# Meshtastic Documentation

An offline copy of the essential pages of meshtastic.org, for firmware and apps version 2.8 · taken November 2026

Meshtastic is free software for small, cheap LoRa radios that pass short text messages and positions from one to another with no phone service and no internet. These are the project's own pages on setting a radio up, the phone app, the settings, range, antennas, and the two radios an ARK Vault uses. Every page says the address it came from.

For the plain-English version, with what to do first, see the library's **Mesh radio guide**.

The pages are unchanged, except that steps given in tabs (Android, Apple, web) are printed one after another, the steps for the command-line tool are left out, and links point to the website.

© Meshtastic contributors. The documentation is released under the GNU General Public License, version 3 (its text is at the end); its source is at github.com/meshtastic/meshtastic. Meshtastic® is a registered trademark of Meshtastic LLC. Meshtastic software components are released under various licenses, see GitHub for details. No warranty is provided - use at your own risk. This copy was made for the ARK library; it is not made or endorsed by Meshtastic.

# Contents


**Start here**

What Meshtastic is 4
How the mesh works 15
Radio settings and range 21
Range tests: how far people have reached 30
Encryption: who can read what 45

**Setting up a radio**

Getting started 48
Putting Meshtastic on an ESP32 radio (the web flasher) 67
Putting Meshtastic on an nRF52 radio (drag and drop) 68
First settings 72
Joining the local mesh 79

**The phone app (Android)**

First start 88
Connecting to a radio 93
Messages and channels 100
The list of radios (nodes) 110
Map and waypoints 119

**The phone app (Apple)**

The app for iPhone, iPad and Mac 126

**Settings**

LoRa: region, preset and hops 127
Channels 145
Device: roles 166
Choosing the right role (Meshtastic blog) 179
Position 183
Power 197
Bluetooth 213
Security 224
User: names 229
Tips 236

**Modules**

Range test 244
Traceroute 257

**Hardware**

Antennas 261
LoRa antennas 263
Heltec WiFi LoRa 32 (the ARK Vault's radio) 267
Heltec WiFi LoRa 32: buttons 299

SenseCAP Card Tracker T1000-E (the pocket radio) 300

**Questions**

Frequently asked questions 306
Licensing and trademark 308

**The licence: GNU General Public License version 3** 316

# Start here

## What Meshtastic is

From https://meshtastic.org/docs/introduction/

Meshtastic® is a project that enables you to use inexpensive LoRa radios as a long range off-grid communication platform in areas without existing or reliable communications infrastructure. This project is 100% community driven and open source!

17:41



MESHTASTIC

Primary Channel

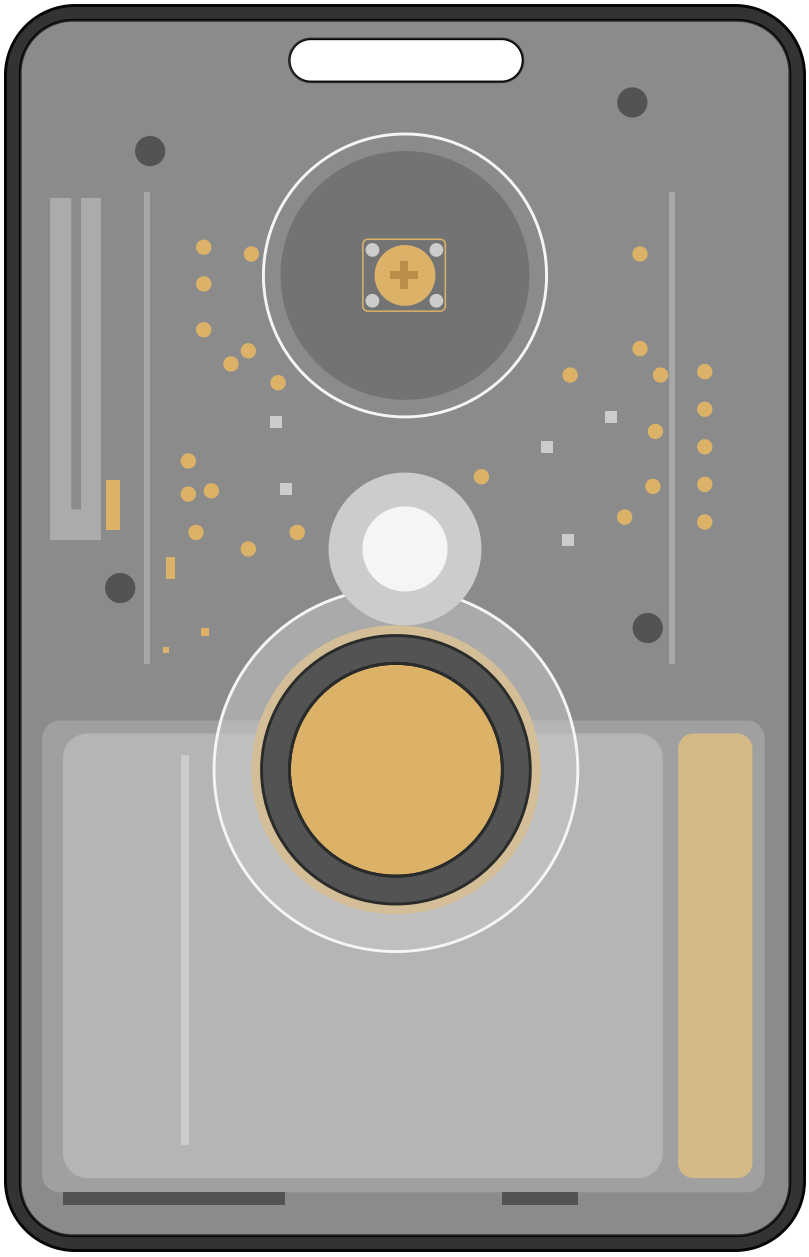


Send a message to the mesh

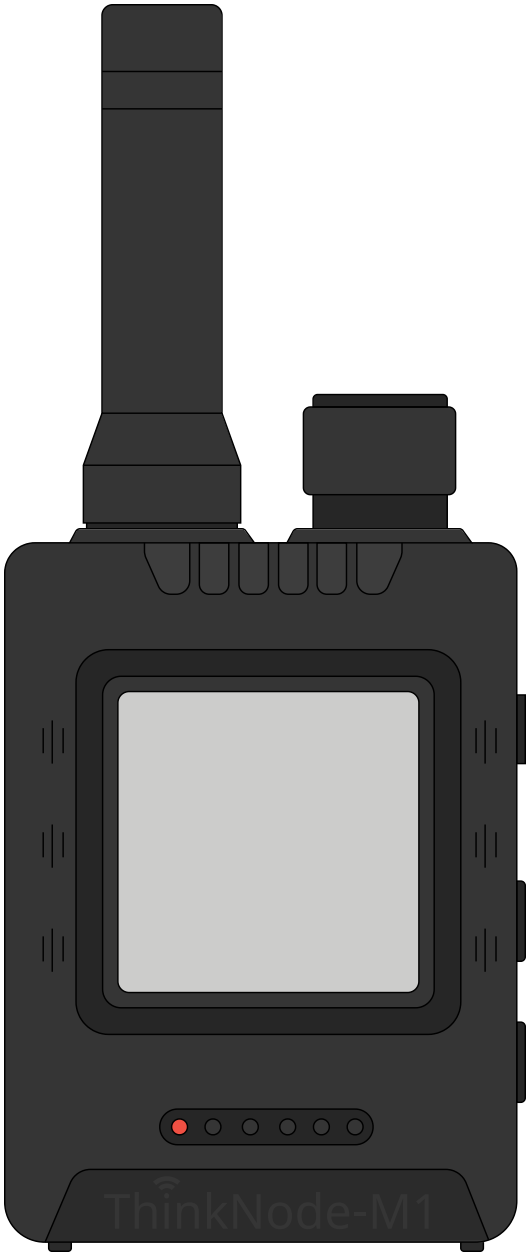
Type...



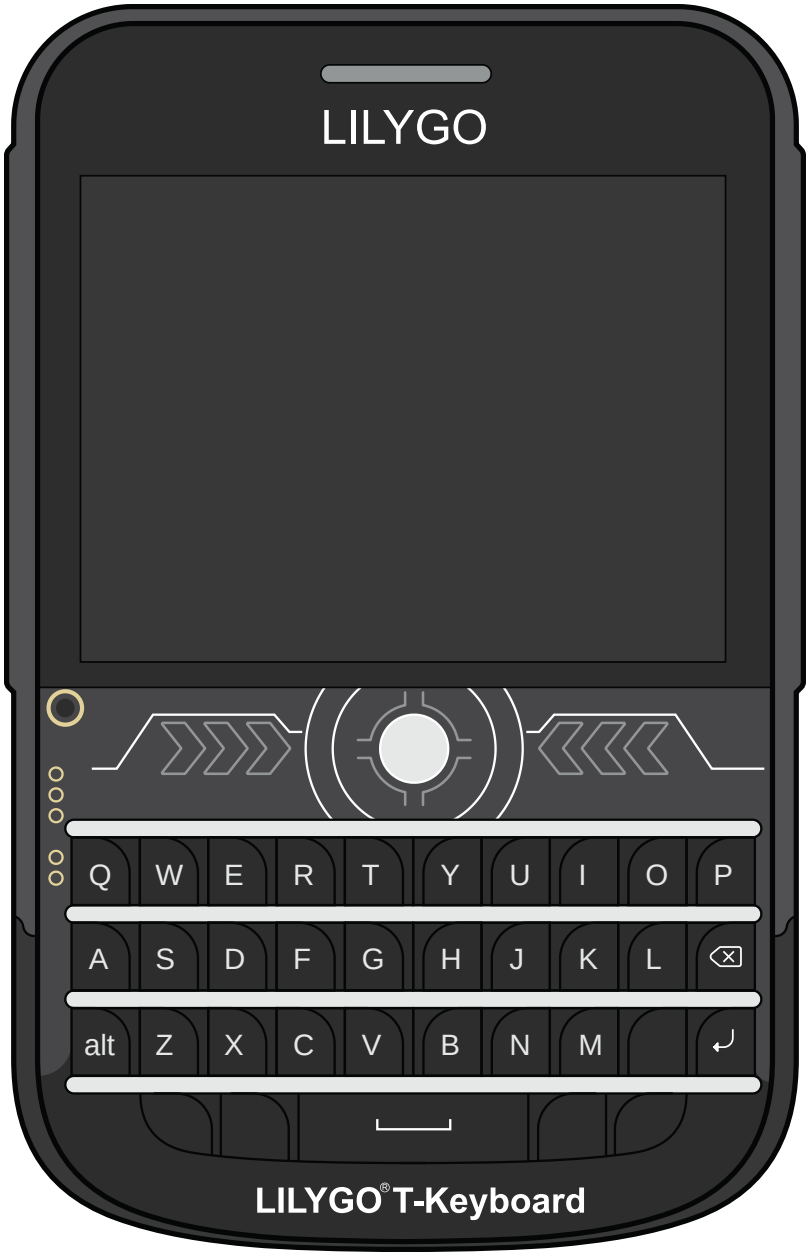
Client



Client



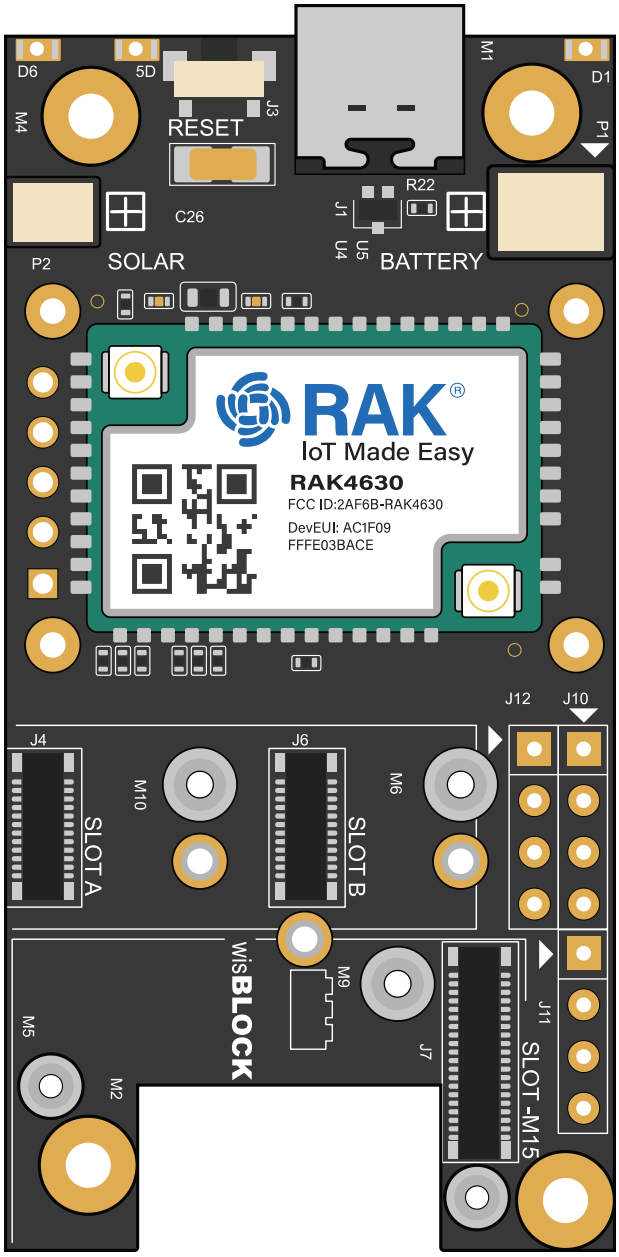

Client


LILYGO
Q W E R T Y U I O P
A S D F G H J K L
alt Z X C V B N M
LILYGO® T-Keyboard

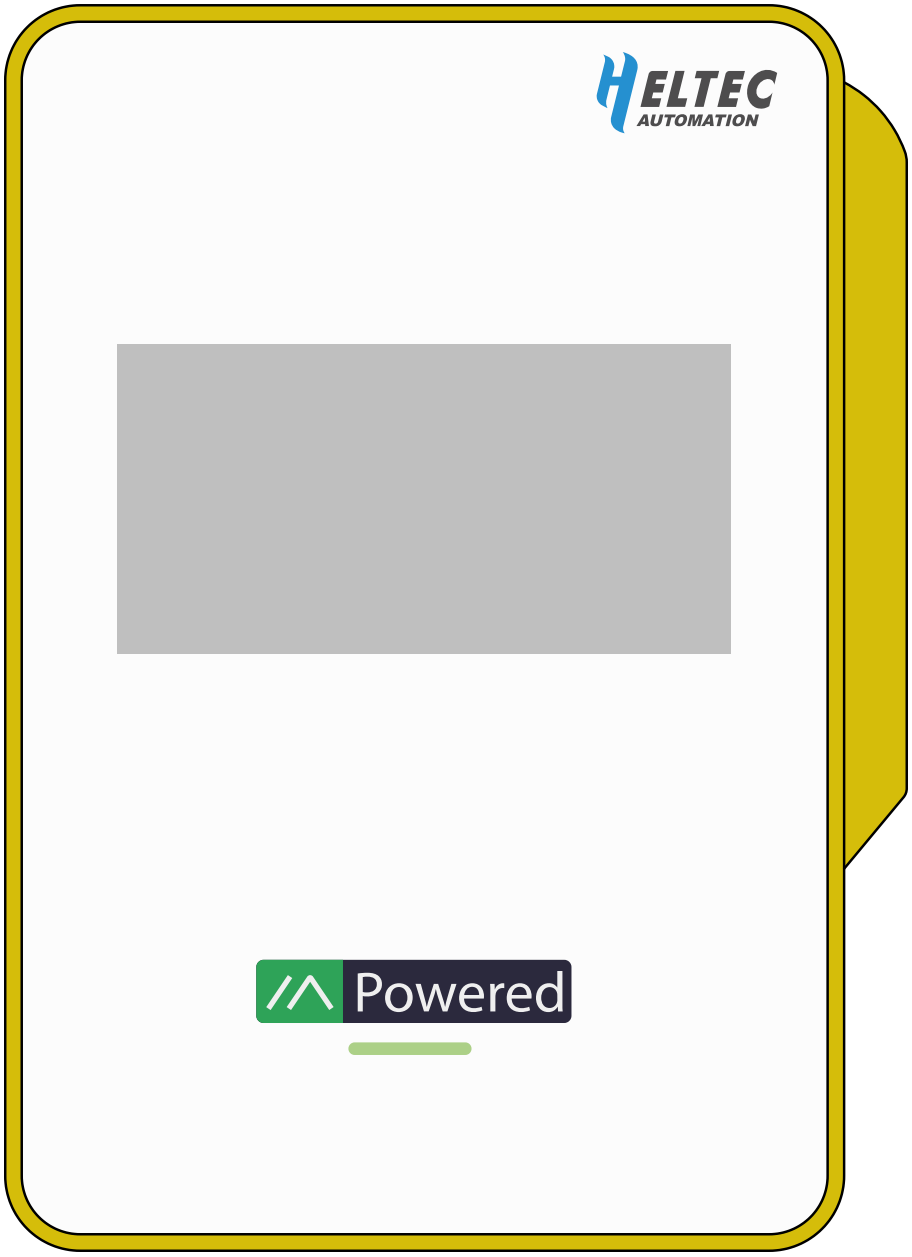
Client


WARNING: OPERATION WITHOUT AN ANTENNA
MAY PERMANENTLY DAMAGE THE DEVICE.
Station G2

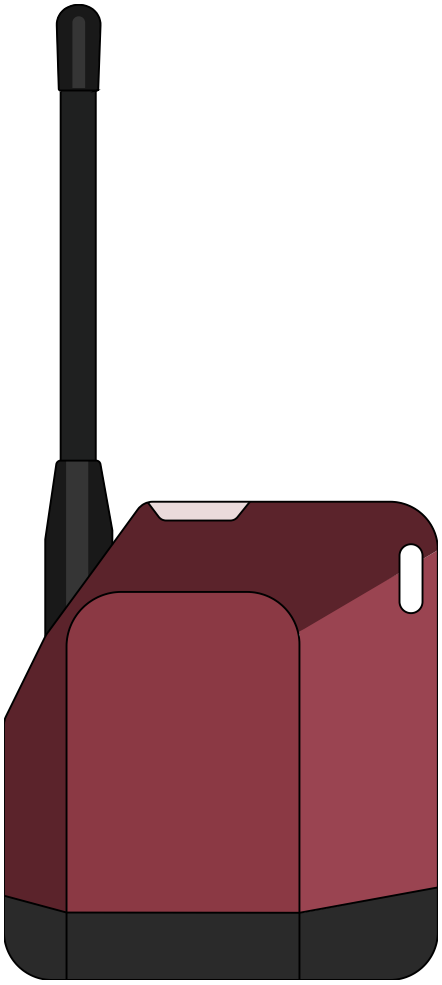
Router


D6
5D
M1
D1
M4
RESET
J3
P1
R22
C26
J1
U4
U5
P2
SOLAR
BATTERY
RAK®
IoT Made Easy
RAK4630
FCC ID:2AF6B-RAK4630
DevEUI: AC1F09
FFFE03BACE
J12
J10
J4
J6
M10
M6
SLOT A
SLOT B
WisBLOCK
M9
J7
J11
M5
M2
SLOT -M15

Client




Client



Client
meshtastic-cli

$ meshtastic --listen

Listening for messages...

LoRa
Bluetooth
WiFi
USB

## Features

- Long range (*331km record by MartinR7 & alleg*)
- No phone required for mesh communication
- Decentralized communication - no dedicated router required
- Encrypted communication
- Excellent battery life
- Send and receive text messages between members of the mesh

- Optional GPS based location features
- And more!

## How it works

Meshtastic utilizes LoRa, a long-range radio protocol, which is widely accessible in most regions without the need for additional licenses or certifications, unlike ham radio operations.

These radios are designed to rebroadcast messages they receive, forming a mesh network. This setup ensures that every group member, including those at the furthest distance, can receive messages.

Additionally, Meshtastic radios can be paired with a single phone, allowing friends and family to send messages directly to your specific radio. It's important to note that each device is capable of supporting a connection from only one user at a time.

If you are interested in a more technical overview of how Meshtastic works, visit the overview section below:

Technical Overview

## Contributors

Meshtastic is an open source project available on GitHub. Our generous volunteers donate their personal time to write and maintain this codebase. If you would like to contribute see our GitHub, join our Discord server, and read up on our Meshtastic Discussions.

Contribute!

## Start using Meshtastic

Hopefully your "Getting Started" experience is straight forward and headache free. If you encounter any issues, please consider updating our documentation to improve future user experiences or reach out on the forum or Discord.

Our support is 100% volunteer based. We are passionate about the project and hope to help newcomers become Meshtastic experts!

Getting Started

# How the mesh works

From https://meshtastic.org/docs/overview/mesh-algo/

## Current Algorithm

The routing protocol for Meshtastic is designed to support all kinds of different use-cases. Most importantly, it does not assume merely static nodes. If you want to test its theoretical performance, you can have a look at the simulator. The protocol is heavily influenced by the mesh routing algorithm used in RadioHead (which was used in very early versions of this project). It has four conceptual layers and (since version 2.6) uses a different approach for broadcasts and direct messages.

### A Note About Protocol Buffers

Because we want our devices to work across various vendors and implementations, we use Protocol Buffers pervasively. For purposes of this document you mostly only need to consider the MeshPacket and Sub-packet message types.

### Layer 0: LoRa Radio

All data is converted into LoRa symbols which are sent to the radio for transmission. The details are described elsewhere, but it is worth noting that in addition to the converted packet bytes described below, there is also a preamble sent at the start of any data packet.

This preamble allows receiving radios to synchronize clocks and start framing. Below 1 GHz, Meshtastic uses a preamble length of 16, which is longer than the minimum preamble length of 8, to let SX126x LoRa receivers sleep for a while, which lowers power consumption. Regions in the 2.4 GHz band use a preamble length of 12.

After the preamble comes the LoRa Physical Header, which contains information about the packet length as well as a sync word to distinguish networks. For Meshtastic, it is set to `0x2B`.

### Layer 1: Unreliable Zero Hop Messaging

This layer is conventional non-reliable LoRa packet transmission. A packet generated by a Meshtastic device has the following representation before encoding for transmission:

| Offset | Length | Type | Usage |
|---|---|---|---|
| 0x00 | 4 bytes | Integer | Packet Header: Destination. The destination's unique NodeID. `0xFFFFFFFF` for broadcast. Little-endian. |
| 0x04 | 4 bytes | Integer | Packet Header: Sender. The sender's unique NodeID. Little-endian. |
| 0x08 | 4 bytes | Integer | Packet Header: The sending node's unique packet ID for this packet. Little-endian. |
| 0x0C | 1 byte | Bits | Packet Header: Flags. See the header flags for usage. |
| 0x0D | 1 byte | Bits | Packet Header: Channel hash. Used as hint for decryption for the receiver. |

| Offset | Length | Type | Usage |
|---|---|---|---|
| 0x0E | 1 byte | Bytes | Packet Header: Next-hop used for relaying. |
| 0x0F | 1 byte | Bytes | Packet Header: Relay node of the current transmission. |
| 0x10 | Max. 239 bytes | Bytes | Actual packet data: the encoded `Data` protobuf, plus 12 bytes when PKC or AEAD encryption is used. Unused bytes are not transmitted. |

### Packet Header Flags

| Index | # of Bits | Usage |
|---|---|---|
| 0 | 3 | HopLimit (see note in Layer 3) |
| 3 | 1 | WantAck |
| 4 | 1 | ViaMQTT (packet came via MQTT) |
| 5 | 3 | HopStart (original HopLimit) |

### Usage Details

- **Packet Header:** is described directly by the `PacketHeader` class in the C++ source code. But indirectly it matches the first portion of the `MeshPacket` protobuf definition. Note that the packet header is not encoded using a protobuf, but is sent as raw bytes. This both saves airtime and allows receiving radio hardware to optionally filter packets before waking the main CPU.
- **Packet Header - NodeIDs:** A node's number is a CRC-32 of its public key. Before the node has a key, for example while its region is unset, the number comes from its MAC address.
- **Packet Header - Unique ID:** The ID is a large, 32 bit ID to ensure there is enough unique state to protect an encrypted payload from attack.
- **Payload:** An encrypted and packed protobuf encoding of the SubPacket protobuf. Only the SubPacket is encrypted, while headers are not. This allows the option of eventually allowing nodes to route packets without knowing anything about the encrypted payload. For more information, see the encryption and Protobuf API Reference. A packet that would exceed the maximum length isn't sent.

### Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA)

Meshtastic adopts CSMA/CA, similar to what is used in Wi-Fi. This means that all transmitters must perform Channel Activity Detection (CAD) before attempting to transmit. If the channel is considered busy, the node will wait until it is not anymore. Since once the channel becomes idle multiple nodes might want to start transmitting, a node has to wait a random multiple of slot times. The slot time is the time needed to reliably perform CAD. The number of slot times to wait is randomly picked from a contention window (CW), whose size depends on the current channel utilization. The contention window is larger for a higher channel utilization, to limit the chance of collisions.

### Layer 2: Reliable Zero Hop Messaging

This layer adds reliable messaging between the node and its immediate neighbors only.

The default messaging provided by Layer 1 is extended by setting the `WantAck` flag in the MeshPacket protobuf. If `WantAck` is set, the following documentation from mesh.proto applies:

> This packet is being sent as a reliable message, we would prefer it to arrive at the destination. We would like to receive an ACK packet in response.
>
> Broadcast messages treat this flag specially: Since ACKs for broadcasts would rapidly flood the channel, the normal ACK behavior is suppressed. Instead, the original sender listens to see if at least one node is rebroadcasting this packet (because naive flooding algorithm). If it hears that, the odds (given typical LoRa topology) are very high that every node should eventually receive the message. So FloodingRouter.cpp generates an implicit ACK which is delivered to the original sender. If after some time we don't hear anyone rebroadcast our packet, we will timeout and re-transmit, using the regular resend logic.

If a transmitting node does not receive an ACK (or NAK) packet after a certain expiration time, it will use Layer 1 to attempt a re-transmission of the sent packet. A reliable broadcast is sent a maximum of three times in total, counting the first transmission, and a reliable direct message a maximum of five. If no ACK or NAK has been received by then the local node will internally generate a NAK (either for local consumption or use by higher layers of the protocol). The re-transmission expiration time is based on the maximum time it would take to receive an (implicit) ACK, taking into account the airtime of the sent packet, the current channel utilization, and a processing delay. For direct messages, the intended recipient will also send a real ACK all the way back to the original sender, but the device will only retransmit when it received no ACK at all.

### Layer 3: Multi-Hop Messaging

## Broadcasts using Managed Flooding

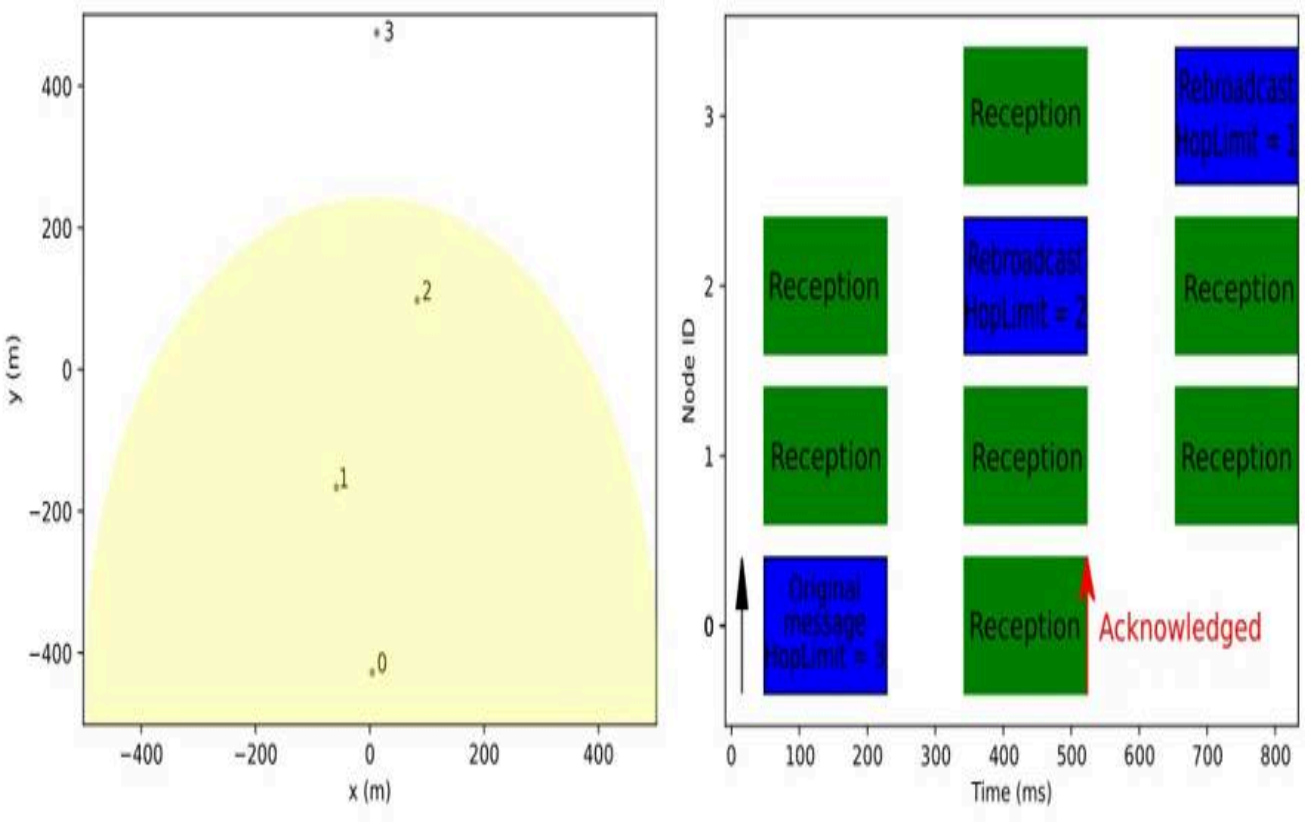
Given the various use-cases and scenarios Meshtastic supports, most of our protocol is built around flooding, meaning that every node rebroadcasts a packet it receives, up till a certain hop limit. However, an important difference in Meshtastic is that before rebroadcasting, a node listens a short while to see if another node has already rebroadcasted the packet. If so, it will not rebroadcast. "Managed Flooding" is therefore a better term.

The principle is as follows. If any mesh node sees a packet with a HopLimit other than zero, it will decrement that HopLimit and attempt to rebroadcast on behalf of the original sending node. In order to promote letting nodes that are further away flood the message, such that the message eventually reaches farther, the contention window (see Layer 1) for a flooding message depends on the Signal-to-Noise Ratio (SNR) of the received packet. The CW size is small for a low SNR, such that nodes that are further away are more likely to flood first and closer nodes that hear this will refrain from flooding. The exception is the `ROUTER` and

`ROUTER_LATE` roles, which rebroadcast even if they hear another node rebroadcasting. `CLIENT_BASE` does the same for packets to or from its favorited nodes.

### Example

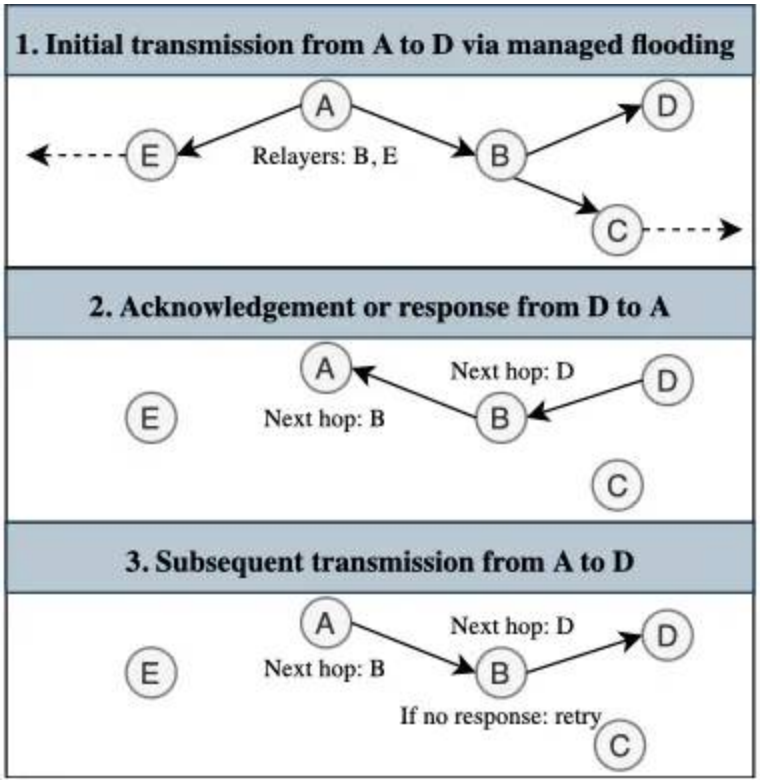
Below you see an example topology consisting of four nodes in `CLIENT` mode, where at a certain point node 0 wants to send a broadcast message. Due to limited coverage, it only reaches nodes 1 and 2. Since node 2 is farther away, its SNR is lower and therefore starts rebroadcasting earlier than 1. After node 0 received this rebroadcast, its message is acknowledged. Note that a message is already acknowledged once a rebroadcast from any Meshtastic node (whether or not it has the same encryption key) is received. Since node 1 heard the rebroadcast by 2, it will not rebroadcast again. Node 3 heard the message for the first time and the HopLimit is not yet zero, so it starts a rebroadcast for potential other receivers.




## Direct messages using Next-Hop routing

Since version 2.6, Meshtastic uses a different approach for direct messages. Initially, the managed flooding approach as mentioned before is used to reach the destination. We'll then keep track of the node(s) that are trying to relay the packet for us. If upon a successful delivery, a response comes back (e.g., a NodeInfo response, acknowledgment or traceroute) and the node that relays this towards you was also (one of) the node(s) that relayed the original packet, it will be denoted as next-hop from now on. This means that instead of letting all nodes try to relay the packet, only the node for which the next-hop byte matches will. Note

that this is determined per hop, so if there is an asymmetric link or a node on older firmware in between, managed flooding will be used on this hop. When a node moves, or RF conditions change, it might be the next-hop is not valid anymore. Therefore, a node will always fall back to managed flooding at the last retransmission attempt if it doesn't hear its next-hop relay. The procedure is visualized in the following diagram:




## Regular broadcast intervals

Without additional configuration, a node broadcasts only its NodeInfo, every 3 hours by default (`device.node_info_broadcast_secs`).

Device telemetry and position are off by default:

1. Device telemetry
    - Turned on with `telemetry.device_telemetry_enabled`
    - Interval set by `telemetry.device_update_interval`
    - Default: 1 hour, or 12 hours for `ROUTER` and `ROUTER_LATE`
2. Position
    - Shared only on channels where position precision is above 0
    - Interval set by `position.position_broadcast_secs`
    - Default: 1 hour, or 12 hours for `ROUTER` and `ROUTER_LATE`, with smart broadcast sending sooner after the node moves

As meshes grow larger and traffic becomes more contentious, the firmware will increase these intervals. This is in addition to duty cycle, channel, and air-time utilization throttling.

For meshes larger than 40 nodes (nodes seen in the past 2 hours), the firmware scales back position and telemetry intervals using the following algorithm:

```
ScaledInterval = Interval * (1.0 + (NumberOfOnlineNodes - 40) * 2^SF / (BW_kHz * 100))
```

`SF` and `BW_kHz` are the spreading factor and bandwidth in use. On Long Fast (SF11, 250 kHz), each node over 40 adds 0.08192. For example, a mesh of 62 online nodes on Long Fast stretches a 1-hour interval to about 168 minutes. Nodes in the `ROUTER`, `ROUTER_LATE`, `SENSOR`, `TRACKER`, and `TAK_TRACKER` roles aren't scaled.

# Radio settings and range

From https://meshtastic.org/docs/overview/radio-settings/



info

Meshtastic is **not** LoRaWAN, Helium or TTN (TheThingsNetwork). Meshtastic uses the full spectrum frequency range designated to LoRa technology per region. This allows for several hundred possible frequency channels in the US region alone.



info

> Power limits will generally be lifted in the software if `is_licensed` is set to `true`. See Ham Mode for more information.

## Frequency Slot Calculator

Filter regions

Region: US
Modem Preset: LONG_TURBO Default Frequency Slot: 14
Frequency Slot: 14 Number of slots: 52
Frequency of slot: 908.75

## Europe Frequency Bands

EU countries will generally be covered by the Radio Equipment Directive (2014/53/EU) and subsequent amendments, which is then enacted in local laws. See additional information about Radio equipment Directive 2014/53/EU. The bands below were selected based on the maximum power and the duty cycle available. It may be possible to use other frequency bands for specific applications, but these are not currently supported "out of the box".

### 433 MHz

The maximum power allowed for Europe is +10 dBm ERP (Effective Radiated Power).

The band range is from 433 to 434 MHz.

There are four frequency slots defined with the standard radio preset `LongFast`. After factory reset the radio will be set to frequency slot 4 with a center frequency of 433.875 MHz.

### 868 MHz

The maximum power allowed for Europe is +27 dBm ERP (Effective Radiated Power).

The band range is from 869.40 to 869.65 MHz. This is less than the 863–870 MHz range defined as SRD (Short Range Devices) Band, but allows for a higher ERP and a duty cycle of 10%.

There is one frequency slot defined with the standard radio preset `LongFast`. After factory reset the radio will be set to frequency slot 1 with a center frequency of 869.525 MHz.

It is worth noting that 868 MHz is generally the most popular frequency band for Meshtastic in Europe.

## North America Frequency Bands

### 915 MHz (ISM Band)

The maximum output power for North America is +30 dBm ERP (Effective Radiated Power).

The band range is from 902 to 928 MHz.

In North America, the available Frequency Slots depend on the bandwidth setting, which is included in the selected Radio Preset. The standard Preset, LongFast, provides 104 Frequency Slots. After a factory reset, the radio defaults to LongFast with a Frequency Slot of 0, which translates to Slot 20 (centered at 906.875 MHz) based on the channel hashing algorithm.

# Data Rates

## Considerations

Various data-rate options are available when configuring a frequency slot and are inversely proportional to the theoretical range of the devices.

- **Spreading Factor (SF)** - How much we "spread" our data over time.
  - Each step up in Spreading Factor doubles the airtime to transmit.
  - Each step up in Spreading Factor adds about 2.5db extra link budget.
  - **Note:** Spreading Factors 5 and 6 are only supported on 2nd generation LoRa chips (SX126x, LR11xx, SX128x). 1st generation chips (SX127x / RF95) are limited to SF7-SF12. If you select SF5/6 on an unsupported device, the firmware will automatically clamp it to SF11.
- **Bandwidth** - How big of a slice of the spectrum we use.
  - Each doubling of the bandwidth is almost 3db less link budget.
  - Bandwidths less than 31 may be unstable unless you have a high quality Crystal Oscillator.
- **Coding Rate** - How much redundancy we encode to resist noise.
  - Increasing coding rate increases reliability while decreasing data-rate.
  - 4/5 - 1.25x overhead
  - 4/6 - 1.5x overhead
  - 4/7 - 1.75x overhead
  - 4/8 - 2x overhead

## Presets

Meshtastic has nine general-purpose LoRa presets. These are the most common settings and have been proven to work well:

| Radio Preset | Alt Preset Name | Data-Rate | SF / Symbols | Coding Rate | Bandwidth | Link Budget |
|---|---|---|---|---|---|---|
| Short Range / Turbo | Short Turbo | 21.88 kbps | 7 / 128 | 4/5 | 500 kHz[1] | 140 dB |
| Short Range / Fast | Short Fast | 10.94 kbps | 7 / 128 | 4/5 | 250 kHz | 143 dB |
| Short Range / Slow | Short Slow | 6.25 kbps | 8 / 256 | 4/5 | 250 kHz | 145.5 dB |

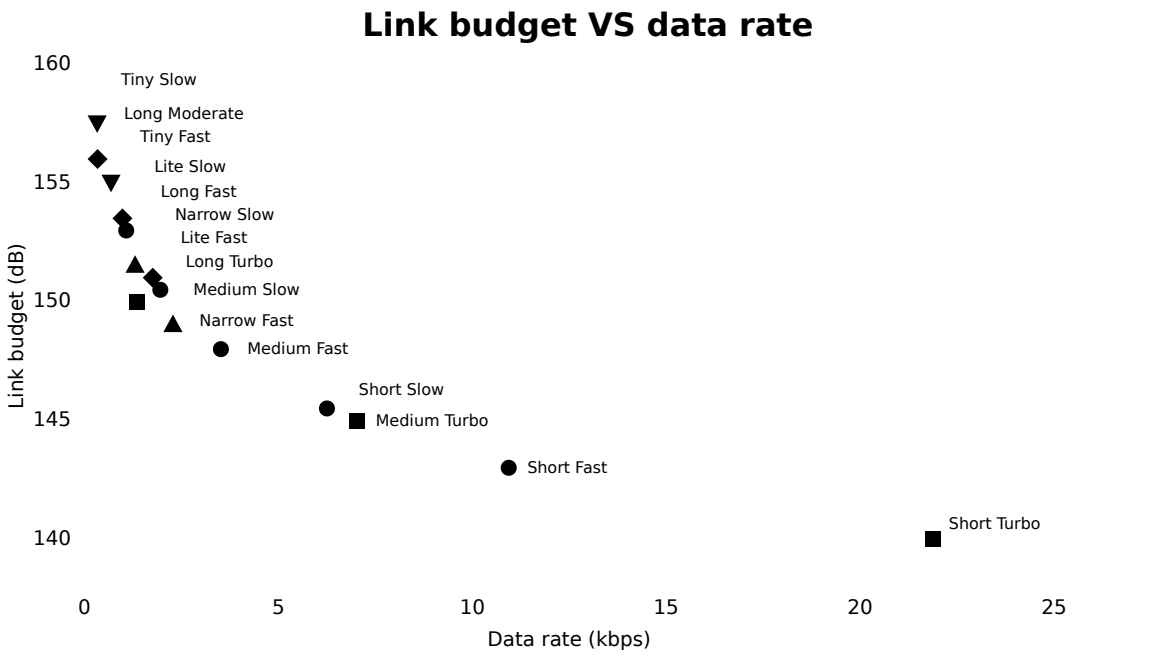
| Radio Preset | Alt Preset Name | Data-Rate | SF / Symbols | Coding Rate | Bandwidth | Link Budget |
|---|---|---|---|---|---|---|
| Medium Range / Turbo | Medium Turbo | 7.03 kbps | 9 / 512 | 4/5 | 500 kHz[1] | 145 dB |
| Medium Range / Fast | Medium Fast | 3.52 kbps | 9 / 512 | 4/5 | 250 kHz | 148 dB |
| Medium Range / Slow | Medium Slow | 1.95 kbps | 10 / 1024 | 4/5 | 250 kHz | 150.5 dB |
| Long Range / Turbo | Long Turbo | 1.34 kbps | 11 / 2048 | 4/8 | 500 kHz[1] | 150 dB |
| Long Range / Fast | Long Fast | 1.07 kbps | 11 / 2048 | 4/5 | 250 kHz | 153 dB |
| Long Range / Moderate | Long Moderate | 0.34 kbps | 11 / 2048 | 4/8 | 125 kHz | 156 dB |



note

The link budget used by these calculations assumes a transmit power of 22 dBm and an antenna with 0 dB gain. Adjust your link budget assumptions based on your actual devices. Data-rate in this table is the theoretical max but doesn't account for packet headers, hops and re-transmissions. Calculations based on data from the official Semtech LoRa calculator.

- ▼ 15.6 kHz
- ▲ 62.5 kHz
- ◆ 125 kHz
- ● 250 kHz
- ■ 500 kHz




Filter presets

## Narrowband presets

Three narrowband preset families serve bands where the 250 kHz and 500 kHz general-purpose presets are not permitted. Each family belongs to specific regions and can't be selected anywhere else:

- **Lite** — `EU_866`, the 865.6 - 867.6 MHz European SRD band.
- **Narrow** — `EU_N_868`, plus the 70cm and 1.25m amateur regions (`ITU1_70CM`, `ITU2_70CM`, `ITU3_70CM`, `ITU2_125CM`).
- **Tiny** — the 2m amateur regions (`ITU1_2M`, `ITU2_2M`, `ITU3_2M`), where many administrations cap the occupied bandwidth well below 62.5 kHz.

| Radio Preset | Alt Preset Name | Data-Rate | SF / Symbols | Coding Rate | Bandwidth | Link Budget |
|---|---|---|---|---|---|---|
| Lite / Fast | Lite Fast | 1.76 kbps | 9 / 512 | 4/5 | 125 kHz | 151 dB |
| Lite / Slow | Lite Slow | 0.98 kbps | 10 / 1024 | 4/5 | 125 kHz | 153.5 dB |
| Narrow / Fast | Narrow Fast | 2.28 kbps | 7 / 128 | 4/6 | 62.5 kHz | 149 dB |
| Narrow / Slow | Narrow Slow | 1.30 kbps | 8 / 256 | 4/6 | 62.5 kHz | 151.5 dB |

| Radio Preset | Alt Preset Name | Data-Rate | SF / Symbols | Coding Rate | Bandwidth | Link Budget |
|---|---|---|---|---|---|---|
| Tiny / Fast | Tiny Fast | 0.68 kbps | 7 / 128 | 4/5 | 15.6 kHz | 155 dB |
| Tiny / Slow | Tiny Slow | 0.33 kbps | 8 / 256 | 4/6 | 15.6 kHz | 157.5 dB |

### Custom settings

Custom settings can be applied by using supported software.

The settings apply without a restart. A channel's QR[2] code and URL carry its LoRa settings, so share the updated QR code or URL with your other nodes.

Some example settings:

| Data-rate | SF / Symbols | Coding Rate | Bandwidth | Link Budget | Note |
|---|---|---|---|---|---|
| 37.50 kbps | 6 / 64 | 4/5 | 500 kHz | 129 dB | Fastest possible speed |
| 3.125 kbps | 8 / 256 | 4/5 | 125 kHz | 143 dB | |
| 1.953 kbps | 8 / 256 | 4/8 | 125 kHz | 143 dB | |
| 1.343 kbps | 11 / 2048 | 4/8 | 500 kHz | 145 dB | |
| 1.099 kbps | 9 / 512 | 4/8 | 125 kHz | 146 dB | |
| 0.814 kbps | 10 / 1024 | 4/6 | 125 kHz | 149 dB | |
| 0.610 kbps | 10 / 1024 | 4/8 | 125 kHz | 149 dB | |
| 0.488 kbps | 11 / 2048 | 4/6 | 125 kHz | 152 dB | |
| 0.073 kbps | 12 / 4096 | 4/5 | 31 kHz | 160 dB | Low resilience to noise |
| 0.046 kbps | 12 / 4096 | 4/8 | 31 kHz | 160 dB | High resilience to noise |

The link budget used by these calculations assumes a transmit power of 17 dBm and an antenna with 0 dB gain. Adjust your link budget assumptions based on your actual devices.

These frequency slot settings may not have been tested. Use at your own discretion. Share on the Meshtastic Discussions page with your successes or failure.

## Cryptography

The pre-shared key (PSK) used by the devices can be an AES128 or AES256 sequence. Alternatively, encryption can be turned off, which may be useful if you are operating under a ham radio license.

## Footnotes

1. The Turbo presets use 500 kHz bandwidth, which is not legal to use in all regions. They are unavailable in `EU_868` and in every narrowband region. ↩ ↩[2] ↩[3]

2. The Meshtastic QR code (or Channel URL) allows users to quickly share channel and LoRa settings, making it easy to configure multiple nodes with matching settings for communication. Scanning a QR code applies all included channel settings and LoRa configuration settings, so be sure to review what these settings include before proceeding. Only scan QR codes from trusted sources.

