# OpenEMS

- Getting Started
- DDSU666 (Direct chint meter ) + mbpoll Command Reference
- Running OpenEMS Edge on a Raspberry PI
- DTSU666 3-Phase + mbpoll Command Reference

# Getting Started

# ZeroTier Remote Access Guide

This guide explains how to connect to the NFE Raspberry Pi from anywhere in the world using ZeroTier. It covers both SSH access and troubleshooting common issues.

---

# Table of Contents



---

# Prerequisites

Before starting, you need:

- A Mac or Windows laptop
- A ZeroTier account (free at https://my.zerotier.com)
- ZeroTier One installed on your computer
- Network authorization from the network administrator

---

# Network Information

**Network ID:** `2873fd00f2d70904` **Network Name:** `my-first-network` **Raspberry Pi ZeroTier IP:** `10.135.127.86` **Raspberry Pi Username:** `nfetestpi2`

---

# Installing ZeroTier on Your Computer

## Mac:

1. Download ZeroTier One: https://www.zerotier.com/download/
2. Install it normally
3. After installation, the ZeroTier icon will appear in the top-right menu bar

## Windows:

1. Download ZeroTier from https://www.zerotier.com/download/
2. Install and launch it
3. ZeroTier icon will appear in the system tray

---

# Joining the Network

## Method 1: Using the ZeroTier Menu (Mac - Recommended)

1. Click the ZeroTier icon in your menu bar
2. You'll see "My Address:" with your device ID
3. Click on the network ID `2873fd00f2d70904` if it's already listed
4. Or select "Join New Network..." and enter: `2873fd00f2d70904`
5. The status will show "REQUESTING_CONFIGURATION"

## Method 2: Using Command Line

Mac/Linux:

```
sudo zerotier-cli join 2873fd00f2d70904
```

Windows (run as Administrator):

```
zerotier-cli join 2873fd00f2d70904
```

## Authorization

After joining, you need to be authorized:

1. Contact the network administrator
2. Provide them with your device's MAC address or Device ID
3. They will authorize your device at https://my.zerotier.com
4. Once authorized, your device will receive an IP address like `10.135.127.xxx`

# Verify Connection

Mac/Linux:

```
sudo zerotier-cli listnetworks
```

You should see:

```
200 listnetworks 2873fd00f2d70904 my-first-network ... OK PRIVATE ... 10.135.127.xxx/24
```

The status should show **OK** and you should have an IP address assigned.

---

# SSH Access to Raspberry Pi

Once connected to the ZeroTier network, you can SSH to the Raspberry Pi:

```
ssh nfetestpi2@10.135.127.86
```

Enter the password when prompted.

**Note:** If you get a password prompt but it keeps failing, try using the `-v` flag for verbose output:

```
ssh -v nfetestpi2@10.135.127.86
```

# Optional: Set Up SSH Keys (Recommended)

To avoid entering passwords every time:

1. Generate SSH key on your computer (if you don't have one):

```
ssh-keygen -t ed25519 -C "your-email@example.com"
```

2. Copy your public key:

```
cat ~/.ssh/id_ed25519.pub
```

3. Add it to the Pi's authorized keys (via SSH or Raspberry Pi Connect):

```
mkdir -p ~/.ssh
nano ~/.ssh/authorized_keys
# Paste your public key, save and exit

# Set correct permissions
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

4. Now you can SSH without a password:

```
ssh nfetestpi2@10.135.127.86
```

---

# Troubleshooting

## Issue: ZeroTier shows "REQUESTING_CONFIGURATION"

**Cause:** Your device hasn't been authorized on the network yet.

**Solution:**

1. Go to https://my.zerotier.com
2. Log in and navigate to network `2873fd00f2d70904`
3. Click "Member Devices" tab
4. Find your device and check the "Auth" checkbox

## Issue: ZeroTier shows "OFFLINE"

**Cause:** ZeroTier service isn't running properly.

**Solution for Mac:**

```
# Restart ZeroTier service
sudo launchctl unload /Library/LaunchDaemons/com.zerotier.one.plist
sudo launchctl load /Library/LaunchDaemons/com.zerotier.one.plist
```

```
# Verify it's online
sudo zerotier-cli info
```

You should see:

```
200 info <device-id> 1.16.0 ONLINE
```

**Solution for Windows:**

- Restart the ZeroTier service from Services (services.msc)
- Or restart the ZeroTier One application

# Issue: Can't ping or SSH to Raspberry Pi

**Symptoms:**

```
ping 10.135.127.86
# Shows: "No route to host" or "Request timeout"
```

**Solutions:**

1. **Check if ZeroTier is running on both devices:**

```
sudo zerotier-cli info
# Should show: ONLINE
```

2. **Verify both devices are on the same network:**

```
sudo zerotier-cli listnetworks
# Both should show network 2873fd00f2d70904 with status OK
```

3. **Try changing WiFi networks:** Sometimes the initial WiFi network blocks ZeroTier's peer-to-peer connections. Try connecting to a different WiFi network or mobile hotspot.
4. **Check for RELAY connection:**

```
sudo zerotier-cli peers
```

If the Pi shows as "RELAY" instead of "DIRECT", there's a NAT traversal issue. Try:

- Restarting ZeroTier on both devices
- Leaving and rejoining the network
- Changing WiFi networks

5. **Restart ZeroTier on both devices:**

Mac:

```
sudo launchctl unload /Library/LaunchDaemons/com.zerotier.one.plist
sudo launchctl load /Library/LaunchDaemons/com.zerotier.one.plist
```

Raspberry Pi (via Raspberry Pi Connect):

```
sudo systemctl restart zerotier-one
```

6. **Leave old networks:** If you have multiple networks joined, leave unused ones:

```
# List networks
sudo zerotier-cli listnetworks

# Leave old network
sudo zerotier-cli leave <old-network-id>
```

# Issue: SSH password keeps failing

**Solutions:**

1. Try verbose SSH to see what's happening:

```
ssh -v nfetestpi2@10.135.127.86
```

2. Make sure you're using the correct username (`nfetestpi2`, not `pi`)
3. Set up SSH keys instead (see SSH Keys section above)

---

# Setting Up ZeroTier on a New Raspberry Pi

If you need to set up ZeroTier on a new Raspberry Pi, follow these steps:

# Prerequisites

- Raspberry Pi with Raspberry Pi OS installed
- Internet connection

- Access to the Pi (via Raspberry Pi Connect, monitor/keyboard, or local SSH)

# Installation Steps

1. **Install ZeroTier on the Raspberry Pi:**

```
curl -s https://install.zerotier.com | sudo bash
```

2. **Join the network:**

```
sudo zerotier-cli join 2873fd00f2d70904
```

3. **Verify the Pi joined:**

```
sudo zerotier-cli listnetworks
```

You'll see status as "ACCESS_DENIED" initially.

4. **Authorize the Pi:**

- Go to https://my.zerotier.com
- Log in and navigate to network `2873fd00f2d70904`
- Click "Member Devices" tab
- Find the new Pi device (you can identify it by the MAC address or hostname)
- Check the "Auth" checkbox
- Note the "Managed IP" assigned to the Pi (e.g., `10.135.127.xxx`)

5. **Verify connection:**

```
sudo zerotier-cli listnetworks
```

Should now show:

```
200 listnetworks 2873fd00f2d70904 my-first-network ... OK PRIVATE ztxxxxxx 10.135.127.xxx/24
```

6. **Enable SSH (if not already enabled):**

```
sudo systemctl enable ssh
sudo systemctl start ssh
```

7. **Make ZeroTier start on boot:**

```
sudo systemctl enable zerotier-one
```

8. **Test connection from another device:**

```
# From your computer (already on ZeroTier network)
ping <new-pi-ip>
ssh <username>@<new-pi-ip>
```

9. **Update this documentation** with the new Pi's IP address!

---

# Advanced: VNC Access (Remote Desktop)

If you need graphical access to the Raspberry Pi:

1. **Enable VNC on the Pi:**

```
sudo raspi-config
# Navigate to: Interface Options → VNC → Enable
```

2. **Install VNC Viewer on your computer:**
   https://www.realvnc.com/en/connect/download/viewer/
3. **Connect using the ZeroTier IP:**

- Open VNC Viewer
- Enter: `10.135.127.86`
- Enter Pi username and password

---

# Quick Reference

## Useful Commands

