09 - RouteTrack Pi — Dashboard Autostart (Gunicorn + systemd)

Date: December 25, 2025
Category: Raspberry Pi / GPS / Flask / systemd

Backlink: 08 – RouteTrack Pi — Local Web Dashboard (Flask API + Leaflet Map)

Project Goal

At this stage RouteTrack already has a working local web dashboard (Flask + Leaflet).
This page productionizes that dashboard so it behaves like a real appliance:

- Starts automatically on boot
- Stays running in the background
- Restarts if it crashes
- Runs the Flask app using **Gunicorn** (production WSGI server)

This is especially important because this Pi will be powered off frequently and used on-the-go.

Why Gunicorn + systemd?

Gunicorn

Gunicorn is a production-grade Python WSGI server that runs Flask reliably with multiple workers.

systemd

systemd provides:

- Autostart on boot
- Automatic restart on failure
- Central logging via `journalctl`
- Clean service management (`start/stop/status`)

Install Gunicorn (venv)

Install into the existing RouteTrack virtual environment:

```
/opt/routetrack/venv/bin/pip install gunicorn
```

My output confirmed it was already installed:

```
Requirement already satisfied: gunicorn in /opt/routetrack/venv/lib/python3.13/site-packages (23.0.0)
```

Create the Dashboard systemd Service

Create the service file:

```
sudo nano /etc/systemd/system/routetrack-dashboard.service
```

Paste the unit file:

```
[Unit]
Description=RouteTrack Local Dashboard (Gunicorn)
After=network-online.target
Wants=network-online.target
```

```
[Service]
# Run the service under my normal user (not root)
User=zippyb
Group=zippyb

# Ensure relative paths work (app:app loads from this folder)
WorkingDirectory=/opt/routetrack/web

# Start Gunicorn on port 5000, listening on all interfaces
# -w 2 = 2 worker processes (lightweight + responsive on Pi)
ExecStart=/opt/routetrack/venv/bin/gunicorn -w 2 -b 0.0.0.0:5000 app:app

# Keep it alive if it crashes
Restart=always
RestartSec=3

# Basic hardening
NoNewPrivileges=true
PrivateTmp=true

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

Reload systemd and Enable the Service

Reload units:

```
sudo systemctl daemon-reload
```

Enable + start the service:

```
sudo systemctl enable --now routetrack-dashboard.service
```

Verify the Service Is Running

Check status:

```
systemctl status routetrack-dashboard.service --no-pager -l
```

My output confirmed it is running and listening properly:

- Service is **active (running)**
- Gunicorn started successfully
- Listening on `http://0.0.0.0:5000`
- Two workers booted

Example output:

```
Active: active (running)
Main PID: 9358 (gunicorn)
Listening at: http://0.0.0.0:5000 (9358)
Booting worker with pid: 9359
Booting worker with pid: 9360
```

Operational Notes

Where logs are stored

Because this is managed by systemd, logs are available via:

```
journalctl -u routetrack-dashboard.service --no-pager -l
```

Last 50 lines:

```
journalctl -u routetrack-dashboard.service -n 50 --no-pager -l
```

Accessing the dashboard

From any device on the same network:

- `http://<PI-IP>:5000`

Find the Pi’s IP:

```
hostname -I
```

Run On Demand (Manual Service Controls)

Start:

```
sudo systemctl start routetrack-dashboard.service
```

Stop:

```
sudo systemctl stop routetrack-dashboard.service
```

Restart:

```
sudo systemctl restart routetrack-dashboard.service
```

Status:

```
systemctl status routetrack-dashboard.service --no-pager -l
```

Next Steps

Now that the dashboard is always online, the next improvements will focus on making RouteTrack truly “truck ready”:

1. Shift mode (Start Shift / End Shift buttons)
2. Offline mapping options (for no cell coverage)
3. Exports (GeoJSON/CSV) and backup strategy for on-the-go use

---

Revision #2
Created 25 December 2025 15:59:08 by Nate Nash
Updated 25 December 2025 21:16:49 by Nate Nash