   For more details, see:

   - Channel Config
   - LoRa Config

   ↩

# Range tests: how far people have reached

From https://meshtastic.org/docs/overview/range-tests/

**Ground to Ground Ground to Air**

### Current Ground Record: 331km

- **Range:** 331km (205 miles)
- **Record Holders:** *MartinR7* & *alleg*
- **Source:** reddit

### Modem Settings

Very Long Slow Preset

- **Frequency:** 868 MHz
- **Bandwidth:** 62.5
- **Spread Factor:** 12
- **Coding Rate:** 4/8

### Node A

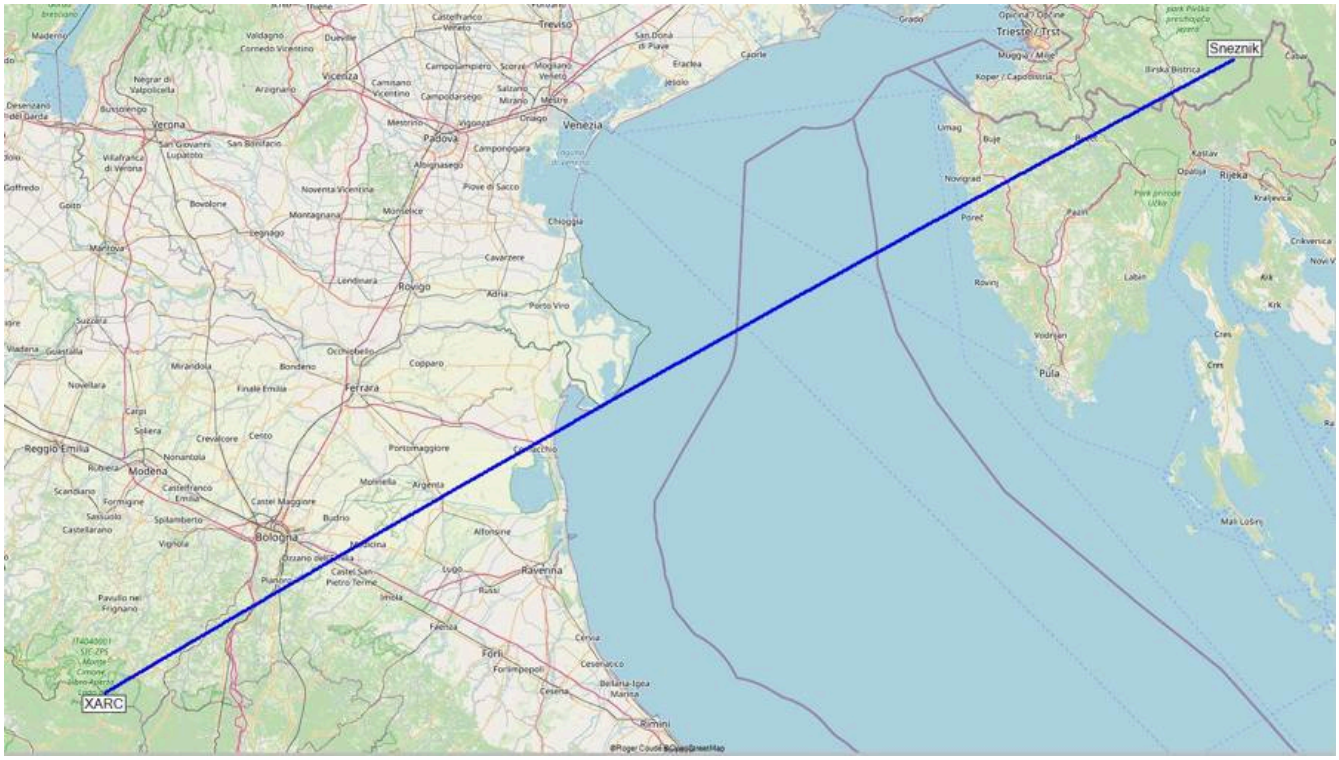
- **Device:** RAK4631 Core+RAK19003+RAK1906+INA219
- **Firmware Version:** 2.3.6
- **Antenna:** 55cm collinear 868mhz (AliExpress)

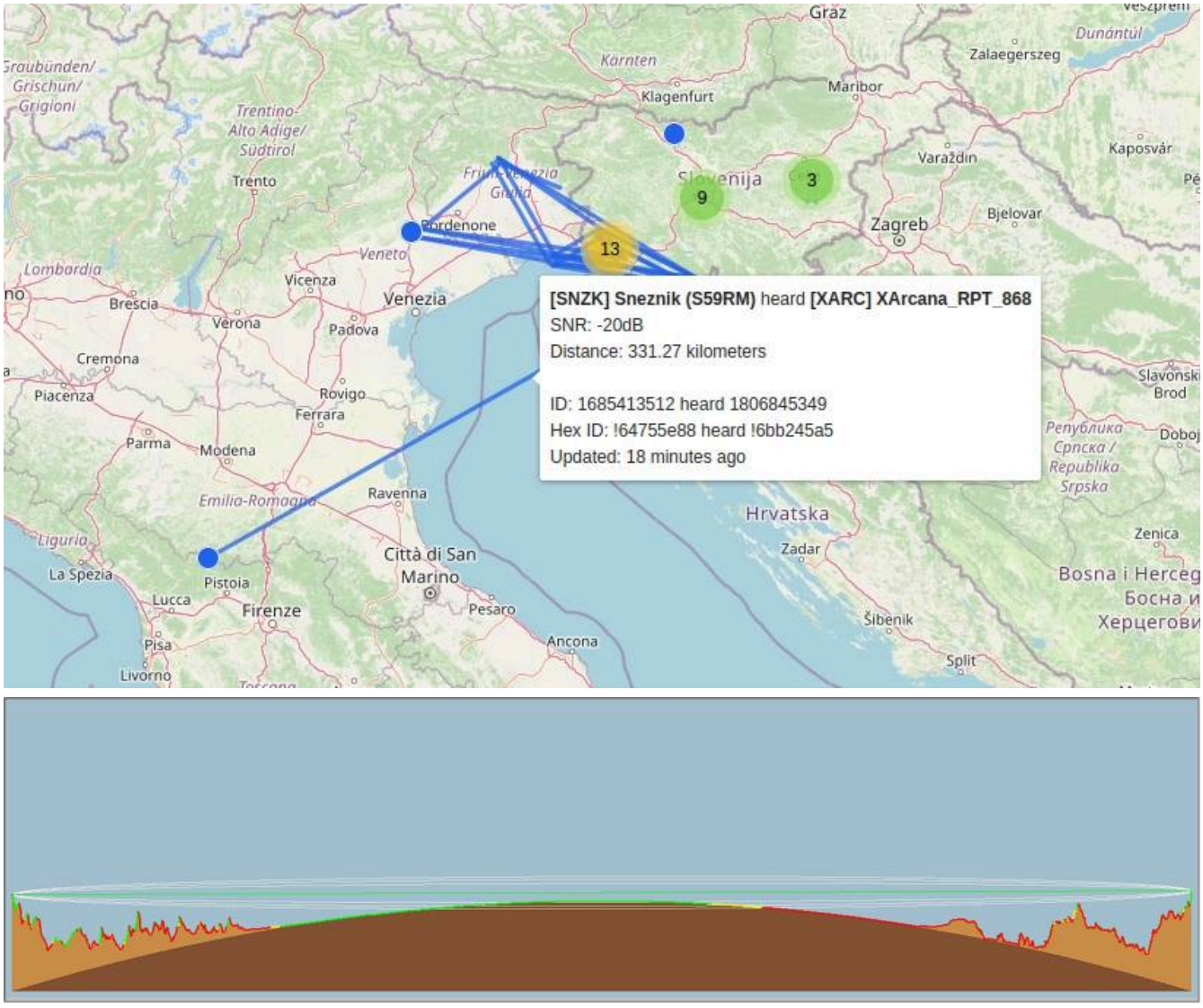
### Node B

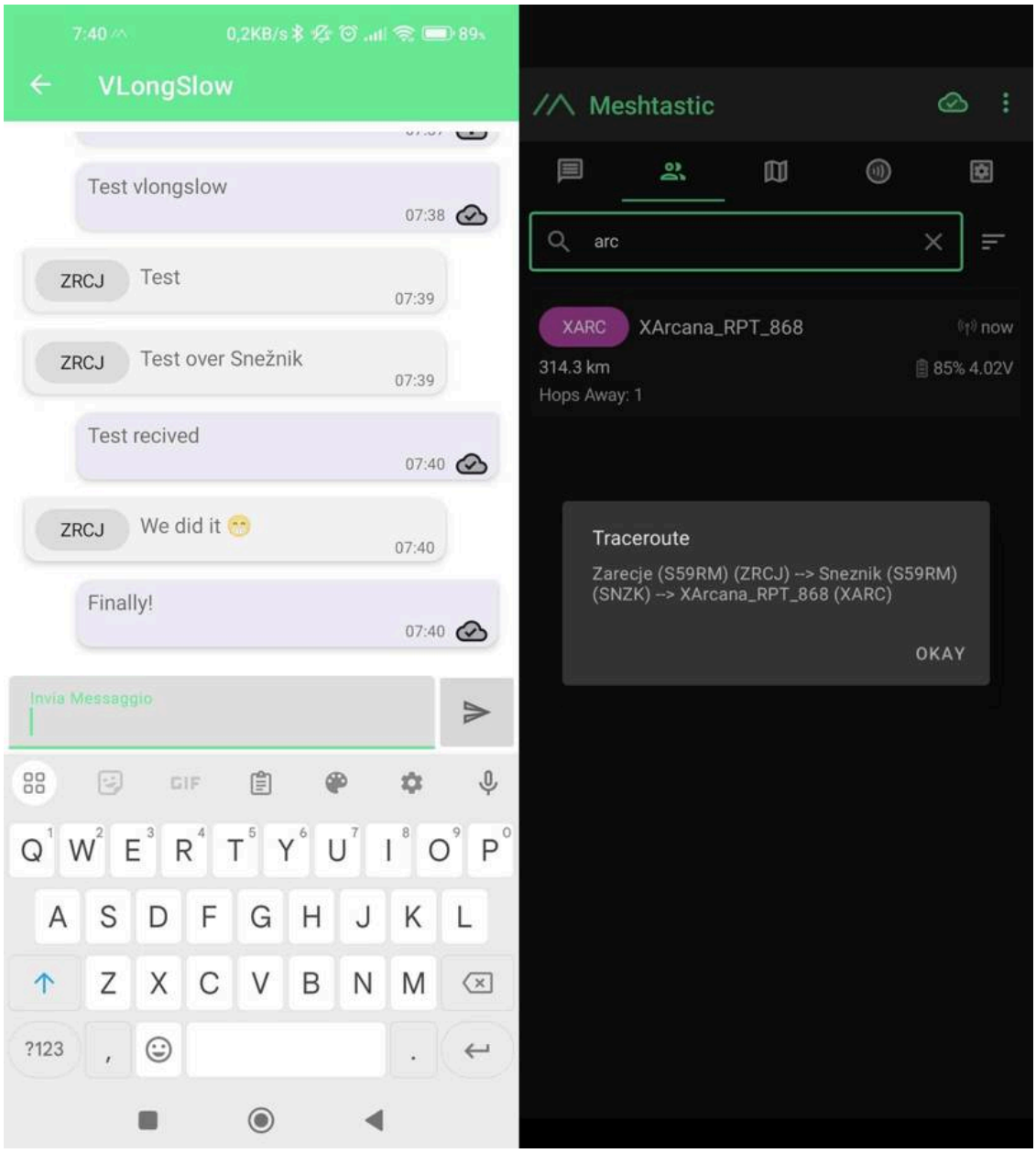
- **Device:** RAK4631 Core+RAK19003+RAK1901
- **Firmware Version:** 2.4.2
- **Antenna:** RAKARJ17






Sneznik
XARC
Venezia
Padova
Trieste / Trst
Rijeka
Pula
Ravenna
Ferrara
Rovigo
Bologna
Modena
Reggio Emilia
Verona
Vicenza
Treviso
Rimini


[SNZK] Sneznik (S59RM) heard [XARC] XArcana_RPT_868
SNR: -20dB
Distance: 331.27 kilometers
ID: 1685413512 heard 1806845349
Hex ID: !64755e88 heard !6bb245a5
Updated: 18 minutes ago
Graz
Kärnten
Klagenfurt
Maribor
Zalaegerszeg
Varaždin
Kaposvár
Slovenija
Zagreb
Bjelovar
Trento
Pordenone
Veneto
Vicenza
Venezia
Padova
Verona
Brescia
Lombardia
Cremona
Piacenza
Rovigo
Ferrara
Parma
Modena
Emilia-Romagna
Ravenna
Città di San Marino
Pesaro
Ancona
Pistoia
Firenze
Lucca
Pisa
Livorno
La Spezia
Liguria
Hrvatska
Zadar
Šibenik
Split
Zenica
Bosna i Herceg
Slavonski Brod
Doboj
9
3
13




### Previous Ground Record: 254km

- **Range:** 254km (158 miles)
- **Record Holders:** *kboxlabs*
- **Source:** Meshtastic Discourse (archive)

### Modem Settings

Long_Fast Preset

- **Frequency:** 915 MHz
- **Bandwidth:** 250
- **Spread Factor:** 11
- **Coding Rate:** 4/8

### Node A

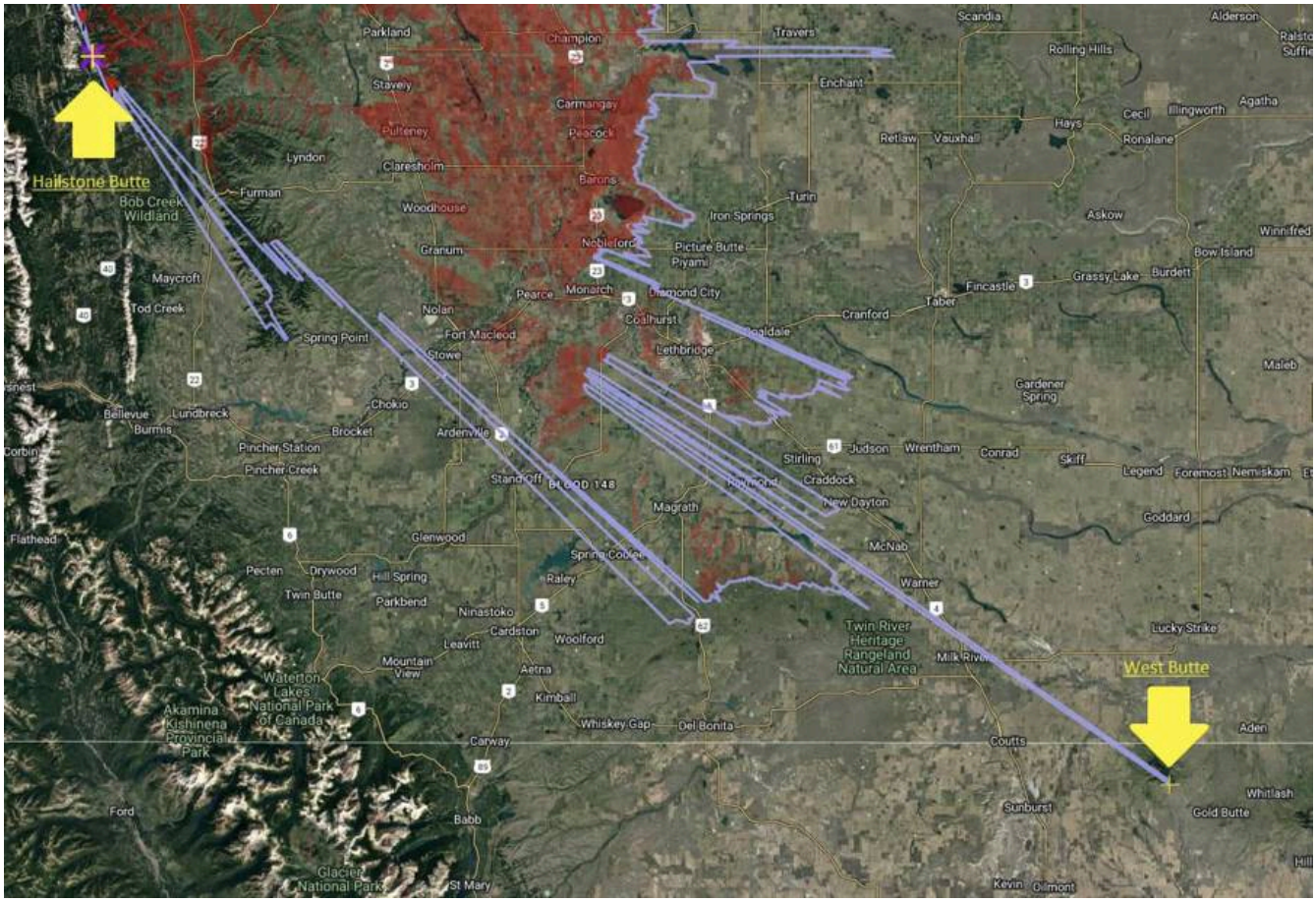
- **Device:** RAK4631 Core with RAK 5005-O Base Board
- **Firmware Version:** 2.1.17
- **Antenna:** 902-928 MHz 5.8 dBi Slinkdsco Outdoor

### Node B

- **Device:** RAK4631 Core with RAK 19003 Mini Base Board
- **Firmware Version:** 2.1.18
- **Antenna:** Standard LoRa 915 MHz 60mm 2dBi Omnidirectional








### Previous Record 166km

- **Range:** 166km (103 miles)
- **Record Holder:** *PuzzledPancake*
- **Source:** Meshtastic Discourse (archive)

### Modem Settings

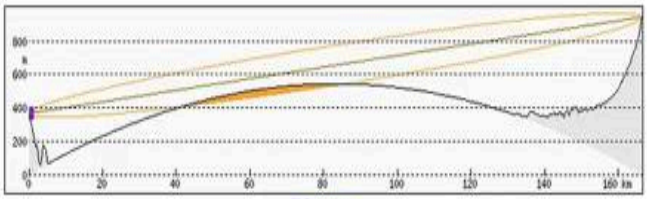
- **Frequency:** 868 MHz
- **Bandwidth:** 125
- **Spread Factor:** 12
- **Coding Rate:** 4/8

### Node A

- **Device:** LILYGO TTGO T-Beam w/ SX1262
- **Firmware Version:** 1.2
- **Antenna:** 868 MHz 5dBi Antenna

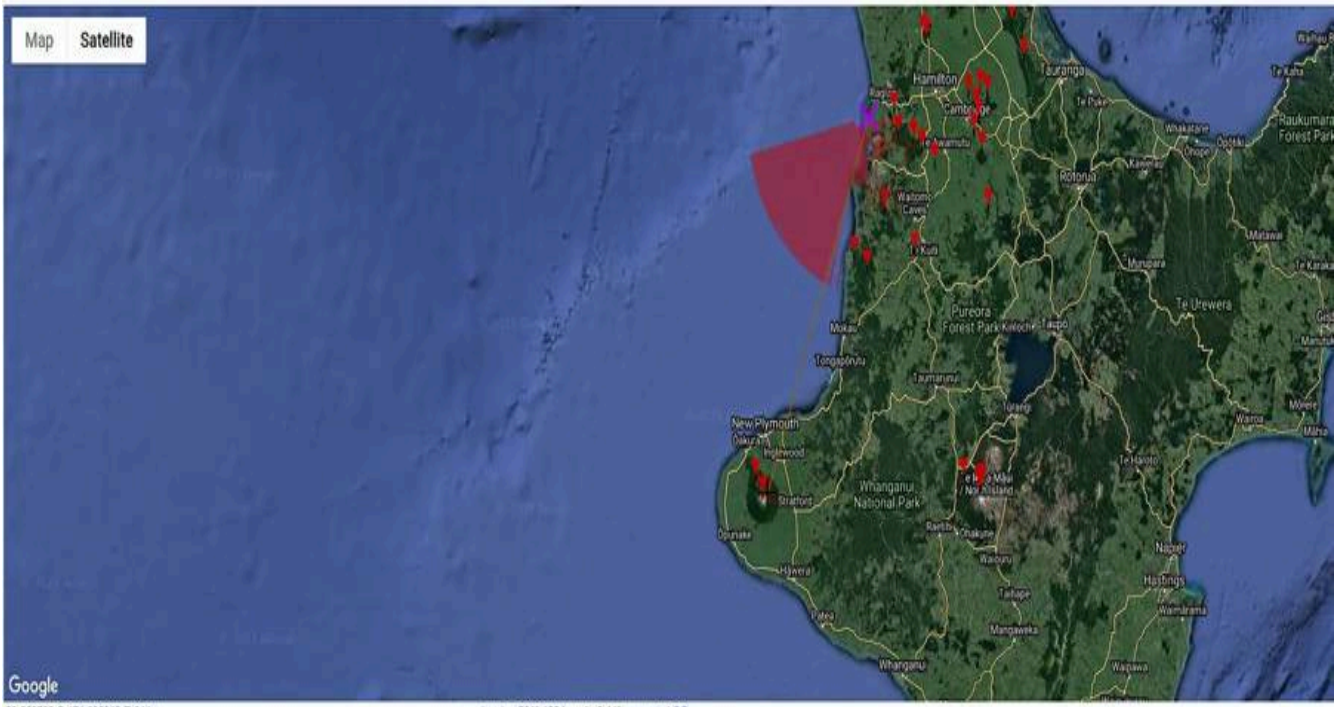
### Node B

- **Device:** LILYGO TTGO T-Beam w/ SX1262
- **Firmware Version:** 1.2
- **Antenna:** 868 MHz Vertical 6dBi




▼ Parameters

| | | |
|---|---|---|
| ☑ show scale | frequency (MHz, e.g. 5800) 865 | y range (e.g. -20,100) |
| ○ flat Earth<br>◉ curved Earth | refraction (e.g. 1.4) | far end elevation (use 99 for relative to sea level, +99 or +-99 for relative to ground) |
| ○ plate carée<br>◉ great circle | fixed exaggeration (e.g. 2) | Reset |




39.269758 S 174.095919 E 944m bearing 201° 166 km alt -0.44° compute LOS






### Resources Used

- http://www.heywhatsthat.com

**Ground to Air**

### Current Air Record: 206km

- **Range:** 206km (128 miles)
- **Record Holders:** *StarWatcher, CVR, rook, kboxlabs*
- **Source:** Meshtastic Discourse (archive)

### Modem Settings

Long_Fast Preset

- **Frequency:** 915 MHz
- **Bandwidth:** 250
- **Spread Factor:** 11
- **Coding Rate:** 4/8

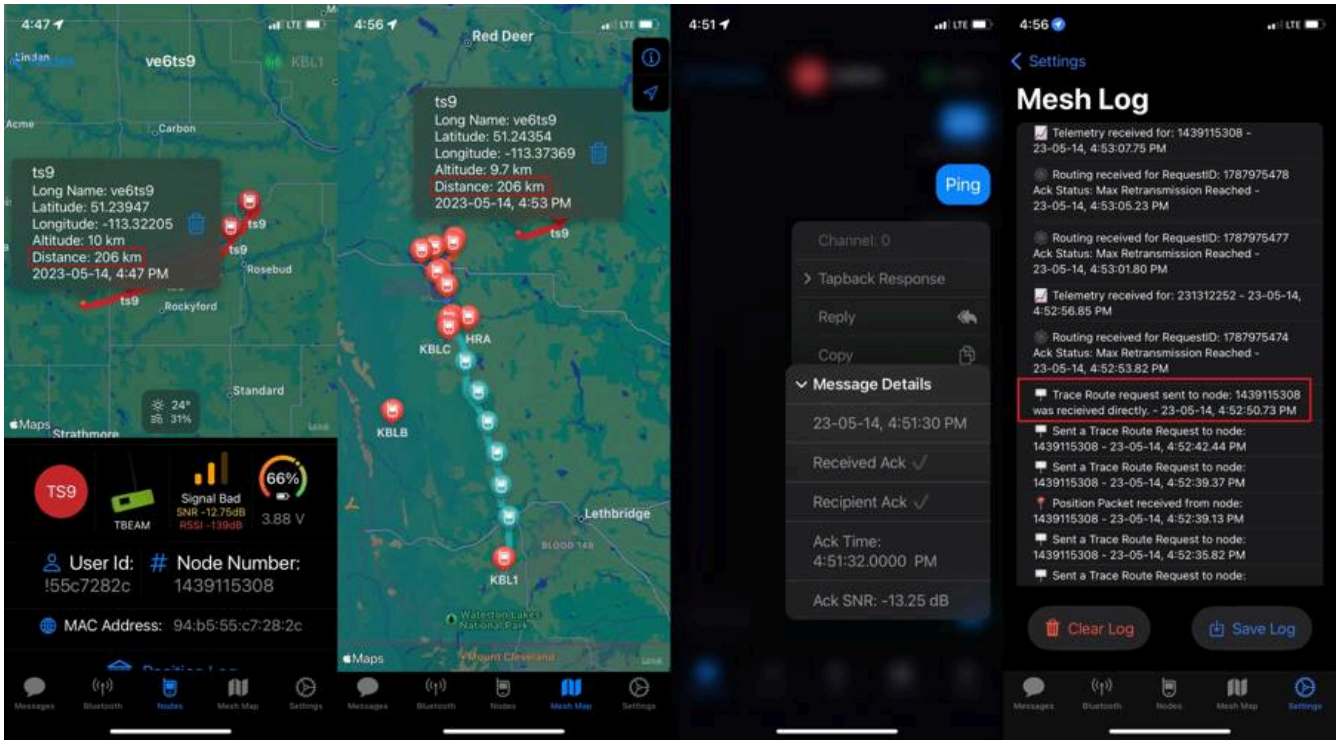
### Node A

- **Device:** LILYGO TTGO T-Beam
- **Firmware Version:** 2.1.10
- **Antenna:** Stock Antenna

### Node B

- **Device:** LILYGO TTGO T-Beam
- **Firmware Version:** 2.1.10 (modified to place GPS in flight mode)
- **Antenna:** Stock Antenna




ve6ts9
ts9
Long Name: ve6ts9
Latitude: 51.23947
Longitude: -113.32205
Altitude: 10 km
Distance: 206 km
2023-05-14, 4:47 PM
Rosebud
Rockyford
Standard
Strathmore
TS9
TBEAM
Signal Bad
SNR -12.75dB
RSSI -139dB
66%
3.88 V
User Id:
!55c7282c
Node Number:
1439115308
MAC Address: 94:b5:55:c7:28:2c
Red Deer
ts9
Long Name: ve6ts9
Latitude: 51.24354
Longitude: -113.37369
Altitude: 9.7 km
Distance: 206 km
2023-05-14, 4:53 PM
KBLC
HRA
KBLB
Lethbridge
KBL1
Maps
Ping
Channel: 0
Tapback Response
Reply
Copy
Message Details
23-05-14, 4:51:30 PM
Received Ack
Recipient Ack
Ack Time:
4:51:32.0000 PM
Ack SNR: -13.25 dB
Settings
Mesh Log
Telemetry received for: 1439115308 - 23-05-14, 4:53:07.75 PM
Routing received for RequestID: 1787975478 Ack Status: Max Retransmission Reached - 23-05-14, 4:53:05.23 PM
Routing received for RequestID: 1787975477 Ack Status: Max Retransmission Reached - 23-05-14, 4:53:01.80 PM
Telemetry received for: 231312252 - 23-05-14, 4:52:56.85 PM
Routing received for RequestID: 1787975474 Ack Status: Max Retransmission Reached - 23-05-14, 4:52:53.82 PM
Trace Route request sent to node: 1439115308 was recieived directly. - 23-05-14, 4:52:50.73 PM
Sent a Trace Route Request to node: 1439115308 - 23-05-14, 4:52:42.44 PM
Sent a Trace Route Request to node: 1439115308 - 23-05-14, 4:52:39.37 PM
Position Packet received from node: 1439115308 - 23-05-14, 4:52:39.13 PM
Sent a Trace Route Request to node: 1439115308 - 23-05-14, 4:52:35.82 PM
Sent a Trace Route Request to node:
Clear Log
Save Log

# Encryption: who can read what

From https://meshtastic.org/docs/overview/encryption/

Meshtastic encrypts the payload of every packet it sends over LoRa. Channel messages use the channel's key, direct messages use a key that only the two nodes share, and broadcasts can also carry a signature that shows which node sent them.

The packet header is never encrypted. Nodes need it to relay packets, including packets they can't decrypt. The rebroadcast mode controls whether a node relays those.

A node decrypts packets before passing them to a connected client over Bluetooth, serial, Wi-Fi, or Ethernet. For MQTT, Encryption Enabled sets whether the node uploads packets encrypted or decrypted.

## Channel messages

Each channel has its own pre-shared key (PSK). Meshtastic encrypts the channel's messages with AES-CTR using that key. A 16-byte key gives AES-128 and a 32-byte key gives AES-256. Anyone with the key can read every message on the channel.

The default primary channel uses the key `AQ==`, which is published. Anything sent on a channel with the default key can be read by anyone. To keep a channel private, set a new key and share it only with the people you want on that channel. Nodes that only have the default key can't read your private channel, and you can't read theirs unless you also keep a channel with the default key.

AES-CTR doesn't detect a message that was changed in transit. On firmware 2.8.1 and later, a channel can use AES-CCM instead by turning on Use AEAD. Each packet then carries a 12-byte tag, and nodes reject any packet that was altered or forged by someone without the key. Every node on the channel must turn it on, because nodes with the setting on and off can't read each other's messages on that channel.

## Direct messages

A direct message is encrypted with a key that only the sender and the recipient can compute. Each node has a public and private key pair. The sender combines its private key with the recipient's public key using X25519, and the recipient arrives at the same shared key from its side. That key encrypts the message with AES-256-CCM, so only the two nodes can read it, and the recipient knows the message came from the sender.

A node learns other nodes' public keys from the NodeInfo they broadcast, and it won't send a direct message to a node whose key it doesn't have. Traceroute, NodeInfo, position, and acknowledgment packets normally use the channel key, even when they're addressed to one node.

Direct messages aren't signed. The shared key already shows who sent them.

On firmware 2.8.1 and later, an acknowledgment for a direct message can carry a short proof, made with the shared key, that the recipient really received the message. Without it, anyone holding the channel key could forge an acknowledgment. The sending node checks the proof and reports the result to its client. A proof is attached only when both nodes have each other's public keys.

## Signed broadcasts

Broadcasts are encrypted with the channel key, so anyone with that key could otherwise send a broadcast that claims to come from your node. Nodes on firmware 2.8 and later sign their broadcasts with XEdDSA, using the same key pair they use for direct messages. Other nodes check the signature against the sender's public key and drop any broadcast whose signature doesn't match.

A signature adds 66 bytes, so a broadcast too large to carry one is sent unsigned. Nodes in licensed mode send everything unencrypted and sign every packet that fits instead. STM32WL-based nodes don't sign or check signatures.

The Packet Signature Policy sets what a node does with unsigned packets. The default accepts them, so the mesh keeps working with nodes that don't sign.

## Node identity

A node's number is a CRC-32 of its public key. Generating a new key gives the node a new number, and other nodes see it as a different node. Back up your keys before you erase and reflash a node; see Backup and Restore.

Nodes trust the first public key they hear for a node number, because there is no central authority to vouch for keys. Known Limitations explains what that allows and how signing narrows it.

## Admin messages

Remote administration uses the same encryption as direct messages. A node accepts admin messages only from nodes whose public keys are set as its admin keys, up to three. Each change must also carry a session passkey the node issued within the last 5 minutes, which limits how long a captured admin message could be replayed.

The older admin channel works only when Admin Channel Enabled is on.

## Is it as secure as Wi-Fi WPA3, HTTPS TLS 1.3, or Signal?

**No.** WPA3, TLS 1.3, and Signal also use AES, but they add protections that don't fit in LoRa packets or on small devices.

### Forward secrecy

Forward secrecy means that traffic captured today can't be decrypted later, even if a key leaks. Meshtastic doesn't provide it. Anyone who records channel traffic and later obtains the key can decrypt everything they recorded, which is known as harvest now, decrypt later. The key can leak when it's shared with the wrong person, when a node is stolen, or through a bug.

AES with a 256-bit key is considered resistant to quantum attack[1]. The X25519 key exchange used for direct messages and signatures isn't, because quantum-resistant schemes don't fit in LoRa packets or on Meshtastic hardware.

If you use private channels:

- **Do not configure private channels on unattended nodes.** Nodes relay traffic they can't decrypt, so an unattended router doesn't need the key, and physical access to a node makes the key easy to extract.
- Anything sent on a channel can be stored and decrypted later by anyone who gets the key, even after you delete the messages.
- Change channel keys from time to time.

### Integrity

Integrity means a message can't be changed without the key. AES-CTR doesn't provide it: someone who knows or guesses a message's content can alter it, or forge a new one, without knowing the key. Channels set to AES-CCM and signed broadcasts both detect this. Direct messages and admin messages are always protected.

### Authentication

Authentication means a node can prove who it is. Anyone with a channel's key can send an unsigned packet that claims to come from any node. With the default Packet Signature Policy, other nodes still accept it. `BALANCED` and `STRICT` reject unsigned broadcasts from nodes known to sign. Treat the sender of an unsigned channel message as a claim, not a proof.

Before sending anything sensitive in a direct message, confirm the recipient's public key with them some other way.

## Footnotes

1. On the quantum resistance of AES-256, see this Cryptography Stack Exchange question and the NIST Post-Quantum Cryptography FAQ, under "To protect against the threat of quantum computers, should we double the key length for AES now?" ↩

# Setting up a radio

## Getting started

From https://meshtastic.org/docs/getting-started/

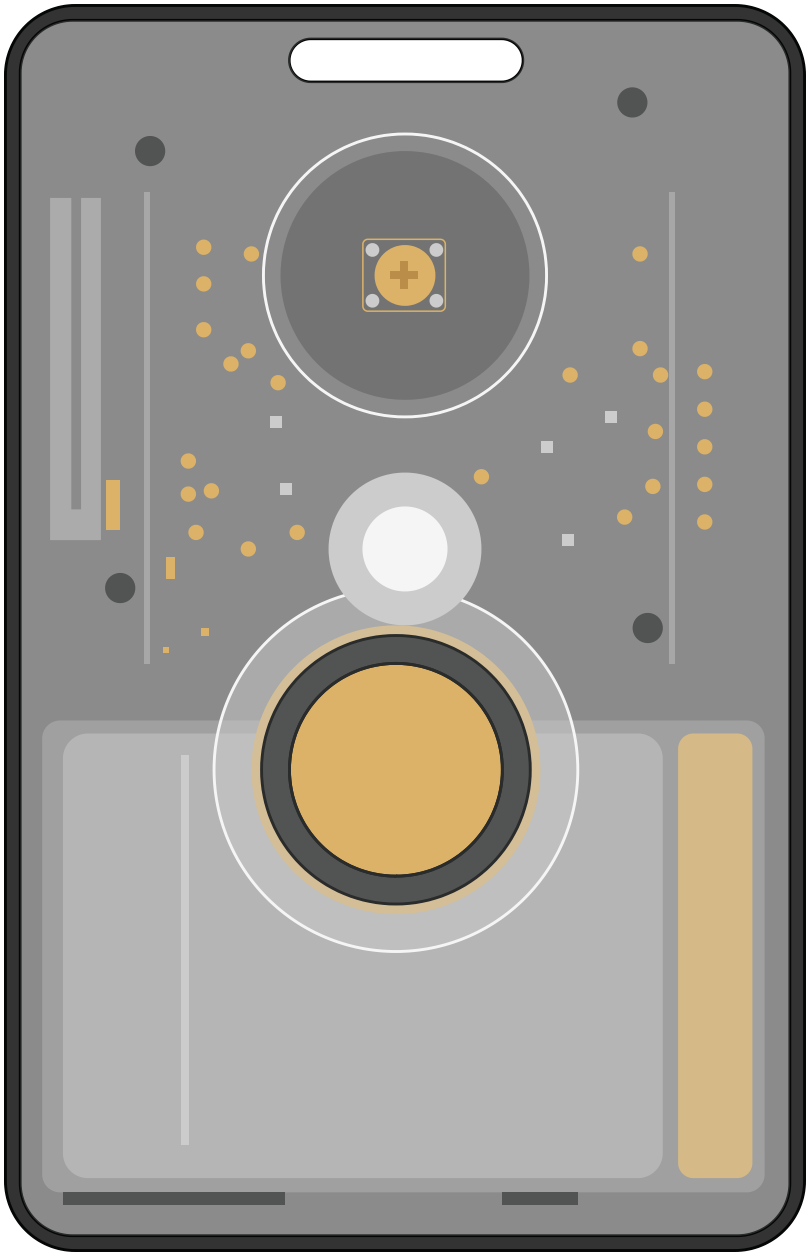
### How Meshtastic Works

Meshtastic creates a mesh network where devices communicate using LoRa radio. Connect your phone or computer to a radio via Bluetooth, Wi-Fi, or USB — and communicate with others across vast distances without any internet or cell service.

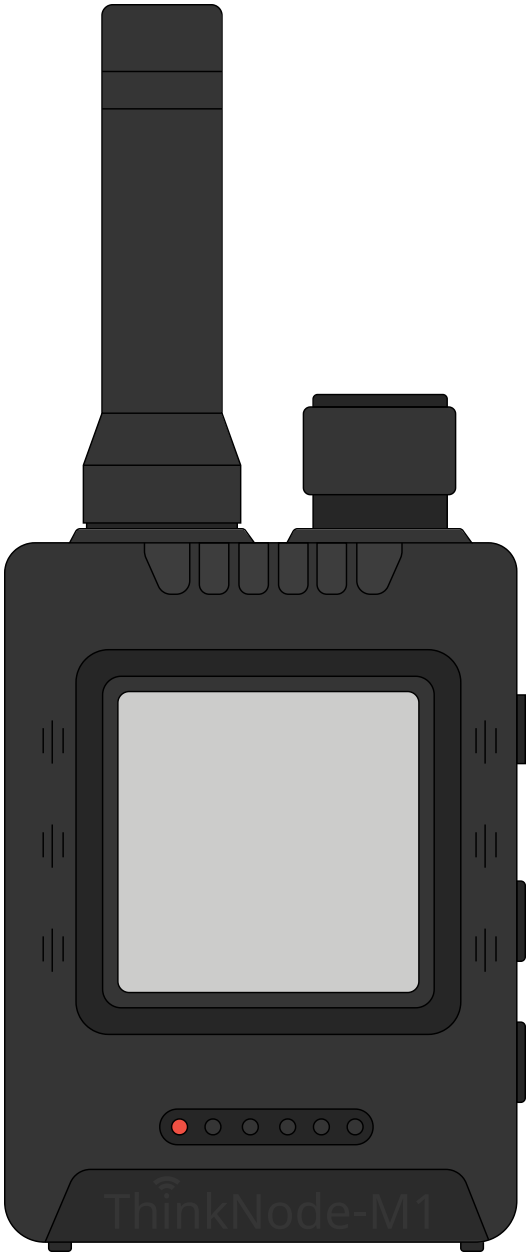




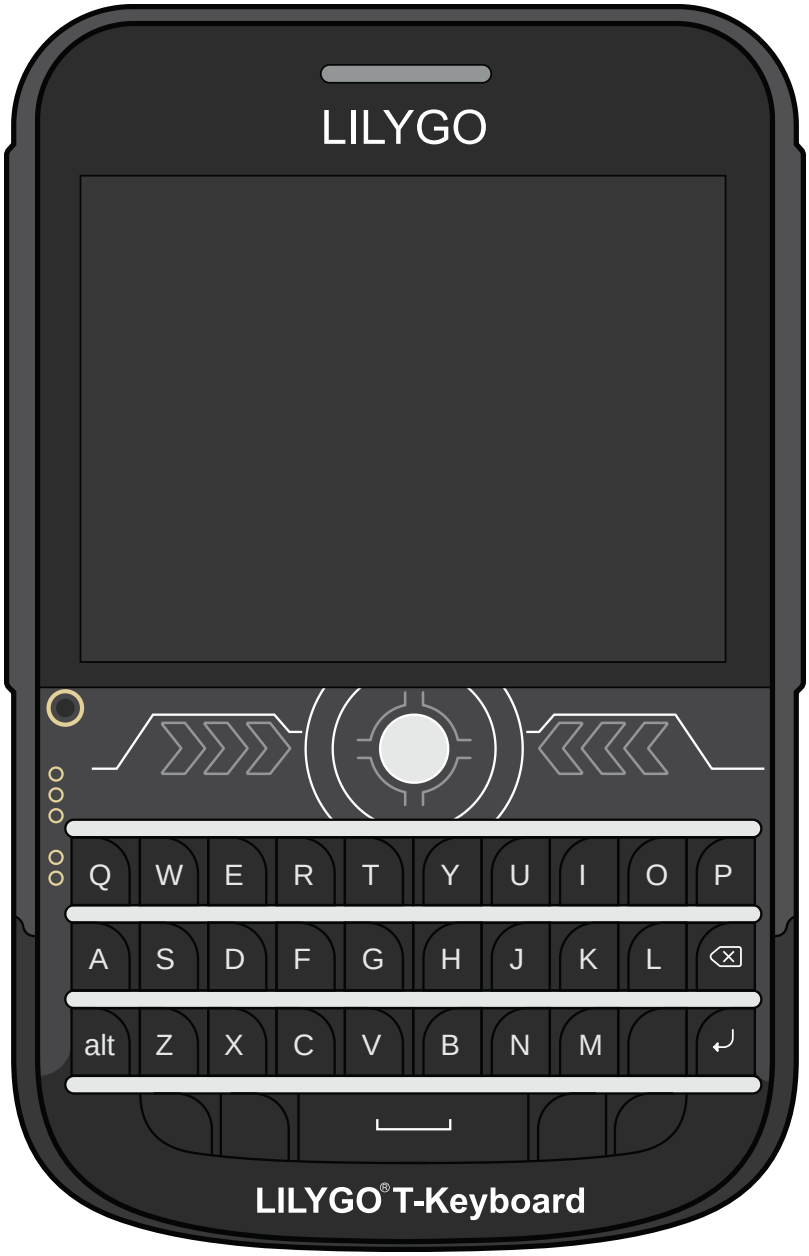
Client



Client



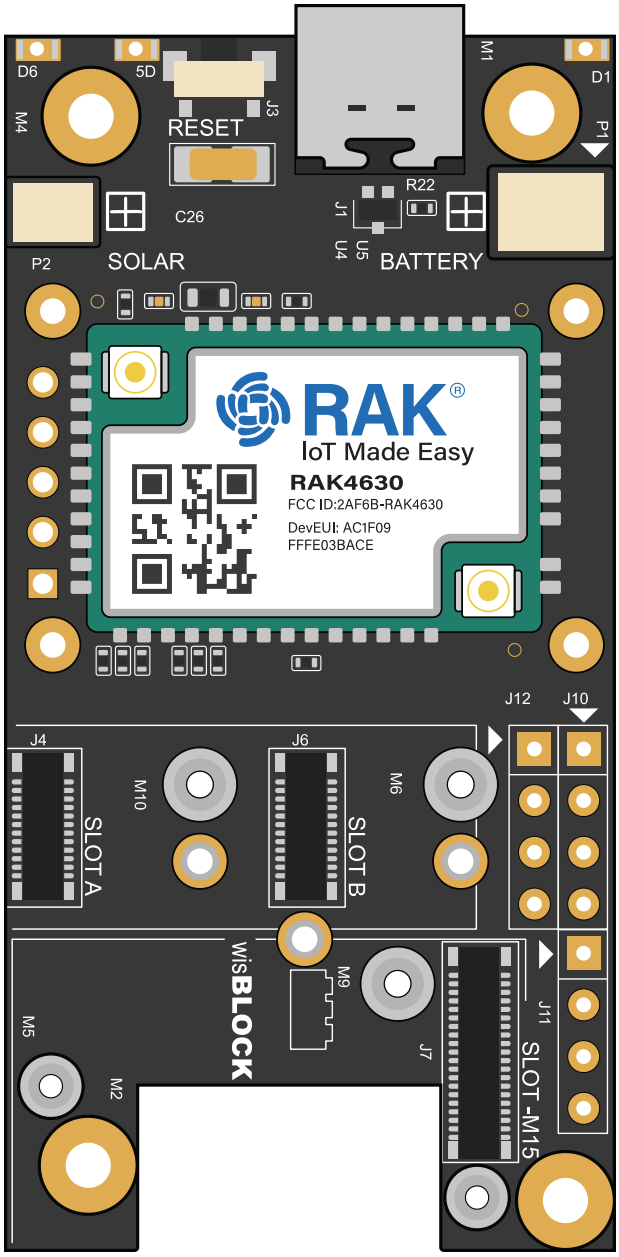

Client


LILYGO
Q W E R T Y U I O P
A S D F G H J K L
alt Z X C V B N M
LILYGO® T-Keyboard

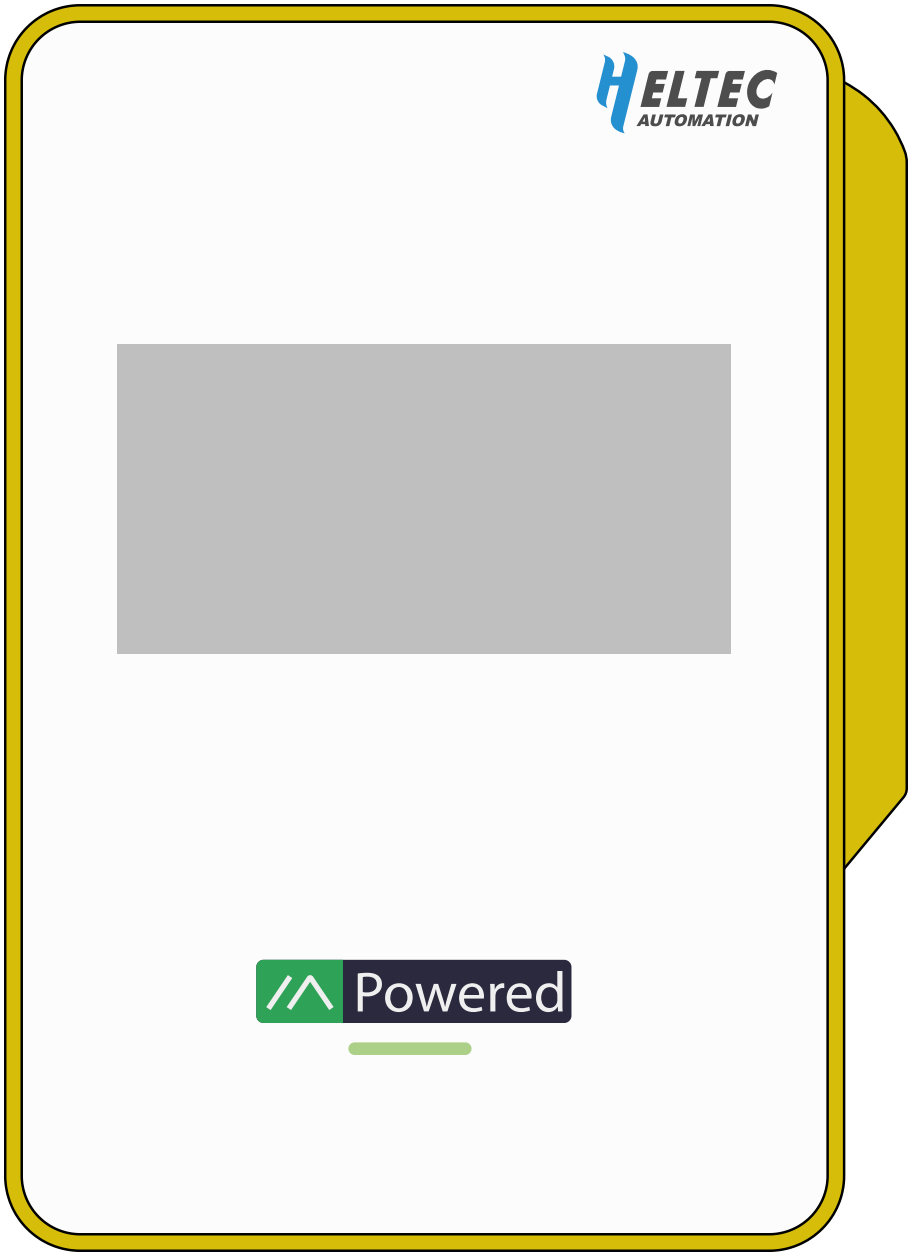
Client


WARNING: OPERATION WITHOUT AN ANTENNA
MAY PERMANENTLY DAMAGE THE DEVICE.
Station G2

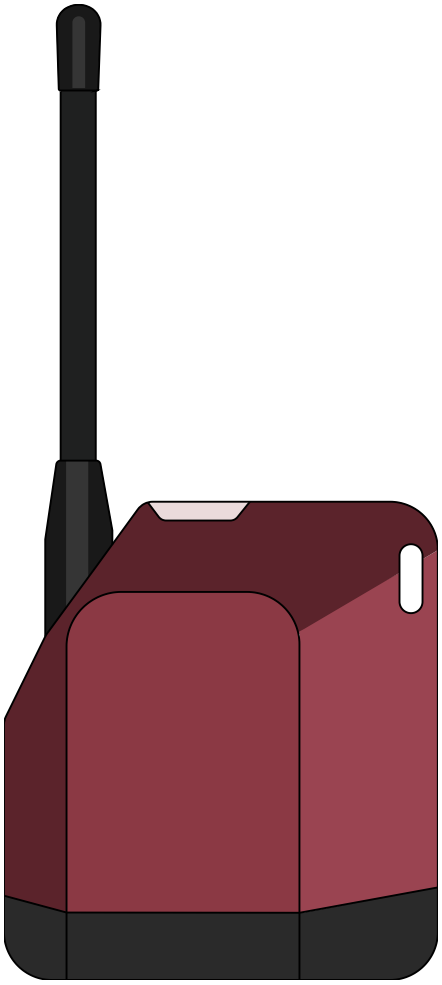
Router


D6
5D
M1
D1
M4
RESET
J3
P1
R22
C26
J1
U4
U5
P2
SOLAR
BATTERY
RAK®
IoT Made Easy
RAK4630
FCC ID:2AF6B-RAK4630
DevEUI: AC1F09
FFFE03BACE
J12
J10
J4
J6
M10
M6
SLOT A
SLOT B
WisBLOCK
M9
J7
J11
M5
M2
SLOT -M15

Client




Client



Client
meshtastic-cli

$ meshtastic --listen

Listening for messages...

LoRa
Bluetooth
WiFi
USB

## Supported Hardware

Before you begin, it's important to determine which kind of hardware you're using. Meshtastic works closely with our Partners and Backers who produce officially supported hardware. These devices are tested, documented, and recommended for the best Meshtastic experience.

There is also a wide range of community supported hardware available; however, these devices are not officially supported by Meshtastic. Please reach out to the community via Discord for assistance.

Maker hardware comes from small, independent hardware makers who want to build hardware for, and support, the Meshtastic project.

Below you'll find examples of supported hardware organized by MCU type. Once you've identified your device, we'll walk you through verifying your data cable, flashing the firmware, and connecting and configuring your device.

