or

“How to overload Habitat with >33000 radiosondes”

```
Mark Jessop        |  VK5QI    | vk5qi@rfhead.net
Michael Wheeler    |  VK3FUR   | m@mwheeler.org
```

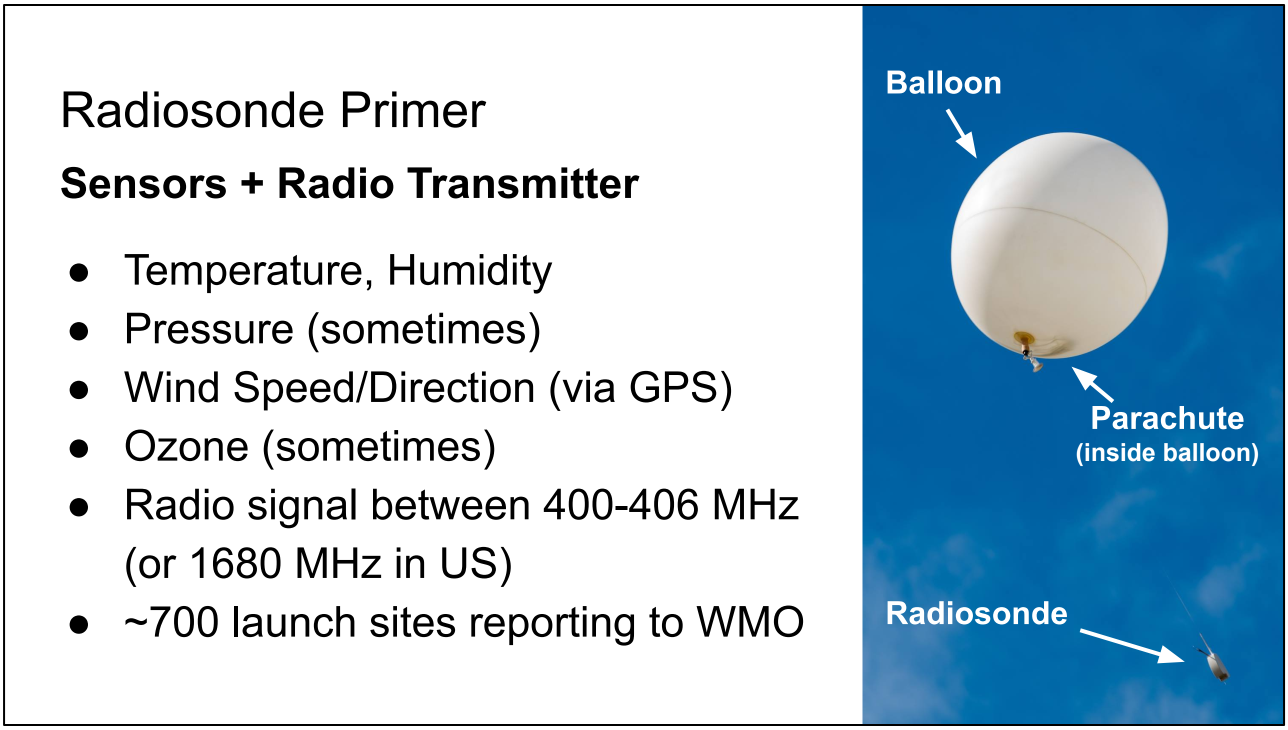
github.com/projecthorus/radiosonde_auto_rx

This presentation was given at UKHAS2019, on the 7th of September 2019.

Talk Overview

- Radiosonde 101
- Why hunt radiosondes?
- Common Radiosonde Types
- Radiosonde-Auto-RX
- SondeHub

Talk overview...



Radiosonde Primer

Sensors + Radio Transmitter

- Temperature, Humidity
- Pressure (sometimes)
- Wind Speed/Direction (via GPS)
- Ozone (sometimes)
- Radio signal between 400-406 MHz (or 1680 MHz in US)
- ~700 launch sites reporting to WMO

This being the UKHAS conference, I’m going to assume you’re somewhat familiar with how a high-altitude balloon flight works and focus on the radiosonde itself.
So a radiosonde is a meteorological instrument which collects data on the earth’s atmosphere and transmits it back to the ground via a radio link.
Essentially all radiosondes nowdays have temperature and humidity sensors, and a GPS receiver to measure wind speed vectors. Sometimes you find add-on payloads like Ozone and radiation sensors too.
The majority of radiosondes transmit somewhere between 400 and 406 MHz, though in the US some transmit around 1.7GHz.
As of last month, there were 700 radiosonde launch sites reporting to the world meteorological organization, most of which launch at least once a day.

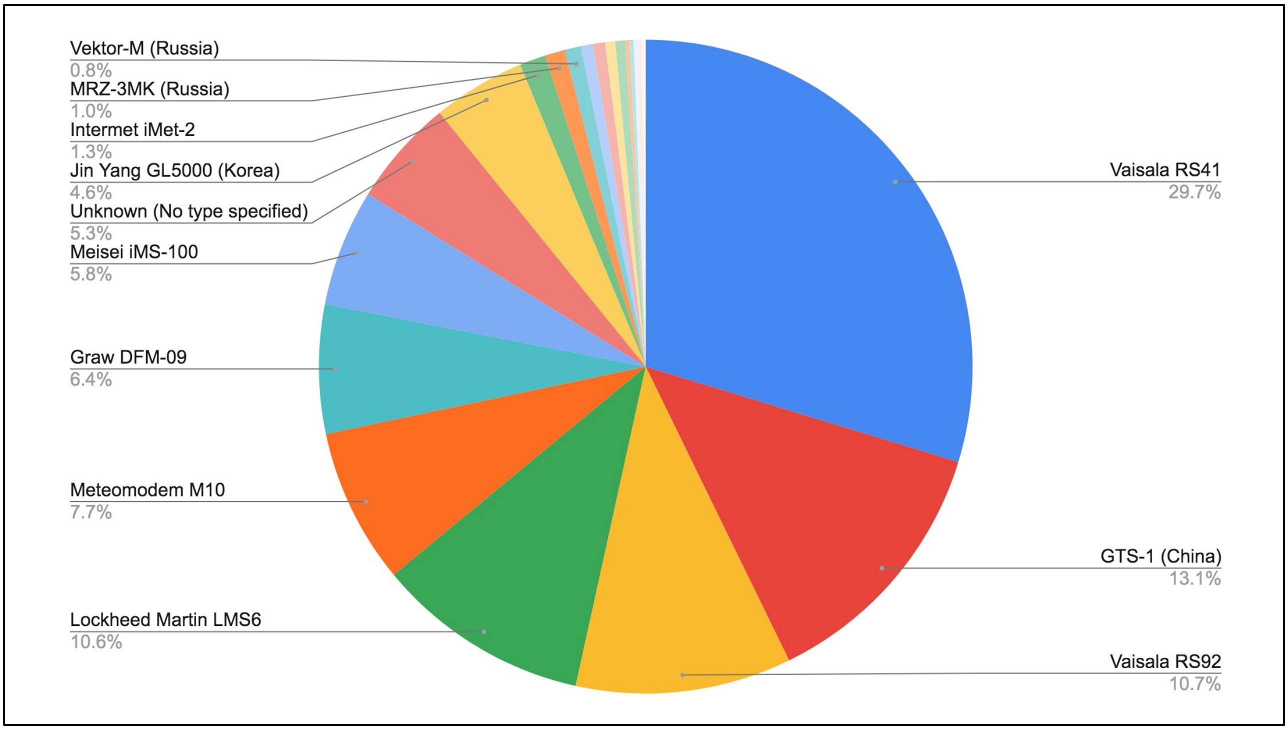
Why bother chasing radiosondes?

- Clean up the Met Bureau’s mess
- Good practice for your own HAB launch
- Test out chase-car systems in a situation identical to a ‘real’ chase
- Free balloon trackers!
- Christmas Decorations?

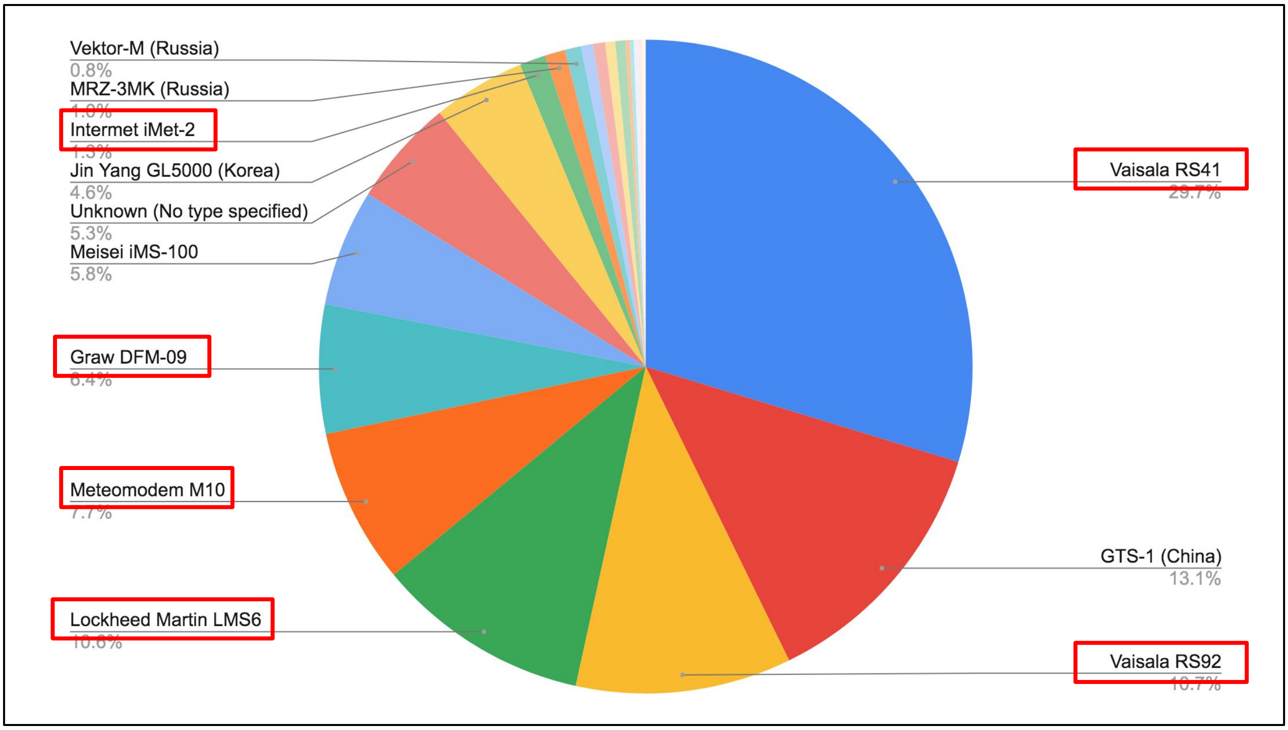
So why do we bother?
Well, of those 700+ radiosondes launches per day, precisely none of them are actively recovered by the launching organisation.
So, we can help clean up the met bureau’s mess, and shame them on Twitter.
It’s great practice for your own HAB launch, and it lets you test out your chase car systems and receiver hardware in a situation identical to a ‘real’ balloon chase.
Depending on the radiosonde you are chasing, you might be able to re-program it for use in one of your own flights.
… and they make great christmas decorations...

