MWPTools

Tools for MultiWii NAV and iNav flight logging on free OS

<https://github.com/stronnag/mwptools>

"Sweet dreams and flying machines"[1]

Overview

mwp is a mission planner and flight logger for MSP (Multiwiii Serial Protocol) compatible flight controller firmware (Multiwii and iNav at least).

From its MultiWii origins **mwp** has evolved to support navigation capabilities in iNavflight <https://github.com/digitalentity/cleanflight/tree/inav-dev>, in particular telemetry logging and blackbox replay. **iNav** is now the main development target, however the manual is currently biased toward MW mission planning.

The aim is to at least enable users on free operating systems to:

- Plan a mission using a geographic display;
- Edit missions, way point and actions;
- Upload the mission to a capable iNav / MultiWii-NAV flight controller;
- Download a mission from a capable iNav / MultiWii-NAV flight controller;
- Load and save missions to a file, in a format compatible with WinGui (and ezgui);
- Track the mission in real time on open source compatible platforms, where suitable telemetry devices are available;
- Provide a geospatial tools for replay of Blackbox log files;
- Mission logging and log conversion to at least KML and GPX.

Obsolete tools in the **mwptools** suite include:

- **pidedit** : A simple PID editor;
- **switchedit**; A simple switch editor;
- **mspsim** : A simulator for MSP (MultiWii Serial Protocol). **mspsim** provides enough of the MSP to facilitate the development of applications such as **mwp** and **pidedit**.

Caveat

The author has flown his GPS capable multi-rotors using iNav, MultiWii-NAV and **mwp**, including autonomous flight modes; it appears to work well, however there is **NO WARRANTY. You have been warned**.

Note also that this document frequently lags behind the source code. You are advised to check the applications (--help) for additional options not listed in this document, and note that the source code is the ultimate documentation.

1 James Taylor, Fire and Rain. Full line is 'sweet dreams and flying machines in pieces on the ground', so let's skip the final part.

mwp – “A mission planner for the rest of us”

mwp is the mission planner component of **mwptools**. It provides a graphic user interface for creating, editing and managing MultiWii NAV missions.

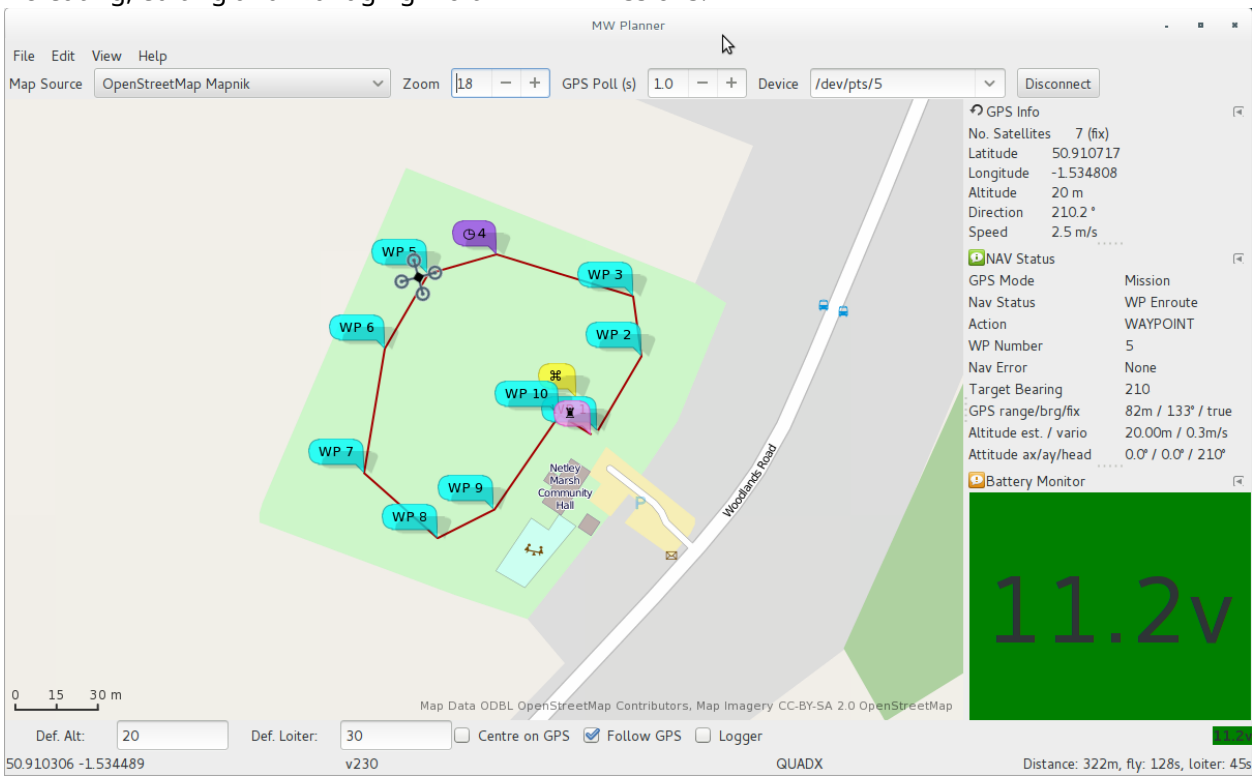


Illustration 1: **mwp** window showing a mission and mission tracking

Illustration 1 Shows the **mwp** main window and the main user interface elements.

- A menu bar. The menu options are described below;
- A setting bar, allowing the selection of the map source, zoom level, GPS update and the serial device used to connect to the flight controller. In this example, a pseudo-terminal is used with **mspsim** (the **mwptools** MSP simulator) providing the GPS and navigation status;
- The main body of the application window it divided into a number of panes:
  - The map pane shows the map view and the current mission. If GPS following is enabled, the aerial vehicle is also shown;
  - A dock area. This can display a variety of information, include the mission (in tabular form), the current NAV status, the current GPS data and the battery status. In the screen shot, the mission tote is hidden, but can be enabled from the View menu. The dock is further described in the section “Using the Dock“. Note that many of the screen shots following were taken before the dock was implemented;
- An options bar, where default values of altitude and loiter time may be set, and toggles control whether the map is centred on the GPS, whether the vehicle icon follows the GPS, whether logging is enabled and whether audio prompts are made (see Illustration 16);
- A status bar shows the current map centre position, the firmware version, the vehicle type and mission statistics (distance, flight time and loiter times). Jumps are taken into account

when calculating these values, unless the jumps have infinite repeats.

Settings Bar

The settings bar controls how **mwp** displays the map, connects to a telemetry device and sets zoom and polling rates:

- The map source. This is pull down list of map sources. It uses the libchamplain defaults, augmented by Bing Aerial and user additions (e.g. the Bing Proxy described later);
- The map zoom. The levels are defined by the individual map sources, typically from 0 (whole world) to 19 (covering 100-200m on a typical screen);
- The telemetry device. This is may be an OS defined device name (with optionally a baud rate) or a TCP/IP definition for use with a ESP8266 WiFi / serial device; examples:
  - /dev/ttyUSB0 (e.g. a 3DR radio);
  - /dev/rfcomm1 (bluetooth);
  - A bluetooth UID (e.g. 98:D3:31:FC:32:44), the preferred nomenclature for BT on Linux;
  - /dev/3dr@9600 (Linux udev renamed 3DR radio device, forced to 9600 baud)
  - /dev/pts/6 (pseudo-terminal for testing with mspsim);
  - tcp://mwp-fc:23 (TCP connection to ESP8266 node 'mwp-fc' using port 23);
  - :6666, localhost:6666, udp://:6666, udp://localhost:6666 (UDP connection for testing);

    At the risk of confusion, mwp and mspsim can either bind to a UDP port (effectively the server, or act as a UDP client). If no hostname is given, the application instigates a UDP server, if a hostname is given, it expects something to be listening for UDP on the given port. For mspsim / mwp is should not matter which is which, for samples/parseraw.rb (a LTM test tool), mwp should listen (i.e. be invoked without a host name).
- The Connect / Disconnect button connects the telemetry device. The auto button does so automatically.

Opening Missions

Existing missions can be loaded from the File / Open menu item. A standard file chooser, with filters of 'Mission' (*.xml, *.mission) will load existing mission files, including those created with WinGui or EZGUI. Where the mission was previously saved by **mwp**, the mission file will include map centre and zoom information, so loading a mission will zoom to the mission location.

In recent versions, mwp will provide a preview of the mission, if the mission is ‘new’ a geometric outline is shown; once the mission has been saved (or a preview saved with CTRL-V), then a geospatial preview is shown.

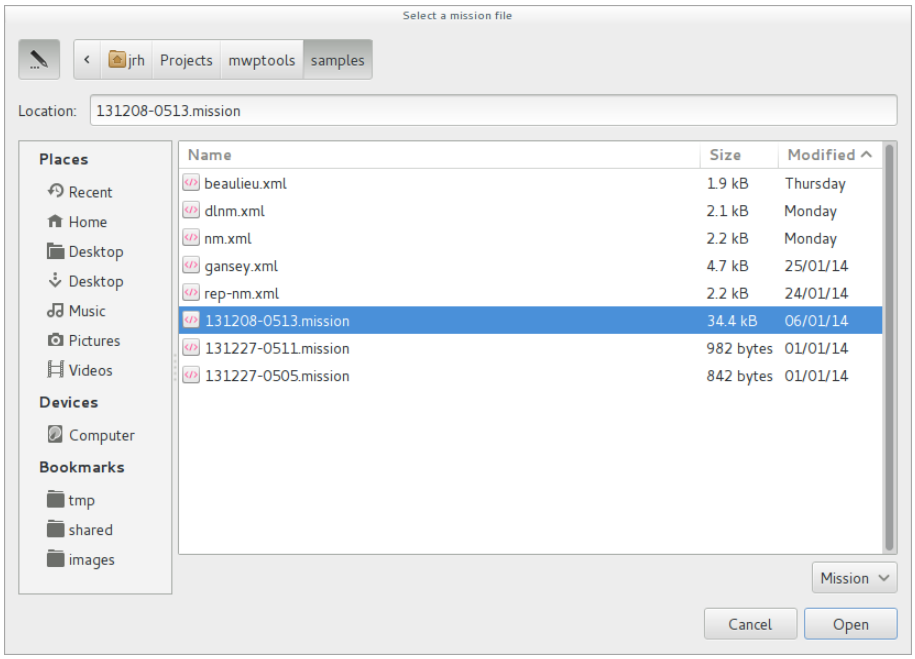


Illustration 2: Opening a mission file (EOSBandi / WinGui sample file)