## ESP32

The ESP32 chip is equipped with both Wi-Fi and Bluetooth, making it ideal for devices that need web interface access or Wi-Fi-based configuration. ESP32-S3 variants offer improved performance.

### Partner Hardware

#### Seeed Studio

SenseCAP Indicator — 4" touchscreen driven by ESP32-S3 and RP2040 Dual-MCU



#### RAK Wireless

RAK3312 Core module — ESP32-S3-based WisBlock modular core



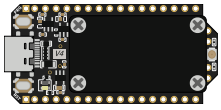
#### Elecrow

ThinkNode M2 — Portable ESP32-S3 device built for outdoor use



#### HELTEC®

LoRa32 V4 — ESP32-S3 board with built-in 0.96 inch OLED display



#### LILYGO®

T-Deck / T-Deck Plus / T-Deck Pro — Standalone devices with screen and keyboard



### Backer Hardware

#### B&Q Consulting

Station G2 — High power LoRa transceiver for licensed ham operation




#### M5Stack

Cardputer Mesh Kit — Pocket messaging terminal with keyboard, LoRa cap, and GNSS



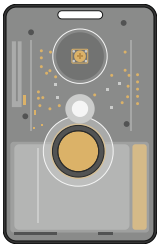
## nRF52

The nRF52 chip is much more power efficient than the ESP32 chip and easier to update via UF2 bootloader, but is only equipped with Bluetooth (no Wi-Fi). Ideal for battery-powered and solar deployments.

### Partner Hardware

#### Seeed Studio

Card Tracker T1000-E — IP65-rated card-sized tracker with GPS



#### RAK Wireless

WisMesh Tag — Portable location tracker with IP66 rating



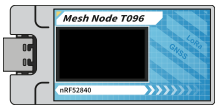
#### Elecrow

ThinkNode M3 — nRF52840 with LR1110 radio and GPS




#### HELTEC®

Mesh Node T096 — Ultra-low-power nRF52840 node with color TFT display and GNSS




#### LILYGO®

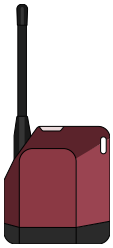
T-Echo — All-in-one unit with E-Ink screen, GPS, and battery in injection-molded case



### Backer Hardware

#### muzi WORKS

R1 Neo — Custom-designed nRF52840 device with GPS



## RP2040/RP2350

The RP2040 and RP2350 are dual-core chips developed by Raspberry Pi. Cost-effective options for DIY projects.

### Partner Hardware

#### RAK Wireless

- RAK11310 Core module — RP2040-based WisBlock modular core with SX1262

### Community Supported

- Raspberry Pi Pico + Waveshare LoRa Module (Note: **Bluetooth on the Pico W is not yet supported by Meshtastic**)

## STM32

The STM32WL from STMicroelectronics puts an Arm Cortex-M4 and a Semtech SX126x radio on one chip — the lowest part count and power draw of any Meshtastic target, with no Wi-Fi or Bluetooth.

Due to low flash memory and RAM, a number of features have been excluded from STM32WL firmware by default (specific variants may re-enable specific features).

▼ Features excluded from STM32WL firmware by default

- ATAK/TAK forwarding
- Codec2 audio
- MQTT
- Remote Hardware module
- Waypoints
- On-device display and menu UI
- Input peripherals (rotary encoders, trackballs, buttons/keyboards) and buzzer
- Power monitoring and power-metrics telemetry
- Timezone database
- RTTTL ringtones
- Signed-packet authenticity policy

### Community Supported

- RAK3172/RAK3372 Core module — STM32WLE5 WisBlock module for embedding into custom hardware
- Seeed Wio-E5 — STM32WLE5 wireless module, mini dev board, and dev kit
- STMicroelectronics Nucleo-WL55JC — Official STM32WL evaluation board with onboard ST-LINK



info

> If your device is not listed above, please review our supported devices to determine which MCU your device has or contact us in Discord with any questions.

## Verify Data Cable



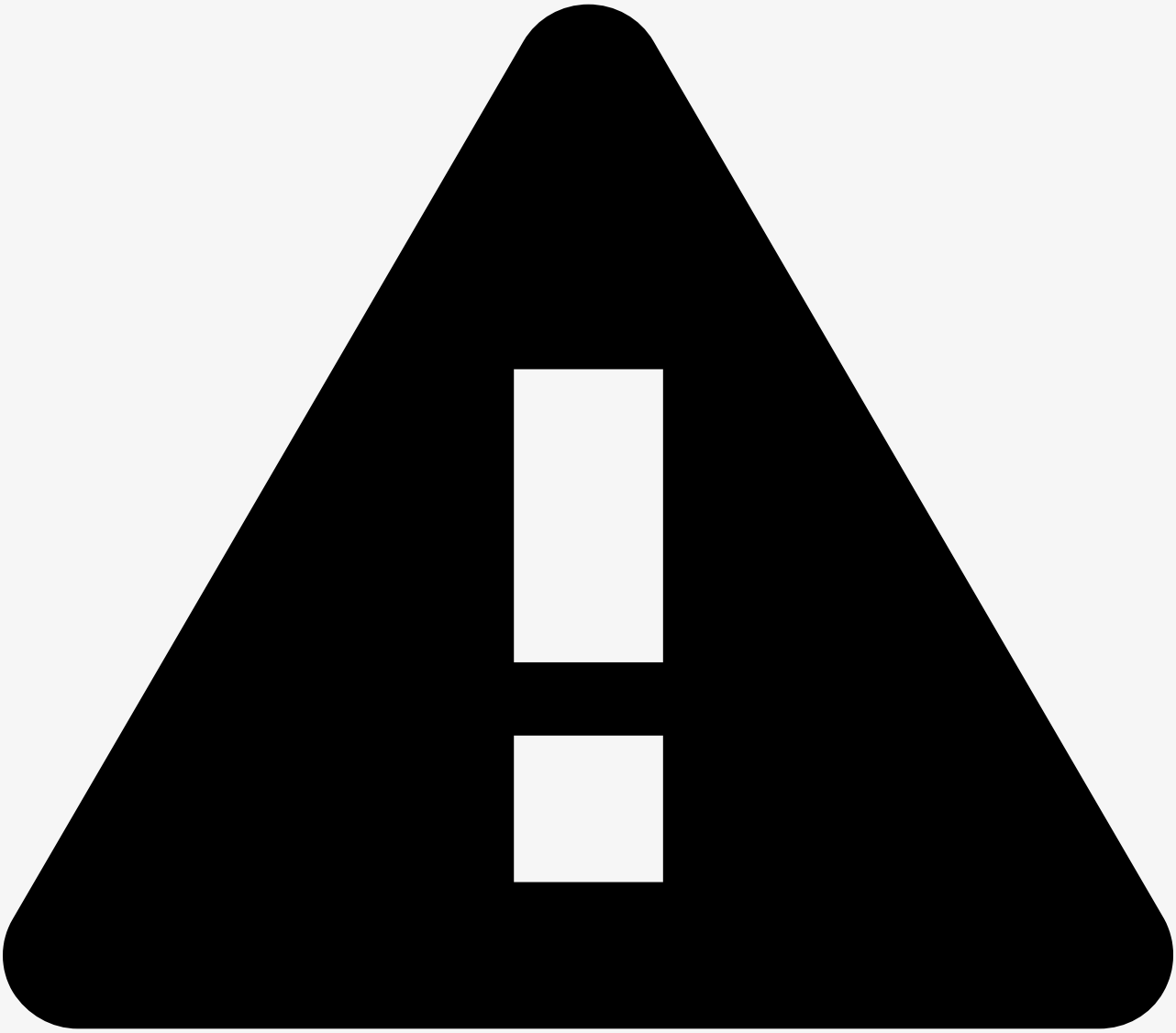
STOP! Put The Power Cable Down!

> Never power on the radio without attaching an antenna as doing so could damage the radio chip!

Prior to connecting your Meshtastic device to the computer, you should perform the following basic checks.

Some cables only provide *charging*, verify that your cable is also capable of *transferring data* before proceeding. To check if your cable can also transfer data, try connecting it to another device (like a phone) and see if you can copy a file to or from it. If the file transfer works, then your cable is also able to transfer data and you can continue.

## Install Serial Drivers



caution

nRF52/RP2040/RP2350 devices typically do not require serial drivers. They use the UF2 bootloader which makes the devices appear as flash drives. Do *NOT* download the USB device drivers unless required to install UF2 support.

If you require serial drivers installed on your computer, please choose one of the options below and install it before continuing.

Install ESP32 Drivers

Install nRF52/RP2040/RP2350 Drivers

## Flash Firmware

After completing the previous steps, you can now flash the Meshtastic firmware onto your device. To proceed, select the appropriate device type for your device.

Flash ESP32 Firmware

Flash nRF52/RP2040/RP2350 Firmware

Flash STM32 Firmware

## Connect and Configure Device

After flashing the Meshtastic firmware onto your device, you can now move on to initial configuration.

Connect and Configure Device

# Putting Meshtastic on an ESP32 radio (the web flasher)

From https://meshtastic.org/docs/getting-started/flashing-firmware/esp32/web-flasher/

## Flash Device

1. Plug in your device
2. Visit flasher.meshtastic.org **requires Chrome or Edge browser*
3. Follow the instructions

### Connect and Configure Device

After flashing the Meshtastic firmware to the device, you can proceed with the initial configuration.

Connect and Configure Device

# Putting Meshtastic on an nRF52 radio (drag and drop)

From https://meshtastic.org/docs/getting-started/flashing-firmware/nrf52/drag-n-drop/

## Flash Firmware



info

> You may now use the Meshtastic Web Flasher to download and copy firmware to your nRF52, RP2040, or RP2350-based devices. Alternatively, follow the instructions below to download and install firmware.

If you have just performed a Factory Erase, your device may automatically startup in bootloader mode, and you may skip to "Copy the appropriate firmware..." for your respective device below:

### nRF52

1. Download and unzip the latest firmware from Meshtastic Downloads.
2. Connect your device to your computer with a USB data cable.
3. Put the device into bootloader mode. On most devices, double-press the reset button.
4. Notice a new drive will be mounted on your computer (Windows, Mac, Linux, or Android).
5. Open this drive and you should see three files: `CURRENT.UF2`, `INDEX.HTM`, and `INFO_UF2.TXT`.
6. Copy the appropriate firmware file (`firmware-DEVICE_NAME-X.X.X-xxxxxxx.uf2`) from the release onto this drive.
7. Once the file has finished copying onto the drive, the device will reboot and install the Meshtastic firmware.

### RP2040

1. Download and unzip the latest firmware from Meshtastic Downloads.
2. Press the BOOTSEL button and while keeping it pressed, connect the device to your computer via a USB cable.
3. Notice a new drive will be mounted on your computer (Windows, Mac, Linux, or Android).
4. Open this drive and you should see two files: `INDEX.HTM` and `INFO_UF2.TXT`.
5. Copy the appropriate firmware file (`firmware-DEVICE_NAME-X.X.X-xxxxxxx.uf2`) from the release onto this drive.
6. Once the file has finished copying onto the drive, the device will reboot and install the Meshtastic firmware.

## Connect and Configure Device

After flashing the Meshtastic firmware to the device, you can proceed with the initial configuration.

Connect and Configure Device

### Potential Flashing Edge Cases



info

Before flashing confirm that you have RAK4631 and not a RAK4631-R If this is not the case, fear not. The hardware is identical but requires changing the bootloader.



info

Previous versions of the Meshtastic firmware may save stale data, causing devices to get stuck in a crash loop during startup. If you experience issues when upgrading your nRF52 device from a previous version of Meshtastic, you may need to perform a full factory reset of the internal flash memory.

Follow the guide to factory erase your nRF52 device before continuing to flash firmware.

# First settings

From https://meshtastic.org/docs/getting-started/initial-config/

## Supported Clients per Connection Type

Depending on your connection, some configuration options are not fully supported. Find out which client is best for your type of connection.

**Serial Bluetooth Network**

### Serial

- Python CLI
- Web Client
- Android App

**Bluetooth Network**

### Bluetooth

- Android App
- Web Client

**Network**

### Network



info

Connecting over network is only supported on ESP32 devices.

- Web Client
- Android App
- iOS App
- Python CLI

# Set Regional Settings

In order to start communicating over the mesh, you must set your region. This setting controls which frequency range your device uses and should be set according to your regional location.

**Android Apple CLI Web**

## Android

1. Follow the setup and usage instructions for Meshtastic Android.
2. Open the app, connect to the device from your phone over USB Serial or Bluetooth.
3. Once paired and connected, Click **SET YOUR REGION** under the connected device card. or navigate to **Settings > LoRa**.
4. Select the region from the list according to your regional location. Click **Send**

**Apple CLI Web**

## Apple



info

Configuration of Region, Modem Preset and Hop Limit is available on iOS, iPadOS and macOS at Settings > Radio Configuration > LoRa.

**CLI Web**

## CLI

1. Install Meshtastic PythonCLI

```
pip3 install --upgrade pytap2
pip3 install --upgrade meshtastic
```

2. Run the following command, replacing `<REGION-CODE>` with the region code listed above according to your regional location.

```
meshtastic --set lora.region <REGION-CODE>
```

## Region Codes

| Region Code | Description | Frequency Range (MHz) | Duty Cycle (%) | Power Limit (dBm) |
|---|---|---|---|---|
| UNSET | Unset | N/A | N/A | N/A |
| US | United States | 902.0 - 928.0 | 100 | 30 |
| EU_433 | European Union 433 MHz | 433.0 - 434.0 | 10 | 10 |
| EU_868 | European Union 868 MHz | 869.4 - 869.65 | 10 | 27 |
| EU_866 | European Union 866 MHz | 865.6 - 867.6 | 2.5 / 10 | 27 |
| EU_N_868 | European Union 868 MHz (Narrow) | 869.4 - 869.65 | 10 | 27 |
| CN | China | 470.0 - 510.0 | 100 | 19 |
| JP | Japan | 920.5 - 923.5 | 100 | 13 |
| ANZ | Australia & New Zealand | 915.0 - 928.0 | 100 | 30 |
| ANZ_433 | Australia & New Zealand | 433.05 - 434.79 | 100 | 14 |
| KR | Korea | 920.0 - 923.0 | 100 | 23 |
| TW | Taiwan | 920.0 - 925.0 | 100 | 27 |
| RU | Russia | 868.7 - 869.2 | 100 | 20 |
| IN | India | 865.0 - 867.0 | 100 | 30 |
| NZ_865 | New Zealand 865 MHz | 864.0 - 868.0 | 100 | 36 |
| TH | Thailand | 920.0 - 925.0 | 10 | 27 |
| UA_433 | Ukraine 433 MHz | 433.0 - 434.7 | 10 | 10 |
| MY_433 | Malaysia 433 MHz | 433.0 - 435.0 | 100 | 20 |
| MY_919 | Malaysia 919 MHz | 919.0 - 924.0 | 100 | 27 |
| SG_923 | Singapore 923 MHz | 917.0 - 925.0 | 100 | 20 |
| KZ_433 | Kazakhstan 433 MHz | 433.075 - 434.775 | 100 | 10 |
| KZ_863 | Kazakhstan 863 MHz | 863.0 - 868.0 | 100 | 30 |
| BR_902 | Brazil 902 MHz | 902.0 - 907.5 | 100 | 30 |

| Region Code | Description | Frequency Range (MHz) | Duty Cycle (%) | Power Limit (dBm) |
|---|---|---|---|---|
| PH_433 | Philippines 433 MHz | 433.0 - 434.7 | 100 | 10 |
| PH_868 | Philippines 868 MHz | 868.0 - 869.4 | 100 | 14 |
| PH_915 | Philippines 915 MHz | 915.0 - 918.0 | 100 | 24 |
| NP_865 | Nepal 865 MHz | 865.0 - 868.0 | 100 | 30 |
| ITU1_2M | ITU Region 1 Amateur 2m | 144.0 - 146.0 | 100 | 30 |
| ITU2_2M | ITU Region 2 Amateur 2m | 144.0 - 148.0 | 100 | 30 |
| ITU3_2M | ITU Region 3 Amateur 2m | 144.0 - 148.0 | 100 | 30 |
| ITU2_125CM | ITU Region 2 Amateur 1.25m | 220.0 - 225.0 | 100 | 30 |
| ITU1_70CM | ITU Region 1 Amateur 70cm | 430.0 - 440.0 | 100 | 30 |
| ITU2_70CM | ITU Region 2 Amateur 70cm | 420.0 - 450.0 | 100 | 30 |
| ITU3_70CM | ITU Region 3 Amateur 70cm | 430.0 - 450.0 | 100 | 30 |
| LORA_24 | 2.4 GHz band worldwide | 2400.0 - 2483.5 | 100 | 10 |

`EU_433` and `EU_868` are limited to a 10% duty cycle, calculated every minute over a rolling hour. A node that reaches the limit stops transmitting until it's allowed again. `EU_866` is limited to 2.5%, or 10% for a node in the `ROUTER` or `ROUTER_LATE` role.

`EU_868`, `EU_866`, and `EU_N_868` cover the same European band with different channel plans, and each accepts only its own presets. Selecting a preset that belongs to one of the others switches the region to it.

The `ITU1_*`, `ITU2_*`, and `ITU3_*` regions are amateur radio allocations and can only be selected in licensed mode. **Do not use them without an amateur radio license.** Review the privileges and restrictions of operating under an amateur license first: encryption is not permitted, and you must transmit your call sign.

The listed power limit is the ceiling the firmware applies, not your national limit. Some countries do not allocate the full range shown, for example 220–222 MHz in the USA and Canada. Check your national band plan before transmitting.

Refer to LoRa Region by Country for a more comprehensive list.

## Continue Configuration

Now that you have set the LoRa region on your device, you can continue with configuring any additional configs to suit your needs.

Setting the region leaves your node on a default modem preset, and that default is not always what the mesh near you runs. Many communities have moved to a faster preset, and some use their own channel. A node whose settings do not match hears nothing from them. Joining your local mesh covers how to find the settings in use around you and join.

Joining a Mesh

Device Configuration

# Joining the local mesh

From https://meshtastic.org/docs/getting-started/joining-your-local-mesh/

Two nodes reach each other only when they share their LoRa settings. The defaults are where most nodes start, and in many areas enough nodes stay there that the defaults act as a public mesh anyone can join. That is a convention rather than a guarantee, so a node on the defaults may find a busy mesh, or may hear nothing at all.

This page covers how to find the settings in use around you, and how to join once you know them.

## What the defaults are

Three settings decide who a node can hear.

| Setting | Default | Effect |
|---|---|---|
| Region | Unset until you choose one | The frequency band the node is legally allowed to use |
| Modem preset | Long Fast, or Long Turbo when the region is first set to US on the device screen | Range against speed, by setting bandwidth, spreading factor, and coding rate together |
| Primary channel | The default public channel | The key that decrypts traffic, and the frequency slot |

A modem preset bundles bandwidth, spreading factor, and coding rate into one named choice, balancing how far a message carries against how long it occupies the air. Presets exist so a community can agree on a single setting instead of three, and the default suits most areas. Setting the three individually is still available by turning off **Use Preset**, covered in the LoRa configuration.

Those same three parameters decide what the radio can demodulate, so the preset also decides who a node can hear. You choose your preset, and everyone else chooses theirs, and you reach only the ones whose choice matches yours. Two nodes on different presets don't have a slower link between them, they have no link at all, at any distance.

Meshtastic defines the presets so these three values don't have to be dialed in by hand. The rule underneath is LoRa's: two radios only hear each other when all three match, which is why local communities tend to settle on one preset together.

The preset also decides where the node listens. Where the primary channel name is unchanged, the preset name picks the frequency slot, so two presets put a node on two different frequencies. In the US region, Long Fast listens on slot 20 at 906.875 MHz while Long Turbo listens on slot 14 at 908.750 MHz.

## Why the defaults may not match your area

Three things commonly separate a new node from the mesh near it.

Many communities have outgrown Long Fast. Congestion on a shared, duty-cycled medium gets worse as a mesh grows, so dense metropolitan meshes have been moving to faster presets for years. The reasoning is covered in Is LongFast Holding Your Mesh Back?.

The default itself has moved. Setting the region to US on the device screen for the first time starts that node on Long Turbo, while setting the same region from a client or the CLI leaves it on Long Fast. Two nodes set up on the same day through different paths can therefore sit nearly 2 MHz apart, and a new node can hear nothing while an established local mesh carries on without it.

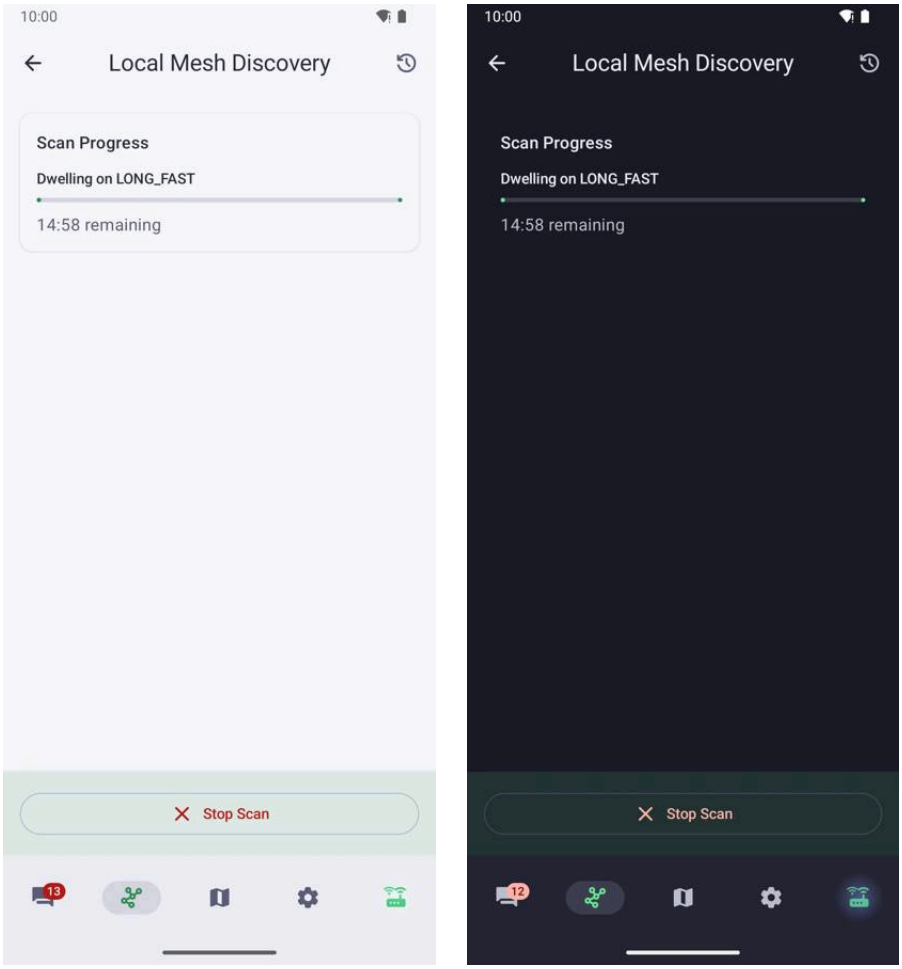
Some meshes run their own settings entirely. Events frequently adopt a preset for the duration, and some communities use a custom channel name or key so their traffic is not on the public channel.

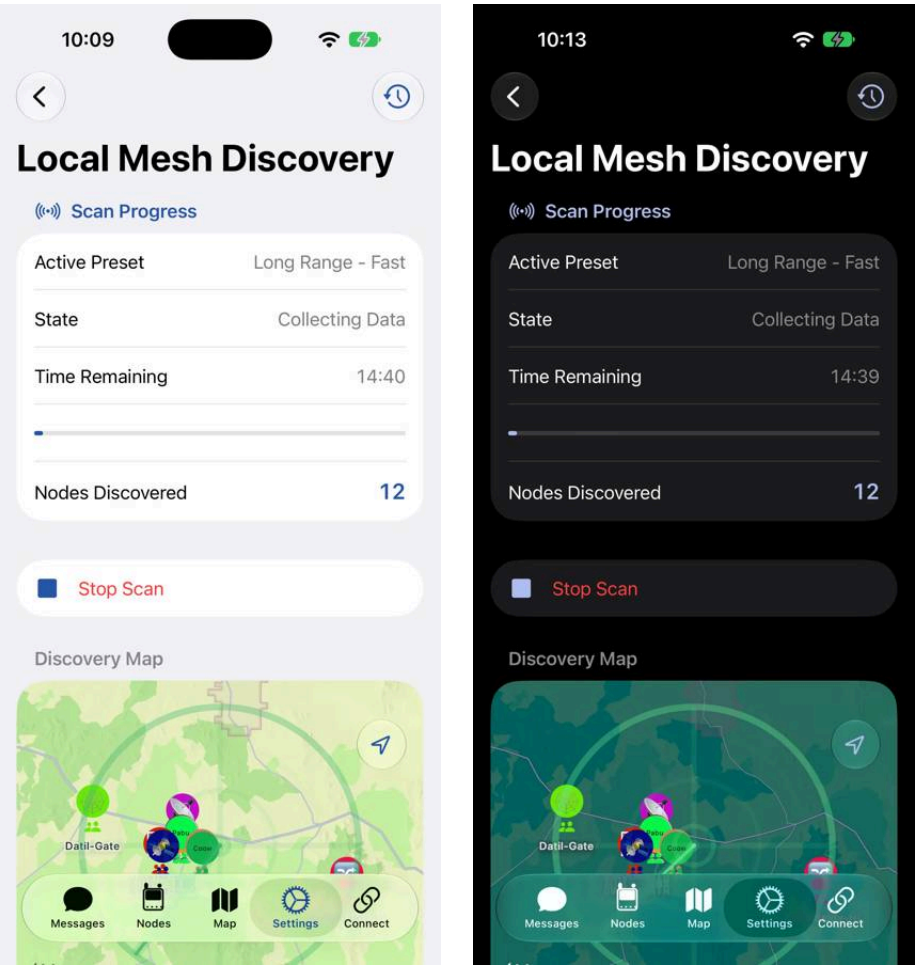
What silence means depends on what you already know. If nodes are known to be active near you, from a local group or from a previous setup that worked, hearing nothing points at settings rather than range. If nobody has confirmed a mesh in your area, silence on its own says very little, and the question becomes whether anyone is there at all. Either way the next step is the same.

## Finding the settings in use near you

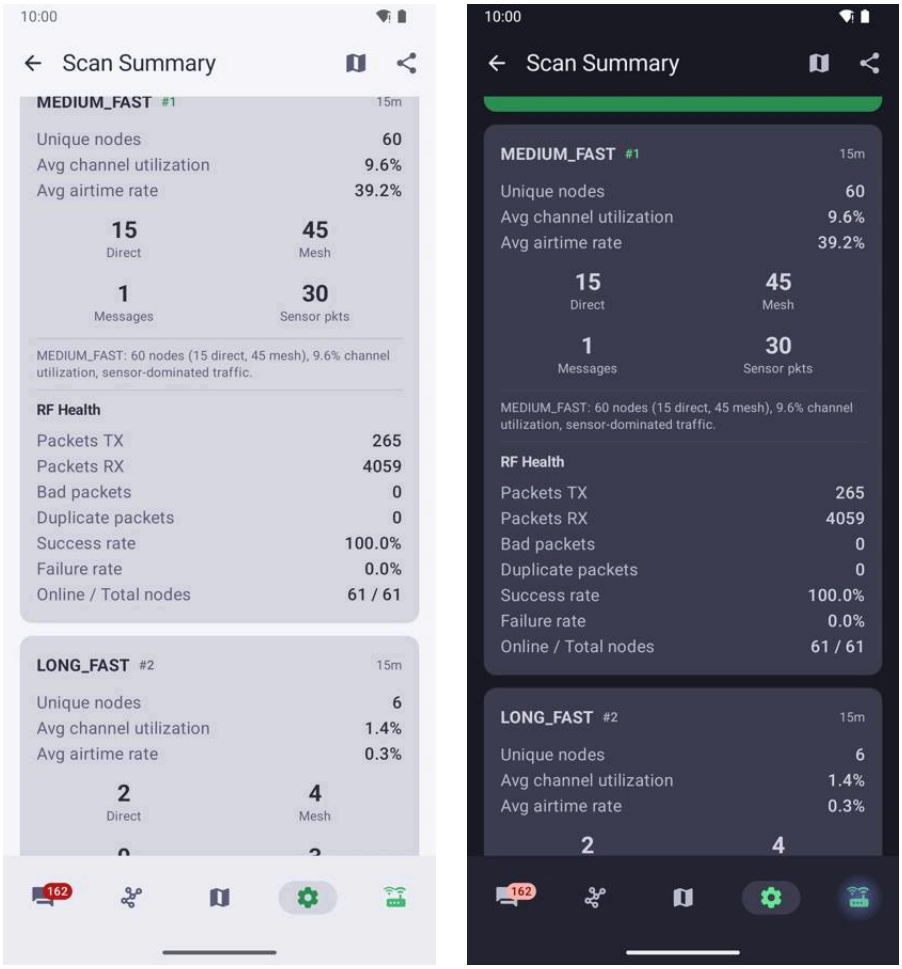
### Scan for them

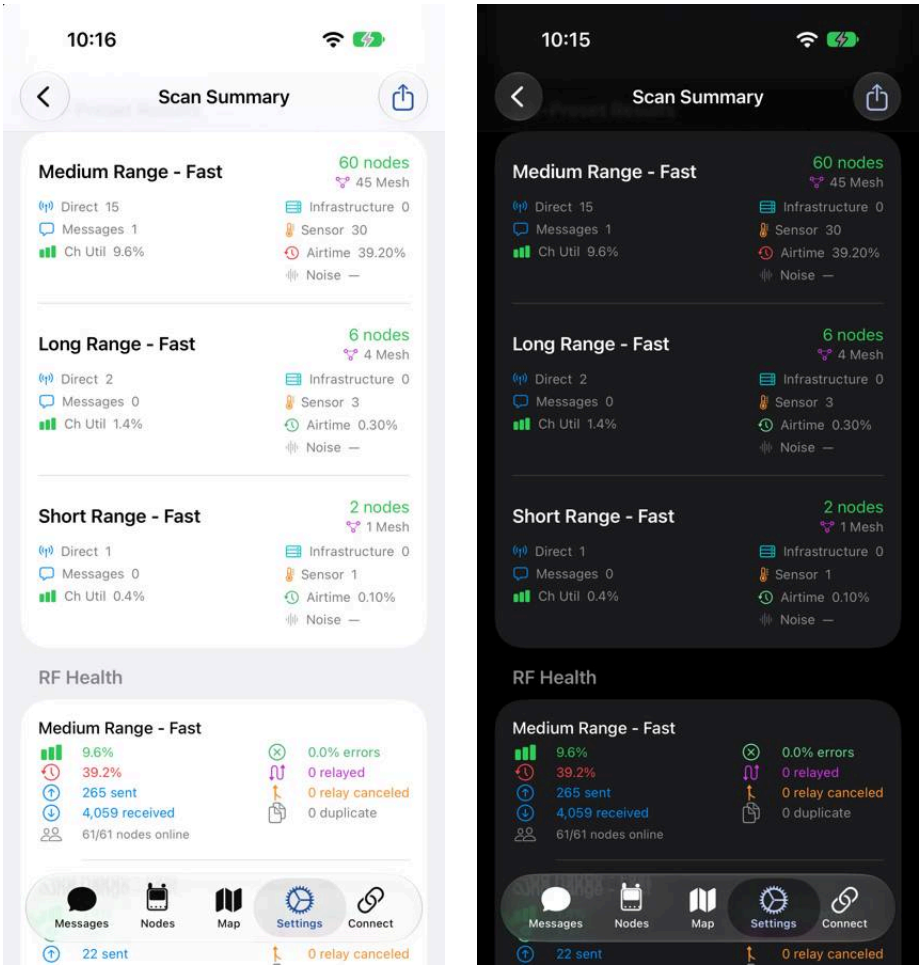
Local Mesh Discovery cycles the radio through a set of presets, listens on each for a dwell time you choose, and ranks them by what it hears. It is available in the Android and Apple clients.


10:00
Local Mesh Discovery
Scan Progress
Dwelling on LONG_FAST
14:58 remaining
Stop Scan
13
10:00
Local Mesh Discovery
Scan Progress
Dwelling on LONG_FAST
14:58 remaining
Stop Scan
12




A scan reports what it heard on each preset, ranked by activity. Here the same area carries 60 nodes on Medium Fast and six on Long Fast, which is what a node still sitting on the default looks like in a community that has moved.


10:00
Scan Summary
MEDIUM_FAST #1
15m
Unique nodes 60
Avg channel utilization 9.6%
Avg airtime rate 39.2%
15 Direct
45 Mesh
1 Messages
30 Sensor pkts
MEDIUM_FAST: 60 nodes (15 direct, 45 mesh), 9.6% channel utilization, sensor-dominated traffic.
RF Health
Packets TX 265
Packets RX 4059
Bad packets 0
Duplicate packets 0
Success rate 100.0%
Failure rate 0.0%
Online / Total nodes 61 / 61
LONG_FAST #2
15m
Unique nodes 6
Avg channel utilization 1.4%
Avg airtime rate 0.3%
2 Direct
4 Mesh
162




A preset scan listens on the default frequency slot with the default public channel, which is what makes one scan comparable across presets. A mesh whose primary channel carries a custom name sits on a different frequency slot, and one with a custom key cannot be decoded, so neither appears in a scan. An empty result means nothing is running on the public channel nearby, not that nothing is running.

### Ask the local community

Regional groups publish their settings to people joining, which is how to find a mesh a scan cannot see. The local groups directory lists Discord servers, Telegram groups, and websites by region.

## Joining once you know the settings

Set the region first, then match the modem preset. Both live in the LoRa configuration. Changing the preset takes effect immediately and does not require re-pairing.

Where a mesh uses its own channel, add it as described in channel configuration. The fastest route is a channel URL or QR code shared by the group, which carries the channel name, key, and LoRa settings together.

## Keeping a private channel on the public mesh

Wanting private conversations does not mean leaving the mesh. Matching the modem preset is what puts a node on the same mesh as everyone else; holding a channel's key is what lets it read that channel. Those are separate, so a node can carry the public channel and a private one at the same time.

The trap is renaming the primary channel. Channel 0 feeds the frequency slot, so giving it a private name moves the node to a different frequency and leaves it alone on a mesh of one. That is the most common way a node ends up unreachable while looking correctly configured.

Add private groups as secondary channels on 1 through 7 instead. They share the primary channel's LoRa settings, carry their own keys, and leave the frequency slot untouched, so the node stays reachable on the public mesh while those conversations stay unreadable to it. Channel configuration covers adding them and sharing the key.

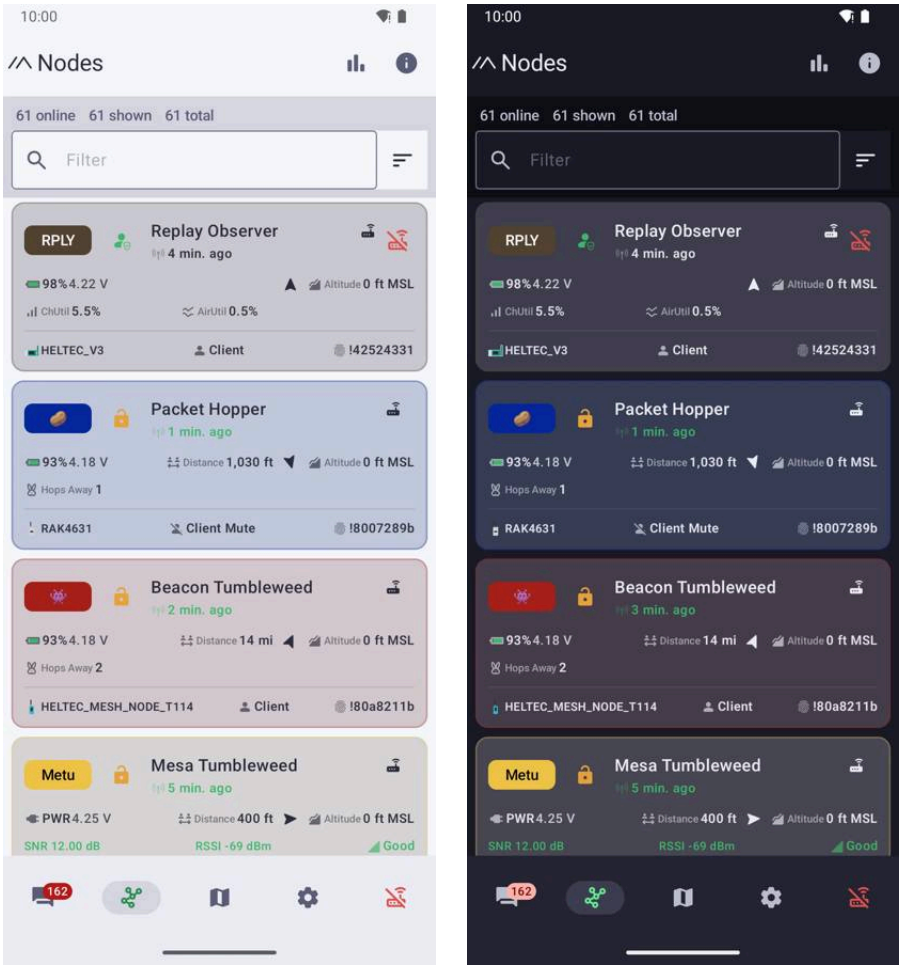
## Checking whether it worked

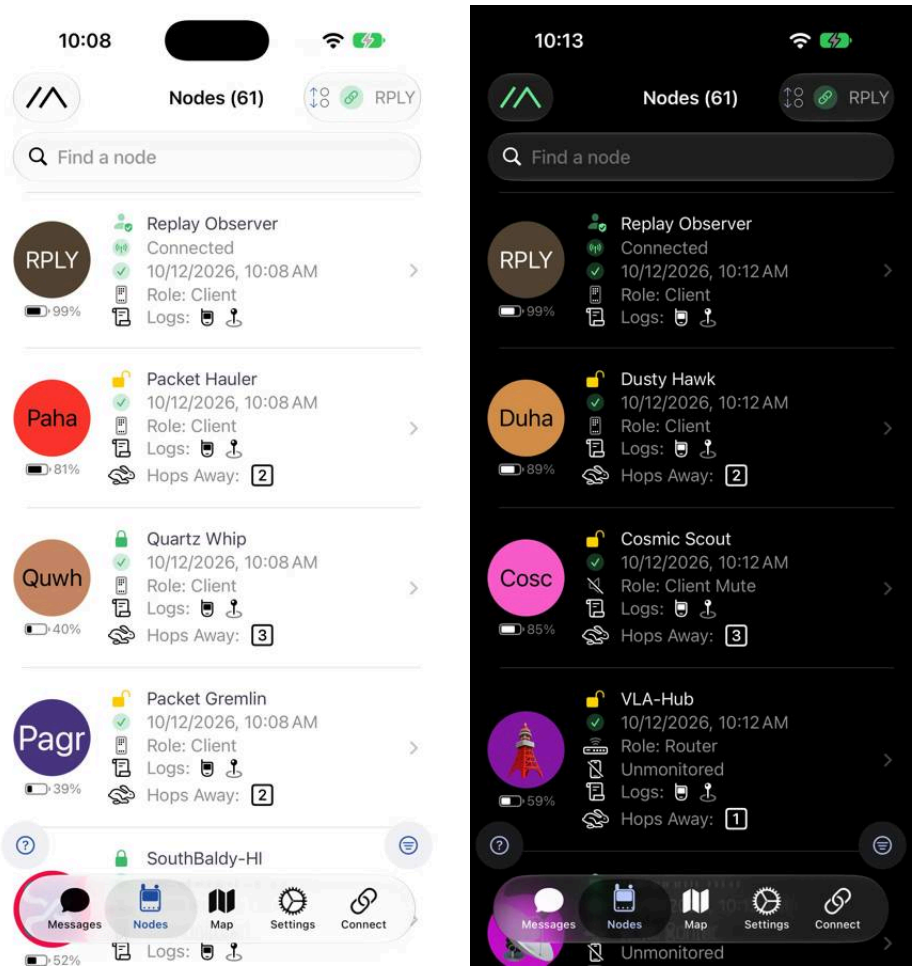
Send a direct message rather than a channel message. The node a direct message is addressed to acknowledges it, and only that acknowledgment confirms delivery. When a nearby node relays the message, the sending node treats the relay as an acknowledgment and stops retrying, so delivery stays unconfirmed until the recipient answers. When no node relays it and the retransmissions run out, the client reports a delivery error.

A channel message gets a weaker check. The clients request an acknowledgment on channel messages too, but no single node is expected to answer a broadcast, so the sending node counts it as delivered as soon as it overhears another node repeat it. If no node repeats it, the client reports a delivery error, the same as a failed direct message. The silent failure is narrower: one nearby node repeats the message, the client receives a success acknowledgment, and no node beyond that one ever receives it.

## After changing settings

The node database persists across a configuration change and across a reboot, so nodes heard on previous settings stay in the list and are no longer reachable. A node carried to an event and switched to the settings the event runs keeps every node it heard at home, listed with its name and last-heard time, looking exactly like a node still in reach.


10:00
Nodes
61 online 61 shown 61 total
Filter
RPLY
Replay Observer
4 min. ago
98% 4.22 V
Altitude 0 ft MSL
ChUtil 5.5%
AirUtil 0.5%
HELTEC_V3
Client
!42524331
Packet Hopper
1 min. ago
93% 4.18 V
Distance 1,030 ft
Altitude 0 ft MSL
Hops Away 1
RAK4631
Client Mute
!8007289b
Beacon Tumbleweed
2 min. ago
93% 4.18 V
Distance 14 mi
Altitude 0 ft MSL
Hops Away 2
HELTEC_MESH_NODE_T114
Client
!80a8211b
Metu
Mesa Tumbleweed
5 min. ago
PWR 4.25 V
Distance 400 ft
Altitude 0 ft MSL
SNR 12.00 dB
RSSI -69 dBm
Good
162




Firmware 2.8.1 and later records which LoRa settings each node was heard on, and clients use it to mark nodes not heard on your current LoRa settings. Configuration tips covers what that means for a working mesh.

## Next steps

LoRa Configuration

Radio Settings

Configuration Tips

# The phone app (Android)

## First start

From https://meshtastic.org/docs/software/android/user/onboarding/

This page covers the first-launch flow of the Meshtastic Android app, what each permission is for, and how to revisit them later.

### First Launch

When you open the app for the first time, the app guides you through an introductory flow that configures essential permissions and settings. Complete each step in order or skip it — nothing here is a one-time offer. Every permission can be reviewed and granted later from the **Permissions** section of **Settings** inside the app.

#### Welcome Screen

The welcome screen introduces Meshtastic with three feature rows:

| | |
|---|---|
| **Stay Connected Anywhere** | Communicate off-the-grid with your friends and community without cell service. |
| **Create Your Own Networks** | Easily set up private mesh networks for secure and reliable communication in remote areas. |
| **Track and Share Locations** | Share your location in real-time and keep your group coordinated with integrated GPS features. |

Tap **Get started** to proceed through the setup flow.




## Permissions

The app requests several permissions during setup. Each one serves a specific purpose, and some are required for core functionality.

### Bluetooth Permission

Bluetooth is the primary connection method between your phone and Meshtastic radio. The **Bluetooth** screen shows what the permission buys you:

- **Discovery** — find and identify Meshtastic devices near you.
- **Configuration** — wirelessly manage your device settings and channels.

On Android 12 and newer, Android asks once, for **Nearby devices**, and that one grant covers both scanning and connecting. Without it, you'll need to use USB or TCP connections instead.

## Location Permission

> ℹ️ **Note:** Location is not required for Bluetooth on Android 12 and newer. **Android 11 and older** show one location step, on the Bluetooth screen, rather than two — those releases treat a Bluetooth scan as a location capability, so the app asks for Location instead of "Nearby devices". Asking twice would push you toward the point where Android stops offering the dialog at all (a second denial on Android 11; the "Don't ask again" checkbox on Android 10 and older). On **Android 12 and newer** the two are separate: "Nearby devices" is declared `neverForLocation`, and declining Location does not stop you finding or connecting to a radio.

Meshtastic also uses your location for:

- Showing your position on the mesh map
- Calculating distances to other nodes
- Sharing your GPS coordinates with other mesh members (if enabled)

On Android 12 and newer, choose **Precise** to share your position with the mesh: an approximate grant still shows you on the map, but position sharing needs precise location.

Grant **"While using the app"**. The app does not request background location — `ACCESS_BACKGROUND_LOCATION` is not in its manifest — so Android will not offer an "Always" option, and position updates happen while the app is in the foreground or running its foreground service.

Declining leaves the rest of the app working: on Android 12 and newer, Bluetooth is unaffected and only the map position and position sharing are disabled. On Android 11 and older, Bluetooth scanning also stops, because that is the permission Android gates it behind — and system **Location Services** must also be switched on for a scan to return anything.

## Notifications Permission

Notifications alert you to:

- Incoming messages from channels and direct messages
- New nodes joining the mesh
- Low battery on a remote node

> 💡 **Tip:** You can fine-tune notification preferences later in Android system settings — the app creates a separate notification channel per category (plus a few internal ones, like the background service), so you can enable or silence them individually.

## Critical Alerts Permission

Critical alerts are high-priority notifications that break through Do Not Disturb — for emergency mesh alerts and urgent messages.

This step is not a runtime permission prompt. There is no grant/deny dialog: the button opens the Android system settings page for the app's **Alerts** notification channel, where you turn the breakthrough behavior on yourself. Tap **Configure Critical Alerts** to open that page, or **Skip** to move on — you can reach the same page later from Android's notification settings for Meshtastic. This step appears only if you granted notifications on the previous screen — skip or decline them and setup ends there.

### Reviewing permissions later

The **Permissions** section of **Settings** summarizes where every runtime permission stands. On Android 12 and newer it lists five: **Nearby devices permission** (Bluetooth), **Location permission**, **App Notifications**, **Camera permission** (scanning channel and contact QR codes) and **Local network permission** (finding radios over Wi-Fi by mDNS). On Android 11 and older a single **Location permission** row covers both Bluetooth and location, so there are four. The last two are never asked for during setup, only when a feature first needs them.

The section reads *All allowed* when every permission is granted, *Nothing needs your attention* when some have simply never been asked for, and names a count when one is denied — in which case it expands itself. Tap the row to expand or collapse it at any time:

| State | What tapping the row does |
|---|---|
| **Allowed** | Opens the system page, so you can review or revoke it |
| **Not asked yet** | Requests it |
| **Denied — tap to allow** | Explains what the permission is for, then asks again if you agree |
| **Blocked — tap to open system settings** | Android will no longer show its dialog, so this opens the page where you can turn it back on |
| **Not required on this version of Android** | Nothing — the permission does not exist on your device |

This matters most for notifications. If you decline them during setup, this row is the way back: Android stops showing the dialog once you have declined firmly (a second denial), at which point this row switches to **Blocked** and sends you to the system settings page instead. The notification prompt exists only on Android 13 and newer — on older versions notifications are on by default and managed from Android's own settings.

## After Setup

After you grant permissions, the app opens the main interface. Your first action should be connecting to a Meshtastic radio — see Connections for detailed instructions.

> 💡 **Tip:** If you skipped any permissions during setup, open the **Permissions** section of **Settings** in the app. Every runtime permission is listed there with its

current state and a way back to it — including notifications, which the system will not prompt for a second time on its own.

Features also ask in context. On the **Connect** tab, a card above the device list explains what the Bluetooth permission is for and offers **Grant permission**; once Android stops prompting, that button becomes **Open settings**.

New to Meshtastic? The getting started guide on meshtastic.org covers hardware selection, initial radio configuration, and your first mesh setup.

