Power System Wiring and Safety

Complete wiring diagrams, fusing, wire gauge selection, weatherproof connectors, and battery telemetry monitoring.

- Wiring a Solar Power System for LoRa Repeaters
- Monitoring Battery State via Meshtastic Telemetry

Wiring a Solar Power System for LoRa Repeaters

Proper wiring is the difference between a node that runs reliably for years and one that fails intermittently or becomes a fire hazard. (Battery life, not wiring, sets the maintenance interval — plan battery replacement per the chemistry's cycle life.) This page covers the complete wiring path from solar panel to LoRa load, including fusing strategy, wire gauge selection, connector types, weatherproofing, and cable management inside enclosures.

System Wiring Overview

A correctly wired solar system follows this signal path:

```
Solar Panel(s)
 │
 ├── [Fuse #1: Panel → Controller] ── MC4 cable to controller PV input
 │
Charge Controller (MPPT/PWM)
 │
 ├── [Fuse #2: Controller → Battery] ── to battery positive terminal
 │
Battery Pack (LiFePO4/LiPo + BMS)
 │
 └── [Fuse #3: Battery → Load] ── to load (LoRa node, 5V regulator, etc.)
```

Fuse each wire segment at its source so the fuse protects that segment. A typical solar node uses three: PV-to-controller, controller-to-battery, and battery-to-load. Do not rely on one fuse to protect dissimilar-gauge segments. Each fuse protects the wire segment between it and the next power source, limiting fault current to what the wire can safely carry. (The binding requirement in code and standards is overcurrent protection sized to protect each conductor near its source — the exact number of fuses depends on the topology; PWM vs MPPT and parallel PV strings change what is needed.)

Fusing Requirements

Segment	Fuse Type	Rating (for 10 W panel, 7 Ah battery)	Placement
Panel → Charge controller (PV in)	Blade fuse or ANL	Next standard size ≥ Isc × 1.56 (here ~10 A). Optional for a single panel — see note below.	Per the controller manual, typically near the panel junction box positive terminal
Charge controller → Battery	Blade fuse or ANL	Next standard size at or below the conductor's ampacity, and ≥ charge controller rated output × 1.25 (here ~15 A for a ~10 A controller)	Within 7 inches of battery positive terminal (ABYC E-11)
Battery → Load	Blade fuse or resettable PPTC	3 - 5 A (sized to wire gauge, not load)	Within 7 inches of battery positive terminal

Fuse ratings are sized to protect the *wire*, not the load. Size the wire for at least 125% of the continuous load current, then choose a fuse at or **below** the wire's ampacity (the next standard size that does not exceed the conductor's ampacity, per NEC 240.4 / 240.6) so the fuse trips before the wire can be overloaded. Do **not** set the fuse above the wire's ampacity. ANL/bolt-down fuses are commonly used above ~30 - 40 A; MAXI blade fuses cover up to ~80 A. For small LoRa systems (under 10 A total), a waterproof inline blade-fuse holder rated for outdoor use (~$2 as of 2026-06-08) is adequate.

About the PV-side fuse (Isc × 1.56): *Isc* is your panel's short-circuit current, printed on the panel label. The 1.56 multiplier comes from NEC 690.8 (1.25 × 1.25). This fuse exists mainly to protect against reverse current back-fed from *other parallel panel strings*; a single panel feeding one controller has no other source to push reverse current, so a series PV fuse is **optional** for a single-panel system. For parallel strings, size it to the next standard fuse at or above Isc × 1.56 and follow the controller manual for placement.

Wire Gauge Selection

AWG	Conductor Area (mm²)	Max Ampacity (60 °C insulation, bundled)	Typical LoRa System Use
AWG 22	0.33 mm²	3 A	Sensor wiring, signal lines
AWG 20	0.52 mm²	5 A	Load output for single ESP32 node
AWG 18	0.82 mm²	7 A	Load output for small system (5 A load)
AWG 16	1.31 mm²	10 A	Battery-to-controller runs under 3 m

AWG	Conductor Area (mm²)	Max Ampacity (60 °C insulation, bundled)	Typical LoRa System Use
AWG 14	2.08 mm²	15 A	Battery-to-controller runs 3 - 10 m; 10 W panel to controller
AWG 12	3.31 mm²	20 A	20 - 40 W panel runs; Pi gateway battery cables
AWG 10	5.26 mm²	30 A	40 - 100 W panel runs over 5 m
AWG 8	8.37 mm²	40 A	100 W+ systems; long battery cable runs

Always use **stranded copper wire** with UV-resistant and temperature-rated insulation (XLPE or THWN-2 for outdoor; silicone for inside enclosures near heat). Solid wire is not suitable for mobile or vibrating installations. Use tinned copper wire in marine environments.

