MWPTools

Tools for MultiWii NAV on free OS (and CF mission logging)

"Sweet dreams and flying machines"[1]

Overview

The MultiWii flight controller is currently (early 2014) undergoing some exciting development; namely the introduction of way point navigation. This impressive development has somewhat transformed the previously cross platform (albeit ugly Java) MultiWii experience, into effectively, a proprietary (Microsoft) operating system experience. Whilst I prefer cross platform solutions, I have some sympathy for the NAV author's choice, given the unpalatable alternative of updating the Java MultiWiiConf.

This project attempts to address the Windows dependency by providing tools for MultiWii-NAV mission planning on free operating systems. The principal application, **mwp** is a mission planner. Whilst it is unlikely that **mwp** can approach the functionality of WinGui[2], the aim is to at least enable users on free platforms to:

- Plan a mission using a geographic display;
- Edit missions, way point and actions;
- Upload the mission to a capable MultiWii-NAV flight controller;
- Download a mission from a capable MultiWii-NAV flight controller;
- Load and save missions to a file, in a format compatible with WinGui;
- Track the mission in real time on open source Multiwii compatible platforms, where suitable telemetry devices are available;
- Mission logging and log conversion to at least KML and GPX.

Other 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

There is not, to the best of the author's knowledge (as of early 2014), any protocol documentation for the changes to MSP to support MultiWii-NAV; the applications in this suite have been developed from knowledge gleaned from postings in the MultiWii forum, GPS NAV thread <http://www.multiwii.com/forum/viewtopic.php?f=8&t=3989>, and by reverse engineering of WinGui

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.

2 WinGui is an open source MultiWii management program on Microsoft proprietary operating systems by András Schäffer (EOSBandi). It may be downloaded from https://code.google.com/p/mw-wingui/. András also authored the NAV support in MultiWii (for which I and many other were extremely grateful, at least until r1717).

source code. Neither is an entirely satisfactory basis for development, compared to a proper specification, but it's the best available to date.

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

In addition, the applications support the functions of the Naze32 Cleanflight FC (mission logging (not planning), PID and switch configuration).

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.

Flight Controller variant

In MultiWii r1717, the MSP "capabilities" has been implemented in a somewhat unfortunate way, in that it overwrites capabilities previously defined for Naze32 Cleanflight flight controllers. As of 2014-10-17, it is almost impossible for **mwptools** to detect whether it is running on a nav capable Multiwii FC, or another FC that may offer other functions. Whilst there is some hope that common sense will later prevail (and the Cleanflight MSP_API_VERSION is a not a great help along this road), in order to protect the user from this, **mwptools** is cautious in detecting FC capabilities, and allows the user to define which FC is being used, as follows:

- A symbolic link may be made to the **mwp** and **switchedit** applications suffixing the FC model, thusly:
  - mwp-mw -> mwp # non-nav multiwill;
  - mwp-mwnav -> mwp # multiwii-nav;
  - mwp-cf -> mwp # cleanflight;
  - mwp-auto -> mwp # auto-dectect for mw and cleanflight (MSP_API_VERSION, post 2014-10-24);
  - and likewise for **switchedit** (swithedit-cf etc.).
- **mwp** may be invoked with –flight-controller=(mw|mwnav|cf|auto)
- There is a gsettings key, 'fctype', which can be set to mw/mwnav/cf/auto;
  - Otherwise, you get the FC type chooser dialogue:

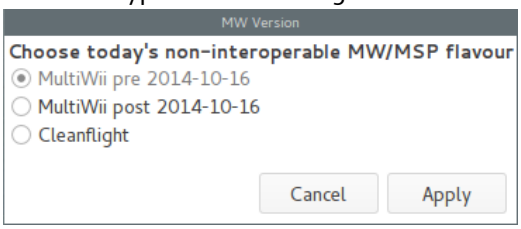

Illustration 1: Choose the flight controller variant

  - Selecting Cancel (at startup) causes the application to terminate instantly.
- The author will be delighted to remove this inconvenience should the various FC authors see sense and restores interoperability, or a more general agreement on FC identification is agreed by

the responsible parties.

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.

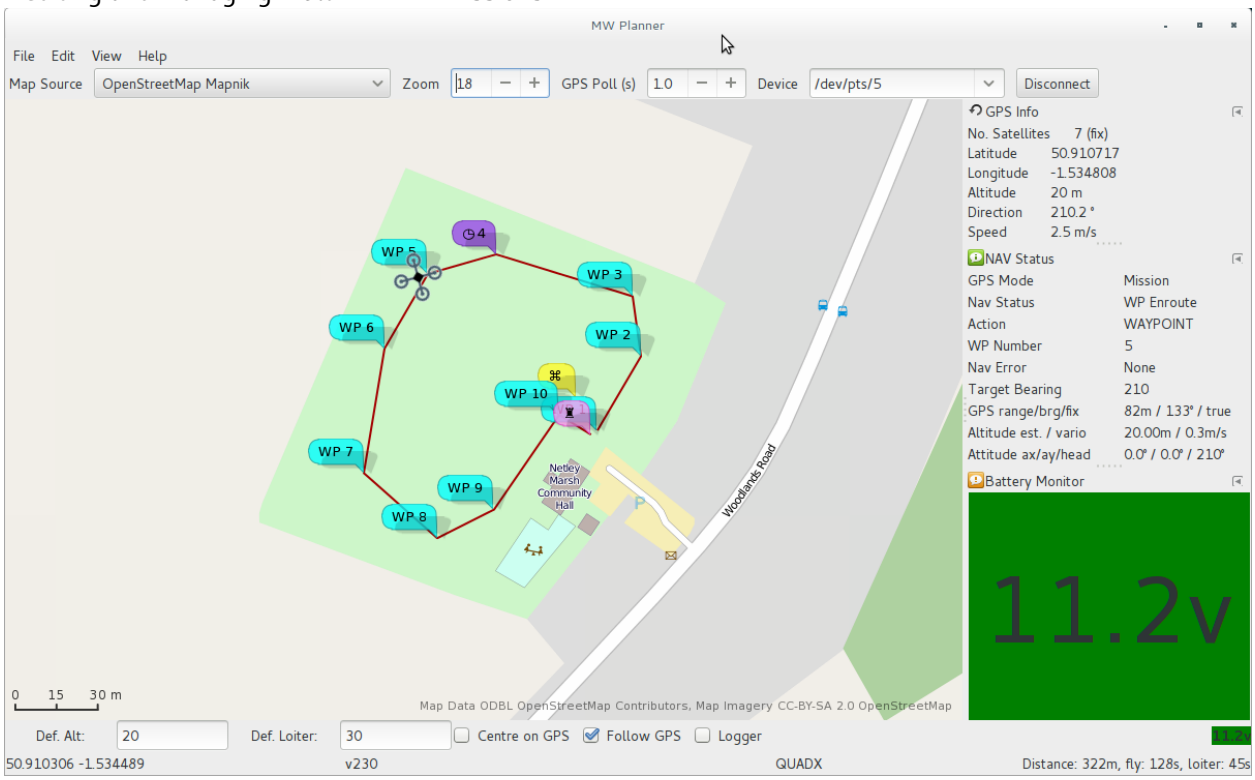

Illustration 2: **mwp** window showing a mission and mission tracking

Illustration 2 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 17);

- A status bar shows the current map centre position, the MultiWii version, the MultiWii 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.

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. 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.