```
# Check ZeroTier status
sudo zerotier-cli info

# List joined networks
sudo zerotier-cli listnetworks

# Join a network
```

```
sudo zerotier-cli join <network-id>

# Leave a network
sudo zerotier-cli leave <network-id>

# Check peer connections
sudo zerotier-cli peers

# SSH to Raspberry Pi
ssh nfetestpi2@10.135.127.86
```

# Network Details

- **Network ID:** `2873fd00f2d70904`
- **Network Name:** `my-first-network`
- **Pi IP:** `10.135.127.86`
- **Pi Username:** `nfetestpi2`

---

*Last updated: 2026-03-28*

# DDSU666 (Direct chint meter ) + mbpoll Command Reference

**Version**: 1.0
**Prepared For**: Field & Deployment Teams
**Platform**: Raspberry Pi / Linux
**Tool**: mbpoll

## 1. Purpose

This document explains how to communicate with the CHINT DDSU666 Direct Smart Meter using Modbus RTU and the mbpoll tool.

It covers:
• Reading electrical parameters
• Setting meter addresses
• Verifying communication
• Preparing for OpenEMS integration

## 2. Hardware & Software Requirements

**Hardware**
• DDSU666 (Direct Version)
• RS485 → USB Converter
• Raspberry Pi / Linux PC
• Correct RS485 wiring (A(converter)↔24(meter com-port terminal), B(converter)↔25(meter com-port terminal), GND recommended)

**Software**
Install mbpoll:

> “
> sudo apt update
> sudo apt install mbpoll

Check serial port:

```
ls /dev/ttyUSB*
```

Example output:

```
/dev/ttyUSB0
```

## 3. Communication Parameters (Confirmed)

| Parameter | Value |
|---|---|
| Protocol | Modbus RTU |
| Baud Rate | 9600 |
| Data Bits | 8 |
| Parity | None |
| Stop Bits | 2 |
| Format | 8N2 |
| Float Order | Big Endian |
| Addressing | 0-Based |

These parameters must always be used.

## 4. Standard mbpoll Format

All commands follow this format:

```
mbpoll -m rtu -b 9600 -P none -s 2 -t 4:float -B -0 -r <register> -c <count>
/dev/ttyUSB0 -a <id> -1
```

Where:

- <register> = Modbus register
- <count> = Number of values
- <id> = Meter address (NO.)

## 5. Electrical Parameters

**Summary:**

```
```

> Voltage: 0x2000
> Current: 0x2002
> Active Power: 0x2004
> Power Factor: 0x200A
> Frequency: 0x200E
> Energy: 0x4000

## 5.1 Voltage (V) — Register 0x2000

> “
> **Meter ID = 1**
> mbpoll -m rtu -b 9600 -P none -s 2 -t 4:float -B -0 \
> -r 0x2000 -c 1 /dev/ttyUSB0 -a 1 -1
> **Meter ID = 2**
> mbpoll -m rtu -b 9600 -P none -s 2 -t 4:float -B -0 \
> -r 0x2000 -c 1 /dev/ttyUSB0 -a 2 -1

## 5.2 Current (A) — Register 0x2002

> “
> **Meter ID = 1**
> mbpoll -m rtu -b 9600 -P none -s 2 -t 4:float -B -0 \
> -r 0x2002 -c 1 /dev/ttyUSB0 -a 1 -1
> **Meter ID = 2**
> mbpoll -m rtu -b 9600 -P none -s 2 -t 4:float -B -0 \
> -r 0x2002 -c 1 /dev/ttyUSB0 -a 2 -1

## 5.3 Active Power — Register 0x2004

> “
> **Meter ID = 1**
> mbpoll -m rtu -b 9600 -P none -s 2 -t 4:float -B -0 \
> -r 0x2004 -c 1 /dev/ttyUSB0 -a 1 -1
>
> **Meter ID = 2**
> mbpoll -m rtu -b 9600 -P none -s 2 -t 4:float -B -0 \
> -r 0x2004 -c 1 /dev/ttyUSB0 -a 2 -1

### 5.4 Power Factor — Register 0x200A

> **Meter ID = 1**
> mbpoll -m rtu -b 9600 -P none -s 2 -t 4:float -B -0 \
> -r 0x200A -c 1 /dev/ttyUSB0 -a 1 -1
> **Meter ID = 2**
> mbpoll -m rtu -b 9600 -P none -s 2 -t 4:float -B -0 \
> -r 0x200A -c 1 /dev/ttyUSB0 -a 2 -1

### 5.5 Frequency (Hz) — Register 0x200E

> **Meter ID = 1**
> mbpoll -m rtu -b 9600 -P none -s 2 -t 4:float -B -0 \
> -r 0x200E -c 1 /dev/ttyUSB0 -a 1 -1
> **Meter ID = 2**
> mbpoll -m rtu -b 9600 -P none -s 2 -t 4:float -B -0 \
> -r 0x200E -c 1 /dev/ttyUSB0 -a 2 -1

### 5.6 Energy (kWh) — Register 0x4000

> **Meter ID = 1**
> mbpoll -m rtu -b 9600 -P none -s 2 -t 4:float -B -0 \
> -r 0x4000 -c 1 /dev/ttyUSB0 -a 1 -1
> **Meter ID = 2**
> mbpoll -m rtu -b 9600 -P none -s 2 -t 4:float -B -0 \
> -r 0x4000 -c 1 /dev/ttyUSB0 -a 2 -1

## 6. Reading Multiple Values

Use **-c** option to read multiple registers.

**Example ( Voltage + Current Together)**

>