Photos courtesy Artur Greficz

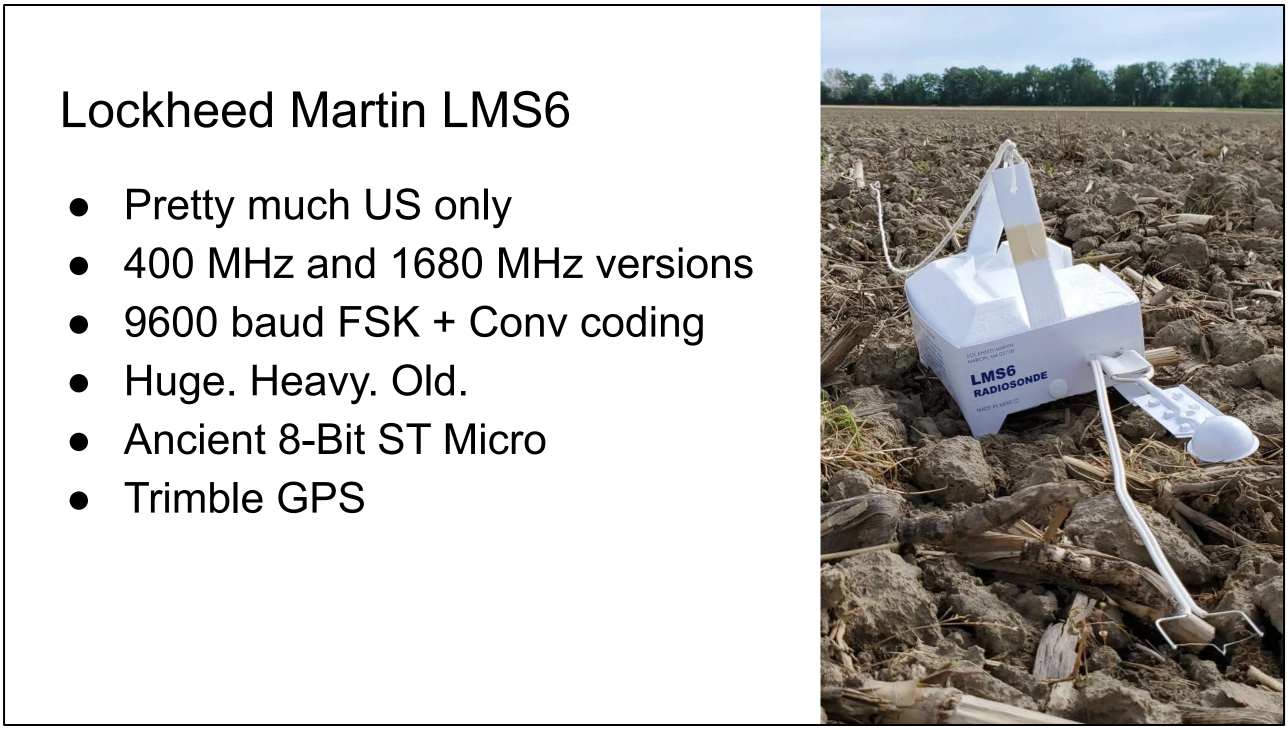
…..



There are many radiosonde types used worldwide… I’m going to focus in <click> on a few of ones you as the audience are more likely to encounter.



There are many radiosonde types used worldwide… I'm going to focus in <click> on a few of ones you as the audience are more likely to encounter.



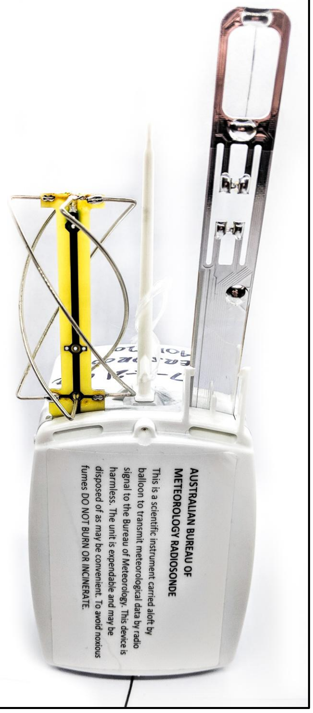
Lockheed Martin LMS6

- Pretty much US only
- 400 MHz and 1680 MHz versions
- 9600 baud FSK + Conv coding
- Huge. Heavy. Old.
- Ancient 8-Bit ST Micro
- Trimble GPS

First up the Lockheed Martin LMS6.
This is a true US style Radiosonde - It's big, and it's heavy and is a pretty old design.
It even comes with a little rolled up bag so you can send it back to the national weather service,
Though considering the case is hot glued together I'm not really sure how they plan to re-use them.
It comes in 400 MHz and 1680 MHz versions both transmitting at 9600 baud, and uses an ancient 8-bit ST Micro and a Trimble GPS.

Vaisala RS92

- Obsolete, but still in use in US (1680 MHz) & some other places (400-406 MHz)
- 4800 baud FSK (2400 + manchester)
- 400 / 1680 MHz versions
- Lots of custom silicon…
- Half-a-GPS
- Neat L-Band QFH antenna


AUSTRALIAN BUREAU OF
METEOROLOGY RADIOSONDE
This is a scientific instrument carried aloft by
balloon to transmit meteorological data by radio
signal to the Bureau of Meteorology. This device is
harmless. The unit is expendable and may be
disposed of as may be convenient. To avoid noxious
fumes DO NOT BURN OR INCINERATE.

The Vaisala RS92 is an interesting beast which is still flown in a few places around the world, including the US, Australia and Brazil, but it is being phased out due to being well and truly obsolete.

What's interesting about is is that the main microcontroller and the radio transmitter IC are custom silicon, which makes reverse engineering is somewhat difficult. Also, it doesn't have a complete GPS receiver onboard, instead the position calculations are performed at the ground station.

The little L-Band helical antenna is pretty cool though, and can be re-used for reception of iridium satellites and the like.

Graw DFM-09

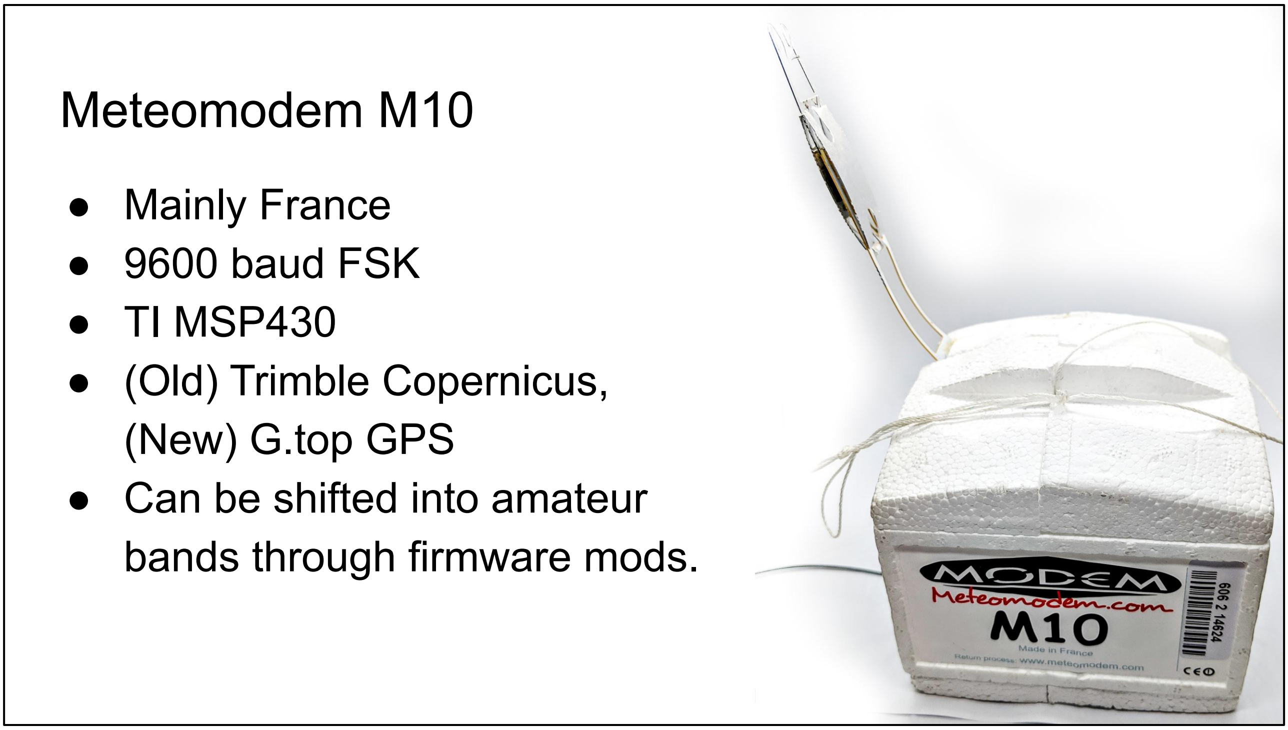
- Found in a few places around Europe
- 2500 (!?) baud FSK
- STM32F100 (same as RS41)
- Telit Jupiter JN3 GPS
- Not sure if anyone has custom firmware for these…
- Upgraded DFM17 coming real soon now

The Graw DFM-09 is mainly common in Germany, where it’s made, and runs 2500 baud FSK which kind of messes with my head a bit.
It’s got a fairly capable microcontroller and GPS receiver, but I’m not sure if anyone has had a go at writing their own firmware for these yet.
There’s also a newer and smaller version, the DFM-17, which is beginning to be seen out in the wild recently.

Intermet iMet-4

- Belgium, Israel, a few other places
- 1200 baud AFSK (really Intermet?)
- Similar size / form factor to RS41
- STM32 Cortex-M4!
- uBlox M8Q GPS
- Texas Instruments CC115 Radio
- Would be awesome for re-use if they were more common...

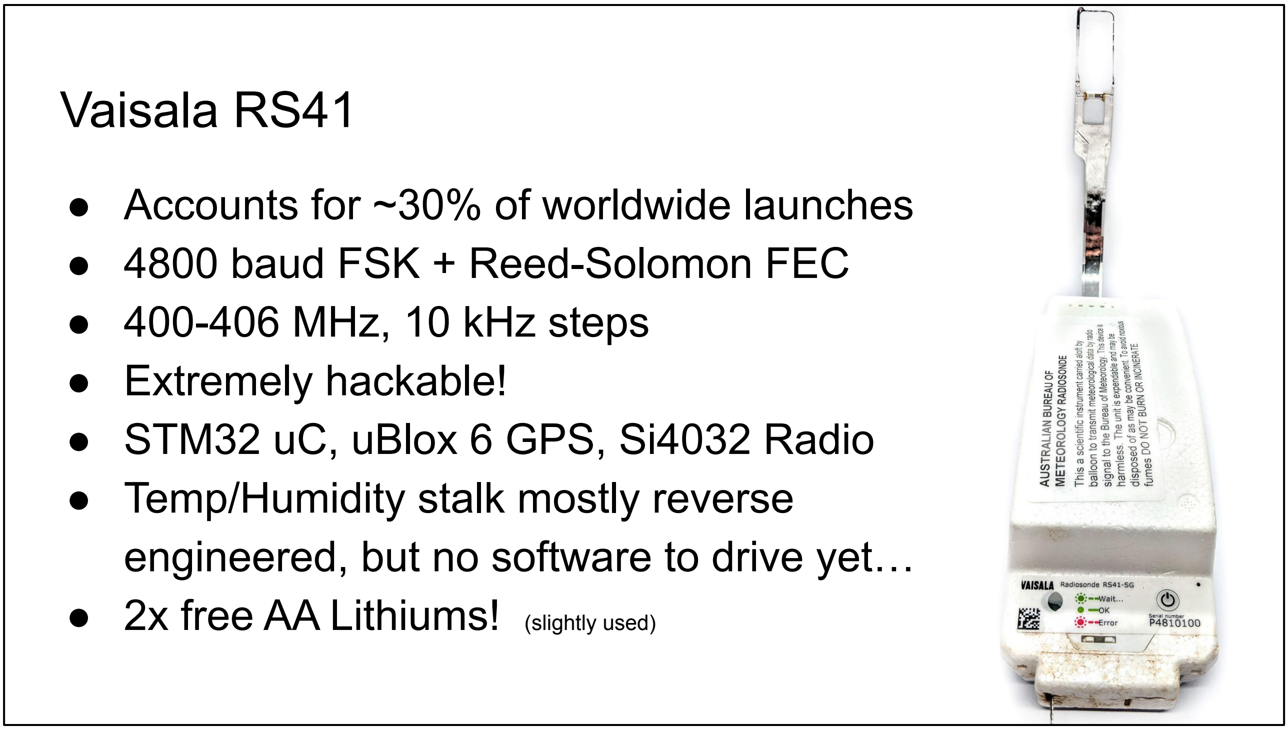
The Intermet iMet4 is a really nice radiosonde, and if they were more commonly used they would be a great candidate for re-use.
They’ve got a pretty powerful STM32, a modern uBlox chipset, and a very capable TI radio chip.
Unfortunately iMet chose to have the sonde transmit 1200 baud AFSK (as opposed to straight FSK), which throws away about 8dB of demod performance due to the double modulation. It also doesn’t transmit a unique serial number in the telemetry stream, which makes dealing with the telemetry from these a bit of a pain.



