Project Horus DVB-S Payload Reception Notes

Author: Mark Jessop
Last Updated: 2021-02-18

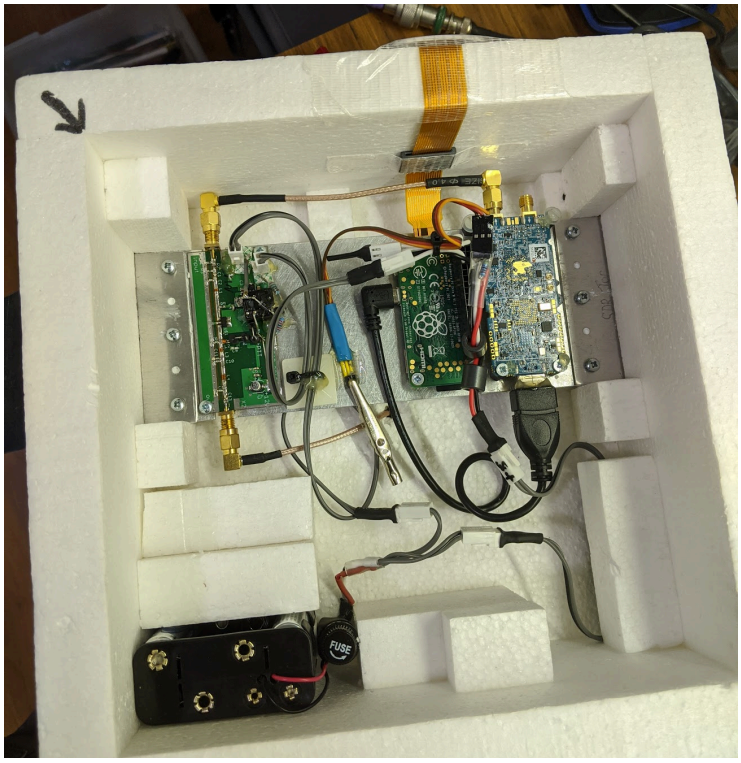


Figure 1 - DVB-S Payload Internals. Left-to-right: G4BAO 70cm driver amp, Raspberry Pi Zero W, LimeSDR Mini. 12v battery pack at bottom-left.

Introduction

Upcoming Project Horus launches will feature a first for the project - live video! While we have launched many GoPros and still image cameras in the past, live video was always something on the bucket list. With modern advancements in Software-Defined Radios, it's become relatively easy to piece together a digital video transmitter suitable for use in a high-altitude balloon payload.

This document provides some information about the transmitter, and what equipment and software is required to receive the live video during one of our balloon launches. The first few launches will be using relatively low data rates, such that reception should be possible with a fairly modest receiver setup.

Transmitter Payload Overview

The aim of this payload is to capture live video, perform video compression, and then modulate the video onto a RF signal. I have chosen to use the DVB-S (Digital Video Broadcast - Satellite) standard for a few reasons:

- It's widely supported by a range of free and open-source software, both on the transmit and receive sides.
- DVB-S uses single-carrier QPSK, which has somewhat laxer requirements on amplifier linearity than multi-carrier standards like DVB-T.
- It's possible to encode DVB-S on a low-power platform such as a Raspberry Pi Zero W (DVB-S2 is a bit much for it)

Unlike many other groups that have launched video transmitter payloads on the 23cm band, I've chosen to run ours on the 70cm band. We have plenty of spectrum between 440 and 450 MHz which is relatively un-used, and we know all the local users of it (EARC's DVB-T repeater, and the DVB-T repeater up at Pt Pirie) and can coordinate with them. Using 70cm also means we have less path loss to deal with, increasing our chances of getting a decent video stream.

The DVB-S payload is based around a LimeSDR Mini, running custom firmware developed by F5OEO to help offload some of the modulation work onto the LimeSDR's FPGA. Capture and compression (H264) of video is performed by a Raspberry Pi Zero W, using a PiCam v2.1, and running open-source encoding software available here: https://github.com/darksidelemm/dvbsdr
The output signal from the LimeSDR mini (~10 mW) is amplified to approx 800mW using a G4BAO 70cm driver amplifier.
The RPi, LimeSDR Mini, and amplifier are all mounted to an aluminium heat spreader plate manufactured by Peter VK5KX, which helps dissipate the ~7W of heat the entire system produces during operation. Software on the RPi monitors the temperature of the heat spreader plate and will cut power to the amplifier and LimeSDR (~6W of power) if it gets dangerously hot.

The entire payload draws about 8W for the <1W of RF power it generates - not particularly efficient! It's expected to have about 4 hours of battery life, more than enough for a typical High-Altitude Balloon flight.

