A BME680/BME68x reports gas resistance under Environment Telemetry, not the Air Quality module.

Position Module

Controls how and when GPS position is broadcast.

```
meshtastic --set position.position_broadcast_secs 1800 # Broadcast every 30 min
meshtastic --set position.broadcast_smart_minimum_distance 100 # Smart broadcast if moved 100m
meshtastic --set position.gps_update_interval 120 # Update GPS position every 2 min
meshtastic --set position.gps_mode ENABLED # DISABLED to save power on static nodes
```

For static repeaters: Disable GPS or set a fixed position instead of active GPS polling to save power:

```
meshtastic --set position.gps_mode DISABLED
meshtastic --setlat 47.6062 --setlon -122.3321 --setalt 150
```

External Notification Module

Triggers a buzzer, LED, or vibration motor on message receipt. Useful for personal portable nodes where you can't always watch the screen.

```
meshtastic --set external_notification.enabled true
meshtastic --set external_notification.alert_message true
meshtastic --set external_notification.alert_bell true # Triggers buzzer
meshtastic --set external_notification.output_ms 1000 # Buzzer on for 1 second
```

Store and Forward Module

Enables a node to store messages and deliver them to nodes that were offline when the message was sent. Requires an ESP32 device with onboard PSRAM (e.g. T-Beam v1.0+, T3S3). Records are held in volatile PSRAM (lost on reboot or power loss), not flash. nRF52 boards (RAK4631, T-Echo) cannot act as a Store and Forward server at all.

```
meshtastic --set store_forward.enabled true
meshtastic --set store_forward.records 0 # Max records to store; default 0 auto-sizes to ~2/3
of PSRAM (~11,000 records). 100 here is just an example
meshtastic --set store_forward.history_return_max 25 # Return up to 25 historical messages on
request
```

Note: [Store and Forward](#) is resource-intensive and delivery is best-effort and request-driven, not guaranteed. Enable it only on an ESP32 device with onboard PSRAM; nRF52 devices cannot run a Store and Forward server. This feature provides some of the functionality that MeshCore's room servers offer natively.

Serial Module

Bridges mesh messages to/from a connected serial device (microcontroller, computer, sensors). Enables custom IoT integrations without modifying firmware.

```
meshtastic --set serial.enabled true
meshtastic --set serial.baud BAUD_115200
meshtastic --set serial.timeout 0 # No timeout
meshtastic --set serial.mode TEXTMSG # Treat serial data as text messages
```

Use case: Connect an Arduino to a Meshtastic node via serial. The Arduino reads sensor data and sends it as a serial string, which the Meshtastic node broadcasts as a mesh message.

Canned Messages Module

Pre-loads a set of frequently used messages that can be selected and sent without typing. Especially useful on devices without keyboards (T-Echo, T1000-E).

```
meshtastic --set canned_message.enabled true
meshtastic --set-canned-message "OK, copy|Need assistance|ETA 15 minutes|At base|En route"
```

Messages are separated by pipe characters. On devices with buttons, scroll through the list and press to send. On the T-Echo, use the rotary encoder to navigate and click to send.

Module configuration best practices

- **Don't enable everything:** Each active module adds to channel traffic. Enable only what you actually use.
- **Set conservative intervals:** Telemetry and position intervals of 15 - 30 minutes are appropriate for most deployments. Shorter intervals increase traffic without proportional benefit.
- **Test after configuration:** After enabling any module, verify it behaves as expected before relying on it operationally.
- **Back up your config:** After a working configuration, export it via the web app so you can restore it after a firmware update.

Revision #5

Created 2026-05-03 04:04:18 UTC by Mesh America Admin

Updated 2026-06-09 12:13:09 UTC by Mesh America Admin