Opening the selected file loads the mission and displays the mission data on the map, and in the tote, replacing any previous mission data (note this example WinGui mission would not load on a real FC, due to the excessive number of way points):

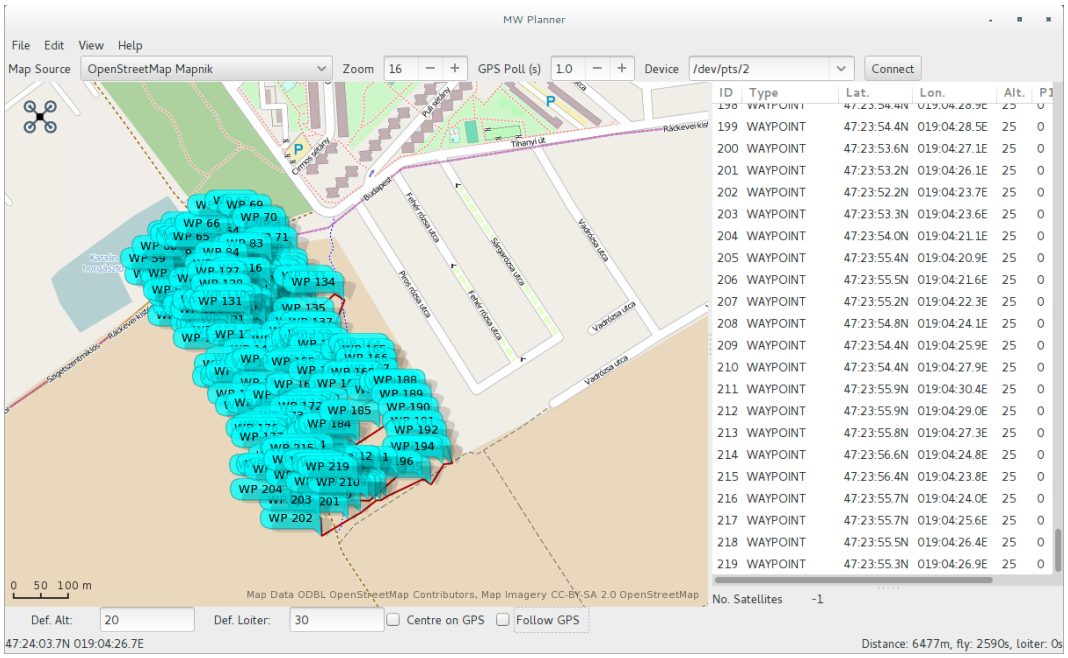


Illustration 3: Import of EOSBandi / WinGui sample file (don't fly this one)

Creating and editing missions

New way points and actions may be created in two ways:

1. Left mouse button click on the map, see the setting auto-wp-edit for how can be switched;

2. From the right mouse button pop-up menu in the mission tote.

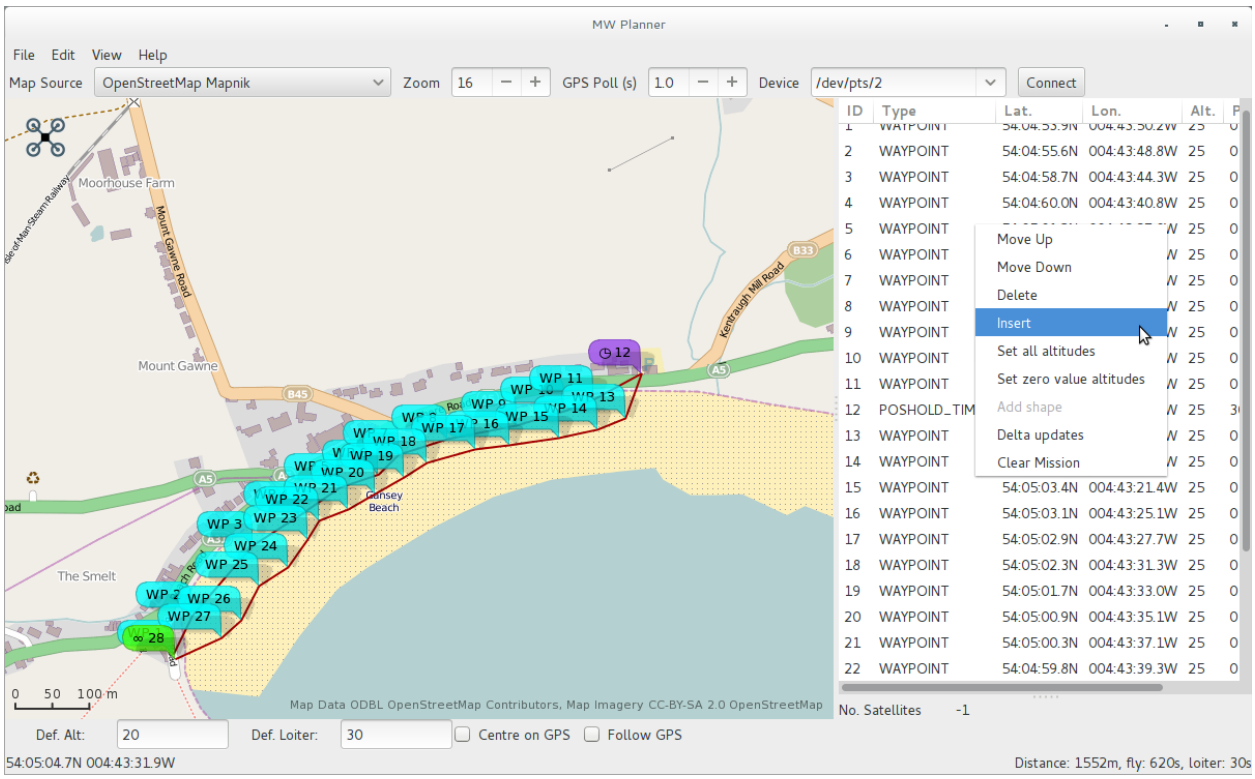


Illustration 4: Mission pop-up menu (let's go for a [quick] pint at "The Shore" hotel)

In Illustration 4, the mission right mouse button pop-up menu has been invoked, and the “Insert” action selected.

Way points and actions may be created for new missions, or modified in existing (loaded) missions.

Mission points are created either by a left click (see also auto-wp-edit) on the map or by the mission pop-up menu, “Insert” action (see Illustration 4). Depending on how the point is created, it will be displayed in a slightly different way. This distinction is somewhat artificial, but seemed logical at the time --- it may be changed in the future. This is shown in Illustration 5, and explained further below.

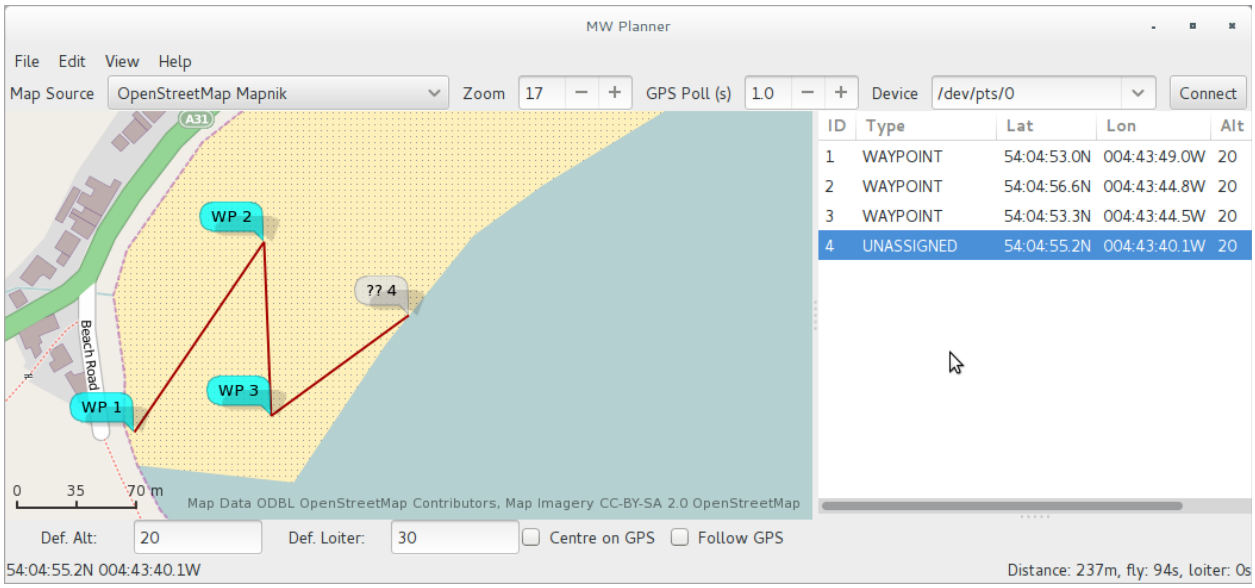


Illustration 5: Entering way points in **mwp**

In Illustration 5, way points 1,2, and 3 have been entered by left click on the map (as WAYPOINT is the most common way point type, this is the default). Way point 4 (??4) has been entered from the mission pop-up menu, and thus has type of UNASSIGNED. UNASSIGNED way points are *not* saved or uploaded as part of a mission; they must be converted to a 'real' mission way point or action (see Type below).

Editing way points

Way points are edited in the tote. When a row is selected, the tote column headers will change to indicate the data fields appropriate to the point type (in particular the “parameters” P1,P2,P3 whose interpretation is dependent on the point type.

- Position. The position of a way point may be changed by dragging the way point icon on the map. If the display mode is decimal degrees (see Preferences and Settings), then the position may also be edited in the tote. Where the position is DMS.s, then the position cannot currently be edited in the tote;
- Order. The order of way points may be changed by either:
  - Using the “Move Up” and “Move Down” entries from the mission pop-up menu; or
  - Dragging the tote item to the desired position. In order to drag, the entry must be 'grabbed' on the ID column (Illustration 6). In that screen-shot (below), way point 7 is being dropped between way points 3 and 4.
  - At the end of the drop, the list and markers on the map will be re-ordered.

