iNavFlight Missions

Overview

iNav supports autonomous flight using waypoints. In order to use this capability, it is also necessary to utilise and configure some supporting technologies, including:

- A GCS (Ground Control Station). The GCS will typically provide functions to create waypoint (WP) missions, upload WP missions to the flight controller (FC), validate the mission, execute the mission and log the mission;
- Telemetry Hardware. In order to transfer the mission to the FC and monitor the mission in real time during mission execution it is necessary to install and configure a telemetry system between the GCS and the multicopter.

This wiki topic describes the software currently available and some of the telemetry options. Please also see the wiki page on more general navigation mode options.

Before you get started on a waypoint mission, you need to know what the expectation is. Namely, the constraints of what is needed in order for the waypoint to be loaded in the FC and the FC to allow you to ARM without a message of "Navigation Is Unsafe". If you get this message after loading your mission, one of the following is the cause:

- You configured WP/PH/RTH or Failsafe RTH and you don't have a good accuracy GPS fix
- You try to arm into RTH/PH/WP
- You have waypoint mission in memory and your first waypoint is too far from your current position
- HODP is too low

The default distance for the first waypoint is configured with the 'nav_wp_safe_distance' value (default of 10000cm, ~ 300 feet).

The MSP (MultiWii Serial Protocol) messages defining mission navigation are documented. This message set is supported by the Mission Planner for INAV and mwp ground stations.

Ground Control Stations

Currently there are a number of GCS applications widely used for iNav mission management, including Mission Planner for INAV (Android), MobileFlight (IOS) and mwp (Linux). In future, other options may become available, particularly as the MAVLink protocol becomes supported by iNav. However, MAVLink based tools will only provide monitoring.

Mission Planner for INAV and mwp (at least, maybe MobileFlight as well) support mission planning (they share a common mission definition file format, so missions can be used in either tool), mission upload / download, mission monitoring and mission logging.

Note: Earlier versions of this article recommended ezgui for use on Android. ezgui is no longer maintained and Mission Planner for INAV is the recommended Android application.

Mission Planner for INAV (Android)

Mission Planner for INAV can be downloaded from Google Play Store. There is a free version which limits number of waypoints to 5 and (very reasonably priced) paid-for version with additional functionality. The application is not open source. For questions and help the RCG "Mission Planner for INAV" thread can be used: RC Groups support forum.

Droid Planner 2 (Android)

Droid Planner 2 can also be downloaded from the GitHub. It is free and released under GNU Public License v3.

Droid Planner only supports iNav's one-way mavlink protocol. The following telemetry data is displayed:

Vehicle position on map, active flightmode, heading, altitude, speed.

A broken connection recovers once restored after any amount of time. The flight track remains on screen even when data link is broken -> lost model recovery. Log files can be opened in PC software Mission Planner.

mwp (Linux / FreeBSD)

mwp can be downloaded from Github. mwp is open source (GPL 3). It is available only as a source distribution and it is necessary to compile and install the application. Build instructions and dependencies are provided for Ubuntu and Fedora. Arch Linux users can install mwp from the AUR (Arch User Repository).

In addition to mission planning and logger, mwp also supports the replay of blackbox logs against a geospatial background (requires blackbox-tools). mwp also includes numerous poorly documented scripts for mission and blackbox analysis, as well as an overly comprehensive user guide.

There is a RC Groups support forum

Mobile Flight (IOS / iphone).

Mobile Flight: Configuration and ground control app for iNav (and Betaflight) on iPhone http://www.rcgroups.com/forums/showthread.php?t=2601895&highlight=ios.

iNav Configurator

Since version 1.9.2, the iNav configurator provides rudimentary mission planning capabilities. As it has no ability to local store files, missions must the saved to EEPROM if it is necessary to store the mission for transport. In the field, eeprom-stored missions can be restored using stick commands.

Potential solutions for other platforms

mwp can be run in a virtual machine on MS Windows and OSX / macOS, using virtualisation tools such as VirtualBox and Parallels.

WinGUI is a Windows program developed for Multiwii-nav. It is currently somewhat abandoned, but would be a viable basis for developing a Windows program for iNav navigation (or better, supporting both Multiwii and iNav, as do the other tools described here). Should anyone wish to rescue this fine application, the source code (GPL v3) may be found at https://code.google.com/archive/p/mw-wingui/.