## Related Topics

- Connections — pair your first radio
- Messages & Channels — send your first message
- Nodes — see who else is on your mesh
- Map & Waypoints — view node positions
- Settings — Radio & User — configure your radio and user profile

# Connecting to a radio

From https://meshtastic.org/docs/software/android/user/connections/

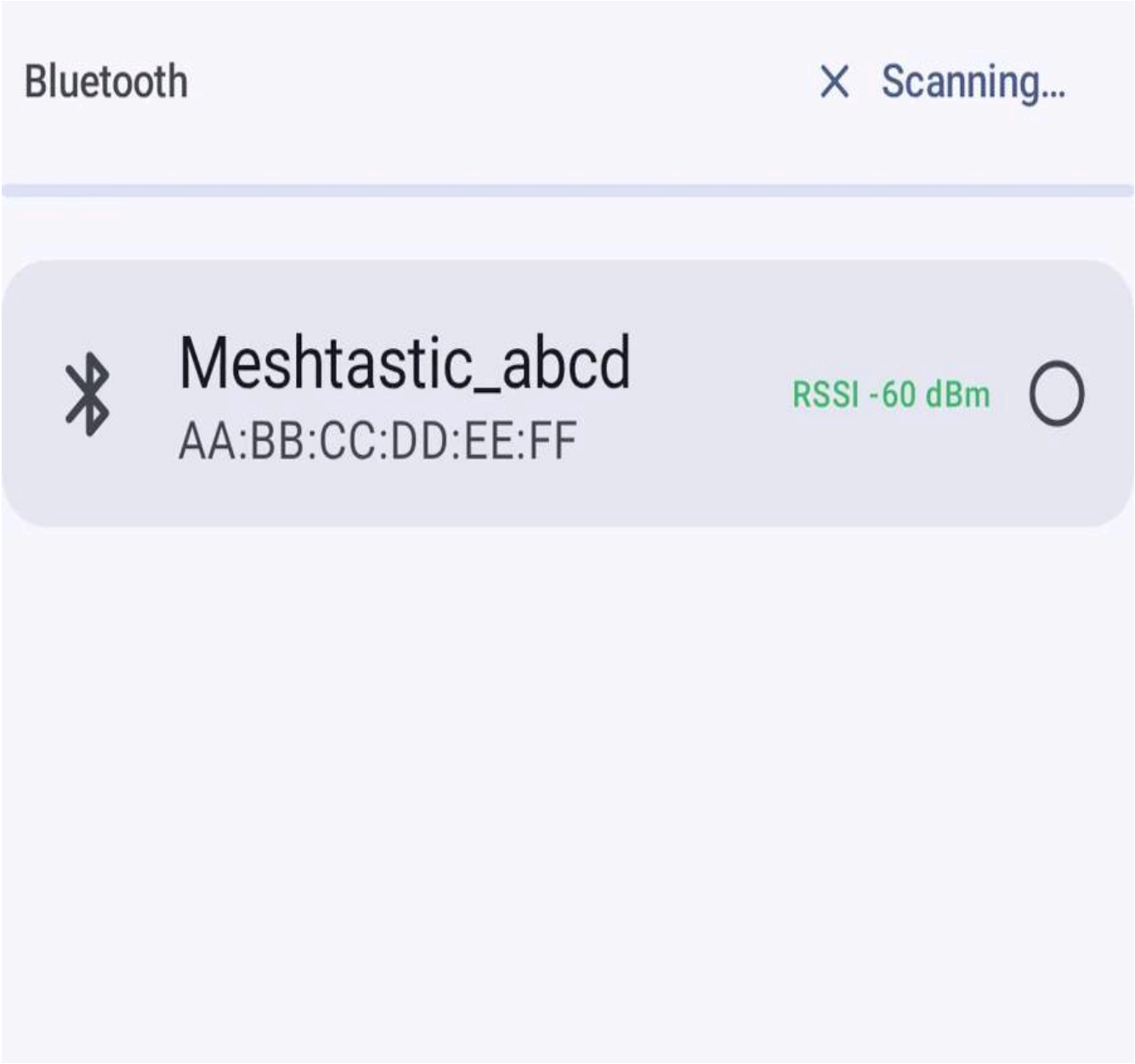
Meshtastic supports multiple transport methods to communicate between your phone or desktop and a radio.

## Bluetooth (BLE)

Bluetooth Low Energy is the default and most common connection method on Android.

### Pairing a Radio

1. Power on your radio. Most radios advertise over Bluetooth as soon as they boot — there is no pairing mode to enter. Radios with a color touchscreen ship with Bluetooth switched off, so turn it on from the radio's own on-screen menu first.
2. Open the app and navigate to the **Connect** tab.
3. Tap **Scan for Bluetooth devices** — nearby Meshtastic radios will appear.
4. Select your radio from the list.
5. Android asks you to pair. If your radio has a screen, it shows a six-digit PIN — type that into the Android dialog. If your radio has no screen, the PIN is `123456`.




You can change the pairing method, or turn Bluetooth on for a radio that ships with it off, under **Settings → Device configuration → Bluetooth** — see Settings — Radio & User. For more information, see Bluetooth configuration on meshtastic.org.

Use the transport selector — a segmented button row below the connection card — to switch between the Bluetooth, Network, and USB transports (one is active at a time):




> 💡 **Tip:** If your radio doesn't appear, check that it isn't already connected to another phone, or out of range.

The screen names anything on the app's side that is blocking a scan, with the fix attached:

| What you see | What it means |
|---|---|
| A card asking for **Nearby devices** | The permission has not been granted. **Grant permission** requests it; once Android stops prompting, the button becomes **Open settings**. |
| **Bluetooth is off** | The adapter is disabled — the card opens Bluetooth settings. |
| **Bluetooth scanning also needs location services** | Android 11 and older only: the permission is held but the system location toggle is off. |
| No card, and **No Bluetooth devices seen** under the header | Nothing on this side is blocking the scan. The radio is out of range, off, or already connected elsewhere. |

The explanation lives in that card, not in the scan control: tapping **Scan for Bluetooth devices** after you have declined once asks Android again directly.

## Connection Status

| Icon | State | Description |
|---|---|---|
| 🟢 | Connected | Active radio link established |
| 🟡 | Connecting | Handshake in progress |
| 🔴 | Disconnected | No active connection. The app retries automatically, with a growing delay between attempts |
| ◦ | Device sleeping | The radio is in light sleep — the app is waiting for it to wake and reconnect, not failing |

These are the four states the app models. "Device sleeping" is normal on power-saving configurations and needs no action.

When connecting, a status indicator shows the current connection state — tap **Stop Connecting** to abandon the attempt:




With no radio chosen yet, the connection card reads **No device selected**. A pane with nothing to list says so under its header, with a hint:

| Pane | What it shows |
|---|---|
| Bluetooth | **No Bluetooth devices seen**, and "Ensure you're within range of the device." Start a scan from the header. |
| Network | **No network devices seen**, and "Ensure you're connected to the same network as the device." Shown only while nothing has been discovered and **Recent Network Devices** is empty. **Add device manually…** stays below it. |
| USB | **No USB devices detected**, and "Connect a device with a USB data cable to use serial." |

## Troubleshooting Bluetooth

- **Radio not found:** Turn Bluetooth off and back on. On Android 11 and older, also check that system location services are switched on — those releases do not return scan results without them.
- **Bluetooth scan couldn't start:** Try again, and toggle Bluetooth off and on if it repeats.
- **Connection drops:** Move closer to the radio; check for interference.
- **Pairing failed, or pairing did not complete:** Check that the **Nearby devices** permission is granted, then pair again.
- **Pairing rejected:** Forget the device in Android Bluetooth settings and retry.
- **Could not establish a stable connection after repeated attempts:** The app stopped retrying after three failed handshakes — a radio that keeps failing here is

usually crashing on reconnect. Power-cycle the radio, then tap it again on the **Connect** tab to start a fresh attempt.

## USB Serial

USB connections provide a wired alternative, useful for desktop or when Bluetooth is unavailable.

### Setup

1. Connect your radio to your phone with a USB data cable. Charge-only cables carry no data lines, and a radio on one never appears in the list.
2. The app prompts for USB permission — tap **Allow**.
3. The connection is established automatically.

ℹ️ **Note:** USB connections require OTG support on Android devices.

### Troubleshooting USB

- **USB permission denied:** Unplug the radio and plug it back in — Android asks again on reconnect.
- **No radio in the list:** Check that the cable carries data rather than only power, and that the phone supports OTG.

## TCP/IP (Network)

Some Meshtastic radios support Wi-Fi/Ethernet connectivity, allowing TCP-based connections over your local network. Get the radio onto your network first. Connect to it over Bluetooth or USB, open **Settings → Device configuration → Network**, and under **Wi-Fi Options** turn on **Wi-Fi enabled** and enter the **SSID** and **Password**. The Network screen appears only for radios whose hardware supports Wi-Fi or Ethernet. Once the radio has an address, come back and connect to it over the network.

### Connecting over the Network

1. Make sure the radio is on the same local network as your phone/desktop.
2. On the **Connect** tab, select **Network** in the transport selector.
3. Choose the radio one of two ways:
    - **Scan for network devices** — toggle this on to auto-discover radios that advertise themselves on the local network (mDNS / `_meshtastic._tcp`). Discovered devices appear in the list; tap one to connect.
    - **Add device manually…** — enter the radio's IP address (or hostname) and port (default: `4403`).
4. Previously-used network addresses are remembered under **Recent Network Devices** for quick reconnection (touch & hold to remove one).

Network discovery uses mDNS, which only works when both your phone and the radio are on the same subnet. If the phone is not on Wi-Fi at all, the app warns that a network scan may find nothing. On Android 17 and newer, the app needs the **Local network permission** to reach a radio on your own Wi-Fi at all — not only to discover it — so typing the address by hand does not work around a denied permission. Grant it from the card on the Network pane, or from the **Permissions** section of **Settings**. A radio on a public address, or one reached over a VPN, needs no permission.

### When to Use TCP

- Radio is on the same local network
- Testing with a simulated radio
- Environments where Bluetooth has interference issues

### Wi-Fi Provisioning for mPWRD-OS

**Settings → Wi-Fi Provisioning for mPWRD-OS** is a separate, narrower tool. It sends Wi-Fi credentials over Bluetooth to **mPWRD-OS** devices only, using their own protocol — it does not configure Wi-Fi on an ordinary Meshtastic radio. It is available on both Android and desktop.

1. Open the screen and wait while the app finds the device over Bluetooth.
2. Tap **Scan for Networks**, then pick a network from **Available Networks** — or turn on **Hidden network** and type the name into **Network Name (SSID)**.
3. Enter the **Password** and tap **Apply**.

If the scan reports **No networks found** or fails outright, move the phone closer to the device and scan again. If **Failed to apply Wi-Fi configuration** comes back, check the password and try again.

## After Your First Connection

Being connected is not the same as being able to transmit.

A radio leaves the factory with no LoRa region set, and it does not transmit until you set one. When you connect such a radio, the **Connect** tab shows a **Set your region** card; tap it to open the LoRa screen and choose the region you are in.

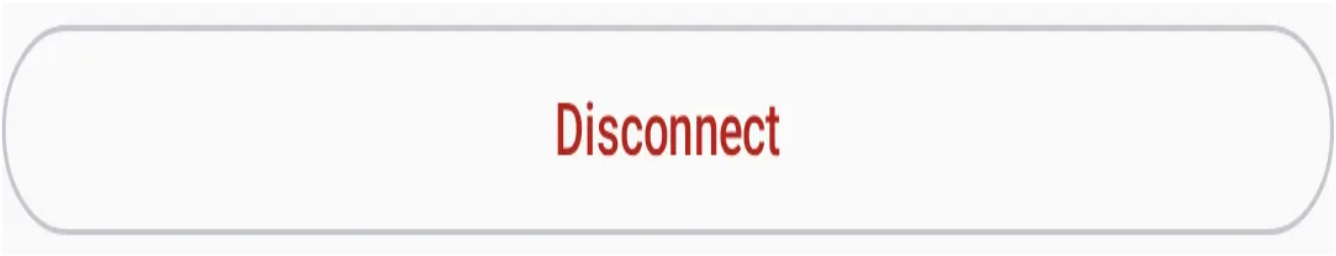
Once the region is set, the tab warns you if the radio is receive-only: a **Transmit is disabled** card, reading "This device can receive but will not send anything over LoRa." Tap it to open the same screen and turn **Transmit Enabled** back on. Only one of the two cards appears at a time — an unset region already stops the radio transmitting, so the app names that first and holds the transmit card back until you have set a region.

For more information, see Settings — Radio & User.

## Reconnection Behavior

The app reconnects to the last selected radio on startup. You can switch transports from the **Connect** tab at any time.

To disconnect, tap the disconnect button on the **Connect** tab:




## Desktop Connections

On Desktop (Linux/macOS/Windows), the app supports:

- **Bluetooth (BLE)** — via the Kable library; works on macOS, Linux, and Windows
- **USB Serial** — primary wired connection method
- **TCP/IP** — for network-connected radios

See Desktop App for platform-specific details and keyboard shortcuts.

## Related Topics

- Getting Started — first-launch setup and permissions
- Settings — Radio & User — Bluetooth, region, and network configuration
- Messages & Channels — send your first message once the radio is connected
- Nodes — see who else is on your mesh
- Desktop App — desktop-specific connection details
- Supported devices — full list of compatible radios on meshtastic.org

# Messages and channels

From https://meshtastic.org/docs/software/android/user/messages-and-channels/

Meshtastic supports two communication modes: **channel broadcasts** and **direct messages**.

## Channels

Channels are shared communication groups. All nodes configured with the same channel key can read and send messages on that channel.

### Default Channel

Every Meshtastic radio comes with a default **LongFast** channel. It is encrypted with a well-known default key, so anyone running Meshtastic on the same preset can read it.

### Channel Security

Each channel carries a lock icon that shows how well it is protected. Tap the icon to see the same explanation inside the app.

| Icon | What it means |
|---|---|
| Green closed lock | The channel is securely encrypted, with either a 128-bit or a 256-bit AES key. |
| Yellow open lock | The channel is not securely encrypted — it uses no key at all, or a well-known one-byte key — and it does not carry precise location. |
| Red open lock | Not securely encrypted, and the channel carries precise location data. |
| Red open lock with a warning badge | Not securely encrypted, carrying precise location data, and uplinking that data to the internet over MQTT. |

Key length alone does not change the icon: a 128-bit key and a 256-bit key both show the green lock.

> 🔒 **Security:** Always configure a unique PSK for private communications. The default channel is intentionally open so new users can discover the mesh — but you should create a separate encrypted channel for anything sensitive.

### Adding a Channel

1. Connect to your radio. The **Channels** row stays grayed out until the app has a connection — see Connections.
2. Go to **Settings**, then tap **Channels** under **Configuration**.
3. Tap the **+** button to add a channel. The editor opens on the new entry.
4. Set the channel name and the **PSK**, and choose whether the channel uses MQTT uplink and downlink. Naming a new channel generates a fresh 256-bit key for you; the refresh icon beside **PSK** generates another one.
5. Tap **Save** to close the editor. The change is still only on your phone.

6. Tap **Send** at the bottom of the channel list to write the changes to the radio. **Cancel**, or leaving the screen without tapping **Send**, throws them away.
7. Optional: share the channel URL or QR code with the people who need access.

Tapping an existing channel opens the same editor, where you can change the name, the PSK, MQTT uplink and downlink, and position precision. Every edit on this screen — adding, editing, deleting, or dragging a channel into a new order — waits on **Send** the same way.

### Archived Channels

A conversation belongs to the channel it was held on, not to the slot that channel occupied. When a channel leaves your radio — you scan a different channel QR, replace a channel's key, or switch the modem preset — its conversation is **archived** rather than deleted or merged into whatever takes the slot.

An archived conversation:

- stays in the **Channels** list under the name the channel had, below the channels currently on your radio, marked with a history icon;
- keeps its full message history, searchable as usual;
- is read-only — you cannot send, reply, or react, because the channel is no longer on the radio to send on, and it is not offered as a share target;
- comes back automatically if you re-add that exact channel, taking its messages back to the live slot.

Changing the modem preset archives the channel too: the preset is part of a default channel's identity, so **LongFast** and **MediumFast** are different channels even though neither the name nor the key changed. Changing only the region does not archive anything.

To remove an archived conversation for good, long-press it in the conversation list and delete it.

## Direct Messages

Direct messages (DMs) go to one specific node. When both radios hold each other's public keys, your radio encrypts the message to that node's public key, so no one else on the mesh can read it — not even nodes that share your channel.

Your radio must already hold the other node's public key before it can send a DM. Keys travel inside node info, which nodes broadcast periodically, so the key usually arrives on its own once you have heard from that node. Until it does, a radio that has its own key pair — the default — refuses the send rather than falling back to channel encryption, and the message shows **Recipient key unavailable**.

A public-key conversation carries a key icon in its top bar. A green closed lock means the direct message is protected by public-key encryption; a red key-off icon means the node's public key changed and no longer matches the one your radio stored. Tap the icon for the details.

## Sending a Direct Message

1. Open the **Messages** tab.
2. Select a conversation, or tap a node in the node list.
3. Type your message and tap **Send**.

## Managing the Conversation List

The **Messages** tab lists your conversations. Each row shows what you need at a glance, and you can act on it directly:

- **Unsent drafts survive.** Type into a conversation and leave without sending, and the text is still there when you come back. The row shows it as `Draft: …` in place of the last message — an unsent draft is the thing the row is waiting on *you* for.
- **Unread badge.** A count sits on the row until you open the conversation.
- **Swipe right to mute** (swipe again to unmute) and **swipe left to delete**. Deleting asks first; muting shows a snackbar with **Undo**.
- **Touch & hold to select** one or more conversations, then use the action bar to **Pin**, **Mark unread**, mute or delete them together. Pinned conversations carry a pin marker and rise to the top of **their own section**.
- **The list is split into Channels and Direct Messages**, each with a collapsible header and each sorted independently — so a pinned direct message rises within its own section, not above the Channels one.

## Conversation Bubbles

On Android 11 and later, a message notification can be opened as a floating **bubble** that stays on top of whatever else you are doing. Tap the bubble icon on the notification to promote a conversation; Android remembers the choice per conversation, and the system Bubbles settings control whether they are offered at all.

## Message States

A status label appears under **your own** outgoing messages only (incoming messages from others show no status label):

| State | Meaning |
|---|---|
| Sending… | Queued or already handed to the radio, not yet resolved either way. Both stages share this text, but the icon and color change as it progresses — a yellow upload cloud while queued, a blue arrow once the radio has it |
| Delivered to recipient | The strongest confirmation for a direct message — an acknowledgment came back |
| Delivered to mesh | For a channel broadcast, the message reached the mesh (broadcasts have no per-recipient ack) |

| State | Meaning |
|---|---|
| Relayed, not confirmed by recipient | For a direct message, shown in a warning color — the message was relayed but no acknowledgment has come back yet |
| Routing via SF++ chain… | Being routed/buffered by the Store & Forward Plus Plus chain |
| Confirmed on SF++ chain | Confirmed delivered via the SF++ chain |
| Error | Delivery failed — tap the status for the specific reason (see Delivery Errors) |

## Delivery Errors

When a message fails to deliver, the error indicator shows what went wrong:

| Error | Meaning | What to Do |
|---|---|---|
| No route | No path exists to the destination node | The recipient may be offline or out of mesh range. Try later or move closer. |
| No radio interface | No radio interface available to send | Check that your radio is connected and available. |
| Failed to deliver to mesh | Retries exhausted. The same label covers three underlying causes — a relay refusing (NAK), a plain timeout, and running out of retransmits | Move closer, improve signal, or wait for conditions to improve. Tap the error for the specific cause. |
| Rate limited | The mesh is throttling you for sending too fast | Wait before sending again. |
| Not authorized | The destination refused the request | Check you have the right channel and keys for that node. |
| Recipient needs your key | Direct-message encryption could not complete because the other node does not have your public key yet | Exchange node info — the key travels with it. Common on a first DM to a new contact. |
| Recipient key unavailable | You do not have the recipient's public key | Wait for their node info to arrive, or ask them to broadcast it. |
| Could not send encrypted message | Encryption failed for this direct message | Verify both nodes have exchanged keys and are on compatible firmware. |
| Admin session expired | A remote-admin session timed out | Reopen the remote node's settings to start a new session. |
| Admin key not authorized | The target node does not accept your admin key | Verify the admin key matches on both nodes. |
| Channel/key mismatch | Destination channel/key does not match | Verify both nodes share the same channel and PSK. |
| Message is too large to send | Message exceeds maximum payload size | Shorten the message and try again. |

| Error | Meaning | What to Do |
|---|---|---|
| No app response | App or plugin did not respond to the request | Retry or check the destination app or module state. |
| Duty cycle limit | Regional airtime limit reached | Wait for the duty cycle window to reset. |
| Invalid request | Malformed or invalid request | Retry after updating or restarting the app if this persists. |

> 💡 **Tip:** Most delivery errors resolve themselves. If a node is intermittently reachable, the mesh will retry. For persistent **No route** errors, check that intermediate Router nodes are online.

## Message Features

### Quick Chat

Pre-configured messages for rapid communication, useful when typing is impractical (gloves, small screen, urgent):

- The quick chat row is hidden until you turn it on. Open a conversation, tap the overflow menu in the top bar, then tap **Show quick chat menu**. **Hide quick chat menu** puts the row away again.
- The row carries one built-in entry, the 🔔 alert bell. It appends an alert message that includes a bell character, which clients that support it flag as an alert. Every other button on the row is one you created.
- Add, edit, reorder, and delete your own entries from the same overflow menu — tap **Quick chat options**.



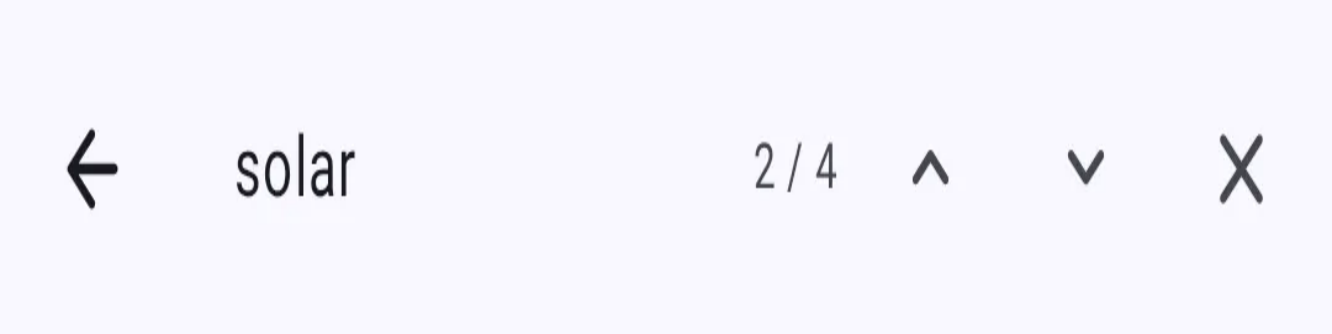

Each quick chat entry has a **Name** — the button label, capped at five characters, forced to uppercase, and filled in for you from the message text — and the **Message** it carries. A switch decides what tapping the button does. A new entry starts on **Instantly send**, so a tap sends the message straight away; turn the switch off and the label changes to **Append to message**, which puts the text in the input field for you to edit first.




### Searching Messages

You can search the full history of any conversation directly from the chat screen:

1. Open a conversation (a channel or a direct message).
2. Tap the **search icon** in the top bar.
3. Type into the **Search messages…** field. The search runs as you type, across all stored messages in that conversation.
4. Use the **N / M** result counter and the **previous / next arrows** to jump between matches, which are highlighted in the conversation.




> 💡 **Tip:** Search is full-text and stays within the conversation you opened it from — it doesn't search across other channels or contacts. It matches against the messages already stored on your device, so it works fully offline.

## Message Bubbles

Messages appear as chat bubbles — sent messages on the right, received messages on the left. Each bubble shows the sender, timestamp, and delivery status. Messages with replies include a quoted preview of the original message above the response.

## Text Formatting

Messages support lightweight inline **Markdown**. Received messages render the styling with the syntax characters removed:

| Type | Syntax | Renders as |
|---|---|---|
| Bold | `**bold**` | **bold** |
| Italic | `*italic*` | *italic* |
| Strikethrough | `~~strike~~` | ~~strike~~ |
| Inline code | `` `code` `` | monospace `code` |
| Link | `[label](https://example.com)` | a tappable **label** |

When composing, focus the message field and type at least three characters to reveal a **formatting toolbar** below the input. Select text and tap a style to wrap it (tap again to remove it); with no selection, a style inserts an empty pair with the cursor between the markers. The link button opens a dialog to enter a URL. As you type, the field shows the styled text, but the message you send still contains the Markdown characters.

> 💡 **Tip:** Formatting is carried as literal characters on the mesh — the same bytes iOS sends. Clients that don't support Markdown (older apps, plain firmware clients) will show the raw **/~~ characters. URLs, email addresses, and phone numbers are still auto-linked whether or not you use Markdown.

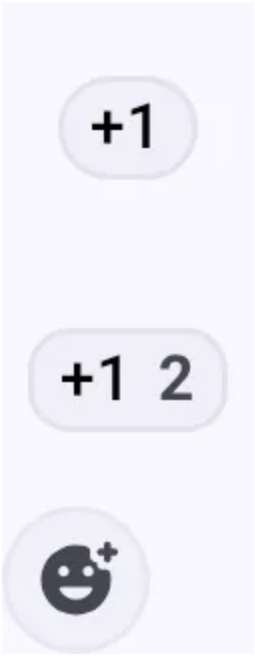
## Mentions

Type @ while composing to mention a node — a picker suggests matching contacts as you type. In a received message, a mention appears as a highlighted chip showing the node's name; tap it to jump straight to that node's detail page.

## Reactions

React to messages with emoji:

- **Touch & hold** a message — or double-tap it — to raise a quick reaction bar above the bubble. Opening the bar sends nothing.
- Tap an emoji in the bar to send it; tap **More reactions** to open the full picker, or anywhere outside the bar to dismiss it without sending. A reaction is a real mesh packet, so it only goes out when you pick an emoji.
- Reactions appear below the message bubble
- Multiple users can react to the same message
- React to your own messages or others' messages




> 💡 **Tip:** Reactions are lightweight — they use minimal mesh bandwidth compared to full text messages.

## Replying

**Swipe a message to the right** to reply to it — the composer opens with that message quoted. Swiping past the reply threshold arms the action; releasing before it springs back with nothing sent. Reply is also in the actions sheet, reached by touching & holding and then tapping **More message actions**.

## Day Separators

Messages are grouped by day. The separator above the first message of each day reads **Today** or **Yesterday** for the two most recent days, and the date itself for older ones.

### Jump to Latest

Scrolling back through a conversation raises a jump-to-latest control. When messages arrive while you are scrolled up, it names the most recent sender and adds a count of the other unread messages. That count is messages, not people — five unread from one person reads as their name **+4**.

### Message Actions

Touch & hold or double-tap a message to open the quick reaction bar, then tap **More message actions** (the overflow icon on that bar) to open the actions sheet. The emoji row runs across the top of the sheet — that is where reactions live — and beneath it, along with the message's timestamp and delivery status, are:

- **Reply** — quote the message in your response
- **Copy** — copy the message text to the clipboard
- **Translate** — translate a received message into your device language, and toggle between the original and translated text (Google Play build only; uses on-device translation). The first translation into a language asks to download a one-time language model and tells you its size, then translates once the download finishes. If the download fails, or the message is already in your language, the app says so instead of translating
- **Select** — start multi-select, so you can act on several messages at once
- **Delete** — remove the message from this phone. It works on any message in the conversation, yours or not, and does not remove it from anyone else's radio or phone

### Message Priority

The app sends every message you compose at the same, default priority — there is no emergency or alert tier to choose, and nothing in the app raises a direct message above a channel broadcast. Any prioritising between them happens in firmware, not here. (The app does mark some of its own internal traffic, such as admin and traceroute packets, as reliable or background, but that is not something you control from the message composer.)

### Message Limits

- **Maximum length:** 200 bytes (approximately 200 characters for ASCII text)
- The 200-byte cap applies to the in-app composer — the mesh payload limit itself is 233 bytes, so messages from other senders (e.g., App Functions) may arrive slightly longer
- **Rate limiting:** The mesh enforces airtime fairness; heavy message volume may be throttled
- **Delivery:** Messages are retried automatically if no acknowledgment is received

## Best Practices

- Use channels for group coordination
- Use direct messages for private person-to-person communication
- Keep messages short — mesh bandwidth is limited
- Configure encryption for sensitive communications

## Related Topics

- Nodes — tap a node to start a direct message
- Settings — Radio & User — configure channel encryption and presets
- MQTT — bridge channel messages to the internet
- Channel configuration — detailed channel settings on meshtastic.org

# The list of radios (nodes)

From https://meshtastic.org/docs/software/android/user/nodes/

The Nodes screen lists every node visible on your mesh.

## Node list

The node list shows every node your node has heard, including:

- **Node name** — user-configured long name
- **Short name** — 4-character identifier
- **Signal quality** — SNR, RSSI, and a quality word, shown only for nodes your node heard directly. In the Complete layout a node reached through a relay shows its hop count here instead; a node heard only over MQTT shows neither
- **Last heard** — time since last communication
- **Distance** — estimated distance (if positions are shared)
- **Battery** — remote node battery level (if telemetry is enabled)

### Choosing what the list shows

The list has two densities, set at **Settings → Node Layout**. **Complete** shows every field a node has reported and hides the ones it hasn't. **Compact** fits more nodes on screen and lets you pick the fields yourself — **Power**, **Last Heard Time**, **Relative Last Heard Time**, **Distance and Bearing**, **Hops Away**, **Signal (Direct Only)**, **Channel**, and **Device & Role**. The **Environment Metrics** toggle applies to both densities. A preview above the toggles shows the effect before you leave the screen.

### Node Status indicators

| Indicator | Meaning |
|---|---|
| Green last-heard time | Node heard within the last 2 hours |
| Plain last-heard time | Node not heard for over 2 hours |
| Orange last-heard time with a crossed-out signal icon | Not heard since your node's LoRa settings changed, so it can't be reached from here |
| Struck-through name | Node you have ignored |
| ⭐ Favorite | Node you marked as a favorite. |

There is no separate "away" tier, but the orange unreachable state takes precedence over the green one: a node can be online and still be unreachable on your current settings. Nodes known only over MQTT are never shown as unreachable.

### Node roles

Nodes can be configured with different roles that affect their mesh behavior:

| Role | Description |
|---|---|
| Client | Standard end-user node |
| Client Base | Treats favorited-node traffic as Router Late priority; all other traffic as Client |
| Client Mute | Receives but doesn't retransmit |
| Client Hidden | Like Client Mute, plus hides from node list |
| Router | Prioritizes message forwarding; stays awake to relay |
| Router Late | Infrastructure node that rebroadcasts once, but only after all other modes (provides supplemental coverage) |
| ~~Router Client~~ | ⚠️ **Deprecated** (removed in firmware 2.3.15) — no longer selectable; use Router or Client instead |
| ~~Repeater~~ | ⚠️ **Deprecated** (removed in firmware 2.7.11) — no longer selectable; use Router instead |
| Tracker | Optimized for position reporting at regular intervals |
| Sensor | Optimized for telemetry reporting |
| TAK | Interoperates with TAK systems (sends/receives CoT) |
| TAK Tracker | TAK position reporting only |
| Lost and Found | Sends its position to the default channel as a text message at regular intervals, to help recover a lost node |

## Choosing a role

Most users should keep the default **Client** role. Consider a different role when:

- **Router** — You have a node in a fixed, elevated location with reliable power (rooftop, hilltop). Routers stay awake continuously to relay messages for others and are essential for extending mesh coverage. Don't use Router on battery-powered handheld nodes.
- **Router Late** — An infrastructure node that always rebroadcasts packets once but only after all other routing modes have had their turn. Provides supplemental coverage for local clusters without competing with primary routers.
- **Client Base** — Treats traffic from/to your favorited nodes with Router Late priority (ensuring those messages get extra relay coverage) while handling everything else as a normal Client.
- **Client Mute** — You want to receive mesh traffic but not contribute to relaying. Useful for monitoring-only nodes or to reduce congestion in dense areas.
- **Tracker** — An unattended node whose sole purpose is broadcasting its GPS position (e.g., a vehicle, pet, or asset). Sleeps between broadcasts to conserve battery.
- **Sensor** — An unattended node reporting environmental telemetry (temperature, humidity, air quality). Similar power profile to Tracker.
- **TAK / TAK Tracker** — Only needed if interoperating with ATAK/WinTAK systems. See TAK Integration for details.

  💡 **Tip:** The mesh works best when most nodes are **Client** or **Router**. Too many Client Mute nodes reduce mesh resilience; too many Routers in a dense area can

cause congestion. A good rule of thumb: one Router per 5–10 Clients in your area.

### Security indicators

Each node carries one security icon beside its name in the node list. Tap it to read what it means, and choose **Show All Meanings** in that dialog for the full legend. The detail screen shows the same state in words, as a **Security** row that opens the same dialog.

| Icon | Meaning | Shown for |
|---|---|---|
| Person with a shield check (green) | **Verified contact** — you verified this node's key in person by exchanging contact QR codes, so its identity is confirmed. The strongest trust the list shows. Your own connected node carries it too | Any firmware version |
| Nodes icon with a shield check (green) | **Signed node** — this node signs its broadcasts with its identity key, so its identity is consistent over time, but you haven't verified it in person. On firmware 2.8 the icon appears from the version alone, before any signed broadcast has been heard | Firmware 2.8 or newer, and any node whose signed broadcast your node has verified |
| 🔒 Closed lock | A public key is on file and matches, so direct messages to this node are encrypted | Firmware before 2.8, or no reported version |
| 🔓 Open lock | No public key has been received for this node, so it can't be direct messaged — use **Request User Info** on the node detail page to ask for one | Firmware before 2.8, or no reported version |
| ⚠️ Mismatch | **Public key mismatch** — a different key arrived for this node after one was stored. Investigate before trusting | Any firmware version |

Every node on firmware 2.8 or newer signs its broadcasts, so on that firmware the signed state is the baseline and the locks aren't shown.

Direct messages always use public-key encryption, so your node needs the other node's public key before it can send one. It refuses the send rather than falling back to channel encryption. Keys arrive inside node info, which is why an open lock usually clears itself once that node is heard from properly.

A mismatch never replaces the key you already hold. The app keeps the first key it recorded and refuses the new one, as the firmware does, so a stray or hostile node info can't silently break encrypted messaging to a contact. To clear a mismatch, first confirm through another trusted channel that the key change was intentional — a factory reset causes one. Then touch & hold the node, choose **Remove**, and let the two nodes exchange keys again the next time yours hears it.

## Quick Actions

From the node list, you can:

- **Tap** a node to view its detail page
- **Touch & hold** for quick actions:

- Mark/remove favorite
- Mute/unmute notifications
- Send a direct message
- Trace route
- Ignore/unignore
- Remove

Touch & hold **your own node** instead and you get one action, **Update status**, which opens the User settings screen with the cursor already in the Status Message field. It only appears while the node is connected and running firmware 2.8 or newer — see Settings — Radio & User for the field itself.

## Sharing a contact

On a node's detail screen, tap **Share Contact** to produce a link and a QR code for that node. From the same dialog, **Share link** opens the Android share sheet (on desktop it copies the link instead), **Write to NFC tag** saves it to a writable NFC tag, and **Copy** puts it on the clipboard. While that dialog is open and in front of you, the phone also offers the same link to any NFC reader, so someone can take the contact by tapping their phone against yours with no tag involved.

Sharing your own contact this way marks it as verified in person, so whoever imports it sees the **Verified contact** icon rather than the signed one. Relaying someone else's contact passes on only what your app had already recorded about them.

To add someone else's contact, use the import button on the node list and choose **Scan Shared Contact QR Code**, **Scan Shared Contact NFC**, or **Input Shared Contact URL**. The app asks you to confirm with **Import Shared Contact?**, and warns you when the contact is one you already have.

## Filtering & sorting

### Text search

Type in the search field to filter nodes by name or short name. The filter updates in real time as you type.

### Filter toggles

| Filter | Description |
|---|---|
| **Hide offline nodes** | Show only nodes heard within the last 2 hours |
| **Only show direct nodes** | Show only nodes your node heard directly, with no relay in between |

| Filter | Description |
|---|---|
| **Include unknown** | Show nodes that haven't sent user info yet. **On by default**, so a node heard before its info arrives stays visible; these carry a badge marking them incomplete, and can't be direct messaged until their user info brings a public key |
| **Exclude infrastructure** | Hide infrastructure-role nodes (Router, Router Late, Client Base, and legacy Repeater nodes) and any node that can't be messaged, whatever its role |
| **Exclude MQTT** | Hide nodes heard only via MQTT internet bridge |
| **Hide unheard nodes** | Hide nodes your own node hasn't heard since its LoRa settings changed. Off by default, and it does nothing on firmware that doesn't report whether a node was heard on the current settings |
| **Signed only** | Show only nodes whose signed broadcasts your node has actually heard and verified. Stricter than the icon: a node on 2.8 shows as signed by its firmware version before any signed broadcast arrives, and a contact verified in person on older firmware isn't signed at all |
| **Encrypted only** | Show nodes with a matching public key on file, the key an encrypted direct message needs. A node with a key mismatch is excluded |
| **Only show ignored Nodes** | Replace the list with the nodes you have ignored. Every other node is hidden while this is on, and a banner appears at the top of the list to take you back |

## Nodes not heard on your current Settings

When your node's LoRa settings change — a different preset, region or frequency slot — nodes heard under the old settings are still in the list but can no longer be reached. The app marks them with an orange last-heard time and a crossed-out signal icon, and shows a banner at the top of the list: **N nodes not heard on your current LoRa settings**.

The banner offers two actions:

- **Keep** dismisses the banner and leaves every node in place.
- **Remove** deletes those nodes from the list. Your favorites and your own node are never removed.

**Remove** is a bulk delete and there is no undo, so use **Keep** if you expect to switch back to the old settings. Removed nodes reappear if your node hears them again.

The **Hide unheard nodes** filter does the same hiding without deleting anything.

## Sort options

| Sort | Description |
|---|---|
| **Last heard** | Most recently heard nodes first |
| **A-Z** | Sorted by node long name |
| **Distance** | Nearest nodes first (requires position sharing) |
| **Hops away** | Fewest relay hops first |
| **Channel** | Grouped by channel index |

| Sort | Description |
| --- | --- |
| **via MQTT** | Grouped by MQTT vs. node-heard |
| **via Favorite** (default) | Favorited nodes first, then the rest |

## Nodes per hop

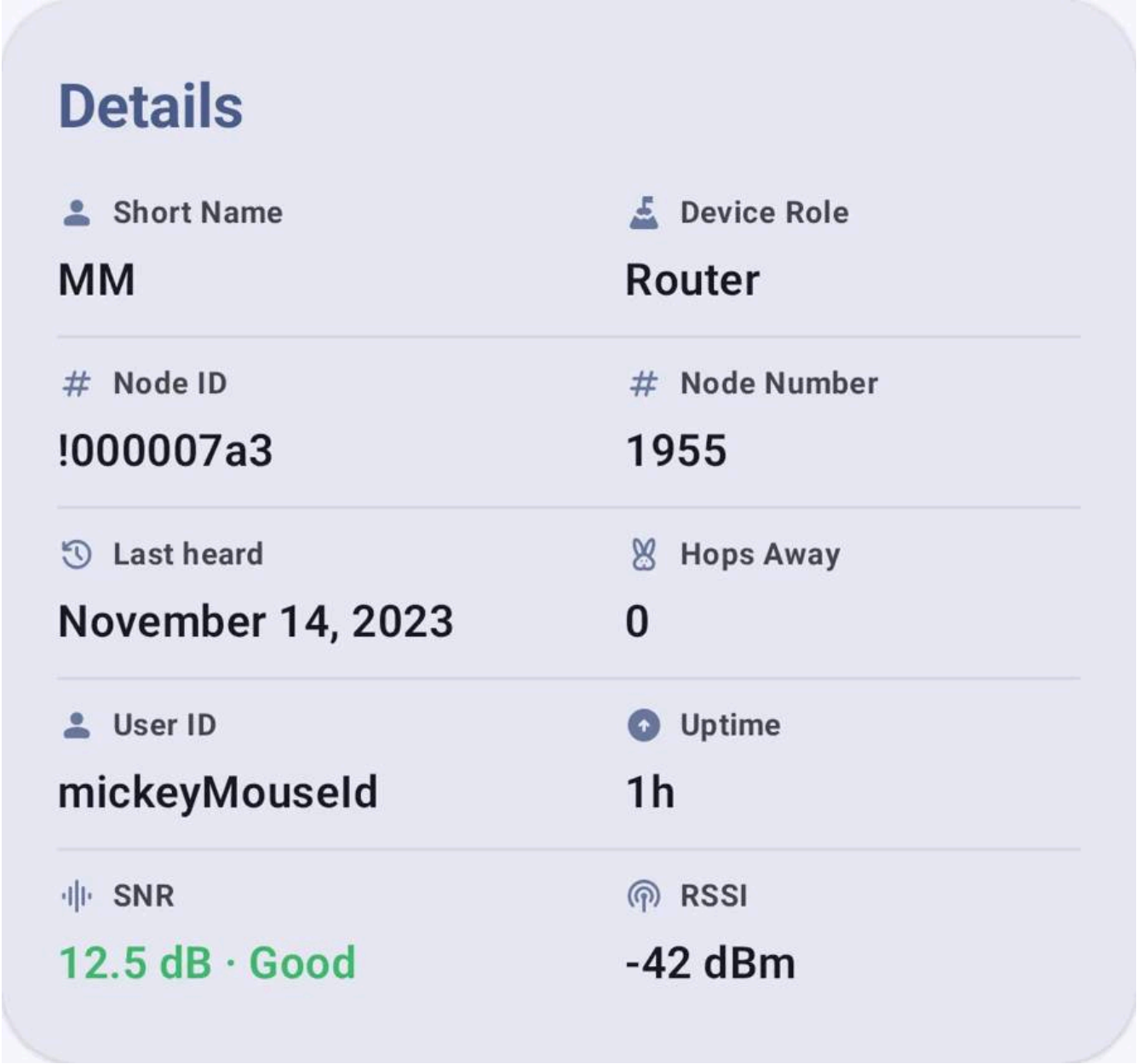
Tap the hop-histogram icon in the node list's app bar to open a bar chart of how many nodes sit at each hop distance (0 = direct, 1 = one relay away, and so on). Filter the chart to a **last heard** window — **All**, **1 Hour**, **8 Hours** or **24H** — to see how the mesh looks right now versus over a longer period. It's a quick way to gauge how busy and spread out your local mesh is.

## Node detail

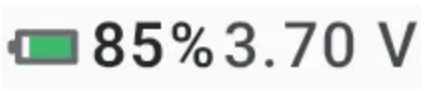
Tapping a node opens the detail view with comprehensive information. See Node Metrics for full details on metrics and telemetry.

Signal quality is rated against your modem preset. The same SNR can be good on a long-range preset and poor on a faster one. Traceroute and neighbor-info SNR colors use that preset too. RSSI text has its own strength colors; it affects the quality rating only when a noise-floor reading is also available.

The Details card carries the node's short name, role, IDs, last heard time, hops away, uptime, and its SNR and RSSI:




Inline status indicators show key metrics at a glance:

| Indicator | Screenshot |
| --- | --- |
| Signal quality |   |
| Battery level |   |
| Hop count |   |

| Indicator | Screenshot |
| --- | --- |
| Last heard |   |
| Distance |   |

## Hardware support status

When a node's hardware is recognized, the detail view names the device and marks its support status, taken from the Meshtastic device registry rather than the app:

| Mark | Meaning |
| --- | --- |
| Rosette (green) | **Supported** — hardware the Meshtastic project actively supports |
| Wrench (sky blue) | **Independent maker hardware** — built and tested by an independent maker, shown as its own rung whether or not the project has promoted it yet |
| Unverified (red) | **Supported by Meshtastic Community** — hardware the project does not actively support, including legacy boards |

The label always accompanies the mark, so the status is never conveyed by colour alone.

## Device links ("I want one")

When a node's hardware is recognized, the detail view shows a collapsible **"I want one"** section linking to places to buy or learn more about that device: the vendor's product page, product variants, and regional marketplace listings (such as AliExpress, Amazon, and supported retailers), filtered to your country. Each link opens through the `msh.to` redirect service. Devices with no matching links don't show the section.

A full, browsable directory of every link is also available at **Settings → Device Links**. The item is hidden while you have Settings open for a remote node.

Some of these are affiliate links. Both places say so above the links: product links may be affiliate links, and purchases may earn Meshtastic a commission.

## Remote shell

**Open terminal**, in the node's **Administration** card, opens a shell on a Linux node running `meshtasticd` 2.8.2 or later, carried over the mesh. The node accepts it only from a radio whose public key is in its **Security → Admin key** list, the same list remote administration uses; the app starts that admin session first if there isn't one, and the card's title shows whether it is active. The terminal takes the whole window. If the node never answers, the screen says so after a minute.

The mesh carries a few hundred bytes a second, and the terminal is built around that:

- Once the node has echoed the first character of a line, what you type appears **underlined** at the cursor until the node echoes it back. At a prompt that doesn't echo, such as a password, nothing you type is shown; in line mode the command box masks its text and keeps it out of history.
- The subtitle shows the measured round trip. When the node stops answering, a line at the top of the terminal says how long ago it was last heard.
- The two rows of extra keys give **ESC**, **TAB**, arrows, **HOME**, **END**, **PGUP** and **PGDN**. **CTRL** and **ALT** are sticky: tap once to apply to the next key, twice to lock, a third time to release. **CTRL** then **c** interrupts a running command. A hardware keyboard's Ctrl and Alt work as usual.
- **Line mode**, in the ⋮ menu, gives you a command box: each command goes out in one transmission, ↑ and ↓ recall earlier commands without asking the node, and chips offer common commands for a `meshtasticd` host. It is the cheapest way to use the shell on a busy mesh.
- The ⋮ menu also pastes from the clipboard and changes the text size.

Colour, progress lines and `clear` display as they would in a desktop terminal. Full-screen programs such as `vi` or `top` redraw the whole screen for each key, which the mesh cannot carry; the terminal shows a notice instead.

## When no nodes appear

The list stays empty until your node hears another node.

- **No device connected** — the app isn't connected to a node. See Connections.
- **Searching for nodes** — the node is connected and listening, but nothing has arrived yet. Check that its region and modem preset match the mesh around you, and leave **Include unknown** on so a node that hasn't yet sent its name still appears. See Settings — Radio & User.
- A node you expect is missing — check the filter toggles. **Only show direct nodes**, **Exclude MQTT**, and **Exclude infrastructure** each hide a whole category of node.

## Related Topics

- Node Metrics — detailed telemetry dashboards for each node
- Messages & Channels — send a direct message to a node
- Map & Waypoints — view node positions geographically
- Local Mesh Discovery — traceroute and neighbor info for topology exploration
- Signal Meter — understand what the signal bars mean

# Map and waypoints

From https://meshtastic.org/docs/software/android/user/map-and-waypoints/

The Map screen shows the geographic positions of nodes on your mesh, along with shared waypoints.

## Map View

The map displays:

- **Node positions** — colored markers for each node reporting location
- **Waypoints** — shared points of interest
- **Your position** — your current GPS location

### Node Markers

Each node that reports a position is shown as a **node chip** marker displaying the node's short name. The chip is colored by the node's own identity color (a stable color derived from its node number) — the same chip used in the node list, so a node looks the same everywhere. Marker color does **not** encode online/offline status. When a node's position updates live, its marker briefly pulses. Nearby markers are clustered as you zoom out.

### Map Controls

- **Zoom** — pinch or use +/- buttons
- **Pan** — drag to explore
- **Center** — tap the location button to center on your position
- **Node tap** — tap a node marker to view details

The floating toolbar provides quick access to the compass, the map type and layers pickers, node filters, Site Planner, and location tracking. Tap the compass to reorient north-up, or tap the location button to center on your current position. On **Google Play** builds a refresh button joins them while a network layer is showing; on **F-Droid** and **Desktop**, refresh a network layer from its own row in the layers sheet instead.



### Filtering the Map

Tap the filter button in the floating toolbar to open **Filter map**. **Display** controls what is drawn: **Only Favorites**, **Show Waypoints**, **Show Precision Circles**, and a slider that hides nodes not heard from recently. **Node roles** is a chip per device role, plus **All** to show every role; a selected chip means that role is shown. **Nodes** narrows the set further with **Hide offline nodes**, **Only show direct nodes**, **Signed only**, **Encrypted only**, **Exclude MQTT**, **Show ignored nodes**, and **Include unknown** — the same rules as the node list's filter toggles.

A dot on the filter button means at least one filter is hiding something — check it before concluding the mesh is quiet. Turning **Show Waypoints** off hides every waypoint, including your own. **Show ignored nodes** adds them to the map rather than showing only them — unlike the node list's **Only show ignored Nodes**.

## Waypoints

Waypoints are shared points of interest, visible to everyone on your mesh.

### Creating a Waypoint

Your radio must be connected — the map ignores a touch & hold while it is not, because saving a waypoint means broadcasting it.

1. Touch & hold the map at the desired location.
2. Enter a name and optional description.
3. Choose an icon/emoji for the waypoint.
4. Tap **Send** to share with the mesh.

Waypoints always broadcast to the whole mesh on the primary channel. Unlike a message, a waypoint cannot be addressed to one channel or sent as a direct message.

### Waypoint Properties

| Property | Description |
|---|---|
| Name | Short identifier (max 29 characters) |
| Description | Optional longer description |
| Icon | Visual marker emoji on the map |
| Locked | If locked, only the creator can edit or delete |
| Expiration | Optional auto-remove date and time |
| Geofence | Optional enter/exit alert area — see Waypoint Geofences |

## Waypoint Expiration

Waypoints can be set to expire automatically:

- **Never** (default) — waypoint remains until manually deleted
- **Timed** — pick a specific date and time; the waypoint is automatically removed once that time passes. Useful for temporary markers like rally points, hazards, or meeting locations.

Expired waypoints are automatically hidden from the map so they don't clutter the display. The expiration countdown is based on the absolute time you picked, not a duration from when the waypoint was created or received.

## Waypoint Geofences

Any waypoint can also define a **geofence** — an alert area — so you or others get notified when a node enters or leaves it:

1. Set a **geofence radius** from the preset chips (or **Off** to disable), or tap **Set area on map** to draw a custom rectangular area instead.
2. Once a region is set, toggle **Notify on enter** and/or **Notify on exit**.
3. Optionally enable **Favorites only** to limit alerts to your favorited nodes.

Since waypoints (and their geofences) are broadcast to the whole mesh, only the **creator** is alerted by default. If someone else shares a geofenced waypoint with you, its detail view offers a **Notify me of crossings** opt-in so you can also receive enter/exit alerts for it.