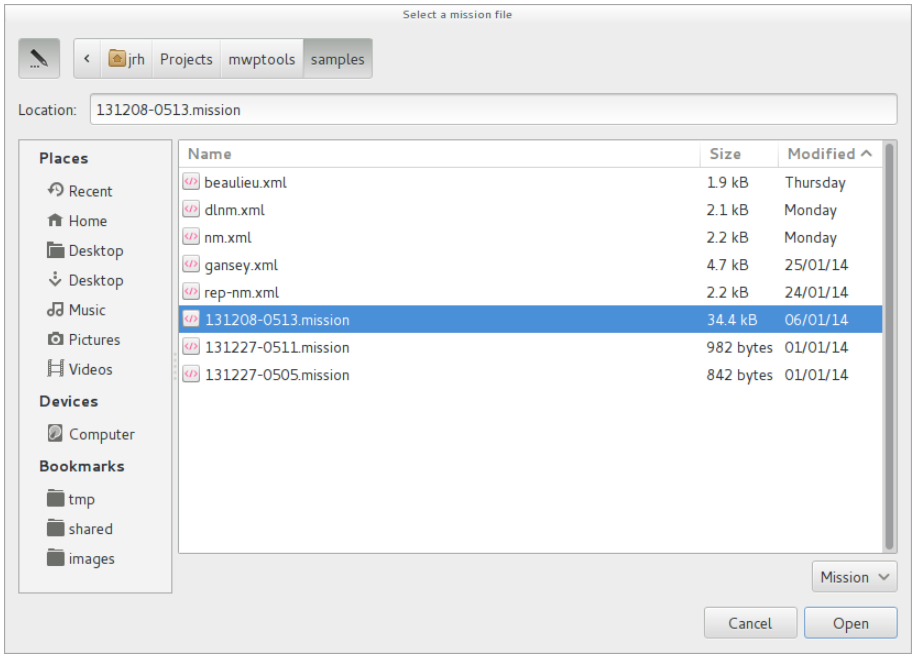

Illustration 3: Opening a mission file (EOSBandi / WinGui sample file)

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):

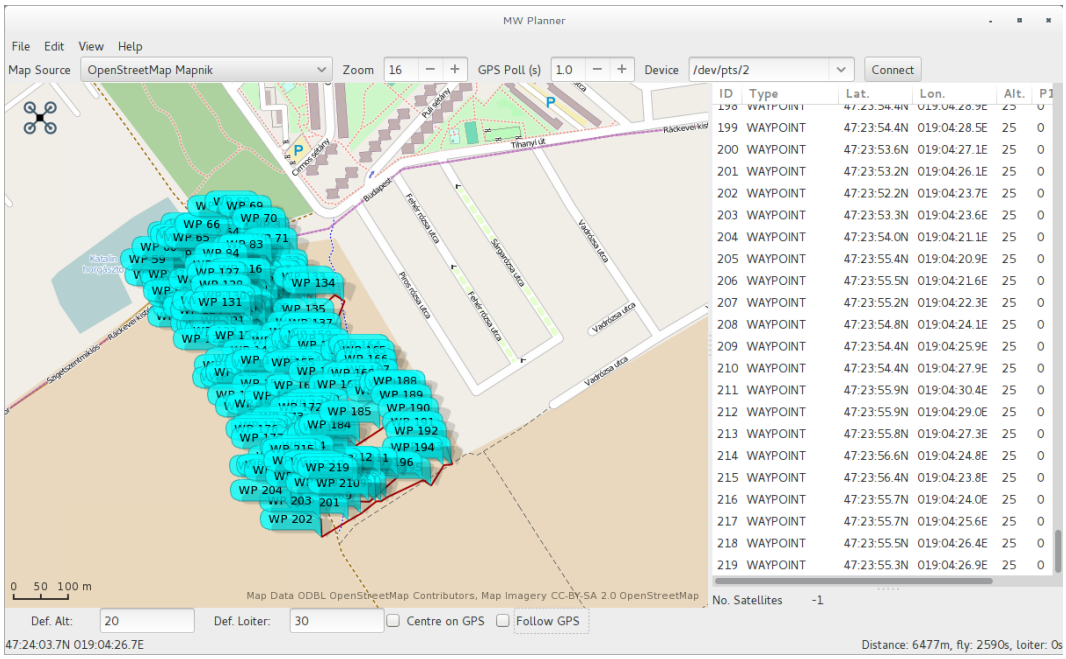

Illustration 4: Import of EOSBandi / WinGui sample file (don't fly this one)

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

1. Right mouse click on the map;
2. From the right mouse button pop-up menu in the mission tote.

- A defaults and settings bar. Here the default altitude and loiter times used for way point definition are shown / edited. There are also display options to follow the GPS, and centre on the GPS. The current battery voltage is also shown, colour-coded according to the battery status setting from the FC;

Opening Missions

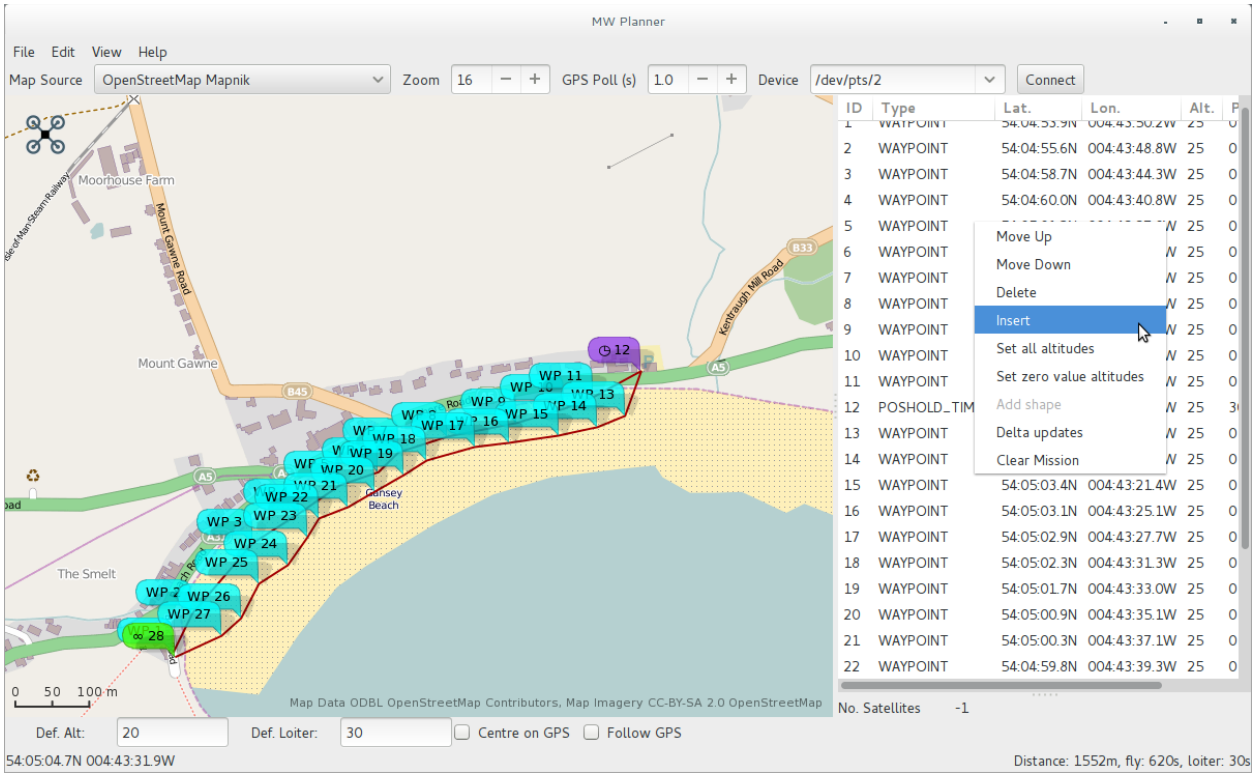

Illustration 5: Mission pop-up menu (let's go for a [quick] pint at "The Shore" hotel)

In Illustration 5, the mission pop-up menu has been invoked, and the “Insert“ action selected.

Creating and editing missions

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

Mission points are created either by a right click on the map, or by the mission pop-up menu, “Insert“ action (see Illustration 5). 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 6, and explained further below.

