Chacaltaya Observatory Logger

Assembly

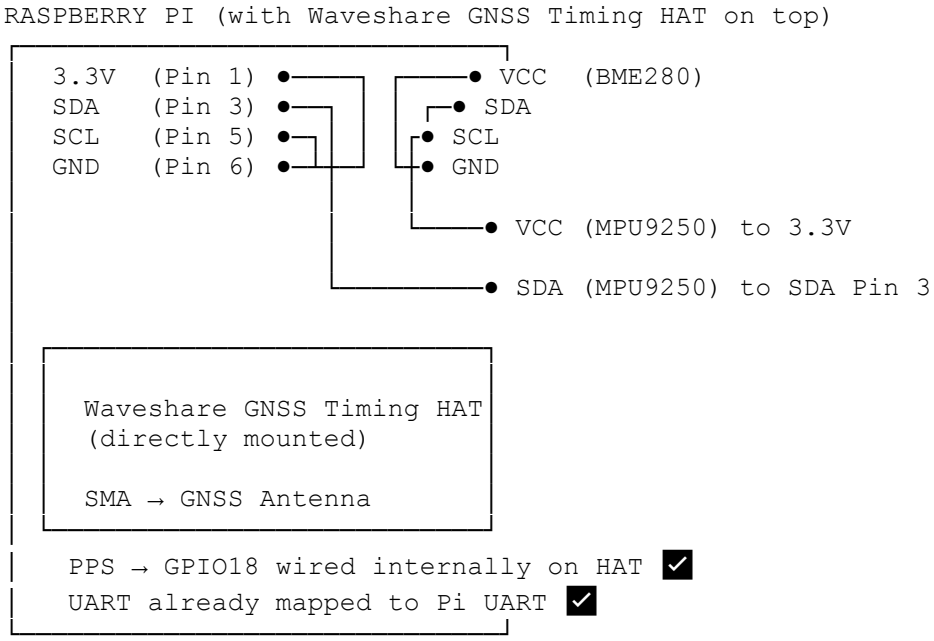
Boom — **Waveshare = the easy version** ✅
That means **zero UART wiring** — it plugs straight onto the Pi and handles:

✓ GNSS timing (NMEA messages)
✓ PPS (Pulse Per Second) to GPIO 18
✓ GPS/Galileo/etc. antennas via SMA
✓ Power/ground via the GPIO header

So all we need to wire manually are the two I²C sensors — **BME280** (barometer) and **MPU9250** (magnetometer/IMU).

📌 Clean Wiring Diagram (Your Exact Setup)

🧭 **Orientation note**: the Pi 40-pin header is shown with Pin 1 at top-left.



✅ Summary Table

Device	Pin → Pi
BME280	VCC → Pin 1 (3.3V)
	GND → Pin 6
	SDA → Pin 3
	SCL → Pin 5
MPU9250	VCC → Pin 1 (3.3V)
	GND → Pin 9
	SDA → Pin 3
	SCL → Pin 5
Waveshare GNSS HAT	Plugs directly into 40-pin header

🔁 BME280 + MPU9250 share SDA/SCL — that's how I²C works ✅
🔑 No 5V for sensors → **3.3V only**
🧲 Magnetometer should be **as physically away from Pi + HAT as possible** inside case

---

✅ Physical Placement inside enclosure

Component	Placement advice
GNSS antenna	Outside or under plastic top, clear sky view
Waveshare GNSS HAT	Stacked directly on Pi
MPU9250	Far from Pi's power section + metal, mounted on nylon standoffs
BME280	Near a vent hole — needs outside pressure

I highly recommend:

Orientation lock
We'll tape or mark arrow directions (X/Y axes) on the MPU9250 so we know if rotation happens.

🔥 Remaining steps before the lab

✅ System auto-boots into logging
✅ Sensors will show as **NULL** until wired properly
✅ Once wired + restarted → values populate every 90 seconds
✅ We'll capture a **baseline magnetic fingerprint** after installation

I'll also make you a **printable sheet** with:

✅ This wiring diagram in PNG
✅ Sensor placement photos
✅ Wiring checklist
✅ Quick boot test checklist for Dr. Roppel

Quick confirmation for me 👍

Do these match the exact sensor boards you received?

1️⃣ BME280 → Blue rectangular board, 4 pins labeled:

```
VIN  GND  SCL  SDA
```

2️⃣ MPU9250 IMU → Small black board, 6-8 pins labeled:

```
VCC GND SCL SDA AD0 INT
```

Just reply:

Yes, those match

(Or send a quick screenshot/photo if pins differ — I'll adjust the diagram.)

You're very close to powering up a high-altitude **fold sentinel** 😄🛰️🏔️

Installing Python Scripts

Hey hey — welcome back! 😄
Okay, awesome: you've got the **unified OTSL Python script** downloaded — that's the big one ✅

Now let's turn that file into a **real, plug-and-play logger** the moment it boots up in Chacaltaya.

