Pi In The Sky Installation Manual

This document covers everything you need to know in order to build, configure and receive telemetry/imagery from your own GPS/radio tracker, using the Pi In The Sky (PITS) board or PITS Zero and a Raspberry Pi computer. It also covers the use of add-on boards (APRS, LoRa) and add-on sensors (DS18B20, BMP180, BME280).

Introduction

This is the PITS board:

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

The board contains the essential components to build a balloon tracker – a GPS receiver (top-left) so that the tracker knows where it is, a radio transmitter (centre-left) so it can transmit that information down to the ground, and a DC-DC power converter (bottom-left) so that it can be powered from batteries. There's also an ADC (Analog to Digital Converter) to monitor battery voltage and current consumption, a temperature sensor, connections to add extra sensors, plus of course the "GPIO" connector so the board can be plugged into a Raspberry Pi board.

The connector is a special "stacking" connector, so that you can add extra boards as well as this one. For example, you could add an APRS board, LoRa board or the Raspberry Pi Sense HAT. Or, if you're feeling adventurous, all of them (yes, it does work).

Because the board is stackable, it is easy to put it on upside down. Please don't do that! Nobody has yet managed to damage the board or the Pi by doing so, but there have been a couple of attempts. However, just follow the steps in the next section, and if in doubt check on the board for the "This Way Up" label!

We also have a PITS Zero tracker, which is smaller and lighter than the original board, and incorporates both RTTY and LoRa radios. It is not however designed for stacking – for that you should use the original board which has a more powerful PSU.

SD Image

This section is not needed by PITS Zero users – they are supplied with a pre-built 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 Raspbian Stretch Lite. We test each new major version as they are released. Please note that Raspbian changes frequently, and sometimes changes in the operating system can require us to change these instructions. We suggest checking http://www.pi-in-the-sky.com/index.php?id=sd-card-image-from-scratch for any updated instructions, and that you follow our twitter feed @pitsproject for the latest news.

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

https://www.raspberrypi.org/downloads/raspbian/

and follow the instructions linked to on that page:

https://www.raspberrypi.org/documentation/installation/installing-images/README.md

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

This section is not needed by PITS Zero users – they are supplied with a pre-built SD card.

<u>Do not connect the PITS board yet.</u>

Insert the SD card into a Pi model B+ or B V2/V3, or an A+ with USB LAN adapter, then make these connections to the 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 Micro USB power socket

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**.
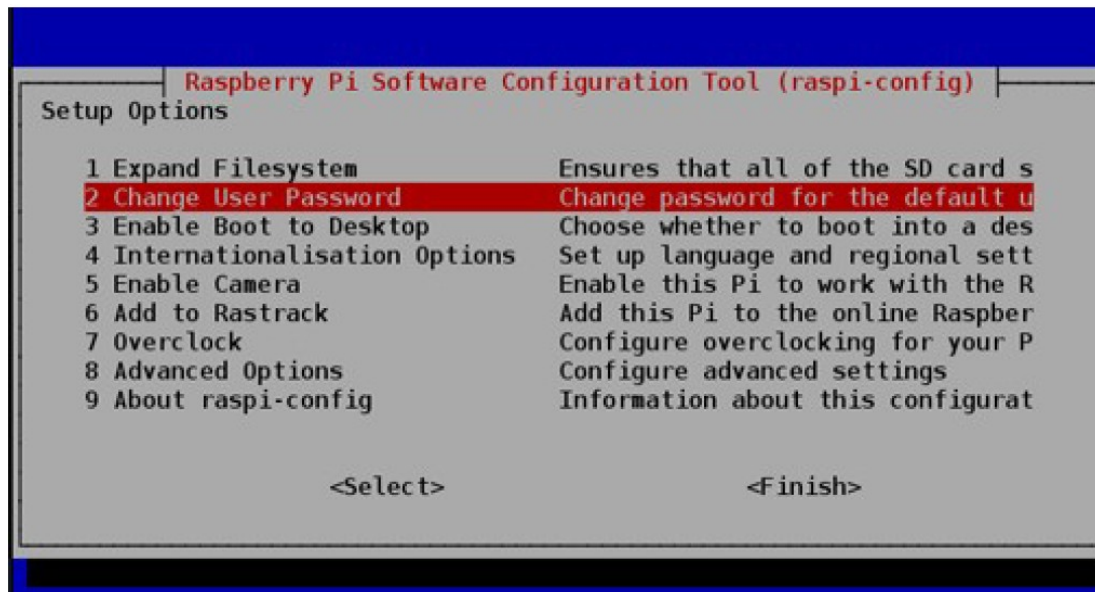
raspi-config

This section is not needed by PITS Zero users – they are supplied with a pre-built SD card.

Once logged in, type the following command:

```
sudo raspi-config
```

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

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

```
1. Enable Camera
2. Advanced Options --> Enable SPI
3. Advanced Options --> Enable I2C
4. Advanced Options --> Disable Serial Login / Enable Serial H/W
5. Advanced Options --> Enable One Wire
```

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 not needed by PITS Zero users – they are supplied with a pre-built SD card.

This section is ONLY needed by users with Pi that has built-in Bluetooth (Pi Zero W, Pi 3 , Pi 3A+, Pi 3B+. We do not recommend the 3B or 3B+ for use in flight due to high power consumption and temperatures.

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

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

This section is not needed by PITS Zero users – they are supplied with a pre-built SD card.

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.

Git

Unless you installed the full-fat Jessie distribution, then you will need to install git with this command:

```
sudo apt-get install git
```

PIGPIO

```
cd
wget https://github.com/joan2937/pigpio/archive/master.zip
unzip master.zip
cd pigpio-master
make
sudo make install
```

The first "make" step takes some time to complete.

Wiring Pi

Next, install the excellent "Wiring Pi" software from Gordon Henderson. This provides a command-line app "gpio" which we can use to test the PITS board, plus some C libraries which the PITS software uses to access the PITS hardware. Install using this command:

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

Install SSDV Software

Now install the equally excellent SSDV (Slow Scan Digital Video) software from Phil Heron. This provides a command-line app which converts between JPG and SSDV formats. In our case we convert from JPG to SSDV and then transmit the resulting packets over the radio link. Install using this command:

```
sudo apt-get install ssdv
```

fswebcam

This is only needed if you wish to use a USB webcam instead of the Pi camera

```
sudo apt-get install fswebcam
```

Tracker Software

The PITS tracker software is on github, and can be installed as follows:

```
cd
git clone https://github.com/PiInTheSky/pits.git
cd pits
./build
```

The build process compiles and links the tracker program, creates a default configuration file, and sets the software up to start automatically when the Pi boots.

This completes the software installation. You just need to shut down the Pi:

```
sudo halt
```

Wait for the Pi to shut down (the screen will scroll some text first, and then go blank). Remove power and other cables from the Pi, and you are ready to connect the PITS board.

Assembly

This section is not needed for PITS Zero which is supplied assembled.

As well as the PITS board, spacers, bolts and pin header (all supplied in the kit), you will need a Raspberry Pi (A+ or B+) and a Pozidrive screwdriver:

Note: If you are using a Pi camera, connect the camera cable to the Pi first. The cable inserts into the connector situated between the Ethernet and HDMI ports, with the silver connectors facing the HDMI port. Do not connect the camera itself at this time. The pictures here do not show the cable.

First, place the two supplied 11mm stand-offs in the Raspberry Pi board. Put them in the 2 holes opposite to the header, screw them in tightly.

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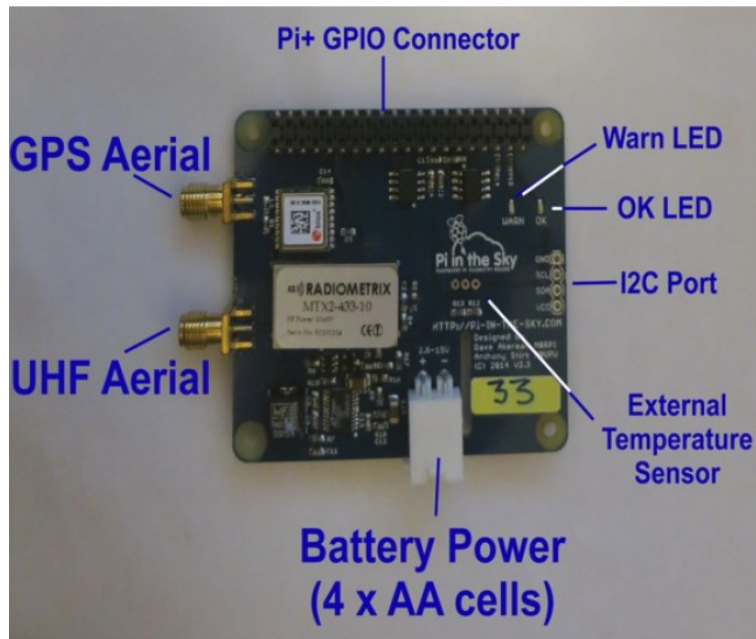
Now place the PITS board on top, ensuring that it is the correct way up (i.e. logo and components on the top).

