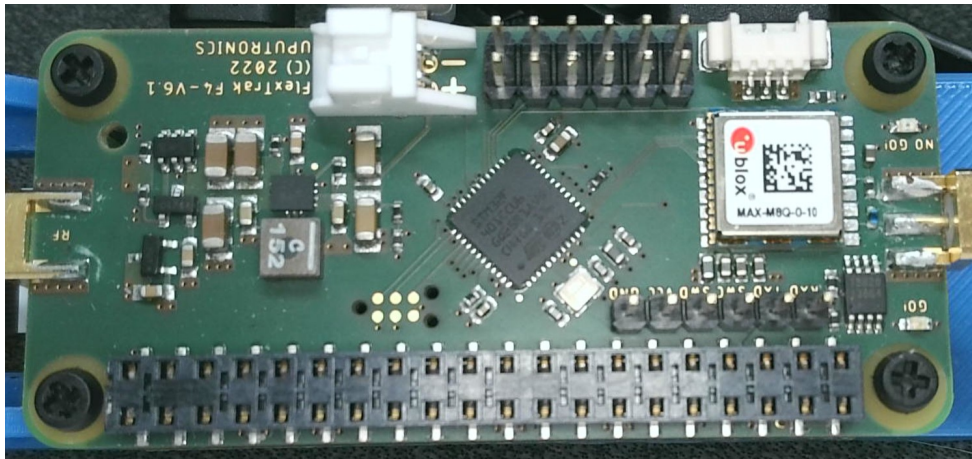# FlexTrak STM Installation Manual

This document covers everything you need to know in order to set up and use your own GPS/LoRa radio tracker, using the FlexTrak tracker board with or without the supplied Raspberry Pi Zero computer. It also covers the use of DS18B20 external temperature sensor and the APRS add-on board.

Introduction

This is the FlexTrak STM board:

What you are looking at is the top of the board, and this side should still be visible after you've connected it to the Pi Zero, as we will discuss in the next section.

FlexTrak is a unique solution since it contains its own micro-controller which runs completely independently of the attached Raspberry Pi. So even if the Pi crashes or reboots then the tracker will keep running. It also means that, once configured, it can be used on its own (just add battery and aerial) to make a lightweight low-power tracker.

Compared to our previous FlexTrak, this board has much more memory and processor power, and this has allowed us to include in the firmware code to predict the landing position and add that to the telemetry.

On the board you will find:

- Buck/boost converter
- STM32F401CC microcontroller
- UBlox GPS receiver rated for up to 50km altitude
- LoRa module for 434 / 868 / 915 MHz operation (according to model)
- Temperature sensor plus socket for external DS18B20 temperature probe
- Pin header for APRS and Sensor/Cutdown add-on boards

The board can be used as a standalone tracker, with settings (such as frequency and payload ID) stored in flash memory. However we expect the vast majority of customers to use it with the supplied Raspberry Pi Zero WH, which adds the following features:

- Take photos, stored on SD card, using a connected Pi camera
- Send photographs to the ground by radio
- Configure the FlexTrak board from a text INI file

You can also connect other Pi HATs to the Pi using stacking connectors. This enables, for example, the addition of other sensors which are read by the Pi and sent to the tracker board for transmission over the radio link to the ground. It also includes our APRS and cutdown/sensor boards which connect to the FlexTrak board directly.

What's In The Kit

In the kit you will find:

- FlexTrak Board (pre-programmed)
- Raspberry Pi Zero W board
- 32GB Micro SD Card (pre-programmed)
- Battery Holder for 4 AA cells (only use Energizer Lithium cells for flight)
- Power cable
- Pin Header, spacers and bolts for mounting FlexTrak on the Pi
- Pi Zero Camera Cable
- Single-ended coax cable (pigtail) from which to make an aerial
- GPS antenna

## *Differences From Pi In The Sky*

- Tracker starts immediately on power-up – no wait for Pi boot
- Tracker runs independently of Pi
- Pi just provides image data, landing prediction, and any extra sensors you add
- Uses LoRa rather than RTTY
- Pi code is Python so easier for most people to modify than the PITS C code
- Programmed SD card is supplied so no need to set one up from scratch

Differences From previous FlexTrak AVR

- More powerful processor
- Landing prediction is done on the FlexTrak board rather than the Pi
- SMA connector for GPS

Assembly

First, place four of the supplied 11mm stand-offs on the Raspberry Pi Zero W board, at each corner and screw in from below. The stand-offs should be on the same side as the header:
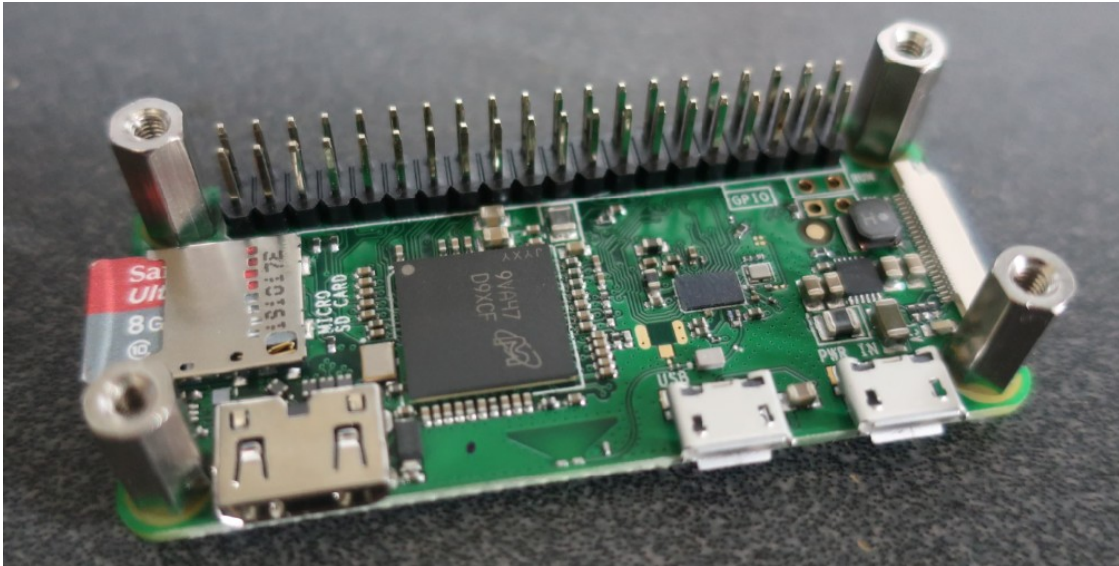

Next, place the supplied 2x20 header onto the Raspberry Pi, checking that it aligns correctly in both directions, and being careful not to bend any pins on the Pi or the header.


Note: If you want to stack another board on top, then use the stacking header supplied with that board. Stacking headers have much longer pins. If you use our APRS or sensor/cutdown boards, then the sequence is Pi Zero then FlexTrak then sensor/cutdown board or APRS board.

Now place the FlexTrak board on top, ensuring that it is the correct way up (i.e. header and components on the top):
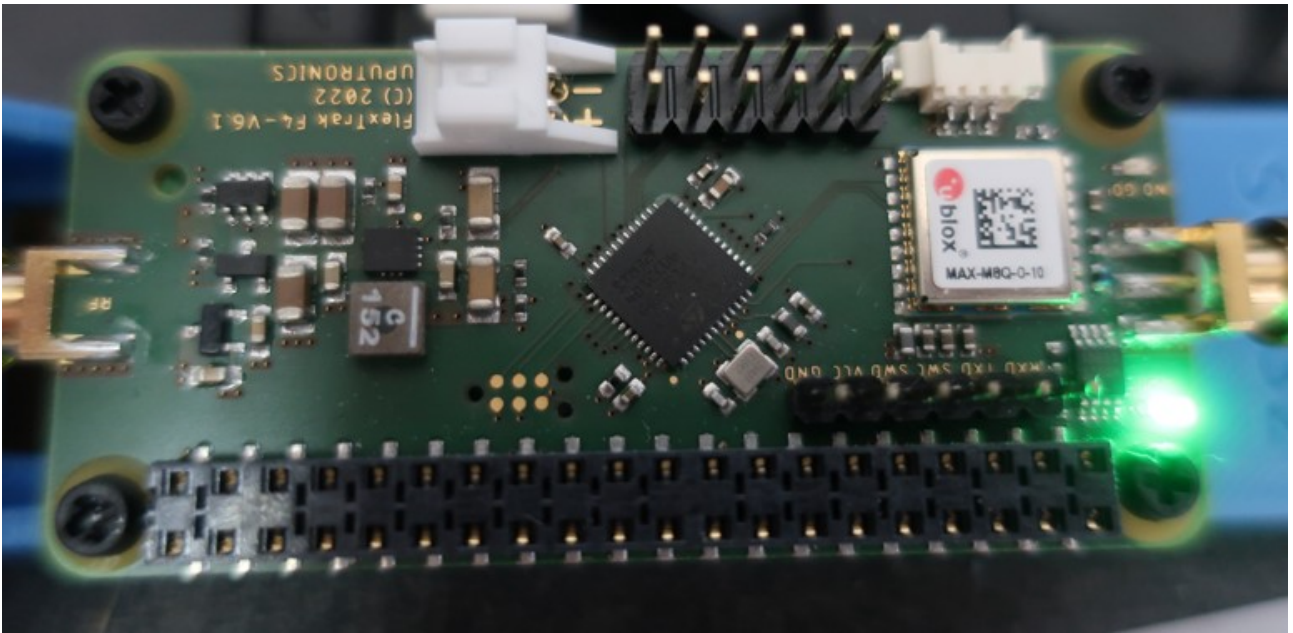

Screw the remaining 4 screws into the stand-offs to secure the board in place. Finally, plug in the micro SD card and, if you are using one, a Pi camera.

FlexTrak Connections
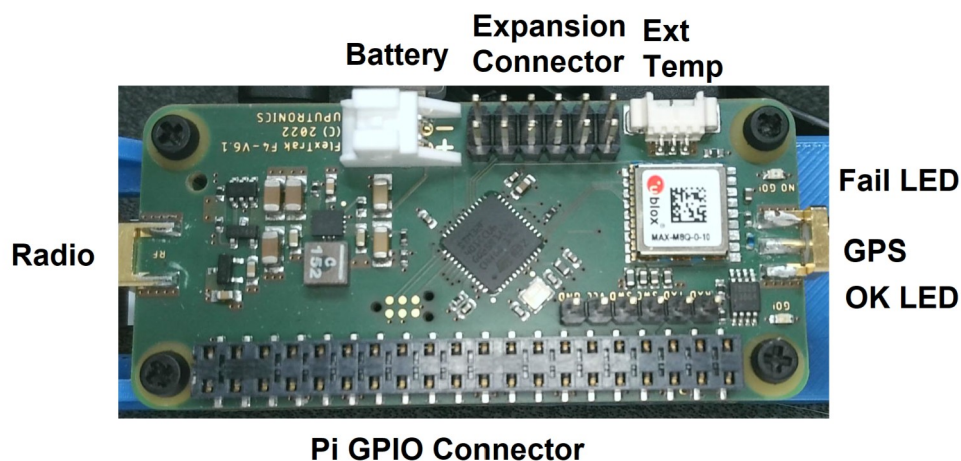
- Radio Aerial Connector – this connects to an aerial made from the supplied "SMA Pigtail".
- Battery Power Connector- A cable is supplied. For testing you can use standard alkaline cells, but for the flight you <u>must</u> use 4 fresh Energizer Lithium cells.
- Expansion Board Pins – For the optional APRS and Cutdown/Sensor boards.
- External Temperature Probe – for the supplied DS18B20 temperature probe.
- OK LED – Flashes green to show valid GPS position.
- GPS Aerial - The GPS aerial needs a clear view of the sky, so place it very close to an external window when testing, and near the top of the payload away from any metal or electronics when flying.
- Warn LED – Flashes red to show no valid GPS position. The amount of time taken to gain lock depends on how visible the sky is from the GPS antenna, and can be several minutes especially if indoors.
- Pi GPIO Connector – Used to connect to the Pi and, if you add more boards with a stacking header, connects to those also.