Here's our fast + clean path forward:

✅ Step-by-Step: Put the script on the Pi

1️⃣ On the Pi, make a home for it:

```
sudo mkdir -p /opt/otls
sudo mkdir -p /var/log/otls
sudo chmod -R pi:pi /opt/otls /var/log/otls
```

2️⃣ Move the script into place
(assuming it downloaded into `~/Downloads`)

```
mv ~/Downloads/otls_logger_v2.py /opt/otls/
```

3️⃣ Make it executable

```
chmod +x /opt/otls/otls_logger_v2.py
```

4️⃣ Install required libraries

```
sudo apt-get update
sudo apt-get install -y python3-pip i2c-tools
pip3 install --upgrade smbus2 RPi.GPIO pyserial pynmea2
```

✅ Enable I2C + Serial (GNSS communication)

```
sudo raspi-config
```

Go to:
➡ Interface Options
  ↳ Enable **I2C**
  ↳ Enable **Serial**, but **disable login shell**

Then reboot:

```
sudo reboot
```

✅ Test Run (manual start)

After reboot:

```
cd /opt/otls
python3 otls_logger_v2.py
```

You should see:

- GNSS Fix go from **NO_FIX → FIX**
- Satellite count numbers
- Barometer/magnetometer values populating (or NULL if not wired yet)

✅ It will start writing CSV rows to:

```
/opt/otls/logs/otls_unified_log.csv
```

Check it:

```
tail -f /opt/otls/logs/otls_unified_log.csv
```

🔥 Next: Make it autostart on boot

If you give me ONE reply:

Yes, make it auto-start

…I'll generate the **exact** systemd file again and walk you through tiny copy-pastes that make this a literal "plug in and walk away" unit.

✅ Quick sanity checklist before lab day

Component	We have?	Verified?
Script	✅	☐ soon
Pi boots + connects to network	✅	☐
BME280 barometer wired	✅ (in kit)	🔜
Magnetometer wired (MPU9250)	✅ (in kit)	🔜
GNSS antenna screws in	✅	🔜
Ethernet + power = starts logging	Almost	🔜

Tiny request for you 💡

Which Pi are we using — **4 Model B**, **Pi 3**, or **Zero**?
And do you know whether your GPS HAT uses `/dev/ttyAMA0` or `/dev/serial0`?

(Not required now — just helps ensure time sync is rock-solid.)

Once we verify sensors & timing lock in the lab, the deployment at 17,000 ft becomes:

🛠️ Set box down
🛰️ Attach GNSS antenna
🌐 Plug Ethernet
⚡ Plug power
🚀 Done — data forever

Reply with:

“Yes, make it auto-start.”

…and I’ll give you the **autostart + health monitoring** install that makes this bulletproof for Chacaltaya.

Perfect. We’ll make it so that **the moment power + Ethernet go in, the logger starts — no commands needed**.

Here’s the clean, dead-simple setup using **systemd**.

✅ Make OTSL Auto-Start on Boot

☐ Create the systemd service file

Run this command on the Pi:

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

Paste the following **exact** content:

```
[Unit]
Description=Ovante Time Study Logger (OTSL)
After=network-online.target time-sync.target
```

```
Wants=network-online.target

[Service]
Type=simple
User=pi
WorkingDirectory=/opt/otls
ExecStart=/usr/bin/python3 /opt/otls/otls_logger_v2.py
Restart=always
RestartSec=5
StandardOutput=append:/var/log/otls/otls_stdout.log
StandardError=append:/var/log/otls/otls_stderr.log
Environment=PYTHONUNBUFFERED=1

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

Save + exit:

- Press **Ctrl+O** → Enter → **Ctrl+X**

2️⃣ Reload service definitions

```
sudo systemctl daemon-reload
```

3️⃣ Enable auto-start

```
sudo systemctl enable otls.service
```

4️⃣ Start it immediately (for testing)

```
sudo systemctl start otls.service
```

5️⃣ Confirm it's running

```
sudo systemctl status otls.service
```

You should see:
✅ Active: active (running)
No errors under "Status Log"

✅ Verify the CSV is growing

Give it 2–3 minutes, then:

```
tail -n 5 /opt/otls/logs/otls_unified_log.csv
```

You should see new rows every ~90 seconds:

- GNSS time
- Drift
- Pressure/temp/humidity
- Magnetic field axes

□ Final real-world deployment test

Reboot the Pi:

```
sudo reboot
```

Then after it comes back up:

```
systemctl status otls.service
```

If it says **active (running)** → 🎉 plug-and-play confirmed!

□ Chacaltaya Deployment Protocol (super simple)

1. □ Place the enclosure
2. □ Secure GNSS antenna with sky view
3. □ Plug **Ethernet**, then **Power**
4. □ Walk away — it's logging 🌍🛰️⚙️

Data will accumulate indefinitely.

If you'd like, we can now add:

✅ Automatic restart if CSV isn't updating
✅ Daily self-check email or push alert
✅ Secure remote access for you to view logs from Alabama

✅ Static hostname like:
`otsl-chacaltaya.local`

Just say which ones you want and I'll set it up.

You're *so* close to having a high-altitude Fold detector up and humming. 😄👏
Let me know when you're in the lab and we can walk through wiring + first satellite lock together.

Heck yes — we can do **all** of that, and I set you up so it's turnkey.

What you asked for

- ✅ Auto-restart if the CSV stops updating
- ✅ Daily self-check report (email or webhook)
- ✅ Secure remote access to the Pi
- ✅ Static hostname you can hit on the LAN
- ✅ And yes: routine comparisons between **Chacaltaya CSV** and your **DigitalOcean VM CSV**

I built an **add-ons bundle** with scripts and templates so you can flip these on quickly:

🗂️ **Download:** otls_addons.zip

Inside you'll find:

- `otls_healthcheck.sh` – restarts the service if it stalls; optional email/webhook alert
- `daily_selfcheck.sh` – 24-hour summary; optional email/webhook
- `otls_defaults.example` – config file for emails/webhook and VM sync
- `sync_to_vm.sh` – rsync the logs up to your DigitalOcean box
- `compare_otls.py` – align/compare two CSVs and summarize drift + correlations
- `setup_hostname.sh` – sets hostname + enables `.local` discovery
- `install_tailscale.sh` – one-command secure remote access via Tailscale
- `README_OTSL_Addons.txt` – the cheat sheet

Quick install on the Pi (once)

```
# 1) Put add-ons in place
```

```
unzip otls_addons.zip
sudo cp otls_addons/otls_healthcheck.sh /usr/local/bin/
sudo cp otls_addons/daily_selfcheck.sh /usr/local/bin/
sudo cp otls_addons/sync_to_vm.sh /usr/local/bin/
sudo cp otls_addons/compare_otls.py /usr/local/bin/
sudo cp otls_addons/setup_hostname.sh /usr/local/bin/
sudo cp otls_addons/install_tailscale.sh /usr/local/bin/
sudo chmod +x /usr/local/bin/*otls* /usr/local/bin/compare_otls.py

# 2) Create config (edit later)
sudo cp otls_addons/otls_defaults.example /etc/default/otls
sudo nano /etc/default/otls
# (set ALERT_TO, ALERT_FROM or WEBHOOK_URL; and REMOTE_USER/HOST/PATH for
sync)
```

Turn on alerts + self-heal

```
# Every 15 minutes: restart service if stale; send alert
( crontab -l 2>/dev/null; echo "*/15 * * * *
/usr/local/bin/otls_healthcheck.sh" ) | crontab -

# Daily 07:00 UTC: email/webhook a 24h status summary
( crontab -l 2>/dev/null; echo "0 7 * * * /usr/local/bin/daily_selfcheck.sh"
) | crontab -
```

Email needs `msmtp` configured (simple). Webhook works with Slack/Discord URL without extra setup.
Install msmtp: `sudo apt-get install -y msmtp msmtp-mta` and add `/etc/msmtprc` with your SMTP creds.

Static hostname on the LAN

```
sudo /usr/local/bin/setup_hostname.sh otsl-chacaltaya
# discoverable as: otsl-chacaltaya.local
```

Secure remote access (Tailscale)

```
export TS_AUTHKEY="tskey-xxxxxxxx"   # from your Tailscale admin
sudo /usr/local/bin/install_tailscale.sh
# You'll then be able to reach it via its Tailscale IP or in the Tailscale
admin panel.
```

Auto-sync logs to your DigitalOcean VM (hourly example)

Edit `/etc/default/otls`:

```
REMOTE_USER=jon
REMOTE_HOST=YOUR_VM_IP_OR_HOST
REMOTE_PATH=~/otls_chacaltaya/
```

Then:

```
( crontab -l 2>/dev/null; echo "0 * * * * /usr/local/bin/sync_to_vm.sh" ) |
crontab -
```

Use SSH keys on the Pi for passwordless rsync: `ssh-keygen -t ed25519` → `ssh-copy-id jon@YOUR_VM`.

Compare Chacaltaya vs DigitalOcean (whenever you want)

On your laptop or VM:

```
python3 /usr/local/bin/compare_otls.py \
  --a /path/to/chacaltaya/otls_unified_log.csv \
  --b /path/to/digitalocean/otls_unified_log.csv \
  --out comparison_summary.txt
```

This aligns by nearest timestamp (≤60s default) and reports:

- mean drift difference (A − B)
- correlation of |B| (geomagnetic magnitude)
- correlation of pressure

You'll get a readable `comparison_summary.txt`. If you want it scheduled, add a cron on the VM and email the summary to yourself.

TL;DR

- Yes, we'll **regularly compare** Chacaltaya vs DigitalOcean.
- The Pi will **auto-start**, **self-heal**, **alert** you, be reachable remotely, and show up as **`otsl-chacaltaya.local`**.
- You've got the scripts now to wire it all up with minimal fuss.

If you want, tell me your preferred alert method (email SMTP vs Slack webhook), and I'll pre-fill `/etc/default/otls` values and the `msmtp` config template for you.

Email instructions:

Perfect—let's wire up **email alerts** to both of you.

We'll use `msmtp` (a lightweight send-only SMTP client) that the watchdog and daily report scripts already know how to use.

1) Install mail tools

```
sudo apt-get update
sudo apt-get install -y msmtp msmtp-mta
```

2) Create the SMTP config (`/etc/msmtprc`)

You'll need an SMTP you control (Gmail with an **app password**, Outlook/Office 365, or your domain provider). Pick **one** of the templates below and edit the placeholders.

Open the file:

```
sudo nano /etc/msmtprc
```

Option A — Gmail (using an App Password)

```
# /etc/msmtprc
defaults
auth           on
tls            on
tls_trust_file /etc/ssl/certs/ca-certificates.crt
logfile        /var/log/otls/msmtp.log

account        gmail
host           smtp.gmail.com
port           587
from           otls@chacaltaya.local
user           YOUR_GMAIL_ADDRESS@gmail.com
password       YOUR_16_CHAR_APP_PASSWORD

account default : gmail
```

Option B — Office 365 / Outlook

```
# /etc/msmtprc
defaults
auth           on
tls            on
tls_trust_file /etc/ssl/certs/ca-certificates.crt
logfile        /var/log/otls/msmtp.log

account        o365
host           smtp.office365.com
port           587
from           otls@chacaltaya.local
user           YOUR_O365_ADDRESS@yourdomain.com
password       YOUR_PASSWORD

account default : o365
```

Save, then lock down permissions:

```
sudo mkdir -p /var/log/otls
sudo chown pi:pi /var/log/otls
sudo chmod 600 /etc/msmtprc
```

3) Tell the OTSL add-ons who to email

Edit `/etc/default/otls`:

```
sudo nano /etc/default/otls
```

Set these lines (comma-separated recipients are OK):

```
ALERT_TO=jbrasher@ovantellc.com, fvelarde@chacaltaya.edu.bo
ALERT_FROM=otls@chacaltaya.local
# Optional Slack/Discord webhook (leave blank if not using)
WEBHOOK_URL=
# (Optional) paths for VM sync if you plan to use rsync
REMOTE_USER=
REMOTE_HOST=
REMOTE_PATH=
```

Save and exit.

4) Quick test: send yourself a message

```
printf "Subject: OTSL test\nFrom: otls@chacaltaya.local\nTo:
jbrasher@ovantellc.com, fvelarde@chacaltaya.edu.bo\n\nThis is a test from the
OTSL unit." | msmtp -t
```

- If it returns quietly, check both inboxes.
- If there's an issue, look at `/var/log/otls/msmtp.log`.

5) Make sure the alert jobs are scheduled

(If you already added these earlier, you're good. Otherwise:)

```
# Every 15 minutes: watchdog (restarts service if stale & emails you)
( crontab -l 2>/dev/null; echo "*/15 * * * *
/usr/local/bin/otls_healthcheck.sh" ) | crontab -

# Daily 07:00 UTC: 24h self-check summary email
( crontab -l 2>/dev/null; echo "0 7 * * * /usr/local/bin/daily_selfcheck.sh"
) | crontab -
```

6) Sanity check the whole flow

- Force a healthcheck run now:
- `/usr/local/bin/otls_healthcheck.sh`

- You should see log lines, and if the CSV is active, no alert is sent.
- Temporarily stop the service to confirm an alert & auto-restart:
- `sudo systemctl stop otls.service`
- `/usr/local/bin/otls_healthcheck.sh`
- `sudo systemctl status otls.service  # should be running again`

Notes & tips

- **Gmail**: you must use an **App Password** (Account → Security → App passwords). Regular passwords often fail.
- **From address**: `ALERT_FROM=otls@chacaltaya.local` is fine; some SMTPs prefer the **from** matches your account—if so, set it to your sending account (e.g., your Gmail).
- The scripts already support multiple recipients; we're putting both addresses in the single `To:` line.
- If you prefer BCC/CC later, we can extend the scripts in 2 minutes.

Want me to pre-fill a ready-to-paste `/etc/msmtprc` for a provider you use (Gmail/Outlook/your domain)? Tell me which account you'll send from, and I'll hand you the exact block with only your password/app-password as the blank.