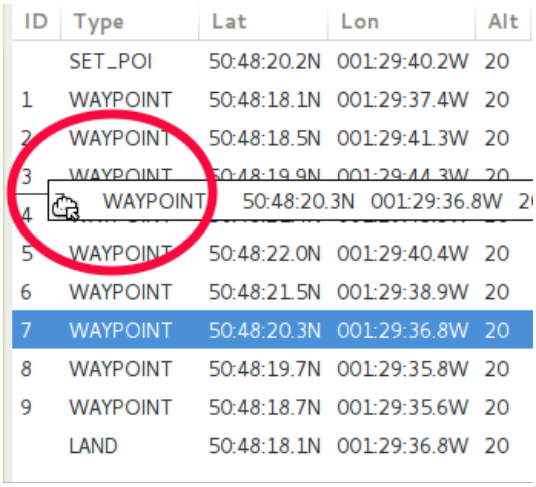


Illustration 6: Dragging a way point to re-order.
Note the 'grab' by the ID column

- Type. The way point type may be selected from a drop down menu embedded in the “Type” column of the tote:

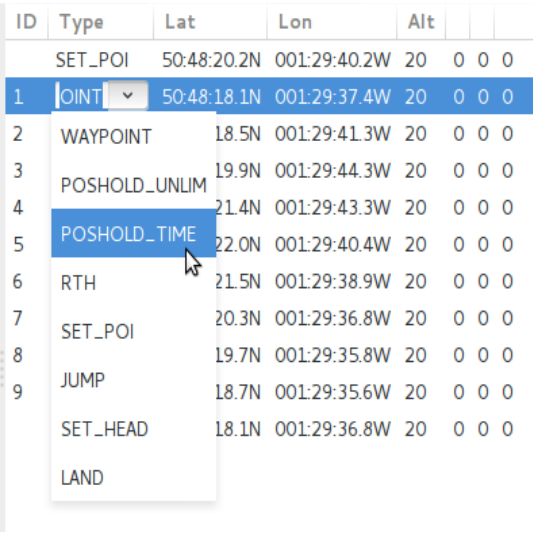


Illustration 7: Changing a way point type

Once the type has been changed, default parameters for that way point type or action will be set. The defaults are discussed in the section “Preferences and Settings“, and may be edited in the tote.

The type may also be set by a right mouse button click on the map symbol.

- Altitude. New points are created with the default altitude (from the “defaults and settings bar”, and as pre-defined in preferences). Some basic validation is performed;
- Parameters P1, P2 and P3. The parameters P1,P2 and P3 are integer values that have a meaning specific to the way-point type or action. For example, for action type of JUMP, P1 is the point to which to jump, and P2 is the number of repeats. The author's understanding of the parameters associated with each way point type or action is given in the reverse engineered protocol documentation (e&oe);
- The use of the parameters is dependent on the flight controller firmware; for example (as of

2016-01-01):
  - inav uses WAYPOINT P1 as speed for the leg, mw-nav does not;
  - mw-nav supports all waypoint types, inav does not (yet).
- Multiple points may be added using the pop-up “Add shapes” option (see below).

Other mission popup menu actions

- Delete. The delete action will delete the selected (highlighted) way point(s). If no way point is selected, this option has no affect;
- Set all altitudes. Sets all selected way point (but not LAND) altitude values to the current “Def Alt” value;
- Set zero value altitudes. As for “Set all altitudes”, but only affects points with an altitude of zero;
- Add shape. Where the mission consists of *exactly one* SET POI point as the first point in a mission (there may also be other extant way-points), this option will display a dialogue to enter the number of points in a shape, the radial distance (from the SET POI to a point), an offset angle and the direction of rotation:

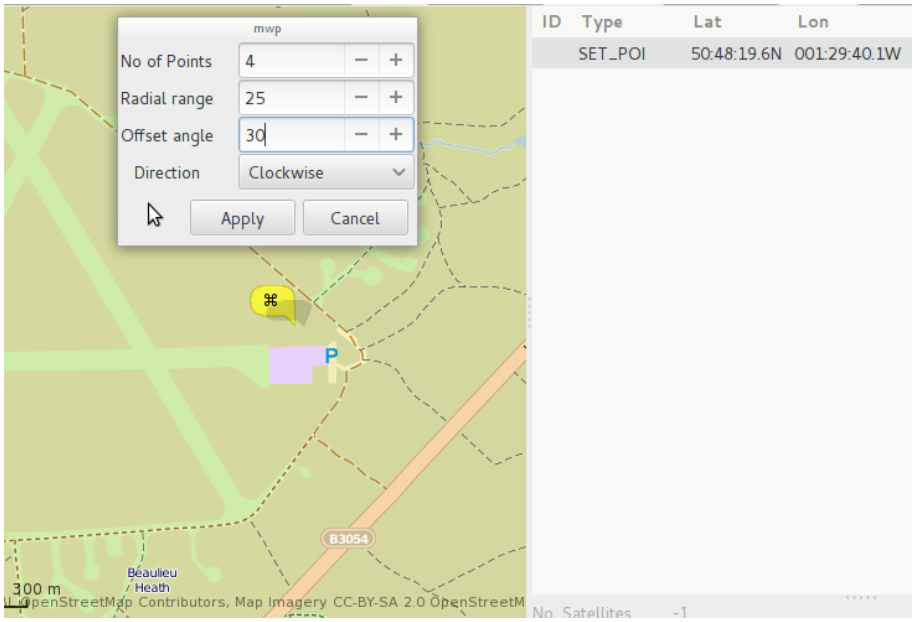


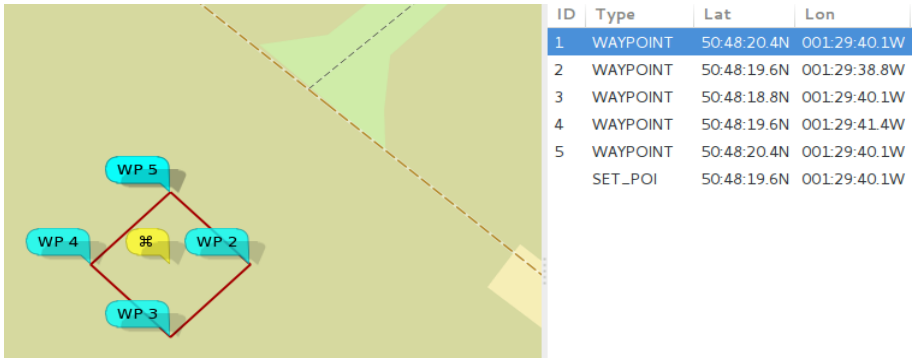
Illustration 8: Defining a shape around a single POI

Apply, with the result:

ID	Type	Lat	Lon
1	WAYPOINT	50:48:20.3N	001:29:39.5W
2	WAYPOINT	50:48:19.2N	001:29:39.0W
3	WAYPOINT	50:48:18.9N	001:29:40.8W
4	WAYPOINT	50:48:20.0N	001:29:41.2W
5	WAYPOINT	50:48:20.3N	001:29:39.5W
	SET_POI	50:48:19.6N	001:29:40.1W

Illustration 9: Inserted shape points

Noting that points 1 and 5 (start and end) are coincident, If you didn't expect to see this geometry, consider what happens with no offset:



ID	Type	Lat	Lon
1	WAYPOINT	50:48:20.4N	001:29:40.1W
2	WAYPOINT	50:48:19.6N	001:29:38.8W
3	WAYPOINT	50:48:18.8N	001:29:40.1W
4	WAYPOINT	50:48:19.6N	001:29:41.4W
5	WAYPOINT	50:48:20.4N	001:29:40.1W
	SET_POI	50:48:19.6N	001:29:40.1W

Illustration 10: Square with no offset

Here we start with the first (and last) points immediately North of the POI; thus, in Illustration 9 we have the same shape rotated by 30°, as specified. If you wanted the lines to be horizontal / vertical, specify an offset of 45° for a square. Shape points are appended to any extant mission points, and the shape tool may be invoked multiple times, for example to create 'concentric' circles;

- Delta updates. This option allows you to make bulk (delta, positive or negative) changes to all positions and / or altitudes:

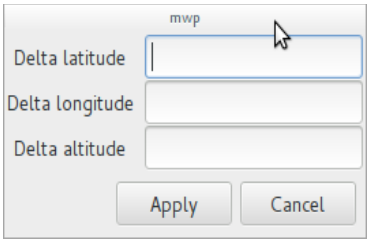


Illustration 11: Delta changes dialogue

- Clear Mission. The Clear Mission option clears the mission. There is no confirmation and no reversion, so be sure you want to do this.

Preferences and Settings

Preferences and default settings are stored using GTK+3 gsettings / dconf (on Linux / other free Unix like platforms, and whatever OS mechanism this maps onto for other operating systems). The gsettings schema is org.mwptools.planner. The following settings are stored.

Key	Usage	Default	Prefs
atexit	A command run on exit (e.g. to restore screensaver or powersaving settings). e.g. gsettings set org.gnome.settings-daemon.plugins.power idle-dim true	(none)	No
atstart	A command run on start (e.g. to override screensaver or powersaving settings). e.g. gsettings set org.gnome.settings-daemon.plugins.power idle-dim false	(none)	No
audio-bearing-is-reciprocal	Whether the *audio* reported bearing is the reciprocal (i.e. bearing from home to machine, rather than from machine to home.	false	No
audio-on-arm	Whether audio messages are automatically enabled on arm and disabled on disarm.	true	No
auto-follow	set auto-follow (GPS position) on start	true	Yes
auto-restore-mission	Whether to automatically download a mission from the FC when mwp does not have a mission loaded on the map, but one is loaded in the FC.	False	No
baudrate	The default baud rate for serial devices.	115200	No
compat-version	The WinGui version stored in mission files. Note that no validation is done dependent on the compat-version value, as any WinGui version specific variations are not known.	2.3-pre8	No
default-altitude	The default altitude for way points	20 (metres)	Yes
default-latitude	The default latitude for the map	50.909528	Yes
default-layout	The default dock layout loaded at startup, unless modified by the -l option. If undefined, .layout is used. See the section “Using the Dock“ for more details.	(none)	No
default-loiter	The default loiter time for POSWAIT_TIME way points (MW only)	30 (seconds)	Yes
default-longitude	The default longitude for the map	-1.532936	Yes
default-map	The default map to be used. If not defined, the default from the underlying toolkit (libchamplain, OSM Mapnik) will be used.	(none)	Yes