As the Pi Zero does not have a network connection, if you wish to connect it to a network we recommend use of a USB-LAN adapter; some of these have a micro USB plug and will plug directly in to the Pi Zero; others have a full size "A" plug and need to connect via an "OTG" USB cable adapter.
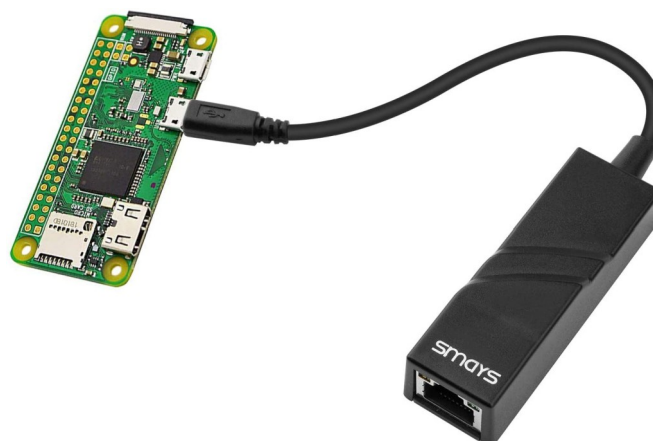
Starting The Tracker

If you have one handy, connect an aerial to the SMA socket, however the tracker will work over a few metres without an aerial. For flight you will need to make an aerial from the supplied pigtail (see later section).

Connect the GPS aerial to the SMA socket.

Connect the Pi Zero to a USB keyboard. You may need a micro USB to USB adapter for this.

Connect the Pi Zero to a HDMI monitor. You may need a mini HDMI to HDMI adapter for this.

In lieu of the above, you can control the Pi over a wired network, by connecting the Pi Zero to that network with a suitable adapter such as this one which includes full-size USB sockets for keyboard etc.:

https://thepihut.com/products/ethernet-hub-and-usb-hub-w-micro-usb-otg-connector

Connect the Pi Zero to power via the micro USB power connector.

The FlexTrak board will start immediately, transmitting telemetry over the radio.

Once GPS lock has been established, the WARN LED will go out and the green OK light will flash.If this doesn't happen within a couple of minutes then find a better location for the tracker. Or, if you have a GPS aerial with a long cable and SMA plug, connect that to the board, with the aerial near a window.

The Pi will boot and will automatically start the Pi tracker software. When that software starts it will send new configuration settings to the FlexTrak board, so if for example you have changed the payload ID, then the tracker board will start sending that new payload ID once the Pi has booted. The new settings are stored in flash memory on the FlexTrak board so will be used immediately that power is applied in future.

Logging In To The Pi

You can login with a connected keyboard/monitor or over a wired network with a USB-LAN adapter and use a suitable terminal program (e.g. putty) to connect to it. We have set the hostname of the Pi to "flexstm", and the login name/password are the Raspberry Pi defaults of pi/raspberry.

Using either method you can change the configuration of the tracker (see the next section). You can also connect the Pi Zero W to a wireless network, and use that or a wired network to load any software updates.

Configuration

The tracker configuration is done via a small text file on the "boot" partition of the SD card. This partition is a normal FAT (Windows) partition so it can be edited by placing the SD card in the SD card reader of any Windows PC or Mac, using a good text editor **(not** Notepad on Windows!)**.** However we recommend editing from the Pi, using nano or whatever editor you prefer.

The file contains many options, and often more are added with each new release of the software. For now we suggest that you only change the "payload" line, choosing a payload name unique to your project; in any case don't change anything unless you know what it does and why you are changing it! Later you will probably wish to set the radio frequency to avoid clashes with any other flights; the remaining options can probably remain as they are.

The tracker is configured on the Pi. Most of the configuration is then sent to the tracker board when the Pi tracker software starts, the exceptions being settings that only affect the Pi e.g. camera settings.

To edit the configuration from the Pi, using nano, type the following commands:

```
sudo nano /boot/flextrak.ini
```

The file is in sections

General Section

```
[General]
SerialDevice=/dev/ttyAMA0
PayloadID=CHANGEME
FieldList=01234569ABCD
```

SerialDevice is the serial port that connects to the tracker. No need to change this!

PayloadID is the name that your tracker will appear as on the live map. It's generally best to keep it short with a max of 6 characters, though it can be up to 32 characters.

FieldList is the list of fields (e.g. latitude, longitude) sent in the telemetry. Each character in the list represents a particular field to include in the telemetry. The fields are:

0	Payload ID
1	Sentence Counter
2	UTC hh:mm:ss
3	Latitude
4	Longitude
5	Altitude
6	Satellites
7	Speed
8	Direction
9	Battery Voltage mV
A	Internal Temperature C
B	External Temperature C
C	Predicted Landing Latitude
D	Predicted Landing Longitude
E	Cutdown Status
F	Last Packet SNR
G	Last Packet RSSI
H	Received Command Count
I	Programmable Field 0
J	Programmable Field 1
K	Programmable Field 2
L	Programmable Field 3
M	Programmable Field 4
N	Programmable Field 5

Landing Prediction

```
[Prediction]
LandingAltitude=200
DefaultCDA=0.7
PayloadWeight=1.0
```

The prediction is calculated during the flight based on the current position, winds measured during ascent, and the measured performance of the parachute during descent.

The prediction cannot be disabled, but you can remove the landing position (fields C and D) from the telemetry field list.

The default CDA and weight values equate to a landing speed of 5m/s, which is what we recommend. It doesn't matter if you have a lighter or heavier payload; so long as the landing speed is 5m/s then you can use the default values.

GPS Setting

```
[GPS]
FlightModeAltitude=2000
```

This is the altitude at which the GPS will switch from pedestrian mode (for more accuracy on the ground) to flight mode (to work above 18km). This just needs to be set to an altitude in metres that is comfortably above your launch and expected landing sites.

LoRa Radio Settings

```
[LORA]
Frequency=434.225
Mode=1
```

The default frequency (in MHz) should be changed if you expect to be flying at the same time as another balloon using the same frequency, in which case you should it should be moved by at least 50kHz i.e. to 434.175MHz or lower, or 434.275MHz or higher.

Mode 1 is best if you are using the Pi to download camera images; if not then Mode 0 is a bit better for range.

Camera Settings

```
[Camera]
High=2000
Rotate=False
LowFullWidth=2592
LowFullHeight=1944
LowFullPeriod=0
HighFullWidth=2592
HighFullHeight=1944
HighFullPeriod=0
LowRadioWidth=320
LowRadioHeight=240
LowRadioPeriod=0
HighRadioWidth=640
HighRadioHeight=480
HighRadioPeriod=0
```

When the tracker is above High metres, then the High___ settings are used, otherwise the Low___ settings are used.

The __Full__ settings are for full-sized images that are *not* sent over the radio; the __Radio__ settings are for smaller images that are sent over radio. Do not be tempted to substantially increase the size of the latter images as radio bandwidth is limited and you will not get many images downloaded during the flight.

The ___Period settings set the number of settings between images; somewhere around 15-30 seconds is good, and setting zero disables photography.

SSDV Settings

```
[SSDV]
LowImageCount=4
HighImageCount=8
```

These set the number of image packets per telemetry packet. For lower altitudes it's best to have more frequent telemetry updates (so a lower count) so you get a better final position as the flight comes in to land.

Remember that your changes will only take effect once the tracker program on the Pi is restarted. You can do this manually with these commands:

```
cd ~/flextrak
sudo killall startup
python3 tracker.py
```

Receiving And Decoding

All of the above has been about setting up the balloon tracker, and that should now be working, picking up its position by GPS and transmitting it by radio. How then do we receive that signal, decode it to get the position, and display it on a map?

This diagram shows the complete system:

The radio signal from the balloon contains data packets – telemetry or image data – encoded into the signal. This is picked up by an aerial which feeds it into a radio receiver which decodes the signal back into packets identical to those that were transmitted.

These packets are then read by a computer and uploaded to the internet where the telemetry is used to show the balloon position on a map, and the image packets are used to build up images. Typically, the sofware that does this also displays the telemetry (GPS data, temperature etc.) and often can provide additional information such as distance and direction to the payload.

A key feature of the system is that it works with multiple receivers, so you can set up a fixed tracker at one location (e.g. school) and a mobile tracker in a vehicle. The latter can upload its own position to the map so that people can see how close the chase vehicle is to the balloon. Additionally, if you inform the high altitude ballooning community of your flight, then those in your country will be very happy to help track your balloon.

Receivers

There are several options for receiving LoRa signals from your balloon, and the choice depends on where you want to set up receivers.

When you fly your balloon, by far the most important location for a receiver is with you when you chase it. That's because you will probably be closer to the flight as it lands than anyone else, and because you need to be able to find out exactly where it has landed. For this, a handheld receiver is ideal, and you can make one with our LoRaGo device and an Android phone or tablet.

It's also good to set up a fixed receiver, which you can use not only for your flight but also to receive other peoples' flights to get practice before it's time for your flight. LoRaGo is a option for this too, connected to a phone or tablet or Windows PC, or you can a permanent receiver with a Raspberry Pi and LoRa board.

So, your main options are:

- Mobile setup with LoRaGo and an Android device
- Fixed setup with Pi LoRa Gateway

Mobile Android Receiver - LoRaGo

For this you need:

- LoRaGo Device:

  https://store.uputronics.com/index.php?route=product/product&product_id=111

- Android phone or tablet with USB OTG capability (only old devices don't have that)

- HAB Explora which you can install from the Google Play Store:

  https://play.google.com/store/apps/details?id=com.daveakerman.HABExplora&hl=en

LoRaGo is a tiny USB-powered LoRa receiver ideal for use with FlexTrak:

LoRaGo comes with a small aerial but you can buy or build a small Yagi aerial then that will increase range substantially.

HAB Explora is an app for Android that provides the essential functions required when chasing a high altitude balloon (ideally one transmitting LoRa packets, but other modes can be integrated):

- Configures LoRaGo and receives telemetry/SSDV from it
- Display of received telemetry – GPS position etc
- Display of distance and direction to payload
- Display of map with balloon and chase positions
- Function to display driving route on the map
- Function to provide navigation to balloon using Google Map or Wyze etc
- Chase car upload to the distributed HAB network
- Backup reception of telemetry from the distributed network

Software Installation

You can request installation on any of your Android devices from this link:

https://play.google.com/store/apps/details?id=com.daveakerman.HABExplora&hl=en

Or, on the device, open the Play Store and search for "habexplora".
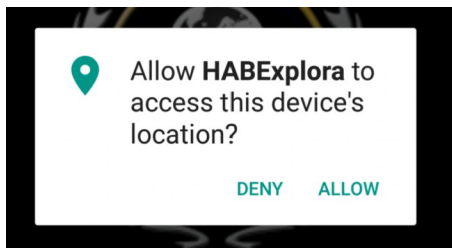
Usage

See this video on how to use the app:

https://www.youtube.com/watch?v=KoOb-ONnecE

Start the app (note: these screenshots are from an earlier version so you may notice some small changes and additions):

When it starts it will ask for permission to use the device's location; since this is needed for almost all program functions then it's best that you agree!