## Managing Waypoints

- Tap a waypoint to see its name, description, and geofence radius. On **Google Play** builds the first tap opens the marker's info bubble — tap the bubble to open the waypoint itself
- **Locked waypoints** can only be changed on the mesh by the node that locked them
- Unlocked waypoints can be edited by any mesh member while connected to a radio — saving re-broadcasts the waypoint
- Confirming a delete removes your own copy. To remove it from everyone else's map too, select **Delete for everyone** in the delete dialog; that box appears only for a waypoint

you may change (unlocked, or locked by you) and only while you are connected

# Map Layers

Tap the layers icon on the map to open **Manage Map Layers**. It imports your own overlays in `.kml`, `.kmz`, or GeoJSON format, including KMZ ground overlays (georeferenced images, such as exported topo or aerial tiles), which drape at their stated bounds. Add one by picking a file with **Add Layer**, opening a file with Meshtastic, or sharing it into the app from another app. **Add Network Layer** instead takes a name and an `https://` URL pointing at a KML or GeoJSON file (`http://` also works on Desktop, but on Android only for `localhost`); that layer then carries its own refresh button in the sheet. On **Google Play** builds the toolbar's refresh button re-fetches every visible network layer at once.

Imported layers are listed with a toggle to show/hide each one and an option to remove it. Each layer — imported or built-in overlay — carries its own opacity slider while it is switched on, so an overlay can be faded back rather than only switched off. This works on the Google Play build, the F-Droid build, and **Desktop**, which shares the same layer store and file picker.

## Site Planner

**Site Planner** estimates RF coverage for a transmitter and draws it on the map as a color-coded overlay. Open it from a map control, or from a node's detail page via **Estimate coverage** (shown only for nodes with a known position). Configure the transmitter (location, frequency, TX power, antenna gain and height), the receiver (sensitivity, height), and simulation options (max range, high-resolution terrain, color palette), then run the estimate. Like map layers, Site Planner works on both the Google Play and F-Droid builds, where the finished estimate is drawn on the map as a coverage overlay. On **Desktop** the same form is shown but the planner opens in your browser; to bring the estimate onto the map, click the transmitter pin in the browser, choose the planner's GeoJSON export, then add the downloaded file under **Manage Map Layers** with **Add Layer**. Use the GeoJSON export, not the KML one — the KML is a ground-overlay image this map cannot draw.

# Position Sharing

## Enabling Position Sharing

Your node shares its GPS position based on:

- **Broadcast Interval** — share the position on a fixed timer
- **Smart Position** — share only once you have moved far enough; **Smart Interval** sets the shortest gap between broadcasts and **Smart Distance** how far you must move
- **Fixed Position** — publish a latitude, longitude, and altitude you enter by hand instead of the GPS reading
- **GPS Mode (Physical Hardware)** — GPS enabled, disabled, or not present on this hardware; offered only while **Fixed Position** is off

Configure position behavior in **Settings → Device configuration → Position**. The screen is only reachable while your radio is connected, and saving it reboots the radio. For the full field list, see Settings — Radio & User.

### Privacy Considerations

> 🔒 **Privacy:** Position data is broadcast to all nodes on your channel. If you don't want your location shared, disable GPS position in settings or use a fixed/fake position. To keep sharing a position without pinpointing yourself, edit the channel in **Settings → Channels**, turn **Precise location** off, and set the slider beneath it — the channel then publishes an approximate area, shown as ± a distance, instead of an exact point.

## Map Sources

Every build offers a base map picker from the map toolbar. **Google Play** builds open on Google's own map types; **F-Droid** and **Desktop** builds open on MapLibre's vector styles. Further down the base map picker, all three offer the same raster base maps:

| Base map | Notes |
|---|---|
| Normal / Satellite / Terrain / Hybrid | Google Play only — Google's own map types |
| Liberty | Default on F-Droid and Desktop. Vector street map |
| Positron | F-Droid and Desktop only. Low-contrast vector map; keeps node markers legible over it |
| Dark | F-Droid and Desktop only. Vector map suited to dark themes |
| OpenStreetMap | Classic raster street tiles |
| OpenTopoMap | Raster topographic |
| USGS Topo / USGS Imagery | US coverage only |
| Esri Topo / Esri Imagery | Topographic and satellite imagery |

Overlays can be toggled on top of any base map, from the layers sheet:

- **Weather radar** — NOAA NEXRAD reflectivity (US coverage)
- **Hillshade** — terrain relief from an online tile source, on **F-Droid** and **Desktop** only; Google Play draws hillshade from downloaded offline terrain instead. Useful for understanding why a link fails, since LoRa range is limited by terrain

### Adding your own tile source

Any XYZ tile endpoint can be added as a base map, on every flavor and on desktop. Open **Manage Custom Tile Sources** at the foot of the base map picker and paste a URL template using `{z}`, `{x}` and `{y}` — plus `{s}` if the provider uses rotating subdomains. A national mapping service, for example:

```
https://wmts.geo.admin.ch/1.0.0/ch.swisstopo.pixelkarte-
farbe/default/current/3857/{z}/{x}/{y}.jpeg
```

On **Android** the template must use `https://`; plain `http://` works only for `localhost`.

Tiles are cached on disk, so panning does not re-download what you were just looking at.

On **Android**, the same screen also imports a local `.mbtiles` archive for fully offline use.

On **F-Droid**, select a vector base map first — Liberty, Positron, or Dark — since a download is defined against a vector style and **Start Download** stays disabled over a raster one. Frame the area you want on screen, then tap **Start Download** in the layers sheet: that creates a paused pack covering the current zoom plus two levels deeper. Press play on the pack's row to actually download it. If the offline database cannot be opened, the section says offline maps are unavailable and **Start Download** stays disabled.

On **Google Play**, the layers sheet's **Offline Manager** section downloads its own offline regions instead — water, roads and administrative boundaries extracted from the public Protomaps basemap dataset, drawn directly as map shapes rather than raster tiles. Frame the area you want, tap **Start Download**, and once it finishes flip **Show on map** to draw it (it's off by default so it never surprises you by covering up the live map). You can still import a pre-made `.mbtiles` archive here too, as before. **Desktop** has neither, though the offline terrain described next does work there.

### Offline terrain

Hillshade and elevation contours can be downloaded for offline use as well, so a downloaded area still shows why a link fails when there is no network. The terrain data comes from Mapterhorn and is credited on the map. On **Google Play**, each downloaded region in the **Offline Manager** list has a **Download Terrain** button and **Hillshade** and **Contours** switches; terrain attaches to that region and shares its storage limit, and an area too large for terrain asks you to zoom in or pick a smaller region. On **F-Droid** and **Desktop**, the layers sheet has an **Offline Terrain** section of its own: frame the area, tap **Start Download**, and the hillshade and contours draw from the downloaded tiles from then on.

### Going offline

An **Offline** pill appears over the map whenever the device has no network connection. On **Google Play** builds, if you've imported a local `.mbtiles` archive, the map switches to it automatically the moment the network drops — no toggle to remember — and switches back once you're reconnected, as long as you haven't picked a different base map yourself in between. On **F-Droid**, a downloaded offline area keeps rendering on its own; the pill is purely informational there. **Desktop** downloads offline terrain but no base-map packs, so the pill is informational there too.

## Related Topics

- Nodes — view and filter your node list
- Node Metrics — signal quality and position history for individual nodes
- Local Mesh Discovery — traceroute and neighbor info for understanding mesh topology
- Units & Locale — distance and coordinate display formats

# The phone app (Apple)

## The app for iPhone, iPad and Mac

From https://meshtastic.org/docs/software/apple/

User and developer documentation for the Meshtastic iOS, iPadOS, macOS, watchOS, and visionOS app.

Use the sidebar navigation to browse the **User Guide** for app features and the **Developer Guide** for contributing to the project. Each guide opens on its own **What's New** page listing the last 12 months of changes.

---

### Quick Links

| Guide | Description |
|---|---|
| What's New | Recent user-facing changes from the last 12 months |
| Getting Started | Connect your first radio and send a message |
| Nodes List | Understanding the mesh network node list |
| Signal Meter | How the LoRa signal quality meter works |
| Units & Locale | How temperatures, distances, and times adapt to your region |
| Architecture | App architecture overview for contributors |

---

In **Ask Chirpy**, local source links open in the in-app documentation viewer. When internet is available, Chirpy may also augment answers with related `meshtastic.org` sources; without service, it continues using local docs only.

# Settings

## LoRa: region, preset and hops

From https://meshtastic.org/docs/configuration/radio/lora/

The LoRa config options are: Region, Use Preset, Modem Preset, Max Hops, Transmit Power, Bandwidth, Spread Factor, Coding Rate, Frequency Offset, Transmit Enabled, Frequency Slot, Ignore Incoming Array, Ignore MQTT, Override Duty Cycle Limit, SX126x RX Boosted Gain, Override Frequency, and PA Fan Disabled. LoRa config uses an admin message sending a `Config.LoRa` protobuf.



note

In order to communicate fully, devices within a mesh must have identical settings for Region and Modem Preset, or identical custom Modem settings.

## LoRa Config Values



note

You must set your device's `lora.region` setting. This will ensure that you are operating within the legal limits for your area.

## Region

Sets the region for your node. Default is `unset`. As long as this is not set, the node screen will display a message and not transmit any packets.

| Region Code | Description | Frequency Range (MHz) | Duty Cycle (%) | Power Limit (dBm) |
|---|---|---|---|---|
| UNSET | Unset | N/A | N/A | N/A |
| US | United States | 902.0 - 928.0 | 100 | 30 |
| EU_433 | European Union 433 MHz | 433.0 - 434.0 | 10 | 10 |
| EU_868 | European Union 868 MHz | 869.4 - 869.65 | 10 | 27 |
| EU_866 | European Union 866 MHz | 865.6 - 867.6 | 2.5 / 10 | 27 |
| EU_N_868 | European Union 868 MHz (Narrow) | 869.4 - 869.65 | 10 | 27 |
| CN | China | 470.0 - 510.0 | 100 | 19 |
| JP | Japan | 920.5 - 923.5 | 100 | 13 |
| ANZ | Australia & New Zealand | 915.0 - 928.0 | 100 | 30 |
| ANZ_433 | Australia & New Zealand | 433.05 - 434.79 | 100 | 14 |
| KR | Korea | 920.0 - 923.0 | 100 | 23 |
| TW | Taiwan | 920.0 - 925.0 | 100 | 27 |
| RU | Russia | 868.7 - 869.2 | 100 | 20 |
| IN | India | 865.0 - 867.0 | 100 | 30 |
| NZ_865 | New Zealand 865 MHz | 864.0 - 868.0 | 100 | 36 |
| TH | Thailand | 920.0 - 925.0 | 10 | 27 |
| UA_433 | Ukraine 433 MHz | 433.0 - 434.7 | 10 | 10 |
| MY_433 | Malaysia 433 MHz | 433.0 - 435.0 | 100 | 20 |
| MY_919 | Malaysia 919 MHz | 919.0 - 924.0 | 100 | 27 |
| SG_923 | Singapore 923 MHz | 917.0 - 925.0 | 100 | 20 |
| KZ_433 | Kazakhstan 433 MHz | 433.075 - 434.775 | 100 | 10 |
| KZ_863 | Kazakhstan 863 MHz | 863.0 - 868.0 | 100 | 30 |
| BR_902 | Brazil 902 MHz | 902.0 - 907.5 | 100 | 30 |
| PH_433 | Philippines 433 MHz | 433.0 - 434.7 | 100 | 10 |
| PH_868 | Philippines 868 MHz | 868.0 - 869.4 | 100 | 14 |
| PH_915 | Philippines 915 MHz | 915.0 - 918.0 | 100 | 24 |
| NP_865 | Nepal 865 MHz | 865.0 - 868.0 | 100 | 30 |
| ITU1_2M | ITU Region 1 Amateur 2m | 144.0 - 146.0 | 100 | 30 |
| ITU2_2M | ITU Region 2 Amateur 2m | 144.0 - 148.0 | 100 | 30 |
| ITU3_2M | ITU Region 3 Amateur 2m | 144.0 - 148.0 | 100 | 30 |

| Region Code | Description | Frequency Range (MHz) | Duty Cycle (%) | Power Limit (dBm) |
|---|---|---|---|---|
| ITU2_125CM | ITU Region 2 Amateur 1.25m | 220.0 - 225.0 | 100 | 30 |
| ITU1_70CM | ITU Region 1 Amateur 70cm | 430.0 - 440.0 | 100 | 30 |
| ITU2_70CM | ITU Region 2 Amateur 70cm | 420.0 - 450.0 | 100 | 30 |
| ITU3_70CM | ITU Region 3 Amateur 70cm | 430.0 - 450.0 | 100 | 30 |
| LORA_24 | 2.4 GHz band worldwide | 2400.0 - 2483.5 | 100 | 10 |

EU_433 and EU_868 are limited to a 10% duty cycle, calculated every minute over a rolling hour. A node that reaches the limit stops transmitting until it's allowed again. EU_866 is limited to 2.5%, or 10% for a node in the ROUTER or ROUTER_LATE role.

EU_868, EU_866, and EU_N_868 cover the same European band with different channel plans, and each accepts only its own presets. Selecting a preset that belongs to one of the others switches the region to it.

The ITU1_*, ITU2_*, and ITU3_* regions are amateur radio allocations and can only be selected in licensed mode. **Do not use them without an amateur radio license.** Review the privileges and restrictions of operating under an amateur license first: encryption is not permitted, and you must transmit your call sign.

The listed power limit is the ceiling the firmware applies, not your national limit. Some countries do not allocate the full range shown, for example 220–222 MHz in the USA and Canada. Check your national band plan before transmitting.

## Use Preset

When enabled, the Modem Preset fields will be adhered to, else the Bandwidth/Spread Factor/Coding Rate will be taken from their respective manually defined fields.

## Modem Preset

Default is unset which equates to LONG_FAST. Presets are pre-defined modem settings (Bandwidth, Spread Factor, and Coding Rate) which influence both message speed and range. The default will provide a strong mixture of speed and range, for most users.

Nodes only hear each other when they share a preset, and where the primary channel name is unchanged the preset also determines the frequency slot. Joining your local mesh covers how to find the preset in use near you.

A US node whose region is first set from its own screen starts on LONG_TURBO instead. LONG_TURBO and LONG_FAST nodes can't hear each other, so set the preset to LONG_FAST to join a mesh that uses it.

The presets are designed to provide further options for optimizing either speed (and reduced network congestion) or range, which can be useful for two real world scenarios:

1. A high number of devices exist in the mesh, or messages are sent very frequently. Faster speeds (and therefore lower radio time per device) can help with mesh network congestion.
2. Maximum range is desired, for long range scenarios where a several second delay in message receipt is acceptable (for instance, attempting to send messages from a town to a distant mountain top).

Not every preset is legal in every band. The node accepts only the presets permitted in the selected Region, and tells connected clients which presets each region allows. A preset the region doesn't permit is replaced with the region's default. When the change comes from remote administration, the whole change is rejected instead.

The general-purpose presets are as follows, and follow a linear pattern of Fastest <--> Slowest, and Shortest <--> Longest range:

1. `SHORT_TURBO` (Fastest, highest bandwidth, lowest airtime, shortest range. It is not legal to use in all regions due to its 500 kHz bandwidth.)
2. `SHORT_FAST`
3. `SHORT_SLOW`
4. `MEDIUM_TURBO` (500 kHz bandwidth, so it carries the same regional restriction as `SHORT_TURBO`.)
5. `MEDIUM_FAST`
6. `MEDIUM_SLOW`
7. `LONG_TURBO` (Performs similarly to `LONG_FAST`, but with 500 kHz bandwidth.)
8. `LONG_FAST` (Default)
9. `LONG_MODERATE`

`EU_868` offers this list without the three Turbo presets, as 500 kHz bandwidth is not permitted in the 869.4 - 869.65 MHz band.

### Narrowband presets

Three narrowband preset families serve bands that can't accommodate the general-purpose presets. Each family is exclusive to its regions and can't be selected anywhere else:

| Presets | Bandwidth | Available in | Default |
|---|---|---|---|
| `LITE_FAST`, `LITE_SLOW` | 125 kHz | `EU_866` | `LITE_FAST` |
| `NARROW_FAST`, `NARROW_SLOW` | 62.5 kHz | `EU_N_868`, `ITU1_70CM`, `ITU2_70CM`, `ITU3_70CM`, `ITU2_125CM` | `NARROW_SLOW` |
| `TINY_FAST`, `TINY_SLOW` | 15.6 kHz | `ITU1_2M`, `ITU2_2M`, `ITU3_2M` | `TINY_FAST` |

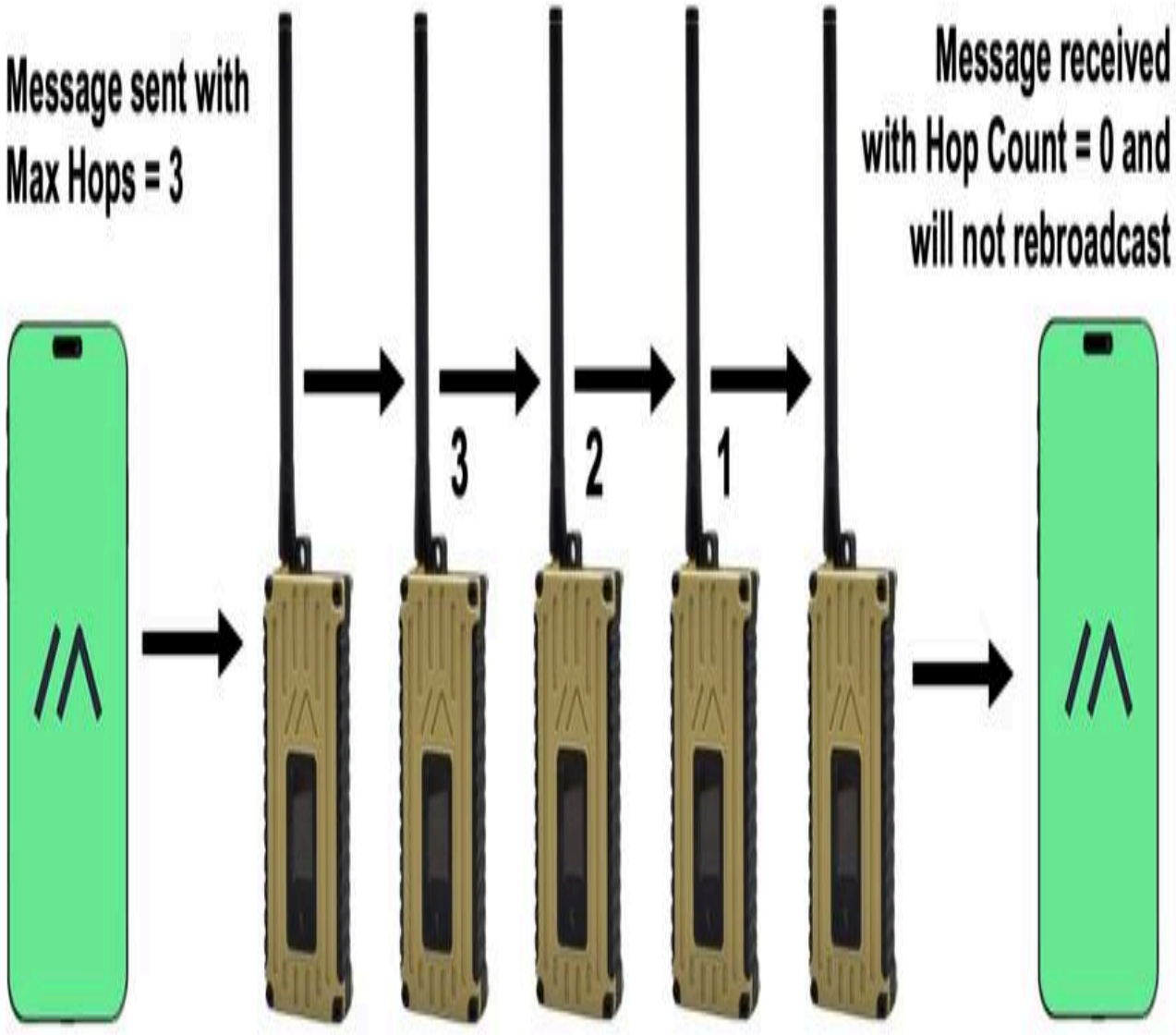
See Narrowband presets for their data rates and link budgets.

`EU_868`, `EU_866`, and `EU_N_868` share the same regulatory domain, so selecting a Lite or Narrow preset while the region is `EU_868` switches the region to `EU_866` or `EU_N_868` respectively. Selecting a general-purpose preset that `EU_868` allows switches it back. The Turbo presets aren't allowed in any of the three.

The Tiny presets transmit in only 15.6 kHz of bandwidth, so **they require a TCXO with a tolerance of ±5 ppm or better.** They work only with SX127x and SX126x radios.

### Max Hops

Maximum number of hops. This can't be greater than 7. Default is 3 which should be fine for most applications. ***Really, 3 is fine.***



## Transmit Power

If zero, then use default max legal continuous power (i.e. something that won't burn out the radio hardware)

In most cases you should use zero here. Units are in dBm.

## Bandwidth

Certain bandwidth numbers are 'special' and will be converted by the device firmware to the appropriate floating point value:

| Special Value | Interpreted As |
|---|---|
| 8 | 7.8 kHz |
| 10 | 10.4 kHz |
| 16 | 15.6 kHz |
| 21 | 20.8 kHz |
| 31 | 31.25 kHz |
| 42 | 41.7 kHz |
| 62 | 62.5 kHz |
| 200 | 203.125 kHz |
| 400 | 406.25 kHz |
| 800 | 812.5 kHz |
| 1600 | 1625.0 kHz |

Any other value is used as-is, in kHz.

Please be aware that values < 62.5 kHz may require a TCXO on some hardware devices. The `TINY_FAST` and `TINY_SLOW` presets use `16` (15.6 kHz) and require one with a tolerance of ±5 ppm or better.

## Spread Factor

A number from 5 to 12. Indicates the number of chirps per symbol as 1[<<]spread_factor.

Older LoRa chips like (SX127x / RF95) only support spreading factors 7 to 12.

## Coding Rate

The denominator of the coding rate. ie for 4/5, the value is 5. 4/8 the value is 8.

## Frequency Offset

This parameter is for advanced users with advanced test equipment, we do not recommend most users use it.

A frequency offset that is added to the calculated band center frequency. Used to correct for crystal calibration errors.

### Transmit Enabled

Allows you to enable and disable transmit (TX) from the LoRa radio. Useful for hot-swapping antennas and other tests.

Defaults to true

### Frequency Slot

This setting controls the actual hardware frequency at which the radio transmits, represented by a frequency slot between 1 and NUM_SLOTS (the maximum for the current region and modem preset). If set to `0`/UNSET, the device reverts to the older channel name hash-based algorithm for determining the frequency slot.

### Ignore Incoming Array

For testing it is useful sometimes to force a node to never listen to particular other nodes (simulating radio out of range). All nodenums listed in the ignore_incoming array will have packets they send dropped on receive (by router.cpp)



note

This option is **deprecated** in the client apps. To ignore nodes, long-press on (or click for web client) a node and mark it as ignored. This acts the same as the ignore list (blocking packets from the ignore node from being received or relayed), with the added benefit of being able to ignore more than three nodes.

## Ignore MQTT

Setting this to option to 'true' means the device will ignore any messages it receives via LoRa that came via MQTT somewhere along the path towards the device. Note this only works when your device and the MQTT node are running at least firmware version 2.2.19.

### OK to MQTT

Acceptable values: `true`, `false`

Default is `false`. When set to `true`, this configuration indicates that the user approves their packets to be uplinked to MQTT brokers. If set to `false`, nodes receiving your packets are requested not to forward packets to MQTT. This configuration only applies to Channels configured with the `defaultpsk` and `eventpsk` keys set in the Meshtastic Firmware; Channels with custom keys ignore this setting.

**Important:** This is not a cryptographic solution but a polite request that is enforced in the official firmware.

### Override Duty Cycle Limit

Setting this option to 'true' means the device will ignore the hourly duty cycle limit in Europe. This means that you might violate regulations if the device transmits too much. By default, this option is set to 'false,' which means the device will stop sending data when it reaches the hourly limit and will start again when it is allowed to do so.

### SX126x RX Boosted Gain

This is an option specific to the SX126x chip series which allows the chip to consume a small amount of additional power to increase RX sensitivity.

### Override Frequency

This parameter is for advanced users and licensed HAM radio operators. When enabled, the channel calculation will be ignored, and the set frequency will be used instead (frequency_offset still applies). This will allow you to use out-of-band frequencies. Please respect your local laws and regulations. If you are a license HAM operator, make sure you enable HAM mode and turn off encryption.

### PA Fan Disabled

If true, disable the built-in PA FAN using pin define in RF95_FAN_EN.

## LoRa Config Client Availability

**Android Apple CLI Web**

### Android



info

LoRa Config options are available on Android.

1. Open the Meshtastic App
2. Navigate to: **Settings > LoRa**

**Apple CLI Web**

## Apple



info

> All LoRa config options are available on iOS, iPadOS and macOS at Settings > Radio Configuration > LoRa.

**CLI Web**

## CLI



info

LoRa config commands are available in the python CLI. Example commands are below:

| Setting | Acceptable Values | Default |
|---|---|---|
| lora.modem_preset | LONG_FAST, LONG_MODERATE, LONG_TURBO, MEDIUM_SLOW, MEDIUM_FAST, MEDIUM_TURBO, SHORT_SLOW, SHORT_FAST, SHORT_TURBO, LITE_FAST, LITE_SLOW, NARROW_FAST, NARROW_SLOW, TINY_FAST, TINY_SLOW (availability depends on the selected region) | LONG_FAST |
| lora.use_preset | false, true | true |
| lora.region | UNSET, US, EU_433, EU_868, EU_866, EU_N_868, CN, JP, ANZ, ANZ_433, KR, TW, RU ,IN, NZ_865, TH, LORA_24, UA_433, MY_433, MY_919, SG_923, KZ_433, KZ_863, BR_902, NP_865, PH_433, PH_868, PH_915, ITU1_2M, ITU2_2M, ITU3_2M, ITU2_125CM, ITU1_70CM, ITU2_70CM, ITU3_70CM | UNSET |
| lora.bandwidth | 8, 10, 16, 21, 31, 42, 62, 125, 250, 500 | 250 |
| lora.spread_factor | 5 ,6, 7, 8, 9, 10, 11, 12 | 11 |
| lora.coding_rate | 4, 5, 6, 7, 8 | 5 |
| lora.frequency_offset | 0 to 1000000 | 0 |
| lora.hop_limit | 1,2,3,4,5,6,7 | 3 |
| lora.tx_power | 0 to 30 | 0 |
| lora.tx_enabled | false, true | true |
| lora.channel_num | 0, 1 to NUM_CHANNELS | 0 |
| lora.ignore_mqtt | false, true | false |
| lora.config_ok_to_mqtt | true, false | false |
| lora.override_duty_cycle | false, true | false |
| lora.sx126x_rx_boosted_gain | false, true | false |
| lora.override_frequency | Any supported frequency the LoRA radio is capable of. Please respect local rules and regulations | 0 |



tip

> LoRa settings apply without a reboot. To set several values at once, chain them in one command.
>
> Example:
>
> ```
> meshtastic --set lora.region US --set lora.modem_preset LONG_FAST
> ```

Set Modem Preset

```
meshtastic --set lora.modem_preset LONG_FAST
meshtastic --set lora.modem_preset MEDIUM_FAST
```

Set Region

```
meshtastic --set lora.region US
meshtastic --set lora.region EU_433
```

Set Hop Limit

```
meshtastic --set lora.hop_limit 2
```

Override Duty Cycle

```
meshtastic --set lora.override_duty_cycle true
meshtastic --set lora.override_duty_cycle false
```

# Channels

From https://meshtastic.org/docs/configuration/radio/channels/

The Channels config options are: Index, Roles, and Settings. Channel config uses an admin message sending a `Channel` protobuf which also consists of a `ChannelSettings` or `ModuleSettings` protobuf.



info

> **Channel Settings** (as described on this page) should not be confused with Modem Preset Settings
>
> Modem Preset Settings contain the modem configuration (frequency settings, spreading factor, bandwidth, etc.) used for the LoRa radio. These settings are identical for all channels and can **not** be unique per channel.
>
> **Channel Settings** contain information for segregating message groups, configuring optional encryption, and enabling or disabling messaging over internet gateways. These settings **are** unique and configurable per channel.

## Channel Config Values

### Index

The channel index begins at 0 and ends at 7.

*Indexing* cannot be modified.

| Index | Channel | Default Role | Purpose |
|---|---|---|---|
| 0 | 1 | `PRIMARY` | Used as `default` channel |
| 1 | 2 | `DISABLED` | User defined |
| 2 | 3 | `DISABLED` | User defined |
| 3 | 4 | `DISABLED` | User defined |
| 4 | 5 | `DISABLED` | User defined |
| 5 | 6 | `DISABLED` | User defined |
| 6 | 7 | `DISABLED` | User defined |
| 7 | 8 | `DISABLED` | User defined |



note

You can **not** have `DISABLED` channels in-between active channels such as `PRIMARY` and `SECONDARY`. Active channels must be consecutive.

## Role

Each channel is assigned one of 3 roles:

1. `PRIMARY` or `1`
    - This is the first channel that is created for you on initial setup.
    - Only one primary channel can exist and cannot be disabled.
    - By default, periodic broadcasts like position and telemetry are sent over this channel.
2. `SECONDARY` or `2`
    - Can modify the encryption key (PSK).
3. `DISABLED` or `0`
    - The channel is no longer available for use.
    - The channel settings are set to default.



note

While you can have a different PRIMARY channel and communicate over SECONDARY channels with the same Name & PSK, a hash of the PRIMARY channel's name sets the LoRa frequency slot, which determines the actual frequency you are transmitting on in the band. To ensure devices with different PRIMARY channel name transmit on the same frequency, you must explicitly set the LoRa frequency slot.

Unless the frequency slot is set explicitly, renaming the PRIMARY channel moves a node off the frequency slot the rest of its area is using. Keeping private groups on SECONDARY channels avoids that, as covered in Joining your local mesh.

## Channel Settings Values

The Channel Settings options are: Name, PSK, Use AEAD, Downlink Enabled, Uplink Enabled, and Mute. Channel settings are embedded in the `Channel` protobuf as a `ChannelSettings` protobuf and sent as an admin message.

### Name

A short identifier for the channel. *(< 12 bytes)*

| Reserved Name | Purpose |
|---|---|
| "" (default) | If left empty on the Primary channel, this designates the `default` channel. |
| `admin` | On Secondary channels, the name `admin`, in any letter case, designates the `admin` channel used to administer nodes over the mesh. It works only when Admin Channel Enabled is on. Note that this is a Legacy feature, see Remote Admin for details. |



note

Matching channel names are required in order to communicate on the same channel with other devices. Example: If your device is using the channel name `LongFast` the device you are attempting to communicate with must also have a channel named `LongFast`.

## PSK

The encryption key used for private channels.

Hex byte `0x01` for the Primary `default` channel.

Must be either 0 bytes (no crypto), 16 bytes (AES128), or 32 bytes (AES256).



note

Matching PSKs are required in order to communicate on the same channel with other devices. Example: If your device is using a channel with the default PSK of `AQ==` the device

> you are attempting to communicate with must also have a matching channel with the same PSK.

### Use AEAD

On firmware 2.8.1 and later, `use_aead` switches the channel from AES-CTR to authenticated encryption, AES-CCM. Each packet carries a 12-byte tag, and nodes reject any packet that was altered or forged by someone without the key.

Every node on the channel must turn it on. Nodes with it on and nodes with it off can't read each other's messages on that channel, even with the same name and PSK. The setting needs a PSK, is experimental, and is off by default.

### Downlink Enabled

If enabled, messages captured from a **public** internet gateway will be forwarded to the local mesh.

Set to `false` by default for all channels.

### Uplink Enabled

If enabled, messages from the mesh will be sent to the **public** internet through any node's configured gateway.

Set to `false` by default for all channels.

## Channel Config Client Availability

## Channel Module Settings

The channel module settings options are: Is muted and position precision. Channel module settings are embedded in the Channel protobuf as a ModuleSettings protobuf and sent as an admin message.

### Is Muted

If enabled, the `is_muted` setting silences invocation of the `ExternalNotificationModule` when messages are received over the specified channel(s). UI notifications such as `"New Message From []"` are also blocked from popup.

Set to `false` by default for all channels.

Control the channel's notification behaviour.

Toggle is_muted on the PRIMARY channel

```
meshtastic --ch-set module_settings.is_muted true --ch-index 0
meshtastic --ch-set module_settings.is_muted false --ch-index 0
```

## Position Precision

The `position_precision` setting allows control of the level of precision for location data that is sent over a particular channel. This can be useful for privacy reasons, where obfuscating the exact location may be desired when sending position data over certain channels.

The `position_precision` value is an integer between 0 and 32:

- A value of 0 means that location data is never sent over the given channel.
- A value of 32 means that location data is sent with full precision.
- On a channel whose key is public, which means no key, the default key, or any single-byte key, the firmware limits precision to 15 bits.
- Values in between indicate the number of bits of precision to be sent, which correspond to a position precision from the table below.
- The public MQTT server filters out precise positions.

| Precision bits | Metric | Imperial |
|---|---|---|
| 10 | 23.3 km | 14.5 miles |
| 11 | 11.7 km | 7.3 miles |
| 12 | 5.8 km | 3.6 miles |
| 13 | 2.9 km | 1.8 miles |
| 14 | 1.5 km | 4787 feet |
| 15 | 729 m | 2392 feet |
| 16 | 364 m | 1194 feet |
| 17 | 182 m | 597 feet |
| 18 | 91 m | 299 feet |
| 19 | 45 m | 148 feet |

The client applications have implemented different levels of precision giving the user a practical range to choose from. Setting across the full range of integers can be done via the Python CLI. See Setting Position Precision for examples on setting different levels of precision using CLI.

**Android Apple CLI Web**

## Android



info

All Channel config options are available for Android.

1. Open the Meshtastic App
2. Navigate to: **Settings > Channels**

**Apple CLI Web**

## Apple



info

> A channel editor is available on the iOS, iPadOS and macOS applications at Settings > Radio Configuration > Channels.

**CLI Web**

## CLI



info

All Channel config options are available in the python CLI. Example commands are below:



tip

> Because the device will reboot after each command is sent via CLI, it is recommended when setting multiple values in a config section that commands be chained together as one.
>
> Example:
>
> ```
> meshtastic --ch-set name "My Channel" --ch-set psk random --ch-set uplink_enabled true --
> ch-index 4
> ```

### Name

Set channel name for the PRIMARY channel

```
# without spaces
meshtastic --ch-set name MyChannel --ch-index 0
# with spaces
meshtastic --ch-set name "My Channel" --ch-index 0
```

### PSK

If you use Meshtastic for exchanging messages you don't want other people to see, `random` is the setting you should use. Selecting `default` or any of the `simple` values from the following table will use publicly known encryption keys. They're shipped with Meshtastic source code and thus, anyone can listen to messages encrypted by them. They're great for testing and public channels.

| Setting | Behavior |
|---|---|
| `none` | Disable Encryption |
| `default` | Default Encryption (use the weak encryption key) |
| `random` | Generate a secure 256-bit encryption key. Use this setting for private communication. |
| `simple0`- `simple254` | Uses a single byte encoding for encryption |

Set encryption to default on PRIMARY channel

```
meshtastic --ch-set psk default --ch-index 0
```

Set encryption to random on PRIMARY channel

```
meshtastic --ch-set psk random --ch-index 0
```

Set encryption to single byte on PRIMARY channel

```
meshtastic --ch-set psk simple15 --ch-index 0
```

Set encryption to your own key on PRIMARY channel

```
meshtastic --ch-set psk 0x1a1a1a1a2b2b2b2b1a1a1a1a2b2b2b2b1a1a1a1a2b2b2b2b1a1a1a1a2b2b2b2b --
ch-index 0
```

Set encryption to your own key on PRIMARY channel (Base64 encoded)

```
meshtastic --ch-set psk base64:puavdd7vtYJh8NUVWgxbsoG2u9Sdqc54YvMLs+KNcMA= --ch-index 0
```



tip

> Use this to copy and paste the `base64` encoded (single channel) key from the meshtastic --info command. Please don't use the omnibus (all channels) code here, it is not a valid key.

Disable encryption on PRIMARY channel

```
meshtastic --ch-set psk none --ch-index 0
```

### Uplink / Downlink

For configuring gateways, please see MQTT

Enable/Disable Uplink on PRIMARY channel

```
meshtastic --ch-set uplink_enabled true --ch-index 0
meshtastic --ch-set uplink_enabled false --ch-index 0
```

Enable/Disable Downlink on SECONDARY channel

```
meshtastic --ch-set downlink_enabled true --ch-index 1
meshtastic --ch-set downlink_enabled false --ch-index 5
```

### Setting Position Precision



info

This is a per-channel setting. The `--ch-index` parameter must be specified to set the position precision for a specific channel, e.g., `--ch-index 0` for the primary channel or `--ch-index 1` for the secondary channel 1.

Set position precision to 13 bits (approx ±3 km)

```
meshtastic --ch-set module_settings.position_precision 13 --ch-index 0
```

Set position precision to full precision (32 bits)

```
meshtastic --ch-set module_settings.position_precision 32 --ch-index 1
```

# Device: roles

From https://meshtastic.org/docs/configuration/radio/device/

The device config options are: Role, Rebroadcast Mode, GPIO for User Button, GPIO for PWM Buzzer, Node Info Broadcast Seconds, Double Tap as Button Press, Timezone Definition, and LED Heartbeat Disabled. Device config uses an admin message sending a `Config.Device` protobuf.

## Device Config Values

## Roles

| Device Role | Description | Best Uses |
|---|---|---|
| CLIENT | App connected or stand alone messaging device. Rebroadcasts packets when no other node has done so. | General use for individuals needing to communicate over the Meshtastic network with support for client applications. |
| CLIENT_MUTE | Device that does not forward packets from other devices. | Situations where a device needs to participate in the network without assisting in packet routing, reducing network load. |
| CLIENT_HIDDEN | Device that only broadcasts as needed for stealth or power savings. | Use in stealth/hidden deployments or to reduce airtime/power consumption while still participating in the network. |
| CLIENT_BASE | Personal base station: always rebroadcasts packets from or to its favorited nodes. Handles all other packets like CLIENT. | Use for stronger attic/roof “base station” nodes to distribute messages more widely from your own weaker, indoor, or less-well-positioned nodes. |
| TRACKER | Broadcasts GPS position packets as priority. | Tracking the location of individuals or assets, especially in scenarios where timely and efficient location updates are critical. |
| LOST_AND_FOUND | Broadcasts location as message to default channel regularly to assist with device recovery. | Used for recovery efforts of a lost device. |
| SENSOR | Broadcasts telemetry packets as priority. | Deploying in scenarios where gathering environmental or other sensor data is crucial, with efficient power usage and frequent updates. |
| TAK | Optimized for ATAK system communication, reduces routine broadcasts. | Integration with ATAK systems (via the Meshtastic ATAK Plugin) for communication in tactical or coordinated operations. |

| Device Role | Description | Best Uses |
| --- | --- | --- |
| TAK_TRACKER | Enables automatic TAK PLI broadcasts and reduces routine broadcasts. | Standalone PLI integration with ATAK systems for communication in tactical or coordinated operations. |
| REPEATER [1] | Infrastructure node for extending network coverage by always rebroadcasting packets once with minimal overhead. Not visible in Nodes list. | Best positioned in strategic locations to maximize the network's overall coverage. Device is not shown in topology. |
| ROUTER | Infrastructure node for extending network coverage by always rebroadcasting packets once. Visible in Nodes list. | Best positioned in strategic locations to maximize the network's overall coverage. Device is shown in topology. |
| ROUTER_LATE | Infrastructure node that always rebroadcasts packets once but only after all other modes, ensuring additional coverage for local clusters. Visible in Nodes list. More details in our blog: Demystifying Router Late | Ideal for covering dead spots or ensuring reliability for a cluster of nodes where placement doesn’t benefit the broader mesh. Device is shown in topology. |



tip

> Still not sure which role to use? Check out our blog post: Choosing the Right Device Role

Looking for ROUTER_CLIENT? This role was deprecated in firmware 2.3.15. Learn more in the 2.3.15 release notes

### Role Comparison

This table shows the **default** values after selecting a preset. As always, individual settings can be adjusted after choosing a preset.

| Device Role | BLE/Wi-Fi/Serial | Screen Enabled | Power Consumption | Retransmit | Prioritized Routing | Visible in Nodes List |
|---|---|---|---|---|---|---|
| CLIENT | Yes | Yes | Regular | Yes | No | Yes |
| CLIENT_MUTE | Yes | Yes | Lowest | No | No | Yes |
| CLIENT_HIDDEN | Yes | Yes | Lowest | Local Only | No | No |
| CLIENT_BASE | Yes | Yes | Regular | Yes | Favorites Only[2] | Yes |
| TRACKER | Yes | No | Regular / Low | Awake Only[3] | No | Yes |
| LOST_AND_FOUND | Yes | No | Regular | Yes | No | Yes |
| SENSOR | Yes | No | Regular / Low | Awake Only[3] | No | Yes |
| TAK | Yes | Optional | Regular | Yes | No | Yes |
| TAK_TRACKER | Yes | Optional | Regular | Yes | No | Yes |
| ROUTER | No[4] | No | High | Yes | Yes | Yes |
| ROUTER_LATE | Yes | Yes | High | Yes | No | Yes |
| REPEATER | Yes | No | High | Yes | Yes | No |

## Notes about Power Saving + Role behavior

### ESP32 Platform Only

Every role except for the `TRACKER` and `SENSOR` roles are capable of traditional *sleep* behavior using the `power.is_power_saving` setting. When a node goes to sleep, in general, the serial console, peripherals, GPS, and Wi-Fi/Bluetooth radio are all shut down to conserve power. The CPU will enter a low power state (light sleep or deep sleep on the ESP32). The LoRA radio will be left on stand-by for receiving any packets which will *wake up* the device to resume services and respond.

Timeout configurations of `display.screen_on_secs` (Screen timeout), `power.min_wake_secs` (Minimum time till sleep), and `power.wait_bluetooth_secs` (Bluetooth timeout) will delay a node's

transition to a sleep state. The timeouts and subsequent transition to sleep states may also be *reset* by external events or actions such as receiving a packet, pressing a button, or connecting via client through serial or other means.

For the `ROUTER` role, the `power.is_power_saving` behavior of sleep is enabled automatically and cannot be turned off.

### ESP32 and NRF52 Platforms

In the `TRACKER` and `SENSOR` roles, there are special sleep behaviors when combined with `power.is_power_saving`. For `TRACKER`, the device will go to sleep for `position.position_broadcast_secs` and upon waking will send a position packet when the GPS provides a valid location and repeat the cycle. For `SENSOR`, the device will go to sleep for `telemetry.environment_update_interval` and upon waking will gather environment data and send a telemetry packet and repeat the cycle. For both of these roles, the LoRA radio does not remain in standby mode to accept packets and wake up the device. Sleep remains constrained to their respective transmission intervals.

## Rebroadcast Mode

This setting defines the device's behavior for how messages are rebroadcasted.

| Value | Description |
|---|---|
| `ALL` | ALL (Default) - This setting will rebroadcast ALL messages from its primary mesh as well as other meshes with the same modem settings, including when encryption settings differ. |
| `ALL_SKIP_DECODING` | ALL_SKIP_DECODING - Same as behavior as ALL, but skips packet decoding and simply rebroadcasts them. **Only available with Repeater role.** |
| `LOCAL_ONLY` | LOCAL_ONLY - Ignores observed messages from foreign meshes that are open or those which it cannot decrypt. Only rebroadcasts message on the nodes local primary / secondary channels. |
| `KNOWN_ONLY` | KNOWN_ONLY - Ignores observed messages from foreign meshes like LOCAL_ONLY, but takes it a step further by also ignoring messages from nodenums not in the node's known list (NodeDB). |
| `NONE` | NONE - Only permitted for SENSOR, TRACKER and TAK_TRACKER roles, this will inhibit all rebroadcasts, not unlike CLIENT_MUTE role. |
| `CORE_PORTNUMS_ONLY` | CORE_PORTNUMS_ONLY - Ignores packets from non-standard portnums such as: TAK, RangeTest, PaxCounter, etc. Only rebroadcasts packets with standard portnums: NodeInfo, Text, Position, Telemetry, and Routing. |

## GPIO for User Button

This is the GPIO pin number that will be used for the user button, if your device does not come with a predefined user button.

## GPIO for PWM Buzzer

This is the GPIO pin number that will be used for the PWM buzzer, if your device does not come with a predefined buzzer.

## Node Info Broadcast Seconds

This is the number of seconds between NodeInfo message (containing i.a. long and short name) broadcasts from the device. The device will still respond ad-hoc to NodeInfo messages when a response is wanted. When the device hears any packet from a node it doesn't know yet, it will send its NodeInfo and ask for a response automatically.

## Double Tap as Button Press

This option will enable a double tap, when a supported accelerometer is attached to the device, to be treated as a button press.

## TZDEF (Timezone Definition)

The `tzdef` setting allows the local offset to be defined for the device. It uses the TZ Database format to display the correct local time on the device display and in its logs.

To set the timezone, use the POSIX TZ Database string for the relevant region. Here is a list of supported timezones. For example, if your region is `America/Los_Angeles`, you would input `PST8PDT,M3.2.0,M11.1.0` as the timezone.

## LED Heartbeat Disabled

If true, disable the default blinking LED (LED_PIN) behavior on the device.

## Device Config Client Availability

**Android Apple CLI Web**

### Android



info

Device Config is available for Android.

1. Open the Meshtastic App
2. Navigate to: **Settings > Device**

**Apple CLI Web**

## Apple



info

> All device config options other than NTP Server are available on iOS, iPadOS and macOS at Settings > Device Configuration > Device.

**CLI Web**

## CLI



info

All device config options are available in the python CLI. Example commands are below:

| Setting | Acceptable Values | Default |
| --- | --- | --- |
| device.role | `CLIENT`, `CLIENT_MUTE`, `ROUTER`, `REPEATER`, `TRACKER`, `SENSOR` | `CLIENT` |
| device.rebroadcast_mode | `ALL`, `ALL_SKIP_DECODING`, `LOCAL_ONLY` | `ALL` |

| Setting | Acceptable Values | Default |
| --- | --- | --- |
| device.button_gpio | `0` - `34` | `0` |
| device.buzzer_gpio | `0` - `34` | `0` |
| device.node_info_broadcast_secs | `3600` - `UINT MAX` | `10800` (3 hours) |
| device.double_tap_as_button_press | `false`, `true` | `false` |



tip

> Because the device will reboot after each command is sent via CLI, it is recommended when setting multiple values in a config section that commands be chained together as one.
>
> Example:
>
> ```
> meshtastic --set device.role CLIENT --set device.tzdef UTC0
> ```

Set the role to client

```
meshtastic --set device.role CLIENT
```

Set tzdef

```
meshtastic --set device.tzdef UTC0
```

## Footnotes

1. This role has been deprecated as of firmware version 2.7.11 ↩
2. **Favorites Only** indicates that the `CLIENT_BASE` will prioritize packets from or to any favorited nodes. From your `CLIENT_BASE`, you should favorite all of your other nodes which are typically within range of the better-positioned `CLIENT_BASE` base station. ↩
3. **Awake Only** indicates retransmissions will occur normally but not while asleep between transmission intervals. ↩ ↩[2]
4. The `ROUTER` role enables Power Saving by default (ESP32 only). When needing to update a `ROUTER` via BLE, it's recommended you utilize an admin channel to temporarily change the role in order to update and then revert when done. ↩

# Choosing the right role (Meshtastic blog)

From https://meshtastic.org/blog/choosing-the-right-device-role/

*Last updated: September 24, 2025*

When setting up your Meshtastic network, configuring the correct Role for each device can be crucial for optimizing performance and ensuring reliable communication. Conversely, the pitfalls of choosing the incorrect Role can lead to congestion and poor performance on your mesh. In this post, we'll explore why you might choose certain device roles and avoid others.

## What is a device Role?

A device role in Meshtastic defines the primary function of a device within the network. Each role is tailored to specific usage and helps in managing the network and the behavior of the device more efficiently. Here are some of the common device roles to consider:

### Client

The defacto role for devices is the `CLIENT` role. This is a flexible, general purpose role for devices which should serve the overwhelming majority of use cases. When in doubt about what role you should go with, simply sticking with Client is a safe bet.

Despite the apparent baggage of the term *Client* in some technological contexts, Clients in Meshtastic do actually repeat / route messages. Unfortunately in the past this has caused some confusion, landing some individuals to incorrectly choose the `ROUTER` role.

### Client Mute

The `CLIENT_MUTE` role is similar to the `CLIENT` role but with one key difference: it does not repeat or route messages. This role is ideal for devices that are intended to be used in areas with high network traffic where additional message routing could cause congestion. By using the `CLIENT_MUTE` role, you can ensure that the device will only send and receive its own messages without contributing to network traffic.

This role is also highly recommended if you are a mesh enthusiast with multiple devices. Select one of your devices to be a `CLIENT` or `CLIENT_BASE` and set the rest to `CLIENT_MUTE` to keep your airtime usage more responsible.

### Client Base

The `CLIENT_BASE` role is similar to `CLIENT`, but has priority in rebroadcasting messages from or to any of its favorited nodes. For mesh enthusiasts with multiple devices, this role is perfect for ensuring all your nearby nodes take full advantage of your stronger, well-positioned attic/roof “base station” node.