Key	Usage	Default	Prefs
default-nav-speed	The default navigation speed (used for timings only, does not affect FC settings)	2.5 (m/s)	Yes
default-zoom	Default map zoom	15	
device-names	A list of serial devices that will be presented in the Connect combo box.	'/dev/3dr', '/dev/rfcomm1', '/dev/mega2560', '/dev/ttyACM0@115200', '/dev/pts/0'	Yes
display-distance	Index into display distance units (0=metres, 1=feet, 2=yards)	0	Yes
display-dms	Display positions as degree:minute:seconds (vice decimal degrees).	false	Yes
display-speed	Index into display speed units (0=metres/sec, 1=kilometer/hour, 2=miles/hour, 3=knots)	0	Yes
espeak-voice	Default espeak voice (see espeak documentation)	en	No
fctype	Flight controller type	auto	No
gpsintvl	Time in ms before GPS is considered 'no signal'	2000	No
heartbeat	A command that is invoked every minute, e.g. to defeat an over zealous screen saver e.g. xscreensaver-command -deactivate	(none)	No
ignore-nm	Set to true to always ignore NetworkManager status (may slow down startup)	false	No
led	Activity LED colour. Well known (e.g. 'green') string or #RRGGBB.	#60ff00	No
log-on-arm	Whether logging is automatically enabled on arm and disabled on disarm.	false	No
log-path	Directory for log files (for replay)	(none)	No
log-save-path	Directory for log files (for saving)	(none) == current directory	No
map-sources	User defined map sources. You can add additional map sources beyond the default sources in libchamplain. The value is the name of file defining such sources. This is further explained in the Map Sources section of this document.	(none)	No
mavph, mavrth	As mavlink does not report status changes, it is necessary to define the rc channel and min / max values for PH and RTH, if you wish to have cute icons displayed for these events. The values are channel, min, max, separated by a colon, e..g. 8:1300:1700.	(none)	No

Key	Usage	Default	Prefs
max-home-delta	Maximum variation of home position without alert (m)	2.5	No
media-player	The media player used to play the low battery alert. If you don't want an audible alert, set this to 'false'. The default is paplay, but any media player capable of playing vorbis files (bleet.ogg) may be used (e.g. mplayer). If the media-player is blank (or cannot be found), then an internal gstreamer implementation is used.	paplay	No
mission-path	Directory for mission files	(none)	No
poll-timeout	Timeout in milliseconds for telemetry poll messages	500	No
rings-colour	Range Ring colour. Well known string or #RRGGBBAA	#ffffff20	No
rth-autoland	Automatically assert land on RTH waypoints, regardless of mission file setting	false	Yes
set-head-is-b0rken	Whether the set head bearing is the reciprocal (i.e. [old] bug in mw nav)	false	No
speak-interval	The interval between audio messages. A value of 0 disables audio messages, if non-zero, must be 15 or greater (seconds).	15	Yes
speech-api	espeak or speechd. Only change this is you know you have the required development files at build time	espeak	No
uilang	"en" do everything as English (decimal points, voice), "ev" do voice as english (so say 'point' for decimals, even if the locale separator is ",")	(none)	No
vlevels	The cell voltages at which colour transitions occur. A ';' separated list. An empty list uses the defaults "3.7;3.57;3.47;3.0".	(none)	No
require-telemetry	If set, mwp will check that FEATURE TELEMTRY is set on the flight controller, and warn the operator if not set.	false	No
flash-warn	If non-zero, mwp will check the free space on the FC's flash chip for Blackbox logs, and warn the operator the flash is full beyond the set percentage. The default of 0 means no check.	0 (percent)	No
forward	Telemetry data that is forwarded to an external device specified by –forward-to=DEVICE.	minLTM	No

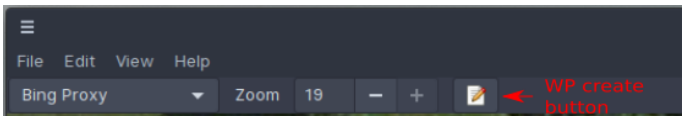
Key	Usage	Default	Prefs
auto-wp-edit	Whether clicking on the map automatically creates a new way point. The default is false, which maintains legacy implementation (clicking automatically creates WPs). If set to true, a switch is placed on the above the map, which must be toggled to allow WP creation. 	false	No
wp-text-style	The text style for WP numbers displayed during mission execution (or mwp log replay), Font, size / colour (including opacity, as RGBA).	Sans 144/#ff000080	No

Table 1: gsettings used by mwp

As well as the shell `gsettings` command, the `dconf-editor` command provides a UI tool to edit the above settings. The settings so marked as also available in the Preference dialogue.

The preferences dialogue is invoked from the menu Edit/Preferences item, and appears as Illustration 12:

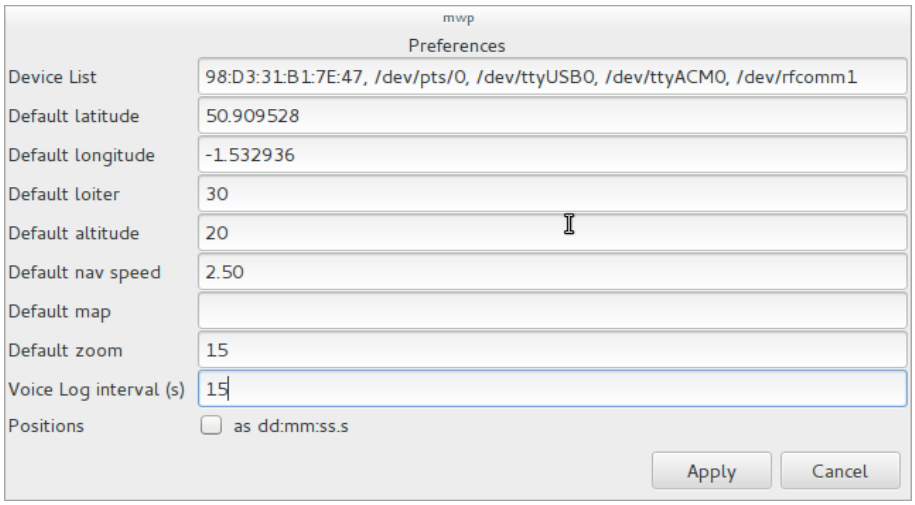


Illustration 12: **mwp** preferences dialogue

The screen shot shows using a Bluetooth address; this option is supported for Linux (only). A proper device node (e.g. /dev/rfcomm1 on Linux) works on all operating systems.

Other menu options

- File / Save. Saves the current mission to an XML fiile.
- File / Upload. Uploads the current mission to the flight controller (where connected and the FC supports sufficient capability). Once uploaded, the mission is then downloaded and compared to the source mission. You are warned if there is a mismatch, and the state of the FC is undefined if a mismatch occurs;
- File / Download. Downloads the mission from the flight controller, replacing (without warning, you selected to do this), the mission in the application;

- Equivalent options of iNav save / restore mission to / from eeprom;
- File / Replay Log File. Replays a log file at 'wall clock' speed;
- File / Load log File. Loads a log file, disregarding 'wall clock' time (i.e. as quickly as reasonably possible);
- File / Replay Blackbox Log File. Replays a log file at 'wall clock' speed;
- File / Load Blackbox Log File. Loads a log file, disregarding 'wall clock' time (i.e. as quickly as reasonably possible);
- File / Quit. Quits the application (without warning .... you get the modus operandi here);
- Edit / Nav Config. Where connected to a suitably capable flight controller, displays a nav config window (the values are now editable and may be applied to the flight controller by clicking 'Apply'). Currently on supported for mw-nav and iNav. The window differs for each FC (mw-nav shown below);

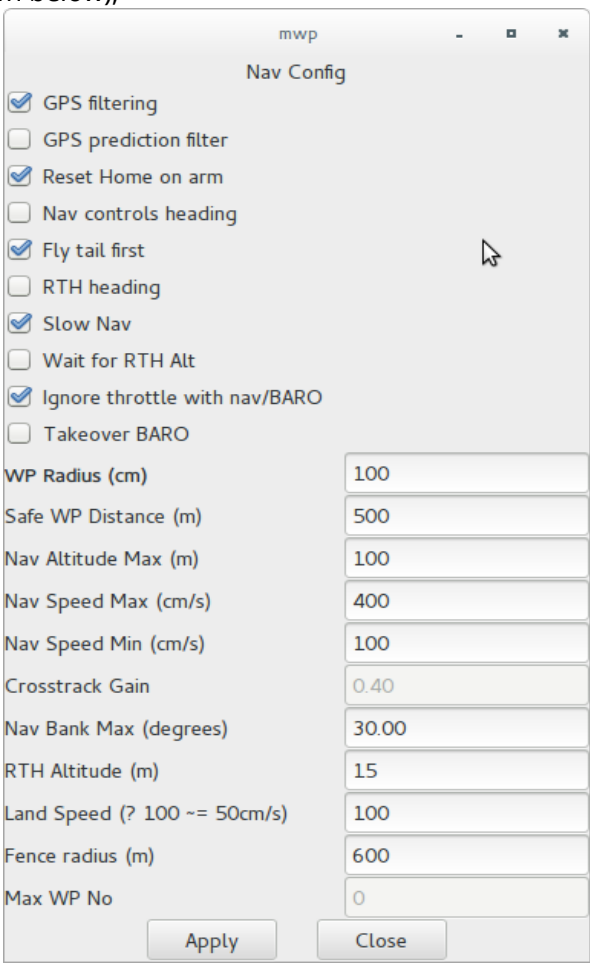


Illustration 13: Nav Config dialogue

- Edit / Set FC Mode. Allows you to set the flight controller type, if auto detection failed or you selected the wrong one;
- View / Centre on Position. Centre the map on a user defined location;
- View / Mission Tote. Forces display of the mission tote in a dockable window;
- View / Nav Status. Where connected to a suitably capable flight controller, displays a nav

status dock window;

- View / GPS Status. Forces display of the GPS status a dockable window;
- View / Voltage in dock. Forces display of the battery voltage (where monitored) in a dockable window;
- View / Radio Status. If you have a 3DR radio for telemetry, this option enables the display of 3DR statistics, otherwise it displays TX RSSI.
- Help / About. Displays a somewhat uninformative 'about' dialogue (however, my wife likes pink, and I like multi-rotors).