For USB, connect the LoRa receiver via the supplied USB OTG cable, taking care to connect the host end to the phone and the slave end to the receiver. Android will ask you if you wish to allow the app to use the device (again, you do!).
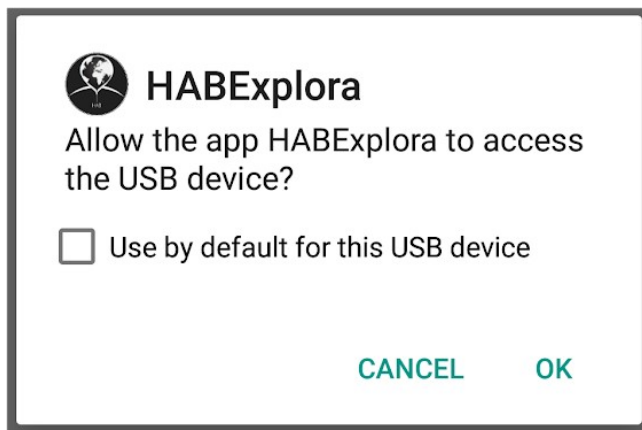
Check the box so that this app is automatically started when you reconnect this USB LoRa device.

Screen Elements

The app has 3 screen sections. At the top is the main menu with 4 buttons:

| Payloads | Direction | Map | Settings |

- Payloads – Shows received telemetry for up to 3 payloads
- Direction – Shows the direction and distance to the selected payload
- Map – Embedded Google map showing balloon(s) and chase device
- Settings – Shows the various setup screens

The bottom of the screen is for status:

| USB | BT | Habitat | GPS | 14:07:42 |

This shows the LoRa USB status, LoRa Bluetooth status, Habitat status and GPS status. These blocks are colour-coded with black-on-dim-yellow meaning not in use; red text shows error or no recent data; green shows good recent data. The bar also shows current UTC time from the phone's GPS. The circles, where present, show the upload status to Habitat (Green = Good, Red = Failed, Grey = Not in use).

The central area is for function-specific screens, and initially just shows a splash screen with app version number.
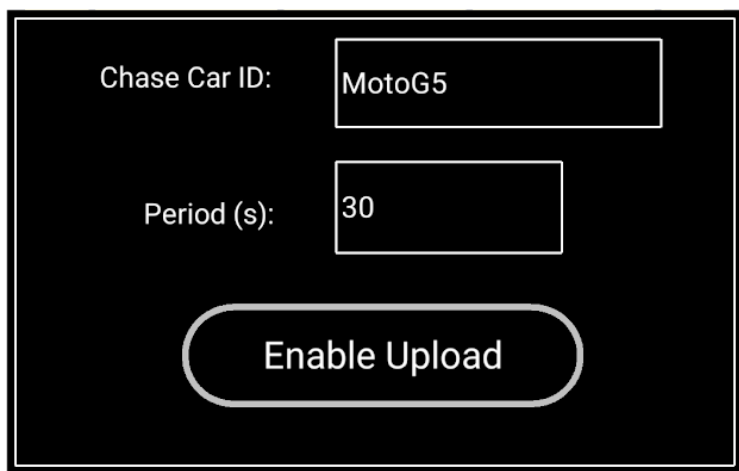
Settings

The first thing we need to do is set up a efw things, on the GPS and USB settings pages. So click on Settings and you will see:

- GPS – Controls upload of phone GPS position
- USB – Settings for the attached USB LoRa receiver
- BT – Setting for an attached Bluetooth LoRa receiver
- Other – Miscellaneous settings e.g. control telemetry downloads from Habitat

Touch the appropriate button to enter your desired section. Once there, make your adjustments and then touch "Apply" or "Cancel". We'' start with GPS:
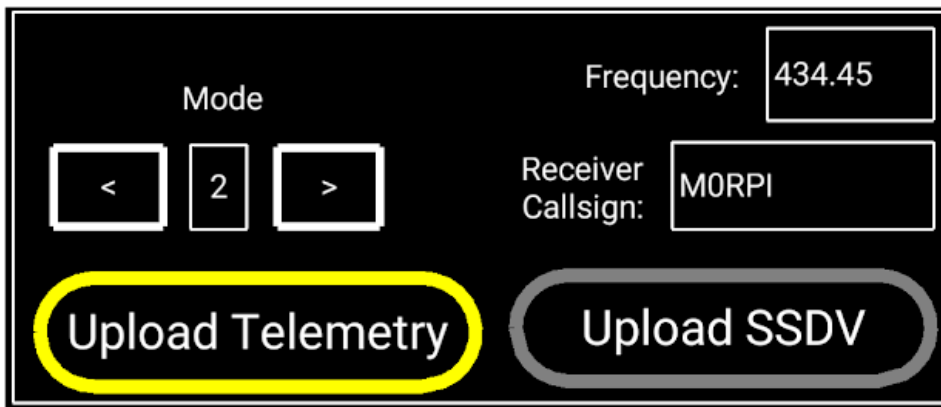
GPS Settings

"Chase Car ID" is what your position will appear as on the spacenear.us map, so use your name or device name or project or school name or, if you have one your ham radio callsign.

"Period" is how many seconds between uploads – 30s is good.

Touch "Enable Upload" (so it is highlighted in yellow) to enable the uploads. Your position will start to appear on the HAB map. You can switch the upload off for now, but remember to switch it back on for your flight.

LoRa USB Settings

"Frequency" is the centre frequency to set the receiver to. **Set it the same as set in your tracker.**

"Mode" is the LoRa mode to set from 0 to 7; most commonly 0 is used for telemetry, and 1 for SSDV with telemetry. **Set it the same as set in your tracker.**

"Receiver Callsign" is used when uploading telemetry and will appear in the listener section against the payload on spacenear.us.

The 2 buttons enable upload of telemetry and SSDV from the tracker.

Sources

You should now be receiving telemetry from the tracker, and you can check that on the Sources screen: touch the status bar on any of the source buttons (USB/Habitat/GPS).
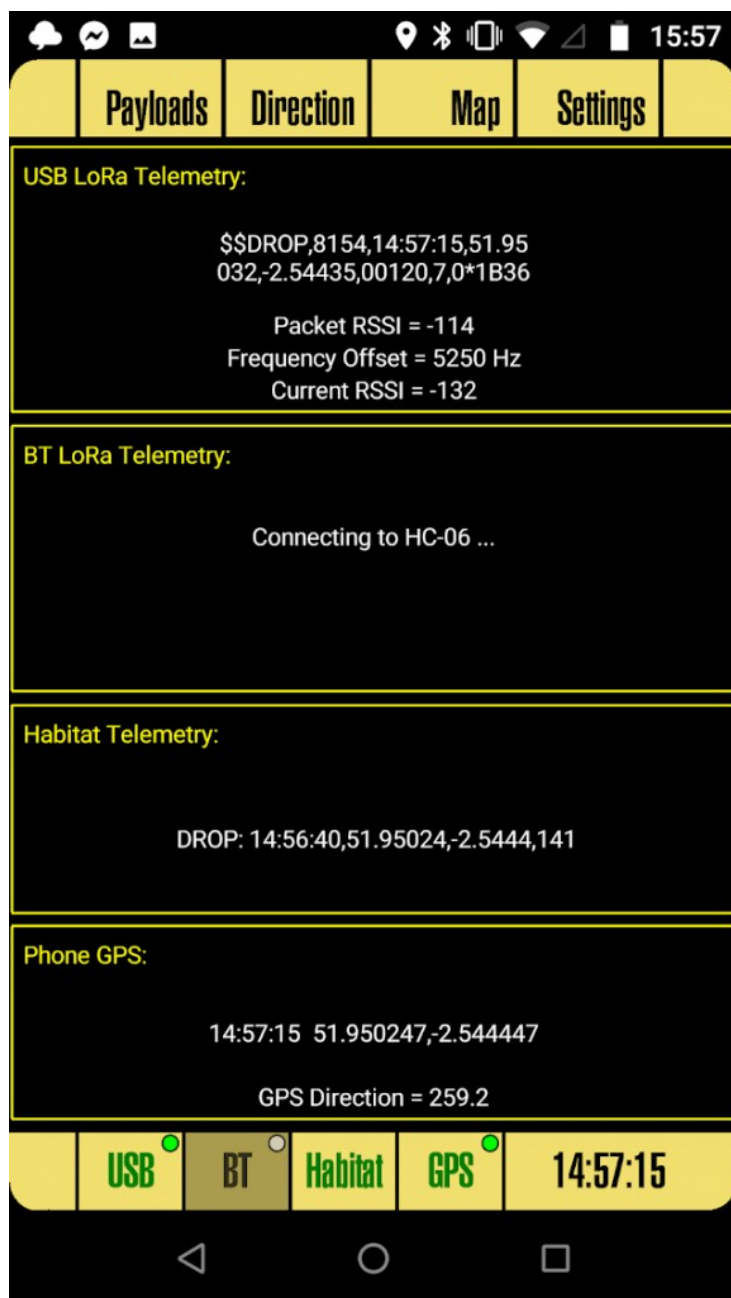
If you don't see telemetry in the top section (the one for LoRaGo) you may need to adjust the received frequency a little, so go to the Settings – USB screen and increase the frequency by 3kHz. If that does not work, reduce it down to 3kHz below the tracker frequency.

Payloads Screen

Up to 3 payloads can be displayed; if a 4th is heard then it will replace whichever of the other 3 has the oldest data. For your testing and flight of course you're only going to see one payload – yours!

Each displayed payload shows the payload ID, time since last message received, UTC time of that message (as sent by the payload), latitude and longitude, altitude (with maximum altitude in brackets), and ascent/descent rate.

If you do have multiple payloads then you can select between them by touching in the box for the desired payload; the selection will then have a thicker border than the others. Once done, the selected payload will be used on the direction screen and for navigation.
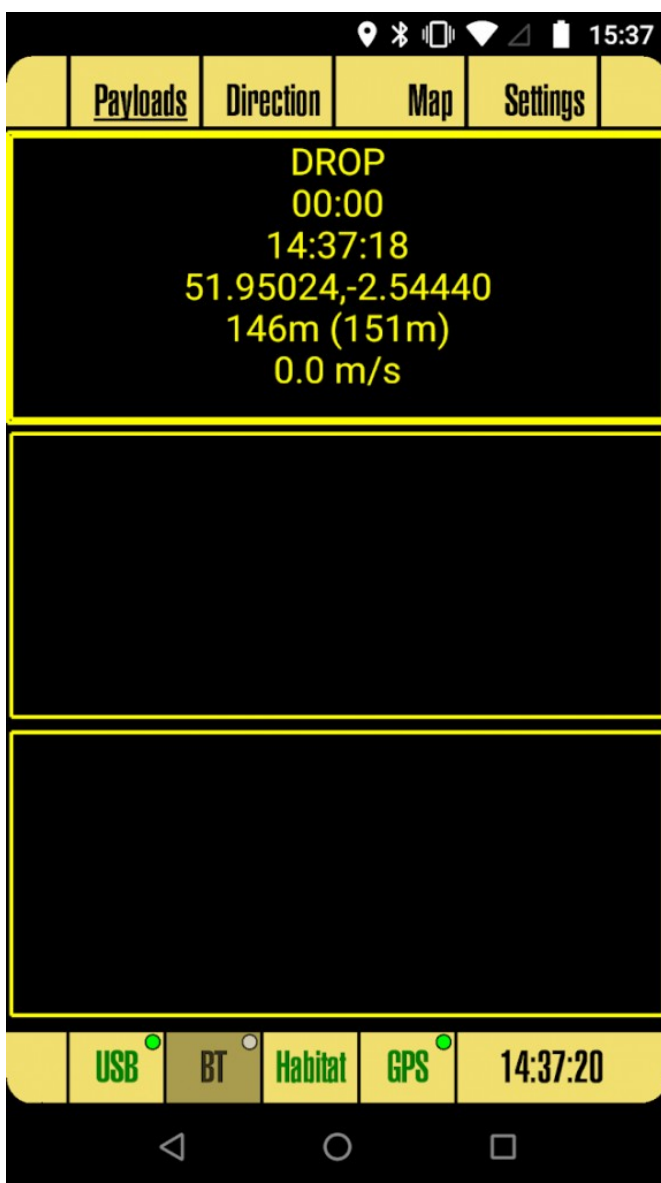
Direction Screen

