

# OFFLINE ROUTER

## User Guide

*QGIS Plugin • Offline Routing • Turn-by-Turn Directions*

Offline Router is a QGIS plugin that computes driving routes between multiple waypoints using a local SpatiaLite routing database — no internet connection required. It produces a styled route layer on the map with full turn-by-turn directions.

<b>Plugin Name</b>	Offline Router
<b>QGIS Version</b>	3.10 or later
<b>Platform</b>	Windows, macOS, Linux
<b>Database Format</b>	SpatiaLite (.sqlite)
<b>Projection</b>	EPSG:4326 (WGS 84) or EPSG:3857 (Web Mercator)
<b>Network</b>	Fully offline — no internet required

## 1. Overview

Offline Router solves two common field-mapping problems: routing in areas with no mobile data coverage, and keeping sensitive route data entirely on your own machine.

The plugin queries a pre-built SpatiaLite road-network database using a Dijkstra shortest-path algorithm, displays the computed route as a vector layer styled with directional arrows, and produces a formatted list of turn-by-turn driving instructions — all inside QGIS, all offline.

**Key capabilities:** Multi-stop routing • Turn-by-turn directions • Metric or Imperial distances • QML-styled output layers • Built-in database builder (Windows)

## 2. Installation

### 2.1 Requirements

- QGIS 3.10 or later
- mod\_spatialite library (bundled with QGIS on Windows; see below for macOS/Linux)
- A SpatiaLite routing database (.sqlite) — see Section 4 to build one


## 2.2 Install the Plugin

1. In QGIS, open Plugins → Manage and Install Plugins → Install from ZIP.
2. Browse to offline\_router.zip and click Install Plugin.
3. The plugin will appear in the Plugins menu and in the toolbar.

## 2.3 mod\_spatialite




The routing engine requires the mod\_spatialite extension to be loadable by Python's sqlite3 module.

<b>Windows</b>	Included automatically with the OSGeo4W/QGIS installer. No action needed.
<b>macOS</b>	Install via Homebrew: brew install spatialite-tools
<b>Linux</b>	Install via APT: sudo apt install libsqlite3-mod-spatialite

 **Note:** The status bar at the bottom of the plugin dialog confirms whether mod\_spatialite was found. If it shows x not found, routing will not work until the library is installed.

## 3. Quick-Start Workflow


The complete workflow from opening the plugin to viewing a route on the map takes less than a minute once a routing database is available.

Step-by-Step Workflow	
<b>1</b>	<b>Set your project CRS</b> Open QGIS Project Properties and set the CRS to EPSG:4326 (WGS 84) or EPSG:3857 (Web Mercator). Offline Router only supports these two projections.
<b>2</b>	<b>Open the plugin</b> Click the Offline Router toolbar icon, or use Plugins → Offline Router.
<b>3</b>	<b>Select a routing database</b> Click Browse... in the SpatiaLite Routing Database panel and navigate to your .sqlite routing file. A green status message confirms the database is valid.
<b>4</b>	<b>Add waypoints</b> Click  Add Waypoint to enter picking mode (the button stays pressed). Click points on the map canvas to set Start, Via stops, and End. Each click adds a colored preview marker on the canvas.
<b>5</b>	<b>Review and reorder stops</b> The waypoints list shows every stop in order. Drag rows to reorder, use the ▲ / ▼ buttons, or double-click any row to pan the map to that point. Use  Remove Selected or Clear All to delete stops.
<b>6</b>	<b>Find Route</b> Click  Find Route. A progress bar appears while the engine runs. When complete, two

	layers are added to the map and the turn-by-turn directions panel is populated.
7	<b>Review the route</b> The map zooms to fit the route. The Route Waypoints layer sits above the Route layer in the layer panel. The directions panel shows total distance, estimated travel time, and each turn instruction.

## 4. Building a Routing Database

The **Build Routing File** button (... icon beside Browse) opens the database builder. This tool converts an OpenStreetMap PBF file into a Spatialite routing database in four automated steps.

 **Windows only:** The database builder uses bundled Windows executables (osmconvert64.exe, spatialite\_osm\_net.exe, spatialite.exe). On macOS and Linux, use these tools directly from the command line, or download a pre-built routing database.

### 4.1 Inputs

<b>Source PBF file</b>	An OpenStreetMap .pbf export for your region. Download from geofabrik.de or overpass-api.de.
<b>GPX boundary file</b>	A GPX track that forms a closed polygon around the area you want to route in. The tool clips the PBF to this boundary before import.
<b>Output .sqlite</b>	The path where the finished routing database will be saved.

### 4.2 Build Steps (automated)

4. Convert the GPX boundary track to an Osmosis .poly filter file.
5. Run osmconvert64.exe to clip the PBF to the boundary polygon.
6. Run spatialite\_osm\_net.exe to import the clipped road network into a raw Spatialite database.
7. Run spatialite.exe with the bundled createrouting-by-car.sql script to generate the optimised routing tables.

The log panel shows live output from each step. When the build succeeds, the new database path is automatically loaded into the main dialog.

## 5. Working with Waypoints

### 5.1 Adding Waypoints

Click **+** Add Waypoint to enter picking mode. The button stays highlighted while active. Each left-click on the map canvas adds a waypoint:

- **First click** — sets the Start point (green marker)


- **Middle clicks** — add Via stops (blue markers), numbered Via 1, Via 2, etc.
- **Last point** — automatically labeled End when a second or later point exists (red marker)

Preview markers appear on the canvas immediately after each click, in a temporary Route Waypoints (Preview) layer. This layer is replaced by the final styled layer when a route is computed.


## 5.2 Reordering Waypoints

- Drag and drop rows in the waypoints list to change the stop order.
- Use the ▲ (up) and ▼ (down) buttons beside the list for single-step moves.
- Double-click any row to pan the map canvas to that waypoint.