Telemetry Hardware

In order to transfer missions from the GCS to the flight controller, and to monitor / log flight data, it is necessary to establish a data link between the GCS and the multirotor. Some popular technologies include:

- Bluetooth
- 3DR (433Mhz / 915Mhz)
- WiFi (ESP8266)
- HC-12 (433Mhz, similar to 3DR)
- Openlrs/Openlrsng devices (such orangerx 433 tx/rx combo)
- LoRA (868 / 433 Mhz options)

Bluetooth

Bluetooth is the easiest solution to get working with minimal effort. A cheap HC-06 BT module is all that is needed (the phone or laptop built-in BT is used on the ground station). Its disadvantage is the range, for most users data loss / dropout will occur over 20m. Its thus useful for testing out configurations, but for many users the limitation of range will call for another solution.

Setup guide

3DR

3DR radios operate in the regionally unlicensed 433MHz and 900MHz bands. They are widely available from online retailers. Detailed documentation is available at from Ardupilot.org. The standard 3DR firmware is designed for the MAVLink protocol. While there is a fork of the firmware available for the MSP (Multiwii Serial Protocol), it does not support recent advances in iNav (MSPv2, LTM); and the current recommendation is just to use the standard firmware with MAVLink options disabled.

3DR is a medium range technology, up to at least 1km. Range is somewhat dependent on baud rate and is well documented

Advanced configuration for 3DR is detailed at the end of this wiki page.

ESP8266

ESP8266 is a small WiFi to serial bridge. It can be used to transport the serial data link over WiFi. It offers reasonable range (c. 300m) and convenience. The author has seen no evidence of interference between ESP8266 devices and 2.4GHz RC links.

Advanced configuration for ESP8266 is detailed at the end of this wiki page, some preliminary data can be found in this RC Groups post. That post demonstrates excellent coverage out to 150m using mwp, ESP07 and ESP01 modules and the standard vendor firmware. The ESP07 module works well with an external antenna.

There is an ezgui howto on ESP8266 devices.

Another, highly detailed how-to for ESP8266 and Cleanflight/Baseflight/INAV is available here. This reports very poor results, possibly due to the native WiFi capability in the phone hosting ezgui (vice the laptop adaptor for the mwp test).

HC-12

HC-12 is a comparable radio technology to 3DR with similar range and performance characteristics. Its configuration and usage with iNav is well documented https://quadmeup.com/diy-wireless-telemetry-link-for-uav/ and https://quadmeup.com/hc-12-433mhz-wireless-serial-communication-module-configuration/. The configuration documented would work equally well in ezgui and mwp. These small radios work really well with good range in FU3 mode / 9600 baud (and very cheap).

Openlrsng

Openlrsng is a full radio control system, mainly used for LRS (long range systems). It supports radio beacon for lost models, failsafe and other characteristics.

For telemetry data, it offers a bi-directional channel, and Frsky, S.Port (both simulated protocols) and serial transparent telemetry are allowed. The telemetry range in this system depends on power, antennas and baudrate. Lowering baudrate, and with good antennas, very long distances have been acchieved with full telemetry at ground station.

Openlrsng can be combined with bluetooth devices at GCS, so you could connect to the model in flight with your phone, tablet or PC. In this case, depending on the protocol used or the complexity of your transmitter or the software in your android device, there are many options, like seeing the data on the LCD screen of the transmitter(er9x, LUA scripts for Taranis..), using of an antenna tracker, practicing a 'follow-me' performance...

A great number of compatible openlrsng devices can be found, from Hobbyking (UHF/LRS orangerx) to ebay and other suppliers.

LoRA

LoRA provides the capability for low power / long range telemetry using similar arrangements as for 3DR and HC-12, with the possibility of extended range. A description of a working setup and albeit short range comparison with 3DR/HC-12 is in the mwptools wiki or as a PDF.

Other solutions

Other solutions include Dragonlink. Contributions to the wiki solicited!