If you have an attic/roof node, set this node to `CLIENT_BASE`. Set your other nodes (typically `CLIENT` or `CLIENT_MUTE`) as favorites on the `CLIENT_BASE`.

## Router and Repeater

The ROUTER role is designed for devices which are intended to primarily route messages to other devices on the mesh. This role is ONLY suitable for stationary devices placed in extremely strategic locations to act as unofficial hubs for routing packets on the mesh. Routers focus on relaying messages from other devices by *cutting in line* before other nodes have a chance to rebroadcast a message, making them key for extending the range and reliability of your mesh. Additionally Routers *always* rebroadcast, whereas most other roles potentially choose to forego rebroadcasting if they hear a neighbor rebroadcasting first.

Another default behavior of Routers is that the device tries to save as much power as possible by attempting to sleep as well as send telemetry packets less frequently than other devices on the mesh. This is because they are chiefly tasked with routing others' traffic rather than originating their own messages.

The REPEATER role behaves very similar to ROUTER in terms of becoming a preferred device for routing packets, however it goes a step further by completely turning off any broadcasted traffic such as telemetry. It only responds to other nodes packets instead of originating messages.

### What is a strategic location anyway?

When evaluating strategic locations for these two roles, consider a tower on mountain peak rather than a tall building. By electing a device to become a Router or Repeater, you are making an implicit choice for the entire mesh to prefer that node for rebroadcasts for any direct neighbors. This strategic placement becomes crucial for ensuring that packets can be delivered to the widest possible audience. Using line of sight viewshed survey tools to determine the optimal location is recommended, but selection can best be determined by collecting some real world data on the mesh first.



### Why improperly applying Router and Repeater roles is harmful

If a poor location is chosen for Routers and Repeaters, it can cause a number of issues:

1. Increased rate of packet collisions

Because Routers and Repeaters always rebroadcast, when too many devices are applied with these roles and the devices are direct neighbors, packets will potentially be rebroadcasted at the same time, creating higher noise levels and packet error rates. This frequently culminates in sporadic delivery failures.

2. Decreased overall range

An improperly located router will potentially prematurely *hop gobble* any packets routing through it. This has the effect of consuming a hop in the routing of a packet before it would be able to reach more strategically located nodes. This can greatly limit range for instance, in the

case of many Routers being deployed in a valley consuming all of the available hops before a packet is able to reach its destination through a more strategically node placed on a peak above the valley.

3. Asymmetrical links

Similarly to the issue of decreased range, the same scenario can also result in asymmetrical communication, wherein a subset of the mesh can send messages to a different group, but that group is unable to reach back with responses through the improperly placed Routers' premature consumption of hops before the message is able to deliver. This phenomena can also lead to a reaction of users increasing the hop limit to compensate for the problem, which unfortunately further increases congestion by utilizing more air time.

## Sensor

The `SENSOR` role is intended for devices which primarily gather and transmit sensor data. These devices still participate in routing messages for other devices, but they prioritize sending their own telemetry data to the network, even in the face of high channel utilization. This role is ideal for environmental monitoring, weather stations, or any application where the device's main function is to collect and report telemetry.

By using the `SENSOR` role in combination with `power.is_power_saving`, the device will attempt to sleep between intervals of sending environmental telemetry, significantly extending runtimes for devices utilizing this combination of settings.

## Tracker

The `TRACKER` role is designed for devices that are primarily used for tracking the location of assets, vehicles, or individuals. Devices in this role periodically send their GPS coordinates to the network via Position packets with a higher priority, allowing for more robust location tracking. Trackers still participate in routing messages, but their main goal is to provide timely location data, even in the face of high channel utilization.

By using the `TRACKER` role in combination with `power.is_power_saving`, the device will attempt to sleep between intervals of sending position data, significantly extending runtimes for devices utilizing this combination of settings.

## Conclusion

Choosing the right device role is crucial for the performance and reliability of your Meshtastic network. By understanding the differences between the common roles, you can optimize your network setup to meet your specific needs and ensure efficient communication across all devices. For more technical information about each role, visit the device configuration docs.

# Position

From https://meshtastic.org/docs/configuration/radio/position/

Position data from GPS is provided by either the radio or your paired phone. It is not required to use Meshtastic, but time calculations require at least one device on the mesh to have either a GPS, RTC, or internet connection for NTP.

The position config uses an admin message to send a `Config.Position` protobuf. The position config options are: GPS Mode, GPS Update Interval, Fixed Position, Smart Broadcast, Smart Broadcast Minimum Distance, Smart Broadcast Minimum Interval, Broadcast Interval, Position Packet Flags, and GPS RX/TX/EN Pins.

## Position Config Values

### GPS Mode

Configures whether the GPS functionality is enabled, disabled, or not present on the node.

Acceptable values: `DISABLED`, `ENABLED`, and `NOT_PRESENT`.

Generally, depending on the device and its configuration, this value will default to either `ENABLED` or `NOT_PRESENT`.

### GPS Update Interval

How often we should try to get GPS position (in seconds), or zero for the default of once every 2 minutes, or a very large value (maxint) to update only once at boot.

### Fixed Position

Acceptable values: `true` or `false`

False by default

If set, this node is at a fixed position. The device will generate GPS updates at the regular GPS update interval, but use whatever the last lat/lon/alt it saved for the node. The lat/lon/alt can be set by an internal GPS, by the mobile device's GPS, or entered manually.

### Smart Broadcast

Acceptable values: `true` or `false`

True by default

Smart broadcast will send out your position at an increased frequency only if your location has changed enough for a position update to be useful.

Smart broadcast complements broadcast interval (doesn't override that setting) but will apply an algorithm to more frequently update your mesh network if you are in motion and then throttle it down when you are standing still. If you use this feature, it's best to leave broadcast interval at the default.

Smart broadcast will calculate an ideal position update interval based on the data rate of your selected channel configuration.

## Smart Broadcast Minimum Distance

Default of `0` is 100 meters

The minimum distance in meters traveled (since the last send) before we can send a position to the mesh if smart broadcast is enabled.

## Smart Broadcast Minimum Interval

Default of `0` is 30 seconds

The minimum number of seconds (since the last send) before we can send a position to the mesh if smart broadcast is enabled.

## Broadcast Interval

Default of `0` is 15 minutes

If smart broadcast is off we should send our position this often.

A position is sent every broadcast interval: the GPS location once a fix is acquired, the saved lat/lon/alt if fixed position is set, otherwise an empty location.

## Position Flags

Defines which options are sent in POSITION messages. Values are stored as a bit field of boolean configuration options (bitwise OR of PositionFlags).

| Value | Description |
|---|---|
| UNSET | Required for compilation |
| ALTITUDE | Include an altitude value (if available) |
| ALTITUDE_MSL | Altitude value is MSL |
| GEOIDAL_SEPARATION | Include geoidal separation |
| DOP | Include the DOP value ; PDOP used by default, see below |
| HVDOP | If POS_DOP set, send separate HDOP / VDOP values instead of PDOP |
| SATINVIEW | Include number of "satellites in view" |
| SEQ_NO | Include a sequence number incremented per packet |
| TIMESTAMP | Include positional timestamp (from GPS solution) |

| Value | Description |
| --- | --- |
| HEADING | Include positional heading (from GPS solution) |
| SPEED | Include positional speed (from GPS solution) |

### GPIO RX/TX/EN for GPS Module

If your device does not have a fixed GPS chip, you can define the GPIO pins for the RX, TX, and EN pins of a GPS module.

## Position Config Client Availability

**Android Apple CLI Web**

### Android



info

Position Config options are available for Android.

1. Open the Meshtastic App
2. Navigate to: **Settings > Position**

**Apple CLI Web**

## Apple



info

> All position config values are available on iOS, iPadOS and macOS at Settings > Device Configuration > Position.

**CLI Web**

## CLI



info

All Position config commands are available in the python CLI. Example commands are below:

| Setting | Acceptable Values | Default |
| --- | --- | --- |
| position.gps_mode | `ENABLED`, `DISABLED`, `NOT_PRESENT` | `ENABLED` or `NOT_PRESENT` |

| Setting | Acceptable Values | Default |
| --- | --- | --- |
| position.gps_update_interval | `integer` (seconds) | Default `0` is 2 Minutes |
| position.fixed_position | `true`, `false` | `false` |
| position.position_broadcast_smart_enabled | `true`, `false` | `true` |
| position.broadcast_smart_minimum_distance | `integer` (meters) | Default of `0` is 100 Meters |
| position.broadcast_smart_minimum_interval_secs | `integer` (seconds) | Default of `0` is 15 Minutes |
| position.position_broadcast_secs | `integer` (seconds) | Default of `0` is 15 minutes |
| position.flags | `UNSET`, `ALTITUDE`, `ALTITUDE_MSL`, `GEOIDAL_SEPARATION`, `DOP`, `HVDOP`, `PDOP`, `SATINVIEW`, `SEQ_NO`, `TIMESTAMP`, `HEADING`, `SPEED` | `UNSET` |
| position.rx_gpio | `integer` (0-39) | `UNSET` |
| position.tx_gpio | `integer` (0-34) | `UNSET` |



tip

> Because the device will reboot after each command is sent via CLI, it is recommended when setting multiple values in a config section that commands be chained together as one. **This is especially important for position values to ensure they are set at the same time and avoid being overwritten by subsequent commands.**
>
> Example:
>
> ```
> meshtastic --set position.fixed_position true --setlat 37.8651 --setlon -119.5383
> ```

Set GPS update interval (Default of 0 is 2 Minutes)

```
meshtastic --set position.gps_update_interval 0
meshtastic --set position.gps_update_interval 45
```

Set Fixed Position - Current Location

```
meshtastic --set position.fixed_position true
```



note

The device will continue to acquire GPS coordinates according to the `gps_update_interval`, but will use the last saved coordinates as its fixed point.

Set and Enable Fixed Position - User Defined

```
meshtastic --setlat 37.8651 --setlon -119.5383
```

Remove and Unset Fixed Position

```
meshtastic --remove-position
```

Enable / Disable Smart position broadcast (Enabled by default)

```
meshtastic --set position.position_broadcast_smart_enabled true
meshtastic --set position.position_broadcast_smart_enabled false
```

Set Position Broadcast Interval (Default of 0 is 15 minutes)

```
meshtastic --set position.broadcast_secs 0
meshtastic --set position.broadcast_secs 60
```



note

It may take some time to see that the change has taken effect. The GPS location is updated according to the value specified on `gps_update_interval` and the mesh will be notified of the new position in relation to the `position_broadcast_secs` value.

Set / Unset Position Flags

```
meshtastic --pos-fields ALTITUDE ALTITUDE_MSL
```

```
meshtastic --pos-fields UNSET
```

# Power

From https://meshtastic.org/docs/configuration/radio/power/



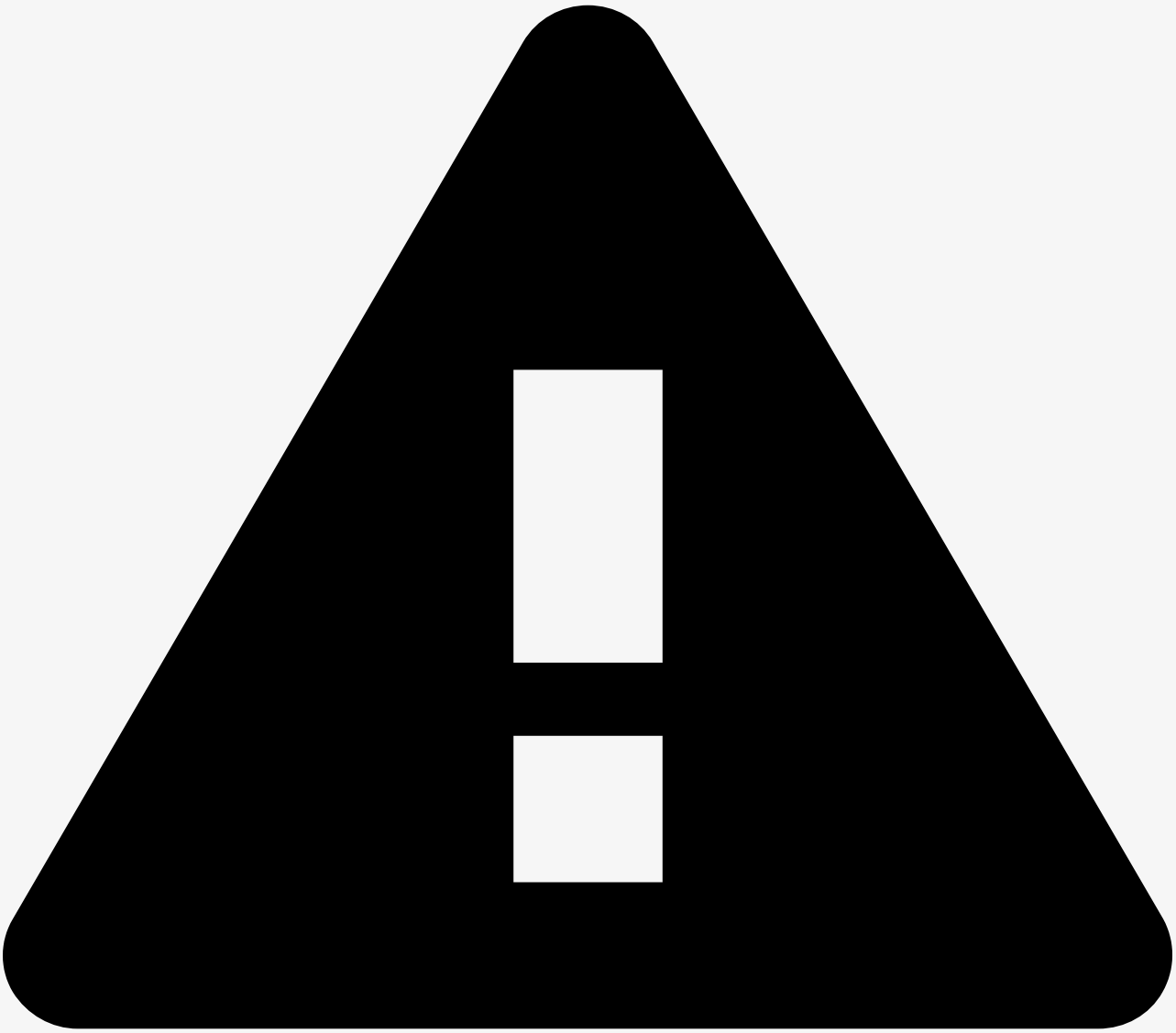
Use Device Config first

Power settings are advanced configuration, most users should choose a role under Device Config to manage power for their device and shouldn't ever need to adjust these settings.

The power config options are: Power Saving, Shutdown after losing power, ADC Multiplier Override, Wait Bluetooth Interval, Light Sleep Interval, Minimum Wake Interval, and Device Battery INA2xx Address. Power config uses an admin message sending a `Config.Power` protobuf.

## Power Config Values

### Power Saving



warning

If enabled, modifications to settings can be made by waking the device through pressing the user button, resetting, or through the admin channel for remote administration.

When activated, this feature disables Bluetooth, Serial, Wi-Fi, and the device's screen to conserve power. This is particularly beneficial for devices relying on low-current power sources, like solar panels. For details on which device roles have this feature enabled by default, please check the Device Config section.

### Shutdown after losing power

Automatically shut down a device after a defined time period if power is lost.

Before setting this, ensure your board correctly detects when it is connected to external power (the "🔌" icon in the app). Otherwise, your device could shut down even when power is present. Many devices detect power status correctly, however for some devices, you will need to modify the ADC Multiplier (below).

### ADC Multiplier Override

Ratio of voltage divider for battery pin e.g. 3.20 (R1=100k, R2=220k)

Overrides the ADC_MULTIPLIER defined in the firmware device variant file for battery voltage calculation.

Should be set to floating point value between 2 and 6

#### Calibration Process (**Attribution**)

1. Install the rechargeable battery.
2. Charge the battery until full. Indication of this state may vary depending on device. At this point, the battery voltage should be 4.2V +-1%.
3. Input the "Battery Charge Percent" displayed on the screen or in your connected app into the calculator below.
4. If "Battery Charge Percent" (e.g., B 3.82V 60%) is not displayed on the screen, it means that the default value of "Operative Adc Multiplier" is too high. Lower the "Operative Adc Multiplier" to a smaller number (it is recommended to decrease by 0.1) until the screen displays "Battery Charge Percent". Enter the current "Operative Adc Multiplier" in use into the "Operative Adc Multiplier" field in the calculator. Also, input the "Battery Charge Percent" displayed on the screen into the calculator.
5. Click the "Calculate" button to compute the "Calculated New Operative Adc Multiplier", and set it as the new "Operative Adc Multiplier" for the device.

> ▼ ADC Calculator



info

This calibration method only maps 4.2V to Battery Charge Percent 100%, and does not address the potential non-linearities of the ADC.

| | |
|---|---|
| Battery Charge Percent: | 65 |
| Current Adc Multiplier: | 2 |
| Calculated New Operative Adc Multiplier: | 2 |
| | |

Default values for ADC Multipliers are:

| Device | Default ADC Multiplier |
|---|---|
| chatter2 | 5.0 |
| diy | 1.85 |
| esp32-s3-pico | 3.1 |
| heltec_v1 & heltec_v2 | 3.2 |
| heltec_v3 & heltec_wsl_v3 | 5.1205 |
| heltec_v4 | 5.1205 |
| heltec_wireless_paper | 2 |
| heltec_wireless_tracker 1.0/1.1 | 5.1205 |
| heltec_T114 | 4.916 |
| m5stack_coreink | 5 |
| nano-g1-explorer | 2 |
| nano-g2-ultra | 2 |
| picomputer-s3 | 3.1 |
| rak4631 | 1.73 |
| rpipico | 3.1 |
| rpipicow | 3.1 |
| station-g1 | 6.45 |
| tlora_v2_1_16 | 2 |
| tlora_v2_1_18 | 2.11 |
| tlora_t3s3_v1 | 2.11 |
| t-deck | 2.11 |
| t-echo | 2 |

## Wait Bluetooth Interval

How long to wait before turning off BLE in no Bluetooth states

`0` for default of 1 minute

## Light Sleep Interval



ESP32 Only

This setting only applies to ESP32-based boards, it will have no effect on nRF52/RP2040 boards.

In light sleep the CPU is suspended, LoRa radio is on, BLE is off and GPS is on

`0` for default of five minutes

## Minimum Wake Interval

While in light sleep when we receive packets on the LoRa radio we will wake and handle them and stay awake in no Bluetooth mode for this interval

`0` for default of 10 seconds

## Device Battery INA2xx Address

If an INA-2XX device is auto-detected on one of the I2C buses at the specified address, it will be used as the authoritative source for reading device battery level voltage. Setting is ignored for devices with PMUs (e.g. T-beams)



Convert hexadecimal to decimal

> I2C addresses are normally represented in hexadecimal and will require conversion to decimal in order to set via Meshtastic clients. For example the I2C address of 0x40 converted to decimal is 64.

## Power Config Client Availability

**Android Apple CLI Web**

### Android



info

Power Config options are available for Android.

1. Open the Meshtastic App
2. Navigate to: **Settings > Power**

**Apple CLI Web**

## Apple



info

> Select Power config options are available on iOS, iPadOS and macOS at Settings > Device Configuration > Power.

**CLI Web**

## CLI



info

All Power config options are available in the python CLI.

| Setting | Acceptable Values | Default |
| --- | --- | --- |
| power.is_power_saving | `true`, `false` | `false` |
| power.on_battery_shutdown_after_secs | `integer` (seconds) | Default of `0` is off |

| Setting | Acceptable Values | Default |
| --- | --- | --- |
| power.adc_multiplier_override | `2-4` (floating point value) | Default of `0` uses firmware values |
| power.wait_bluetooth_secs | `integer` (seconds) | Default of `0` is 1 minute |
| power.ls_secs | `integer` (seconds) | Default of `0` is 5 minutes |
| power.min_wake_secs | `integer` (seconds) | Default of `0` is 10 seconds |
| power.device_battery_ina_address | `integer` (I2C address as decimal) | Default of `0` is no address set |



tip

> Because the device will reboot after each command is sent via CLI, it is recommended when setting multiple values in a config section that commands be chained together as one.
>
> Example:
>
> ```
> meshtastic --set power.is_power_saving true --set power.on_battery_shutdown_after_secs 120
> ```

Enable / Disable Power Saving

```
meshtastic --set power.is_power_saving true
meshtastic --set power.is_power_saving false
```

Enable / Disable Shutdown after losing power

```
meshtastic --set power.on_battery_shutdown_after_secs 120
meshtastic --set power.on_battery_shutdown_after_secs 0
```

Set Wait Bluetooth Interval (Default of 0 is 60 seconds)

```
meshtastic --set power.wait_bluetooth_secs 0
meshtastic --set power.wait_bluetooth_secs 120
```

Set Light Sleep to default (Default of 0 is 5 minutes)

```
meshtastic --set power.ls_secs 0
meshtastic --set power.ls_secs 120
```

Set Minimum Wake Interval (Default of 0 is 10 seconds)

```
meshtastic --set power.min_wake_secs 0
meshtastic --set power.min_wake_secs 120
```

# Bluetooth

From https://meshtastic.org/docs/configuration/radio/bluetooth/

The Bluetooth config options are: Enabled, Pairing Mode and Fixed PIN Value. Bluetooth config uses an admin message sending a `Config.Bluetooth` protobuf.



info

> ESP32 Devices: Bluetooth will be disabled if Wi-Fi is enabled. The Wi-Fi setting takes precedence.

## Bluetooth Config Values

### Enabled

Enables Bluetooth.

### Pairing Mode

Specify pairing mode.

`RANDOM_PIN` generates a random PIN during runtime. `FIXED_PIN` uses the fixed PIN that should then be additionally specified. Finally, `NO_PIN` disables PIN authentication.

### Default Pairing Mode

The default pairing mode will be determined based on whether the device has or does not have a screen attached to it during the first boot (or with a stale device state) unless manually configured via the Bluetooth config options.

- **Screen Attached:** If your device boots up for the first time (or with a stale device state), and a screen is detected, the default pairing mode will be set to `RANDOM_PIN`. Should the attached screen be removed after the device has already been booted, the default pairing mode of `RANDOM_PIN` will remain unless manually changed to `FIXED_PIN` or `NO_PIN`. It is recommended the pairing mode be updated prior to removing the attached screen.
- **No Screen Attached:** If your device boots up for the first time (or with a stale device state), and no screen is detected, the default paring mode will be set to `FIXED_PIN` with the default value listed below unless manually configured to a custom value.

### Fixed PIN

If your pairing mode is set to `FIXED_PIN`, the default value is **123456**. For all other pairing modes, this number is ignored. A custom integer (6 digits) can be set via the Bluetooth config options.



Warning

> It is strongly recommended that you change the default `FIXED_PIN` code on your device. Leaving the default code in place can pose a significant security risk.

## Bluetooth Config Client Availability

**Android Apple CLI Web**

### Android



info

All Bluetooth config options are available for Android.

1. Open the Meshtastic App
2. Navigate to: **Settings > Device configuration > Bluetooth**

**Apple CLI Web**

## Apple



info

> All bluetooth config options are available on iOS, iPadOS and macOS at Settings > Device Configuration > Bluetooth.

**CLI Web**

## CLI



info

All Bluetooth config options are available in the python CLI. Example commands are below:

| Setting | Acceptable Values | Default |
| --- | --- | --- |
| `bluetooth.enabled` | `true`, `false` | `true` |

| Setting | Acceptable Values | Default |
| --- | --- | --- |
| `bluetooth.mode` | `RANDOM_PIN`, `FIXED_PIN`, `NO_PIN` | `RANDOM_PIN` |
| `bluetooth.fixed_pin` | `integer` (6 digits) | `123456` |



tip

> Because the device will reboot after each command is sent via CLI, it is recommended when setting multiple values in a config section that commands be chained together as one.
>
> Example:
>
> ```
> meshtastic --set bluetooth.enabled true  --set bluetooth.fixed_pin 111111
> ```

Enable/Disable Bluetooth Module

```
meshtastic --set bluetooth.enabled true
meshtastic --set bluetooth.enabled false
```

Set a fixed pin

```
meshtastic --set bluetooth.mode FIXED_PIN
meshtastic --set bluetooth.fixed_pin 111111
```

# Security

From https://meshtastic.org/docs/configuration/radio/security/

The security config options are: Public Key, Private Key, Admin Key, Is Managed, Serial Console, Debug Logs, Admin Channel, and Packet Signature Policy.

## Security Config Values

### Public Key

Acceptable values: `bytes`

The public key of the device, shared with other nodes on the mesh to allow them to compute a shared secret key for secure communication. Other nodes also use it to check the device's packet signatures, and the device's node number is derived from it. The firmware generates the key pair once a LoRa region is set.

### Private Key

Acceptable values: `bytes`

The private key of the device, used to create a shared key with a remote device for secure communication.**This key should be kept confidential.**

### Admin Key

Acceptable values: `repeated bytes`

The public keys authorized to send administrative messages to this node, up to three. Over remote administration, the node accepts an admin message only if it arrives encrypted as a direct message from a node whose public key is in this list. The legacy admin channel is a separate, less secure path, available only when Admin Channel Enabled is on.

### Managed Mode

Acceptable values: `true` or `false`

Enabling Managed Mode blocks client applications from writing configurations to a radio(they may be read). Once enabled, radio configurations can only be changed through PKC Remote Admin messages with firmware version 2.5+ or the legacy Admin channel with firmware prior to version 2.5. This setting is **not** required for remote node administration.

Managed Mode requires at least one admin key. If none is set, the firmware turns Managed Mode off and sends a warning to the client.

Before enabling Managed Mode, verify that the node can be controlled via Remote Admin or legacy Admin channel, and that all functions are working properly to prevent being locked out.

### Serial Console

Acceptable values: `true` or `false`

Disabling this will prevent the Serial Console from initializing the Stream API.

### Debug Log

Acceptable values: `true` or `false`

By default, logging is disabled when an API client connects to keep the shared serial link quiet. Set this to true to continue outputting live debug logs over serial or Bluetooth when the API is active.

### Admin Channel Enabled

Acceptable values: `true` or `false`

Allows incoming device control over the insecure legacy admin channel. Enabling this option permits control messages to be received through the older, less secure admin channel.

### Packet Signature Policy

Acceptable values: `COMPATIBLE`, `BALANCED`, or `STRICT`

Sets which packets the node signs, and what it does with packets that carry no signature. This section describes firmware 2.8.2 and later.

| Policy | Signs outgoing packets | Accepts unsigned packets |
|---|---|---|
| `COMPATIBLE` (default) | No | Yes |
| `BALANCED` | Yes[1] | Yes, but ignores unsigned name and role changes from nodes that have signed before |
| `STRICT` | Yes[1] | No[1] |

Under every policy, the node drops a packet whose signature is malformed or doesn't match the sender's known public key. A signature from a node whose public key isn't known yet can't be checked: `COMPATIBLE` and `BALANCED` treat the packet as unsigned, and `STRICT` drops it.

Firmware older than 2.8 strips signatures when it relays a packet, so a signed packet relayed by an older node arrives unsigned. Due to this, `BALANCED` accepts unsigned packets to ease the migration. In a future update, it will be updated to drop unsigned packets from known signers.

Firmware builds without packet signing, such as STM32WL builds, always use `COMPATIBLE`.

## Security Config Client Availability

**Android Apple CLI Web**

## Android

All Security config options are available for Android.

1. Open the Meshtastic App
2. Navigate to: **Settings > Security**

**Apple CLI Web**

## Apple

All Security config options are available on iOS, iPadOS and macOS at Settings > Radio Configuration > Security.

**CLI Web**

## CLI

All Security config options are available in the Python CLI. Example commands are below:

| Setting | Acceptable Values | Default |
|---|---|---|
| security.public_key | `bytes` | `None` |
| security.private_key | `bytes` | `None` |
| security.admin_key | `repeated bytes` | `None` |
| security.is_managed | `true`, `false` | `false` |
| security.serial_enabled | `true`, `false` | `true` |
| security.debug_log_api_enabled | `true`, `false` | `false` |
| security.admin_channel_enabled | `true`, `false` | `false` |
| security.packet_signature_policy | `COMPATIBLE`, `BALANCED`, `STRICT` | `COMPATIBLE` |



tip

> Because the device may reboot after each command is sent via CLI, it is recommended when setting multiple values in a config section that commands be chained together as one.
>
> Example:
>
> ```
> meshtastic --set security.is_managed false --set security.debug_log_api_enabled true
> ```

Disable serial console

```
meshtastic --set security.serial_enabled false
```

Enable debug logging

```
meshtastic --set security.debug_log_api_enabled true
```

## Security Keys - Backup and Restore

### Backup

Public and private keys will be lost and regenerated by performing a firmware erase and reinstall. A new key also gives the node a new node number, so other nodes see it as a different node. It is your responsibility to back up your keys to maintain encrypted direct messaging with existing nodes on your mesh.

Suggested backup methods:

- With the Meshtastic CLI, run: `meshtastic --export-config > config_backup.yaml` - this will save your entire configuration including your public and private keys to a file.
- Copy the keys to a secure location, such as a password protected note.
- Take a screenshot - the most basic way, you must ensure you saved the entirety of both keys.

### Restore

To restore your keys you may:

- Use the Meshtastic CLI to run `meshtastic --configure config_backup.yaml` - This is the fastest way to get your entire node back to where you left it!
- Manually paste or retype the keys into the client app from your saved location.

## Footnotes

1. Traceroutes and oversize packets are unsigned for technical reasons, while Acknowledgments carry a smaller proof instead of a signature. Direct Messages are authenticated via PKC rather than XEdDSA. ↩ ↩[2] ↩[3]

# User: names

From https://meshtastic.org/docs/configuration/radio/user/

The user config options are: Long Name, Short Name, Is Licensed, and Is Unmessageable. User config uses an admin message sending a `User` protobuf.

## User Config Values

### Long Name

A personalized name for your device.

Auto-generated by default.

If you are a licensed ham operator and have enabled `IsLicensed`, this should be set to your ham operator call sign.

### Short Name

A personalized short identifier for your device.

Auto-generated by default.

### Is Licensed (ham)

If you are a licensed ham operator and have considered the privileges and restrictions of using Meshtastic with a ham license, enable this flag.

Disabled by default.

By enabling `IsLicensed`, you should also review the following related configurations:

- **User:** `LongName` (Should be your Call Sign)
- **Channel:** `PSK` (Should be Empty, removing encryption)

### Is Unmessageable

Whether or not the node can be direct messaged

Default is False except for roles that typically would not be monitored for direct messages: ROUTER, ROUTER_LATE, REPEATER, SENSOR, LOST_AND_FOUND, TRACKER, TAK_TRACKER

## User Config Client Availability

**Android Apple CLI Web**

### Android



info

All User config options are available for Android.

1. Open the Meshtastic App
2. Navigate to: **Settings > User**

Apple CLI Web

## Apple



info

> All User config options are available on iOS, iPadOS and macOS at Settings > Device Configuration > User.

**CLI Web**

## CLI



info

All User config options are available in the python CLI. Example commands are below:

Please see instructions for Enabling Ham License



tip

> Because the device will reboot after each command is sent via CLI, it is recommended when setting multiple values in a config section that commands be chained together as one.
>
> Example:
>
> ```
> meshtastic --set-owner 'your node name' --set-owner-short  'NODE'
> ```

Set the LongName Value

```
meshtastic --set-owner 'your node name'
```

Set the ShortName Value

```
meshtastic --set-owner-short  'NODE'
```

Enable Ham Mode, set name to license, disable encryption

```
meshtastic  --set-ham 'CALLSIGN'
```

# Tips

From https://meshtastic.org/docs/configuration/tips/

## Roles

It is strongly recommended to keep your ROLE set to `CLIENT`, `CLIENT_MUTE`, or `CLIENT_BASE`. Only use other roles if you have a specific, well-understood reason to do so. Read our blog post about choosing the best role.

### Recommended Roles

- `CLIENT` nodes receive, send, and intelligently repeat messages to help the mesh network.
    - **This is the right choice for most situations.**
    - Automatically adds smart delays when rebroadcasting to keep the network stable.
    - Perfect for nodes that can help others: rooftop installations, high-visibility locations, or nodes that extend network coverage.
    - Use when in remote areas (hiking, skiing, biking) where every node helps maintain connectivity.
- `CLIENT_MUTE` nodes can send and receive messages but never repeat them.
    - Use with a personal handheld device when located near a more powerful node (such as a rooftop node) in a congested network.
    - Use when multiple nodes are in close proximity. Set the best-positioned node to `CLIENT` and the others to `CLIENT_MUTE`.
    - Perfect for indoor nodes that only talk to one outdoor node - since they can't reach other nodes anyway, repeating messages would just waste bandwidth.
- `CLIENT_BASE` nodes prioritize rebroadcasting messages from your other personal nodes.
    - Use for a stronger or well-positioned attic/roof “base station” node to distribute messages more widely from your own weaker, indoor, or less-well-positioned nodes.
    - Set one node to `CLIENT_BASE`. Set your other nodes (typically `CLIENT` or `CLIENT_MUTE`) as favorites on the `CLIENT_BASE`.
    - Perfect for ensuring all your nearby nodes take full advantage of your stronger, well-positioned attic/roof node.



*One example of a 'Client' node. Photo credit: Cully@KBOXLABS*

### **Avoid** `ROUTER`**and** `REPEATER`

Using `ROUTER` or `REPEATER` roles unnecessarily can cause serious network issues:

- Increased risk of packet collision.
- Reduced message delivery rates.
- Decreased effective network range due to unnecessary hop consumption.

These roles are for very specific applications. Before changing from `CLIENT`, carefully review the documentation to understand the implications. Additionally, coordinating this with your local community to ensure its impact is a positive one rather than a negative one.

Remember: A network of `CLIENT` nodes with a small number of well-placed `ROUTERS` is usually the most efficient and stable configuration.



*One example of a 'Router' node. Photo credit: Cully@KBOXLABS*

## (Not) Sharing Your Location

Telemetry is shared over your PRIMARY channel. This means that if your node has acquired GPS coordinates from an integrated GPS chip, or from your mobile device, your coordinates will be sent to the mesh over this channel, using its defined encryption (if any).

By default the PRIMARY channel's name is LongFast with the encryption key "AQ==" (Base64 equivalent of Hex 0x01). If this is left unchanged, your location will be shared with all nodes in range that are also using the default channel.

### Using Position Precision

Meshtastic now supports the ability to control the precision of location data sent over a particular channel.

The `position_precision` setting allows control of the level of precision for location data that is sent over a particular channel. This can be useful for privacy reasons, where obfuscating the

exact location may be desired when sending position data over certain channels.

For detailed information on position precision settings and how to configure them, please refer to the Position Precision documentation.

### Sharing Location on a Private Secondary Channel

> This is a newer feature that only works for firmware 2.7.1+

To share your location on a private secondary channel while keeping the default primary channel unencrypted, follow these steps:

1. Disable location sharing on the primary channel (Channel 0) to prevent broadcasting your position to all users.
2. Select a secondary channel where you want to share your location. Enable position sharing and choose your desired precision level.
3. Understand automatic location updates:
    - Only one channel receives your automatic, live location updates.
    - This is the lowest-indexed secondary channel (excluding Channel 0) with location sharing enabled.
    - If multiple secondary channels have location sharing enabled, only the one with the lowest index will receive automatic position broadcasts.
    - Channels with location sharing enabled but not receiving automatic broadcasts can still be used for manual position requests.

#### Example Scenario

Imagine you're in the following channels:

- Channel 0: "Everyone" (location OFF)
- Channel 1: "CampChat" (location ON)
- Channel 2: "Friends" (location ON)
- Channel 3: "TutuTuesdayGroup" (location OFF)

#### How It Works

- CampChat (Channel 1) receives your live, automatic location updates because it's the lowest-indexed channel with location sharing enabled.
- Friends (Channel 2) can request your location manually, but it won't receive automatic updates.
- TutuTuesdayGroup (Channel 3) allows chatting but does not share or receive location data.

## Rebroadcast "Public" Traffic

Meshtastic nodes will rebroadcast all packets if they share LoRa modem settings, regardless of encryption (unless Rebroadcast mode is set to `LOCAL_ONLY`).

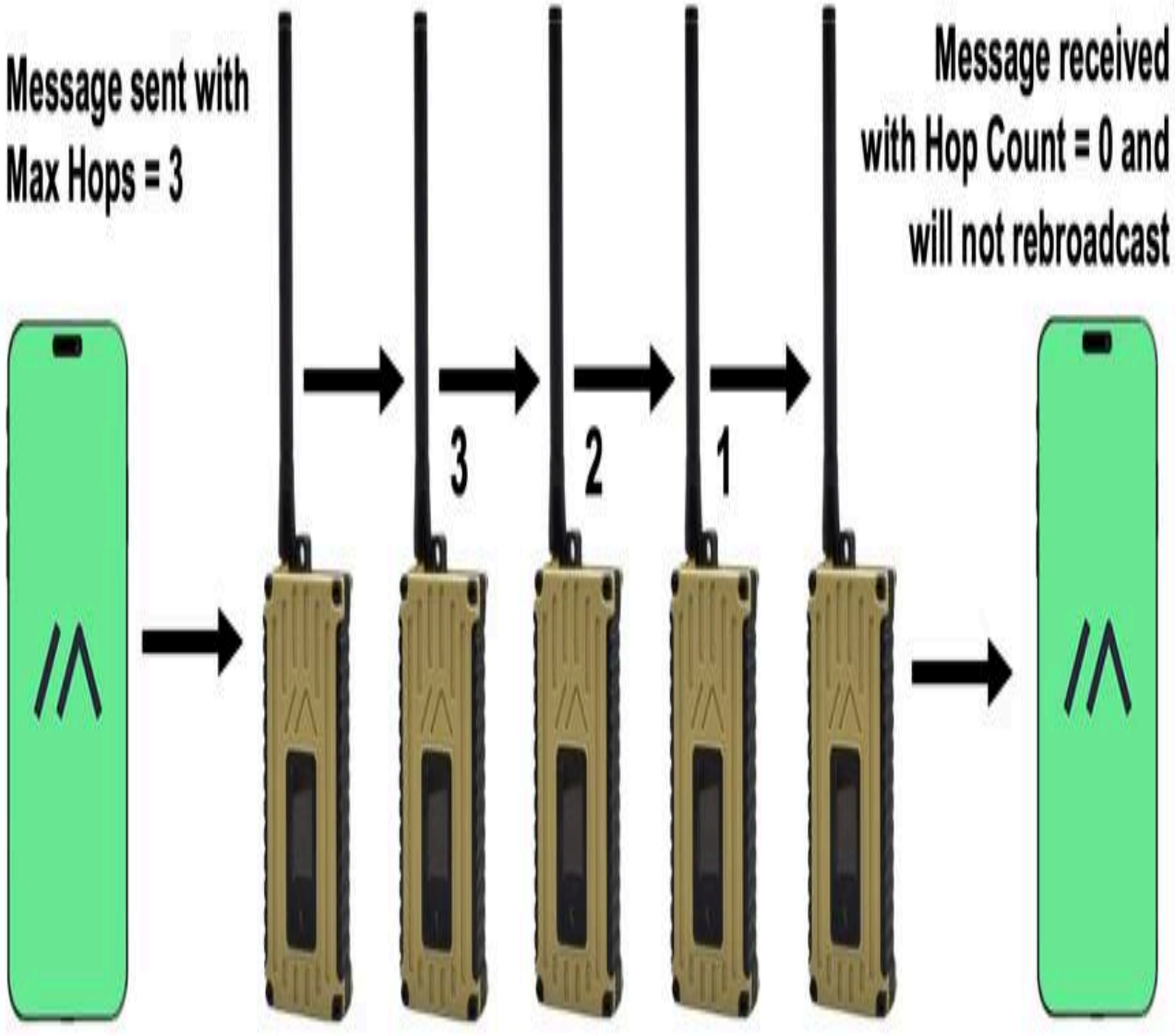


info

If you would like your nodes to include/expand the "public" mesh, you must use the default modem preset `LONG_FAST`. If you change your PRIMARY channel name, you must manually set the LoRa channel to the default for your region (see above).

## Hop Count

It is **strongly** recommended to leave your MAX HOPS set to 3 unless you're sure you need more (or less) to reach your destination node. Unnecessarily high hop counts often lead to network issues. If you need to increase the hop count, apply changes only to nodes at the network's edges, not those near the middle.




## Chat Channels and LoRa Frequency Slots

Previously, Meshtastic used the word "channels" to define two different configuration properties: Messaging Channels & LoRa Channel Numbers. However, this could lead to confusion since "channels" was being used to refer to two distinct concepts. To clarify the distinction, the decision was made to change "LoRa Channel Number" to "LoRa Frequency Slot".

### Radio Config: Channels

These configure "message groups" and include your PRIMARY and SECONDARY channels. All SECONDARY channels use the same LoRa modem config as your PRIMARY channel (including LoRa frequency slot).

There are 8 total messaging channels. Channel 0 is your PRIMARY channel, with channels 1-7 available for private group messaging and/or special channels such as `admin`.

### Radio Config: LoRa Frequency Slot

This property, formerly known as "LoRa Channel Number", configures the frequency for the radio. Check out the frequency calculator to view the relationship between "frequency slot" and radio frequency.

## Presets and unreachable nodes

Changing the modem preset, region, or frequency slot moves the radio without moving the node database, which persists across configuration changes and reboots. Nodes heard before the change stay in the list afterward, indistinguishable from nodes that are still reachable.

Firmware 2.8.1 and later records the LoRa settings each node was heard on, and clients use it to mark nodes not heard on your current LoRa settings. It reports only that a node has not been heard on the settings in force; a radio cannot observe a channel it is not tuned to, so nothing can say which settings that node is now using. Nodes that arrived over MQTT are never marked, and nothing is erased when settings change, so returning to a previous configuration restores the previous marks.

Test reachability with a direct message rather than a channel message. Only the recipient's acknowledgment confirms delivery. A node that relays the message also stops the retries, which proves only that a nearby node heard it, and a delivery error appears only when no node relays it. A channel message counts as delivered as soon as any one node repeats it, so a success there proves only that a nearby node heard it, not that it reached anyone beyond.

Before changing a mesh-wide setting, confirm what the nodes around you are using. See Joining your local mesh.

## Best Practices

- If you are part of a large mesh and don't know what a setting does, don't change it (unless you're super curious).
- TEST your settings and hardware before you install in hard-to-reach locations.
- Connecting a node to the public MQTT server may publish the locations of all nodes in your mesh to the internet.

## Keeping things quiet

Things can get noisy in bigger meshes, especially when your device has a buzzer. This can become problematic at times, particularly in situations involving essential communications in an area with a chatty public channel.

To mute a channel, see: channel.mute

For muting particular nodes, try using the python CLI:

```
# mutes a node in the NodeDB
meshtastic --set-muted-node '!xxxxxxxx'

# un-mutes a node in the NodeDB
meshtastic --remove-muted-node '!xxxxxxxx'
```

If a particular node is bothering you and you don't want to receive their messages at all, try ignoring a node. Which can be performed through the various client applications.

Using the python CLI:

```
# ignores a node in the NodeDB
meshtastic --set-ignored-node '!xxxxxxxx'

# un-ignores a node in the NodeDB
meshtastic --remove-ignored-node '!xxxxxxxx'
```

# Modules

## Range test

From https://meshtastic.org/docs/configuration/module/range-test/

This module lets you test the communication range between two Meshtastic nodes. Only nodes with the module enabled will show range test messages on their connected clients.

- Sender: fixed node that sends sequential packets ("Sender message interval" between 30-60s);
- Receiver: mobile node (typically you) optionally with onboard or phone GPS.

Nodes are in range as long as the sequential packets can be received.

The receiving node can be used to save the messages along with the GPS coordinates at which they were received into a .csv file. This .csv file can then be integrated into Google Earth, Google Maps - My Maps, or any other program capable of processing .csv files. This can enable you to visualize your mesh. Ensure you have position precision in the default channel set to high; otherwise, the collected data will not record your true location.



info

Be sure to turn off the module or disable sending when not in use. This will use a lot of time on air, slow down your mesh, and spam your channel. The module will automatically turn off after 8 hours.

The range test module config options are: `enabled`, `sender`, `save` and `clear_on_reboot`. Range Test Module config uses an admin message sending a `ModuleConfig.RangeTestConfig` protobuf.

# Range Test Module Config Values

## Enabled

Enables the range test module. **Both Sender and Receiver must have the module enabled.** On the Apple and Android apps to receive any range test messages the module must be enabled. The device screen may continue to show range test packets from older firmware versions even if the module is disabled.

## Sender Interval

How long to wait between sending sequential test packets. 0 is default which disables sending messages.

## Recommended Sender Settings

| **Radio Setting** | `range_test.sender` |
|---|---|
| Long Fast | 30 |
| Medium | 15 |
| Short Fast | 15 |

## Save CSV File



info

**Leave disabled when using the Android or Apple apps.** Saves directly to the device's flash memory (without the need for a smartphone), and is only available on ESP32-based devices.

If enabled, all received messages are saved to the device's flash memory in a file named `rangetest.csv`.

To access this file, first turn on the Wi-Fi on your device and connect to your network. Once you can connect to your device, navigate to `meshtastic.local/rangetest.csv` (or your_device_ip/rangetest.csv) and the file will be downloaded automatically. This file will only be created after receiving initial messages.

To prevent filling up the storage, the device will abort writing if there is less than 50kb of space on the filesystem.

### Remove CSV file

When enabled, any previously created `rangetest.csv` file is removed from the device whenever a reboot is invoked; effectively deleting any past test results.

## Range Test Module Config Client Availability

**Android Apple CLI Web**

### Android



info

Range Test Config options are available for Android.

1. Open the Meshtastic App
2. Navigate to: **Settings > Range Test**

Android exports a rangetest.csv file from packets in the Debug Log. To clear old packet history data: `Debug Panel > Clear`

**Apple CLI Web**

## Apple



info

> All range test module config options are available on iOS, iPadOS and macOS at Settings > Module Configuration > Range Test.

Apple apps also have the option to download logged position data which is stored on your iPhone/iPad/Mac. Access this by clicking on the Nodes tab, selecting a node, then select Position Log and click Save. This data file does not require the Range Test module to be active.

**CLI Web**

### CLI



info

Range Test module config options are available in the python CLI. Example commands are below:

| Setting | Acceptable Values | Default |
| --- | --- | --- |
| range_test.enabled | `true`, `false` | `false` |

| Setting | Acceptable Values | Default |
| --- | --- | --- |
| range_test.save | `true`, `false` | `false` |
| range_test.clear_on_reboot | `true`, `false` | `false` |
| range_test.sender | `integer` (Seconds) | `0` |



tip

> Because the device will reboot after each command is sent via CLI, it is recommended when setting multiple values in a config section that commands be chained together as one.
>
> Example:
>
> ```
> meshtastic --set range_test.enabled true --set range_test.save false
> ```

Enable / Disable the range test

```
meshtastic --set range_test.enabled true
meshtastic --set range_test.enabled false
```

Enable / Disable range test save

```
meshtastic --set range_test.save true
meshtastic --set range_test.save false
```

Enable range test sender (send every 60 seconds)

```
meshtastic --set range_test.sender 60
```

Disable range test sender

```
meshtastic --set range_test.sender 0
```

# Application Examples

## Google Earth Integration

Steps:

1. Download and open Google Earth
    1. Select File > Import
    2. Select CSV
    3. Select Delimited, Comma
    4. Make sure the button that states “This dataset does not contain latitude/longitude information, but street addresses” is unchecked
    5. Select “rx lat” & “rx long” for the appropriate lat/lng fields
    6. Click finish
2. When it prompts you to create a style template, click yes.
    1. Set the name field to whichever column you want to be displayed on the map (don’t worry about this too much, when you click on an icon, all the relevant data appears)
    2. Select a color, icon, etc. and hit OK.

Your data will load onto the map, make sure to click the checkbox next to your dataset in the sidebar to view it.

## My Maps

You can use My Maps. It takes CSVs and the whole interface is much easier to work with.

See Google's Import map features from a file instructions.

You can style the ranges differently based on the values, so you can have the pins be darker the if the SNR or RSSI (if that gets added) is higher.

## OpenStreetMap - uMap

For an open source solution you could use uMap. A service based on the popular OpenStreetMap project.

Visit uMap in your preferred language.

1. Click on "Create a map"
2. Click on the "Import data"-Icon or press CTRL + I.
3. Choose your rangetest.csv and click "Import". The dataformat is recognized and the locations are imported onto the map.

# Traceroute

From https://meshtastic.org/docs/configuration/module/traceroute/

## Overview

Due to the limited bandwidth of LoRa, Meshtastic does not keep track of the nodes a message used to hop to the destination. However, from firmware 2.0.8 on, there is a Traceroute Module that can show you this.

Only nodes that know the encryption key of the channel you use can be tracked. Also note that a message may arrive via multiple routes due to duplication because of rebroadcasting. This module will only track the hops of the first packet containing the traceroute request that arrived at the destination.

In order to use it, make sure your devices use firmware version 2.0.8 or higher.

From firmware version ≥ 2.5, the route back to the origin is recorded, along with the Signal-to-Noise Ratio (SNR) for each link. Nodes with older firmware or those unable to decrypt the traceroute will be shown as "Unknown" (represented by an ID equal to 4294967295 or 0xFFFFFFFF in hexadecimal) if all nodes in the route are using at least firmware version 2.3.12.