```
mbpoll -m rtu -b 9600 -P none -s 2 -t 4:float -B -0 \
-r 0x2000 -c 2 /dev/ttyUSB0 -a 1 -1
```

## 7. Meter Address

> **Register: 0x0006**
>
> ```
> mbpoll -m rtu -b 9600 -P none -s 2 -t 4:int16 -0 \
> -r 0x0006 -c 1 /dev/ttyUSB0 -a 2 -1
> ```
>
> **Example Output:**
> [6]: 2

## 8. Changing Address

- Only one meter connected.
- Power cycle after change.

> **Change Address (2 → 1)**
> ```
> mbpoll -m rtu -b 9600 -P none -s 2 -t 4:int16 -0 \
> -r 0x0006 /dev/ttyUSB0 -a 2 -W -- 1
> ```
> **Power Cycle**
> Turn OFF → ON the meter.
> **Verify**
> ```
> mbpoll -m rtu -b 9600 -P none -s 2 -t 4:int16 -0 \
> -r 0x0006 -c 1 /dev/ttyUSB0 -a 1 -1
> ```

## 9. Health Check

Voltage should be 220–240V.

> ```
> mbpoll -m rtu -b 9600 -P none -s 2 -t 4:float -B -0 \
> -r 0x2000 -c 1 /dev/ttyUSB0 -a 1 -1
> ```

## 10. Common Issues

**Timeout Errors**

- Duplicate IDs
- Wrong wiring
- Wrong stop bits
- Multiple meters during setup

**Incorrect Float Values**

Use -B option.

**Current / Power = 0**

Normal when:

- No load
- Bench testing

## 11. Deployment Workflow

For large installations:

- Connect one meter
- Convert to Modbus
- Set address
- Test voltage
- Install
- Repeat
- Never deploy duplicate addresses.

> “
> Connect → Configure → Test → Install

## 12. System Status

✔ Modbus Enabled
✔ 9600 8N2 Confirmed
✔ Big-Endian Floats
✔ Multi-meter Bus Working
✔ Ready for OpenEMS

# Running OpenEMS Edge on a Raspberry PI

This guide explains how to:

1. Install and run **OpenEMS Edge on a Raspberry Pi**
2. Configure WebSocket + simulation
3. Run **OpenEMS UI on a macOS machine (not on the Pi)**
4. Connect the UI to the Edge over the internet

The structure is intentionally split into two clear parts:

- **PART A — Raspberry Pi (Edge)**
- **PART B — macOS Machine (OpenEMS UI)**

---

# PART A — Raspberry Pi (Edge)

## 1. Raspberry Pi OS Setup (Headless)

Use Raspberry Pi Imager:

- Device: Raspberry Pi 4
- OS: Raspberry Pi OS Lite (64-bit)
- Configure:
  - Hostname

    - Username & password
    - Enable SSH
    - Configure Wi-Fi
    - Enable Raspberry Pi Connect

Boot the Pi and connect via:

- SSH
- OR Raspberry Pi Connect

---

# 2. Install Java 21 (Temurin ARM64)

On the Pi:

```
sudo apt update
sudo apt install -y wget apt-transport-https gpg

wget -qO - https://packages.adoptium.net/artifactory/api/gpg/key/public \
 | gpg --dearmor | sudo tee /etc/apt/trusted.gpg.d/adoptium.gpg > /dev/null

echo "deb https://packages.adoptium.net/artifactory/deb \
$(awk -F= '/^VERSION_CODENAME/{print $2}' /etc/os-release) main" \
 | sudo tee /etc/apt/sources.list.d/adoptium.list

sudo apt update
sudo apt install -y temurin-21-jdk
```

Verify:

```
java -version
```

---

# 3. Install OpenEMS Edge

```
mkdir -p ~/downloads
cd ~/downloads
wget https://github.com/OpenEMS/openems/releases/download/2025.11.0/openems-edge.jar
sudo chmod +x openems-edge.jar
```

```
sudo mkdir -p /usr/lib/openems
sudo mv openems-edge.jar /usr/lib/openems/
sudo mkdir -p /etc/openems.d
```

---

# 4. Configure systemd Service

Create:

```
sudo nano /etc/systemd/system/openems.service
```

Paste:

```
[Unit]
Description=OpenEMS Edge
After=network.target

[Service]
User=root
Group=root
Type=notify
WorkingDirectory=/usr/lib/openems
ExecStart=/usr/bin/java -Dfelix.cm.dir=/etc/openems.d/ \
  -jar /usr/lib/openems/openems-edge.jar
SuccessExitStatus=143
Restart=always
RestartSec=10
WatchdogSec=60

[Install]
WantedBy=multi-user.target
```

Enable & start:

```
sudo systemctl daemon-reload
sudo systemctl enable openems
sudo systemctl start openems
```

Check:

```
systemctl status openems
```

# 5. Access OpenEMS Config Manager

On the Pi, open:

```
http://localhost:8080/system/console/configMgr
```

# 6. Add Required Components

In Config Manager:

1. Add a **Scheduler** (any default scheduler)
2. Add **Controller.Api.Websocket**
   - Port: 8085
   - Enabled: true

Restart Edge:

```
sudo systemctl restart openems
```

Verify WebSocket is listening:

```
sudo ss -lntp | grep 8085
```

You should see Java listening on port 8085.

# 7. (Optional) Add Simulation Components

In Config Manager, add:

- Simulator ESS
- Simulator Grid Meter
- Simulator PV

Save and restart OpenEMS.

==========

==========

# PART B — macOS Machine (OpenEMS UI)

==========

==========

## 1. Install Docker Desktop

```
brew install --cask docker
open -a Docker
```

Wait until Docker reports "Docker is running".

Verify:

```
docker version
```

You must see both Client and Server.

---

## 2. Clone OpenEMS Source

```
git clone -b 2025.11.0 https://github.com/OpenEMS/openems
cd openems
```

---

## 3. Build OpenEMS UI Image

```
docker build . \
  -t openems_ui \
  -f tools/docker/ui/Dockerfile.edge
```

---

# 4. Run OpenEMS UI

Replace YOUR_PI_IP with:

- Public IP
- OR VPN IP
- OR Public DNS

Example:

```
docker container run \
  -e WEBSOCKET_HOST=YOUR_PI_IP \
  -p 80:80 \
  -p 443:443 \
  --restart unless-stopped \
  --name openems_ui_container \
  openems_ui
```

---

# 5. Open the UI

On your Mac:

```
http://localhost/login
```

Default login:

- Username: admin
- Password: admin

If UI shows "disconnected":

- Confirm port 8085 is listening on the Pi
- Confirm WebSocket controller exists
- Confirm WEBSOCKET_HOST is correct

---

# Why UI Runs on macOS and Edge Runs on Pi

- Edge interacts with hardware and benefits from native systemd management
- UI behaves like a stateless web application and is ideal for containerization
- Separating them improves stability and flexibility

---

# Troubleshooting

- UI loads but no connection → check port 8085
- Nothing listening on 8085 → WebSocket not configured
- Docker errors on Mac → ensure Docker Desktop is running

---

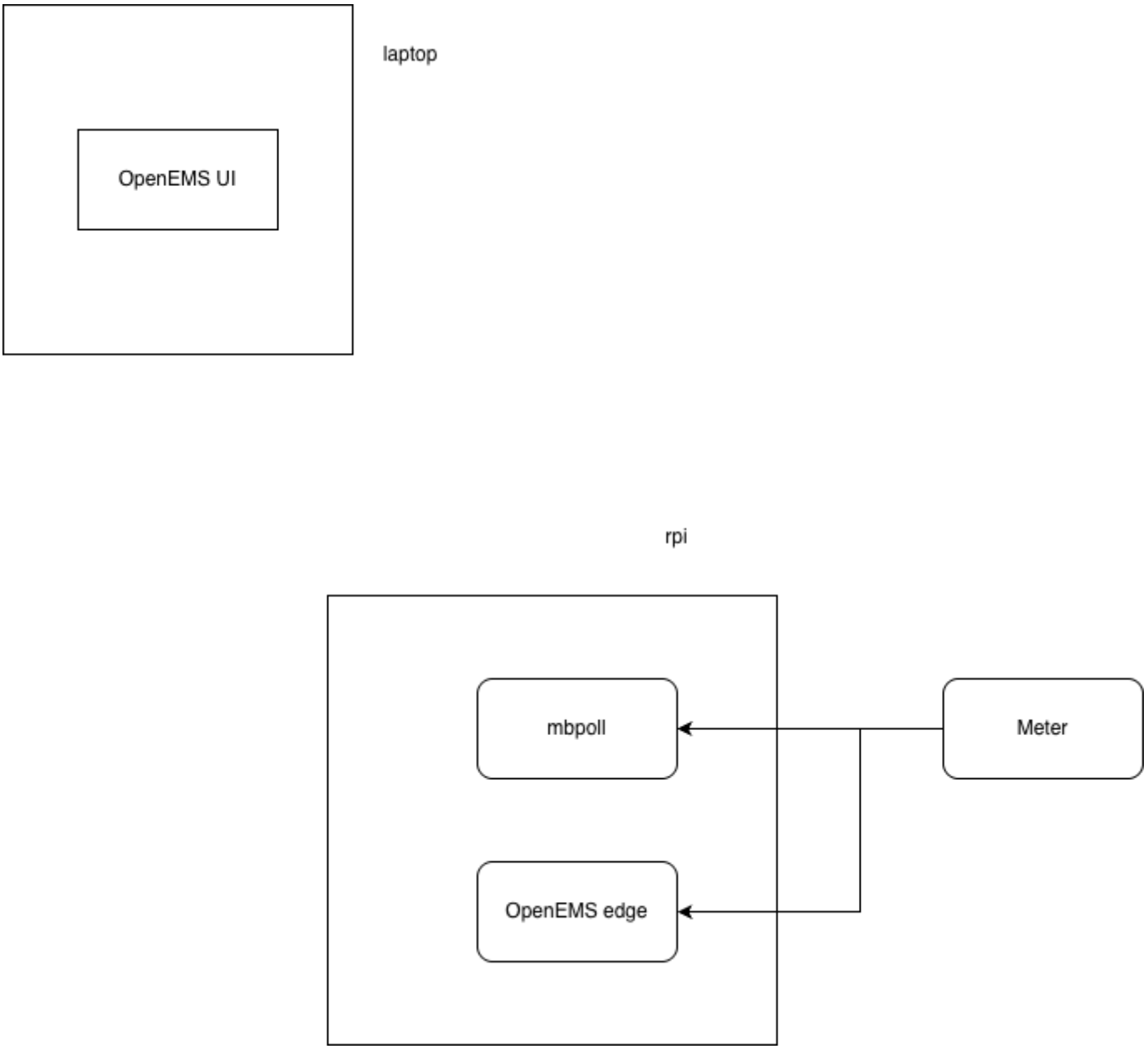
# Security Notes

- Prefer VPN/tunnel (Tailscale/WireGuard) over port forwarding
- Use SSH key-based authentication
- Do not expose ConfigMgr (8080) publicly

---

You now have:

- OpenEMS Edge running on Raspberry Pi
- WebSocket enabled
- Optional simulation components
- OpenEMS UI running on macOS
- UI connected over internet


laptop
OpenEMS UI
rpi
mbpoll
Meter
OpenEMS edge

# DTSU666 3?Phase + mbpoll Command Reference

---

# ? Standard Communication Settings (DTSU666)

Most DTSU666 meters use:

Protocol : Modbus RTU
Baud Rate : 9600
Data Bits : 8
Parity : None
Stop Bits : 2
Mode : Big-endian
Table : Input Registers (FC04)

Base communication block used in all commands:

-m rtu -b 9600 -P none -s 2

---

# ? Critical Flags (Important)

| Flag | Meaning | Required? |
|---|---|---|
| `-t 3` | Input Registers (Function 04) | ✅ Yes |
| `:float` | 32-bit floating point | ✅ Yes |
| `-B` | Big-endian word order | ✅ Yes |
| `-0` | Zero-based addressing | ✅ Yes |
| `-a <id>` | Slave address | ✅ Yes |

| Flag | Meaning | Required? |
|---|---|---|
| `-1` | Poll once | Optional |
| `-l 5000` | Poll every 5 seconds | Optional |