For voltage drop calculation in long cable runs:

```
Voltage_drop (V) = 2 × I (A) × R_per_meter (Ω/m) × Length (m)

Target: keep drop to less than 3% of system voltage.
For 12 V system, 3% = 0.36 V maximum drop.

Example: 5 A load, 5 m one-way run, AWG 14 (0.0083 Ω/m):
 Drop = 2 × 5 × 0.0083 × 5 = 0.415 V (3.5% - marginal, upgrade to AWG 12)
```

Weatherproof Connectors

MC4 Connectors (Panel Wiring)

MC4 (Multi-Contact 4 mm) connectors are the industry standard for solar panel connections. Common MC4 connectors are rated IP67/IP68, UV-resistant, up to 1000 - 1500 V DC and 30 - 40 A depending on the specific part and cable gauge (current rating scales with conductor cross-section). Never use non-MC4 connectors on the panel-side wiring - the exposed conductors in DIY terminal connections will corrode and introduce resistance. Crimp MC4 connectors with the correct MC4 crimper (not pliers) to ensure proper contact retention. MC4 pairs from different manufacturers (e.g., Stäubli vs Amphenol) are nominally cross-compatible but may have reduced IP rating when mixed - use matched pairs.

Anderson Powerpole Connectors (Load Connections)

Anderson Powerpole connectors (PP15, PP30, PP45 contacts) are the amateur radio and telecom standard for DC power distribution. They are genderless, stackable, and all three contact sizes share the same housing. Wire range is per contact: PP15 ~16 - 20 AWG, PP30 ~12 - 16 AWG, PP45 ~10 - 14 AWG. Crimp the PP30 (30 A) contact onto 12 - 16 AWG wire with a Powerpole ratchet crimper (e.g., Powerwerx TRIcrimp) or similar. ARES (Amateur Radio Emergency Service) has standardized on red (+) and black (−) 30 A Powerpoles for all portable power connections.

Other Connectors

Connector	Rating	Use Case
XT60	~30 A continuous (60 A surge)	High-current battery connections in drone/RC-derived builds
JST PH 2.0 mm	2 A	LiPo cell to embedded board (standard on most Adafruit/SparkFun boards)
JST XH 2.54 mm	3 A	Sensor connections inside enclosure
Dean's Ultra T-plug	30 A	Legacy RC packs; avoid for new designs

Polarity Protection

Reverse polarity can instantly damage unprotected charge controllers, LoRa boards, and BMS units. Many commercial controllers and BMS units include reverse-polarity protection and survive a miswire, but do not rely on it. **Before first connection, verify polarity with a multimeter:** set it to DC volts, put the red probe on the wire you believe is positive and the black probe on the negative; a positive reading confirms your assignment, a negative reading means the leads are swapped. Then implement at least one of the following:

1. **Asymmetric connectors:** MC4 (panel), Powerpole (load), JST (board) are all polarised - they cannot be connected backwards if crimped correctly.
2. **Schottky diode on the input:** A 3 A / 40 V Schottky diode (e.g., 1N5822) in series with the positive line blocks reverse connection. It wastes ~0.3 V at light load, rising to ~0.475 - 0.5 V near its 3 A rating, and continuously dissipates that voltage drop, so it is best for low-current inputs.

3. **P-channel MOSFET reverse protection:** A P-channel MOSFET (e.g., AO3401, IRF9540) provides near-zero-drop reverse polarity protection. Standard in commercial MPPT charge controller input stages.

Cable Routing and Strain Relief in Enclosures

Inside IP65/IP67 enclosures (Polycase WQ series, Bud Industries NBF, PolyBox), cables enter through compression cable glands. Rules:

- Use **double-sealed cable glands** (IP68 rated) for any cable entering an outdoor enclosure - single-seal glands allow moisture wicking along the cable jacket.
- All cable entries should be **on the side or bottom** of the enclosure, never on the top, to prevent water pooling at the seal.
- Leave a **drip loop** on the outside of each cable entry - a short downward curve below the gland that water follows away from the entry point before the cable turns upward.
- Inside the enclosure, route cables along the walls and use **cable ties on standoffs**, not across the PCB or battery. Keep power cables away from antenna cables to prevent RF interference.
- Add a **silica gel desiccant pack** (~1 - 3 g per liter of enclosure volume, so roughly 1 - 2 g per 0.5 L) and include a reusable humidity-indicator card; regenerate or replace the desiccant when the indicator shows above ~40 - 50% RH rather than on a fixed annual schedule.
- Secure the battery with **hook-and-loop strap or foam padding** to prevent it from shifting and chafing cables during thermal expansion/contraction.