This screen shows the distance and direction from your location to the latest position received from the payload.
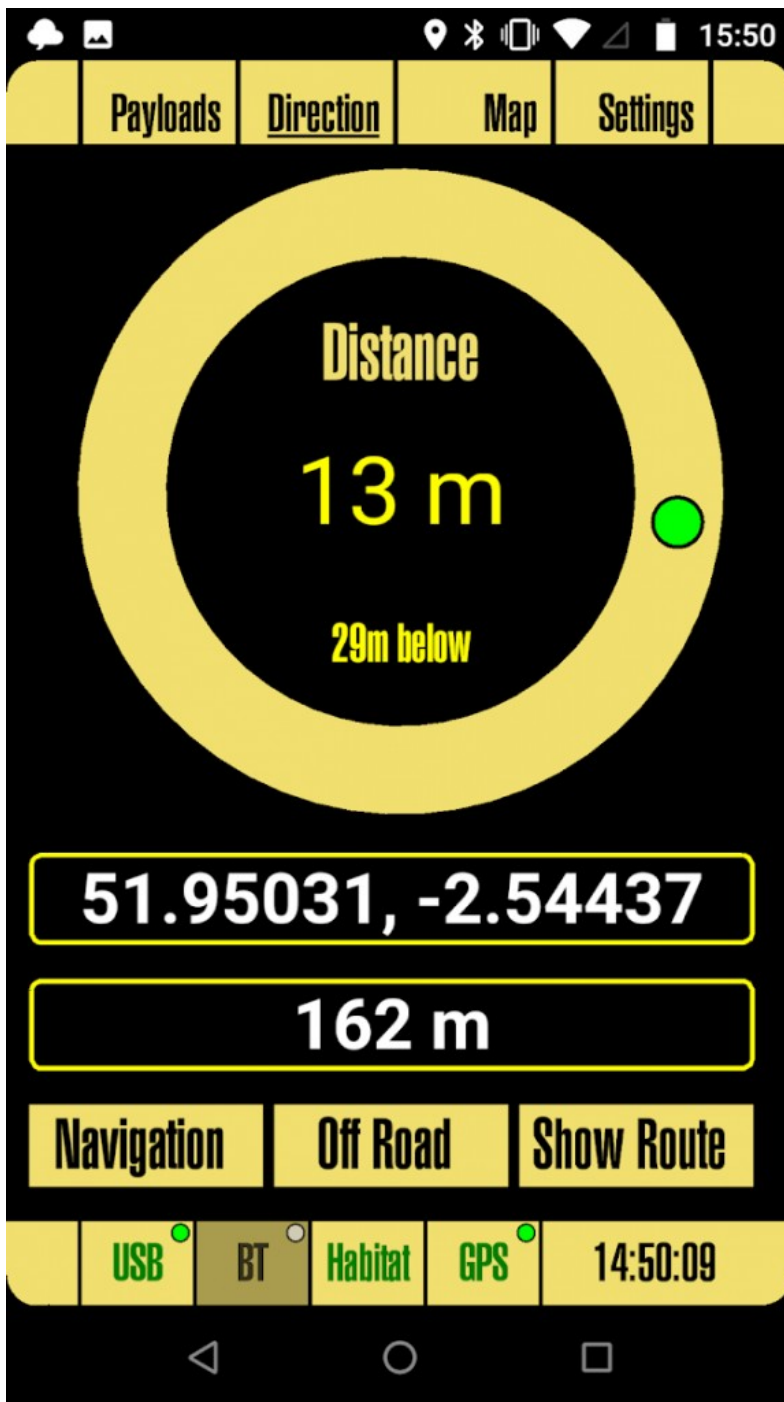
If your phone has a magnetic compass then that will be used; if not then the GPS "heading" value will be used. The former gives direction to the phone's orientation; the latter gives the the phone's direction of movement and for tor this to be accurate you need to be moving, so if you aren't then walk approx 10-20 metres in a straight line. If the phone isn't reporting a heading then the compass marker will be in red, and will show the payload direction relative to North (North = straight ahead).

Below the compass there are blocks for the payload latitude, longitude and altitude. Below these are buttons for navigation and map directions to the payload:

- Navigation invokes a navigation app. First time you do this Android will ask you to choose which app you want to use for navigation (e.g. Waze, Google Maps).
- Off Road is the same, but uses a different Android "intent" which means that Android will ask again for your choice and will remember that choice separately. So you can choose a different app e.g. Back Country Navigator or another OS mapping app.
- Show Route uses an embedded (i.e. within this app) Google Map to show road directions to the payload.

If you later want to change which apps are invoked from the Directions screen, go to Android Settings –> Apps, choose the existing app, then choose "Open by default", then "Clear defaults".

Map Screen

This shows an embedded Google map:
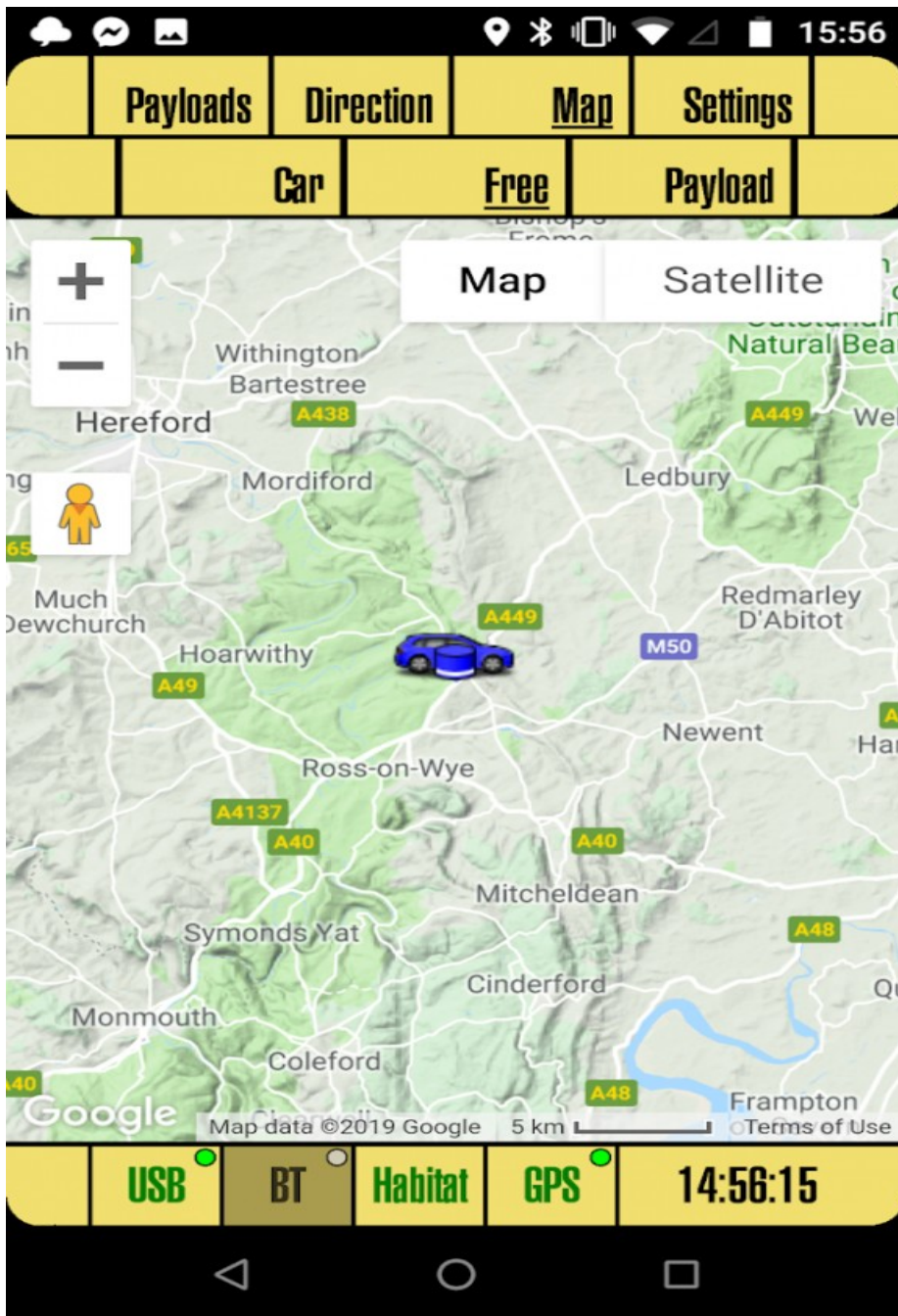
The map will initially centre on your location, but will then allow you to drag the map around at will. If you wish it to remain centred on your location then touch the "Car" button; if you wish it to centre on the payload then touch the "Payload" button.

Raspberry Pi Receiver - LoRa Gateway

To build a LoRa gateway you need:

1. An Uputronics LoRa board:

    https://store.uputronics.com/index.php?route=product/product&product_id=68

2. Raspberry Pi
3. Internet connection to the Pi (wired or wireless)
4. 70cm (434MHz) antenna

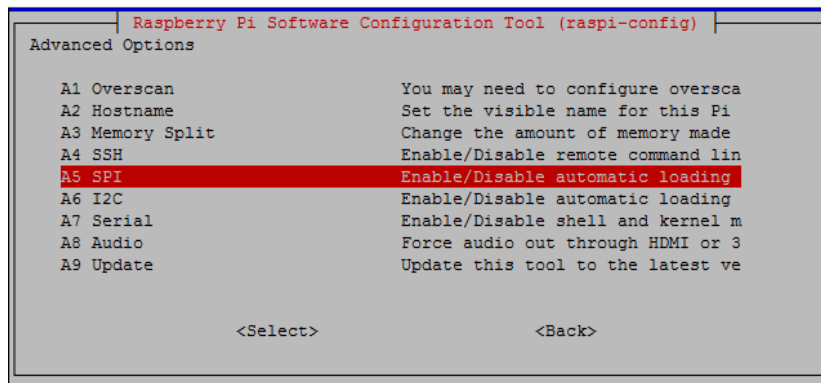Physical installation is very simple – just push on to the Pi using a standard pin header extender and secure with standoffs and bolts (all supplied).

Software Installation

Next, burn an operating system onto a suitable SD card. For this purpose anything from 8GB and up should be fine. The following instructions are for Raspberry Pi OS Lite.