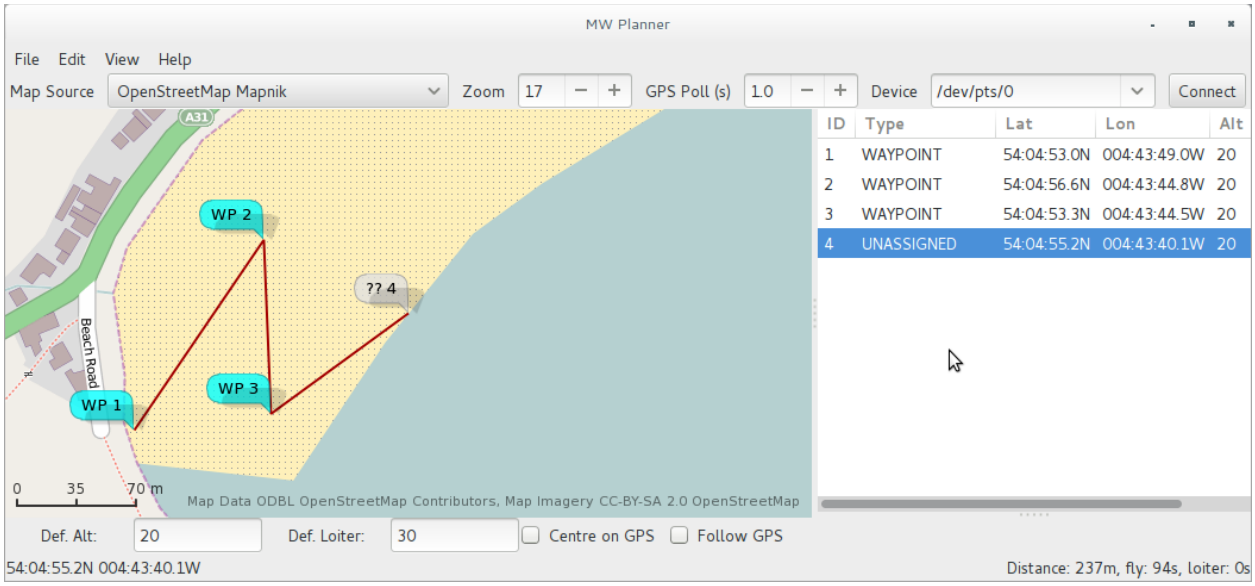

Illustration 6: Entering way points in **mwp**

In Illustration 6, way points 1,2, and 3 have been entered by right click on the map (as this 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 those 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 7). In that screen-shot (below), way point 7 is being dropped between way points 3 and 4.

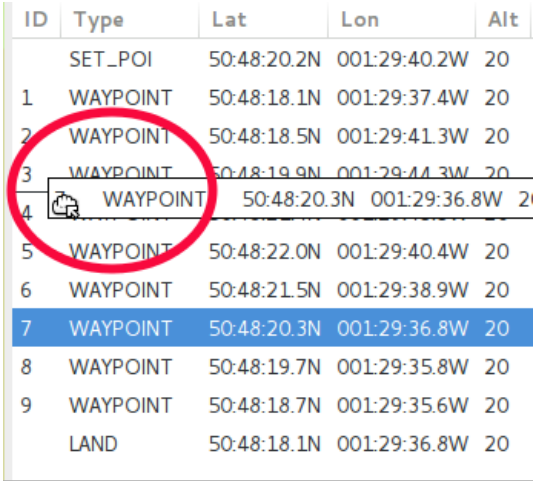

Illustration 7: Dragging a way point to re-order.
Note the 'grab' by the ID column

  - At the end of the drop, the list and markers on the map will be re-ordered.
- Type. The way point type may be selected from a drop down menu embedded in the "Type" column of the tote:

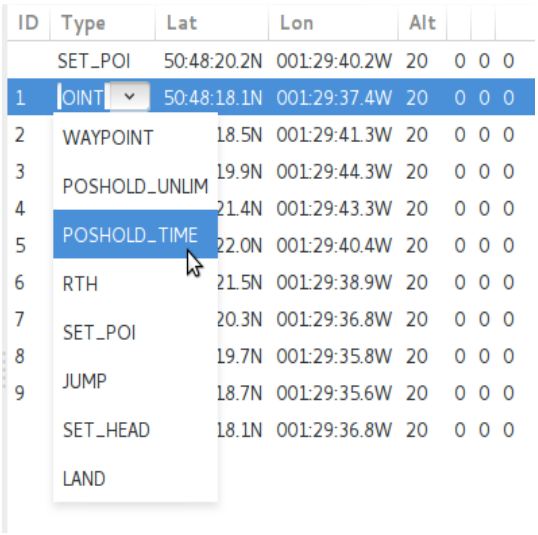

Illustration 8: Changing a way point type

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.

- 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);
- 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:

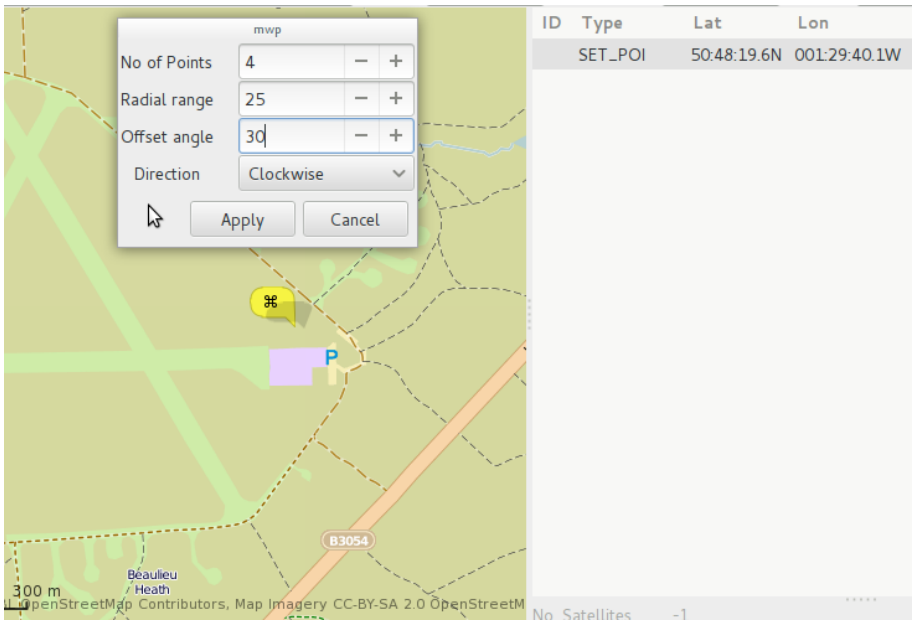

Illustration 9: Defining a shape around a single POI

Apply, with the result:

Illustration 10: Inserted shape points

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

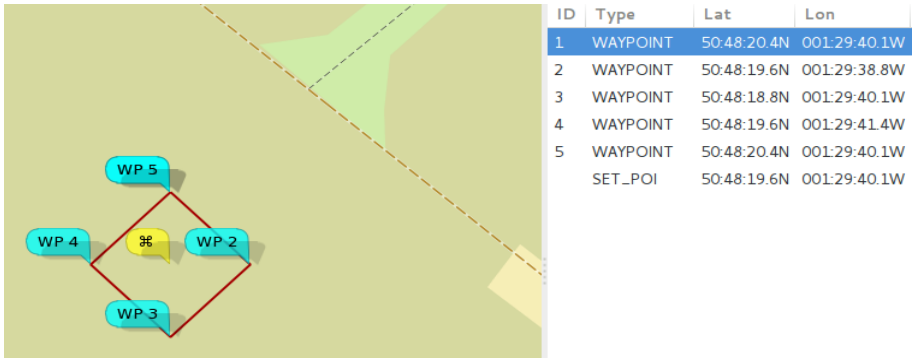

Illustration 11: Square with no offset

Here we start with the first (and last) points immediately North of the POI; thus, in Illustration 10 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:

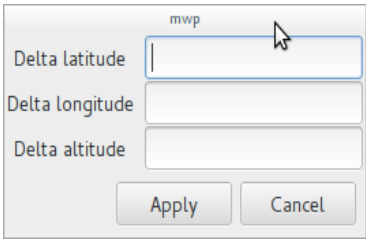

Illustration 12: Delta changes dialogue

- 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).	(none)	No
atstart	A command run on start (e.g. to override screensaver or powersaving settings).	(none)	No
audio-bearing-is-reciprocal	Whether the audio reported bearing is the reciprocal (i.e. bearing from home to quad, rather than from quad to home.	false	No
audio-on-arm	Whether audio messages are automatically enabled on arm and disabled on disarm.	false	No

Key	Usage	Default	Prefs
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-loiter	The default loiter time for POSWAIT_TIME way points	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
default-nav-speed	The default navigation speed (used for timings only, does not affect FC settings)	2.5 (m/s)	Yes
default-zoom	Default zoom level	15	Yes
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-dms	Display positions as degree:minute:seconds (vice decimal degrees).	false	Yes
espeak-voice	Default espeak voice (see espeak documentation).	default	No
heartbeat	A command that is invoked every minute, e.g. to defeat an over zealous screen saver ('xset -display :0 s reset').	(none)	No
log-on-arm	Whether logging is automatically enabled on arm and disabled on disarm.	false	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
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).	paplay	No
set-head-is-b0rken	Whether you consider mw-nav set head to work properly or not (i.e. as reciprocal, (the author has this set to true)). This may be fixed post MW pre 2.4 r1729, as a patch has been submitted.	false	No