Meteomodem M10

- Mainly France
- 9600 baud FSK
- TI MSP430
- (Old) Trimble Copernicus, (New) G.top GPS
- Can be shifted into amateur bands through firmware mods.

The MeteoModem M10, made and found mostly in France, bucks the trend a little by using a TI MSP430 microcontroller, and unlike all the others, the firmware is *not* read protected, and can apparently be dumped and modified so the sonde transmits on the amateur radio 70cm band. One of these was used as the primary tracker in an amateur launch out of France a few months back.



Vaisala RS41

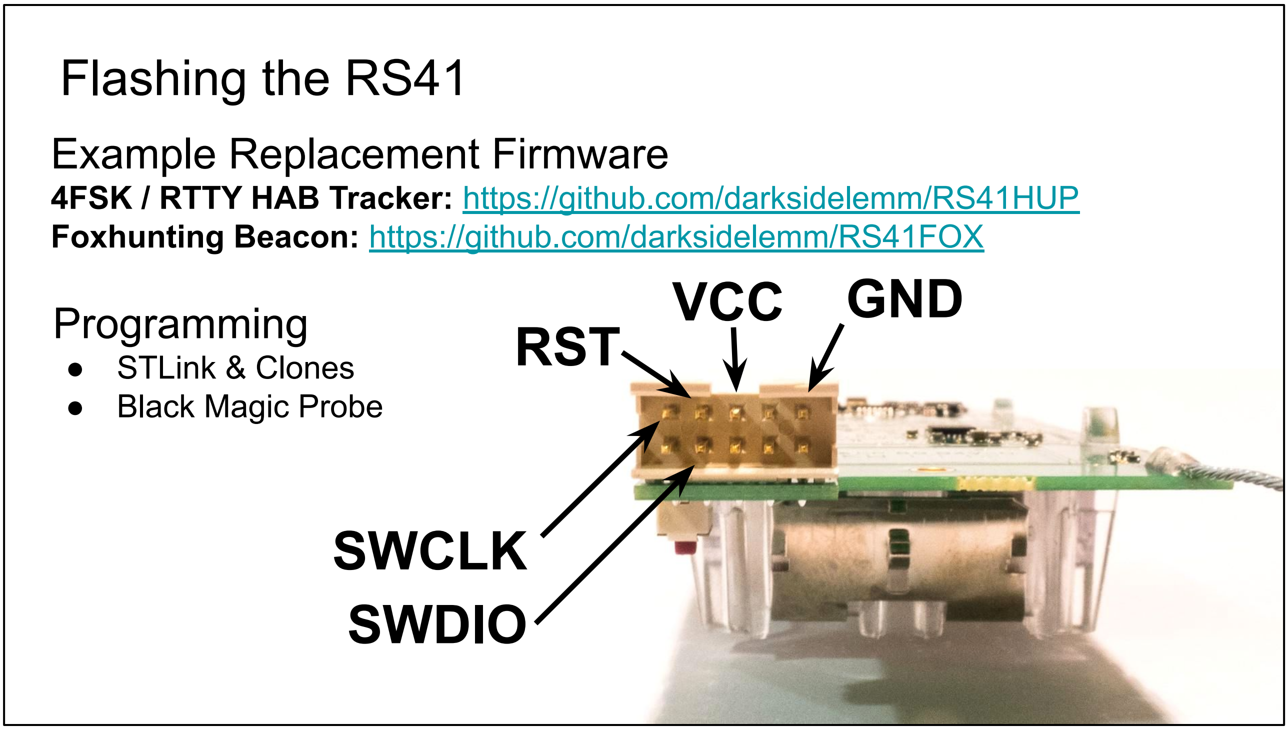
- Accounts for ~30% of worldwide launches
- 4800 baud FSK + Reed-Solomon FEC
- 400-406 MHz, 10 kHz steps
- Extremely hackable!
- STM32 uC, uBlox 6 GPS, Si4032 Radio
- Temp/Humidity stalk mostly reverse engineered, but no software to drive yet…
- 2x free AA Lithiums! (slightly used)

AUSTRALIAN BUREAU OF METEOROLOGY RADIOSONDE
This a scientific instrument carried aloft by balloon to transmit meteorological data by radio signal to the Bureau of Meteorology. This device is harmless. The unit is expendable and may be disposed of as may be convenient. To avoid noxious fumes DO NOT BURN OR INCINERATE
VAISALA Radiosonde RS41-SG
Wait...
OK
Error
Serial number
P4810100

Last, but definitely not least, the Vaisala RS41, or as I like to call it: “The STM32 development board that literally falls from the sky”.

According to WMO stats, the RS41 is launched from 186 sites around the world, accounting for nearly 30% of all worldwide launches, making it the most common radiosonde.

And being the most common, it’s had the most amount of reverse engineering applied to it. Thankfully, it’s nothing like it’s predecessor, and apart from the sensor stalk, uses off-the-shelf parts. Those parts, being a STM32F100 micro, a uBlox 6 GPS and a Si4032 radio, are getting on a bit, but are pretty well known, having also been used in the HAB community for a while. The temperature and humidity sensors are a bit of a mystery still, though some good reverse engineering work has been done, and it hopefully won’t be too long before we see some software to drive them. You also get 2x free Lithium AA batteries, usually at about 80% capacity if you recover the sonde shortly after landing.



Flashing the RS41

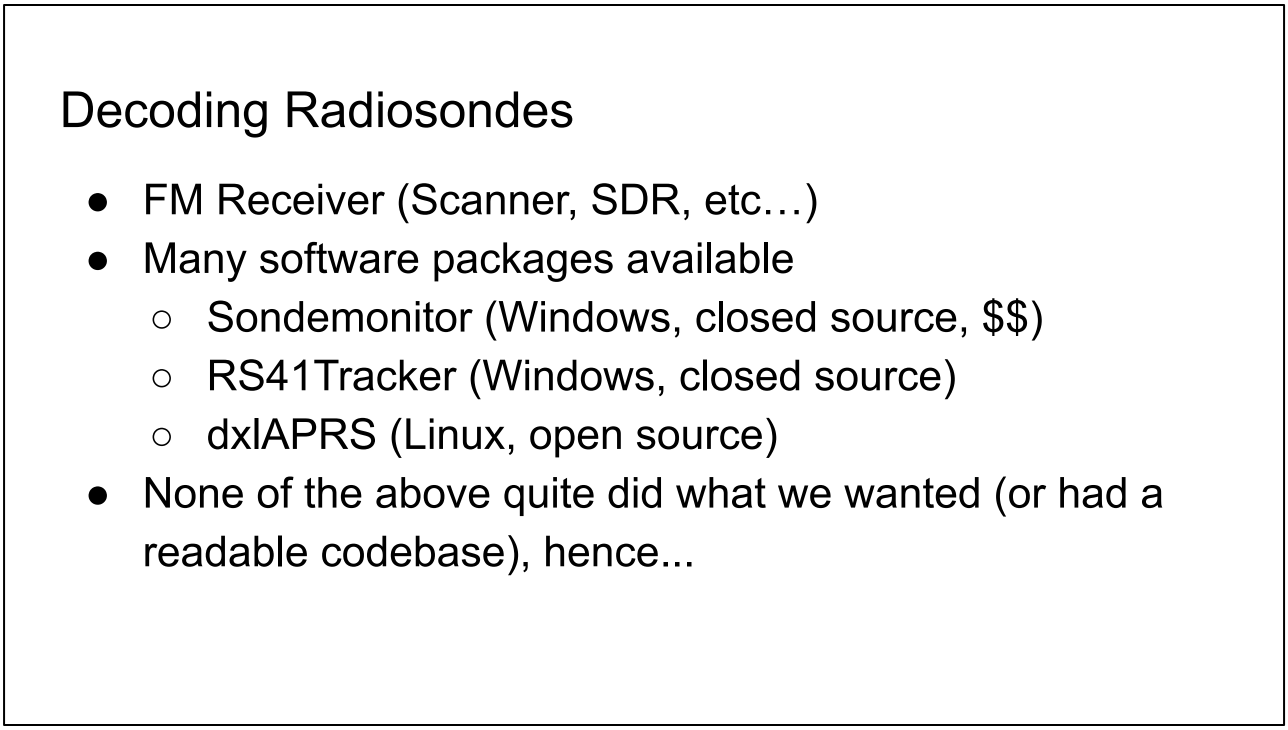
Example Replacement Firmware

4FSK / RTTY HAB Tracker: https://github.com/darksidelemm/RS41HUP
Foxhunting Beacon: https://github.com/darksidelemm/RS41FOX

Programming

- STLink & Clones
- Black Magic Probe

Reflashing a RS41 is easy, as Vaisala handily broke out all the programming pins to a header at the bottom of the board. Grab some firmware off GitHub, hook up a STLink programmer and away you go!



Decoding Radiosondes

- FM Receiver (Scanner, SDR, etc…)
- Many software packages available
  - Sondemonitor (Windows, closed source, $$)
  - RS41Tracker (Windows, closed source)
  - dxlAPRS (Linux, open source)
- None of the above quite did what we wanted (or had a readable codebase), hence...

All of the radiosonde I've shown so far can be decoded by multiple software packages.
SondeMonitor is a pretty venerable radiosonde decoding package and decodes all of the above, and has been around for at least 12 years, but is closed source and costs about 25 quid. The developer is a really nice guy though!
RS41Tracker is a fairly recent one targeted specifically at RS41s, and has a neat little map and can do some APRS upload stuff. It is free, but it's also closed source.
The dxlAPRS tracking suite has also been around for a while, and works with RS41, RS92 and DFM sondes, and is open source, but its code base is best described as opaque and is difficult to modify.

When I first started seriously getting into radiosondes, none of the above did quite what I was after (or in the case of RS41Tracker, even existed), so I ended up writing my own.

Radiosonde Auto-RX

- Automatically Scan, Receive, & Decode Radiosonde telemetry using RTLSDR
- Python wrapper around sonde demodulators by rs1729
- Well tested FSK demod by David Rowe
- Supports Vaisala, Graw, Meteomodem, iMet, LMS6
- Targeted at RPi 2 / 3 Platform
- Currently **~132 RX stations** worldwide

github.com/projecthorus/radiosonde_auto_rx

The main reason was because here in Australia, radiosonde frequencies are not particularly consistent, and would change from launch-to-launch.
Needed a way to scan the radiosonde band to automatically find and decode sondes.
I based this around a set of open source sonde decoders written by rs1729, and wrote python code to automatically scan for sondes, and decode and upload telemetry to the web. It currently supports all the sonde types i've mentioned in this presentation, with support for more being actively developed. I've also recently started to use David Rowe's frequency shift keying demodulator, which helps with decode performance.

When I presented on this at the beginning of the year there were 60 stations running, as of today that number is 132, which freaks me out a little bit.

How to get started...