Telemetry Protocols

Data is transferred between the GCS and the FC using a "Telemetry Protocol". Currently, iNav offers two protocols (MSP and LTM), both of which are supported by ezgui and mwp. There is also a minimal implementation of MAVLink (mwp already supports this MAVLink subset), this will allow other tools to be used, such as the cross-platform QGroundControl. The MAVLink implementation only supports push telemetry (i.e. mission monitoring, not mission planning).

MSP - MultiWii Serial Protocol

MSP is the 'native' messaging protocol for iNav. It is well supported by the configurator, ezgui, mwp and many OSDs. It is all you need to upload missions and monitor flights. Its one disadvantage for mission monitoring is that it is a polled protocol, that is the GCS has to request data and the FC responds. This is not really an issue for some data links such as BT and WiFi, but the half-duplex nature of 3DR, where there is significant time cost in switching between receive and transmit modes, significantly limits the performance for mission monitoring.

mwp (and possibly other ground stations) can mitigate this performance hit by using MSP for configuration, mission upload / verification and monitoring prior to arming, and when configured in the FC, switching to LTM for mission monitoring when armed. This switch-over is automatic and transparent to the user.

LTM - Light Telemetry

LTM is a 'push' telemetry protocol; that is the FC sends data unsolicited to the GCS. This avoids the 'half-duplex' time penalty of MSP on 3DR radios. Unlike MSP, LTM only provides flight data, thus if you need the GCS to select a vehicle icon based on the multirotor type (QUADX, TRI etc), offer additional functions based in the FC firmware version or upload waypoints, then it is necessary to share the serial port on the FC between MSP and LTM; MSP is used when unarmed and LTM when armed. Both ezgui and mwp handle the switch-over automatically.

You can find documentation / specification for the LTM implementation in Inav in the iNav Wiki.

LTM will operate effectively over low data rate links. Currently the iNav implementation pushes c. 300 bytes /sec in its fastest rate, so 4800 baud over the air rate would suffice. iNav provides a configuration options for 'medium' and 'slow' LTM rates, further reducing the required baud rate, which may in turn increase range for some radio options.

LTM is supported by ezgui, mwp and (for OSD, ltm-osd-simple)

MAVLink

MAVLink is a full-feature, highly capable protocol used by PX4, PIXHAWK, APM and Parrot AR.Drone platforms (inter alia). The implementation for iNav is 'push telemetry' only, so it can only be used for flight monitoring, not mission planning.

The initial implementation in iNav is supported by ezgui, Droid Planner 2, mwp and QGroundControl. Probably some of the Android apks for Mavlink will work with this telemetry protocol. Tower (Droid Planner 3) currently doesn't work (is this still true?)

Configuring the Flight Controller

Ports & port sharing

If order to use mission planning or just flight monitoring, it is necessary to configure a port on the flight controller. Due to the often limited number of ports, multiple devices and potential baud rate clashes, some compromises may have to be made.

- Most users will want MSP on UART1 at 115200 baud (or better) for the typically shared USB connection for flashing and configuration;
- For reliable GPS performance, it is recommended to run the GPS on a hardware serial port;

- A Blackbox logger typically requires a high baud rate;
- You can only have MSP enabled on two ports;
- Telemetry can run at a slow rate, even on soft serial.

From this, some configuration examples; both these examples assume a PPM RX:

Simple, short range 'park flyer'

- UART1 MSP (USB and Bluetooth), same baud rate (typically 115200)
- UART2 GPS

Advanced, black box and telemetry: F1 hardware

- UART1 MSP (unarmed), Blackbox (armed). The baud rates may differ (e.g. 115200 MSP, 250000 BBox);
- UART2 GPS
- Softserial MSP and LTM (MSP unarmed, LTM armed), maximum 19200 baud.

Advanced, black box and telemetry: F3 hardware

- UART1 MSP (unarmed), Blackbox (armed). The baud rates may differ (e.g. 115200 MSP, 250000 BBox);
- UART2 GPS
- UART3 MSP and LTM (MSP unarmed, LTM armed). No speed limit, but 3DR / HR-12 will have better range at low rates, and there is no benefit to higher rates.

Using a serial RX is more difficult, particularly for F1 devices. For F3, in the final example, putting the serial RX (Sbus, SpekSat) on UART3 and using soft serial for for MSP+LTM would be an acceptable solution.