Finally, screw the remaining 2 screws into the stand-offs to secure the board in place.
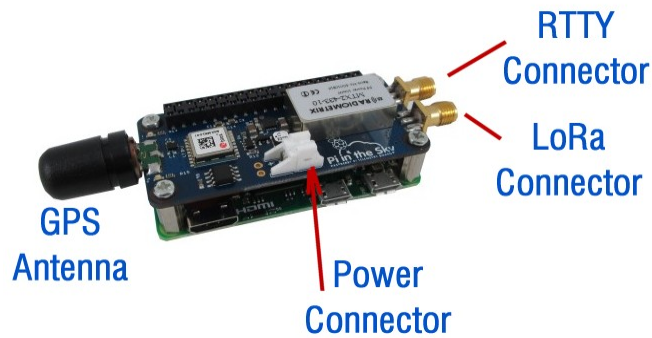
PITS Connections

- UHF Aerial - The kit includes an "SMA Pigtail" from which you can make a simple but very effective aerial (see section later in this document).
- GPS Aerial - This is included. The GPS aerial needs a clear view of the sky, so place it very close to an external window.
- Battery Power - A cable is supplied. For testing you can use standard alkaline cells, but for the flight you must use Energizer Lithium cells. You can also test using a normal 5V PSU connected to the Pi power socket (recommended).
- I2C Port - This allows the connection of external I2C devices. Currently the PITS software supports the BMP085 and BMP180 temperature/pressure sensors, and the BME280 temperature/pressure/humidity sensor.
- External Temperature Sensor - Available to connect a extra DS18B20 temperature sensor to the board, usually to measure the external air temperature.
- OK LED - flashes when GPS is getting a position lock; solid when it has obtained one.
- Warn LED - flashes when GPS has no lock and no time. 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 - plugs into the Pi. Take care to make sure it is aligned correctly!

So, for testing:

1. Connect the pigtail to the radio socket (centre-left)
2. Connect the GPS antenna to the GPS socket (top-left)
3. Reconnect the LAN cable to the Pi
4. Reconnect the USB keyboard to the Pi
5. Reconnect the HSMI monitor cable to the Pi

PITS Zero Connections

- RTTY Connector - The kit includes two "SMA Pigtails" from which you can make simple but very effective aerials (see section later in this document). One connects to the RTTY socket.
- LoRa Connector – as above, this connects to an aerial made from an "SMA Pigtail". The 2 aerials should be kept apart from each other – 170mm is sufficient – in order to prevent them from affecting each other (reducing the range).
- GPS Aerial - This is included. The GPS aerial needs a clear view of the sky, so place it very close to an external window when testing.
- Power Connector- A cable is supplied. For testing you can use standard alkaline cells, but for the flight you must use 3 fresh Energizer Lithium cells. You can also test using a normal 5V PSU connected to the Pi power socket (recommended).
- OK LED - flashes when GPS is getting a position lock; solid when it has obtained one.
- Warn LED - flashes when GPS has no lock and no time. 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.

As the 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

Reconnect power to the Pi micro USB power socket. The Pi will boot <u>and will automatically start the tracker software</u>.

You should see the red WARN light come on once the Pi has booted. This will stay on until the GPS receives lock, which first time could take 2 minutes or so. If after 5 minutes there is no lock, move the GPS antenna to a window, or outside. The receiver is very sensitive and you should get a lock anyway so long as it has sight of a window.

Once GPS lock has been established, the WARN LED will go out and the green OK light will flash.

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 pits/tracker
sudo killall startup
sudo ./tracker
```

The screen should look something like this:

```
RASPBERRY PI-IN-THE-SKY FLIGHT COMPUTER
=======================================

RPi Hardware     : BCM2708
RPi Revision     : 0010
RPi Model A+ or B+
PITS+ Board

RTTY Payload ID = 'PISKY'
Radio baud rate = 300
GPS Logging enabled
Telemetry Logging enabled
Camera Enabled
Adding custom camera parameters '' to raspistill calls
Image size changes at 2000m
RTTY Low image size 240 x 320 pixels
RTTY High image size 640 x 480 pixels
RTTY: 1 Telemetry packet every 4 image packets
RTTY: 60 seconds between photographs
Full Low image size 640 x 480 pixels
Full High image size 2592 x 1944 pixels
Full size: 60 seconds between photographs
MTX2 Frequency to be set to  434.250MHz
MTX2 command  is @PRG_410E76A0
V2.4 or later board with I2C ADC
RTTY: $$PISKY,1,00:00:00,0.00000,0.00000,00000,0,0,0,0.0,0.0,-0*BD2F
RTTY: $$PISKY,2,11:42:46,51.95030,-2.54441,00153,0,0,12,22.2,0.0,1831*56D3
```

Note that this runs the tracker only, not the camera script and not the APRS script.

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 just uncomment everything! 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.

So, to change the payload name, type this command:

```
sudo nano /boot/pisky.txt
```

Please note the file is /boot/pisky.txt – there is a master copy of this file elsewhere but editing that won't change the configuration! Please use the above path only.

The payload line is initially