Hardware

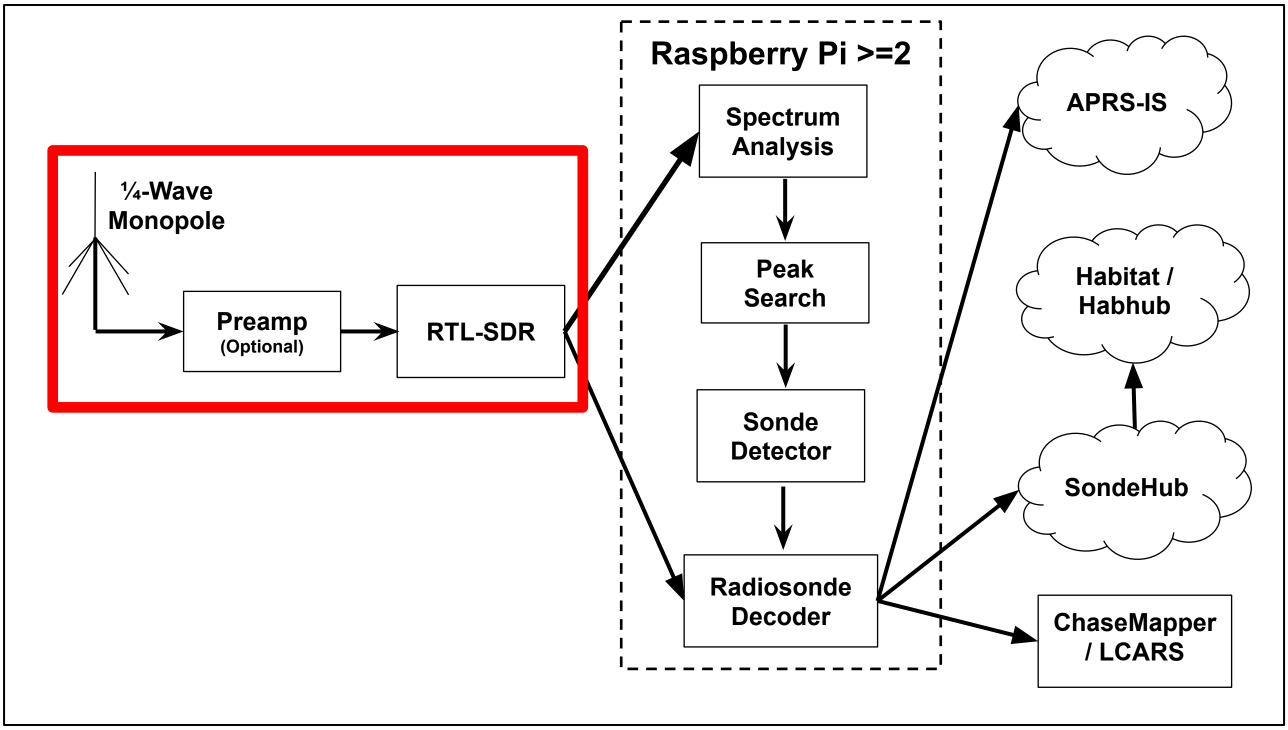
- Raspberry Pi 2 or 3 (or some other Linux PC)
- RTLSDR (*with TCXO* - need frequency stability!)
- Antenna (¼-wave) + (optional) Preamp
- Total cost: < £100

RTL-SDR.COM
QUICKSTART SETUP GUIDE: RTL-SDR.COM/QSG
DVB-T+DAB+FM+SDR
RTL2832U R820T2 TCXO+BIAS T+HF
V.3

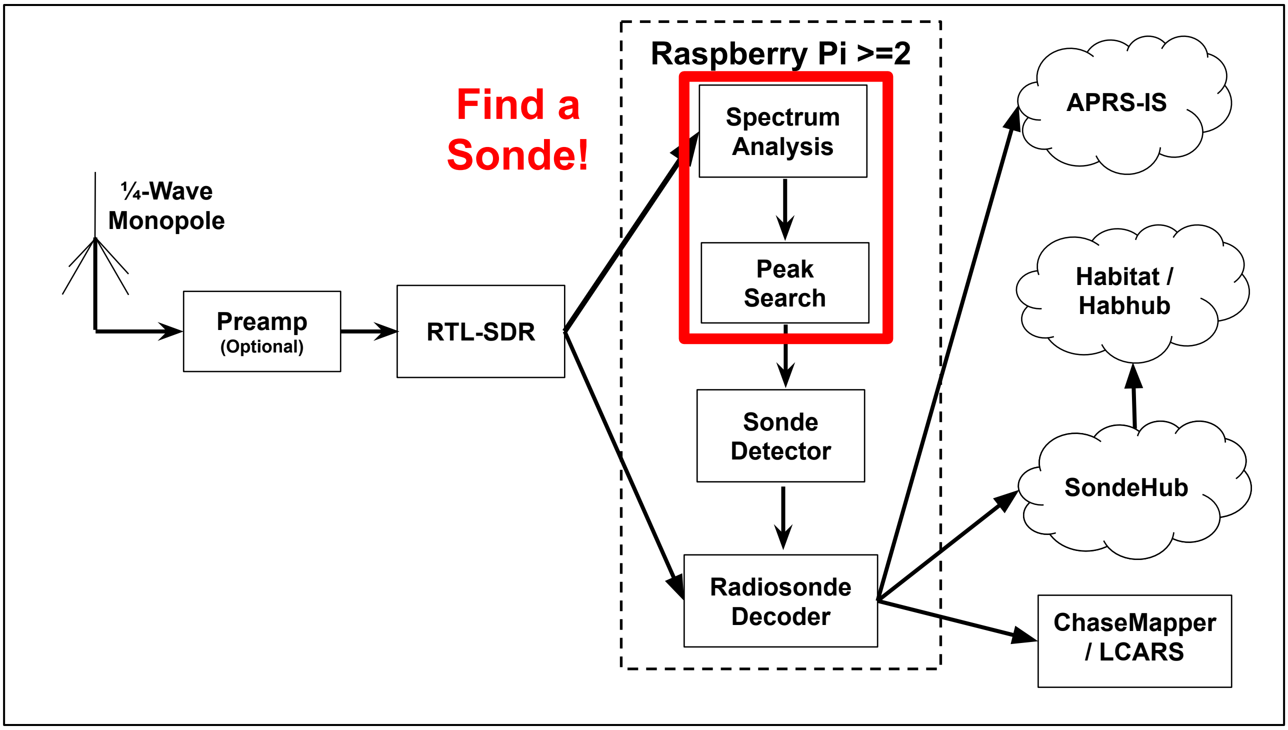
Software

- Guide: https://github.com/projecthorus/radiosonde_auto_rx/wiki

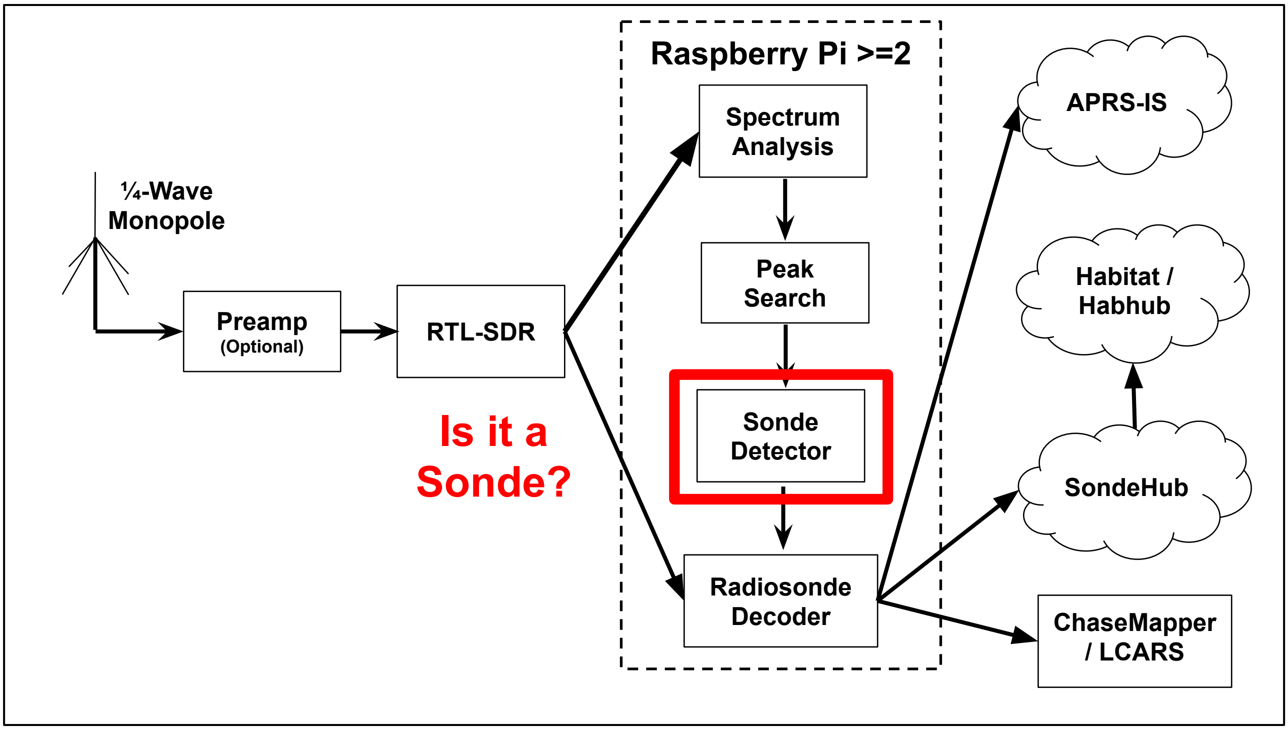
All of the code is available on github, with an installation guide available at the link. To get started, you need some kind of Linux machine, an RTLSDR, and an antenna. I recommend running auto_rx on a Raspberry Pi, and with one of the ‘version 3’ RTLSDRs. There’s also excellent preamps designed for the radiosonde band available from uputronics, using one of these is not mandatory, but it can help.



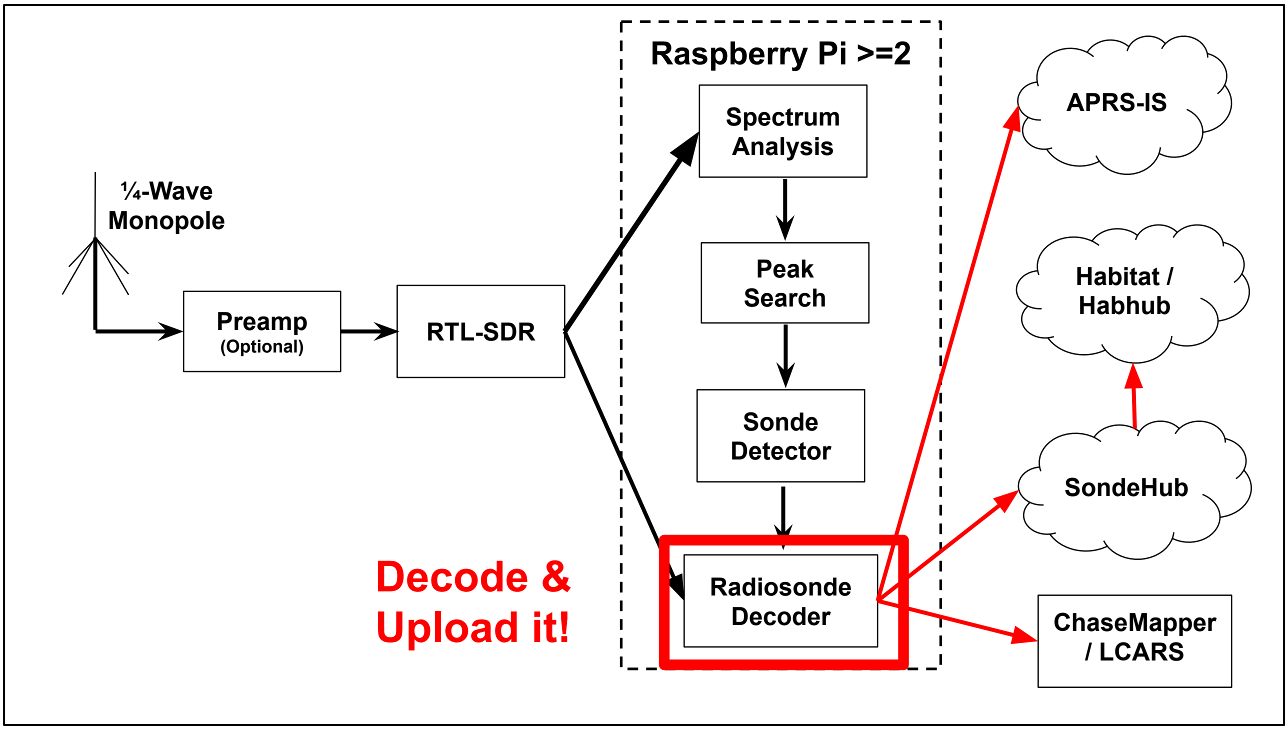
So here’s a bit of a flowchart showing how auto_rx operates.
On the left we have the RF hardware, capturing the radio signal and converting it into digital samples which we can process.