First, run raspi-config:

```
sudo raspi-config
```

```
┌──────┤ Raspberry Pi Software Configuration Tool (raspi-config) ├──────┐
│ Advanced Options                                                      │
│                                                                       │
│    A1 Overscan              You may need to configure oversca         │
│    A2 Hostname              Set the visible name for this Pi          │
│    A3 Memory Split          Change the amount of memory made          │
│    A4 SSH                   Enable/Disable remote command lin         │
│    A5 SPI                   Enable/Disable automatic loading          │
│    A6 I2C                   Enable/Disable automatic loading          │
│    A7 Serial                Enable/Disable shell and kernel m         │
│    A8 Audio                 Force audio out through HDMI or 3          │
│    A9 Update                Update this tool to the latest ve          │
│                                                                       │
│                                                                       │
│                 <Select>                       <Back>                 │
│                                                                       │
└───────────────────────────────────────────────────────────────────────┘
```

Then choose Advanced Options –> SPI and enable SPI.

It's also worthwhile to change the hostname (Advanced Options –> Hostname) to something like "LoRaGateway". Finally, close the program and agree to reboot the Pi.

Once rebooted, login again. We now have some software to install. First, install wiringPi, which is used for the SPI library and to read the status of the LoRa module:

```
sudo apt-get install wiringpi
```

The gateway software uses the curl library for internet access (uploading telemetry data and/or image data), so install that:

```
sudo apt-get install libcurl4-openssl-dev
```

and the ncurses library used for the screen display:

```
sudo apt-get install libncurses5-dev
```

Finally, install the gateway software itself:

```
cd ~
git clone https://github.com/PiInTheSky/lora-gateway.git
cd lora-gateway
make

cp gateway-sample.txt gateway.txt
```

That completes the installation, so now for the configuration. The main settings are in a file gateway.txt in the above folder (/home/pi/lora-gateway). Here's a simple example:

```
tracker=MYCALLSIGN
frequency_1=434.450
mode_1=1
AFC=Y
```

This firstly sets your callsign, which if you are a radio amateur would normally be your radio callsign, but it can be something else.

The next part sets the frequency and mode for the second LoRa device (the one in position "CE1"). Frequency is in MHz and should match the frequency of your FlexTrak tracker. Same applies for the mode.

For full details on all settings see the README.md file.

For an aerial, you can make a quarter-wave aerial or buy something commercially, for which we recommend a collinear aerial, for example:
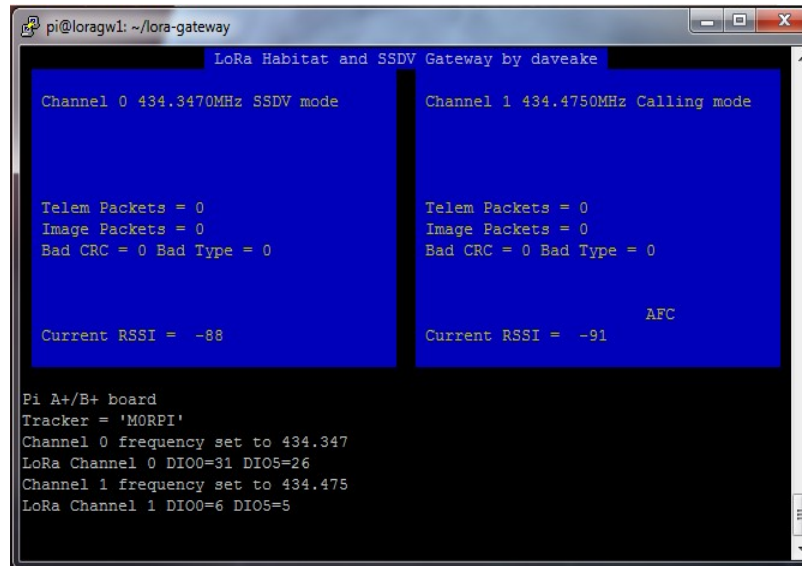
https://www.nevadaradio.co.uk/product/watson-w-30/

Ideally, get the aerial high and away from obstructions.

Usage

To run, just type

```
sudo ./gateway
```

and you will see a screen like this:
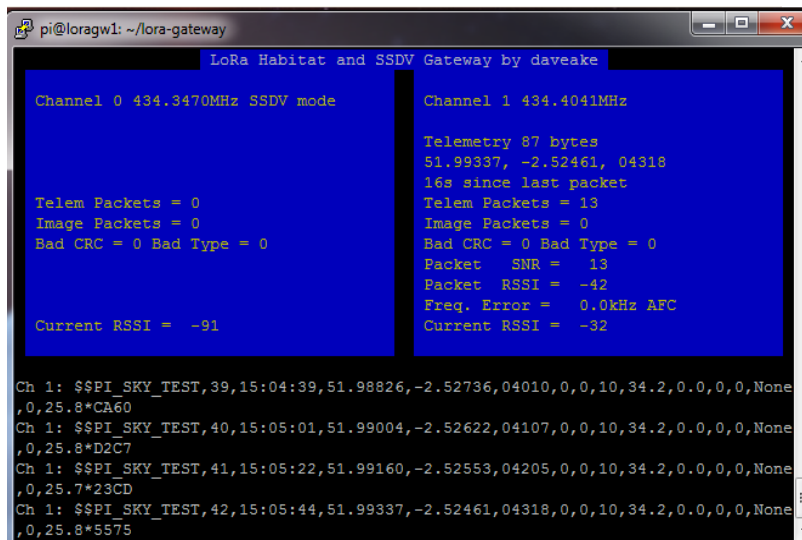
Now start up your tracker and, all being well, you should soon start to see the packets landing at the gateway:

Getting On The Map

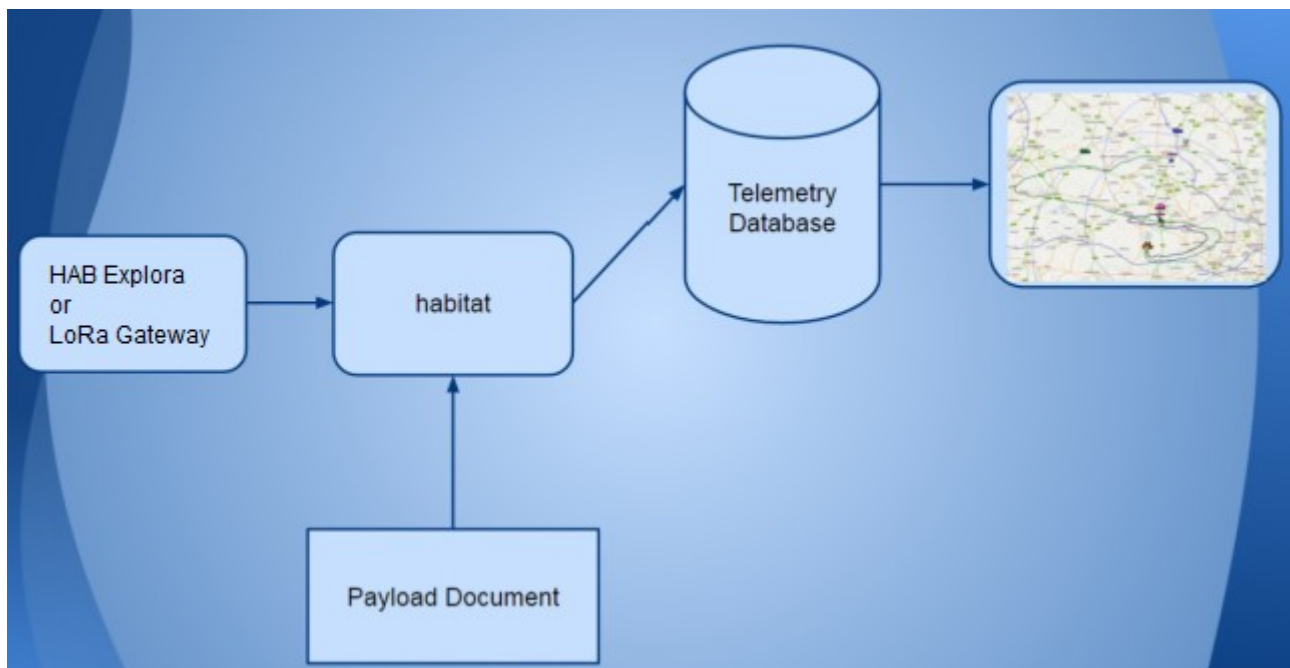The above diagram shows how data from HAB Explora or the LoRa Gateway gets onto the map. First, the receiving software connects via the internet to the "habitat" server. Multiple receivers may be doing this (remember, you may have one at school and one in your chase vehicle, plus the high altitude balloon community may be uploading too) so even if you miss some data in the chase vehicle (e.g. it's in a tunnel) then hopefully someone else will have uploading the missing data.

All uploaded data goes into a database, and the latest good data then gets fed to the live map at tracker.habhub.org. All of this though relies on that "Payload Document" box at the bottom.

The purpose of the payload document is to tell the habitat software how to interpret the incoming payload telemetry. This telemetry varies in structure from one payload to another, as some payloads might for example send 2 or 3 different temperature readings, a pressure reading, and perhaps humidity too, all in addition to the usual latitude, longitude and altitude. Every payload therefore needs a payload document otherwise it will not appear on the map.

habitat knows which document to use, by inspecting the start of the telemetry packet where it expects to find the "Payload ID". For FlexTrak this defaults to PIAVR, but you should have changed that by now as per the instructions earlier.

To create a payload document for your payload, start at this URL:

http://habitat.habhub.org/genpayload/

where you will see...

payload configuration documents

payload_configuration documents describe a payload (no prizes for guessing that). They contain instructions to habitat: how to parse telemetry that listeners upload; and radio settings: how to setup listeners' radios. If you fly one payload multiple times, you shouldn't have to change payload configuration document.

| create new |
| start from existing |

flight documents

flight documents describe the time and date of a planned flight. They allow us to add the flight to the calendar, set the title on the tracker, and select only relevant telemetry to go into the flight archive (i.e., exclude test data).

| create new |
| start from existing |
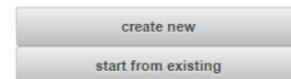
A flight document contains a list of payload configuration documents - i.e., the payloads that are going to be launched. You can add more than one, if there are several payloads on one balloon/flight train - but for multiple launches, create a flight document per balloon.

Click on "start from existing", as we are going to use an existing document as a starting point, to save time. Type in "FLEXTRAK" in the box, and you will see this:

| habitat | genpayload |

Database browser

Search prefix: flextrak

| Search | Prev Page | Next Page | Cancel |

| FLEXTRAK | FLEXTRAK | ae4f1f24c62b814939d029ba4bc2aa6e |
| | "FLEXTRAK SAMPLE CHANGE PAYLOAD NAME ABOVE AND IN BELOW CONFIGURA... | 2020-06-15T14:23:09+01:00 |

then click on the entry in the resulting list.

It is **vital** that you now change some settings to match your payload, otherwise your payload won't appear on the map and you won't be able to easily find the payload document later.

First, the "payload name"; I suggest you enter your payload ID here, followed by frequency and LoRa mode, which helps other people track your flight. Then change the description to something like "XYZ School HAB flight 2020"

Next, click the Edit button to the right of "Parser Configuration", to show this screen:

Set Description as on the previous screen.

Set Callsign to match what you set in flextrak.ini **otherwise this won't work and you payload won't appear on the map.**

If you've not added any extra sensors, then that's the only change you need to make; we describe how to add fields for extra sensors in the section about connecting those sensors. So, click Save to save the new payload ID, then Save on the next screen to save the payload document.

So, provided that:

- Your tracker is running
- It has a GPS lock (flashing green LED)
- You are receiving and decoding in HAB Explora and/or LoRa Gateway
- You checked the "Upload" button in HAB Explora, and/or set "Enable_Habitat=Y" in the LoRa gateway gateway.txt
- You have an internet connection
- Your payload document was saved
- Your payload document matches your tracker settings

then your tracker should appear on the map:

https://tracker.habhub.org/

If it doesn't appear within a few seconds, then check the logtail (http://habitat.habhub.org/logtail/) which should tell you what is wrong.