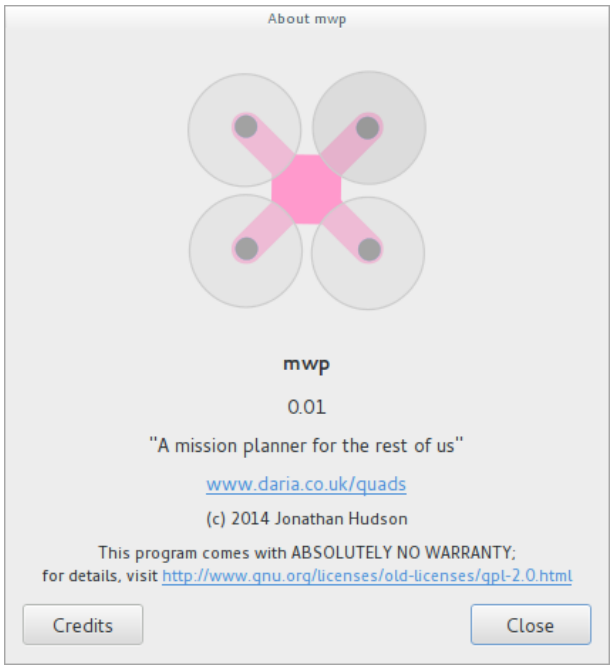


Illustration 14: Uninformative "About" dialogue

The screen shot below shows a configured dock, with multiple dock display options enabled:

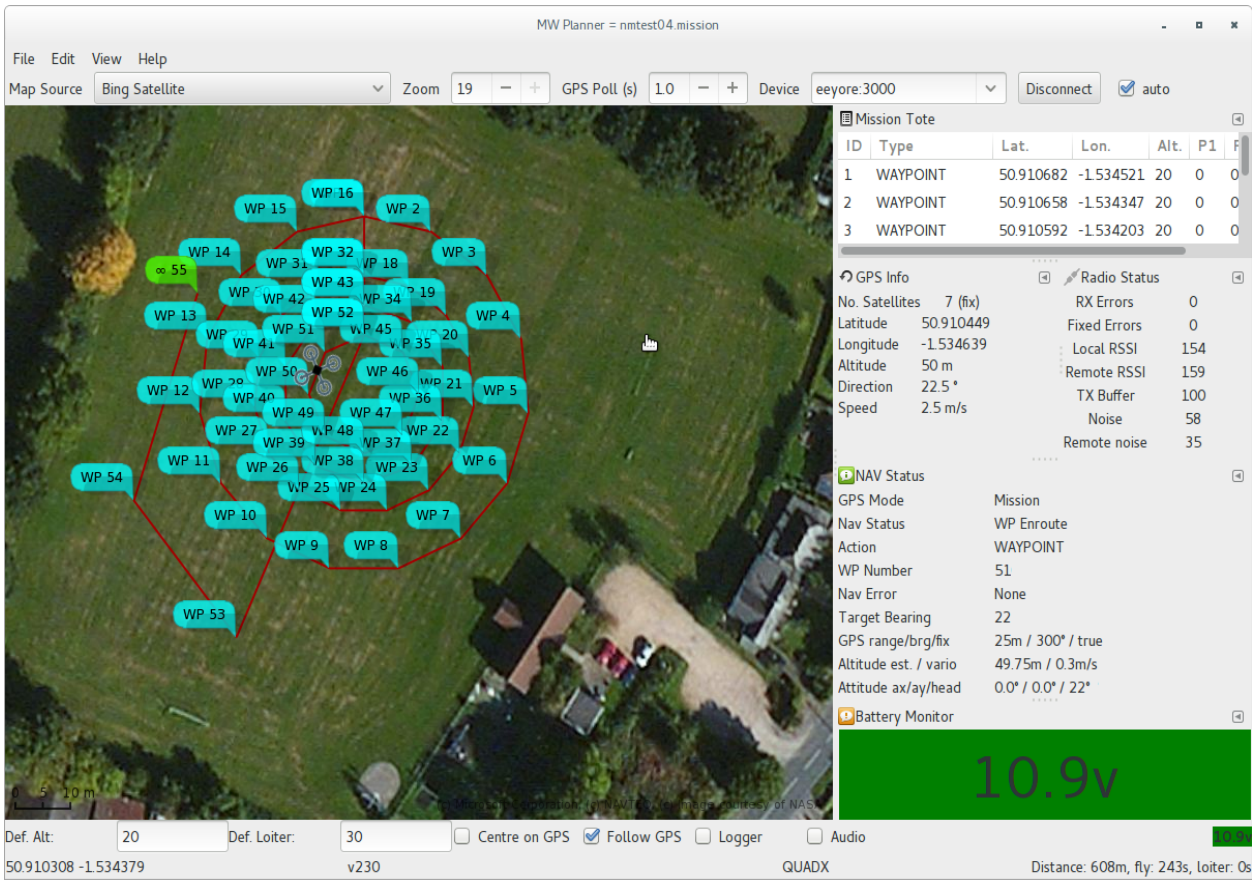


Illustration 15: Fully configured dock showing nav (mw) status

Logging

The screen-shots so far have been taken prior to the addition of the logging capability. As of v0.01, **mwp** supports logging. Logging is started / stopped by the Logger checkbox:

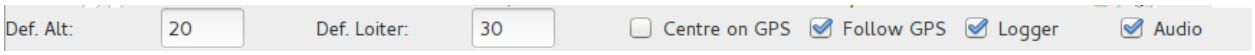


Illustration 16: Logging option

When logging is invoked, a log file is created in the current working directory:

```
mwp_YYYY-MM-DD_HHmmSS.log
```

where YYYY is the year, MM month, DD day, HH hour, mm minutes SS seconds in the local time zone.

The file format is “streamed json”, i.e. a json document for each logged event, for example (in this example, the vehicle is sitting on a window sill in my house, at the time there was no voltage monitoring on the vehicle, nor any mission loaded):

```
{"type":"status","utime":1391973539,"gps_mode":0,"nav_mode":0,"action":0,
 "wp_number":0,"nav_error":10,"target_bearing":0}
{"type":"comp_gps","utime":1391973539,"bearing":0,"range":0,"update":1}
{"type":"altitude","utime":1391973539,"estalt":1.03,"vario":0}
{"type":"attitude","utime":1391973539,"angx":-0.4,"angy":-1.3,"heading":273}
{"type":"analog","utime":1391973539,"voltage":0,"power":0,"rssi":0,"amps":0}
{"type":"raw_gps","utime":1391973539,"lat":50.9150417,"lon":-1.5304901,
 "cse":21.1,"spd":0.13,"alt":21,"fix":1,"numsat":7}
```

For each entry, there is a type field describing the data that follows, and “Unix” timestamp (utime, time_t), followed by type specific data.

The script (in the samples) **mwp-log2gpx.rb** is a Ruby language script that converts the log to a GPX file. **mwp-log2kml.rb** converts the log file to KML. Other scripts are provided to import the data into an SQL database (making it easy to find the maximum range, altitude etc), and to export from an SQL database to gpx and kml, using an 'armed' filter. SQL support is provided using the Sequel ORM, and databases should be defined using Sequel URIs. As of mid 2015, these sample scripts probably need updating.

Audio Prompts

Audio prompts are enabled from the options bar (Illustration 16). In order for audio prompts to be made, it is necessary that the flight controller is connected, and the preferences' “Voice Log Interval” value is 15 or more (seconds).

Using the Dock

mwp uses the GNOME Docking Library (gdl) to provide a dock capability. Items in the dock may be hidden, iconified or torn off into a separate window (that may then be returned to the dock). The section explains how use gdl in **mwp**. There is a short video illustrating dock actions at <https://vimeo.com/147958984>.

If you exit the application using the File / Quit menu option, dock setting are saved. If you exit by closing the window, dock settings are not saved.

Note that when **mwp** is updated and **the dock is expanded by adding new dock items**, *any previously saved dock layouts are invalidated, and you will have to manually recreate them*.

When **mwp** is first started, the dock contains the mission tote and two iconified items (the GPS Status and Nav Status windows [highlighted in red]).

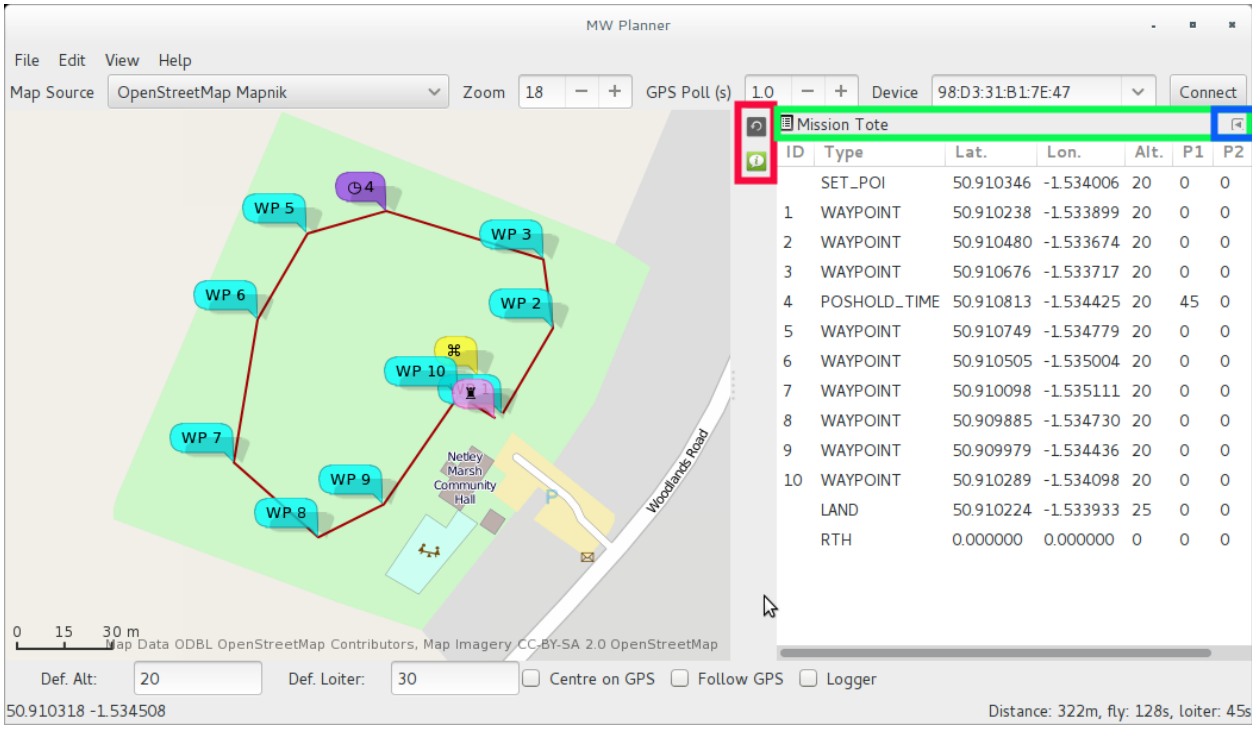


Illustration 17: Initial Dock display

In Illustration 17, three areas of the dock are highlighted:

- In red, the dock icons. Clicking on these will restore the window (either to the dock, or as a

separate window);

- In green, the dock tab bar. Where multiple items are in the dock, the tab icon may be dragged to reposition the docked window. In also has a pop-up menu, that allows the item to be completely hidden (but recoverable from the View menu), and;
- In blue, a iconify widget that will add the item to dock icon bar (the bit in red).