---

# ? Phase?to?Neutral Voltages (L1?N, L2?N, L3?N)

Register: `0x2006`

mbpoll -m rtu -b 9600 -P none -s 2 \
-t 3:float -B -0 \
-r 0x2006 -c 3 \
/dev/ttyUSB0 -a 1 -1

---

# ? Line?to?Line Voltages (L1?L2, L2?L3, L3?L1)

Register: `0x2000`

mbpoll -m rtu -b 9600 -P none -s 2 \
-t 3:float -B -0 \
-r 0x2000 -c 3 \
/dev/ttyUSB0 -a 1 -1

---

# ? Phase Currents (L1, L2, L3)

Register: `0x200C`

mbpoll -m rtu -b 9600 -P none -s 2 \
-t 3:float -B -0 \

-r 0x200C -c 3 \
/dev/ttyUSB0 -a 1 -1

---

# ? Total Active Power (kW)

Register: `0x2012`

mbpoll -m rtu -b 9600 -P none -s 2 \
-t 3:float -B -0 \
-r 0x2012 -c 1 \
/dev/ttyUSB0 -a 1 -1

---

# ? Frequency (Hz)

Register: `0x2044`

mbpoll -m rtu -b 9600 -P none -s 2 \
-t 3:float -B -0 \
-r 0x2044 -c 1 \
/dev/ttyUSB0 -a 1 -1

---

# ? Total Active Energy (kWh)

Register: `0x4000`

mbpoll -m rtu -b 9600 -P none -s 2 \
-t 3:float -B -0 \
-r 0x4000 -c 1 \
/dev/ttyUSB0 -a 1 -1

---

# ? Continuous Polling Example (Every 5 Seconds)

```
mbpoll -m rtu -b 9600 -P none -s 2 \
-t 3:float -B -0 \
-r 0x2006 -c 1 \
-l 5000 \
/dev/ttyUSB0 -a 1
```

---

# ? Change Slave Address (Only One Meter Connected)

⚠ Use Holding Registers (Function 03) for configuration.

Register: `0x002E`

```
mbpoll -m rtu -b 9600 -P none -s 2 \
-t 4:int16 -0 \
-r 0x002E \
/dev/ttyUSB0 -a 1 -W -- 5
```

Changes slave ID from 1 → 5.

---

# ? Quick Communication Test

Fastest health check:

```
mbpoll -m rtu -b 9600 -P none -s 2 \
-t 3:float -B -0 \
-r 0x2006 -c 1 \
/dev/ttyUSB0 -a 1 -1
```

If you see ~230V → communication confirmed.

---

# ? Troubleshooting Quick Guide

| Symptom | Likely Cause |
|---|---|
| Timeout | Wrong address or wiring |
| 4200+ volts | Reading wrong register |
| -8192 values | Wrong data type |
| Nonsense numbers | Missing `-B` |
| No `/dev/ttyUSB0` | USB-RS485 not detected |