```
payload=CHANGEME
```

and to change it, use the cursor keys to move the cursor over "CHANGEME". Type in your new payload name and remove any excess characters with the DEL/BkSp keys.

To finish editing, press CTRL and X. If you have made any changes then you will be asked if you want to save them (Y) or not (N).

Restart the tracker and check that your changes follow through; if not then either you edited the wrong file or you didn't save the changes.

For a full description of all common settings, see the next section.

Configuration Options

Here will will just cover the common options that apply to the PITS board only (there are separate sections in this document to explain the sections on the APRS and LORA add-on boards). These options are at the top of the file:

`payload=PISKY`	This is the payload name, which is what will appear on the map and live image page. Names on the latter will be truncated to 6 characters.
`disable_monitor=N`	Set to "Y" to disable the monitor. This extends battery life but is not a good idea if you want to use the monitor!
`frequency=434.250`	This is the frequency that the transmitter will be set. If you are flying more that one tracker, or other people are flying balloons at the same time, then each needs its own frequency. For the EU the legal range is 434.04 to 434.79MHz.
`baud=300`	Baud rate, which controls how quickly information is sent. 300 baud is a good speed to set if your tracker is going to send images as well as telemetry; if not then set it to 50 baud.
`camera=Y`	Enables use of the Pi camera. Set to N to disable use of the camera, or U for a USB webcam.
`#camera_settings=-rot 90`	This adds parameters to the **raspistill** commands. Normally none are needed, but if for example the camera is mounted upside-down or at right-angles, then you will need to add a parameter to correct that. See the raspistill documentation for details of available commands.
`low_width=240`	This and the following line set the dimensions of live images (those used for radio transmission) taken at low altitudes (i.e. on the ground, and the first minutes of the flight). Please resist the temptation to set these to a large size, as your tracker will still be transmitting a large image from the ground when it's on the way up.
`low_height=320`	
`high=2000`	Sets the altitude, in metres, above which the live images switch to the larger dimensions.
`high_width=640`	This and the following line set the dimensions of live images taken at higher altitudes. Do not set these much larger than the defaults as larger images take a long time to transmit.
`high_height=480`	
`image_packets=4`	The tracker sends both telemetry and image data down to the ground. These items (packets) are interleaved so, by default, it sends 4 image

	packets and then 1 telemetry packet before repeating. You can increase this number in order to speed up the transmission of images a little, at the expense of less frequent position updates. At low altitudes, (below the "high" parameter), this value is ignored and the tracker sends 1 telemetry packet for every image packet. This helps to get the best final position before the payload lands.
`logging=GPS,Telemetry`	Enables logging of GPS and/or radio telemetry, to the files gps.txt and telemetry.txt respectively. Both files are placed in the /home/pi/pits/tracker folder.
`info_messages=2`	Tells the tracker to send 2 information messages over the radio when it starts. These contain the amount of free SD card space and the Pi IP address.
`Power_Saving=N`	Set to **Y** to enable GPS power saving. Since most flights only last about 3 hours, and a set of 4 AA cells last about 20 hours, we suggest that you leave this switched off.
`Flight_Mode_Altitude=1500`	This controls the altitude above which the GPS is placed in "flight mode". Flight mode is necessary for any flight that goes above 12km altitude. Below the specified altitude, the GPS is placed in "pedestrian mode" which provides more accurate and stable readings of the landing position. If it is possible for your flight to land at an altitude of more than 1500m then you should increase the value accordingly.

Deleting Images / Log Files

Eventually, image files can fill up your SD card. This should take a long time to happen, especially if you do a lot of testing (good!) or have a small SD card (bad!) or doa lot of flights (good!). You can delete the files manually by logging in to the Pi, or you can choose the easier option:

Create the file /boot/clear.txt, either from the Pi (sudo vi /boot/clear.txt) or by inserting the SD card in a PC and creating the file from there in the small partition that will appear as a drive letter. The contents of the file do not matter.

Next time the tracker program is started, the image files and log files will be deleted, as will the clear.txt file itself. i.e. it's a one-shot use. This prevents the images from being deleted again if the Pi is accidentally rebooted.

The deleted files are:

- `Camera images (files under /home/pi/pits/tracker/images/)`
- `GPS log file (home/pi/pits/tracker/gps.txt)`
- `Telemetry log file (home/pi/pits/tracker/telemetry.txt)`

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 is picked up by an aerial which feeds it into a radio receiver. The receiver converts the radio signals to audio signals which are then fed to a program on PC. That program listens to the audio tones and converts them to binary 1's and 0's to form telemetry or image packets. These packets are then 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.

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.

There are different types of radio that can be used, and prices vary greatly from about £20 for a USB TV receiver up to £500 or more for a radio ham receiver. These options are discussed below.

Radios and Scanners