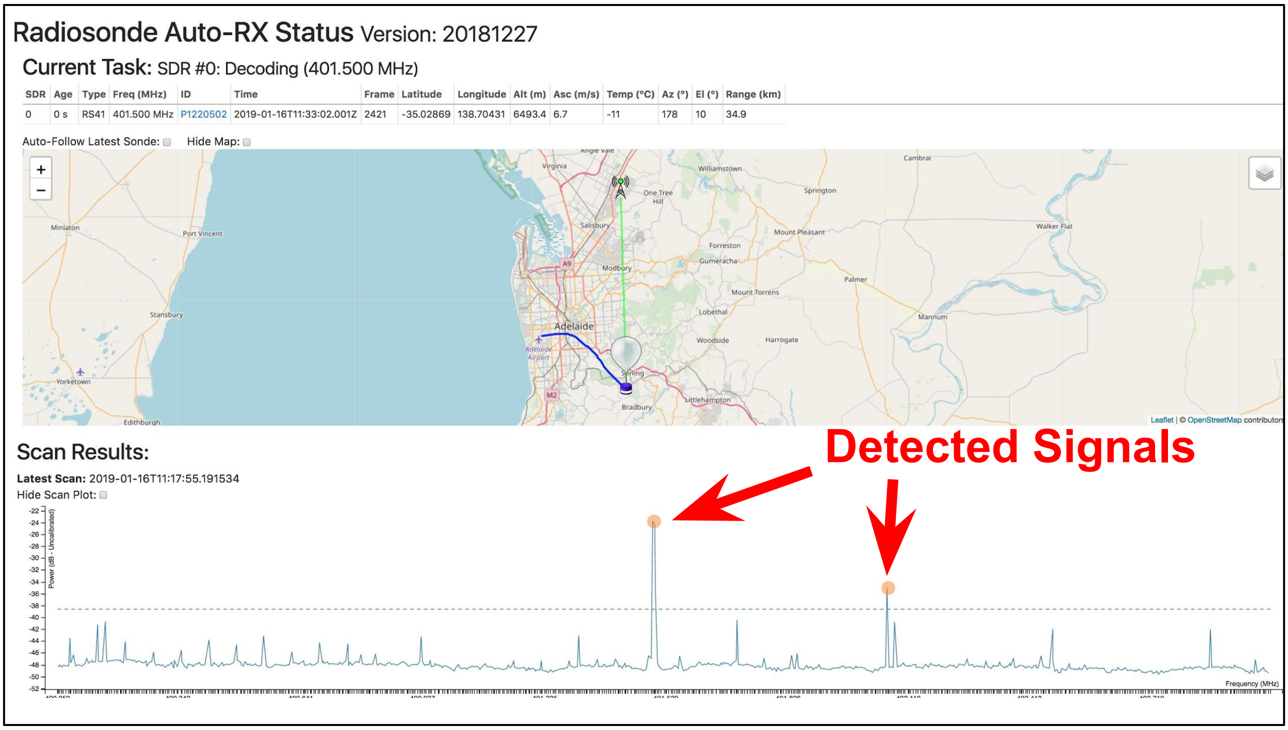
The first thing auto_rx does with this data is scan through the radiosonde band and look for signal which are above a set signal-to-noise ratio threshold.



Once it has a list of peaks, it looks at each of them in turn and attempts to determine if the signal is a radiosonde.
It does this by cross-correlating the receiver signal with a set of known reference signals. If the correlation score is above a particular threshold, then the signal is most likely a radiosonde.



Auto_rx then locks on to that radiosonde and starts decoding it.
The received telemetry data can be uploaded to the APRS Internet Service, to the Habhub tracker via our ‘Sondehub’ service, which I’ll get to later,
And also to local mapping services, such as my chasemapper browser-based map, and Dave’s LCARS mapping app.



Radiosonde Auto-RX Status Version: 20181227

Current Task: SDR #0: Decoding (401.500 MHz)

SDR	Age	Type	Freq (MHz)	ID	Time	Frame	Latitude	Longitude	Alt (m)	Asc (m/s)	Temp (°C)	Az (°)	El (°)	Range (km)
0	0 s	RS41	401.500 MHz	P1220502	2019-01-16T11:33:02.001Z	2421	-35.02869	138.70431	6493.4	6.7	-11	178	10	34.9

Auto-Follow Latest Sonde: □ Hide Map: □

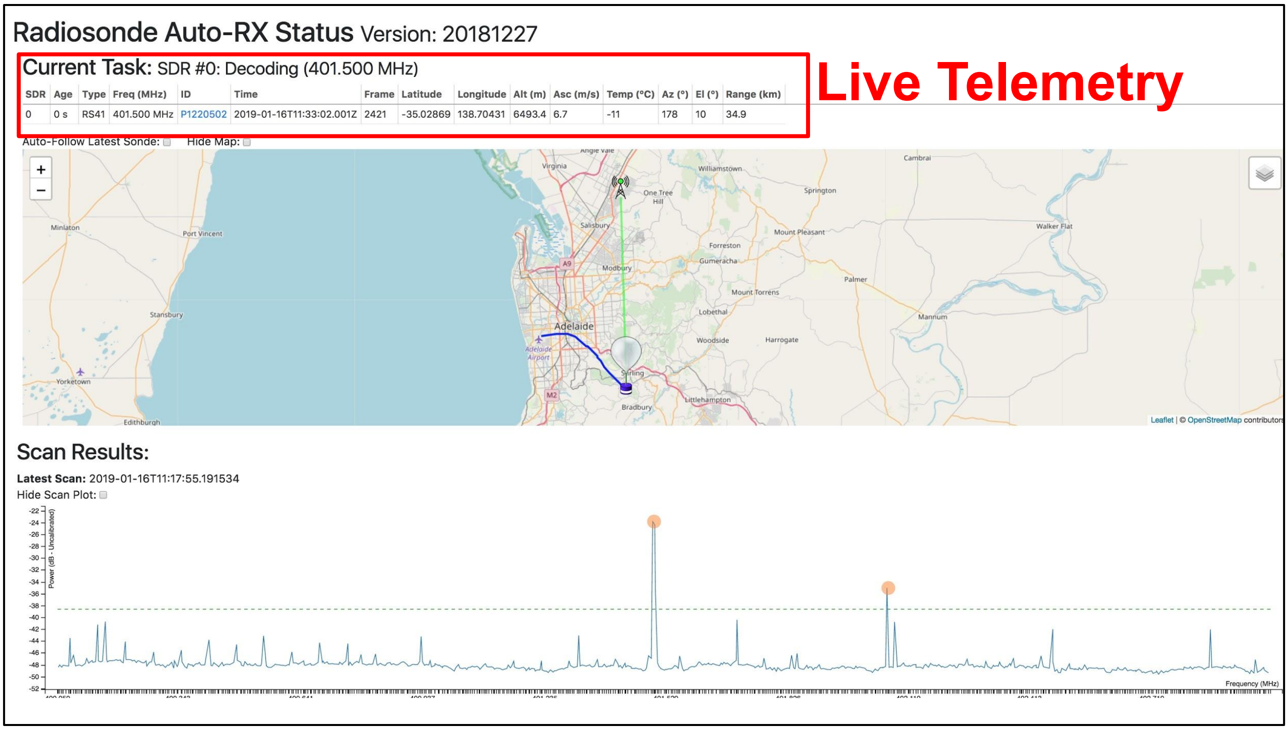
Scan Results:

Latest Scan: 2019-01-16T11:17:55.191534

Hide Scan Plot: □

Detected Signals

Auto_rx also has a web interface that shows results of the scanner step,



Radiosonde Auto-RX Status Version: 20181227

Current Task: SDR #0: Decoding (401.500 MHz)

SDR	Age	Type	Freq (MHz)	ID	Time	Frame	Latitude	Longitude	Alt (m)	Asc (m/s)	Temp (°C)	Az (°)	El (°)	Range (km)
0	0 s	RS41	401.500 MHz	P1220502	2019-01-16T11:33:02.001Z	2421	-35.02869	138.70431	6493.4	6.7	-11	178	10	34.9

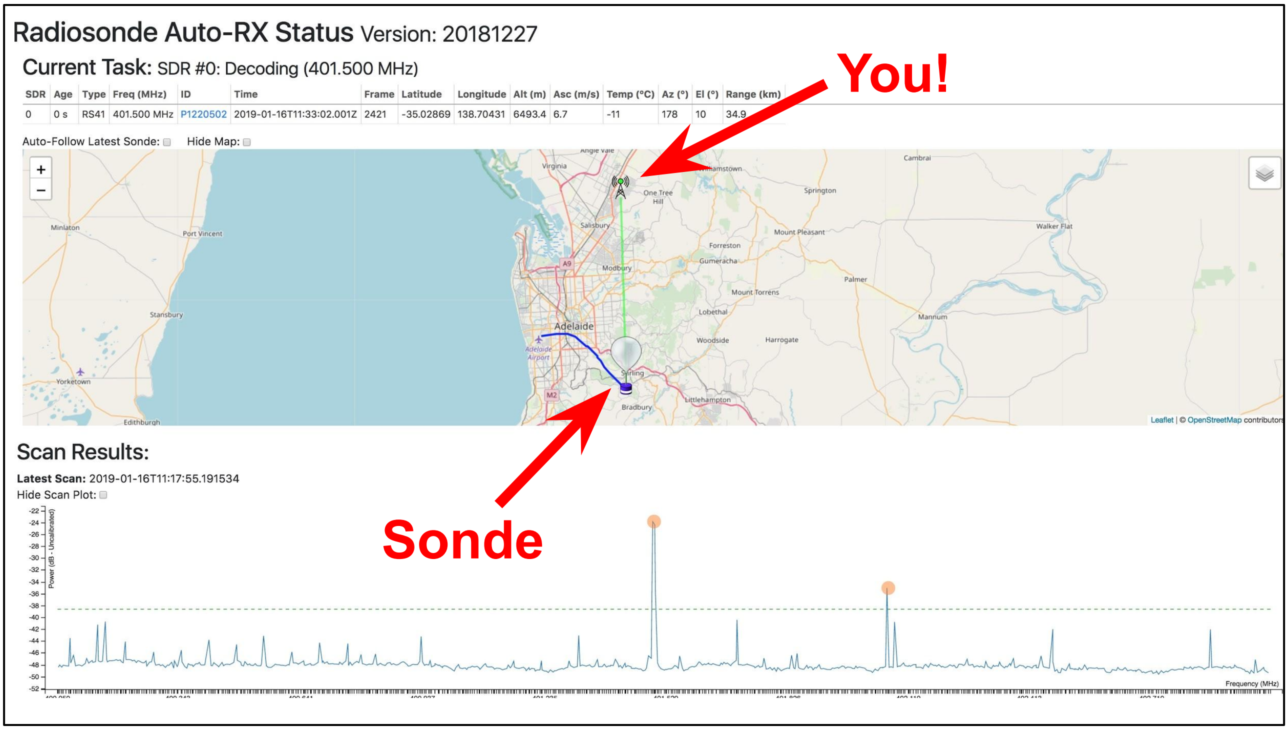
Auto-Follow Latest Sonde: □ Hide Map: □

Scan Results:

Latest Scan: 2019-01-16T11:17:55.191534

Hide Scan Plot: □

… any telemetry that’s currently being decoded, and



Radiosonde Auto-RX Status Version: 20181227

Current Task: SDR #0: Decoding (401.500 MHz)

SDR	Age	Type	Freq (MHz)	ID	Time	Frame	Latitude	Longitude	Alt (m)	Asc (m/s)	Temp (°C)	Az (°)	El (°)	Range (km)
0	0 s	RS41	401.500 MHz	P1220502	2019-01-16T11:33:02.001Z	2421	-35.02869	138.70431	6493.4	6.7	-11	178	10	34.9

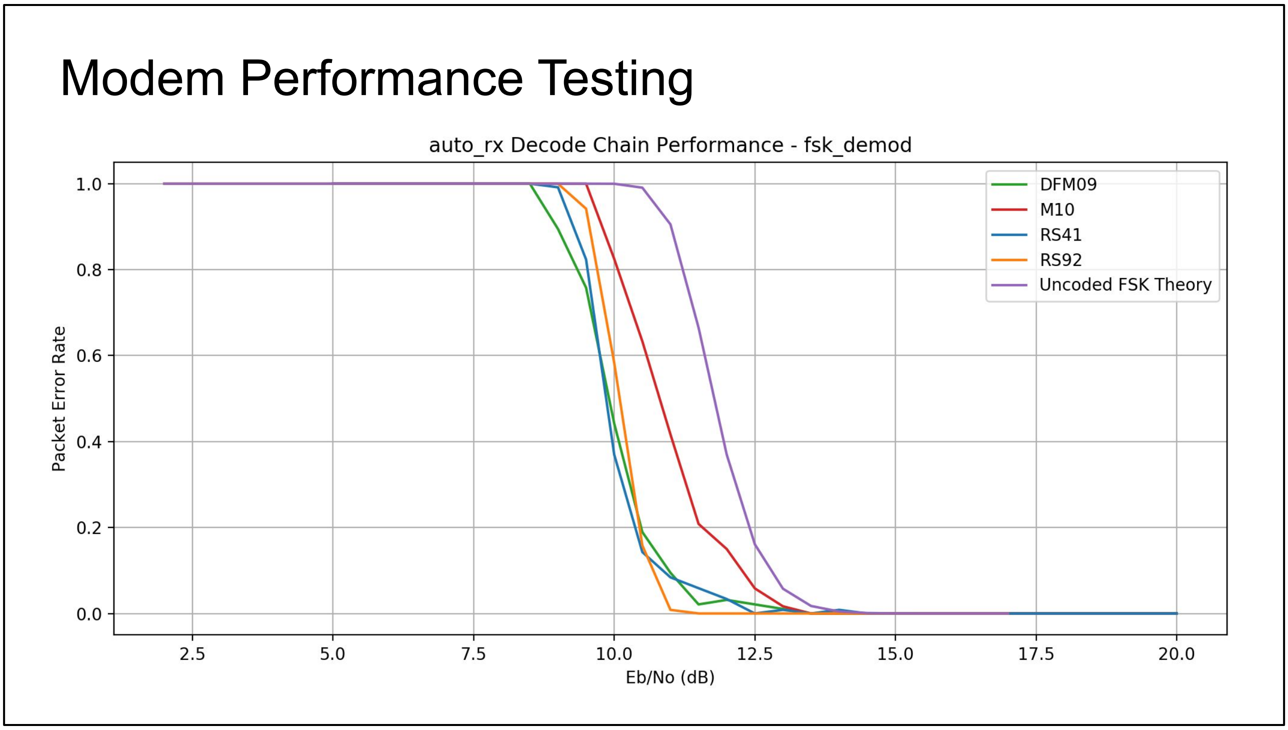
Auto-Follow Latest Sonde: □ Hide Map: □

Scan Results:

Latest Scan: 2019-01-16T11:17:55.191534

Hide Scan Plot: □

Where the balloon currently is on a map.
If you want to decode more than one radiosonde, you can currently do this by adding more RTLSDRs.

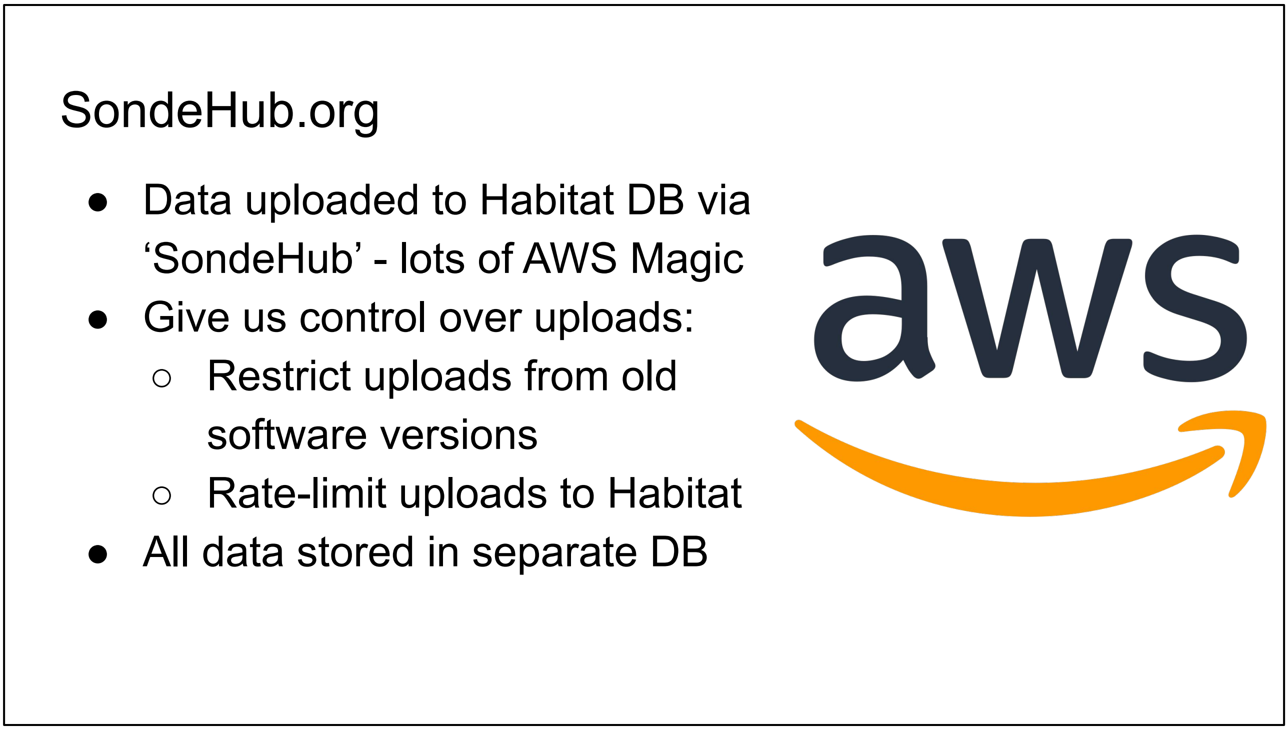


Modem Performance Testing

Something i’ve learnt from working with David Rowe is a modem is worthless without performance measurements. So, within auto_rx I’ve spent a lot of effort on testing the performance of the FSK demodulators to make sure they are up to scratch.

This plot shows the packet error rate of a few of the modems vs SNR, along with the theoretical performance of un-coded FSK.

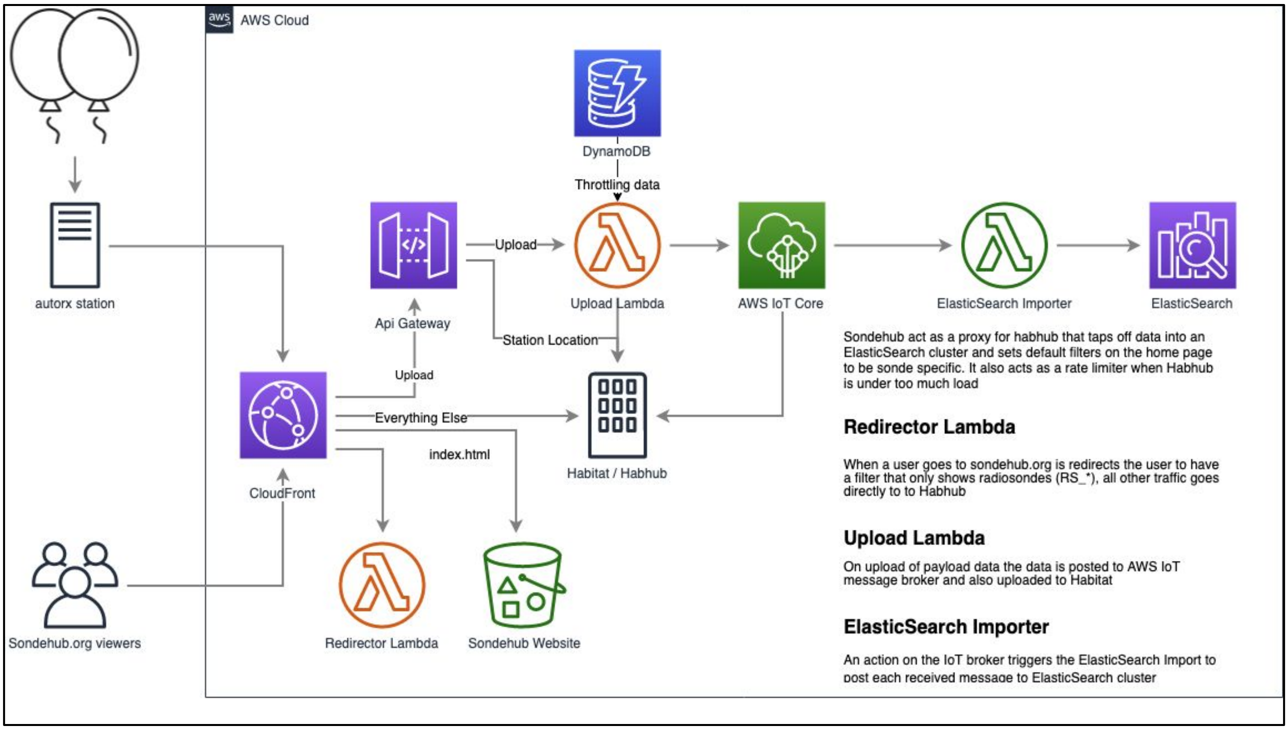
Yes, all of them appear to perform *better* than theory, but this is just because all of the radiosondes are using some form of forward error correction, giving them a few dB of coding gain.



SondeHub.org

- Data uploaded to Habitat DB via ‘SondeHub’ - lots of AWS Magic
- Give us control over uploads:
  - Restrict uploads from old software versions
  - Rate-limit uploads to Habitat
- All data stored in separate DB

As i mentioned before, data that is uploaded to the Habitat goes via our ‘Sondehub’ service.
Sondehub is a whole lot of amazon web service magic which gives us a lot of control over data flows.
We can restrict uploads from old versions of auto_rx, rate-limit users that might be uploading too quickly, and store a copy of the telemetry in our own database.