Mission Planning

iNav currently supports a subset of the WP / Mission MSP "specification". The following waypoint types are available (iNav 1.1).

- Waypoint (leg speed addition)
- Infinite position hold
- RTH (auto land available from 1.2 RC1)

The following MW-NAV / MSP functions are **not** yet implemented:

- Timed Position Hold
- Set POI
- Jump
- Set Heading

- Land

ezgui and mwp support iNav WP navigation; they both use the mission definition originally implemented in WinGui, thus mission definitions are interchangeable between these application (and mw-nav if you limit the mission features to the common subset).

ezgui and mwp both provide interactive WP editing on a geospatial background and mission upload to / download from the multicopter. At least for mwp (to be confirmed for ezgui), the mission upload process also downloads the mission and compares the two. **You should not attempt to fly a mission unless it has validated**.

On F1 baords (Naze, Flip32), you can defined 30 waypoints, for F3 and better FCs, 60 waypints can be defined.

Missions are initiated by a switch setting on the RC TX. It can also be aborted at any time turning this switch(NAV WP) off.

A mission is manually terminated by RTH, infinite position hold or reaching the end of the waypoint list. In the latter case, the vehicle will enter a position hold state until the pilot takes manual control (by negating the TX WP state).

An 'in progress' mission flight may be aborted prior to reaching one of the above end points by:

- Switching out of WP mode; or
- Invoking RTH.

Mission / Flight Monitoring

Prior to engaging any automated mode, it is advisable to verify that you have reasonable satellite performance. Even with 10+ satellites and HDOP < 1.5, there is a remote possibility that you might experience 'a bad satellite day'; there's an example described in issue 431. An easy way to verify you have good coverage is to try POSHOLD before executing a mission (or RTH).

Waypoint Mode

As soon as iNav starts a leg on a WP mission, it will attempt to reach the leg altitude, so if you have

- WP 1, altitude 10m
- WP 2, altitude 50m

On engaging WP mode, iNav will attempt to reach 10m altitude. On passing WP 1, iNav will attempt to reach 50m. Altitude is not taken into consideration in determining when a waypoint is reached (only latitude / longitude).

WP mode is only disengaged under the following circumstances:

1. GPS is lost (will switch to emergency landing and land)
2. Failsafe took over (Radio link lost)
3. Pilot manually disables WP mode by either turning off the switch or enabling RTH
4. The end of the mission is reached.

Advanced configuration

3DR

Hardware

3DR radios are sold either as a pair of air station / ground station or individually. Functionally, the air / ground radios are identical, the air side having a tty/serial connection and the ground side having a USB interface for connecting to a computer. For ezgui (and mwp), it is easier to use a Bluetooth bridge. This bridge is also recommended for mwp, as it avoids any potential RF interference from the USB cable and allows the more flexible placement of the ground antenna. In order to use the 3DR / BT bridge, it is necessary to have 'air side' devices at both ends of the link. It is then necessary to 'back-to-back' the ground 3DR and the BT device example field setup and provide power. A voltage regulator and an old lipo works really well. The HR-12 description provides the canonical connection diagram, a 5V regulator or BEC may be used.

Firmware

The 3DR radios will ship with a version of the Sik Firmware. This firmware is optimised for MAVLink (it understands MAVLink framing, reports RSSI to a MAVLink GCS). There is a fork (of an older version) that provides similar capabilities (understands MSP framing, reports RSSI to a MSP GCS (ezgui, mwp) for MSP; however its use is no longer recommended as it does not understand MSPv2 or LTM, so it somewhat pointless.

Configuration

Prior to use, it is advisable to configure the 3DR radio to meet local regulations for unlicensed use and to optimise the air speed for maximum range. This can be done either through a graphical user interface or a serial terminal interface using tools such as `picocom` / `screen` / `putty`. The graphical tool will run on Linux using 'mono'. For the following discussion, the serial AT command set is used.