The PITS radio signal is at about 434MHz which is classed as UHF (Ultra High Frequency) and is part of what radio amateurs call the "70cm band". This band was chosen because it can be used at low power without a licence, and because there are many amateur radio receivers that can be tuned to those frequencies. It is worthwhile contacting your local amateur radio club to see if anyone there is willing to help with tracking, and they may even loan some equipment.

So the first requirement is that the radio can be tuned to the 434MHz band. The other main requirement is that it can be set to "SSB" which is an option like "AM" and "FM" on regular radios. There are 2 SSB settings called "LSB" and "USB" (this USB is nothing to do with USB ports on a computer) and suitable radios will offer both.

There are 2 classes of radio that can receive 434MHz SSB transmissions:

- Radio transceivers – these can be used to receive or transmit. Transmit mode can only be legally used with an amateur radio licence, but you do not need a licence to own a transceiver or to use it for receiving. Suitable models include the Yaesu 817 and Yaesu 790; the latter is an old model only available used.
- Radio scanners – these are receive-only and typically cover a very wide frequency range and not just the 434MHz band. Most scanners especially those sold for receiving aircraft are AM/FM only and not SSB, so cannot be used. Suitable models include the Yupiteru MVT-7100 and ICOM IC-R10 (see picture below).

Most of the above radios can only be purchased used, for example on ebay or from an amateur radio shop. They tend to hold their value well, so if you buy one for a launch then you can sell it afterwards for pretty much what you paid for it.

To use the radio, attach an antenna (for testing any antenna will work – even a small piece of wire pushed into the aerial socket) and switch it on. Set the radio to "USB" mode, using the Mode button (on scanners this may need you to press a SHIFT or FUNC key first – see the manual).

Now tune in to the frequency of your PITS tracker which should also be switched on (the frequency is in the /boot/pisky.txt file that we discussed earlier). On models such as the Yaesu 817 you will need to press the "Band" button first, and then use the tuning dial to set the frequency. Often there are 2 dials – coarse and fine. On scanners the procedure is a bit different – you can type in the frequency on a keypad, and then fine-tune with the tuning dial. As you tune you should hear the PITS signal loud and clear, and the signal level (a bar like on a mobile phone) should be have all bars showing. If you hear nothing then turn the Squelch dial until you can.

When the transmitter and receiver are close to each other, it's possible for the receiver to be overloaded which results in spurious signals. To check that you are not tuned to one of those, once you can hear the signal turn the tuning dial anti-clockwise slightly. If the signal goes up in frequency, you are tuned correctly. If instead it goes down in frequency, keep

turning the dial anti-clockwise until the sound disappears and then comes back again. Now, as you turn the dial anti-clockwise, the frequency will increase; you are now correctly tuned.

Connect the audio jack on the radio to the "line in" or "Mic" socket on your PC using a standard 2-pin or 3-pin 3.5mm audio cable. If your PC has a combo "headset" socket then you will also need a 4-pin to 3-pin adapter (Amazon or ebay), and the same applies if you intend to use a tablet or phone to decode. Proceed to the "Decoding On A PC" section.

SDRs

These are "Software Defined Radios", which rely on a separate computer to process the radio signal and convert it to audio. SDRs connect to the PC via a USB port, and it's up to software on the PC to decide what frequency to tune to, what mode (AM/FM/SSB) to use, etc.

SDRs can be very low cost. For a small budget, get a V3 RTL-SDR – see https://www.rtl-sdr.com/buy-rtl-sdr-dvb-t-dongles/. Like all budget SDRs these are a little bit deaf (not sensitive to weak signals) and are sensitive to interference from other signals, so are best combined with a separate "HAB Amp", that filters and amplifies the signal from the aerial, and is available from where you bought the PITS board.

For a bit more money, get an SDR-Play or AirSpy. Neither need the filtered pre-amp and less you're close to an interfering transmitter.

Any SDR is useless without software, and we recommend SDR Sharp. Other programs include HDSDR and SDR Radio, and there's no reason why you shouldn't install and try all of them and see which you get on best with.

For more information on SDRs and SDR Sharp, see:

https://ukhas.org.uk/guides:sdr_tracker

A nice thing about SDRs is that you get to see the radio spectrum including your tracker's transmissions, and those have a characteristic 2-peak shape:
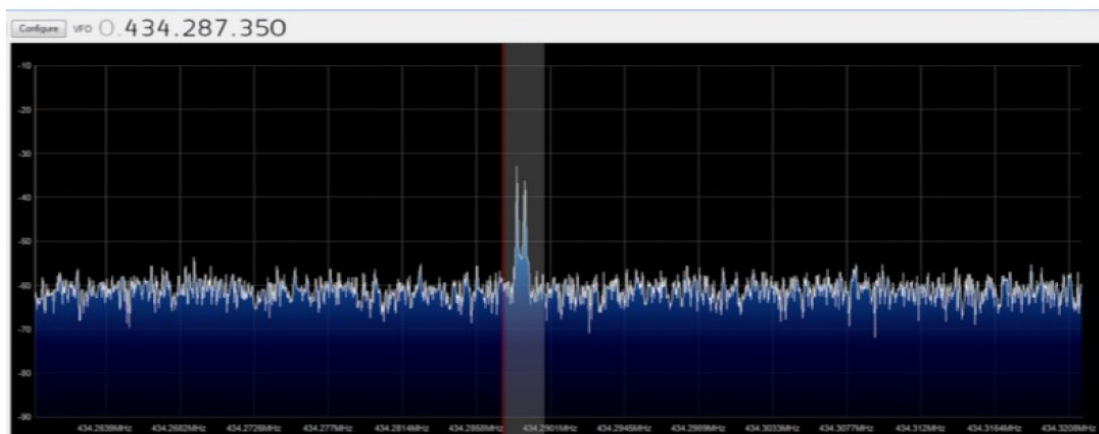
Whichever device you choose, and whichever radio program you use, the result will be an audio signal that needs to be fed into the decoding software (described in the next section). Doing so requires software called VAC (Virtual Audio Cable) which pipes the audio from the SDR software into the decoding software. Again, this is described in the above link.
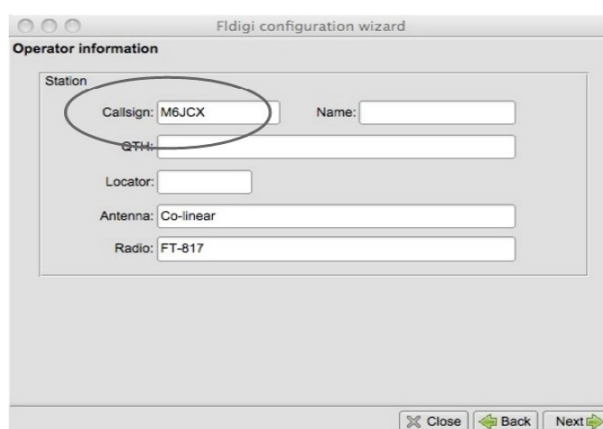
Decoding On A PC

Whether we used a "real" radio to convert the radio signal to an audio signal carried in an audio cable to the line-in socket, or an SDR dongle plugged in to a USB socket with SDR software converting the radio signal to an audio signal on a "virtual" audio cable, we still need to convert the resulting bleeps and whistles to actual data (telemetry and/or images).

On a PC this job is done by dl-fldigi, which is a "software modem" that also uploads the results to some internet servers, from where the live map and live image pages are derived. There's a guide on dl-fldigi here:

https://ukhas.org.uk/guides:tracking_guide

When you start dl-fldigi for the first time, you will be asked to enter a callsign, which could be your school name for example; radio amateurs tend to put in their radio callsign:

The setup will also ask what audio device to listen to. If you are using a radio or scanner, connected to the PC via an audio cable, then you should choose "Line In" or "Microphone"; these options vary from PC to PC and if you aren't sure then just choose the most likely – you can change it later if it's not working. For SDRs, you should select "Virtual Audio Device". If you have selected the correct audio device, and the radio is connected and tuned in (or SDR software running and tuned in), then you should see the PITS signal in the waterfall:
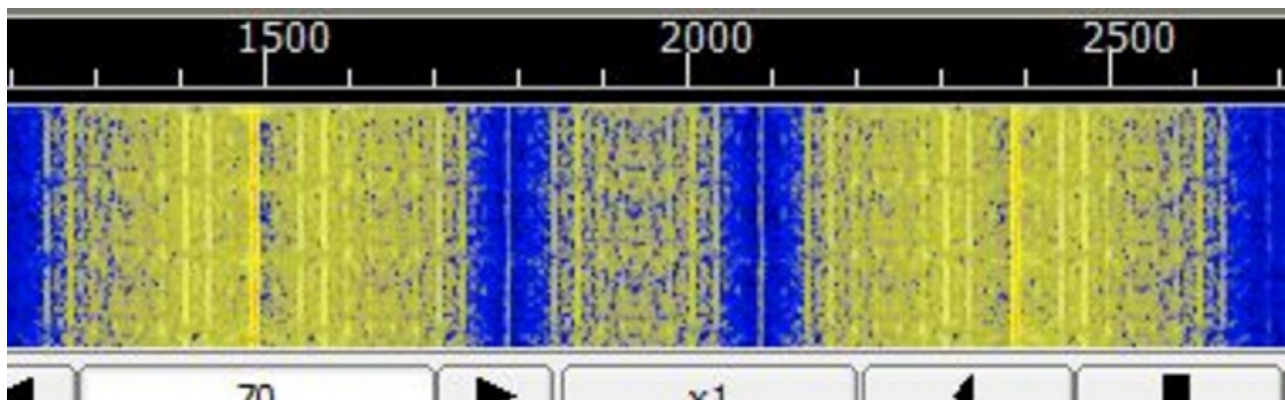
If the signal is off to one side, retune your radio or SDR software so that the 2 bands are roughly in the centre of the waterfall.
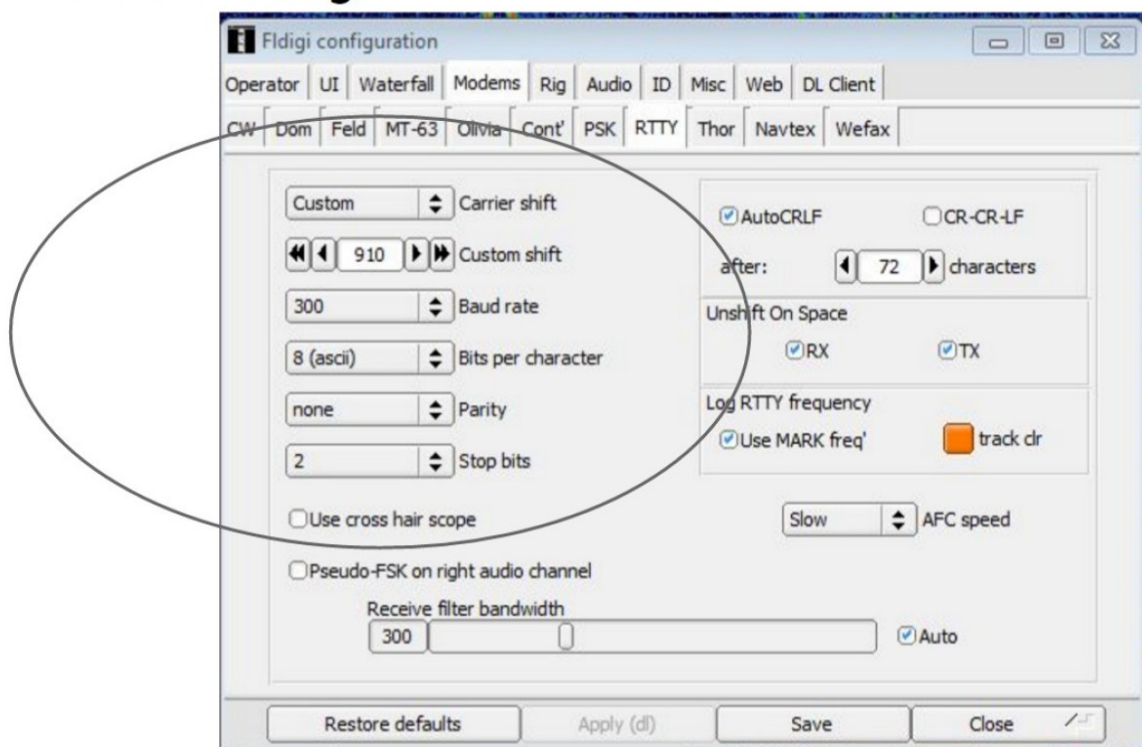
If the signal doesn't look like the above, check that the radio or SDR software are set to USB mode. Also check that you set the sound source correctly.

Next, dl-fldigi needs to be told what sort of signal to expect, and this is "RTTY" as we discussed earlier:
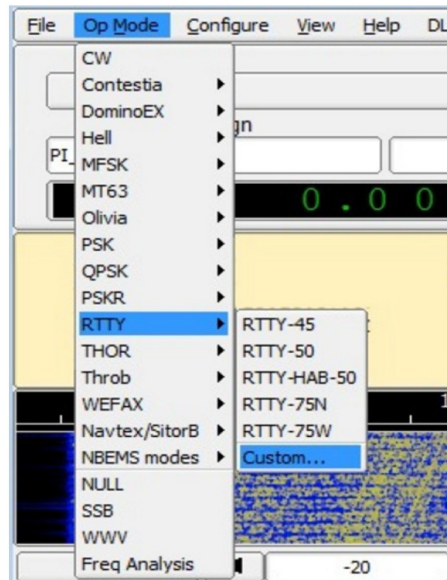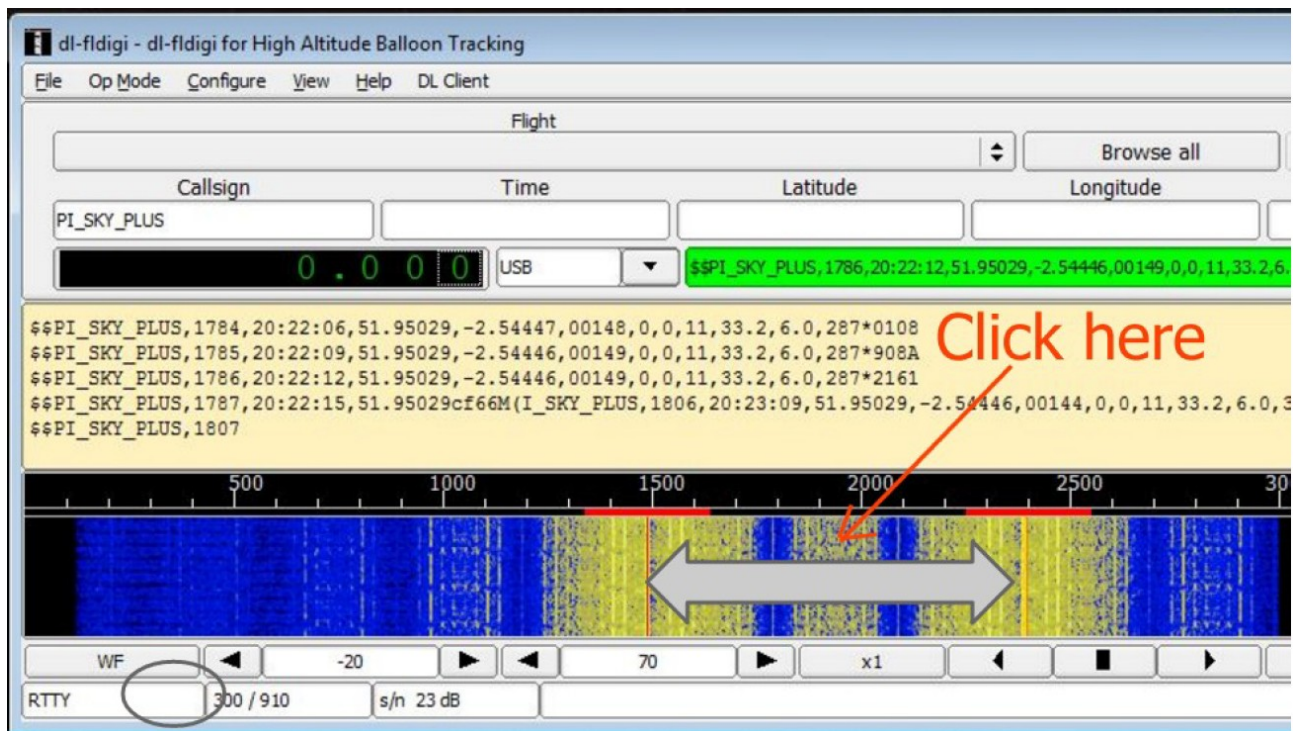

The "Custom..." option above then asks you for the custom settings, which are:

Finally, click in the waterfall area, midway between the 2 yellow signal areas – see below.

Clicking aligns the 2 red cursor lines with the incoming signal; if you don't do that then the data won't be decoded.

If all is well, the centre window will show the data coming from the payload. This will alternately be text (telemetry packets) and rubbish (image packets). In either case, successful packets are shown above and to the right, in the green box (Green means success; red means failure). With the tracker and radio in the same room, pretty much all packets should be green.

If you don't see any readable text for a while, right-click in the text area and choose "Clear" from the menu.

If you still don't get good decodes, check the following:

- For a real radio, that you selected USB mode
- For an SDR, that you selected USB mode
- For an SDR, it's possible that "I and Q" are swapped. SDR software will have an option to swap these 2 over – try changing that setting.
- For a real radio, check that the audio input on the PC does not have any "enhancements" enabled. This is a common default setting on modern PCs.

Decoding On A Phone or Tablet

If you have an Android phone or tablet then you can use that with a radio/scanner, instead of a PC. You cannot currently use an Android phone/tablet with an SDR dongle.

The software you need is available in the Google Play store as "HAB Modem and Tracker", is available via this link:

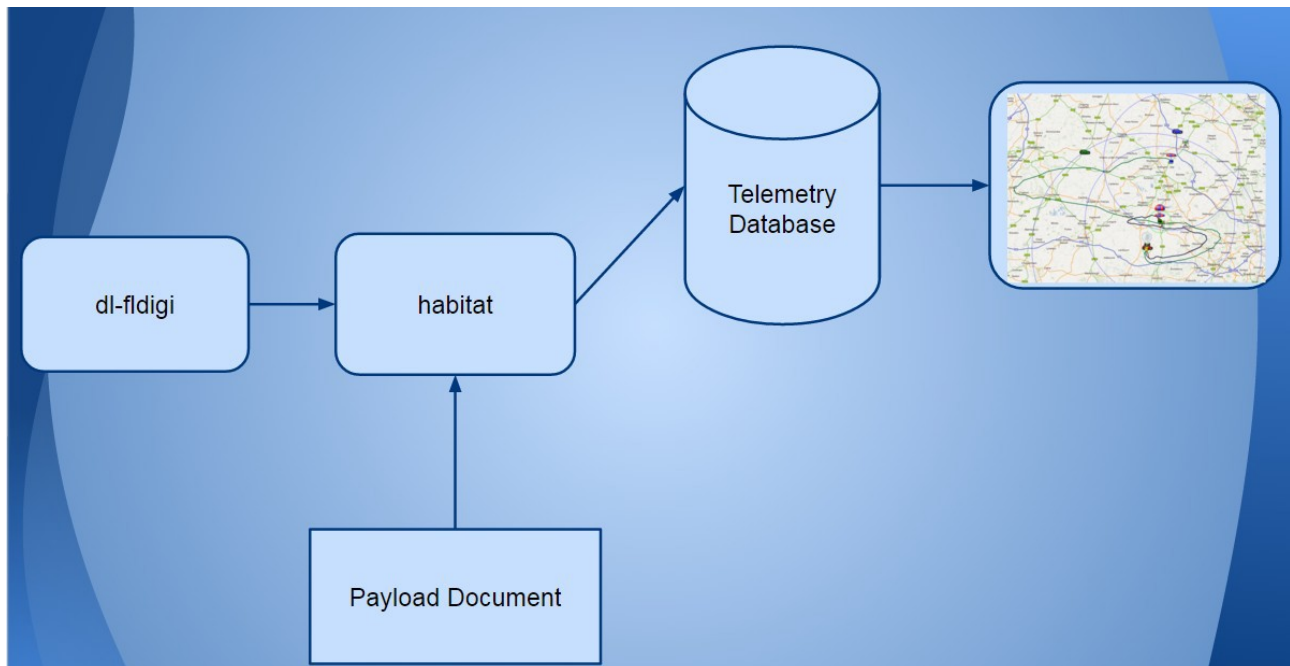https://play.google.com/store/apps/details?id=com.brejza.matt.habmodem&hl=en_GB

and is described here:

https://ukhas.org.uk/projects:hab_modem
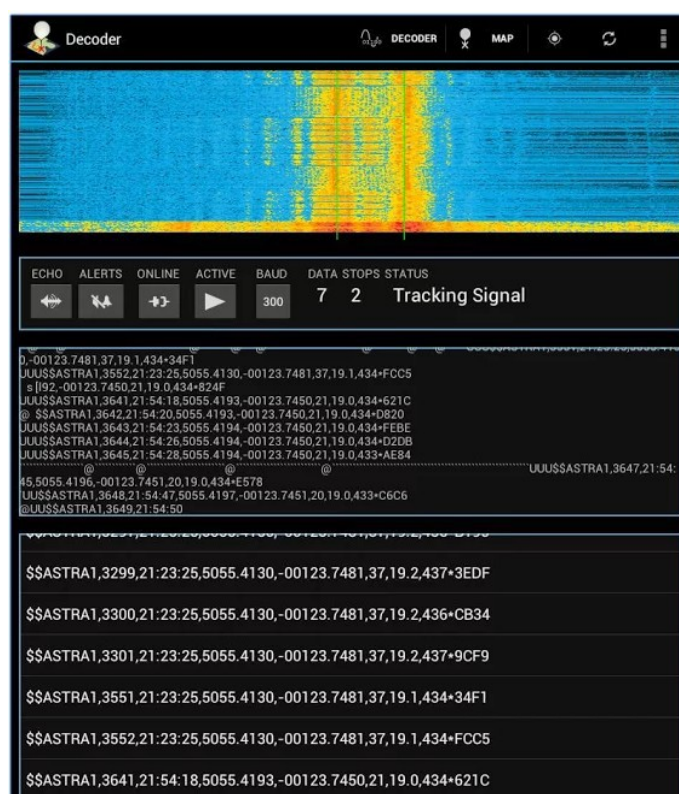
It doesn't decode or upload image packets, but that's the only downside.

Getting On The Map

The above diagram shows how data from dl-fldigi (or the Android HAB Modem) gets onto the map. First, the decoding 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 PITS this defaults to CHANGEME, 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).
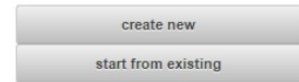
A flight document contains a list of payload configuration documents - i.e., the payloads that are going to be launched. You can add more than one, if there are several payloads on one balloon/flight train - but for multiple launches, create a flight document per balloon.

create new

start from existing

click on "start from existing", as we are going to use an existing document as a starting point, to save time. Type in "PISKY" in the box:

| Search prefix: | PISKY | | Search | Prev Page | Next Page | Cancel |

PISKY	PISKY "Pi In The Sky Board"	8c06fe95e4738fb38dcc239d7b300899 2014-06-18T17:10:57+01:00
PISKY	PISKY "Pi In The Sky Board"	dd29e77c38b5ff889322a119dd08eaed 2014-07-02T14:31:26+01:00
PISKY	PISKY "Pi In The Sky Board"	1f779ef303bd8b7afd691c534d64a29a 2014-07-07T10:59:21+01:00
PISKY	PISKY "Pi In The Sky Board"	1f779ef303bd8b7afd691c534d9debcd 2014-07-07T11:24:25+01:00

then click on the bottom entry in the resulting list.

It is **vital** that you now change some settings to match your payload. First, the "payload name"; I suggest you enter your payload ID here, but it could be something else:

Payload configuration

| Payload name | PYPITS | ✔ | Need not be its callsign. |

| Description | PITS | | Free form, optional. Used only to help you find it later. |

Next, click the Edit button to the right of "Radio and telemetry configuration, to show this screen:

Description	PITS Test		Optional; e.g., Primary, alternative, nighttime mode.
Frequency	434.25	✔	MHz
Mode	USB ▾		
Modulation	RTTY ▾		
Shift (Hz)	910	✔	Shifts supported by legacy fldigi are: 23, 85, 160, 170, 182, 200, 240, 350, 425, 600 and 850. Modern versions support any custom shift.
Encoding	ASCII-8 ▾		
Baud	300	✔	fldigi supports 45, 45.45, 50, 56, 75, 100, 110, 150, 200 and 300 baud
Parity	none ▾		
Stop bits	2 ▾		

Confirm   Cancel

Set the frequency to match what is in your pisky.txt. This doesn't stop your payload from appearing on the map, but it does help others know what frequency to tune to.

Click the Confirm button.

Now, click on the Edit button next to "Parser Configuration":

You <u>must</u> fill in "Callsign" with your callsign from /boot/pisky.txt. If you don't, then your 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
- You are receiving and decoding in dl-fldigi