## 5.3 Removing Waypoints

- **Remove Selected:** select a row and click  Remove Selected.
- **Clear All:** removes all waypoints and the preview layer from the canvas.

## 5.4 Reloading Waypoints

Click  **Reload Waypoints** to import waypoints from an existing Route Waypoints layer in the project. This is useful for re-running a route after closing and reopening the plugin, or for loading a waypoints layer saved from a previous session.

## 6. Options

<b>Remove previous route layers</b>	When checked (default), the Route, Route Waypoints, and Road Routing Nodes layers from the previous run are removed before adding new ones. Uncheck to accumulate multiple route results on the map.
<b>Distance units — Metric</b>	Distances in the directions panel and route attributes are shown in meters and kilometers.
<b>Distance units — Imperial</b>	Distances are shown in feet and miles. The selected unit is remembered between sessions.

## 7. Output Layers

When a route is computed, two layers are added to the Routing (Temporary Layers) group in the layer panel. Waypoints always sit above the Route in the panel, ensuring pin markers are never hidden beneath the route line.

### 7.1 Route Waypoints

A point layer containing all waypoints used in the current route, styled using the bundled Route-Waypoints-Style.qml.

- **Start** — green marker
- **End** — red marker
- **Via N** — blue diamond markers for intermediate stops

Labels (Start, Via 1, End, etc.) are displayed next to each marker with a white halo for legibility.

## 7.2 Route

A MultiLineString layer containing the computed route geometry, styled using the bundled Route-Arrow-Style.qml. The style renders a purple line with directional arrow markers at regular intervals, showing the travel direction along every road segment.

The layer carries five attribute fields:

<b>distance_m</b>	Total route length in meters.
<b>distance</b>	Total length formatted in the selected unit system.
<b>travel_duration</b>	Estimated travel time based on routing costs.
<b>waypoints</b>	Number of waypoints used in the route.
<b>directions_text</b>	Full turn-by-turn directions as a single text string.

☐ **Route directionality:** Each road segment geometry is automatically oriented to match the travel direction (node\_from → node\_to). Segments stored in reverse in the database are flipped before display, so arrow markers always point the correct way along every road.

## 8. Turn-by-Turn Directions

The directions panel at the bottom of the dialog is populated after each successful route calculation.

### 8.1 Summary Bar

The bold summary line shows:

- Number of waypoints and legs
- Total distance in the selected unit
- Estimated travel time

### 8.2 Directions List

Each step in the list shows:

<b>Step number</b>	Sequential instruction number.
<b>Instruction</b>	Turn description: Head N, Turn right, Bear left, Continue straight, U-turn, Arrive at, etc.
<b>Street name</b>	Name of the road where available.


<b>Leg distance</b>	Distance for this individual step.
<b>Cumulative dist.</b>	Distance from the Start at this point in the route.

Waypoint boundaries (Via 1, Via 2, End) are shown as separator lines in the directions list so you can clearly see where each leg ends.

## 9. Layer Panel Organization

All Offline Router layers are placed inside a Routing (Temporary Layers) group at the top of the layer panel. Within the group, layers are always ordered:

8. Route Waypoints (top — markers always visible above the line)
9. Route (route line with directional arrows)
10. Road Routing Nodes (network nodes, added when a database is first loaded)

 **Tip:** The Routing group is a standard QGIS layer group. You can rename it, move it, toggle its visibility, or save the layers as permanent files using Layer → Save As at any time.

## 10. Troubleshooting

<b>mod_spatialite not found</b>	Install the SpatiaLite extension for your OS (see Section 2.3). Confirm the status message at the bottom of the dialog shows ✓ found.
<b>Unsupported Projection warning</b>	Change your QGIS project CRS to EPSG:4326 or EPSG:3857 via Project → Properties → CRS.
<b>No route found</b>	The two endpoints may be in disconnected parts of the network, or one-way restrictions block the path. Try clicking closer to a road intersection.
<b>Route missing road segments</b>	The routing database may not cover the full area. Rebuild the database with a larger boundary GPX (Section 4).
<b>Database tables missing</b>	Run the Build Routing File tool to create a valid database, or check that the .sqlite file was built with the bundled createrouting-by-car.sql script.
<b>Build tool fails (Windows)</b>	Ensure all four executables are present in the plugin's tools/ folder: osmconvert64.exe, spatialite_osm_net.exe, spatialite.exe, and that createrouting-by-car.sql is in the same folder.
<b>Arrows pointing wrong way</b>	Verify the routing database was built with the standard workflow. The directional correction is applied per-segment using node coordinates; if a segment has no node match it is left as-is.

## 11. Plugin File Structure

The installed plugin folder (offline\_router) contains:

<b>plugin.py</b>	Plugin entry point; registers the toolbar icon and menu item.
<b>dialog.py</b>	Main routing dialog: waypoints, options, directions panel, map layer creation.
<b>routing.py</b>	Routing engine: SpatiaLite queries, Dijkstra fallback, directions builder, segment orientation.
<b>advanced_setup.py</b>	Database builder dialog (Windows).
<b>map_tool.py</b>	QGIS map tool for capturing click points on the canvas.
<b>icons/icon.png (.svg)</b>	Plugin toolbar icon.
<b>styles/Route-Arrow-Style.qml</b>	QGIS layer style for the computed route line.
<b>styles/Route-Waypoints-Style.qml</b>	QGIS layer style for the waypoints layer.
<b>tools/</b>	Windows executables and SQL script for the database builder.
<b>LICENSE</b>	Plugin license.

*Offline Router is built on the SpatiaLite spatial database engine and OpenStreetMap road network data. Route quality depends on the completeness of the source OSM data and the coverage of the routing database.*