Flying

Of course, flying a balloon is a lot more than just getting your tracking working. Hopefully by now you've read through our guide to flying:

http://www.daveakerman.com/?p=1732

but if not, **do it now!**

To add to the advice in that document, we have some more specific topics to cover before you launch your balloon:

- Making a payload aerial
- Mounting the Pi camera
- Tracker battery power
- Payload Design
- Flight Document
- Launch Announcement

Making A Payload Aerial

We supply a "pigtail" which is a short length of coax cable with an SMA plug on the end to connect to the tracker. This is not **yet** an aerial; you need to adapt it to make an aerial.

To do this, see this guide: https://ukhas.org.uk/guides:payload_antenna

Or see this video for a solderless method: https://www.youtube.com/watch?v=xemKyV1H6Xs

Mounting The Pi Camera

The camera can be mounted inside or outside your payload, but inside is better as it physically protects the camera. If you do mount it outside make sure that all the edges are taped down so that the payload lines cannot get underneath the camera and remove it (this can happen when the balloon bursts and the line is no longer under tension).

Sometimes it is easier to mount the camera upside down, in which case the images will be inverted. To fix that, set "Rotate=True" in the [Camera] section of flextrak.ini file (see the configuration part of this document).

Tracker Battery Power

For flight you should power the tracker from 4 Energizer Lithium Ultimate AA cells.

We do not recommend other makes of similar cells as they do not perform as well.

Do not use alkaline cells instead; there's a very high risk of your tracker stopping during flight when the cells get too cold.

Do not use a USB powerbank instead; these also do not work well in cold temperatures.

4 AA cells will give a run time of about 20 hours which should be plenty for a flight of 2-3 hours.

Payload Design

The guide linked to above has a section about payload design so please read that.

For the FlexTrak tracker, the GPS antenna connected directly to the tracker itself. Since the GPS should be near the top of the payload box, away from other devices (e.g. video camera) that advice also applies to the tracker. So mount the tracker just under the top lid of the payload box, with the tracker oriented with the GPS at the top and the SMA socket below. The radio aerial SMA then screws into the SMA socket, and the radio aerial is mounted underneath the bottom of the payload box.

Flight Document

As far as the tracking is concerned, there's one more task to do before flying, and that's to create and have approved a "flight document". This document tells others where and when you are launching, and on what frequency etc. Depending on where you are located, you may then find that you have 15-20 others receiving, decoding and uploading your radio signals for you.

Do not make the flight document until you have made <u>and tested</u> your payload document. That's because it's if the payload document doesn't work (often the case if you haven't made one before) then the flight document will be invalid anyway and will be a waste of time.

<u>This does not remove or even reduce the need for you to receive your own telemetry data at launch (so you know it's working) or in the chase vehicle (so you can locate the payload after it has landed.</u> The other listeners will only be able to help once the flight is up above a certain altitude (which depends on the distance between them and the balloon, plus other factors) so <u>do not rely on them</u>.

To create the flight document. Go to this URL:

http://habitat.habhub.org/genpayload/

and choose "create new" next to "flight documents". Fill in the form (it should all be obvious what you need to do) making sure you click on "Add" next to "payloads in this flight", and add your payload document which you can search by payload name. When it's all saved you will be presented with a "document ID" <u>which you **must** keep a copy of.</u>

Flight documents need to be approved by request on IRC (Internet Relay Chat). Due to the attentions of a spammer, IRC now requires registration:

IRC

The HAB community uses IRC (Internet Relay Chat) for general discussion and getting flight documents approved. There are two such channels:

#highaltitude – this is for general chat about ballooning. It's the perfect place to ask any questions you may have, or to announce your flight.

#habhub – this is for getting flight documents approved

To join IRC, open the following URL in a browser:

https://web.libera.chat/

Type in your name in "Nick", **highaltitude** or **habhub** in "Channel", and then click "Start".

We highly recommend you join #highaltitude right now, introduce yourself, and discuss your flight especially any questions you still have. There's a vast amount of experience in flying and tracking balloons available, much more than you will get from any individual flyer, so take advantage of it.

Flight Document Approval

Do not do this until you have proven that your payload document works (i.e. your balloon appears on the map). Just remember this sequence:

1. Set up your tracker
2. Create a payload document for it
3. Upload telemetry from your tracker
4. Check that your balloon appears on the map
5. (if not, check logtail and fix the payload doc)
6. When you have a firm launch date, make the flight doc
7. Ask for that flight doc to be approved

To get approval, create yourself an IRC account (see above) and then join the #habhub channel. Then paste the flight document ID that you created earlier.

Within a few seconds, a bot will check that your flight document ID is valid. It will then display your project name etc., and will (hopefully) say something like "Payload ABC tested 2 days ago". If instead it says that the payload is "untested" then this means that the payload document referred to in your flight document has not been used to put a balloon on the map, so give yourself a slap for not following instructions and go back and check where you went wrong. Typically that would be either that you didn't get your balloon on the web map, or that you created multiple payload documents and referenced the wrong one from your flight document.

If this all sounds a bit draconian, it's there to help prevent people from flying without working tracking which, of course, would be a bad thing.

So, assuming that you got a response of "payload XXX tested ..." then you should await the attentions of an admin. The admins aren't around all the time, so you may not get an

immediate approval, so either stay connected and check later, or you can at anytime just repost the flight ID and the bot will say if it's been approved or not.

Flight Announcement

We also recommend that you join the **#highaltitude** IRC channel to introduce yourself to the ballooning community, and to tell people when your flight is planned to happen.

The channel contains many fellow balloonists, many of whom will be very interested in helping track your flight for you. Getting help in this way is a **very good thing**. Not only will they help fill in gaps in reception during the flight (so you will get better pictures than otherwise) you may well find that one of them gets a better final position than you do in your chase vehicle.

Also, it's a friendly place with a wealth of experience, so if there's anything you don't understand or wasn't answered in this document or the document linked to above (http://www.daveakerman.com/?p=1732) then feel free to ask.

It's especially useful if you connect to that channel when you are preparing the launch itself. You can ask any last-minute questions, let the community know how you're getting on, and then tell them when you're about to launch. They will then tune in to your payload's transmissions and help you track. They can be particularly helpful in telling you where to wait whilst the flight comes in to land, and offer advice on how to retrieve it.

If your first venture into #highaltitude is to say something like "I flew my balloon yesterday and now it's lost please help" (and this happens depressingly often) do not expect miracles; do expect some questions as to why you left it so late! The HAB community is especially helpful to newcomers but they can only help before and during your flight – afterwards is almost certainly too late. There's a high correlation between getting involved in the channel prior to flight and having a successful flight.

After The Flight

Reading Flight Images on a PC

After your flight, you will want to access the Pi photographs and maybe upload them to social media etc. The quickest way to do this is to pop the micro SD card into a laptop and read the images there.

However, neither Windows nor OSX can access the Linux partition on the Pi SD card without special software being installed. Here we discuss the Diskinternals Linux Reader program which works on Windows; there are other programs available for Windows and OSX.

First, download and install the software.

Insert the Pi SD card into your PC's card reader, using an adapter if necessary. Then run the Linux Reader program (if you insert the card after starting the program, then press F2 to refresh the drive list). You will see something like this:

Double-click on the "Linux Ext" partition under "Drives with Removable Storage", and not the entry in "Physical Drives". You will then see the contents of the partition:
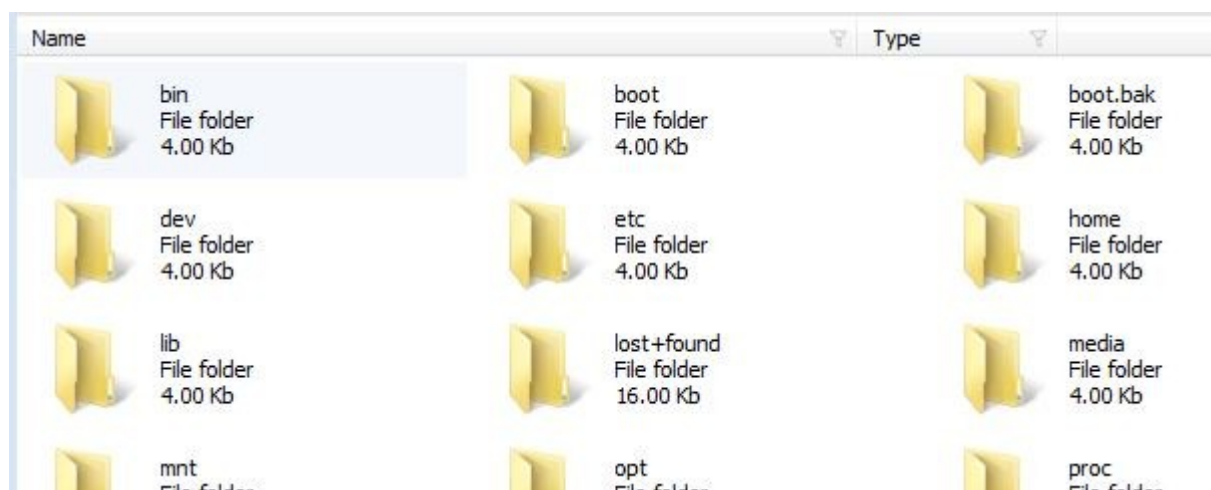
Open (by double-clicking again) the "home" folder, then the "pi" folder, then the "flextrak" folder, then the "images" folder. You will then see the following folders:

- FULL (for full-sized images)
- LORA (for images taken for the LoRa radio channel

Within each of those 2 folders mentioned, you may see a number of dated folders, one per date that the tracker stored images on. Note that the Pi only knows the current date and time if it is connected to the internet or if it has a GPS lock, so any photographs taken otherwise (i.e. disconnected from the internet and without a GPS lock) will have incorrect timestamps and will possibly be stored in the wrong dated folder.

You can view images directly in the program, simply by navigating to the folder you are interested in, and clicking on an image file.

To copy images to a PC drive, first select them (you can select all with CRTL-A or right-click on an image then choose Select All from the menu), and then clock the Save button on the toolbar. The program will then ask where to save the files.

APRS

Automatic Packet Reporting System uses digital amateur radio transmissions to send positional and other information. **To use APRS you must have an amateur radio licence.** In some countries (such as the UK), amateur radio transmissions are not allowed from an airborne device, so in those countries APRS **must not** be used on balloons. In other countries, such as the USA, airborne use is allowed and APRS is consequently a popular choice for ballooning.

Many radio amateurs run APRS receivers which either repeat incoming data packets by radio (digipeaters) or pass messages to APRS servers on the internet (i-gates). It's entirely possible for a packet to bounce from one digipeater to another until it reaches an I-gate and then gets transferred to a server.

APRS frequencies vary throughout the world, and it's vital that you choose a tracker that transmits on the correct frequency, otherwise your transmissions won't be picked up by other receivers.

Wherever you are, all APRS transmissions use the same frequency. This means that if a receiving station is within range of 2 or more transmitters that each transmit at the same time, then it is likely that it won't successfully receive any of them. With balloons, the range can be a long distance so even of the balloons are not close together, they can still conflict with each other. For these reasons, it's important to not transmit too often, particular when at altitude.

Another issue is that APRS packets will be repeated by any digipeaters within range of the transmitter. It's possible from a balloon to hit a number of digipeaters each of which will then repeat the packet (remember, on the same frequency), potentially onto further digipeaters (which most likely already heard the original transmission). APRS packets can define how much of this repeating is allowed, and it's good practice to strongly limit repeating when at altitude.