- You checked the "Online" button in dl-fldigi
- 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.

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. The usual reason for failure is that the number of type fields sent by the payload do not match those in the payload document. By default, the PITS software sends the following:

- Sentence Counter - Integer - Auto-incrementing number that starts at 1 and increments by 1 for each new sentence.
- Time - UTC time in the format hh:mm:ss. This is the same as GMT and is 1 hour behind BST. UTC is used regardless of what time zone your balloon flies in.
- Latitude (decimal)
- Longitude (decimal)
- Altitude in metres
- Horizontal Speed in m/s
- Heading in degrees
- Satellites - Number of satellites used in producing position fix
- Internal Temperature in degrees C (as measured on the PITS board)
- Battery voltage (V)
- Battery current (mA)

Caveats:

- If you are using the PITS Zero board, then the battery voltage and current are not sent.
- If you are using the original PITS board on the model A or B, then the battery current field is not sent.
- If you have added a BMP085/180 PT sensor, then the temperature and pressure will be appended to the above fields.
- If you have added an external DS18B20 temperature sensor, then the temperature will be appended to the above fields.

Finally, if you are also using LoRa, then you need to create a separate payload document for that. We suggest that once you have a working document for RTTY, then you create a new one for LoRa based on the RTTY, and just change the callsign to match the "Payload ID" you set for the LoRa transmissions. You will then have 2 balloons on the map, one on top of the other; during flight they will leapfrog each other as transmissions are uploaded.

Flying

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 and tested 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.

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.) 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 do not rely on them.

For more information and advice about the practicalities of launching and chasing a balloon, see this document:

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

So, back to 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" which you should keep a copy of.

Flight documents need to be approved. To do this, go to this URL:

https://web.libera.chat/

Type in your name in "Nick", **habhub** in "Channel", and then click "Start". After it connects, type in a message:

>   Please approve my flight document XXXXXXXXXXXXX

replacing XXX… with your document ID (you got that earlier) and press ENTER. A bot will then display details about your flight, including when you last tested your tracker. If that all looks OK, and an admin is online and watching then they will approve it. If nobody is around then stay connected and check later.

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

I also recommend that you join the **highaltitude** channel to introduce yourself to the ballooning community. 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.

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:
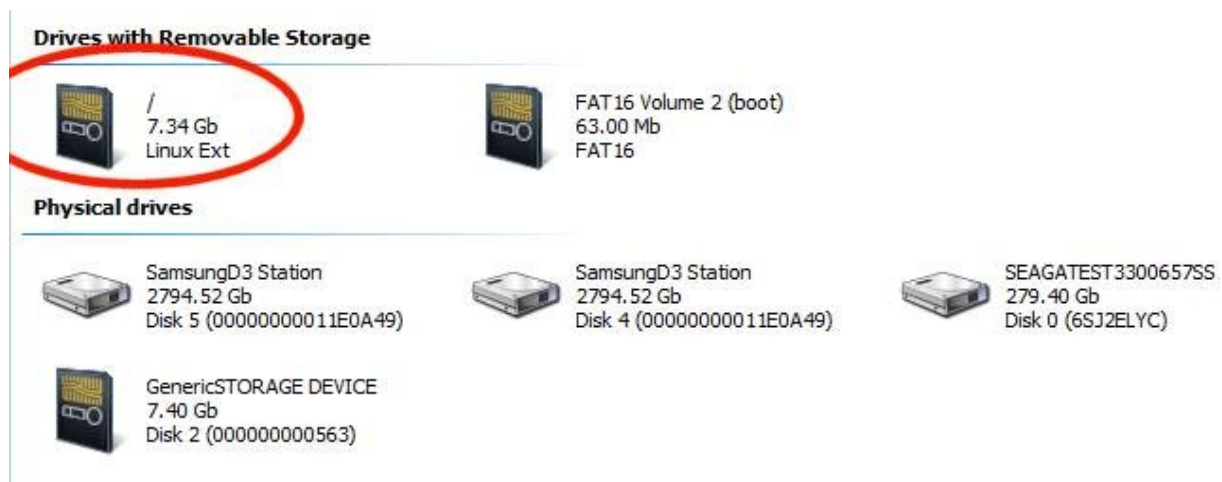

Double-click on the "Linux Ext" partition under "Drives with Removable Storage", and not the entry in "Physical Drives". You will then see the contents of the partition:
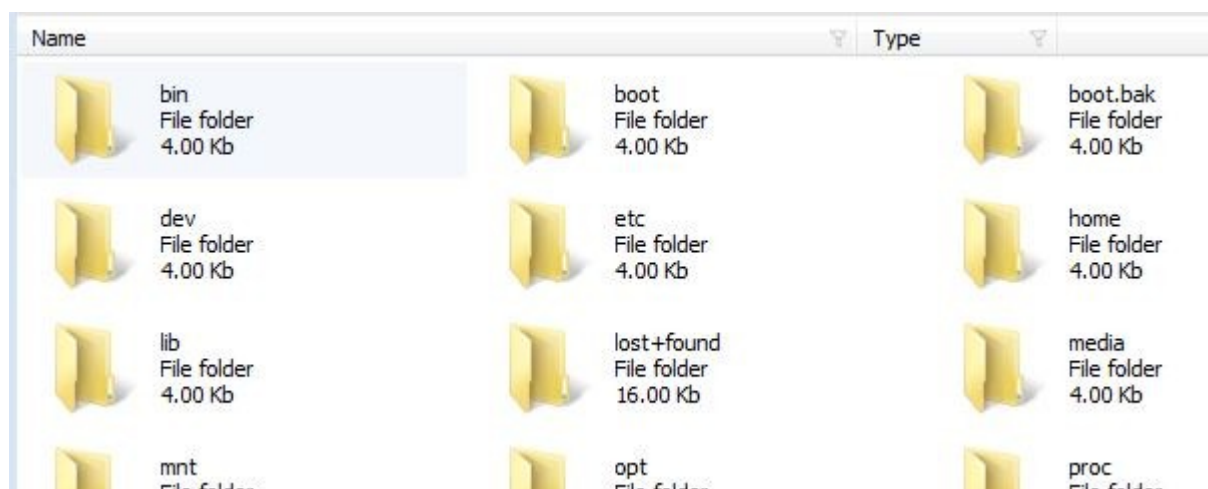
Open (by double-clicking again) the "home" folder, then the "pi" folder, then the "pits" folder, then the "tracker" folder, and finally the "images" folder. You will then see the following folders:

- FULL (for full-sized images)
- LORA0 (for images taken for the LoRa 0 (CE0) channel
- LORA1 (for images taken for the LoRa 1 (CE1) channel
- RTTY (for images taken for the RTTY channel

This is the folder structure shown graphically:
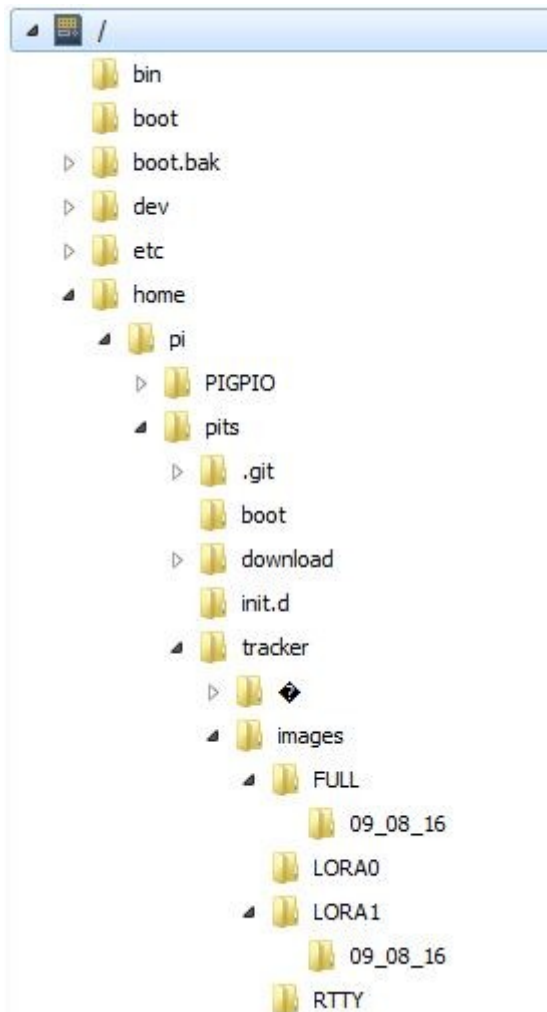
Within each of the 4 folders mentioned, you may see a number of dated folders, one per date that the tracker stored images on. Note that the Pi only knows the current date and time if it is connected to the internet or if it has a GPS lock, so any photographs taken otherwise (i.e. disconnected from the 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:
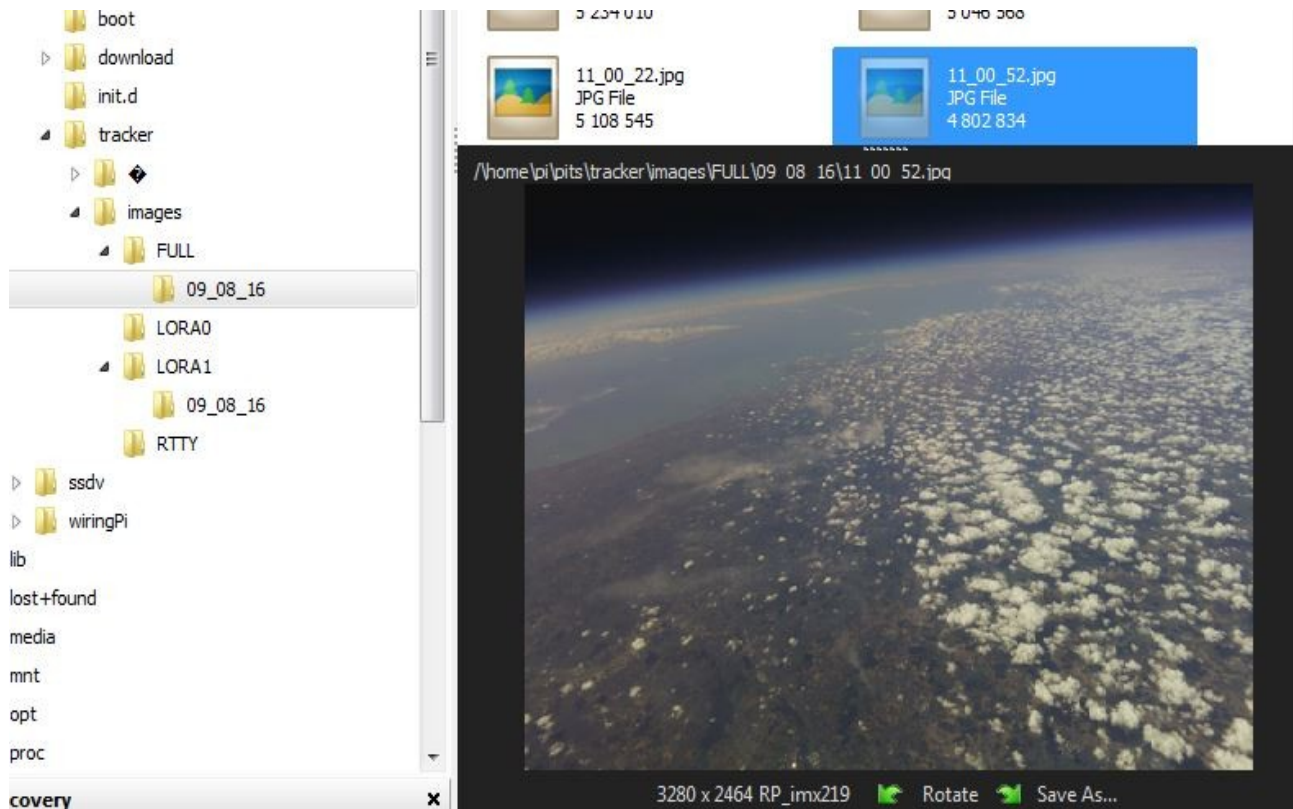
To copy images to a PC drive, first select them (you can select all with CRTL-A or right-click on an image then choose Select All from the menu), and then clock the Save button on the toolbar. The program will then ask where to save the files.

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 cannot 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

We have developed a board that stacks over the PITS board:
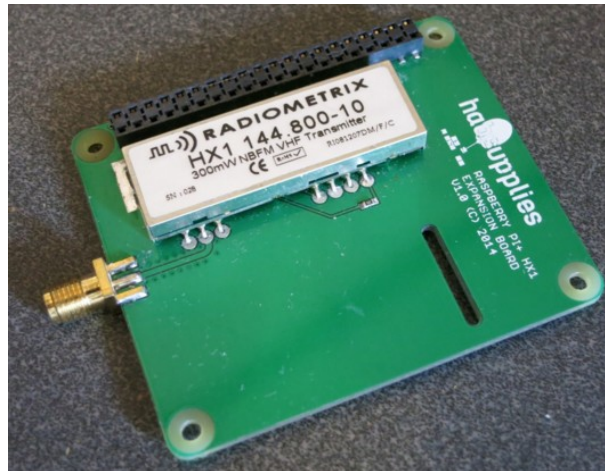
Installation

To install, the supplied stacking header should be placed on the Pi, then the PITS board, then the standard header (that came with PITS), and finally the APRS board on top. Both boards have a slot for the camera cable. *Camera images are sent via the PITS board only, and not over APRS.*

Configuration

As before, configuration is via /boot/pisky.txt. You will find an 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.

- **APRS_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.
- **APRS_ID=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.
- **APRS_Period=1.** This is the period in minutes between transmissions. We chose minutes not seconds so it's not possible to use very short gaps and thus flood the network.
- **APRS_Offset=10.** Sets the time in seconds to delay before sending a packet (see explanation below).

- `APRS_Random=5.` Sets a random time (0 - 4 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.

- `APRS_Altitude=1500.` At altitudes lower than this, APRS packets are set to "WIDE1-1, WIDE2-1". For altitudes above this, see below.