Each direction records up to 8 hops. The firmware records per-hop SNR in 0.25 dB steps, as measured by the reporting node on its incoming link.

## Cost and rate limiting

A traceroute requests acknowledgement, so the mesh retransmits it until a node acknowledges it. It costs more airtime than a text message. Use it as a diagnostic, not as a monitor.

The sending node allows one traceroute every 30 seconds. Exceeding the limit produces no mesh error: the node never transmits the request, and reports the limit to the client locally. The responding node has no limit and answers every request it receives.

The firmware refuses a broadcast traceroute of more than one hop, and drops it silently.

## Channels and encryption

Traceroute uses channel encryption, never public-key encryption. Only nodes holding the key for the channel can add themselves to the route. Nodes without the key still forward the packet, but appear as unknown hops.

The response returns on the channel that carried the request. The channel used to send depends on the client:

| Client | Channel used |
|---|---|
| CLI | Channel 0, or `--ch-index` if given |
| Apple | Always channel 0 |
| Android | The channel the destination was last heard on, otherwise 0 |
| Web | Primary channel |
| Device menu | Channel 0 |

Only the CLI allows choosing the channel.

## Repeater Behavior

Repeater nodes will appear in the traceroute log if they have the encryption key, but if they are not in the node list they might appear as "Unknown" depending on the app.

## Traceroute Module Client Availability

**Android Apple CLI Web**

### Android

Long press a node in the node list and select **Trace Route**. Alternatively, open the node's details, find **Traceroute** under Telemetry, and select the refresh button.

The result appears in a pop-up anywhere in the app. Node details also keeps a history under Telemetry as **Traceroute**, with a chart and a map view.

The buttons in node details enforce the 30 second limit by greying out with a countdown ring, and display no message. The node list long press item has no such limit.

The app applies no timeout. An unanswered request stays in the log as **No Response**.

**Apple CLI Web**

### Apple

Long press a node in the node list and select **Trace Route**, or use **Trace Route** in the Actions section of the node's detail view.

The node's detail view keeps results under Logs as **Trace Route Log**. Settings > Logging > **Trace Routes** lists them for every node. The app posts a **Traceroute Complete** notification when a reply arrives, showing the hop count in each direction.

During the 30 second limit the button becomes a non-tappable countdown reading `Trace Route (in Ns)`. The limit covers all nodes together, not each node separately.

The app applies no timeout. An unanswered request reads “A Trace Route was sent, no response has been received.”

When the firmware rejects the send, the log reads “Trace Route was rate limited. You can send a trace route a maximum of once every thirty seconds.”

**CLI Web**

## CLI

Make sure the CLI is at least version 2.0.6. Then use this command:

CLI traceroute command

```
meshtastic --traceroute 'destinationId'
```

For `destinationId`, use the ID of the node to trace, available from `meshtastic --nodes`. Both the `!ba4bf9d0` and `0xba4bf9d0` forms work. Some shells require quotation marks around the ID.

Traceroute from !25048234 to !bff18ce4

```
meshtastic --traceroute '!bff18ce4'
Connected to radio
Sending traceroute request to !bff18ce4 on channelIndex:0 (this could take a while)
Route traced towards destination:
!25048234 --> !ba4bf9d0 (6.0dB) --> !bff18ce4 (-3.5dB)
Route traced back to us:
!bff18ce4 --> !ba4bf9d0 (-2.75dB) --> !25048234 (5.25dB)
```

The first ID shown is the origin device connected via the CLI. In the example above, the packet traveled through one additional node to reach its destination and return.

The CLI reports SNR in 0.25 dB steps, so values such as `-2.75dB` are normal. A hop with unknown SNR prints as `(?dB)`. The return section appears only when the response carries a complete set of return SNRs.

### Choosing a channel

Traceroute uses channel 0 unless `--ch-index` selects another. Both nodes must hold the key for that channel.

Traceroute on channel 1

```
meshtastic --ch-index 1 --traceroute '!bff18ce4'
```

When the channel index does not exist or is disabled, the command exits without sending, and prints no warning.

### Errors

`Aborting due to: Timed out waiting for traceroute` — no reply arrived in time. The destination is unreachable, out of hops, or on a channel the origin does not share. The wait is `--timeout`, default 300 seconds, scaled by the number of hops.

`Aborting due to: Timed out waiting for connection completion` — the CLI never finished connecting to the local radio. The limit is 30 seconds for serial and TCP, and 60 seconds for BLE. This is a local problem, not a mesh problem.

`Traceroute failed: <REASON>` — the mesh returned a routing error such as `NO_CHANNEL` or `MAX_RETRANSMIT`. The CLI prints the reason and exits normally.

# Hardware

## Antennas

From https://meshtastic.org/docs/hardware/antennas/

### Pages

Antenna Testing
Guidance for testing and using your antennas.

Antenna Reports
Community submitted SWR testing.

Antenna Selection
Considerations for choosing an antenna.

Additional Resources
Antenna designs, tools, and coverage simulators.

### Community Favorites

These antennas have garnered positive feedback and frequent recommendations within our community. However, we encourage you to conduct your own research to ensure the best choice for your needs. We are not responsible for individual experiences or results.

#### Compact / Handheld

| Frequency | Model | Purchase Links |
|---|---|---|
| 915 or 868 | GIZONT 17cm | muzi works AliExpress |
| 915 | LINX ANT-916-CW-HW-SMA | Mouser DigiKey |
| 915 | ZIISOR TX915-JKS-20 (200mm) | eBay |
| 915 | CDEBYTE TX915-JKD-20 (5-pack) | AliExpress |
| 915 | Rabbit Labs / ZIISOR | Amazon US |
| 433 | Rabbit Labs / ZIISOR | Amazon US |
| 906 | Smiley 906 Slimline | Smiley Antenna |

#### Base Station / Repeater

| Frequency | Model | Purchase Links |
|---|---|---|
| 915 | Alfa AOA-915-5ACM | Rokland |
| 915 | Rokland 32" 5.8 dBi | Rokland |
| 868 | MikroTik 868 Omni 6.5dBi | MikroTik |

## Automotive

| Frequency | Model | Purchase Links |
|---|---|---|
| 915 | Laird MA9-5N | DigiKey |
| 915 | Taoglas TI.16.5F11 | DigiKey |

# LoRa antennas

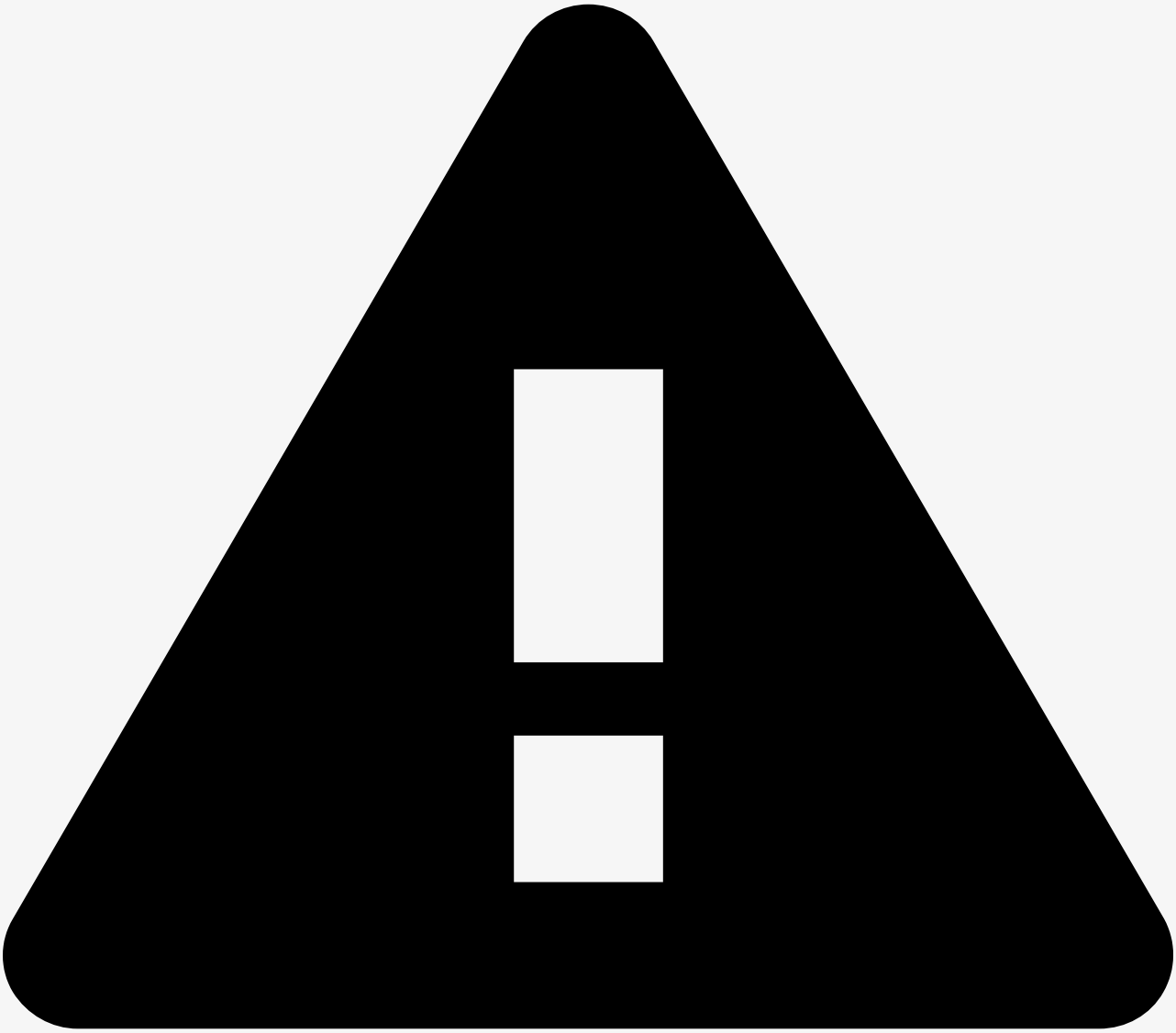
From https://meshtastic.org/docs/hardware/antennas/lora-antenna/

The stock antennas provided with the T-Beam and other boards usually come from 'mixed bags'. They may not have been designed or tuned for your given frequency range, and they may not be of a quality design.

Matching an antenna to the transceiver frequency is important, as is choosing an appropriate design.

The antenna's design will affect:

- Efficiency - The proportion of the signal which leaves the antenna
- Direction in which the signal is transmitted
- Interference by horizontal or vertical polarization
- Amount of signal which is reflected back to the device itself



caution

While the LoRa devices we use for Meshtastic are relatively low power radios, care should be taken *not* to operate any radio transmission device without an antenna or with a poorly matched antenna. Radio signals transmitted without an antenna can reflect back and damage the device.

## Important considerations

### What transmission frequency are you using?

Devices on another frequency will not be able to interact with yours. See this listing by The Things Network for frequencies licensed for specific countries.

### How will you be carrying / transporting the radio?

A large directional antenna will transmit over significantly greater distance than an omni-directional antenna. However, it must be pointed at its target - so it is not optimal for mobile use.

A tuned half-wave whip antenna may have more omni-directional range than the quarter wave stubby; but it will be conspicuous in your pocket.

Many antennas, especially quarter wave stubby antennas, require the use of ground planes to transmit at peak performance.

### Do you want transmission in all directions?

While humans (mostly water) don't attenuate signal greatly (at LoRa frequencies), buildings & walls do. If your antenna is permanently positioned against a building, signal transmitted towards the wall will be largely lost or attenuated.

### Does my Meshtastic device have the right power range, impedance, and connector for the antenna?

For the LoRa devices, it should be 50 Ohm impedance with SMA connector. Many antennas will be recommended for LoRa use in their technical details.

In contrast, a close range, contact-less Personal Area Network antenna, or a huge antenna at the end of length of coax designed for a 100W transmitter, are not going to be operable.

### Cost, quality, and supply service?

The perfect antenna on paper, sourced from the other side of the world with mixed reviews, doesn't compare to a local supplier who has spent time carefully collating all of the antenna data-sheets for comparison *and* holds stock immediately available. Personally, I prefer to pay significantly more for a time saving, quality service.

### How close will the antenna be to my Meshtastic device?

Most cables will significantly degrade the signal strength over any significant distance. It is often more effective to place a node outside than to have it indoors with the antenna outside. The exception might be if there is extreme heat, cold, or humidity, and if the shortest possible low loss cable is used.

Still, a proper enclosure should mitigate bad weather.

## Terminology / references

You could do a lot worse than reading the Wikipedia entry for Antenna, along with the Wikipedia entry for LoRa.

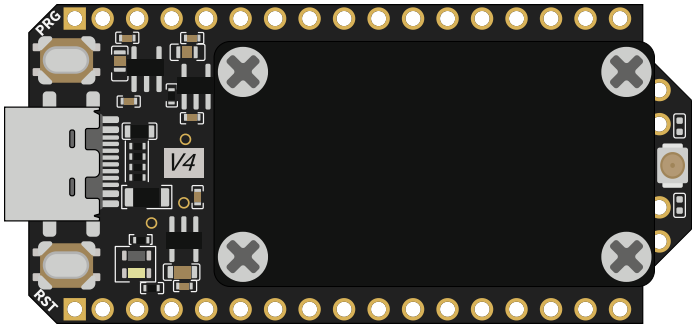
Instead of listing the terms, let us recommend this superb tutorial by Andreas Spiess (the 'guy with the Swiss accent').

# Heltec WiFi LoRa 32 (the ARK Vault's radio)

From https://meshtastic.org/docs/hardware/devices/heltec-automation/lora32/

**LoRa32 V4 LoRa32 V4-R8 LoRa32 V3/V3.1 Wireless Stick Lite V3 Wireless Tracker v1.0 Wireless Tracker v1.1 Wireless Tracker V2 Wireless Paper v1.0 Wireless Paper v1.1**

## V4





info

A newer **V4-R8** variant is also available, which uses the **ESP32-S3R8** chip and has a different pinout and firmware. If your board has the ESP32-S3R8, use the V4-R8 tab instead.

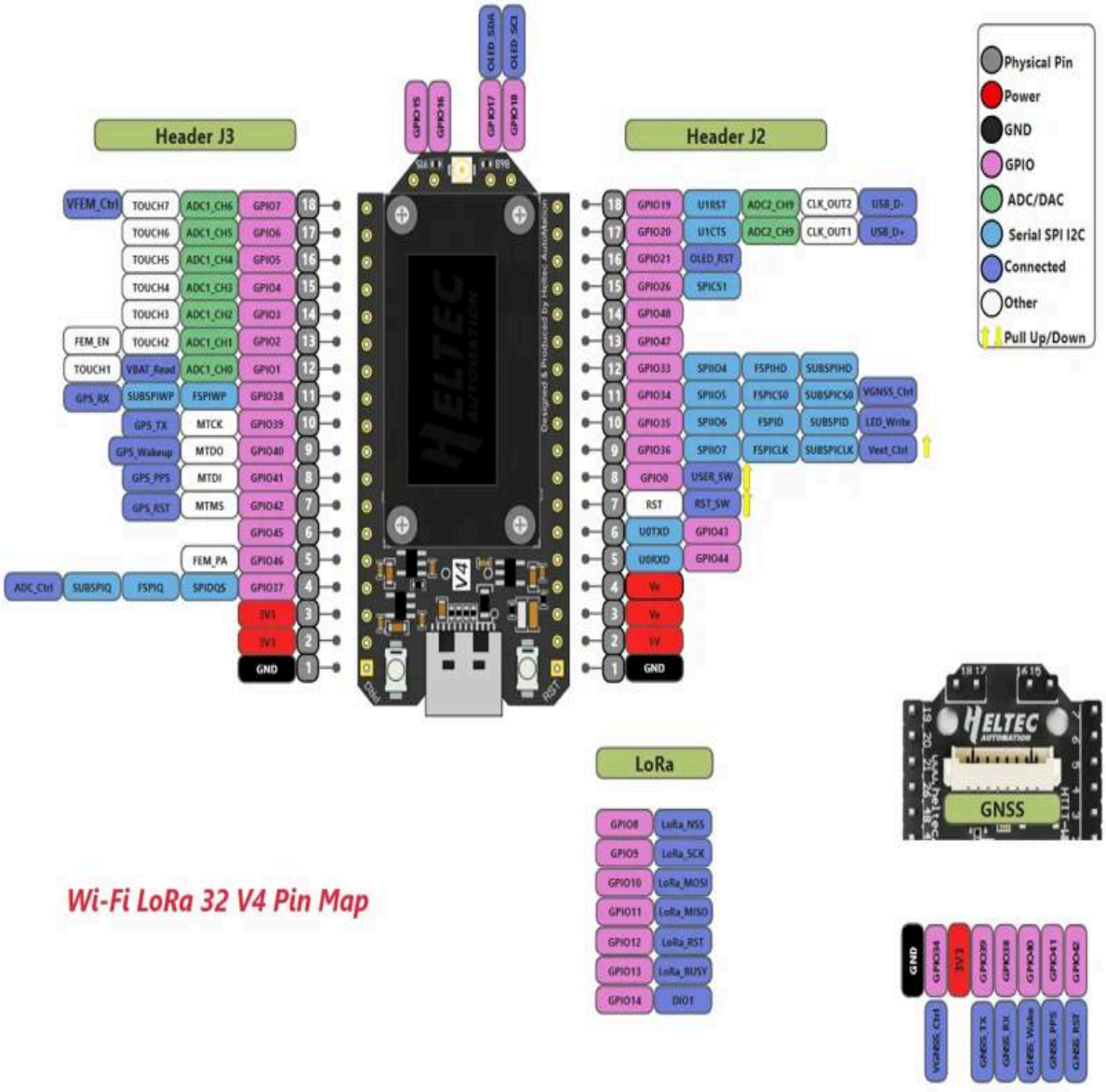
- **MCU:**
  - ESP32-S3R2 (Wi-Fi & Bluetooth)

- **LoRa Transceiver:**
  - Semtech SX1262
- **Frequency Options:**
  - 863 - 870 MHz
  - 902 - 928 MHz
- **Connectors:**
  - USB-C
  - Antenna:
    - Integrated antenna for Wi-Fi/Bluetooth
    - U.FL/IPEX antenna connector for LoRa

## Features

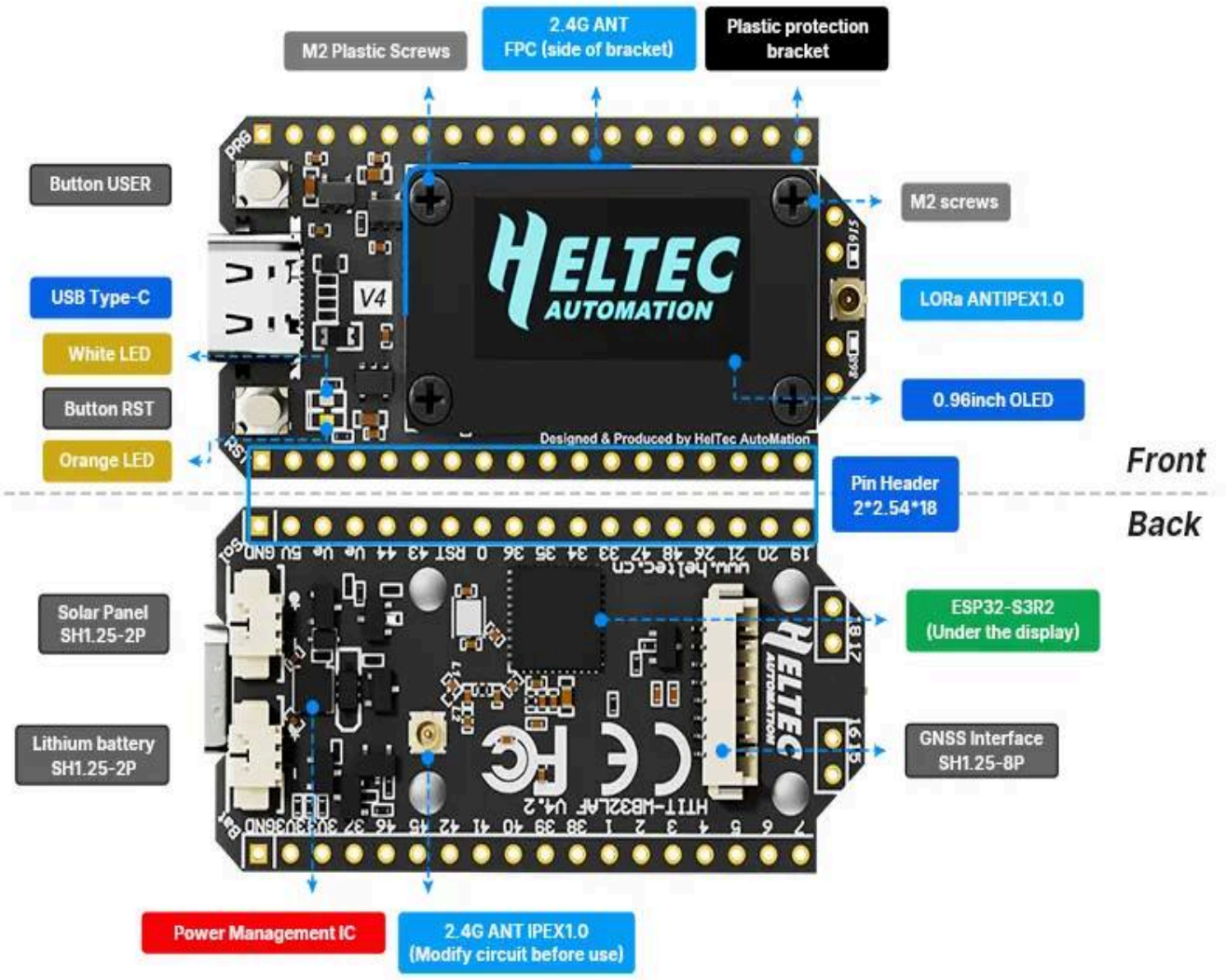
- Built in 0.96 inch OLED display
- High power option: 28±1dBm
- Optimized lithium battery management
- Form factor and pin compatibility with V3/V3.1
- 1.25-2Pin solar interface
- 1.25-8Pin GNSS interface

## Pin Map



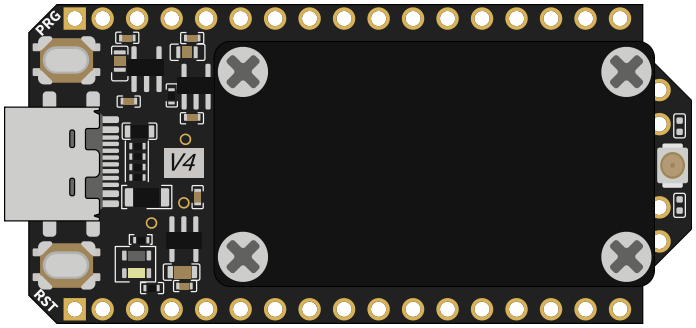

## Resources

- Firmware file: `firmware-heltec-v4-X.X.X.xxxxxxx.bin`
- Purchase links
  - US
    - Rokland
  - International
    - Heltec
    - Hexaspot




**LoRa32 V4-R8 LoRa32 V3/V3.1 Wireless Stick Lite V3 Wireless Tracker v1.0 Wireless Tracker v1.1 Wireless Tracker V2 Wireless Paper v1.0 Wireless Paper v1.1**

## V4-R8






caution

> The **V4-R8** is a separate hardware variant from the original V4. It uses the **ESP32-S3R8** chip with a different pinout and requires its own firmware. Do not flash the original `firmware-heltec-v4-*` firmware on a V4-R8 board.

- **MCU:**
  - ESP32-S3R8 (Wi-Fi & Bluetooth)
- **Flash:**
  - 16 MB external
- **PSRAM:**
  - 8 MB
- **LoRa Transceiver:**
  - Semtech SX1262
- **Frequency Options:**
  - 863 - 870 MHz
  - 902 - 928 MHz
- **Connectors:**
  - USB-C
  - Antenna:
    - Integrated antenna for Wi-Fi/Bluetooth
    - U.FL/IPEX antenna connector for LoRa
  - SH1.25-2Pin solar panel interface
  - SH1.25-8Pin GNSS interface

## Features

- Built in 0.96 inch OLED display (OLED variant) or TFT display (TFT variant)
- High power option: 28±1dBm
- Optimized lithium battery management
- 8 MB PSRAM for maps and more complex UI
- Similar physical form factor to the V4, but the pinout is not compatible

## Pin Map

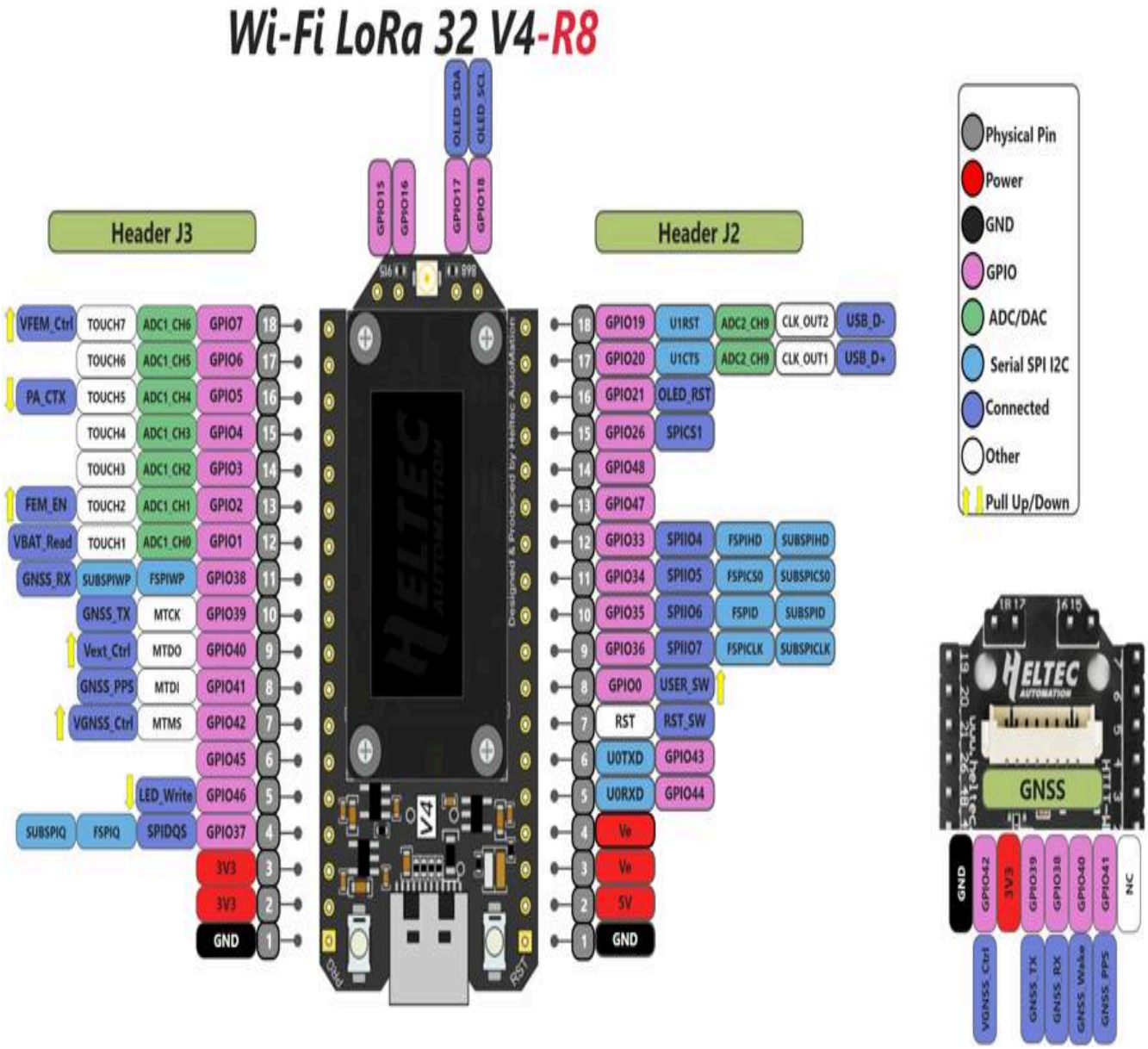


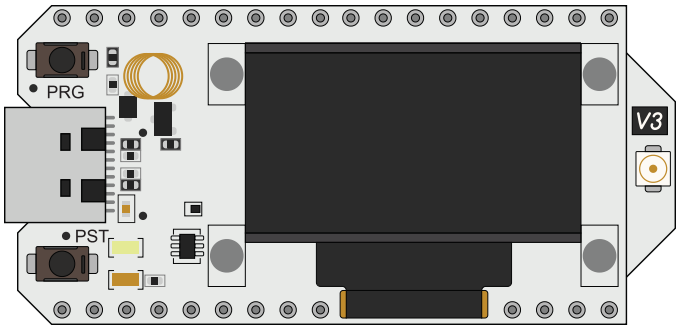

Image Source: Heltec

## Resources

- Firmware files:
  - OLED: `firmware-heltec-v4-r8-oled-X.X.X.xxxxxxx.bin`
  - TFT: `firmware-heltec-v4-r8-tft-X.X.X.xxxxxxx.bin`
- Use the latest Meshtastic firmware (2.7.25 or newer). In the Web Flasher, select **Heltec LoRa32 V4-R8**.
- Purchase links
  - International
    - Heltec

**LoRa32 V3/V3.1 Wireless Stick Lite V3 Wireless Tracker v1.0 Wireless Tracker v1.1 Wireless Tracker V2 Wireless Paper v1.0 Wireless Paper v1.1**

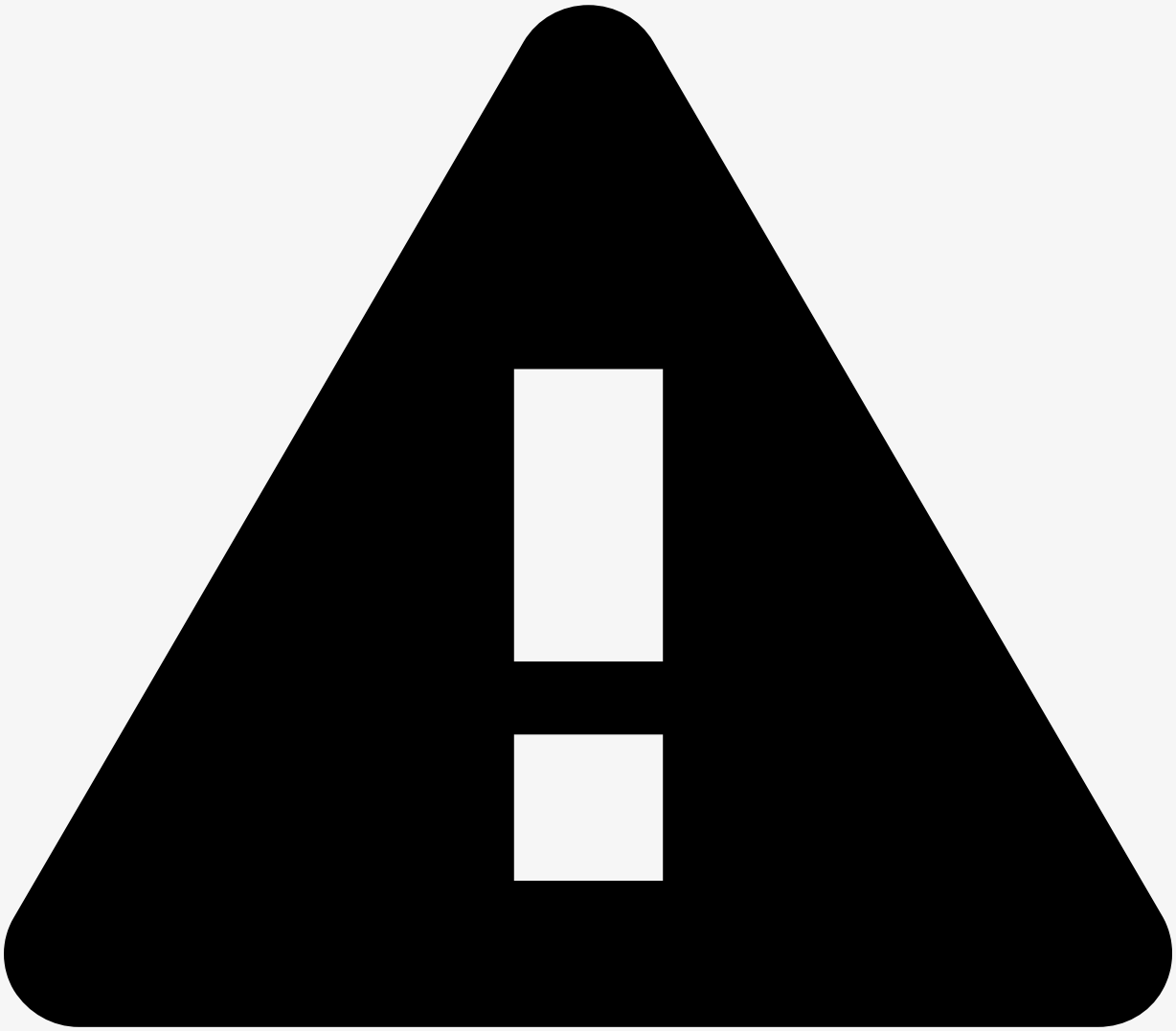
## V3/V3.1





info

This device may have issues charging a connected battery if utilizing a USB-C to USB-C cable. It's recommended to use a USB-A to USB-C cable.



caution

Be careful when interacting with the USB-C port. This device does not have ESD protection for the CP2102 USB to UART bridge chip.

- **MCU:**
  - ESP32-S3FN8 (Wi-Fi & Bluetooth)
- **LoRa Transceiver:**
  - Semtech SX1262
- **Frequency Options:**
  - 433 MHz
  - 470 - 510 MHz
  - 863 - 870 MHz

    - 902 - 928 MHz
- **Connectors:**
    - USB-C
    - Antenna:
        - Dedicated 2.4 GHz metal spring antenna for Wi-Fi/Bluetooth
        - U.FL/IPEX antenna connector for LoRa

## V3.1 differences

Firmware remains the same as V3 below. Compare schematics: V3.0 and V3.1. Key differences:

- Removal of FDG6322C (a dual N & P channel FET) from the V3.1 power supply.
- Antenna filter values in V3.1 (L11 = 1.8pF, C15 = 2.7nH, C24 = 1.8pF) align more closely with ESP32-S3 reference design than V3.0 (L11 = 1.6nH, C15 = 6.9pF, C24 = 2.4pF).

## Features

- Built in 0.96 inch OLED display
- User and Reset Buttons
- No GPS

## Meshtastic I2C Definitions

- SDA: GPIO41
- SCL: GPIO42

## Pin Map

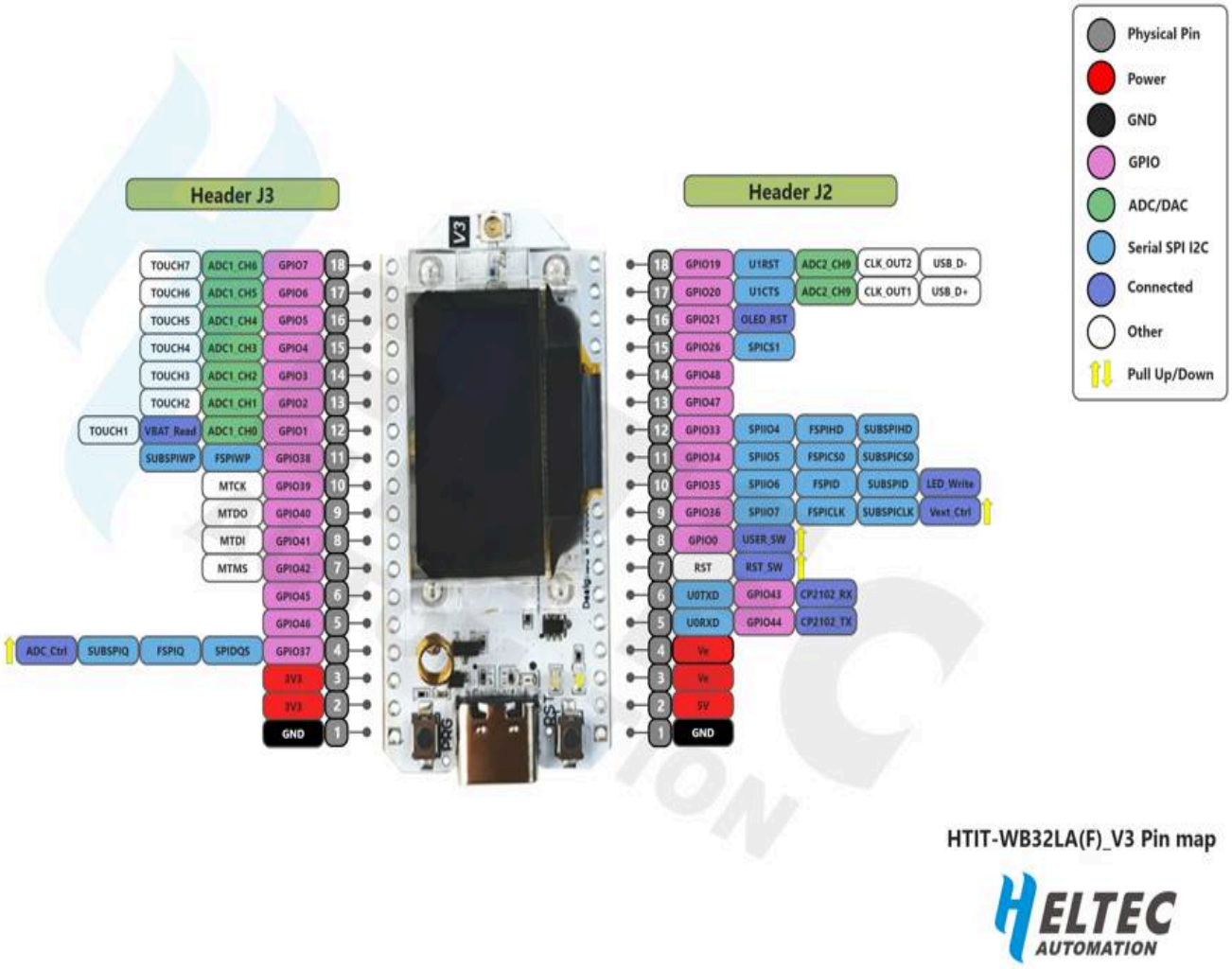



Image Source: Heltec

## Resources

- Firmware file: `firmware-heltec-v3-X.X.X.xxxxxxx.bin`
- Purchase links
  - US
    - muzi works
    - Rokland
  - International
    - Heltec
    - AliExpress
    - Hexaspot

**Wireless Stick Lite V3 Wireless Tracker v1.0 Wireless Tracker v1.1 Wireless Tracker V2 Wireless Paper v1.0 Wireless Paper v1.1**

## Wireless Stick Lite V3



info

> This device may have issues charging a connected battery if utilizing a USB-C to USB-C cable. It's recommended to use a USB-A to USB-C cable.

- **MCU:**
  - ESP32-S3FN8 (Wi-Fi & Bluetooth)
- **LoRa Transceiver:**
  - Semtech SX1262
- **Frequency Options:**
  - 433 - 510 MHz
  - 470 - 510 MHz
  - 863 - 870 MHz
  - 902 - 928 MHz
- **Connectors:**
  - USB-C
  - Antenna:
    - Dedicated 2.4 GHz stamped metal antenna for Wi-Fi/Bluetooth
    - U.FL/IPEX antenna connector for LoRa (next to the V3 icon)

## Features

- No display.
- User and Reset Buttons
- Additional GPIO availability
- No GPS

## Meshtastic I2C Definitions

- SDA: GPIO41
- SCL: GPIO42

## Pin Map

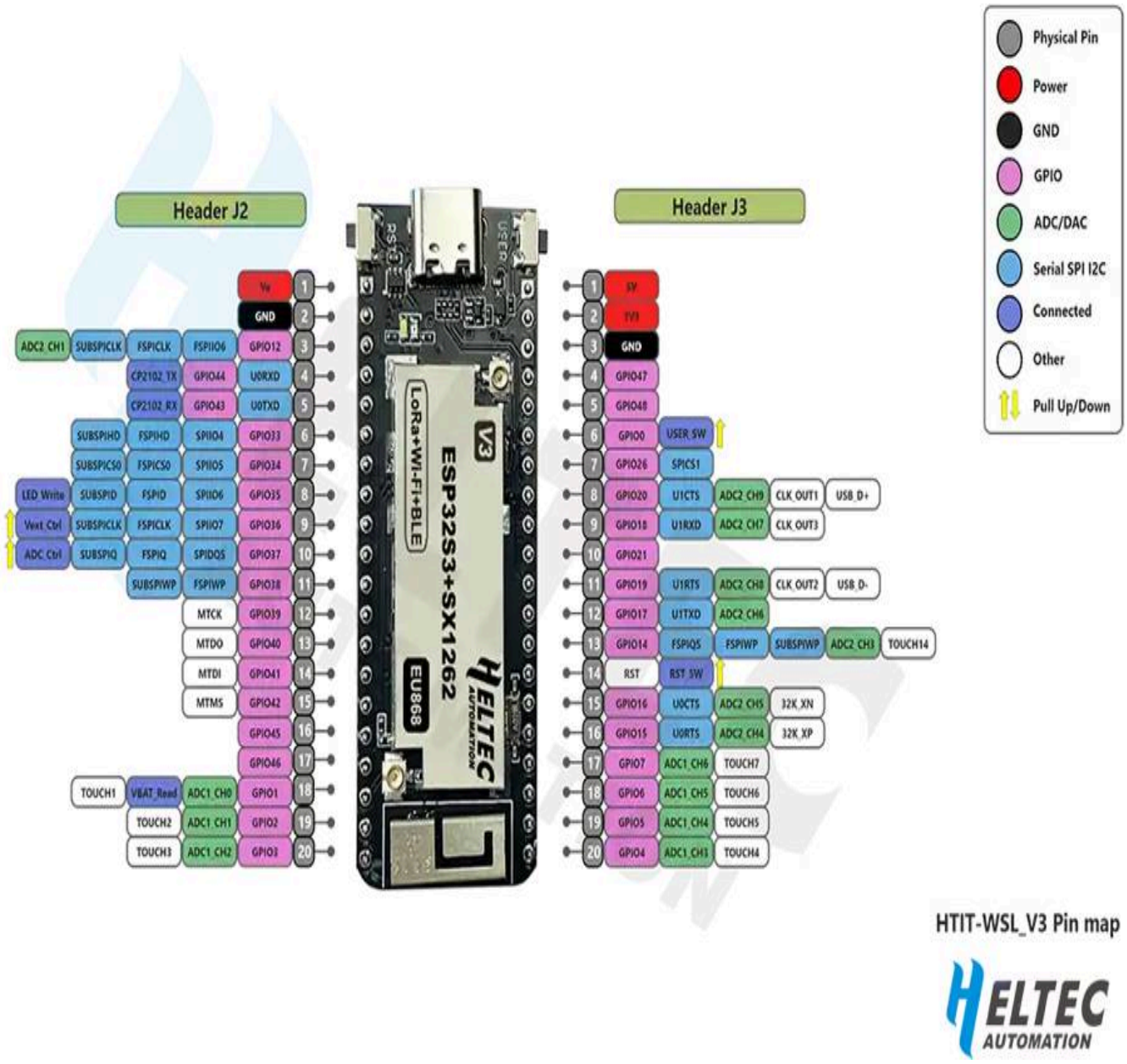



Image Source: Heltec

## Hardware Resources

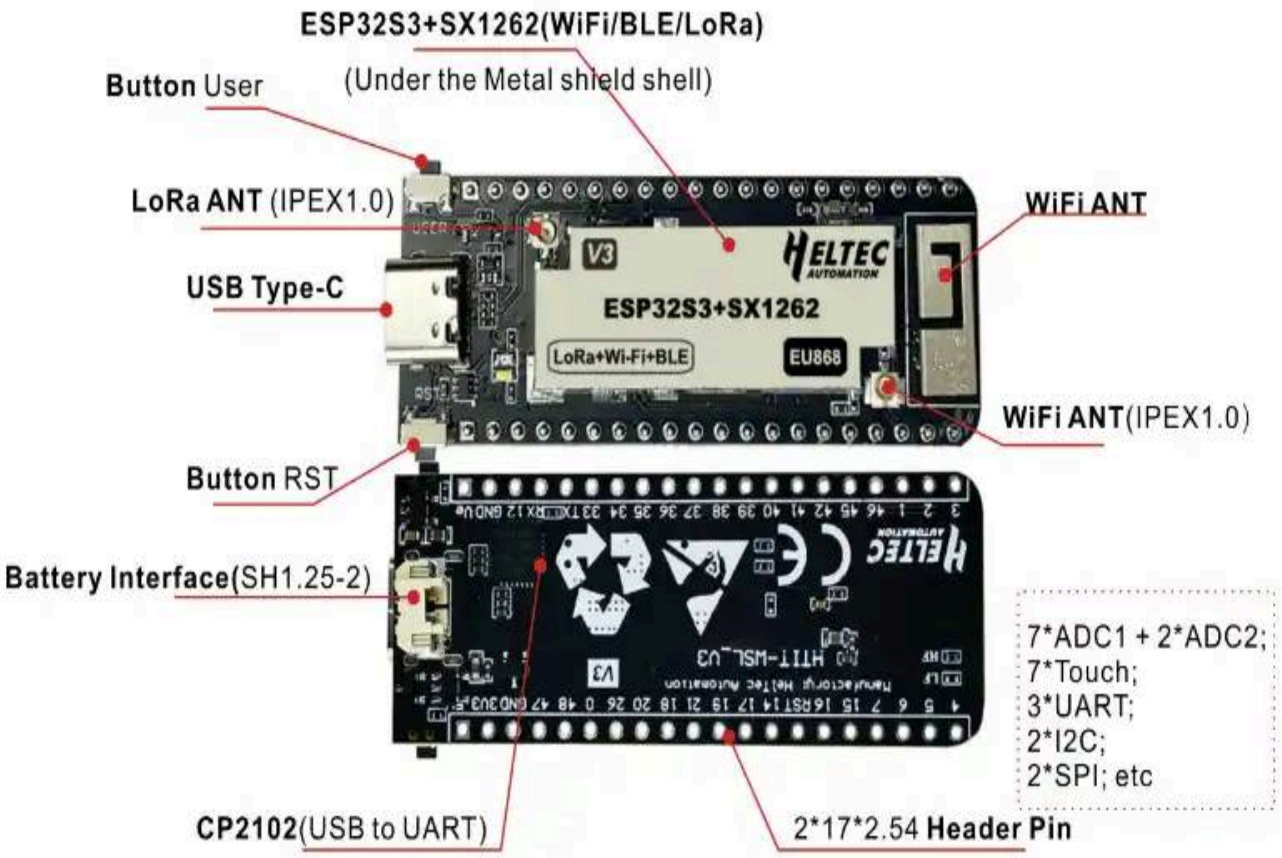



Image Source: Heltec

There are two antenna connectors. You need to connect the LoRa antenna while the Wi-Fi antenna is optional.

### Resources

- Firmware file: `firmware-heltec-wsl-v3-X.X.X.xxxxxxx.bin`
- Purchase Links:
  - US
    - Rokland
  - International
    - Heltec
    - AliExpress
- From Heltec
  - Documentation (docs.heltec.org)
  - Schematic Diagram (PDF)
  - LoRa Node Development Kit (PDF)

**Wireless Tracker v1.0 Wireless Tracker v1.1 Wireless Tracker V2 Wireless Paper v1.0 Wireless Paper v1.1**

## Wireless Tracker v1.0



info

> This device may have issues charging a connected battery if utilizing a USB-C to USB-C cable. It's recommended to use a USB-A to USB-C cable.

