00 - RouteTrack Pi — Project Overview

Project Name: RouteTrack Pi
Category: Embedded Linux · GPS Telemetry · Data Processing · Flask
Platform: Raspberry Pi
Status: Active / Production-Ready Prototype

Project Summary

RouteTrack Pi is a **vehicle-mounted, headless GPS telemetry and route analysis system** built on a Raspberry Pi.
It continuously logs GPS data, processes that data into meaningful metrics (mileage, stops, time-on-site), and presents results through a **local web dashboard** — all without requiring constant internet connectivity.

The system is designed with **reliability, recoverability, and real-world vehicle use** in mind:

- Frequent power loss
- Offline operation
- GPS signal dropouts
- Long-running background services
- Minimal SD card wear

This project mirrors architectural patterns used in **fleet tracking and telemetry systems**, scaled down to run on low-power hardware.

Core Capabilities

- Continuous GPS route logging
- Accurate mileage calculation (Haversine)
- Stop detection & time-on-site tracking
- Daily route summaries
- Shift-based session tracking
- Local interactive map dashboard (Leaflet)
- Fully automated background services
- Offline-first, local-only data storage
- Safe daily processing via systemd timers

Hardware Used

Component	Description
Raspberry Pi 3 B+	Primary compute platform
GlobalSat BU-353N USB GPS Receiver	High-sensitivity USB GPS (NMEA 0183)
128 GB microSD card	OS + data storage
Always-on cooling fan	Wired directly to 5V rail for vehicle use
Vehicle USB power	Portable, ignition-controlled

Why this GPS receiver:
The BU-353N is widely supported on Linux, requires no proprietary drivers, and is well-suited for mobile deployments.

Operating System & Base Software

- **Raspberry Pi OS Lite (64-bit)**
- Headless (no desktop environment)
- SSH-only administration
- NetworkManager for Wi-Fi management
- systemd for all service orchestration

Networking Design

- Multiple saved Wi-Fi networks

- Automatic failover:
  - Home Wi-Fi (higher priority)
  - Phone hotspot (fallback)
- Seamless switching without reboot
- Fully functional offline

GPS Subsystem Architecture

```
USB GPS Receiver
  ↓
Linux USB-Serial Driver
  ↓
/dev/gps0 (udev symlink)
  ↓
gpsd-standalone.service
  ↓
TCP localhost:2947
  ↓
RouteTrack Logger
```

Key Design Choices

- **gpsd socket activation disabled**
- **Dedicated standalone gpsd service**
- Fixed baud rate (4800)
- Stable device path using udev
- TCP access instead of UNIX socket (permission stability)

GPS Service

- **Service Name:** `gpsd-standalone.service`
- **Binary:** `/usr/sbin/gpsd`
- **Device:** `/dev/gps0`
- **Port:** `127.0.0.1:2947`

RouteTrack Directory Layout

All project files live under a single, predictable root:

```
/opt/routetrack/
├── bin/        # Executable scripts
├── data/       # SQLite database and exports
├── logs/       # File-based logs (future use)
├── config/     # Database schema and config files
├── venv/       # Python virtual environment
└── web/        # Flask dashboard application
```

Python Virtual Environment

- **Location:** `/opt/routetrack/venv`
- Isolates RouteTrack dependencies
- Avoids modifying OS-managed Python
- Used by all RouteTrack services

Installed Python Packages

- `flask`
- `gunicorn`
- Standard library only for GPS, SQLite, math

Data Storage (SQLite)

Database File

```
/opt/routetrack/data/routetrack.sqlite
```

Tables

Table	Purpose
`gps_points`	Raw GPS telemetry (append-only)
`stop_events`	Derived stationary events
`daily_summary`	Aggregated per-day metrics
`shifts`	User-controlled session boundaries

Design Philosophy

- SQLite in WAL mode
- Raw data is **never modified**
- Derived data is **fully regenerable**
- Optimized for long-running, low-power systems

GPS Logging Service

Script

```
/opt/routetrack/bin/routetrack-logger.py
```

systemd Service

```
routetrack-logger.service
```

Responsibilities

- Connect to gpsd via TCP
- Subscribe to JSON stream
- Filter TPV messages
- Batch inserts into SQLite
- Auto-reconnect on failures
- Journald logging

Why this matters:
The logger is intentionally lightweight and resilient — it prioritizes **never stopping**, even during

GPS hiccups or reboots.

□□ Route Processing & Intelligence

Processing Script

```
/opt/routetrack/bin/routetrack-process.py
```

What It Calculates

- Mileage (Haversine distance)
- Moving vs stopped time
- Stop events (speed + dwell)
- Daily summaries

Key Rules

- Only `mode = 3` fixes are trusted
- GPS drift filtered
- Speed thresholds applied
- Stop dwell time enforced

Automated Processing (systemd)

Wrapper Script

```
/opt/routetrack/bin/routetrack-run-processor.sh
```

Services

- `routetrack-processor.service` (oneshot)
- `routetrack-processor.timer` (daily)

Automation Workflow

1. Stop logger (release DB lock)
2. Run processor
3. Restart logger

This ensures **zero SQLite locking issues**.

Local Web Dashboard

Stack

- Flask (read-only API)
- Gunicorn (production WSGI)
- Leaflet + OpenStreetMap

Dashboard Files

```
/opt/routetrack/web/
├── app.py
├── templates/
│   └── index.html
└── static/
```

API Endpoints

- `/api/points/<date>`
- `/api/stops/<date>`
- `/api/summary/<date>`

systemd Service

```
routetrack-dashboard.service
```

- Starts on boot

- Auto-restarts
- LAN-accessible
- No database writes (safe concurrency)

Shift Mode

Shift Mode introduces **user-defined session boundaries** independent of calendar days.

Table

shifts

Purpose

- Accurate per-shift metrics
- Handles overnight work
- Prevents test runs from polluting data
- Designed for vehicle usage patterns

Reliability & Operational Design

- All components managed by **systemd**
- Centralized logging via `journalctl`
- Safe service dependencies
- No cron jobs
- SD-card friendly writes
- Offline-first operation
- Power-loss tolerant

Why This Project Matters

RouteTrack Pi demonstrates:

- Embedded Linux service orchestration
- Real-world GPS data handling
- Data pipeline design
- SQLite optimization
- systemd automation
- Web API + dashboard integration
- Production-style architecture on constrained hardware

This is **not a script** - it’s a **system**.

Future Enhancements (Planned)

- GeoJSON / CSV exports
- VPS sync & backups
- Map styling improvements
- GPS health watchdogs
- Authentication (optional)
- Historical route replay

Revision #1
Created 25 December 2025 18:11:35 by Nate Nash
Updated 25 December 2025 18:15:28 by Nate Nash