Monitoring Battery State via Meshtastic Telemetry

Meshtastic and MeshCore both include power telemetry features that allow a node to report its battery voltage and charge level over the mesh network. This page covers enabling these features, configuring voltage ADC pins for different hardware, interpreting voltage as state-of-charge for LiFePO4 batteries, setting low-battery alerts, and visualising data in Grafana. (Firmware command syntax and pin details are as of 2026-06-08; verify against your installed firmware version, as Meshtastic config keys change across 2.x releases.)

Enabling Power Telemetry in Meshtastic

In Meshtastic firmware (2.x), power metrics are part of the **Telemetry Module**. To enable battery reporting:

Via Meshtastic Python CLI

```
# Install CLI: pip install meshtastic
# Set the device-metrics broadcast interval (battery level, voltage, uptime,
# and any attached power sensor). This specifically controls the DEVICE/power
# telemetry cadence, not environment-sensor metrics.
meshtastic --set telemetry.device_update_interval 300
# Sets reporting interval to 300 seconds (5 minutes)

# Verify telemetry module settings
meshtastic --get telemetry
```

Note: per the official Meshtastic telemetry docs, `device_update_interval` configures the interval (in seconds) used to send device/power metrics over the mesh. CLI key names have changed across firmware versions — if the command above is rejected, check the current key names in the Meshtastic telemetry module docs for your firmware version.

Via Meshtastic Web App or Mobile App

1. Open the Meshtastic app and connect to your node.
2. Navigate to **Config → Module Config → Telemetry**.
3. Enable **Device Metrics**.
4. Set the update interval (300 - 3600 seconds; use longer intervals for battery-powered nodes to reduce TX duty cycle).
5. Save and reboot the node.

Once enabled, the node broadcasts a `meshtastic.Telemetry` protobuf packet on the default channel at the configured interval. The packet includes:

- `battery_level`: Integer 0 - 100 (firmware-estimated SoC percentage)
- `voltage`: Float in volts (actual measured ADC voltage)
- `channel_utilization`: Float (% airtime used)
- `air_util_tx`: Float (TX airtime)

Voltage ADC Pin Configuration on Different Boards

Not all Meshtastic hardware platforms use the same pin or divider ratio for battery voltage measurement. The firmware auto-detects the board type from compile-time defines, but custom builds or off-label hardware may need manual configuration. Verify the exact pin and divider against your board's schematic before relying on these values.

Board	ADC Pin (GPIO)	Voltage Divider Ratio	Max Measurable Voltage	Notes
TTGO T-Beam v0.7	GPIO35	1:2 (100 kΩ / 100 kΩ)	~8.4 V	Measures raw single-18650 LiPo voltage (divider documented in community schematics)
TTGO T-Beam v1.1 (AXP192)	AXP192 PMIC register	Internal PMIC ADC	Reported via I²C	Reads VBAT register; very accurate
TTGO LoRa32 v2.1	GPIO35	1:2	~8.4 V	Same GPIO35 divider scheme as T-Beam v0.7 (confirm on the v2.1 schematic)

Board	ADC Pin (GPIO)	Voltage Divider Ratio	Max Measurable Voltage	Notes
Heltec WiFi LoRa 32 v3	GPIO1 (VBAT_Read)	~4.9:1 divider (390 kΩ / 100 kΩ)	Single-cell LiPo (~3.0–4.2 V)	You must drive **ADC_Ctrl (GPIO37) low first** to enable the divider, then read battery voltage on GPIO1. It is **not** a 1:1 / no-divider input. Reads a single LiPo cell.
RAK WisBlock RAK4631	P0.04 (AIN2)	1:2 via RAK5005-O base board	~6 V	Battery sense is on P0.04 / AIN2 (not P0.05/AIN3); reads via nRF52840 SAADC. Verify the AIN index against the WisBlock schematic.
Wispr / Custom ESP32	User-defined GPIO	User-defined	User-defined	Set in platformio.ini or via power.adc_multiplier_override config key

If the reported voltage seems incorrect, verify with a multimeter at the battery terminals. Then check `power.adc_multiplier_override` (confirm this key exists in your firmware version):

```
meshtastic --set power.adc_multiplier_override 2.0
# Multiplies the raw ADC reading by 2.0 (use for 1:2 divider boards)
```

Interpreting Voltage as State-of-Charge for LiFePO4

Meshtastic's built-in SoC estimation uses LiPo voltage thresholds (3.0 - 4.2 V per cell). For LiFePO4 packs, these thresholds are incorrect - LiFePO4 cells operate in the 2.5 - 3.65 V range. The firmware will report incorrect percentages unless you compensate.