For balloons, we want its position to appear on a map. For APRS this is aprs.fi where you can see the current position and speed of your balloon, plus the track so far. APRS flights can also be imported into the habhub tracker map (ask in #highaltitude if you want this done); the advantage is that the habhub map shows the predicted landing position during descent.

To track a balloon via APRS, you should have your own receiver which, ideally, has an internet connection so that it can update the maps. All you need for this is:

1. VHF FM receiver
2. Laptop PC
3. Audio cable to connect receiver audio output to PC audio input
4. Decoding/uploading software

APRS Board Installation

Stack the APRS board **on top** of the FlexTrak board, so it contacts the headers on that board.

Configuration

As before, configuration is via /boot/flextrak.ini. Add or edit the APRS section in this file; to enable APRS transmissions you need to remove the "#" from the start of the callsign field, and fill in your radio callsign.

- `Callsign=<your_callsign>`. Replace <your_callsign> with your amateur radio callsign. **If you don't have one then stop right now and get one!** Transmission without a licence, whether it's from a balloon or just testing on the ground, is illegal. Also, you must use your callsign here and nothing else.
- `SSID=11.` "11" is the usual ID for balloons. You should change this if you're flying 2 balloons at the same time with the same callsign, so each flight has a different ID, but otherwise leave at the default Spinal Tap setting.
- `Frequency=144.8.` Set to the frequency used for APRS in your country.
- `Wide_Altitude=1500.` At altitudes lower than this, APRS packets are set to "WIDE1-1, WIDE2-1". For altitudes above this, see below.
- `HighUseWide2=N.` At altitudes higher than APRS_Altitude, APRS packets are set to "WIDE2-1" if this parameter is Y, or to no path at all (preferred) if set to N.
- `TxInterval=60.` This is the period in seconds between transmissions.
- `PreEmphasis=Y.` Enables pre-emphasis, which increases the modulation for 2200Hz frequencies vs 1200Hz. Most conventional radio receivers will assume pre-emphasis, so enabling it should improve range
- `Random=10.` Sets a random time (0 - 9 seconds in this example) to add to the time that each packet is sent; handy to avoid continuous conflicts with another flight using the same settings.
- `TelemInterval=0.` Number of position packets between sending telemetry packets (zero to disable). The telemetry includes:
  - Sequence Number
  - Satellites used
  - Internal temperature
  - Battery voltage

Receiving

Receiving APRS is both cheap and easy. You need:

- •144MHz (2m) antenna
- •144MHz receiver (e.g. cheap TV SDR; cheap Baofeng handheld transceiver)
- •Audio cable
- •PC
- •APRS decoding software

For a chase car, a dual-mode 2m/70cm magmount antenna is ideal. A suitable receiver is a Baeofeng UV-5R, or pretty much any radio scanner that has an FM (narrow, not wideband which is normally called WFM) mode. Connect to a laptop line-in socket using a suitable cable, on which you can use an APRS decoder such as Direwolf.

Alternatively, you can use an SDR and VNC software instead of the radio and audio cable.

Testing

For the following test, I used an AOR AR8000 scanner and Direwolf software on a Windows PC.

I set the scanner to 144.800 MHz (the APRS frequency used in Europe), and "NFM" (Narrow FM) mode.

I then configured the PITS software, adding my amateur radio callsign to the APRS section (see above), and then started the PITS software. Provided that the PITS tracker has a GPS lock, then it transmits one APRS packet every 1 minute (unless configured otherwise), and this shows up on the RPi monitor like so:
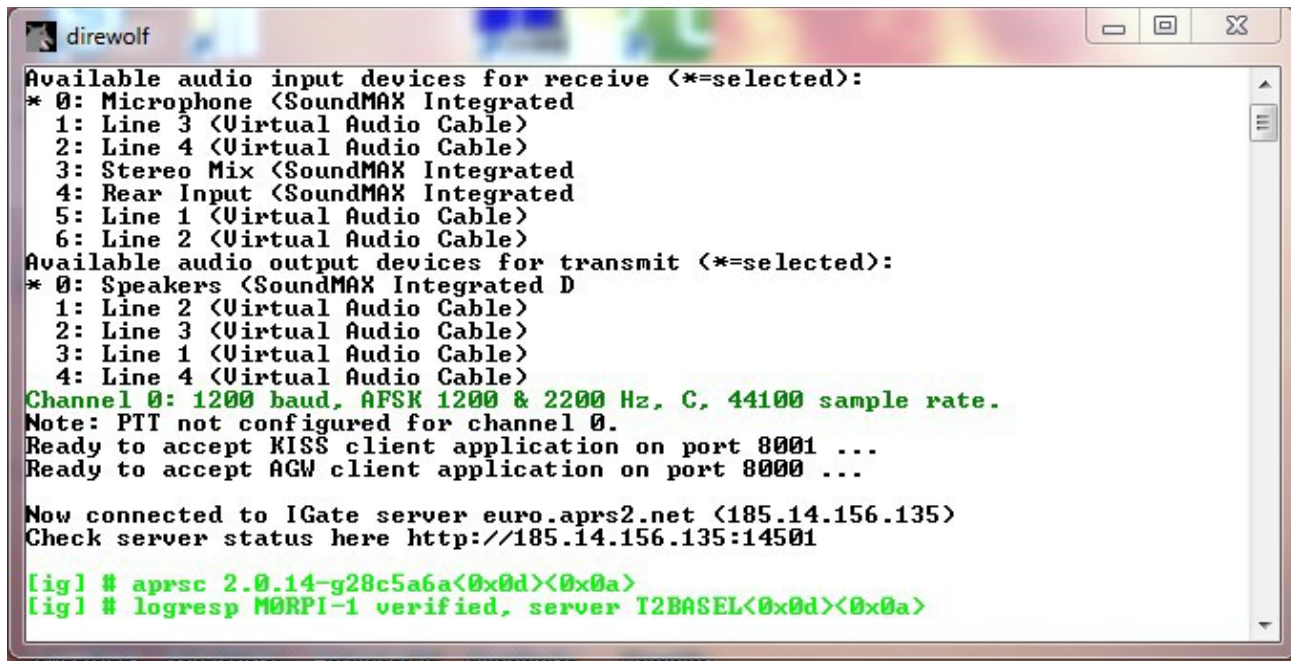
```
Sending APRS message ...
Length=100

cycles_per_bit=40
cycles_per_byte=320
Sending APRS Packet
Playing WAVE 'aprs.wav' : Signed 16 bit Little Endian, Rate 48000 Hz, Mono
```

When this happens, the FM receiver should play the packet. Once this was working, I connected the scanner audio output to the front-panel audio input on my PC, and installed Direwolf software. This software needs very little configuration; it needs to know:

- •Your amateur radio callsign
- •Your APRS passcode (which you can generate from your callsign online)
- •Which audio input device to use (i.e. the one that the radio is connected to)

With these items configured, I ran the software which started up like so:


```
direwolf                                                  [ _ ] [ ▣ ] [ ✕ ]
Available audio input devices for receive (*=selected):
* 0: Microphone (SoundMAX Integrated
  1: Line 3 (Virtual Audio Cable)
  2: Line 4 (Virtual Audio Cable)
  3: Stereo Mix (SoundMAX Integrated
  4: Rear Input (SoundMAX Integrated
  5: Line 1 (Virtual Audio Cable)
  6: Line 2 (Virtual Audio Cable)
Available audio output devices for transmit (*=selected):
* 0: Speakers (SoundMAX Integrated D
  1: Line 2 (Virtual Audio Cable)
  2: Line 3 (Virtual Audio Cable)
  3: Line 1 (Virtual Audio Cable)
  4: Line 4 (Virtual Audio Cable)
Channel 0: 1200 baud, AFSK 1200 & 2200 Hz, C, 44100 sample rate.
Note: PTT not configured for channel 0.
Ready to accept KISS client application on port 8001 ...
Ready to accept AGW client application on port 8000 ...

Now connected to IGate server euro.aprs2.net (185.14.156.135)
Check server status here http://185.14.156.135:14501

[ig] # aprsc 2.0.14-g28c5a6a<0x0d><0x0a>
[ig] # logresp M0RPI-1 verified, server T2BASEL<0x0d><0x0a>
```

Note the "*" against the active audio input device. If this is on the wrong device then close the program, reconfigure, and try again. Once it's correct, then wait for the PITS tracker to send its position again. When this happens, the result should appear in Direwolf. If it doesn't then try turning the radio volume up. Here's what you should get:

```
M0RPI-11 audio level = 27    [NONE]
[0] M0RPI-11>APRS,WIDE1-1,WIDE2-1:!/46;uMnLAO    /A=000469!!$!  /M0RPI,3'C,http:/
/www.pi-in-the-sky.com
Position, BALLOON, Generic, (obsolete. Digis should use APNxxx instead)
N 51 57.0054, W 002 32.6703, alt 469 ft
!!$!  /M0RPI,3'C,http://www.pi-in-the-sky.com
```

At this stage, the balloon position has been uploaded to the APRS infrastructure, and appears on the APRS map. To find it, type in your tracker callsign (including the SSID - e.g. M9XYZ-11 - in the "Track callsign:" search box. Here's what I saw when I did this:

FAQ

Why doesn't my Pi boot?

Make sure the SD card is properly inserted; if in doubt remove it and re-insert. The green "ACT" light should flash after power-up, and then flicker as the Pi boots; if not then you will probably see that it is very weakly lit, which means that the Pi cannot boot from the SD card. Next most likely issue would be the power supply.

What batteries do you recommend?

AA size Energizer Lithium cells.   These are specified to work at down to -40C and are known to be reliable in high altitude balloon flights.

Do not use other types of cell. Do not use a Powerbank.

What run time should I expect?

About 19-20 hours for FlexTrak with the above cells, assuming no extra boards connected.

It all works except there are no images being downloaded

Check that the camera is connected, that images are configured correctly in flextrak.ini. Also check that the SD card isn't full.

If you made your own SD card then check that the camera is enabled in raspi-config, and that you didn't skip the installation of SSDV.

It doesn't appear on the map

Check that you have GPS lock (i.e. the telemetry includes your position), that your receiver is set to upload to the internet, and that you have created a payload document for your payload ID.

Manual Installation

SD Image

This section is in case you need to replace the supplied pre-programmed SD card.

Your first task is to place an operating system onto a Micro SD card, so the Pi can be booted. Any card 8-32GB will do fine; we use Sandisk Ultra but there are other makes and models of course. **Always use a reputable make from a reputable source**.

First, download the latest a Raspbian Lite image from the RPi download page, following these instructions. We *do not* recommend using the full version of Raspbian as it boots into X (a graphical screen) by default, and our instructions assume otherwise; it is also rather large and will take up a lot of space on your SD card.

If you use a different distribution and encounter errors then we may not be able to help.

The following instructions work for the Lite version of Raspberry Pi OS. We test each new major version as they are released. Please note that Raspberry Pi OS changes frequently, and sometimes changes in the operating system can require us to change these instructions. We suggest you follow our twitter feed @pitsproject for the latest news.

You should download the latest Raspberry Pi OS image from the official Raspberry Pi website:

https://www.raspberrypi.org/software/operating-systems/

and follow the instructions linked to on that page.

There are separate instructions depending whether you are using Linux, Windows or a Mac to write the image to your SD card.

When you are done, follow the instructions in the next section to boot the operating system on your Raspberry Pi.

Initial Boot

Insert the SD card into an Raspberry Pi.