Key	Usage	Default	Prefs
~~show-scary-warning~~	~~Defines whether a scary warning concerning the dangers of flying an undocumented protocol is displayed when you upload way points to the flight controller.~~ *This option is no longer used. As **mwp** has now been flight tested, there is no longer a "scary warning".*	~~true~~	~~No~~
speak-interval	The interval between audio messages. A value of 0 disables audio messages, if non-zero, must be 15 or greater (seconds).	0	Yes
fctype	Flight controller type (mw,mwnav,cf,auto)	(none)	No
update-interval	Update rate for telemetry polling	1 (second)	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".	''	No
quaduri			

Table 1: gsettings used by mwp

The preferences dialogue is invoked from the menu Edit/Preferences item, and appears as Illustration 13:

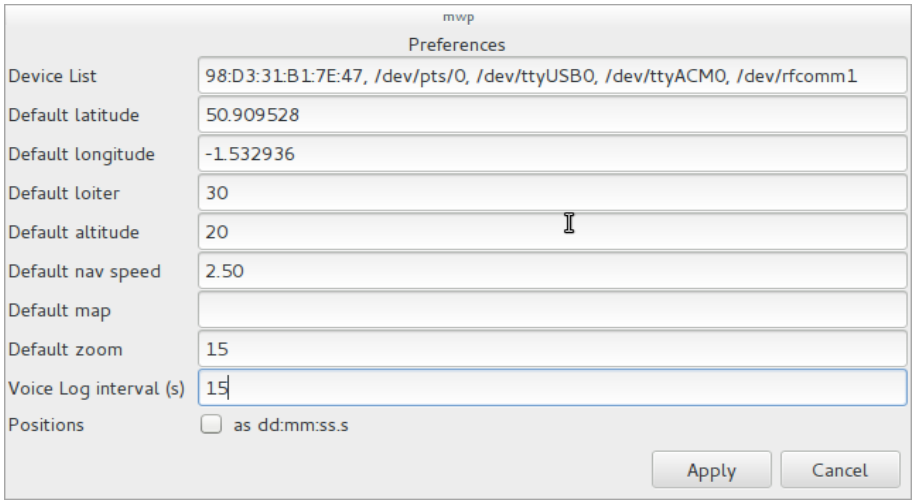

Illustration 13: **mwp** preferences dialogue

The screen shot shows using a Bluetooth address; this option was supported for Linux (only) in earlier versions; it is no longer supported. Use a proper device node (e.g. /dev/rfcomm1 on Linux) on all operating systems.

Other menu options

- File / Save. Saves the current mission to an XML file.
- 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;
- 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 / 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');

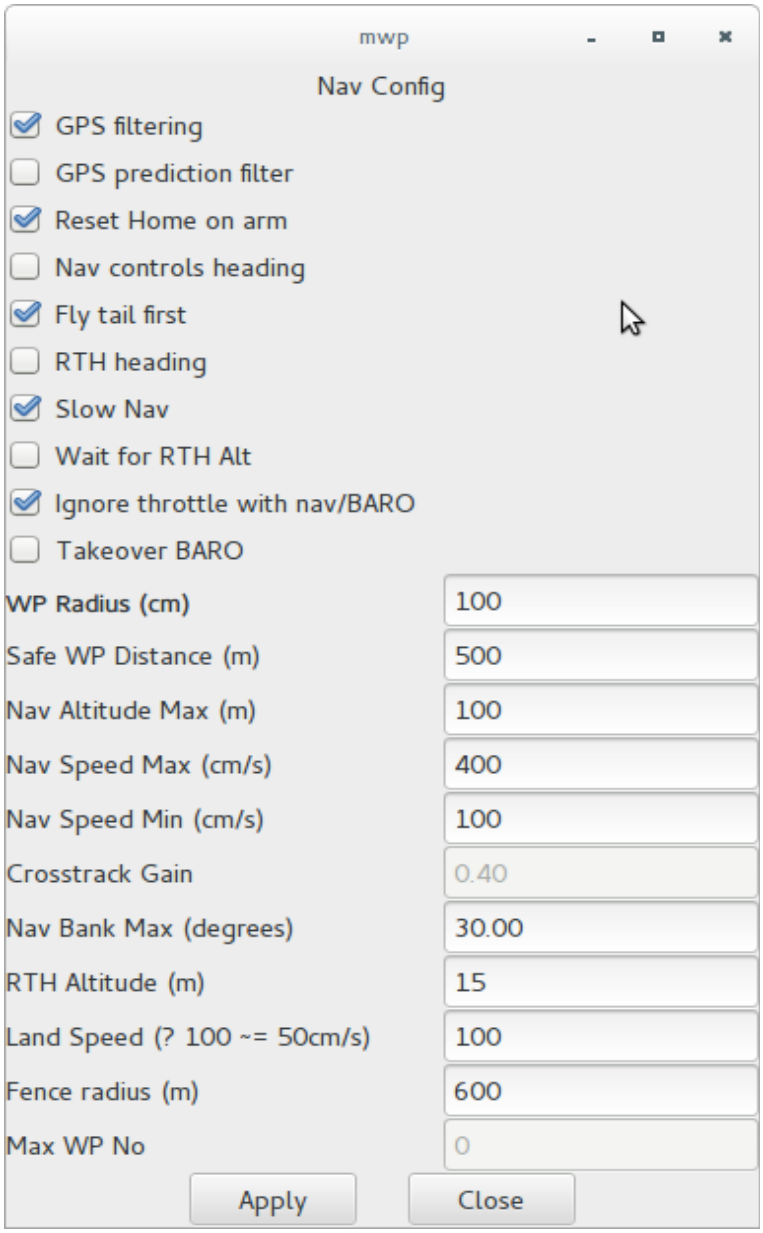

Illustration 14: Nav Config dialogue

- 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 / 3DR Radio Status. If you have a 3DR radio for telemetry, this option enables the display of 3DR statistics;
- Help / About. Displays a somewhat uninformative 'about' dialogue (however, my wife likes pink, and I like MW multi-rotors).

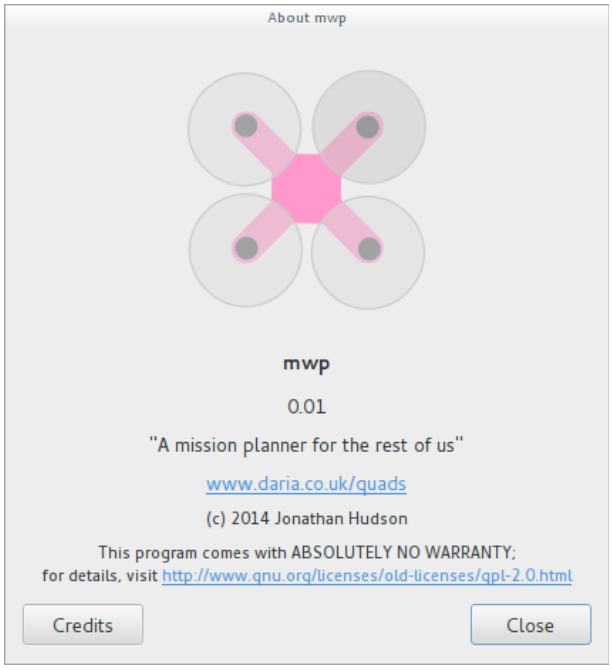

Illustration 15: Uninformative "About" dialogue

The screen shot below shows a fully loaded dock, with all display options enabled:

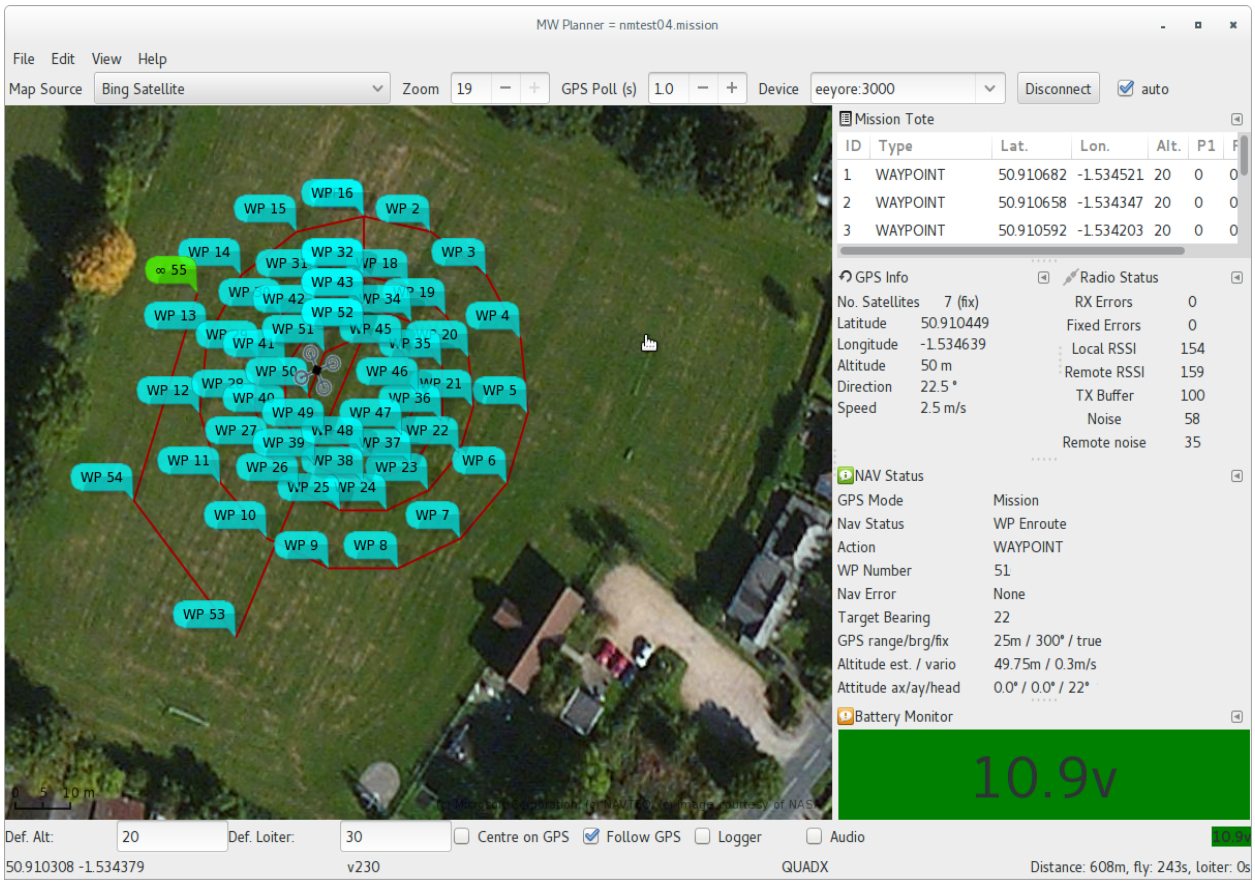

Illustration 16: Fully configured dock showing flight status

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 17: 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.

Audio Prompts

Audio prompts are enabled from the options bar (Illustration 17). 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**.

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: *When the dock is updated in the code (e.g. when the Telemetry Statistics item was added (2014-12-16)), then the saved layout becomes invalid, and the most likely outcome of trying to add a new dock item (e.g. Telemetry Statistics), is that the application crashes. Unfortunately, the application cannot detect this occurrence, so if (when) this happens, the only remedy is to delete the saved layout.*

```
$ rm -f ~/.config/mwp/.layout.xml
```

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]).

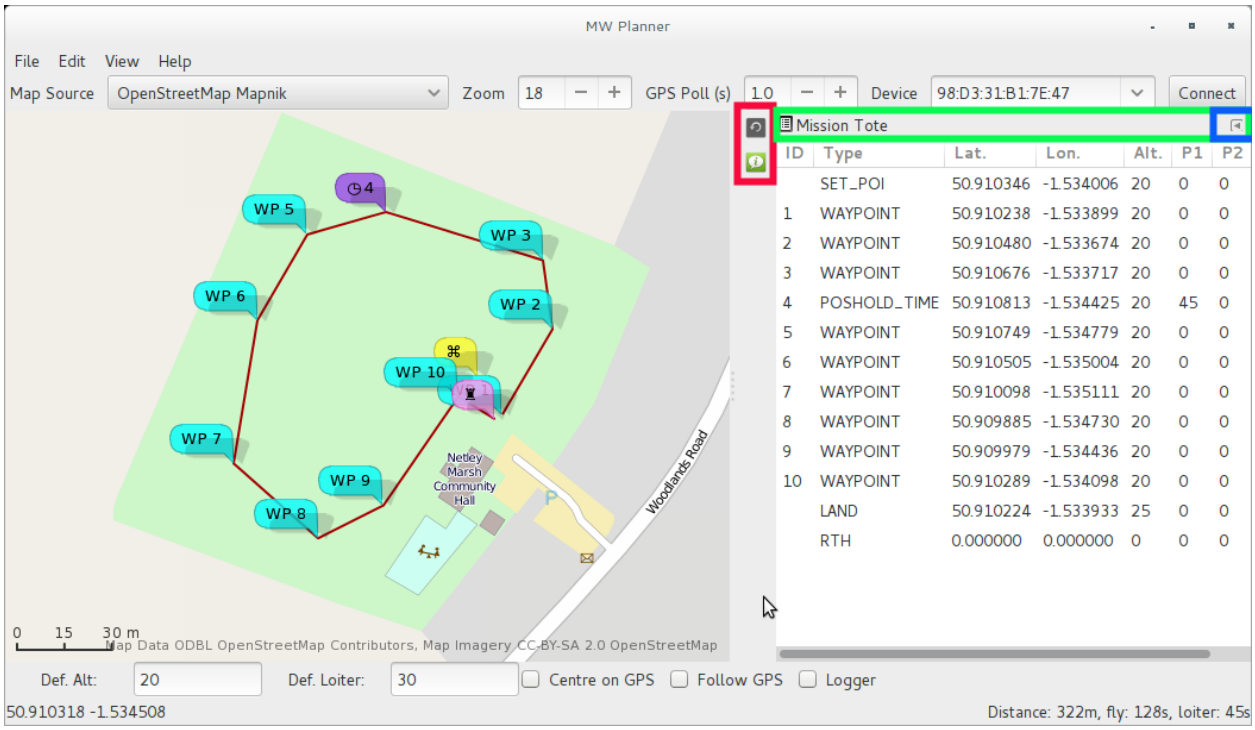

Illustration 18: Inital Dock display

In Illustration 18, 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 19, 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 figure Illustration 20, 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 iconifed dock item) from the View menu.