- `APRS_HighPath=Y.` At altitudes higher than APRS_Altitude, APRS packets are set to "WIDE2-1" if this parameter is Y, or to no path at all if set to N.

- `APRS_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.

- `APRS_Telemetry=Y.` Adds the following values to the packet:
  - Sequence Number
  - Satellites used
  - Internal temperature
  - Battery voltage

The timing works like this: if Offset and Random are set to zero, and Period to 1, then packets are sent at midnight, 1 minute past, 2 minutes past, etc. i.e. 00:00:00, 00:01:00, 00:02:00. If the Period is 2, then the timings are 00:00:00, 00:02:00, 00:04:00 etc. Now, if the offset is 10, they will be 00:00:10, 00:02:10, 00:04:10 etc. So it is possible to set how often the transmissions happen and when they happen.

If you are part of a group launch, or you know of other APRS flights in the area, then you can use these settings to avoid the APRS packets conflicting (i.e. transmitting at the same time).

If instead you are flying alone, there may still be conflicts with other APRS transmissions including ground-based ones. It is possible that your transmissions permanently coincide with those from another transmitter that uses the same time settings. To avoid this, use the Random parameter to add a random delay before each packet.

Important

If you have a monitor connected to the Pi via HDMI, this will prevent APRS from working (it redirects the sound output to HDMI instead of PWM). So, if the APRS signal is just a carrier with no modulation, remove the HDMI cable and reboot. Or (we have yet to test this) you can achieve the same by using raspi-config to direct audio to the audio jack instead of HDMI.

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. <u>If this is on the wrong device then close the program, reconfigure, and try again</u>. 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:
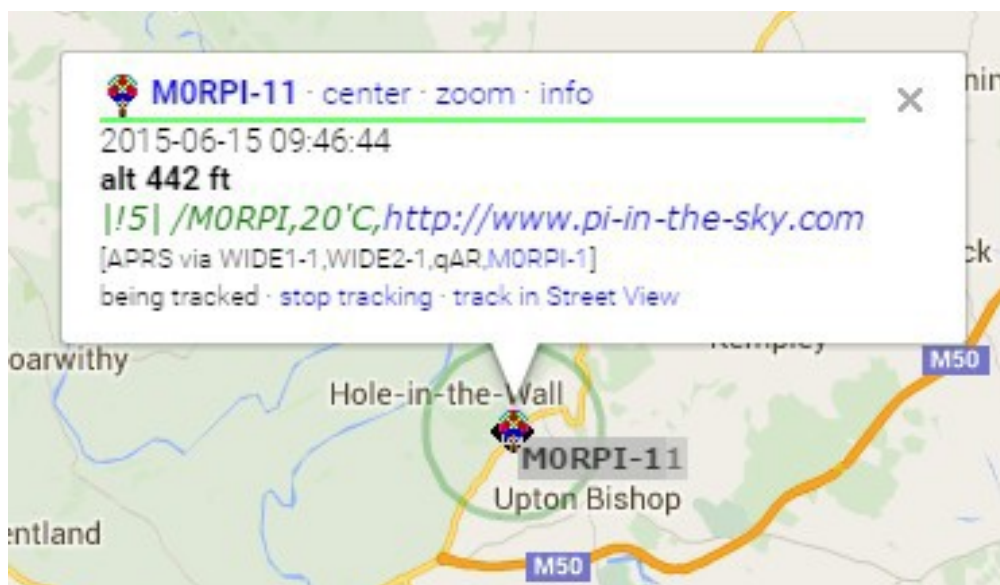
LoRa

LoRa – Long Range radio – is a radio modulation scheme owned by Semtech. The PITS software includes support for Semtech LoRa chips, and HopeRF LoRa modules (which use the same silicon). The PITS LoRa board is supplied with a single HopeRF module but can accept two.

LoRa has several advantages over RTTY:

- Higher bandwidths available

- Longer range for the same bandwidth

- Can be used as an uplink (i.e. sending data to the balloon)

- Less expensive receiving equipment

- Dedicated receivers can be built using a Pi (no PC needed)

To track a balloon with LoRa and PITS, you need a LoRa board to stack onto your PITS board and Pi; for PITS Zero the LoRa module is included. **You also need a separate LoRa board on another Pi to act as a receiver on the ground.** This is our LoRa board (this one populated with 2 modules – normally it would be just 1):

The second LoRa position (CE1) is not compatible with earlier Pi In The Sky tracker boards (because those boards used an SPI A/D converter. To use this second position you need V2.5 or later of the Pi In The Sky tracker board (which has an I2C A/D converter instead).

We recommend using a 434MHz module; 868MHz modules are available but there are more people with 434 LoRa kit, and aerials are easier to come by.

Installation

To install, the supplied stacking header should be placed on the Pi, then the PITS board, then the standard header (that came with PITS), and finally the LoRa board on top. Both boards have a slot for the camera cable.

You will need to make an aerial for the LoRa transmitter; one is not supplied with the board.

Configuration

There are some additional settings which relate to LoRa, all in the /boot/pisky.txt file as usual. The only ones that you must set (the rest are optional) are:

    LORA_Frequency_0=434.400
    LORA_Payload_0=PI_SKY_TEST
    LORA_Mode_0=1

"_0" refers to the module in the CE0 position; "_1" would refer to the module in CE1 position. *For PITS Zero, use _1.*

LORA_Frequency_0 sets the frequency in MHz. Take care to use a frequency that is legal to transmit on (e.g. in the UK make sure you comply with IR2030. Also make sure that the frequency is at least 20kHz away from that used by the PITS RTTY transmitter.

LORA_Payload_0 sets the payload ID, which should be unique (to avoid conflicts with other balloons), and that means **different to the RTTY payload ID that you also set.** Also remember that if you are sending images (SSDV) then those only allow for 6-character payload IDs. So if you do something like using "THINGY-RTTY" and "THINGY-LORA", **both** will end up as "THINGY" on the SSDV page, which will cause confusion and possibly corrupted images. So, always use IDs that are unique within the first 6 characters.

LORA_Mode_0 sets the "mode" of transmissions. We recommend mode 0 for just telemetry, or mode 1(which transmits faster albeit with less arange) if you also wish to transmit images. Both use 20.8kHz bandwidth and thus are compliant with IR2030 for continuous transmission.

LoRa Gateway

Making A LoRa Gateway

A nice thing about LoRa is that the receiver needs very very little processor power - even a lowly PIC or AVR has plenty enough power to do the job - because the hard work is done inside the LoRa chip. However, having received the data it would be good to upload it to the internet so that others can follow the flight on a map, and/or view the pretty pictures being received, and for this a Raspberry Pi is ideal.

So to build a LoRa gateway you need:

1. A second Pi In The Sky LoRa board
2. Raspberry Pi (A+ or B+ or B V2 or B V3)
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 (supplied).

Software Installation

Next, burn an operating system onto a suitable SD card. For this purpose anything from 4GB should be fine. The following instructions are for Raspbian Jessie Lite, and no other operating systems have been tested.

First, run raspi-config:

```
sudo raspi-config
```
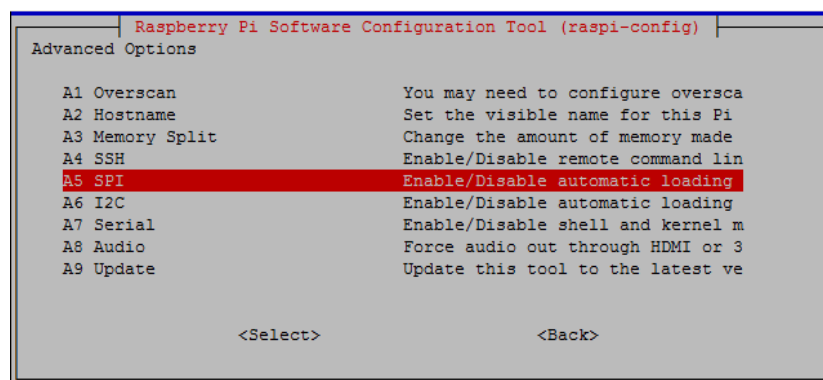

Then choose Advanced Options –> SPI and enable SPI.

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_0=434.450
mode_0=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 first LoRa device (the one in position "CE0"). Frequency is in MHz and should match the frequency of the tracker that you intend to receive.
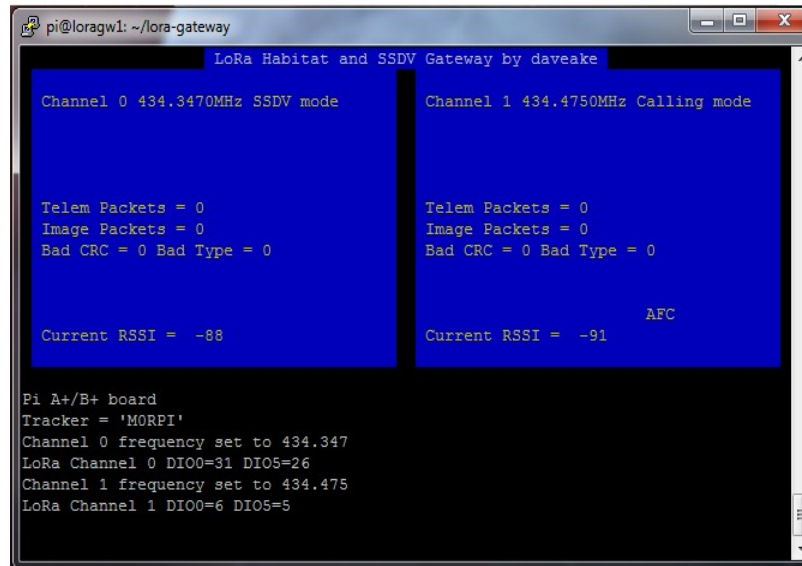
Finally set the mode to match your transmitter.

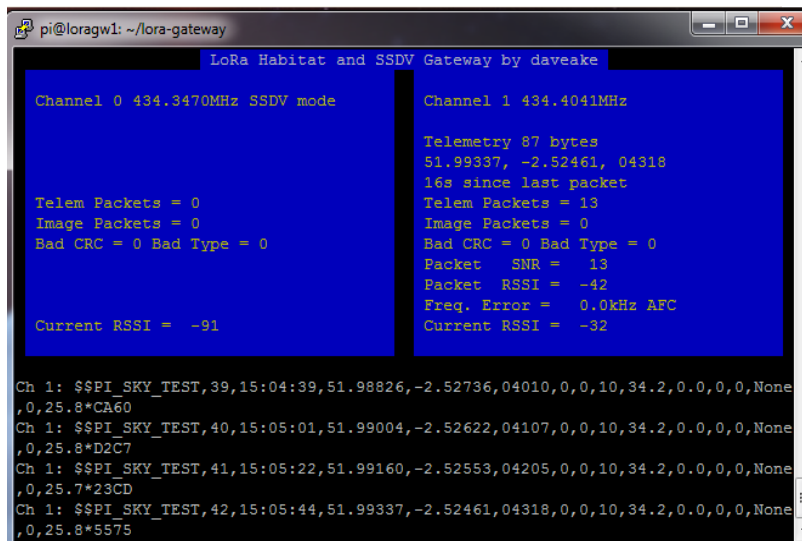For full details on all settings see the README.md file.

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:

FAQ

Why doesn't my Pi boot?

Make sure you followed the instructions correctly on the Raspberry Pi site. Then try another card. 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. Other makes of primary (not rechargeable) Lithium cells are available, and these should be fine however few people, if any, have tested them for HAB flights, so their performance may not be as good.

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

What run time should I expect?

About 20 hours for PITS with the above cells, or 15 hours with PITS Zero, which is plenty.

Both the above assume that you are using a model A or A+ or Zero, that you disable the monitor output, and that the batteries are not powering anything else.

Can I use a model B+ or V2 B+ or V3 Raspberry Pi?

Yes, however you will vastly reduce the run time for no benefit whatsoever, especially the V3. We recommend using one of these developing, configuring and testing only; for flights use a model A/A+ or Zero.

It all works except there are no images being downloaded

Check that the camera is connected, that the configuration file (pisky.txt) has a line to enable the camera, and that the camera is enabled in raspi-config. Also check that the SD card isn't full. Finally, check 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 the "Online" box under "DL Client" in dl-fldigi is checked, and that you have created a payload document for your payload ID; see the payload document section above.

How heavy is the PITS board?

The weight breakdown is:

- Pi In The Sky Main Board - 19.2g
- Raspberry Pi A+ - 21.5g
- Header + Standoffs - 9.3g
- GPS antenna -   55.0g
- Battery Holder+Clip - 13.4g
- Energizer L91 Lithium (AA) - 14.5g x 4
- So the total launch weight of the above is about 180g.