For the first launch, we are going to use the following transmitter settings:

- **Transmit Frequency:** 445 MHz
- **Transmitter Power:** ~800mW
- **Transmitter Polarisation**: Vertical (***!!! - Those with Horizontal Yagis watch out!***)
- **Modulation:** DVB-S, QPSK, r=½ FEC
- **Symbol Rate:** 1Msps
- **Video Resolution:** 704x400

This will produce fairly low quality video, but will give the best chance for the payload to be received. Once we understand what the achievable signal-to-noise ratio (SNR) is for a typical flight, the symbol rate (and image quality) can be increased on future launches.

Receiving DVB-S Signals - Hardware

The most ideal way to receive signals from this payload is with a dedicated DVB-S receiver like the Minitiouner Express (~AUD$150 shipped - https://www.datv-express.com/), however it's also very possible to receive it with Software Defined Radio receiver setups, such as the ubiquitous RTLSDR devices. You will also need a 70cm antenna with some gain, and a low-noise preamplifier.
While some DVB-S set-top boxes may tune down to 445 MHz, the very low bitrates that we will be transmitting from the payload will likely be incompatible with them.

Link Budget

To get an idea of what kind of equipment is required to receive signals from the payload, we're going to run through a brief link budget for the entire system. We'll make the following assumptions:

- **Transmitter Power:** 500mW (+27 dBm)
- **Transmitter Frequency:** 445 MHz
- **TX-RX Range:** 100km
- **Transmit Antenna Gain:** 0 dBi
- **Symbol Rate:** 1 Msps (QPSK)
- **Effective Bitrate:** 2Mbit/s
- **Occupied Bandwidth:** ~1.4 MHz (measured)

So, onto our link budget:

Transmit Power:	+27 dBm
TX Antenna Gain:	+0 dBi
TX EIRP:	+27 dBm
Path Loss:	125.4 dB
RX EIRP:	-98.4 dBm

Let's factor in a nominal 70cm receiver setup: a 7-element yagi (~9dBi), and a MiniKits PGA103 preamp (2 dB NF):

RX Antenna Gain:	+9 dBi
Power into RX:	-89.4 dBm
RX System NF:	2dB
RX Noise:	-174 + 10log(1.4 MHz) + 2 = -110.5 dBm

C/N:	21.1 dB
Eb/N0:	21.1 - 10log(1e6 / 1.4e6) = 22.6 dB
Required Eb/N0:	4.5
Link Margin:	~18 dB

So, from this, we can see that even a fairly modest receive setup may be able to receive video from this payload at a distance of 100km! It's worth noting that these calculations assume no impairments from local noise, and no implementation loss (which is unlikely to be the case). Fading caused by the payload swinging will also eat into the link margin.

It's important to note that this payload is very much experimental - it may not perform as well as expected, or it may not work at all!

Suggested Receivers

Software-Defined Radio Receivers

There's a huge range of SDRs available which are fit-for-purpose in this application, provided you put a low-noise-figure preamp in front of them - ideally one with band-pass filtering. Examples include:

- RTLSDR v3 - https://www.rtl-sdr.com/buy-rtl-sdr-dvb-t-dongles/ - ~AUD$40 shipped.
- Airspy Mini - https://www.itead.cc/airspy-mini.html - ~AUD$150 shipped

Note that I've only tested DVB-S reception under Linux using RTLSDRs. The Windows software (SDRAngel) has support for a wider range of SDRs, though I haven't specifically tried reception using anything other than a RTLSDR.

Dedicated DVB-S Receivers

The 'Minitiouner Express' (https://www.datv-express.com/ , ~AUD$150 shipped) is a dedicated DVB-S receiver. It has pretty good receive performance out-of-the-box compared to the RTLSDR (~3dB better) and is more targeted at those users interested in experimenting with DVB-S/S2.
It only works with the 'Minitiouner' software (https://wiki.batc.org.uk/MiniTioune_software), which is Windows only.
I won't be covering how to use this bit of hardware in this guide. If you decide to purchase one of these and need assistance setting it up, please contact me.

Antennas & Preamps

Regardless of what receiver option you go with, you will have the best chance of receiving signals from the payload if you have a yagi antenna with reasonable gain, and a preamplifier.