If the item tab bar icon (left-most in the green area) is dragged from the dock, the item will appear as a separate window.

By default, the battery monitor does not appear in the dock (but may be added from the View menu). In Illustration 18, the battery monitor has been added to the dock, and the mission tote has been moved to the dock icon bar.

The battery monitor window will resize the text to provide as visible as possible a voltage window. As for the all the dock windows, this window may be dragged on the desktop (and resized). In Illustration 18, the battery monitor window has been dragged from the dock and may be resized as an ordinary window. The detached window may be added to the dock by dragging the window's “tab bar” back into the dock, or added back to the dock icon bar using the iconify button (the left facing arrow to the right of the window's “tab bar”. If the detached window is closed, then it becomes hidden, and may be reattached to the dock (as an iconified dock item) from the View menu.

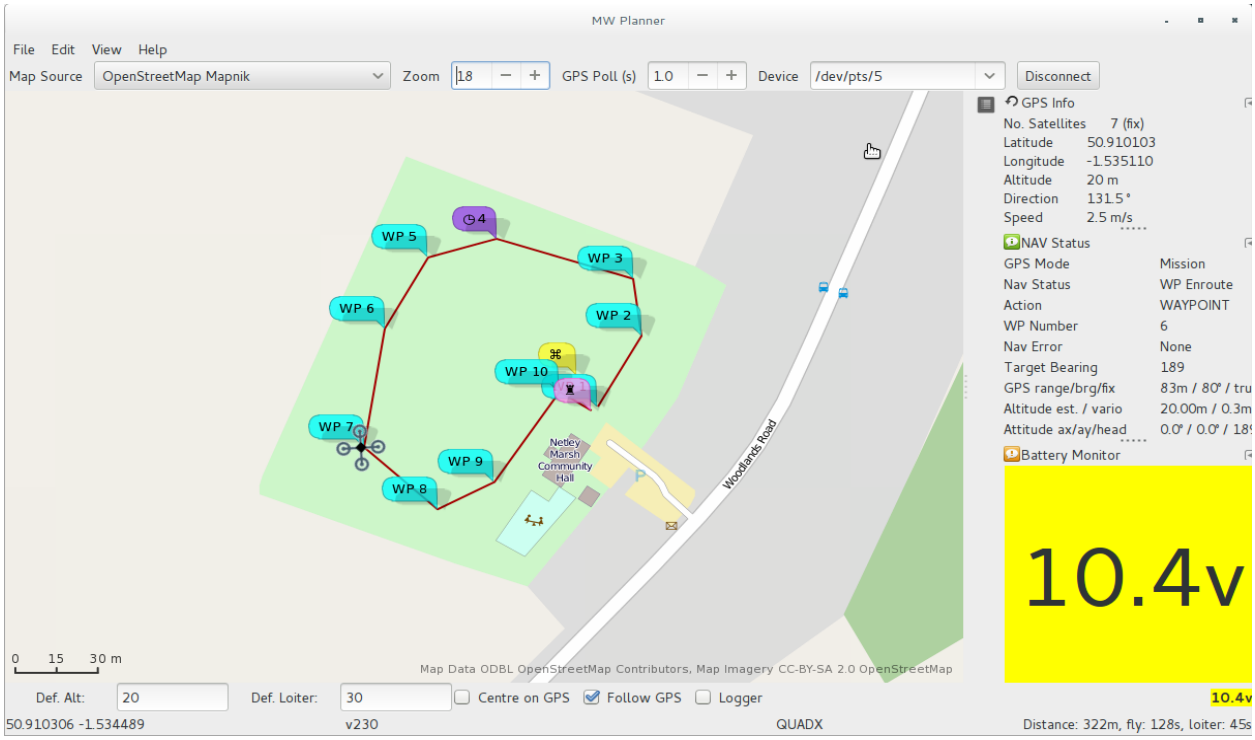


Illustration 18: Dock with battery monitor

The View menu contains items that can be added to the dock, and a Layouts menu. The Layouts menu offers options of 'Save' and 'Restore' in order to manage multiple dock arrangements.

- **Save** provides a text entry where you may enter a name for the saved layout. If the name exists, the contents are replaced, otherwise a new layout is saved. All layouts are saved under ~/.config/mwp as <NAME>.xml. There is a default of .layout (i.e.

~/.config/mwp/.layout.xml). It is deliberately difficult to remove this, but its content may be easily replaced.

- **Restore** provides a check-box list of existing layouts. Selecting one and clicking OK will cause that layout to be loaded into the dock.

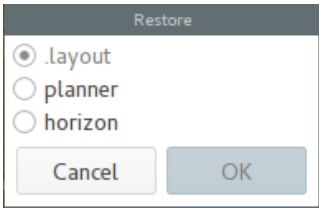


Illustration 19: Layout "Restore" dialogue

Available Dock Items

The available dock items are shown below:

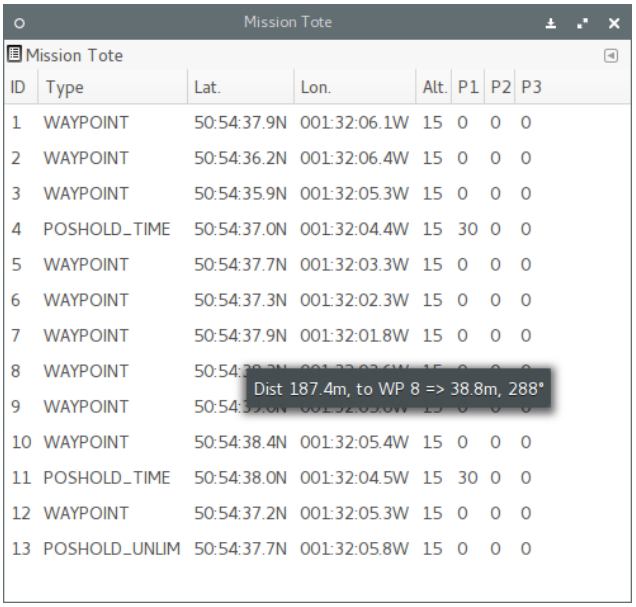


Illustration 20: Mission Tote (with tooltip shown)