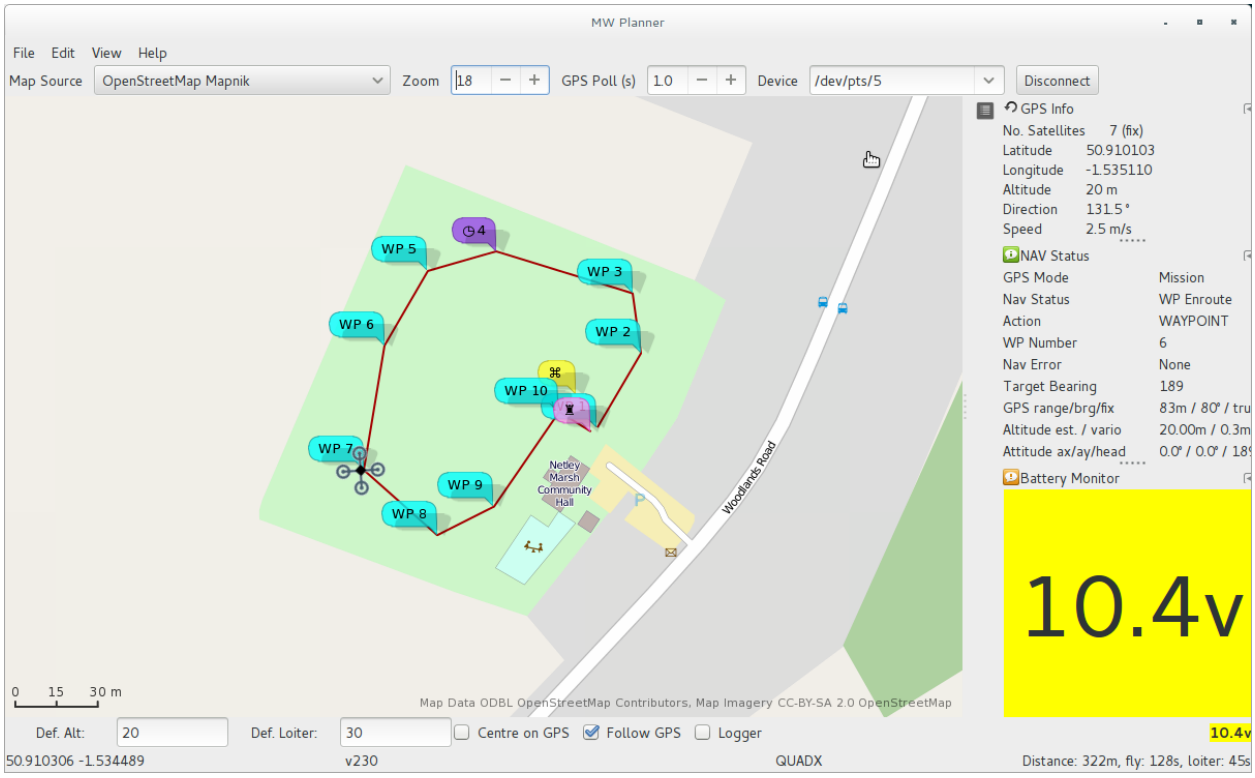

Illustration 19: Dock with battery monitor

Illustration 20: Battery window detached from 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. There is a 'quadkeys' patch for libchamplain floating around that enables Bing maps to be supported, apply at your own risk; this will not support other sources that require rewritten URIs based on scale (such as Ovi, Yahoo, Yandex etc).

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

- Define the "map-sources" key in the settings, this is the name of a JSON file in `~/.config/mwp` defining the map sources (or rather 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:

```
{
  "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).

Quadkey proxy

As applying the quadkeys patch is onerous, a second mechanism exists whereby **mwp** will provide a proxy that translates the TMS keys that libchamplain understands into quadkey notation that Bing needs, and fetches the tiles for you from a user defined quadkeys source. This allows the use of Bing imagery without patching libchamplain.

In order to enable the quadkeys proxy, you need to do a couple of configuration activities:

1. Copy the sample `quadkeys_proxy.json` file to `~/.config/mwp/sources.json` (or append it if you have other custom sources);
2. Review `~/.config/mwp/sources.json` and verify that the default proxy port does not clash with anything else you might be using. Do not change anything other than port number (unless you really understand what you're doing);
3. Ensure that the gsettings key 'map-sources' is set to allow user defined map sources;

```
$ gsettings set org.mwptools.planner map-sources 'sources.json'
```

4. Ensure that the gsettings key 'quaduri' that defines the real quadkeys source is also set (there is no default) is shown below, #Q# indicates where the per tile quadkey value goes).

```
$ # This sets aerial imagery only
$ gsettings set org.mwptools.planner quaduri 'http://a0.ortho.tiles.virtualearth.net/tiles/a#Q#.png?g=131'
$ # whereas, this will use aerial imagery with roads overlayed.
$ gsettings set org.mwptools.planner quaduri 'http://h0.ortho.tiles.virtualearth.net/tiles/h#Q#.png?g=131'
$ # Your choice ...
```

5. Convince yourself you are content with the Bing Terms of Service;
6. Enable "Bing Proxy" as the map source, set it as default if you wish.

If you are running an old version of libsoup (e.g. Ubuntu 14.04 or earlier), ***this will not work***, and you need run a standalone proxy instead of the internal one.

- Build the external proxy by changing to the mwptools/mwp source directory and make qproxy:

```
$ cd <path_to>/mwptools/mwp
$ make qproxy
$ # Copy somewhere useful ..
$ cp qproxy ~/bin
$ # or
$ sudo cp qproxy /usr/local/bin/
```

- Follow the steps 1-3 above;

- Launch qproxy with arguments of the port from `~/.config/mwp/sources.json` and the URI from step 4 above:

```
$ qproxy 21303  'http://a0.ortho.tiles.virtualearth.net/tiles/a#Q#.png?g=131'
```

- Now start **mwp.**

The script `samples/mwp_qproxy.sh` automates this (you will need to edit it to confirm the port and URI).

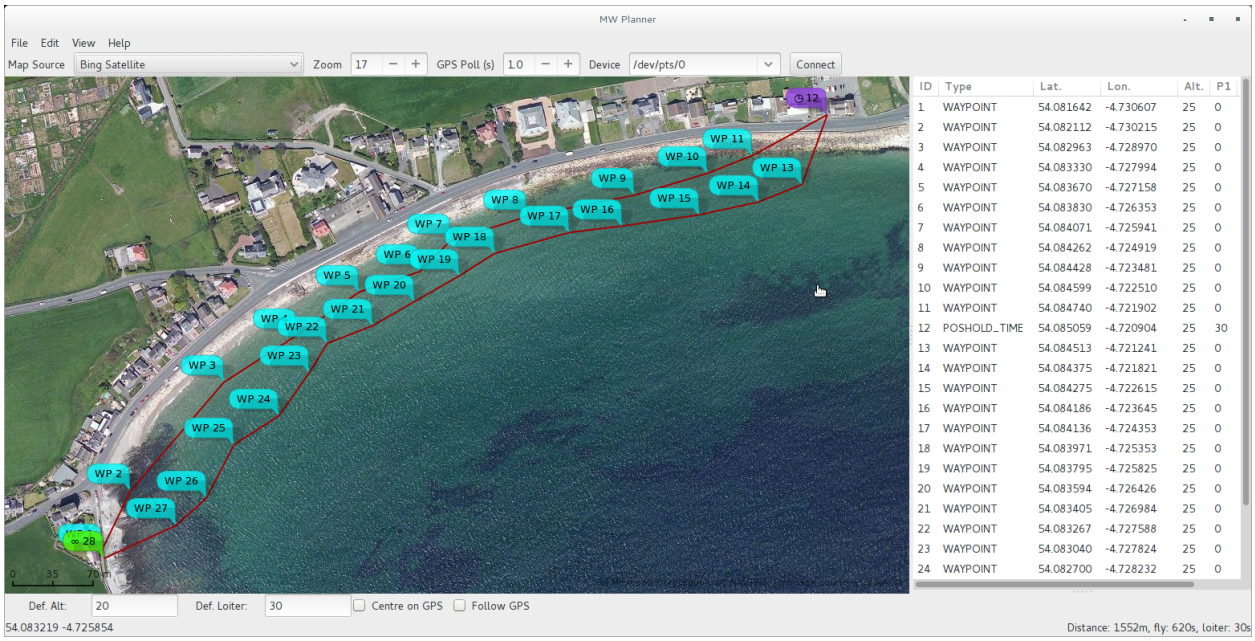

Illustration 21: Don't fly this when the tide is in (Bing maps)

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

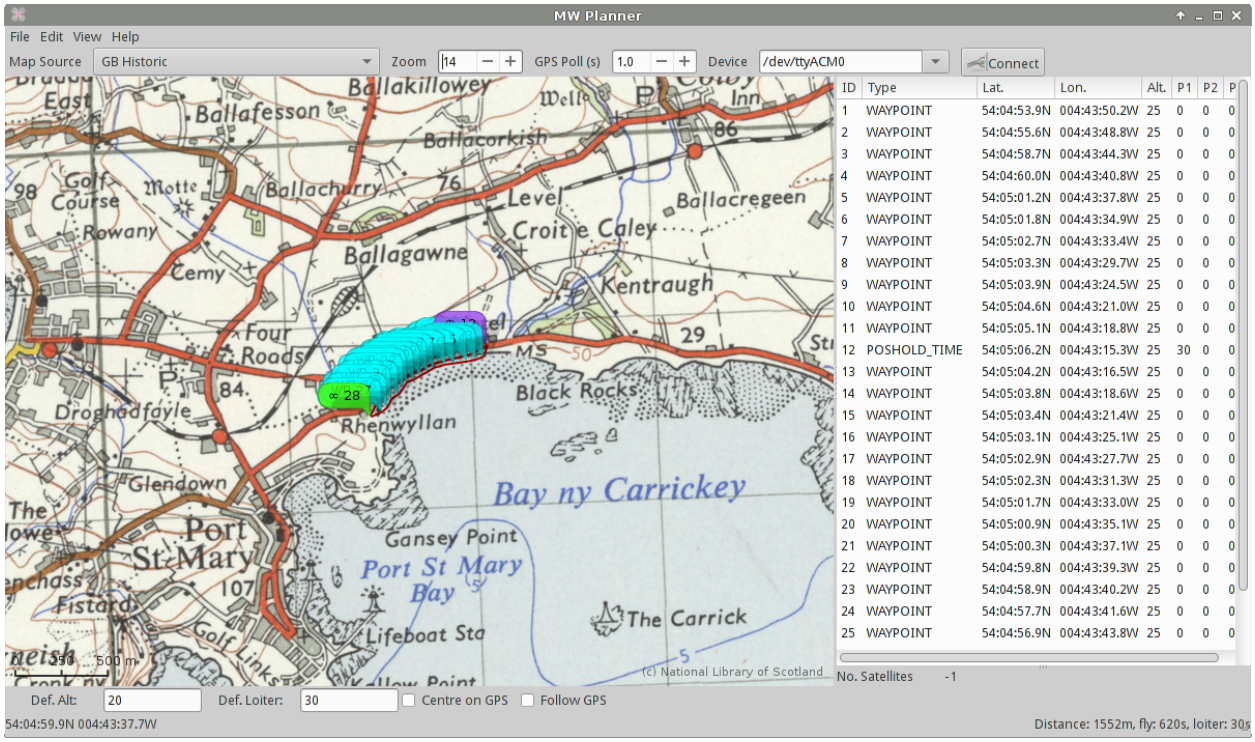

Illustration 22: "You can take the boy out of the island, but you can't take the island out of the boy"

Installation

The **mwptools** applications have a number of installation targets:

- make install-system: Requires root. Installs under `/usr`;
- make install-local: Installs under $HOME, XDG directories. Applications in `$HOME/bin`. Directories are created as needed.

Default for make install is install-local.

Note that for the .desktop files to be found for install-local, it is necessary that `$HOME/bin` is in the path when the desktop environment is started; note also that `~/.profile` is sourced by the desktop environment. It is also necessary for XDG_DATA_DIRS to contain `$HOME/.local/share`.

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.

Short Cut keys

In the map window, the following keys 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

Table 2: Short-cut keys

Building

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

- vala (0.20 or later recommended, may work with earlier), and gcc (or clang);
- GTK+-3.0 (and normal dependencies, 3.8 or later tested, 3.10+ preferred);
- libchamplain (0.12 or later);
- libchamplain dependencies (Clutter mainly);
- gdl (GNOME Docking library);
- libespeak.

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.

Specifically on Ubuntu, the version of libchamplain (as of Ubuntu 13.10) does not support bounding boxes, you should use the mwpu target, rather than **mwp**. In Ubuntu 14.04, the default libchamplain is sufficient.

mwptools is developed and tested on Arch Linux (ARM, x86_64, ia32); it is also tested on Ubuntu.

In particular, for Ubuntu 13.10 (or earlier), and platforms using versions of libchamplain c. 0.12.3, rotation of the vehcile is disabled, as this causes the vehicle to be incorrectly geo-located on the map. Libchamplain 0.12.6 (at least) have no such issues.

Protocol Documentation

In order to attempt to understand the new NAV protocol, a reverse engineered understanding is published as https://docs.google.com/document/d/16ZfS_qwc-rJeA7N5Tx0DA6wtgxl6HdGgaz-jE3lHBWs/edit?usp=sharing. I hope this is useful. Corrections appreciated.

pidedit

pidedit is a PID editor.

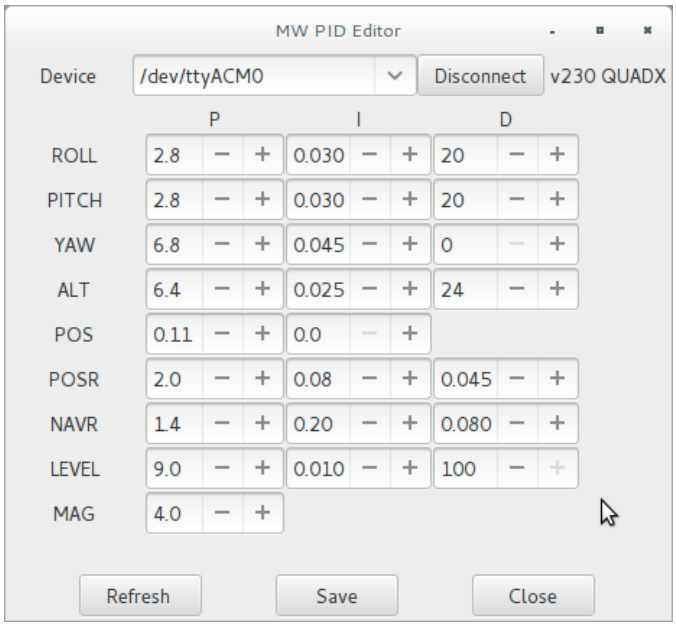

Illustration 23: **pidedit** - trivial PID editor

The application should be somewhat self-explanatory:

- Each PID parameter can be edited;
- Refresh: Refreshes the user interface from the flight controller;
- Save: Saves the values to the FC EEPROM;
- Close: Quits the application.

As for **mwp**, there is a gsettings entry for **pidedit**, the schema is also "org.mwptools.planner", and the only items used are "device-names", a comma separated list and baudrate.

switchedit – switch editor

switchedit offers the editing of switch settings. The screen-shot below shows the current user interface; **pidedit** now also offers the capability to save and restore settings.

Illustration 24 also shows how to specify a baud rate as part of the device name (otherwise defaults to 115200 baud, or the baudrate preference settings).

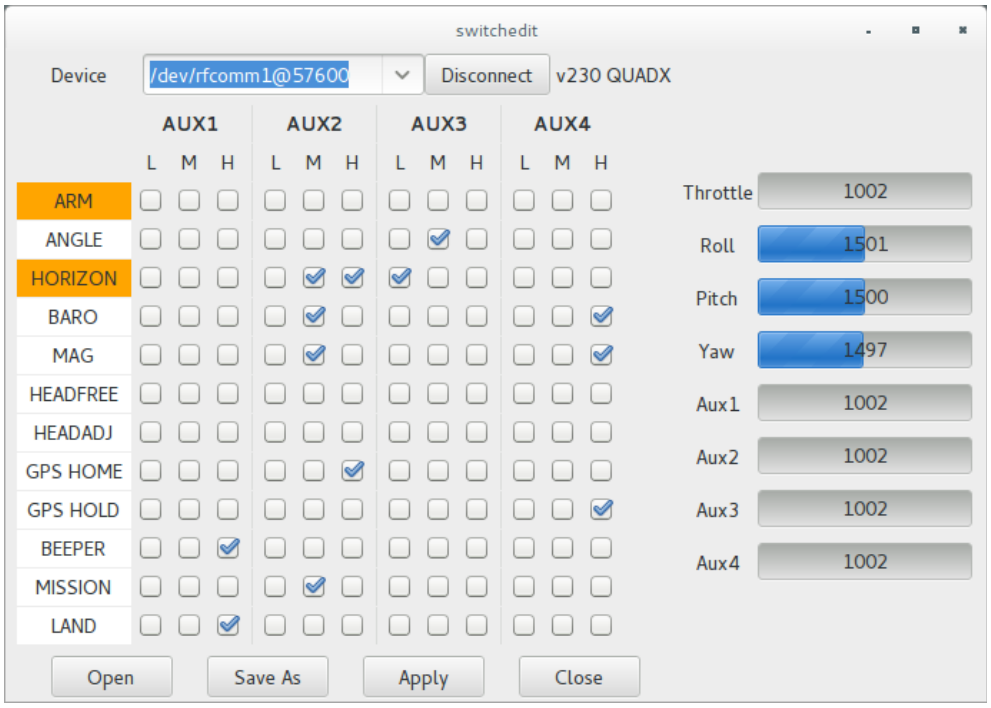

Illustration 24: Editing switch settings

Switchedit supports the cleanflight switch range mode by translating the range modes into discrete MultiWii compatible switch boxes, as above. If you use more than 4 switches or 3 settings, this will not work and you must use the cleanflight configurator.

mspsim – MSP simulator

mspsim is a simple simulator for the MSP protocol (including the undocumented NAV extensions). Its sole purpose has been to allow the author to develop **mwp** without a NAV capable MultiWii platform. Unlike the other tools in the **mwptools** suite, it is unashamedly POSIX dependent, using POSIX pseudo-terminals for communications (it would however be simple to use e.g.com2com on Microsoft OS). It also supports UDP (as do the other tools).

The concept is simple. On start-up **mspsim** allocates a master pseudo-terminal (pty) for its use and displays the slave pty device name that can be used in the application under test (e.g. **mwp** or **pidedit**).

In Illustration 25, **mspsim** is being used to test **mwp**. **mspsim** is invoked with the same mission file as is loaded into **mwp**. The slave pty advertised by **mspsim** is used as the serial device by **mwp. mspsim** responds to MSP commands issued by **mwp**, so the simulated FC configuration, status, GPS location and nav status is played into **mwp**. When the "Play" button is **mspsim** is activated, **mspsim** "flys" the mission, providing simulated GPS and "nav status" to **mwp**.

mspsim is a very simple simulation; it does not honour POSHOLD_TIME loiter time, nor does it honour JUMP; its sole purpose is to simulate the MSP communications. Thus a number of states are hard coded into the simulation (e.g. vehicle type is always QUADX).

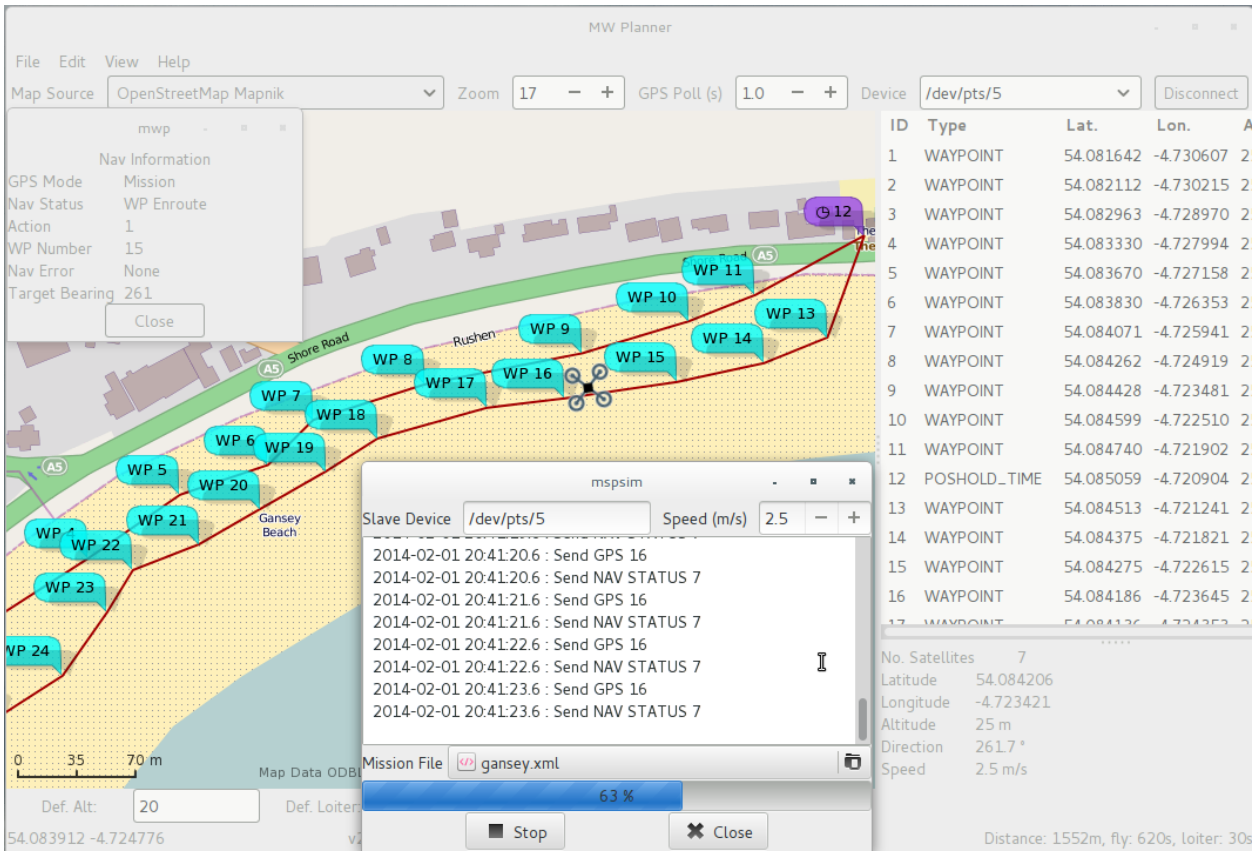

Illustration 25: **mspsim** being used to test **mwp** (pre-dock UI)

The numbers after the command mnemonics in the status list represent the (assumed) message payload size.

The following messages are supported (to some degree) by **mspsim**:

- The Connect / Disconnect buttons connect / disconnect the serial device.;

MSP Messages simulated

Mnemonic	Description, usage
IDENT	Version identity (230).
STATUS	FC status. **mspsim** supplies sensor (0x1f) and cycle time (2345).
MISC	FC settings. **mspsim** supplies minthrottle, maxthrottle, mincommand.
PID	PID settings. **mspsim** supplies values appropriate to an extremely lightweight, somewhat over-powered 250mm MTM QUADX.
ALTITUDE	Returns appropriate data from the simulated mission.
EEPROM_WRITE	Recognised, no response necessary.
SET_PID	Recognised, response sent. **mspsim** does not retain values supplied.
RAW_GPS	**mspsim** supplies values based on the mission file supplied (or 1 satellite, no fix if no file).
SET_HEAD	Recognised, no response necessary.
WP	**mspsim** supplies any values previously received via SET_WP.
SET_WP	Values are stored, and will be returned via a WP query.

Mnemonic	Description, usage
NAV_STATUS	**mspsim** supplies values based on the mission file supplied and the state of "Play" through the mission file.
NAV_CONFIG	Recognised, with a default filled response.
RADIO	Recognised, with a random filled response.
ATTITUDE	Recognised, response based on simulated mission.
COMP_GPS	Returns appropriate data from the simulated mission.

It must be reiterated that the purpose of **mspsim** is not to supply a realistic flight simulator, rather it is to provide responses to test the extended MSP protocol.

mspsim has the following options:

```
$ mspsim -help
 Usage:
  mspsim [OPTION…]

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

Application Options:
  -m, --mission             Mission file
  -x, --exhaust-battery     exhaust the battery (else warn1)
  -u, --udp-port            udp port for comms

$ mspsim -m nmtest04.mission -u 6666 # Use UDP port 6666, set device to localhost:6666 in mwp
```

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.

Command line options

mwp and **mspsim** may be supplied with the name of a mission file. If one is using **mspsim** to test **mwp**, it is advisable to use the same file.

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

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

Application Options:
  -m, --mission           Mission file
  -s, --serial-device     Serial device
  -c, --connect           connect to first device
  --ignore-sizing         ignore minimum size constraint
  --ignore-rotation       ignore vehicle icon rotation on old libchamplain
$ # thusly ...
$ mspsim ../samples/gansey.xml &
$ # see which slave PTY we're offered ... in this case /dev/pts/5
$ mwp -m ../samples/gansey.xml -s /dev/pts/5 -c
```

The `--ignore-rotation` option may be necessary on older versions of libchamplain (though the code attempts to avoid it being necessary for recent Ubuntu (e.g. 13.10).

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 Cleanflight Configurator may be used for configuration of items not addressed by **mwptools**.

Naze 32

The application also works with Naze 32, at least with the cleanflight <https://github.com/hydra/cleanflight> firmware. It may also work with baseflight, but it not tested, due to the latter's GPL abuse.

LTM Light (Telemetry)

mwp supports the ltm <http://www.wedontneednasa.com/2014/02/lighttelemetry-v2-en-route-to-ground-osd.html> protocol, as far as is reasonably possible.

Cf-cli

cf-cli is a command line tool to backup and restore Cleanflight settings. See README.md in the common/ directory, or <http://www.daria.co.uk/cf-cli/>, where you will also find a Windows binary.

Author, copyright and licence

mwptools is written by Jonathan Hudson <jh+mwptools@daria.co.uk> (aka stronnag on the MW forum, rcgroups, #cleanflight etc., but the NSA already knew that). It is licensed under the GPL 2.0 (or later, your choice).

This document describes the 0.01 pre-alpha software version.

Marko S aka marksev1 aka msev reported that localisation of decimal points was not handled correctly (now fixed). He has also bugged me tirelessly on ltm support :).

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
```