- If you use the modified MSP aware firmware, then you can enable MSP framing:

```
ATS6 = 1
```

or both MSP framing and MSP radio status reporting:

```
ATS6 = 2
```

- Otherwise (recommended), just turn message based framing off:

```
ATS6 = 0
```

- You will get better range at lower speeds, it is also good practice to set the air speed and the ground speed to close rates, in order to minimise the probability of serial overruns. Here we set air speed to 24000 baud and ground speed to 19200 baud. Actually, as the data rate is low (and fits within the device buffers, this is not so important; an air speed of 24000 baud and a ground speed of 115200 works as well.

```
ATS1 = 19
ATS2 = 24
```

These settings are more than adequate for both MSP and LTM.

- Another useful setting in the MAX_WINDOW (`ATS15`). If you only intend to MSP (no LTM), then set this to a small value, the minimum is 33; this minimises latency.

```
ATS15 = 33
```

- However, if you intend to also use LTM, set this to highest permitted value (131) to maximise throughput, or experiment with intermediate values `ATS15 = 66`:

```
ATS15 = 131
```

- As MSP and LTM provide checksums, we can disable some error checking / correction:

```
ATS5 = 0
```

- It is necessary to have the same settings on both the air and ground radios for the majority of settings (otherwise the radios will not connect). Repeat the settings using RT rather than AT, then save and reboot both device: do the remote first, e.g.:

```
RTS15 = 131
RT&W
RTZ

ATS15 = 131
AT&W
ATZ
```

If you use Linux and a USB connected ground side (rather than the USB / BT bridge), you can use `udev` to set the device name. You will need to use `lsusb`to find the `serial` parameter for your device. The rule below links the `/dev/ttyUSBx` name to `/dev/3dr`.

```
### /etc/udev/rules.d/66-3dr.rules
# Hextronic radio
```

```
KERNEL=="ttyUSB*", ATTRS{serial}=="A7032PAY", SYMLINK+="3dr"

# GLB radio
KERNEL=="ttyUSB*", ATTRS{serial}=="A8005McD", SYMLINK+="3dr"
```

ESP8266

Firmware

The ESP8266 devices will usually ship with vendor firmware. Follow the link to SDKs, find the latest ESP8266_NONOS_SDK version. There is a Windows specfic flashing tool, or you can use the portable tool. This firmware is recommended for mwp, as you can use it as a transparent UDP / serial bridge (but you can also use the 3rd party firmware TCP bridge).

For ezgui it is recommended to use 3rd party firmware that provides a a transparent TCP / serial bridge. This firmware may also be used in mwp.

Configuration

Configuration of the 3rd party TCP bridge is described in the ezgui howto. For mwp, this device would be defined as:

```
tcp://host:port
```

So using the ezgui example verbatim:

```
tcp://192.168.4.1:23
```

For the vendor firmware, UDP connection, configure the device as an Access Point (AP) with your own ESSID and strong passphrase. It is necessary to define both the local and remote UDP ports (14014 in this example). See the latest firmware documentation for an explanation of the AT commands:

```
AT+CWSAP_DEF="I'mMandyFlyMe","correct horse battery staple",11,4,2,1
AT+CWDHCP=2,0
AT+CWMODE_DEF=2
AT+CIPAP_DEF="192.168.100.100",,"255.255.255.0"
AT+SAVETRANSLINK=1,"192.168.100.101",14014,"UDP",14014
AT+UART_DEF=57600,8,1,0,0
AT+RFPOWER=60
```

Then in mwp, define the connection as (where esp-air is the host name of the air platform device):

```
udp://:14014/esp-air:14014
```

It is possible to access the CLI over a WiFi device, and with some `socat` tricks, also the configurator, which is highly convenient for tuning in the field.

- Access CLI over the UDP link

```
nc -p 14014 -u esp-air 14014
```

- Access CLI over TCP link

```
nc 192.168.4.1 23
```

- Accessing the configurator is a little more complex, `socat` is used to create a pseudo device (pseudo-terminal) linked to the IP connection. The configurator is then connected to the 'Manual Selection' port `/tmp/vc0`.
- For UDP (esp-air is the ESP8266 on the vehicle, esp-gcs is the computer / WLAN interface host name).

```
socat  pty,link=/tmp/vc0,raw  udp-datagram:esp-air:14014,bind=esp-gcs:14014
```

- For TCP

```
socat  pty,link=/tmp/vc0,raw  tcp:192.168.4.1:23
```

It is necessary to kill the socat process to use telemetry again.