Sondehub act as a proxy for habhub that taps off data into an ElasticSearch cluster and sets default filters on the home page to be sonde specific. It also acts as a rate limiter when Habhub is under too much load

Redirector Lambda

When a user goes to sondehub.org is redirects the user to have a filter that only shows radiosondes (RS_*), all other traffic goes directly to to Habhub

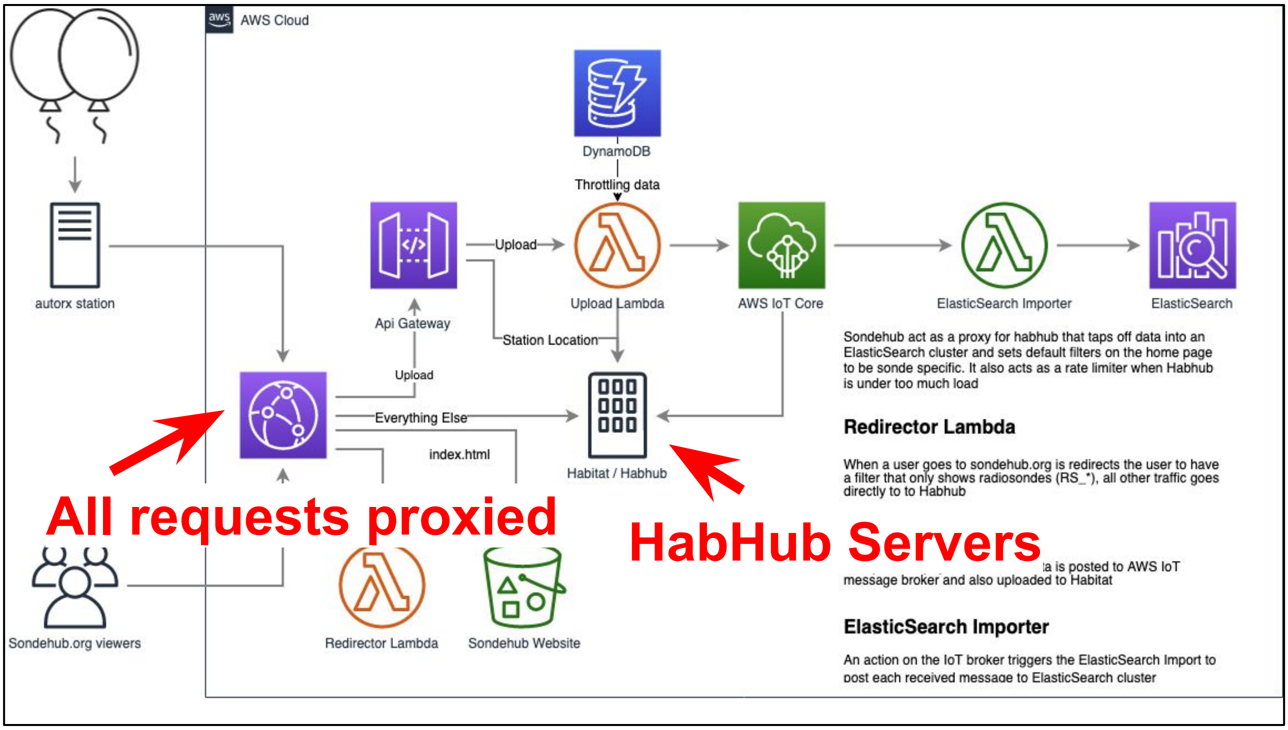
Upload Lambda

On upload of payload data the data is posted to AWS IoT message broker and also uploaded to Habitat

ElasticSearch Importer

An action on the IoT broker triggers the ElasticSearch Import to post each received message to ElasticSearch cluster

.. the specifics of what’s going on within AWS is pretty complicated, and this part is more Michael Wheeler’s doing, but I’ll do my best at explaining it.



Sondehub act as a proxy for habhub that taps off data into an ElasticSearch cluster and sets default filters on the home page to be sonde specific. It also acts as a rate limiter when Habhub is under too much load

Redirector Lambda

When a user goes to sondehub.org is redirects the user to have a filter that only shows radiosondes (RS_*), all other traffic goes directly to to Habhub

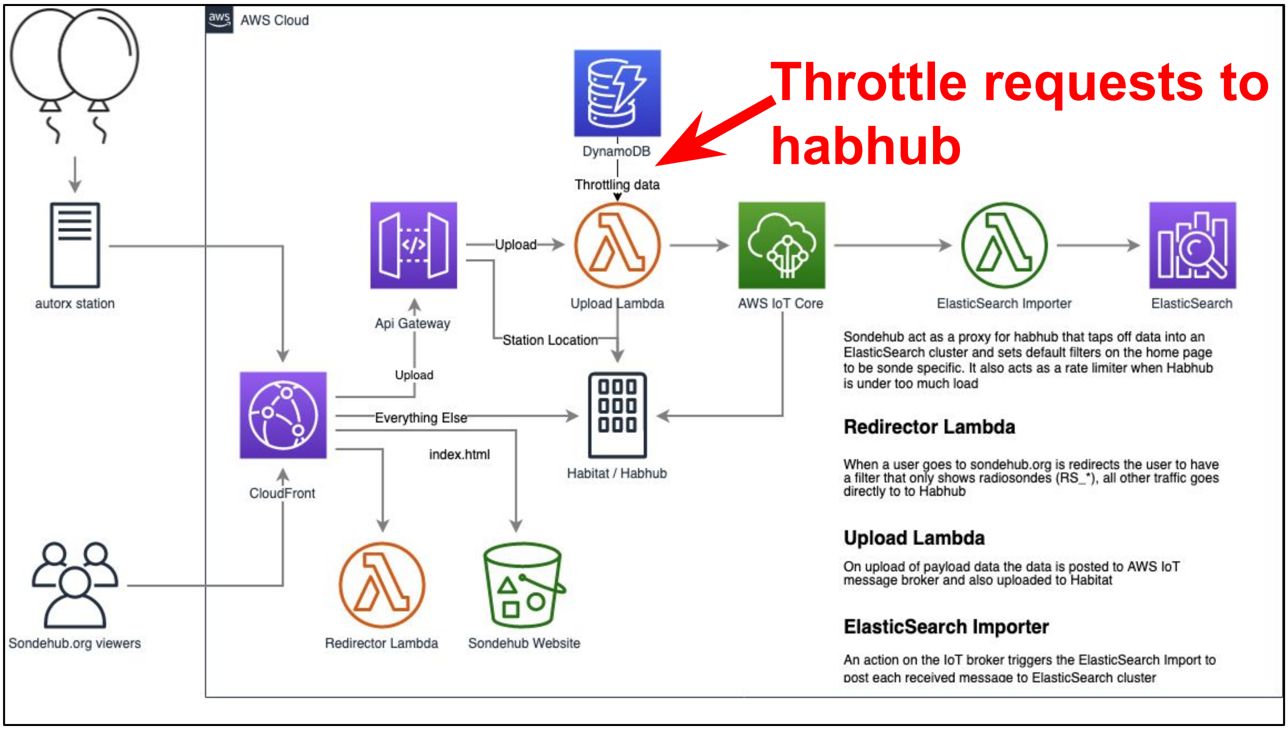
ta is posted to AWS IoT message broker and also uploaded to Habitat

ElasticSearch Importer

An action on the IoT broker triggers the ElasticSearch Import to post each received message to ElasticSearch cluster

So the Sondehub AWS system is essentially acting as a proxy between clients, and the HabHub servers, clients being auto_rx stations, or just anyone visiting sondehub.org.

When someone visits sondehub.org, they get redirected to the tracker website, but we automatically set the filtering to be radiosonde specific, and we also replace the google maps API key with our own, to avoid using up the habhub API key.



Throttle requests to habhub

Sondehub act as a proxy for habhub that taps off data into an ElasticSearch cluster and sets default filters on the home page to be sonde specific. It also acts as a rate limiter when Habhub is under too much load

Redirector Lambda

When a user goes to sondehub.org is redirects the user to have a filter that only shows radiosondes (RS_*), all other traffic goes directly to to Habhub

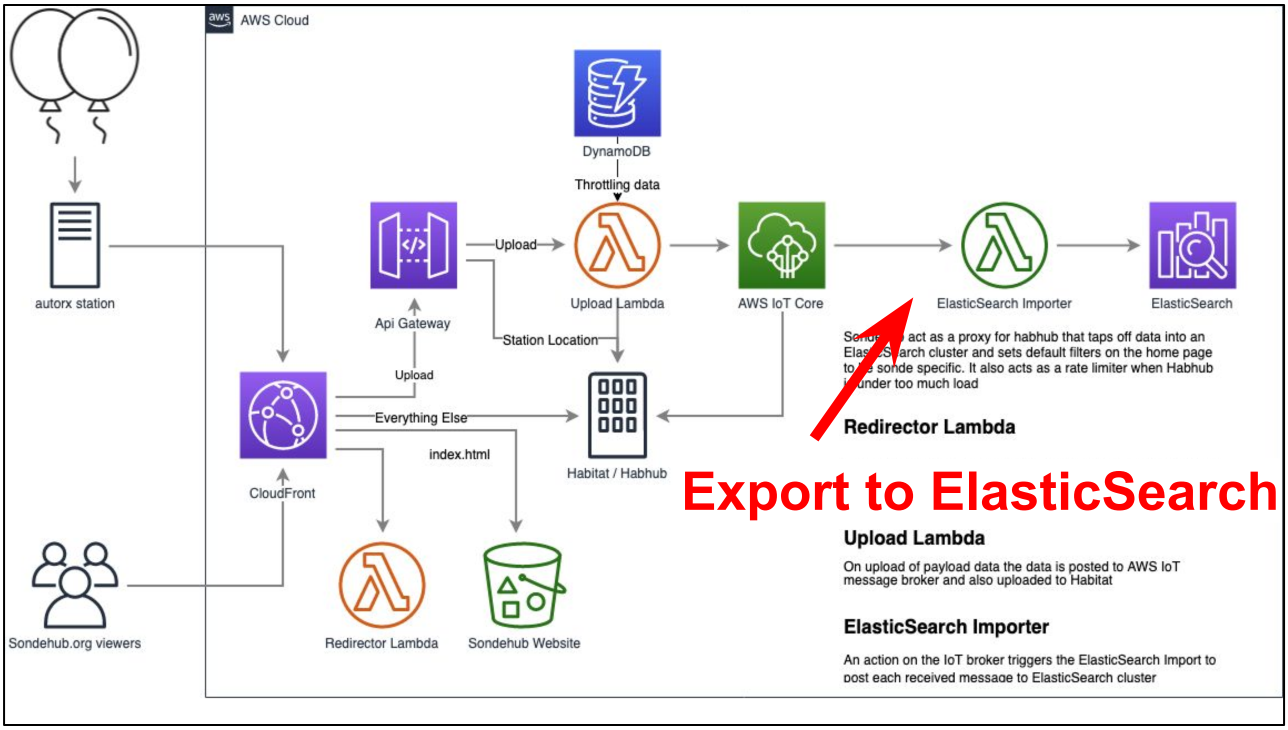
Upload Lambda

On upload of payload data the data is posted to AWS IoT message broker and also uploaded to Habitat

ElasticSearch Importer

An action on the IoT broker triggers the ElasticSearch Import to post each received message to ElasticSearch cluster

All telemetry uploads go via an AWS API Gateway, which lets us apply filtering based on various parameters, like auto_rx version number or callsign. Telemetry is then buffered in a database before being uploaded to habhub. This lets us enable on-demand rate limiting of uploads to the habitat DB, which helps avoid overloading the habitat telemetry parser.