Important: A LiFePO4 cell's voltage during/just after charging is much higher than its *rested* open-circuit voltage. The 3.60–3.65 V "100%" row below is the charge/absorb voltage; a fully charged cell that has rested settles to roughly **3.35–3.45 V**. Read SoC from rested voltage, not from voltage measured under charge or load.

LiFePO4 Single-Cell (3.2 V nominal) Voltage → SoC Table

Voltage (V)	Approximate SoC (%)	Interpretation
3.60 - 3.65 (under charge)	100%	Charge/absorb voltage; rested 100% OCV is ~3.35–3.45 V
3.40 - 3.45	90%	High charge, near rested-full plateau
3.30 - 3.35	70 - 80%	Mid-range - most of usable capacity here
3.27 - 3.30	50%	Flat region - voltage barely distinguishable from 70%
3.22 - 3.25	30%	Still flat; lower usable threshold approaching
3.18 - 3.22	20%	Low battery - alert threshold
3.10 - 3.18	10%	Critical - immediate recharge needed
< 3.10	<5%	BMS will soon disconnect; node will shut down

4S LiFePO4 Pack (12.8 V nominal) Voltage → SoC Table

Pack Voltage (V)	SoC (%)
14.2 - 14.6 (under charge)	100% (end of charge; rested-full ~13.4–13.8 V)
13.6 - 13.8	90%
13.2 - 13.4	70 - 80%
13.0 - 13.2	50%
12.8 - 13.0	30%
12.4 - 12.8	20%
12.0 - 12.4	10%
< 11.8	<5% (BMS cutoff imminent)

Setting Low-Battery Alerts in Meshtastic

Meshtastic does not natively send alert messages when battery drops below a threshold, but there are two approaches to implement this:

Approach 1 - Node-Red / MQTT Alert Pipeline

1. Configure Meshtastic MQTT uplink: `meshtastic --set mqtt.enabled true --set mqtt.address YOUR_BROKER_IP`
2. In Node-Red, subscribe to `msh/US/+/json/LongFast/#` (adjust channel name as needed).
3. When Meshtastic JSON-encoded MQTT output is enabled, battery voltage arrives as a **flat field** inside the telemetry payload, e.g. `msg.payload.payload.voltage` (the published JSON looks like `"payload": {"air_util_tx":0, "battery_level":0, "channel_utilization":0, "voltage":0}`). It is **not** nested under `decoded.telemetry.deviceMetrics` — that dotted path is the protobuf-decoded object shape used by the Python API, not the MQTT JSON shape. Filtering on the wrong layer returns `undefined` and the alert silently never fires. Check a real sample against the Meshtastic MQTT integration docs and filter on the field below your LVD threshold.
4. Route low-voltage events to an alert node (email, PushOver, Telegram bot).

Approach 2 - Meshtastic Python Script (Autonomous Node)

```
import meshtastic
import meshtastic.serial_interface
from meshtastic.mesh_pb2 import MeshPacket

iface = meshtastic.serial_interface.SerialInterface()
LOW_VOLTAGE_THRESHOLD = 3.18 # V per cell for LiFePO4 (20% SoC)

def on_receive(packet, interface):
 if "decoded" in packet and "telemetry" in packet["decoded"]:
 m = packet["decoded"]["telemetry"].get("deviceMetrics", {})
 voltage = m.get("voltage", 0)
 node_id = packet["fromId"]
```

```
 if voltage > 0 and voltage < LOW_VOLTAGE_THRESHOLD:
 print(f"LOW BATTERY: Node {node_id} at {voltage:.2f} V")
 # Send alert message on mesh
 iface.sendText(f"⚠ Low battery: {node_id} {voltage:.2f}V", wantAck=False)

iface.localNode.setOwner("MonitorNode")
iface.addReceiveObserver(on_receive)
input("Press Enter to exit\n")
iface.close()
```