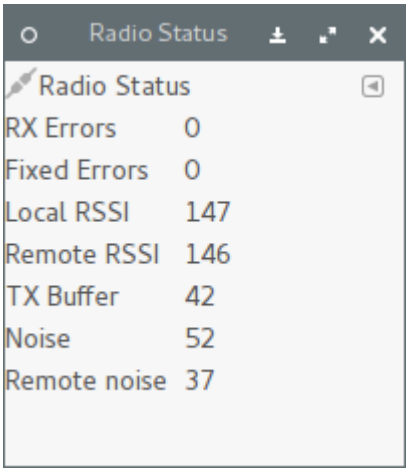


Illustration 21: Radio Status

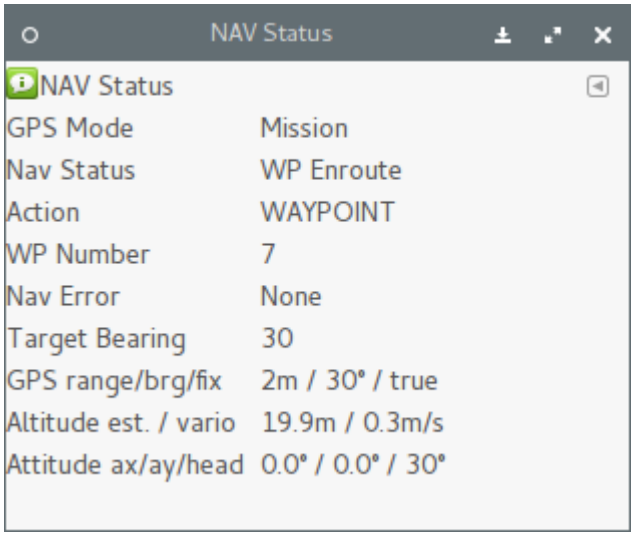


Illustration 22: Navigation Status

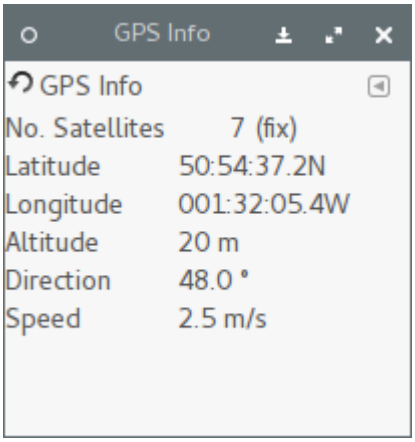


Illustration 23: GPS Information

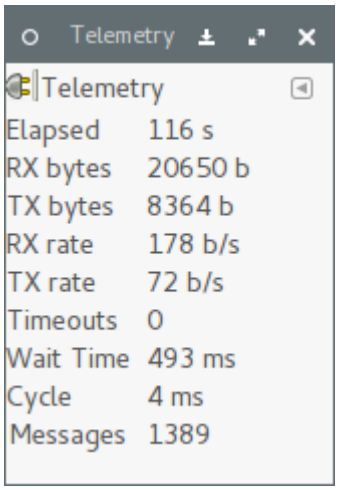


Illustration 24: Telemetry Status

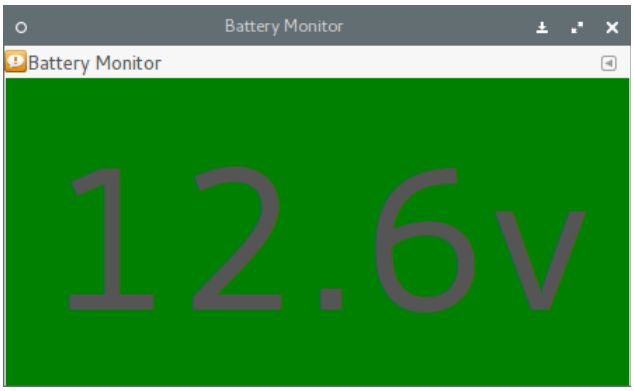


Illustration 25: Battery Status

Illustration 26: Artificial Horizon

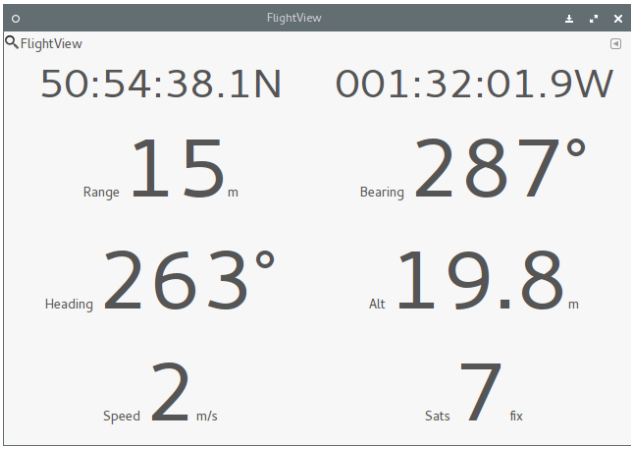


Illustration 27: Flight Information (large font)

Having read this far, it's possible that you are still confused; the dock is quite powerful but somewhat non-intuitive. There is a short video illustrating dock actions at <https://vimeo.com/147958984>.

The video shows:

- Load a mission into the mission tote;
- Load the Nav Status into the dock bar;
- Click the Nav Status icon to view nav status in the dock;
- Move the Nav status view into a window;
- Drag the Nav Status window back into the dock, selecting one of dock locations offered;
- Minimise the Nav Status back to the dock bar (the little arrow);
- Reopen the Nav Status into the dock;
- Hide the Nav Status;
- Restore the Nav Status as a dock icon;
- Reopen Nav Status in the dock.

Map Sources

As may have been mentioned in passing, **mwp** uses libchamplain for the map display. libchamplain, by default provides a set of free, unencumbered, open source map resources. The libchamplain developers take the terms and conditions of commercial mapping sources very (some might say, over) seriously and no commercial sources (Google, Bing etc.) are, by default, included, for fear of the user violating their somewhat ambiguous terms of service.

There are, however, other map sources that can be legitimately added to libchamplain (and hence

mwp). A mechanism is therefore provided to add such sources. By default, this will not permit you to abuse the terms of service of commercial providers, as only 'fixed URI' is ever supported.

As **mwp** has registered for a Bing API key, **mwp** appends a “Bing Proxy” to the list of map sources.

To use an additional map source in **mwp**:

- Define the “map-sources” key in the settings (see gsetting commands below), this is the name of a JSON file in `~/.config/mwp` defining the map sources (or rather under XDG_CONFIG_HOME on Unix like systems, and perhaps something to do with CSIDL_LOCAL_APPDATA on Microsoft OS).

A JSON file, `samples/sources.json` provides an example of the map sources format.

```
{
  "sources": [
    {
      "id": "historic-gb",
      "name": "GB Historic",
      "license": "(c) National Library of Scotland",
      "license_uri": "http://maps.nls.uk/projects/api/index.html",
      "min_zoom": 5,
      "max_zoom": 16,
      "tile_size": 256,
      "projection": "MERCATOR",
      "uri_format": "http://geo.nls.uk/mapdata2/os/seventh/#Z#/#X#/#TMSY#.png"
    },
    {
      "id": "localcache",
      "name": "Private cache",
      "license": "(c) OSM",
      "license_uri": "http://localhost/",
      "min_zoom": 0,
      "max_zoom": 17,
      "tile_size": 256,
      "projection": "MERCATOR",
      "uri_format":
"http://localhost/mapcache/tms/1.0.0/nsites@GoogleMapsCompatible/#Z#/#X#/#TMSY#.png"
    }
  ]
}
```

The latter only works (probably) chez-author. See the libchamplain documentation for map sources definitions (this documentation really exists).

It is not necessary to make any configuration changes other than those documented above..

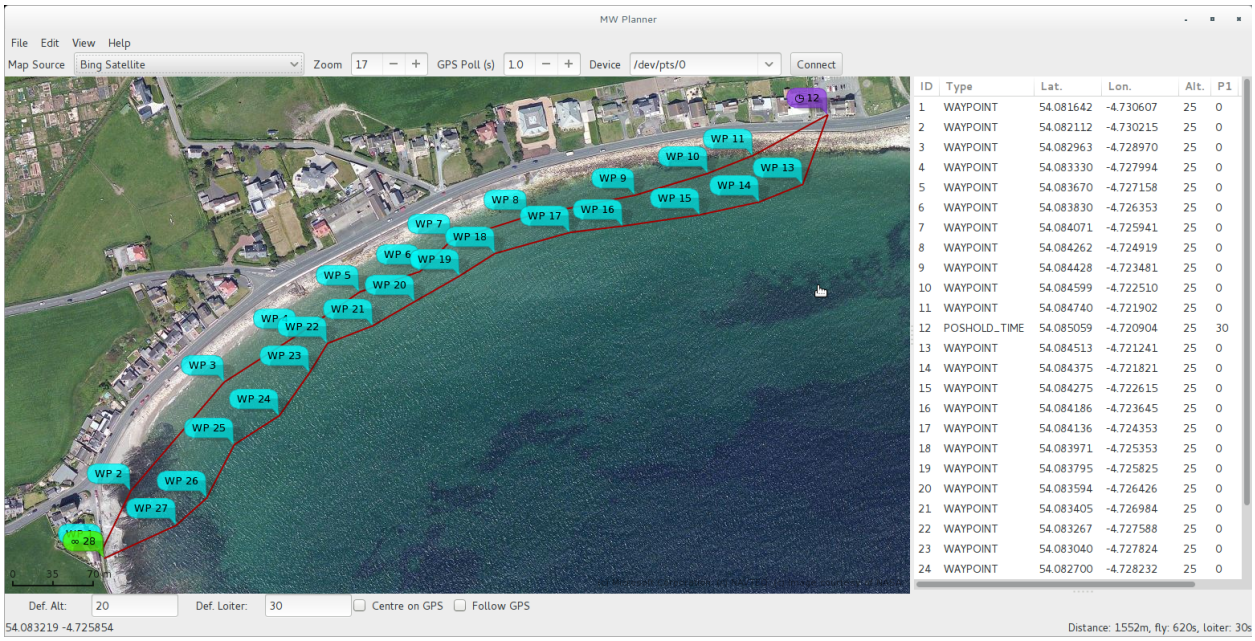


Illustration 28: Don't fly this when the tide is in (Bing maps)

Flash backs to the 1950s work too (the map data, running under the Xfce UI, Arch Linux on a Samsung ARM Chromebook):

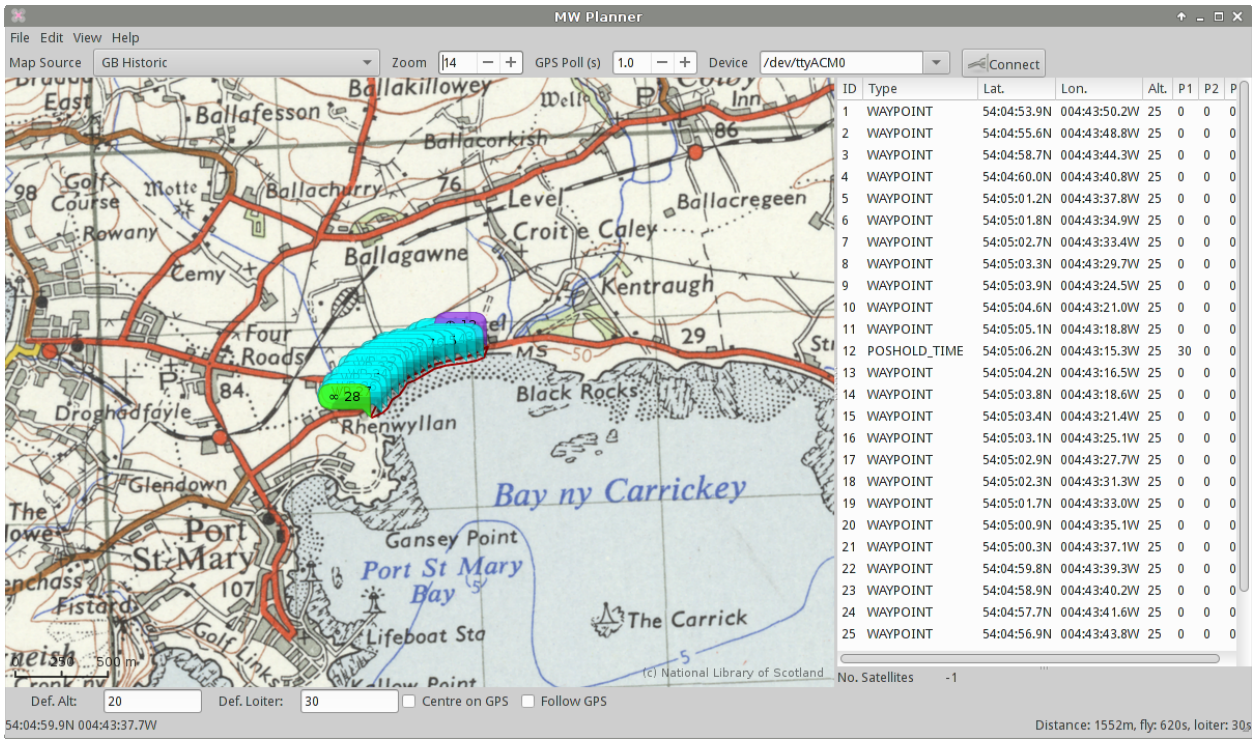


Illustration 29: “You can take the boy out of the island, but you can't take the island out of the boy”

Short Cut keys

In the map window, the following short cut keys (at least) may be used:

Key	Affect
Control +	Zoom map in
Control -	Zoom map out
F11	Toggle full screen
Control f	
Control c	Clear tracking information
Control t	Reset mission timer
Control m	Invoke mission loader
Control u	Upload mission
Control e	Upload mission and store in EEPROM
Control r	Restore Mission from EEPROM
Control v	Preview loaded mission
Control i	Initialise statistics
Control z	Zero flight information

Table 2: Short-cut keys

Installation

On Linux and FreeBSD, in general you should install the packages listed below (or better, by comparison of the dependency files for Arch, Fedora and Ubuntu in the documentation directory), then compile according to the build instructions below.