Sondehub to act as a proxy for habhub that taps off data into an ElasticSearch cluster and sets default filters on the home page to be sonde specific. It also acts as a rate limiter when Habhub is under too much load

Redirector Lambda

Export to ElasticSearch

Upload Lambda

On upload of payload data the data is posted to AWS IoT message broker and also uploaded to Habitat

ElasticSearch Importer

An action on the IoT broker triggers the ElasticSearch Import to post each received message to ElasticSearch cluster

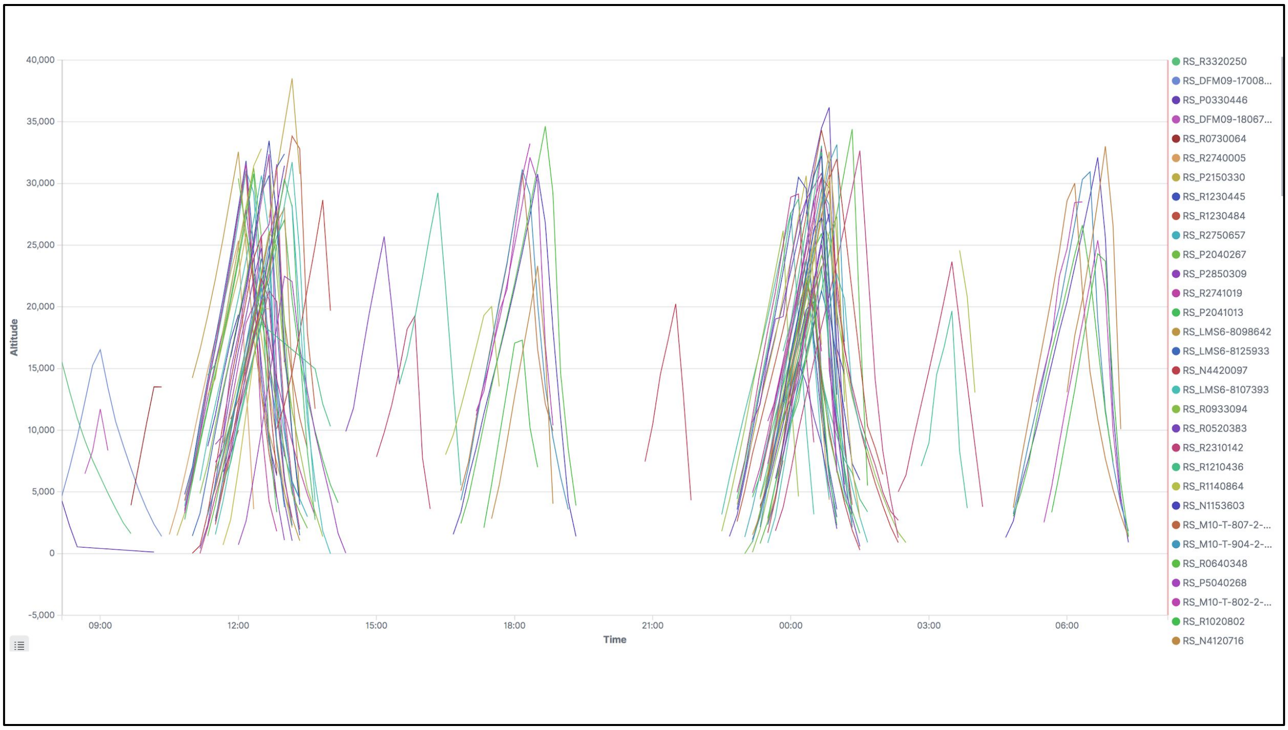
Finally, all telemetry data is also saved to an ElasticSearch DB
Currently we’re not doing much with this data, but the aim is to eventually create a separate tracker instance pointing at it, instead of at the habitat DB.

Total cost 35 quid /month, most of that is Elasticsearch, which is currently storing telemetry from about 33000 radiosonde flights.

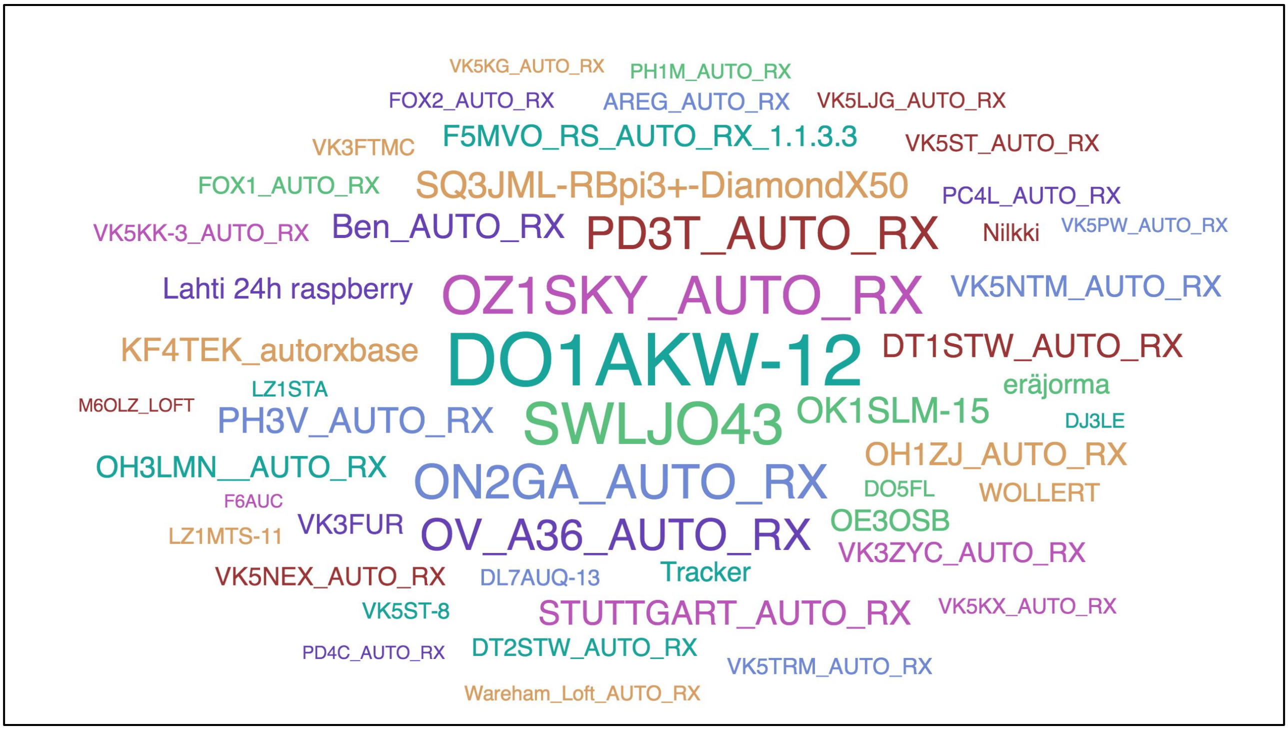
SondeHub Interface Layer - Other Features

- Short links to sonde tracks:
  - ○ sondehub.org/R2310144 → Habhub Tracker
  - ○ sondehub.org/aprs/R2310144 → aprs.fi
- Websockets / MQTT support allows live streaming of data to web clients.
- Analytics!

Sondehub.org also provides a way of providing a short link to a particular radiosonde track, be it on habhub or aprs.fi,
And there’s also a MQTT interface which allows live-streaming of incoming telemetry data to web clients. We eventually plan on using this to drive a separate instance of the habhub tracker.
Also… we can do analytics!

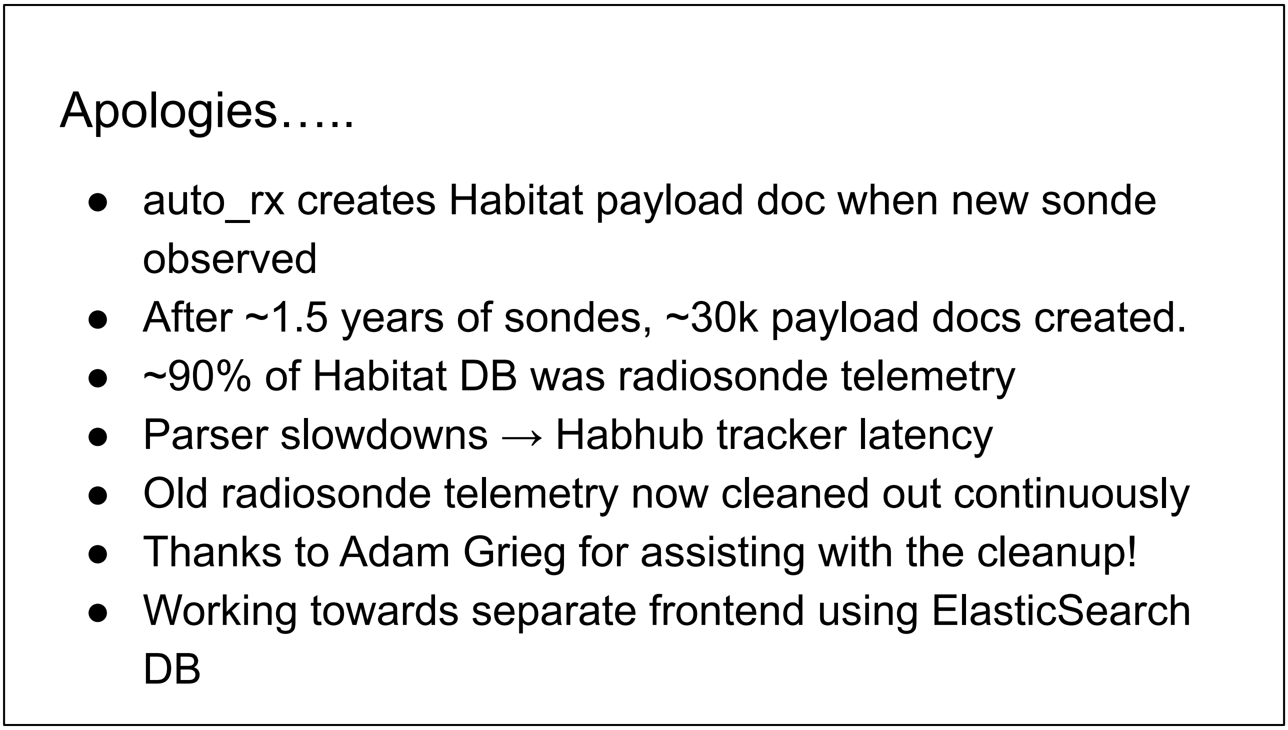


For example, here's a rather noisy plot of the altitude of all radiosondes observed over the last day. We can clearly see lots of sondes around 00 and 12 zulu, along with a few at other times. It also looks like one of the sondes reached 38km altitude, which is pretty unusual.



We can also produce callsign clouds! Here's a few of the most prolific telemetry uploaders over the last day.

Currently the ElasticSearch DB is publically available for a few APIs, but isn't well documented. If you want to have a play with some of the data, come find me in the #habhub channel on IRC or e-mail me.



Apologies…..

- auto_rx creates Habitat payload doc when new sonde observed
- After ~1.5 years of sondes, ~30k payload docs created.
- ~90% of Habitat DB was radiosonde telemetry
- Parser slowdowns → Habhub tracker latency
- Old radiosonde telemetry now cleaned out continuously
- Thanks to Adam Grieg for assisting with the cleanup!
- Working towards separate frontend using ElasticSearch DB

Finally, I do need to make a big apology for how the increasing popularity of auto_rx almost brought the habitat DB to its knees over the last few months.

After about one-and-a-half years of radiosonde tracking, there were upwards of 30000 payload documents added into the habitat database, which resulted in the telemetry parser to be be much slower than usual. This combined with habitat database cleanup operations resulted in the considerable latency in the tracker.

However, thanks to Adam Grieg there is now a script which continuously removes any radiosonde data older than 2 weeks from the habitat db, which has brought the database performance back to normal.

We are working towards transitioning away from the use of habitat, and to our own database and mapping system.

radiosonde autorx

Mark Jessop | VK5QI | vk5qi@rfhead.net
Michael Wheeler | VK3FUR | m@mwheeler.org

github.com/projecthorus/radiosonde_auto_rx