MeshCore Telemetry Equivalent

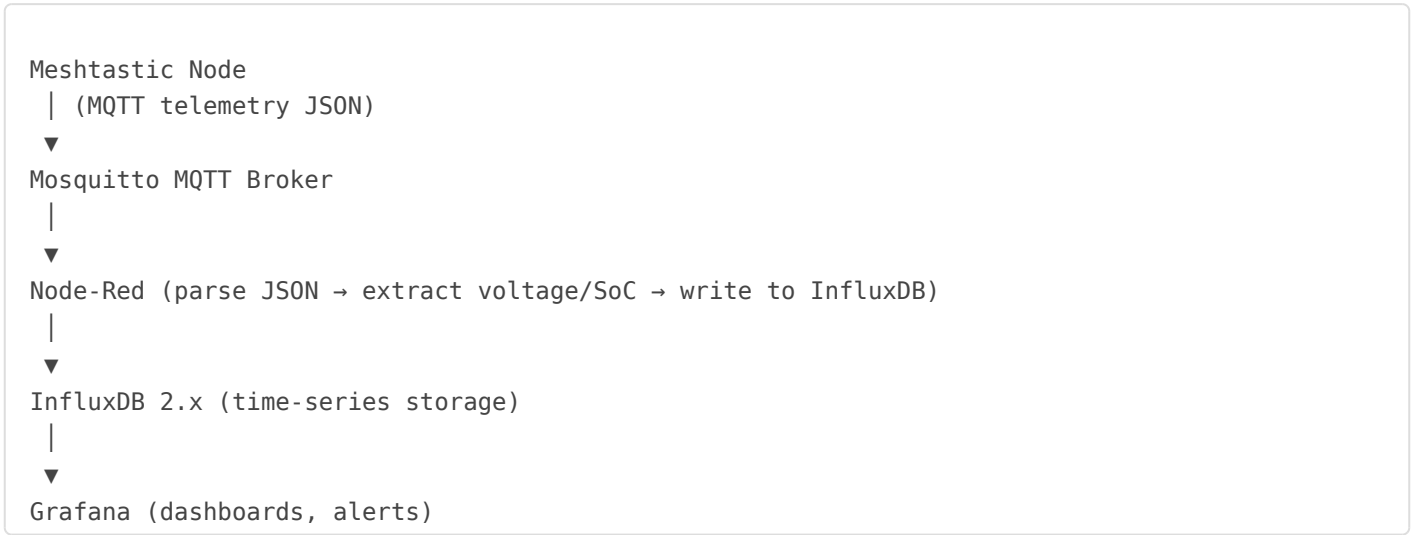
MeshCore also reports battery telemetry, but it does **not** use a "node YAML configuration file." MeshCore repeater and room-server firmware is configured over USB in the **web config tool**, or remotely over LoRa via the **companion mobile app** using the Remote Management feature, and via the **MeshCore CLI commands**. There is no YAML config mechanism in MeshCore firmware.

Use the MeshCore Repeater & Room Server CLI Reference to set the telemetry/advert interval and review available battery-reporting commands for your firmware build.

When telemetry is bridged to MQTT, Grafana can consume it via the Grafana MQTT data source plugin or via InfluxDB (Node-Red → InfluxDB → Grafana).

Graphing Battery Data in Grafana

Architecture



InfluxDB Line Protocol (Node-Red write node)

```
measurement: node_battery
tags: node_id, node_name, location
fields: voltage (float), battery_pct (int), soc_lifepo4 (float)
timestamp: nanosecond UNIX timestamp from packet
```

Grafana Panel Configuration

- **Battery voltage time series:** Use a Time Series panel with threshold bands - green above 3.30 V, yellow 3.18 - 3.30 V, red below 3.18 V (per cell) or scale for your pack voltage.
- **Multi-node SoC gauge:** Use a Gauge panel per node with min=0, max=100, thresholds at 20% (red) and 40% (yellow).
- **Grafana Alerting:** Set an alert rule on `avg(voltage) < 3.18` for any node, with a 15-minute evaluation window (to avoid false alarms from momentary load spikes). Route to PagerDuty, Slack, or email.

A reference Grafana dashboard for Meshtastic power monitoring can be built from the panel configuration above. (A previous version of this page referenced a specific dashboard JSON file in the Mesh America GitHub repository; that path is not yet verified to exist, so treat it as planned rather than published — build your dashboard from the configuration described here until a confirmed repository link is available.)