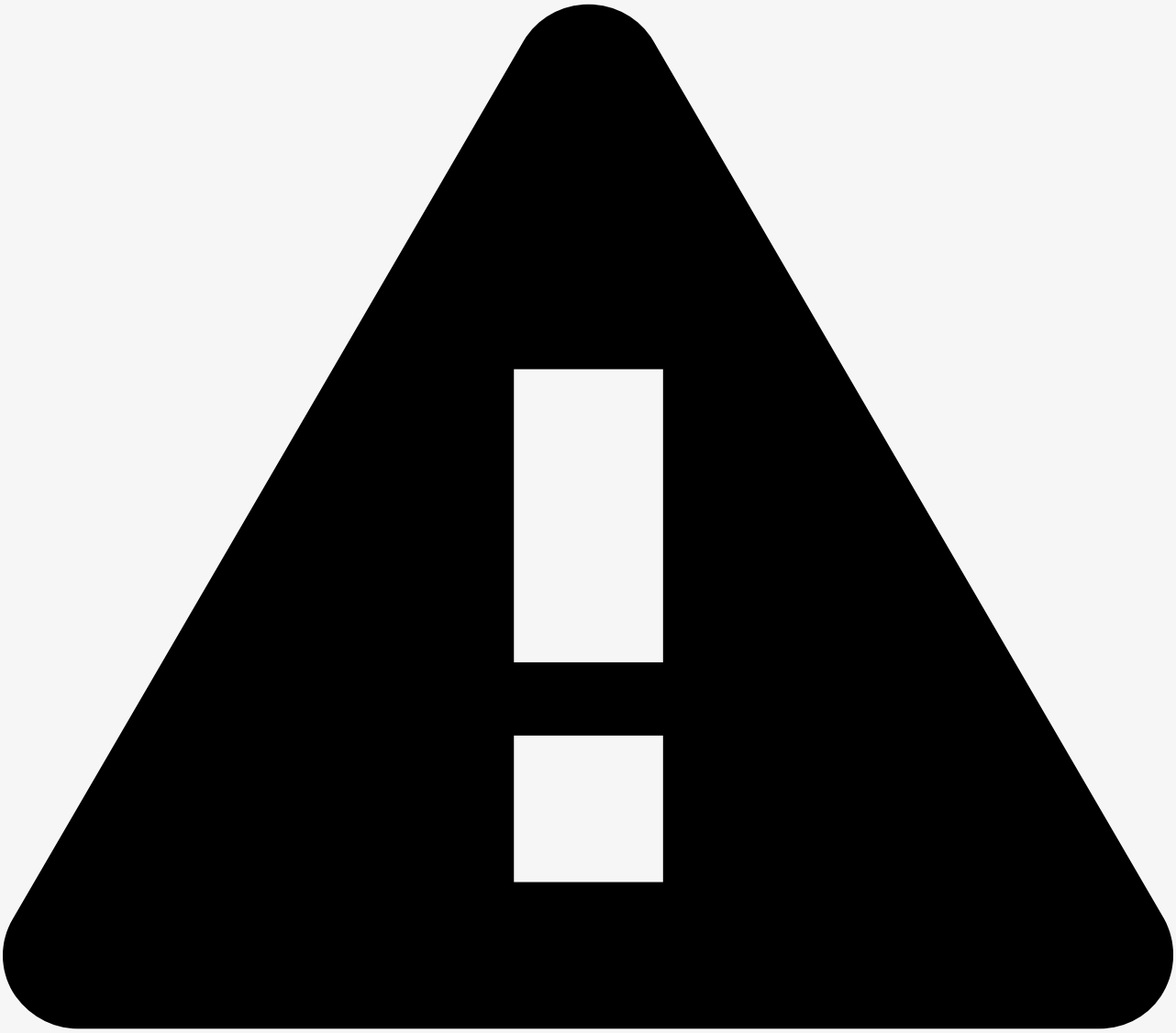
- **MCU:**
  - ESP32-S3FN8 (Wi-Fi & Bluetooth)
- **LoRa Transceiver:**
  - Semtech SX1262
- **Frequency Options:**
  - 470 - 510 MHz
  - 863 - 870 MHz
  - 902 - 928 MHz
- **Connectors:**
  - USB-C
  - Antenna:
    - Dedicated 2.4 GHz metal spring antenna for Wi-Fi/Bluetooth
    - U.FL/IPEX antenna connector for LoRa and GNSS

## Features

- Onboard 0.96-inch LCD display
- User and Reset Buttons

## Flashing

To flash ESP32-S3 devices like the Wireless Tracker, you typically need to place them in Espressif's firmware download mode. Use the "1200bps reset" button in the web flasher to do this. If this method does not work for any reason, you can follow the manual process below.



warning

Do not proceed unless an antenna is connected to avoid possible damage to the device's radio.

The following process will manually place the device into the Espressif Firmware Download mode:

1. Unplug the device.
2. Press and hold the USER button.
3. Plug device in.
4. After 2-3 seconds, release the USER button.

With the device now in the Espressif Firmware Download mode, you can proceed with flashing using one of the supported flashing methods. It's generally recommended to use the Web Flasher. You can select "Heltec Wireless Tracker" from the device drop-down.

## Pin Map

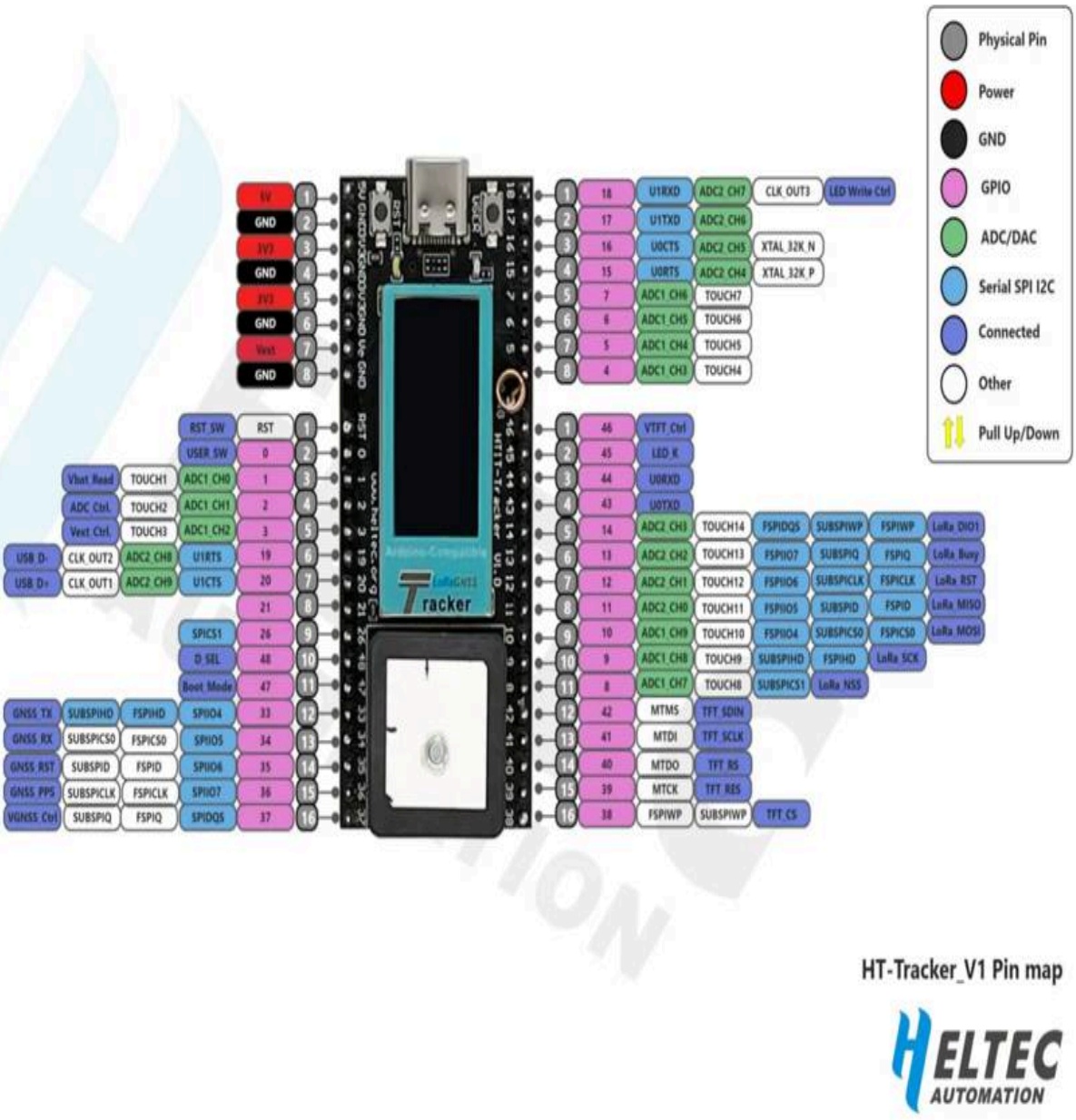



Image Source: Heltec

## Resources

- Firmware file: `firmware-heltec-wireless-tracker-V1-0-X.X.X.xxxxxxx.bin`



note

Heltec revised the Wireless Tracker schematics and released a V1.1, most devices being sold are now V1.1.

**Wireless Tracker v1.1 Wireless Tracker V2 Wireless Paper v1.0 Wireless Paper v1.1**

## Wireless Tracker v1.1



info

This device may have issues charging a connected battery if utilizing a USB-C to USB-C cable. It's recommended to use a USB-A to USB-C cable.

- **MCU:**

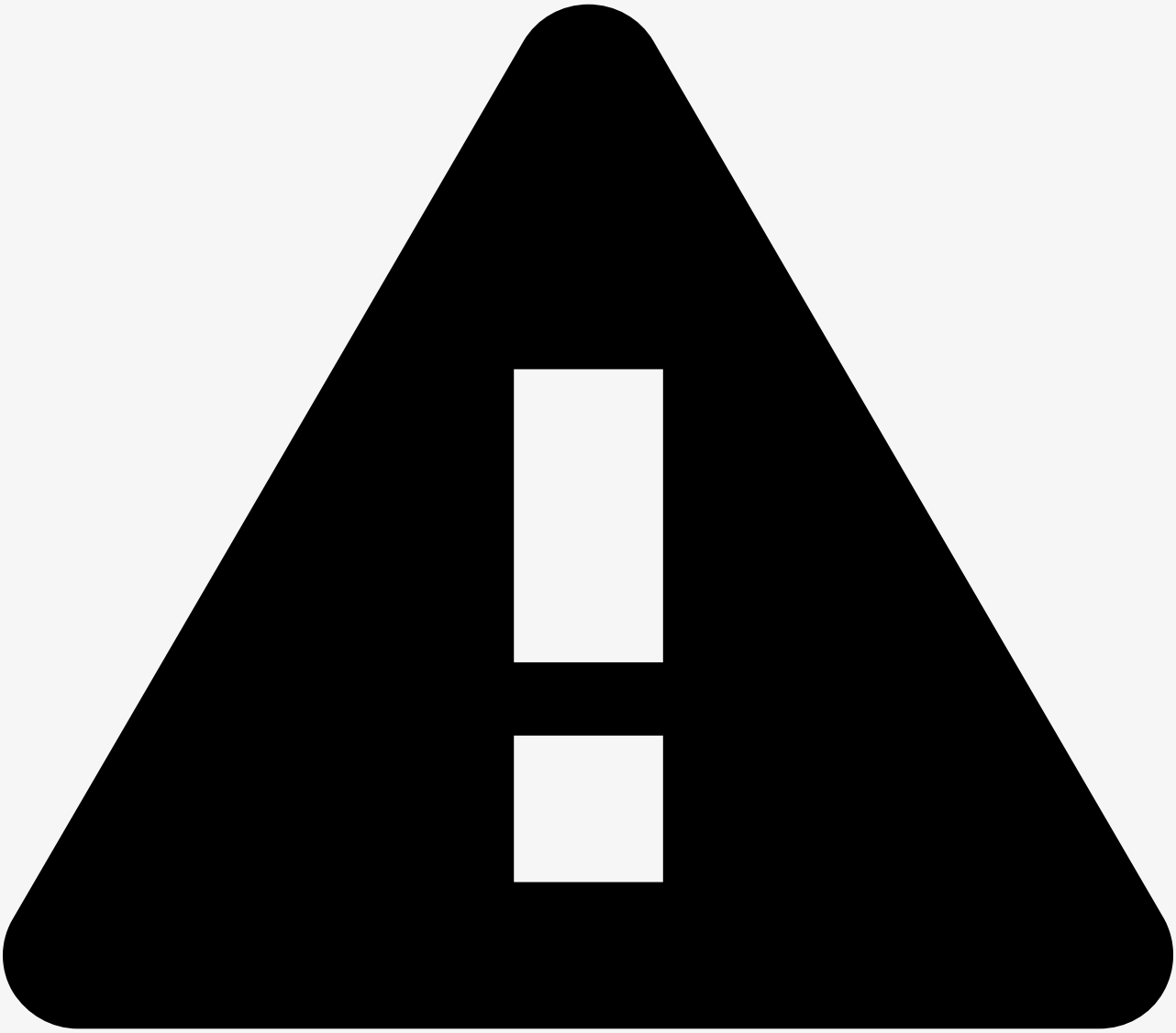
    - ESP32-S3FN8 (Wi-Fi & Bluetooth)
- **LoRa Transceiver:**
    - Semtech SX1262
- **Frequency Options:**
    - 470 - 510 MHz
    - 863 - 870 MHz
    - 902 - 928 MHz
- **Connectors:**
    - USB-C
    - Antenna:
        - Dedicated 2.4 GHz metal spring antenna for Wi-Fi/Bluetooth
        - U.FL/IPEX antenna connector for LoRa and GNSS

## Features

- Onboard 0.96-inch LCD display
- User and Reset Buttons

## Flashing

To flash ESP32-S3 devices like the Wireless Tracker, you typically need to place them in Espressif's firmware download mode. Use the "1200bps reset" button in the web flasher to do this. If this method does not work for any reason, you can follow the manual process below.



warning

Do not proceed unless an antenna is connected to avoid possible damage to the device's radio.

The following process will manually place the device into the Espressif Firmware Download mode:

1. Unplug the device.
2. Press and hold the USER button.
3. Plug device in.
4. After 2-3 seconds, release the USER button.

With the device now in the Espressif Firmware Download mode, you can proceed with flashing using one of the supported flashing methods. It's generally recommended to use the Web Flasher. You can select "Heltec Wireless Tracker" from the device drop-down.

## Meshtastic I2C Definitions

- SDA: GPIO45
- SCL: GPIO46

## Pin Map

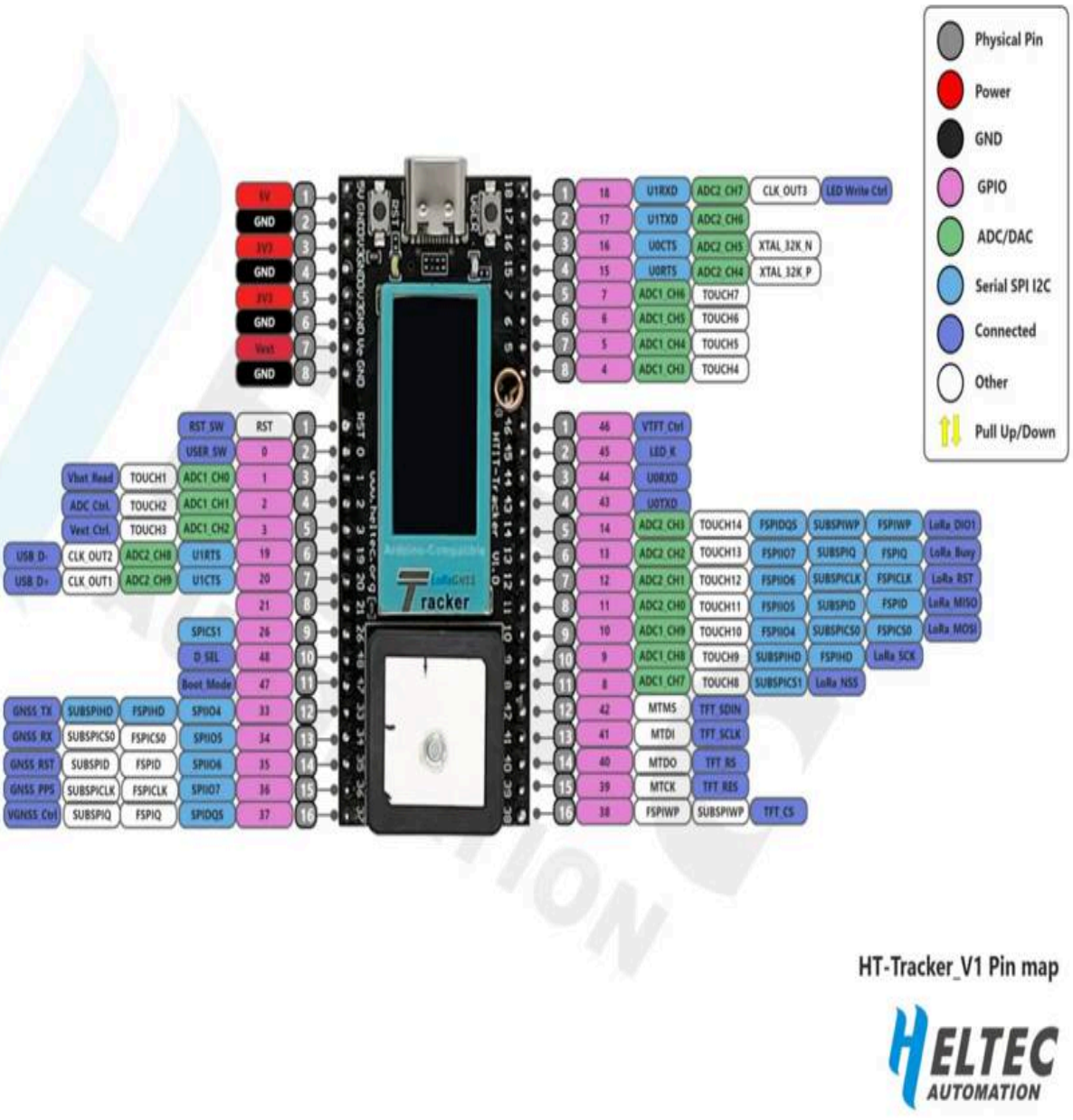


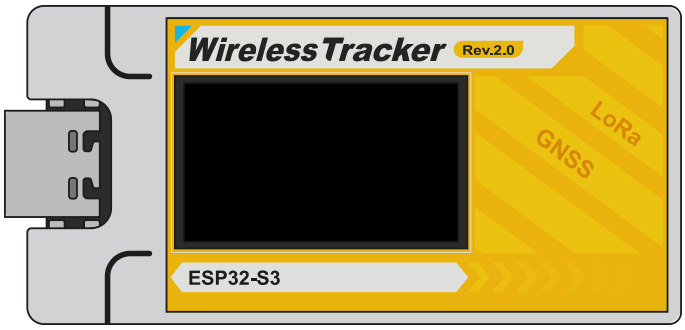

Image Source: Heltec

## Resources

- Firmware file: `firmware-heltec-wireless-tracker-X.X.X.xxxxxxx.bin`
- Purchase Links:
  - US
    - Rokland
  - International
    - Heltec
    - AliExpress

**Wireless Tracker V2 Wireless Paper v1.0 Wireless Paper v1.1**

## Wireless Tracker V2




The Wireless Tracker V2 is the revised version of Heltec's Wireless Tracker, built on the ESP32-S3 with a high-power (28±1 dBm) SX1262 LoRa radio, a dual-band UC6580 GNSS, and a 0.96-inch color TFT display, plus battery and solar power management.

- **MCU:**
  - ESP32-S3FN8 (Wi-Fi & Bluetooth)
- **LoRa Transceiver:**
  - Semtech SX1262 (up to 28±1 dBm)
- **Frequency Options:**
  - 863 - 870 MHz
  - 902 - 928 MHz
- **Navigation Module:**
  - UC6580 dual-band GNSS (GPS, GLONASS, BDS, Galileo, NAVIC, QZSS)
- **Connectors:**
  - USB-C
  - Antenna:
    - U.FL/IPEX antenna connector for LoRa
    - Integrated antenna for Wi-Fi/Bluetooth

### Features

- Onboard 0.96-inch color TFT display (160 x 80).
- Dual-band UC6580 GNSS.
- High-power SX1262 LoRa radio (up to 28±1 dBm).

- Battery and solar smart power management.

### Flashing

To flash ESP32-S3 devices like the Wireless Tracker, you typically need to place them in Espressif's firmware download mode. Use the "1200bps reset" button in the web flasher to do this.

### Resources

- Firmware file: `firmware-heltec-wireless-tracker-v2-X.X.X.xxxxxxx.bin`
- Purchase Links:
    - International
        - Heltec

**Wireless Paper v1.0 Wireless Paper v1.1**

## Wireless Paper V1.0



info

**The Wireless Paper V1.0 is listed as "Wireless Paper V1.0" in the firmware files and on the web flasher.**

This device may have issues charging a connected battery if utilizing a USB-C to USB-C cable. It's recommended to use a USB-A to USB-C cable.

- **MCU:**
  - ESP32-S3FN8 (Wi-Fi & Bluetooth)
- **LoRa Transceiver:**
  - Semtech SX1262
- **Frequency Options:**
  - 433 MHz
  - 470 - 510 MHz
  - 863 - 870 MHz
  - 902 - 928 MHz
- **Connectors:**
  - USB-C
  - Antenna:
    - U.FL/IPEX antenna connector for LoRa
    - Integrated 2.4 GHz PCB antenna

### Features

- Onboard 2.13-inch black and white E-Ink display screen
- User and Reset switches
- No GPS

### Resources

- Firmware file: `firmware-heltec-wireless-paper-v1_0-X.X.X.xxxxxxx.bin`
- Purchase Links:
  - International
    - Heltec
    - AliExpress

**Wireless Paper v1.1**

## Wireless Paper V1.1



info

**The Wireless Paper V1.1 is listed as "Wireless Paper" in the firmware files and on the web flasher.**

This device may have issues charging a connected battery if utilizing a USB-C to USB-C cable. It's recommended to use a USB-A to USB-C cable.

- **MCU:**
  - ESP32-S3FN8 (Wi-Fi & Bluetooth)
- **LoRa Transceiver:**
  - Semtech SX1262
- **Frequency Options:**
  - 433 MHz
  - 470 - 510 MHz
  - 863 - 870 MHz
  - 902 - 928 MHz
- **Connectors:**
  - USB-C
  - Antenna:
    - U.FL/IPEX antenna connector for LoRa
    - Integrated 2.4 GHz PCB antenna

## Features

- Onboard 2.13-inch black and white E-Ink display screen
- User and Reset switches
- No GPS

## Resources

- Firmware file: `firmware-heltec-wireless-paper-X.X.X.xxxxxxx.bin`
- Purchase Links:
  - US
    - Rokland
  - International
    - Heltec
    - AliExpress

# Heltec WiFi LoRa 32: buttons

From https://meshtastic.org/docs/hardware/devices/heltec-automation/lora32/buttons/

## Functionality

Most device functions are menu items rather than dedicated button presses. Hold the user button to open the menu for the screen currently shown. Short presses move through the menu, and another hold activates an item.

The user button is silkscreened **PRG** on the LoRa32 V3 and **PROG** on the V4. Both are the same button.

- **Reset button (bottom):**
  - **Short press:** Reset the device.
- **User button (top, PRG or PROG):**
  - **Short press:** Move to the next screen. With a menu open, move to the next item.
  - **Hold about half a second:** Open the menu for the current screen. With a menu open, activate the highlighted item.
  - **Hold about four seconds, then release:** Shut the device down. A rising four-note tone signals that shutdown is arming. This works only more than 30 seconds after boot.

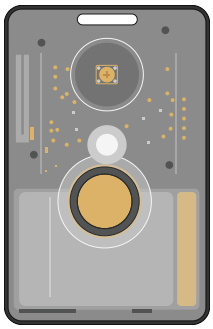
The user button is also the ESP32 BOOT strap. Holding it while powering on enters the serial bootloader instead of starting Meshtastic.

The V4 TFT firmware adds a second button. The OLED firmware does not.

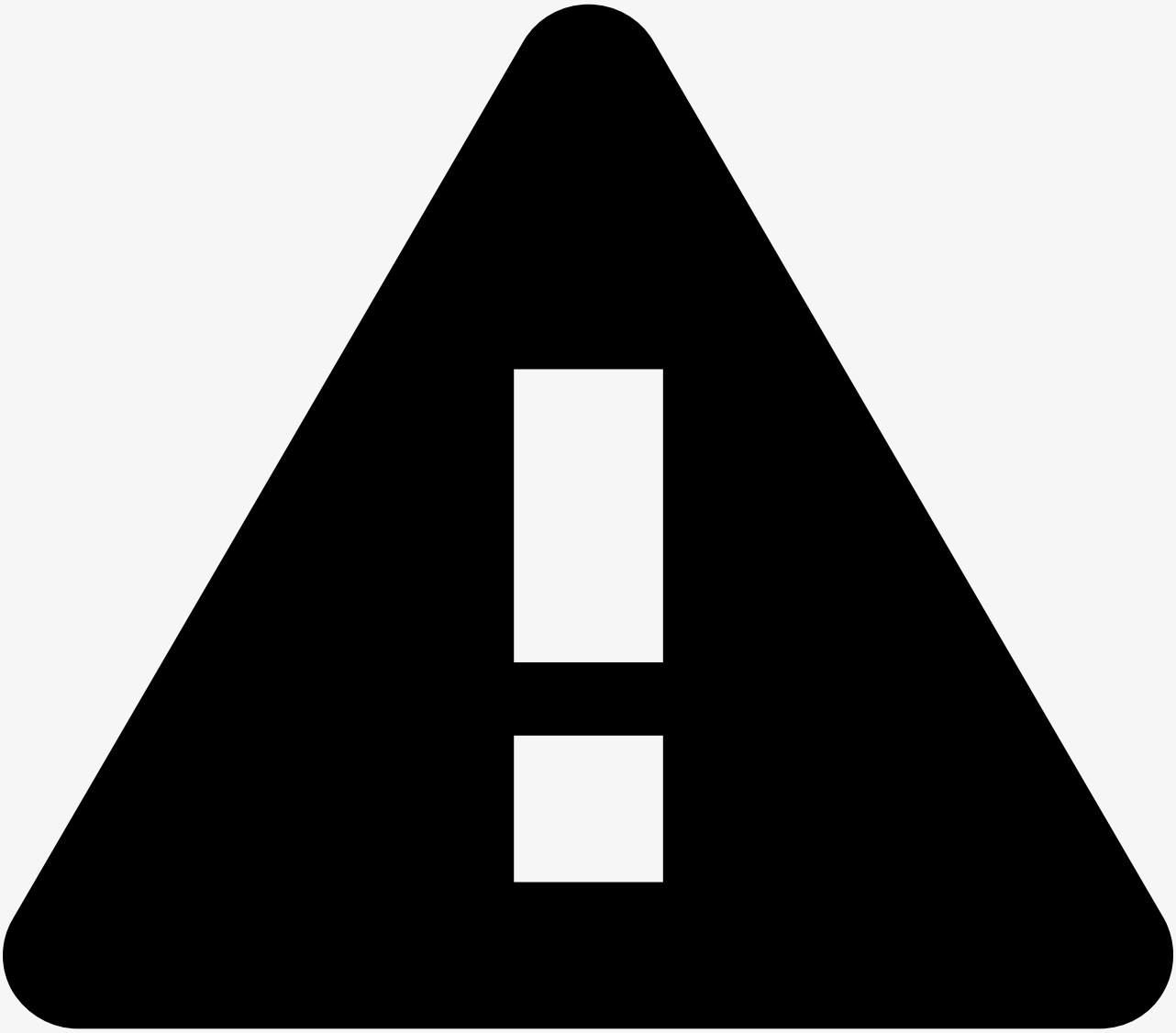
- **Second button (TFT firmware only):**
  - **Short press:** Move to the previous screen. With a menu open, move the highlight up.
  - **Hold about half a second:** Close the menu.

# SenseCAP Card Tracker T1000-E (the pocket radio)

From https://meshtastic.org/docs/hardware/devices/seeed-studio/sensecap/card-tracker/



SenseCAP T1000-E is a high-performance tracker designed for Meshtastic. As small as a credit card, effortlessly fitting in your pocket or attaching to your assets. It embeds Semtech's LR1110, Nordic's nRF52840, and Mediatek's AG3335 GPS module, providing Meshtastic users with a high-precision, low-power positioning and communication solution.



Note

Currently, LR1110 radios are unable to receive Meshtastic packets from the older SX127x radios, it requires a breaking change to fix this. Transmitting works and when hopping through an SX126x radio, you can still receive packets from SX127x radios.

## Specifications

- **MCU**
  - Nordic nRF52840 (supports Bluetooth)
- **LoRa Transceiver**
  - Semtech LR1110
- **Frequency options**

    - 865-928 MHz
- **Navigation Module**
    - Mediatek AG3335 GPS chip
- **Battery Capacity**
    - Rechargeable lithium battery, 700mAh
- **Charging**
    - USB magnetic charging cable

## Features

- **Button and Buzzer**: For user interaction and alerts.
- **Pogo Pins**: Four pins for USB, DFU, serial logging, and charging.
- **Rugged Build**: IP65 rated, waterproof and durable for various environments.

## Functionality

- **User/Program Button (face):**
    - **Long press:** Will signal the device to shutdown after 5 seconds.
    - **Double press:** Sends an adhoc ping of the device's position to the network.
    - **Triple press:** Enables/Disables the GPS Module on demand.
    - **Quadruple press:** Temporarily mute or unmute device.
    - **Hold while the device starts:** Enters bootloader mode for a firmware update. See Flashing.

## Flashing

The T1000-E has one button and no reset button, so the double-press that reaches bootloader mode on most nRF52 devices doesn't apply. This needs the Meshtastic nRF52 Bootloader (formerly OTAFIX) 2.5 or later.

1. Connect the device to your computer with a USB data cable.
2. Press and hold the button to power the device on, and keep holding it for 3 seconds.
3. The device mounts as a USB drive a few seconds later.
4. Copy the firmware `.uf2` file onto the drive, as described in Drag & Drop.

To leave bootloader mode without flashing, eject the drive.

On the bootloader Seeed ships, the button doesn't reach bootloader mode. Follow Seeed's T1000-E guide to enter it, or update the bootloader first. To skip the button entirely on any bootloader, connect the device over USB and run `meshtastic --enter-dfu`. Entering bootloader mode covers every method.

## Resources

- Firmware file: `firmware-tracker-t1000-e-X.X.X.xxxxxxx.uf2`
- Purchase Links:
    - International:
        - Seeed Studio Online Store

        - Seeed Studio Aliexpress Official Store
        - Seeed Studio Amazon Official Store
- User Guide

### Images


SenseCAP
SENSECAP


Magnetic Charging &
Firmware Upgrade
3-Axis Accelerometer
Button
Temperature Sensor
SENSECAP
Buzzer
Light Sensor & LED

# Questions

## Frequently asked questions

From https://meshtastic.org/docs/faq/

### Overview

Where can I get additional help, ask questions, or bond with the Meshtastic community?
How can I contribute to Meshtastic?

### Firmware

Can I update my node's firmware over the mesh?
Which version of firmware should I use?
How often should I update firmware?

### Android Client

What versions of Android does the Meshtastic Android App require?
What does the icon next to the message mean?
How can I clear the message history?
After a fresh firmware install, my node is not connecting via Bluetooth. What should I do?

### Apple Clients

What version of iOS/iPadOS/macOS does the Meshtastic App Require?
How do I get the Apple Meshtastic App?
After a fresh firmware install, my node is not connecting via Bluetooth. What should I do?
Do the Apple applications provide an indication if the message was acknowledged on the mesh?

### Channels[1]

How do I share my Meshtastic Channel with other people?

### Python CLI

How do I find out more about installing (and using) Meshtastic via command line?
How do I find out more about using python to interact?
What if I'm still having issues on Windows 10?

### Devices

How do I turn off an ESP32 T-Beam based device?
How do I turn on an ESP32 T-Beam based device?

Functionality of the T-Beam Buttons
Where do I purchase the device hardware?
I have my hardware. How do I install the firmware and any required drivers?
How do I update the firmware to the latest version?
My device has gone to sleep. Are received messages lost?
My device has gone to sleep and I can't send any messages.
How can I tell the device not to sleep?
I am in Europe and my device seems to stop transmitting after a while, what is going on?
Why does only one RAK Meshtastic Starter kit show up in my node list?

## Amateur Radio (ham)

Meshtastic can be used by both unlicensed people and licensed ham operators.

What is the benefit of using a ham license with Meshtastic?
How do I set my call sign?
Why aren’t licensed and unlicensed modes interoperable? Is this a planned feature?

## Mesh

Does Meshtastic use LoRaWAN?
Will Meshtastic work with (insert LoRa service)
Can I locate a device via triangulation?

## Modules

What are Modules?
What modules do we have available?
I'd like to write a module. How do I get started?

## Footnotes

1. The Meshtastic QR code (or Channel URL) allows users to quickly share channel and LoRa settings, making it easy to configure multiple nodes with matching settings for communication. Scanning a QR code applies all included channel settings and LoRa configuration settings, so be sure to review what these settings include before proceeding. Only scan QR codes from trusted sources.

   For more details, see:

   - Channel Config
   - LoRa Config

   ↩

# Licensing and trademark

From https://meshtastic.org/docs/legal/licensing-and-trademark/

Meshtastic® is a registered trademark of Meshtastic LLC. Meshtastic software components are released under various licenses, see GitHub for details. No warranty is provided - use at your own risk.

## Licensing

Entities with a trademark grant are hereby granted a two-year, non-exclusive and revocable license to use the Meshtastic® logo and trademark on your products or projects, in accordance with our trademark guidelines.

Please ensure the inclusion of the following statement in your support documentation:

> Meshtastic® is a registered trademark of Meshtastic LLC. Meshtastic software components are released under various licenses, see GitHub for details. No warranty is provided - use at your own risk.

Alternatively, if you prefer, you can use our community project logos, nicknamed "M-Powered" and "M-PWRD", which are specifically designed for projects that utilize Meshtastic technology. These logos offer a straightforward way to show compatibility with our platform without needing a trademark grant. For more information on these logos, including usage guidelines and design files, please refer to Community Project Logos: M-Powered & M-PWRD.

## Meshtastic Trademark Policy & Brand Guidelines

Meshtastic, and Meshtastic logo ("Meshtastic Logo"), either separately or in combination, are hereinafter referred to as "Meshtastic Trademarks" and are trademarks of the Meshtastic LLC. Except as provided in these guidelines, you may not use the Meshtastic Trademarks or any confusingly similar mark as a trademark for your product, or use the Meshtastic Trademarks in any other manner that might cause confusion in the marketplace, including but not limited to in advertising, on websites, or on software. In fact, the law obligates trademark owners to police their marks and prevent the use of confusingly similar names by third parties. If you have questions about this policy, please contact the Trademark Supervisor by enquiring at trademark@meshtastic.org.

## Usage That Does Not Require Written Permission

Below are the guidelines for use of the Meshtastic Trademarks where, as long as you are in compliance with the guidelines, no advance written permission is necessary. In all cases, use is permitted only provided that:

- the use is not disparaging to Meshtastic or software distributed by Meshtastic.
- the use does not imply sponsorship or endorsement by Meshtastic.

- proper trademark symbols are used in connection with the Meshtastic Trademarks and the trademark attribution statement must appear as explained in Proper Trademark Use.
- the Logo Usage Guidelines are strictly observed.

## Noncommercial and community web sites

In the past, community members have inquired whether it is permissible to show support for Meshtastic by:

- placing the Meshtastic Trademarks on a personal web site or blog to support open source software.
- making a page on a social networking web service to support open source software.
- linking to the Meshtastic website to provide information or show support for open source software.

The guidelines relating to such usage are set forth in this section.

It is permissible to use the Meshtastic Trademarks on websites to show your support for the open source software, provided that:

- where possible, the design logo hyperlinks to the Meshtastic website, meshtastic.org, or if that is not possible, the site includes a prominent link to the Meshtastic website at meshtastic.org.
- proper trademark symbols are used in connection with the Meshtastic Trademarks and the trademark attribution statement must appear as explained in Proper Trademark Use.
- the Logo Usage Guidelines are strictly observed.
- the site indicates clearly that it is not affiliated with or endorsed by Meshtastic; in addition, where possible: the site must include the text "This site is not affiliated with or endorsed by the Meshtastic project" prominently on any page that includes the Meshtastic Trademarks, and
- if the Meshtastic Trademarks appear in a page header or any area that is designed to be presented on more than one page, the notice must also be designed to be presented on all of those pages as well. (i.e., if the Meshtastic Trademarks appear in a site-wide header, the informational text must appear in that header or an identically site-wide footer.)
- the site does not use visual styling that could be confusing to viewers or visitors as to whether the site is hosted by or on behalf of Meshtastic.
- A copy of the use of the Meshtastic Trademarks is provided to the Trademark Supervisor within seven (7) days of its initial use, for example by sending a URL or other copy of such use.

## Business web sites

In the past, community members have inquired whether it is permissible to show support for Meshtastic by:

- displaying a link to the Meshtastic website using the Meshtastic Trademarks from a business web site.
- displaying the Meshtastic Trademarks as part of a business that utilizes Meshtastic software.

The guidelines relating to such usage are set forth in this section.

It is permissible to use the Meshtastic Trademarks on business web sites, provided that:

- the web site has non-Meshtastic primary branding.
- the design logo hyperlinks to the Meshtastic website, meshtastic.org.
- the use does not imply sponsorship or endorsement by Meshtastic.
- the use of the Meshtastic Trademarks does not imply an association with nor any form of endorsement by Meshtastic.
- proper trademark symbols are used in connection with the Meshtastic Trademarks and the trademark attribution statement must appear as explained in Proper Trademark Use.
- the Logo Usage Guidelines are strictly observed.
- the site does not use visual styling that could be confusing to viewers or visitors as to whether the site is hosted by or on behalf of Meshtastic.
- a copy of the use of the Meshtastic Trademarks is provided to the Trademark Supervisor within seven (7) days of its initial use, for example by sending a URL or other copy of such use.

## Promotional events

In the past, community members have inquired whether it is permissible to use the Meshtastic Trademarks to promote events. The guidelines relating to such usage are set forth in this section.

It is permissible to use the Meshtastic Trademarks in such promotional events, provided that:

- the use does not imply sponsorship or endorsement by Meshtastic.
- the use of the Meshtastic Trademarks does not imply an association with or endorsement of the event of the goods distributed at such event.
- proper trademark symbols are used in connection with the Meshtastic Trademarks and the trademark attribution statement must appear as explained in Proper Trademark Use.
- the Logo Usage Guidelines are strictly observed.
- A copy of the use of the Meshtastic Trademarks is provided to the Trademark Supervisor within seven (7) days of its initial use, for example by sending a URL or other copy of such use.

If you would like to make some non-software goods to give away or sell at the event and don't already have a license to do so, see Non-software goods.

## Publications

It is permissible to use the Meshtastic Trademarks in the title and content of a publication, provided that:

- the use is clearly in reference to Meshtastic.
- the use does not imply sponsorship or endorsement of the publication by Meshtastic.
- proper trademark symbols are used in connection with the Meshtastic Trademarks and the trademark attribution statement must appear as explained in Proper Trademark Use.
- the Logo Usage Guidelines are strictly observed.
- a copy of the use of the Meshtastic Trademarks is provided to the Trademark Supervisor within seven (7) days of its initial use, for example by sending a URL or other copy of such use.

### Community Project Logos: M-Powered & M-PWRD

We are excited to introduce two new logos for our community members: the "M-Powered" and "M-PWRD" logos. These logos are designed to be used by projects that are powered by Meshtastic technology, offering an easy way to indicate compatibility with our platform.

#### Usage Guidelines

- **Variants:**
  - "M-Powered" with the full word "powered."
  - "M-PWRD" with an abbreviated "PWRD" for a cleaner, concise design. This version is particularly recommended for 3D printed objects such as enclosures.
- **Freedom of Use:** Unlike the main Meshtastic logo, using the "M-Powered" or "M-PWRD" logos does not require a trademark grant. These logos are intended for broader use in the community to signify that a project is Meshtastic-compatible.
- **Appropriate Contexts:** In most scenarios, the "M-Powered" or "M-PWRD" logos will be more suitable than the official Meshtastic logo. They are ideal for project documentation, promotional materials, and product designs to show compatibility with Meshtastic.
- **Non-Endorsement:** It's important to note that using these logos does not imply endorsement or sponsorship by the Meshtastic Project. They are purely for indicating compatibility or association with Meshtastic technology.
- **Good Standing Requirement:** Permission to use the M-Powered logo is granted only to individuals or projects that are in good standing within the Meshtastic community. Users who have been banned or are otherwise restricted from community participation due to violations of the Code of Conduct are prohibited from using the M-Powered logo in any form.

We have provided design files for these logos in our Meshtastic Design repository.

## Usage that Require Prior Written Approval

### Social Media

In the past, community members have inquired whether it is permissible to use the Meshtastic Trademarks, including the term "Meshtastic" in a Social Media account. This is permitted provided that the phrasing doesn't indicate official sponsorship by Meshtastic. Examples:

- Meshtastic South Africa Users Group

- Meshtastic North America Community
- Cardiff Meshtastic Club
- Meshtastic UK Association

Examples not permitted:

- Meshtastic
- Meshtastic UK

Meshtastic also grants communities and user groups permission to use the Meshtastic logo, provided it is displayed in a secondary position to the group's primary branding. For example, a hiking club's newsletter might feature their own logo prominently at the top, with the Meshtastic logo placed smaller and below, indicating the technology's support for their activities.

### Domain names

Community members have asked whether it is permissible to use the Meshtastic trademarks, including the term “Meshtastic,” in an Internet domain name. This is not allowed at this time.

Instead, many community groups have found success using alternative domain names that do not directly incorporate the trademark. Examples include:

- https://bayme.sh/
- https://austinmesh.org
- https://meshhessen.de/

### Non-software goods

In the past, community members have inquired whether it is permissible to use Meshtastic Trademarks on non-software goods such as Embedded Systems, Enclosures, Hardware, T-shirts, stickers, and pens.

Community members may request from the Trademark Supervisor designated by the Admins of Meshtastic a license to use the Meshtastic Trademarks on non-software related goods or services, by enquiring at trademark@meshtastic.org. The Trademark Supervisor will be responsible for reviewing samples of the goods and services and managing the relationship.

- The adding our logo or other marks to a non-software good may be allowed, with written permission from the Meshtastic Trademark Supervisor, provided that the non-software good is a 100% original design or the requester has evidence of a grant of use from copyright holder of the original design.
- The adding our logo or other marks to a non-software good is not an endorsement or sponsorship of that good by the Meshtastic Project.

## Unapproved Use

The following uses of the Meshtastic Trademarks are not approved under any foreseeable circumstances.

- Violations of the Logo Usage Guidelines or Trademark Usage Guidelines.
- Any use outside these guidelines not by explicit written permission.
- Except as set forth herein, Meshtastic retains and reserves all rights to the Meshtastic Trademarks and their use, including the right to modify these guidelines.

These guidelines (except for trademark licenses executed for non software goods) may be amended from time to time at the discretion of Meshtastic and such changes will be effective ten (10) days after the changes are posted. Meshtastic may provide you with notice of such changes, but need not do so. You are responsible for checking this site for any changes.

In the past, community members have inquired whether it is permissible to use the Meshtastic Trademarks, including the term "Meshtastic" in software product names. You **may not** use any of Meshtastic Trademarks in either the primary or secondary branding of any third party product or service names. For example, you may not name a product either "BigCo Project Thing", "BigCo distribution of Project", or any similar kind of name where Project is any Meshtastic mark or project name.

## Proper Trademark Use

One of the purposes of Meshtastic is to encourage the use of Meshtastic software to enable the public to come to trust the use of Meshtastic Trademarks. To achieve this purpose it is important that Meshtastic can quickly be identified. Meshtastic has chosen the Meshtastic Logo to identify such software and is granting usage rights in the Meshtastic Logo (and the other Meshtastic Trademarks) as previously described in this document in order to assure widespread availability.

### Trademark Usage Guidelines

#### Guideline

#### Examples

When using the Meshtastic Trademarks you must provide the proper trademark symbols and a trademark attribution statement.

Acceptable: Use for the first instance of the Meshtastic Logo include the ® mark, and include the statement "The Meshtastic logo trademark is the trademark of Meshtastic LLC."

Unacceptable: Never using the ® mark for Meshtastic Logo, nor a trademark statement per the guidelines.

- Always distinguish trademarks from surrounding text with at least initial capital letters or in all capital letters.
- Never pluralize a trademark. Never use "a" or "the" to refer to an instance of the trademark. Always use a trademark as an adjective modifying a noun, or as a singular

noun.
- Never use a trademark as a possessive. Instead, the following noun should be used in possessive form or the sentence reworded so there is no possessive.
- Never translate a trademark into another language.
- Never alter a trademark in any way including through unapproved fonts or visual identifiers.
- Never use or register any trademarks that are confusingly similar to, or a play on, the Meshtastic or Meshtastic Logo.
- Never combine your company name with the Meshtastic name or use the Meshtastic name in a way that it could be perceived that Meshtastic and your company have an organizational link such as a joint venture.
- Never use the Meshtastic Trademarks in a disparaging manner or that violates any federal, state, or international law.

Unacceptable: Cartoon character micturating on the Meshtastic Logo, applying Meshtastic Logo outside permitted uses.

- Never use terminology that states or implies that Meshtastic assumes any responsibility for the performance of your products and services.
- The Meshtastic Logo must be hyperlinked to meshtastic.org, in contexts where such a hyperlink is technically feasible.

Acceptable: Hyperlinking the Meshtastic Logo where feasible.

Unacceptable: Not hyperlinking the Meshtastic Logo where feasible.

Except as prohibited by law, the person or entity who is using the Meshtastic Trademark under the terms of these Guidelines ("User") acknowledges that Meshtastic is the sole and exclusive owner of the Meshtastic Trademarks and agrees that it will do nothing inconsistent with such ownership either during the term of such use or afterwards. Specifically, User will take no action that will interfere with or diminish Meshtastic's right in the Meshtastic Trademarks. The User acknowledges that the Meshtastic Trademarks are valid under the applicable law and that User's utilization of the Meshtastic Trademarks will not create any right, title or interest in the Meshtastic Trademarks. The User agrees not to apply or assist any third party to register the Meshtastic Trademarks or a confusingly similar designation anywhere in the world. If any application for registration is or has been filed by or on behalf of User in any country and relates to any mark which, in the reasonable opinion of Meshtastic, is confusingly similar, deceptive or misleading with respect to, or dilutes or any way damages the Meshtastic Trademark, User shall, at Meshtastic's request, abandon all use of such mark, and any registration or application for registration thereof and shall reimburse Meshtastic for all costs and expenses of any opposition or related legal proceeding, including attorneys' fees, instigation by Meshtastic or its authorized representative. Upon the written statement that the User is not using the Meshtastic Trademarks in accordance with the Guidelines, within ten (10) days, User shall modify its use to comply with the Guidelines or cease using the Meshtastic Trademarks.

## Logo Usage Guidelines

The Meshtastic logo is a trademark of Meshtastic LLC. In order to protect and grow the Meshtastic brand, we have a distinguishable logo. When displaying the Meshtastic logo, please follow our standard Trademark Guidelines. There are other sizes and resolutions of the logo and some are suitable for print.

To join the discussion about Meshtastic Trademark policies, and participate in shaping future policy visit the Meshtastic Discussions page or Meshtastic Discord.

The licence of the pages above

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights.  Therefore, you have
certain responsibilities if you distribute copies of the software, or if
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received.  You must make sure that they, too, receive
or can get the source code.  And you must show them these terms so they
know their rights.

  Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

  For the developers' and authors' protection, the GPL clearly explains
that there is no warranty for this free software.  For both users' and
authors' sake, the GPL requires that modified versions be marked as
changed, so that their problems will not be attributed erroneously to
authors of previous versions.

  Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the manufacturer
can do so.  This is fundamentally incompatible with the aim of
protecting users' freedom to change the software.  The systematic
pattern of such abuse occurs in the area of products for individuals to
use, which is precisely where it is most unacceptable.  Therefore, we
have designed this version of the GPL to prohibit the practice for those
products.  If such problems arise substantially in other domains, we
stand ready to extend this provision to those domains in future versions
of the GPL, as needed to protect the freedom of users.

  Finally, every program is threatened constantly by software patents.
States should not allow patents to restrict development and use of
software on general-purpose computers, but in those that do, we wish to
avoid the special danger that patents applied to a free program could
make it effectively proprietary.  To prevent this, the GPL assures that
patents cannot be used to render the program non-free.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS
```

```
  0. Definitions.

  "This License" refers to version 3 of the GNU General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy.  The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
```

subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is

released under this License and any conditions added under section
7.  This requirement modifies the requirement in section 4 to
"keep intact all notices".

c) You must license the entire work, as a whole, under this
License to anyone who comes into possession of a copy.  This
License will therefore apply, along with any applicable section 7
additional terms, to the whole of the work, and all its parts,
regardless of how they are packaged.  This License gives no
permission to license the work in any other way, but it does not
invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display
Appropriate Legal Notices; however, if the Program has interactive
interfaces that do not display Appropriate Legal Notices, your
work need not make them do so.

A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

a) Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by the
Corresponding Source fixed on a durable physical medium
customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product
(including a physical distribution medium), accompanied by a
written offer, valid for at least three years and valid for as
long as you offer spare parts or customer support for that product
model, to give anyone who possesses the object code either (1) a
copy of the Corresponding Source for all the software in the
product that is covered by this License, on a durable physical
medium customarily used for software interchange, for a price no
more than your reasonable cost of physically performing this
conveying of source, or (2) access to copy the
Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the
written offer to provide the Corresponding Source.  This
alternative is allowed only occasionally and noncommercially, and
only if you received the object code with such an offer, in accord
with subsection 6b.

d) Convey the object code by offering access from a designated
place (gratis or for a charge), and offer equivalent access to the
Corresponding Source in the same way through the same place at no
further charge.  You need not require recipients to copy the
Corresponding Source along with the object code.  If the place to
copy the object code is a network server, the Corresponding Source
may be on a different server (operated by you or a third party)
that supports equivalent copying facilities, provided you maintain
clear directions next to the object code saying where to find the
Corresponding Source.  Regardless of what server hosts the
Corresponding Source, you remain obligated to ensure that it is
available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided
you inform other peers where the object code and Corresponding
Source of the work are being offered to the general public at no

charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License.  If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

  9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work

in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

  A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License.  You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License.  If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all.  For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

  13. Use with the GNU Affero General Public License.

  Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work.  The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time.  Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

  Each version is given a distinguishing version number.  If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation.  If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>

```
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<https://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<https://www.gnu.org/licenses/why-not-lgpl.html>.
```