On Arch Linux, you can install from the AUR, using the aur helper of your choice, the package is called **mwptools-git**, this package will install dependencies, including the AUR **blackbox-tools-git** package.

For non-free / non-Unix like OS, you will have to work out your own installation, as the author is not familiar with such systems.

Building

In order to build **mwp** (and **mwptools**) on any platform a number of infrastructure open source components are required, at least:

- vala (0.30 or later recommended, may work with earlier), and gcc (or clang);
- GTK+-3.0 (and normal dependencies, 3.18 or later tested, 3.22+ preferred);
- libchamplain (0.12 or later);
- libchamplain dependencies (Clutter mainly);
- gdl (GNOME Docking library);
- gstreamer-1.0;
- libespeak or speech-dispatcher

For free OS, these dependencies should be easily satisfied; for propriety platforms, Clutter, gdl and libchampain may (or not, I don't know) be available. For Ubuntu, you will the "-dev" versions of the libraries. For Ubuntu and Fedora, the dependencies are listed in text files in the docs directory.

Minimum version for Ubuntu is 16.04, for Fedora, 26.

```
$ make
$ sudo make install # system install
```

mwptools is developed and tested on Arch Linux (ARM, x86_64, ia32); it is also tested on Ubuntu, Fedora and FreeBSD. As of 2016-01-01 it may compile under Cygwin (and once ran, to a large degree).

Replaying Blackbox logs

As of 2016-02-20-ish, **mwp** will replay blackbox logs directly from the File menu. In order for this to work, you need to have a working **replay_bbox_ltm.rb** (see the bbox-replay directory) on the PATH, and the dependencies discussed in replay_bbox_ltm.pdf must be addressed (which includes having **blackbox_decode** on the PATH as well).

Protocol Documentation

https://github.com/iNavFlight/inav/wiki/MSP-Navigation-Messages

https://github.com/iNavFlight/inav/wiki/Lightweight-Telemetry-(LTM)

Some (useful) information on iNav telemetry options:

https://github.com/iNavFlight/inav/wiki/iNavFlight-Missions

Mission File extensions

mwp writes a single extension to the WinGui XML mission file format, representing the zoom level and centre coordinates appropriate to the mission, for example:

```
  <mwp zoom="18" cx="-1.5347760915756226" cy="50.910293849231742"></mwp>
```

These extensions should be ignored by WinGui / ezgui.

Command line options

```
$  mwp --help
Usage:
  mwp [OPTION?]

  mwp local / 2017-12-27 0.358.592

Help Options:
  -h, --help                          Show help options

Application Options:
  -m, --mission=file-name             Mission file
  -s, --serial-device=device_name     Serial device
  -d, --device=device-name            Serial device
  -f, --flight-controller=fc-name     mw|mwnav|bf|cf
  -c, --connect                       connect to first device (does not set auto flag)
  -a, --auto-connect                  auto-connect to first device (sets auto flag)
  -N, --no-poll                       don't poll for nav info
  -T, --no-trail                      don't display GPS trail
  -r, --raw-log                       log raw serial data to file
  --ignore-sizing                     ignore minimum size constraint
```

```
  --full-screen                       open full screen
  --ignore-rotation                   ignore vehicle icon rotation on old libchamplain
  --dont-maximise                     don't maximise the window
  --force-mag                         force mag for vehicle direction
  --force-nav                         force nav capaable
  -l, --layout                        Layout name
  -t, --force-type=type-code_no       Model type
  -4, --force4                        Force ipv4
  -3, --ignore-3dr                    Ignore 3DR RSSI info
  -H, --centre-on-home                Centre on home
  --debug-flags                       Debug flags (mask)
  -p, --replay-mwp=file-name          replay mwp log file
  -b, --replay-bbox=file-name         replay bbox log file
  --centre=position                   Centre position
  --offline                           force offline proxy mode
  -S, --n-points=N                    Number of points shown in GPS trail
  -M, --mod-points=N                  Modulo points to show in GPS trail
  --rings=number,interval             Range rings (number, interval(m)), e.g. --rings 10,20
  --voice-command=comamnd string      External speech command
  -v, --version                       show version
  --wayland                           force wayland (if available)
  --really-really-run-as-root         no reason to ever use this
  --forward-to=device-name            forward telemetry to
  --perma-warn                        info dialogues never time out
```

Note that commonly used options may be stored in a configuration file:

```
$ cat ~/.config/mwp/cmdopts
# My default options for mwp (50 range rings @ 20m)
--rings 50,20
```

Platform Caveats

The application is untested on big endian CPUs.

Other Configuration Items

It is not currently my intention to address all the elements of MultiWii configuration. The cross-platform tools MultiWiiConf or iNav Configurator may be used for configuration of items not addressed by **mwptools**.

LTM Light (Telemetry)

mwp supports the LTMprotocol, with iNav extension.

https://github.com/iNavFlight/inav/wiki/Lightweight-Telemetry-(LTM)

Cf-cli

cf-cli is a command line tool to backup and restore Cleanflight settings. See README.md in the common/ directory.

Telemetry Devices

Mwp has been tested using the following devices to acquire telemetry data from the vehicle:

- Bluetooth. Works at all practical speeds, but has limited range (typically < 20m);
- 3DR radio. Works well at lower speeds and on soft serial. With LTM, 3DR at 9600bps will

provide 5Hz GPS and 10Hz attitude updates;

- HC-12 433Mhz radio: Like 3DR, only better;
- ESP8266 WiFi – serial device. With the 'transparent bridge', this can provide good range and high data rates; however to achieve best results, a WiFi hub will perform better than using a laptop as an access point;
- LoRA, with Bluetooth bridge. See the mwp wiki for details.

Having tried all these devices, the author uses HC-12 with a dedicated 433Mhz whip antenna. This provides reliable comms even over soft serial at 9600 bps.

With the Cleanflight softserial device configured to support both MSP and Telemetry (LTM or the Mavlink minimal implementation), **mwp** will automatically switch between inefficient polled MSP (unarmed) and efficient pushed telemetry (armed).

Author, copyright and licence

mwptools is written by Jonathan Hudson <jh+mwptools@daria.co.uk> (aka stronnag on the MW forum, rcgroups, #inavflight etc., but the NSA already knew that). It is licensed under the GPL 3 (or later, your choice).

This document describes the infinitately pre-alpha software version, the author appreciates that the manual is always hopelessly out-dated.

Annexe A – MultiWii Configuration

In order to use the full capability of mwptools, it is necessary to configure multiwii to have full GPS and navigation support. The following extract from multiwii's config.h illustrates the settings required. In particular, the highlighted item is essential; most of the others can be set in the Nav Config dialogue.

```
   //Enables the MSP_WP command set , which is used by WinGUI for displaying an setting up
navigation
   #define USE_MSP_WP

   // HOME position is reset at every arm, uncomment it to prohibit it (you can set home
position with GyroCalibration)
   //#define DONT_RESET_HOME_AT_ARM

/* GPS navigation can control the heading */

// copter faces toward the navigation point, maghold must be enabled for it
#define NAV_CONTROLS_HEADING       1    //(**)
// true - copter comes in with tail first
#define NAV_TAIL_FIRST             0    //(**)
// true - when copter arrives to home position it rotates it's head to takeoff direction
#define NAV_SET_TAKEOFF_HEADING    1    //(**)

/* Get your magnetic declination from here : http://magnetic-declination.com/
Convert the degree+minutes into decimal degree by ==> degree+minutes*(1/60)
Note the sign on declination it could be negative or positive (WEST or EAST)
Also note, that maqgnetic declination changes with time, so recheck your value every 3-6 months
*/
#define MAG_DECLINATION  -1.53f   //(**)

// Adds a forward predictive filterig to compensate gps lag. Code based on Jason Short's lead
filter implementation
#define GPS_LEAD_FILTER               //(**)

// add a 5 element moving average filter to GPS coordinates, helps eliminate gps noise but adds
latency comment out to disable
// use it with NMEA gps only
//#define GPS_FILTERING                 //(**)

// if we are within this distance to a waypoint then we consider it reached (distance is in cm)
#define GPS_WP_RADIUS              100      //(**)

// Safe WP distance, do not start mission if the first wp distance is larger than this number
(in meters)
// Also aborts mission if the next waypoint distance is more than this number
#define SAFE_WP_DISTANCE           500      //(**)

//Maximu allowable navigation altitude (in meters) automatic altitude control will not go above
this height
#define MAX_NAV_ALTITUDE           100     //(**)

// minimum speed when approach waypoint
#define NAV_SPEED_MIN              100    // cm/sec //(**)
// maximum speed to reach between waypoints
#define NAV_SPEED_MAX              400    // cm/sec //(**)
// Slow down to zero when reaching waypoint (same as NAV_SPEED_MIN = 0)
#define NAV_SLOW_NAV               0      //(**)
// Weight factor of the crosstrack error in navigation calculations (do not touch)
#define CROSSTRACK_GAIN            .4     //(**)
// Maximum allowable banking than navigation outputs
#define NAV_BANK_MAX 3000                 //(**)

//Defines the RTH altitude. 0 means keep current alt during RTH (in meters)
#define RTH_ALTITUDE               15        //(**)
//Wait to reach RTH alt before start moving to home (0-no, 1-yes)
#define WAIT_FOR_RTH_ALT           1         //(**)

//Navigation engine will takeover BARO mode control
```

```
#define NAV_TAKEOVER_BARO          1         //(**)

//Throttle stick input will be ignored  (only in BARO)
#define IGNORE_THROTTLE            1         //(**)

//If FENCE DISTANCE is larger than 0 then copter will switch to RTH when it farther from home
//than the defined number in meters
#define FENCE_DISTANCE      600

//This governs the descent speed during landing. 100 is equals approc 50cm/sec
#define LAND_SPEED          100
```