1. USB keyboard to any USB socket
2. Monitor to the HDMI socket
3. Wired network to the RJ45 LAN socket*
4. (do this last) Connect an approved power adapter to the Pi USB power socket

* For the Pi Zero and Pi A+ boards, use a USB-LAN adapter with spare USB socket for your keyboard.
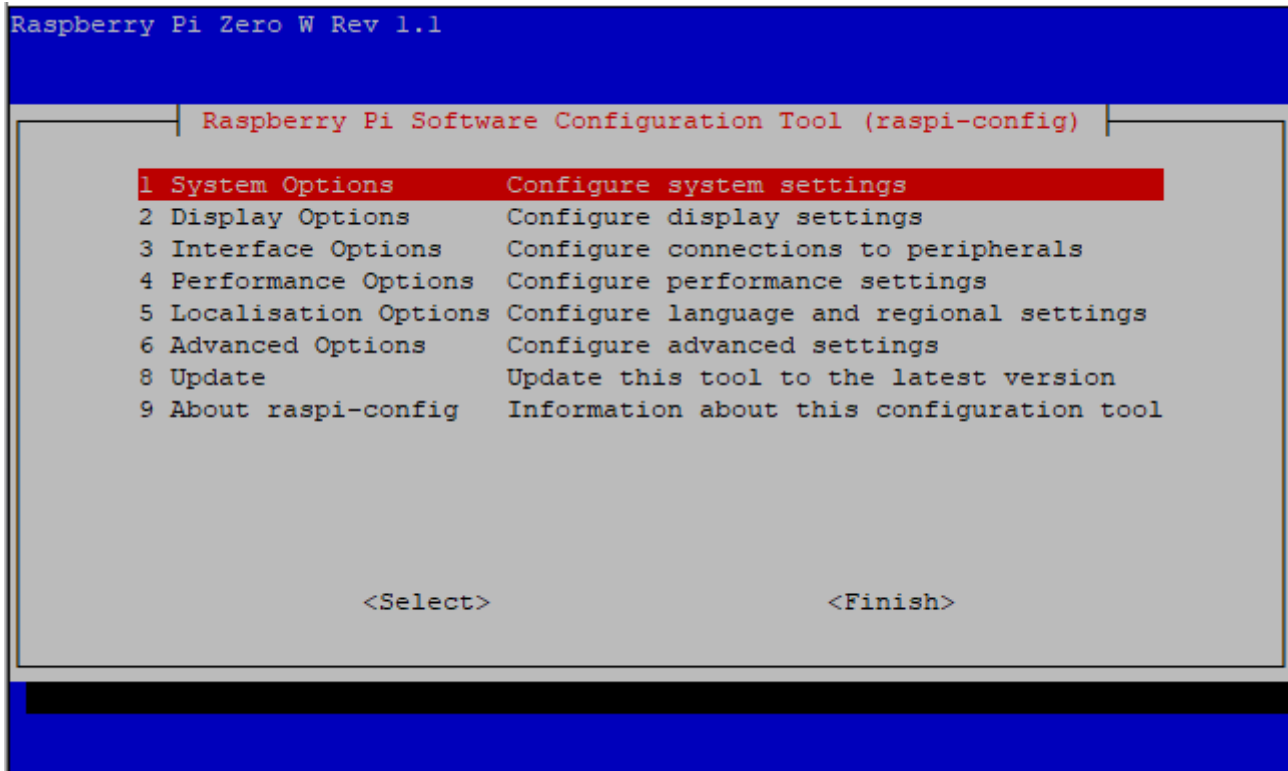
You should initially see a rainbow pattern on the monitor, followed by a lot of scrolling text. Eventually that should stop scrolling and you will see a "Login:" prompt. Type in **pi** and press ENTER. You will then be asked for a password, which is **raspberry**.

raspi-config

Once logged in, type the following command:

```
sudo raspi-config
```

and you will see the following screen (exact contents can vary with version):

```
Raspberry Pi Zero W Rev 1.1

     ┌─────────── Raspberry Pi Software Configuration Tool (raspi-config) ───────────┐
     │                                                                               │
     │     1 System Options       Configure system settings                         │
     │     2 Display Options      Configure display settings                        │
     │     3 Interface Options    Configure connections to peripherals             │
     │     4 Performance Options  Configure performance settings                   │
     │     5 Localisation Options Configure language and regional settings         │
     │     6 Advanced Options     Configure advanced settings                      │
     │     8 Update               Update this tool to the latest version           │
     │     9 About raspi-config   Information about this configuration tool         │
     │                                                                               │
     │                                                                               │
     │                                                                               │
     │              <Select>                          <Finish>                      │
     │                                                                               │
     └───────────────────────────────────────────────────────────────────────────┘
```

Using the cursor keys to navigate the menu, and the ENTER key to select an option, make the following changes:

```
1. Interface Options → Legacy Camera → Enable legacy support
2. Interface Options → Serial Port → Login (No) → Hardware (Yes)
```

When you have made those changes, choose the Finish option and you will be asked if you want the Pi to reboot – answer Yes. Wait for that to complete, then login again.

Serial Port

This section is ONLY needed by users with Pi that has built-in Bluetooth (Pi Zero W, Pi 3, Pi 3A+, Pi 3B+, Pi 4B.

Built-in Bluetooth needs to be disabled so that the full serial port can be used by the FlexTrak board. **You therefore cannot use the Bluetooth module with FlexTrak.**

You need to disable the bluetooth module, and map the now-free serial port to the GPIO pins. To do this, edit /boot/config.txt, using nano or the editor of your choice:

```
sudo nano /boot/config.txt
```

then go to the end of the file and add a new line containing this:

```
dtoverlay=pi3-disable-bt
```

Exit the editor, saving your changes.

Now type this command:

```
sudo systemctl disable hciuart
```

Install Updates

Raspbian Updates

The the Pi rebooted and logged in, run these commands (after each command, press ENTER then wait for the command to complete).

```
sudo apt-get update
sudo apt-get upgrade
```

The second command may take a while.

Software Installation

Run these commands:

```
sudo apt install git
sudo apt install ssdv
sudo apt install minicom
git clone https://github.com/daveake/flextrak.git
sudo apt install python3-picamera
sudo apt install python3-serial
cd flextrak
sudo cp flextrak.ini.sample /boot/flextrak.ini
sudo nano /etc/rc.local
```

change the first line in the file to be:

```
#!/bin/bash
```

then add the following line near the end of the file, before the last line (exit 0):

```
/home/pi/flextrak/startup
```

and save the file.

This completes the software installation.

Updates

Software Update

First, switch the programming switch on FlexTrak to the "P" (Programming) position.

Run these commands to download the latest firmware from the internet and then program the flextrak board:

```
cd ~/flextrak
git pull
```

Firmware Update

To program the device you need:

- ST-LINK V2 programmer
- Connector wires between programmer and extension connector on FlexTrak board

You need to connect the SWDIO, SWCLK and GND pins. **Do not connect any power (Vcc) pin**.

The simplest method is to use the STM firmware programmer software, load the .bin file from our repository, and load onto the FlexTrak board.

Software

If you wish to see the software running, you need to start it manually. Login to the Pi (using a screen and keyboard, or ssh connection) then type the following commands:

```
cd flextrak
sudo killall startup
sudo killall python3
python3 tracker.py
```

The screen should look something like this:

```
Load tracker ...
FlexTrak Module Loaded
Loading config file flextrak.ini
Start tracker ...
AVR module init
Comms module started
AVR module connected
Comms thread
Loop ...
TX: CH0
*
TX: CV
*
TX: CPPIAVR
*
TX: CF01234569ABCD
```

As you can see, the Pi software is written in Python. If you wish you can edit the software. Remember that even with the Pi software stopped, the tracker keeps running and you will still be able to receive its telemetry.

The Pi software is built as a module (flextrak.py), which itself uses several other modules (e.g. lora, gps). There is also a sample tracker program tracker.py which loads the flextrak module and passes the configuration file flextrak.ini to it. **If you want to modify the tracker, then tracker.py is the best one to change.** If for example you want to add extra sensors, then load the module(s) for those sensors in tracker.py, build a partial telemetry string with the values from the sensors, and then pass this string to the flextrak module. It provides a "callback" precisely for this purpose, which means that you can simply return the telemetry string is the result of that callback function.

Serial Protocol

If you wish to write your own software to communicate with the FlexTrak board, either from the Raspberry Pi that it is attached to, or from a PC via an FTDI USB cable, then you need to understand the serial protocol that it uses.

The protocol is based on a set of commands sent from the host (Pi or PC), which cause an action within FlexTrak. Commands all begin with the tilde character "~" and terminate with a CR or LF or CR-LF. The minimum response is a "*" for "command accepted" or "?" for "command rejected", both terminated with CR LF. Some commands have extra responses (see below).

There are also messages sent asynchronously from FlexTrak, for example new GPS positions.

These are the commands:

Function	CMD	Parameters	Response	Ver
Show GPS	~GP	Non Zero = Enable GPS to host	* GPS=xxxx	1
Flight Mode Altitude	~GF	Altitude in metres (max 7999)	*	1
Host Priority Mode	~CH	1 = prioritise host. Times out after 2s	*	1
Set Payload ID	~CP	Payload ID	*	1
Set Field List	~CF	List of field numbers each single digit 0-9 A-Z	*	1
Reset to Defaults	~CR		*	1
Save Settings	~CS		*	1
Show Version	~CV		VER=V1.20 *	1
Set LoRa Frequency	~LF	Frequency in MHz	*	1
Set LoRa Bandwidth	~LB	Bandwidth 7K8, 10K4, 15K6, 20K8, 33K25, 41K7, 62K5, 125K, 250K, 500K	*	1
Set LoRa Error Coding	~LE	Error Coding 5-8	*	1
Set LoRa Spreading Factor	~LS	Spreading Factor 6-12	*	1
Set LoRa Implicit/Explicit Mode	~LI	Not-Zero = Implicit; 0=Explicit	*	1
Set LoRa Low Rate Optimisation	~LL	Not-Zero = On; 0=Off	*	1
Set LoRa TDM Cycle Time	~LT	Cycle time 0 (off) to 60 seconds	*	1
Set LoRa TDM Slot	~LO	Slot -1 (off), 0 to 59	*	1
Set APRS Callsign	~AP	callsign max 6 chars	*	1
Set APRS Frequency	~AF	134 to 174MHz	*	1
Set APRS SSID	~AS	0 to 14	*	1
Set APRS Path Altitude	~AA	altitude above which wide2-2 is used	*	1

Enable Widw2-2 Mode	~AW	Non zero to enable		*	1
Set APRS interval	~AI	Interval in seconds		*	1
Set random APRS interval	~AR	Random +/- delta in seconds		*	1
Enable APRS pre-emphasis	~AM	Non-zero to enable		*	1
Set APRS telemetry count	~AT	Number of regular packets between telemetry		*	1
Clear SSDV Buffer	~SC			*	1
Append hex to SSDV buffer	~SP	partial packet in hex		SSDV=<length> *	1
Append binary to SSDV buffer	~SB	32-byte partial packet in binary		SSDV=<length> *	1
Get SSDV buffer status	~SS			SSDV=<length> *	1
Set SSDV image count	~SI	<low image count>,<high image count>,<high in m>		*	1
Fill in extra field 0-5	~F0 thru ~F5	<field_value>		*	1.2

Messages from FlexTrak are:

Command	Parameters	When	Version
GPS	dd/mm/yyyy,hh:mm:ss,lat,lon,alt,sats	Every 1s	1
SSDV	used length of SSDV buffer	After ~SP / SB / SS	1