Preamplifier Options

I highly recommend the preamplifiers from MiniKits. A few suitable preamps include:

- https://www.minikits.com.au/electronic-kits/rf-amplifiers/rf-preamplifiers/PGA-103-UHF-R2
- https://www.minikits.com.au/electronic-kits/rf-amplifiers/rf-preamplifiers/eme221-70cm

To use these you may also need a 12v 'Bias-Tee' unit:
https://www.minikits.com.au/EME195-Bias-Tee

Antennas

A yagi antenna is highly recommended for best results. The transmission will be **vertically polarised** for the majority of the flight (some tumbling after burst).
There are a range of homebrew 70cm yagi designs out there, including:

- Designs based on VK5DJ's yagi calculator: http://www.vk5dj.com/yagi.html
- https://vk1nam.wordpress.com/2018/06/16/antenna-project-portable-70cm-7el-dl6wu-yagi/

There are also comercial yagi options available:

- Cushcraft 11-element - https://www.strictlyham.com.au/browse-by-type/antennas/beam-antennas/cushcraft-a43011s
- Cushcraft 6-element https://www.strictlyham.com.au/browse-by-type/antennas/beam-antennas/cushcraft-a4496s#.X0DKI5MzZ24
- 7-element yagi from element14 - https://au.element14.com/lprs/yagi-434a/antenna-yagi-7-element-434mhz/dp/2096215

If you have other suggestions to add to this list, please let me know!

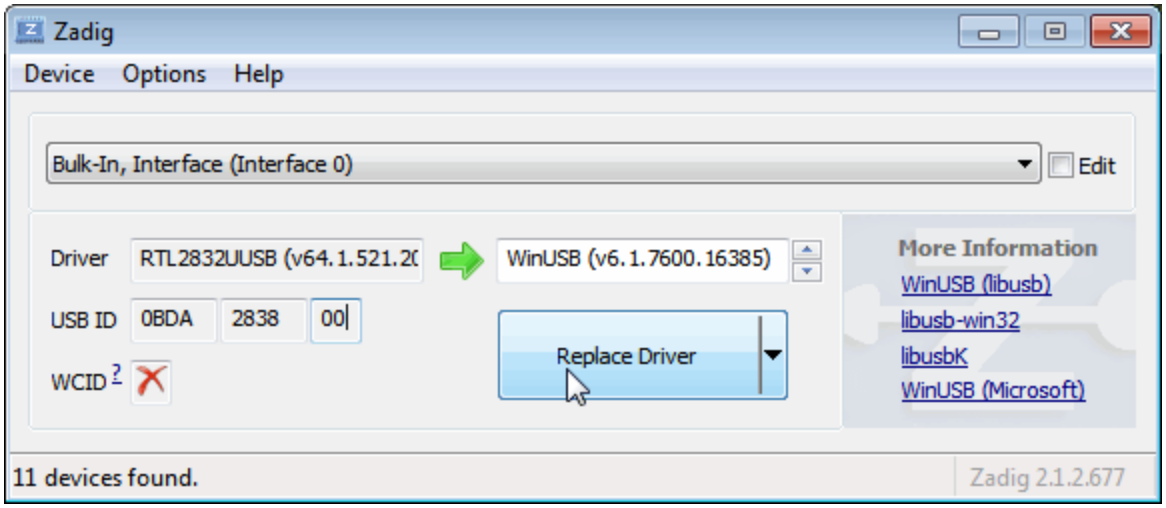
Receiving DVB-S Signals - Windows + SDRAngel

Receiving under Windows using SDRAngel

SDRAngel is a pretty feature-packed SDR application, with all sorts of interesting plugins - including the ability to demodulate DVB-S! It'll work with a range of SDRs, but this guide will focus on reception using a RTLSDR.
Note that SDRAngel's DVB-S demodulator is not amazing, and won't perform as well as a dedicated DVB-S receiver. Unfortunately I haven't found any other alternative software for receiving DVB-S under Windows using a SDR.

Installing the right drivers for your RTLSDR

1. Download the 'Zadig' driver installer from here: https://zadig.akeo.ie/
2. Plug in your RTLSDR. Windows will probably try and install some drivers. Let that complete before continuing.
3. Run the zadig executable as Administrator (right-click and 'Run as Administrator')
4. Go to the Options menu and enable 'List All Devices'
5. **Select** "**Bulk-In, Interface (Interface 0)**" from the drop down list. Note on some PCs you may see something like RTL2832UHIDIR or RTL2832U instead of the bulk in interface. This is also a valid selection. Do not select "USB Receiver (Interface 0) or Interface 1" or *anything else* or you will overwrite that driver! Double check that the USB ID shows "0BDA 2838 00" as this indicates that the dongle is selected.



6. Ensure that WinUSB is selected as the new driver (see above), and click the Replace Driver button.
7. On some PC's you might get a warning that the publisher cannot be verified, but just accept it by clicking on "Install this driver software anyway". This will install the drivers necessary to run the dongle as a software defined radio.

Note that you may need to run zadig.exe again if you move the dongle to another USB port, or want to use two or more dongles together.

Installing SDRAngel

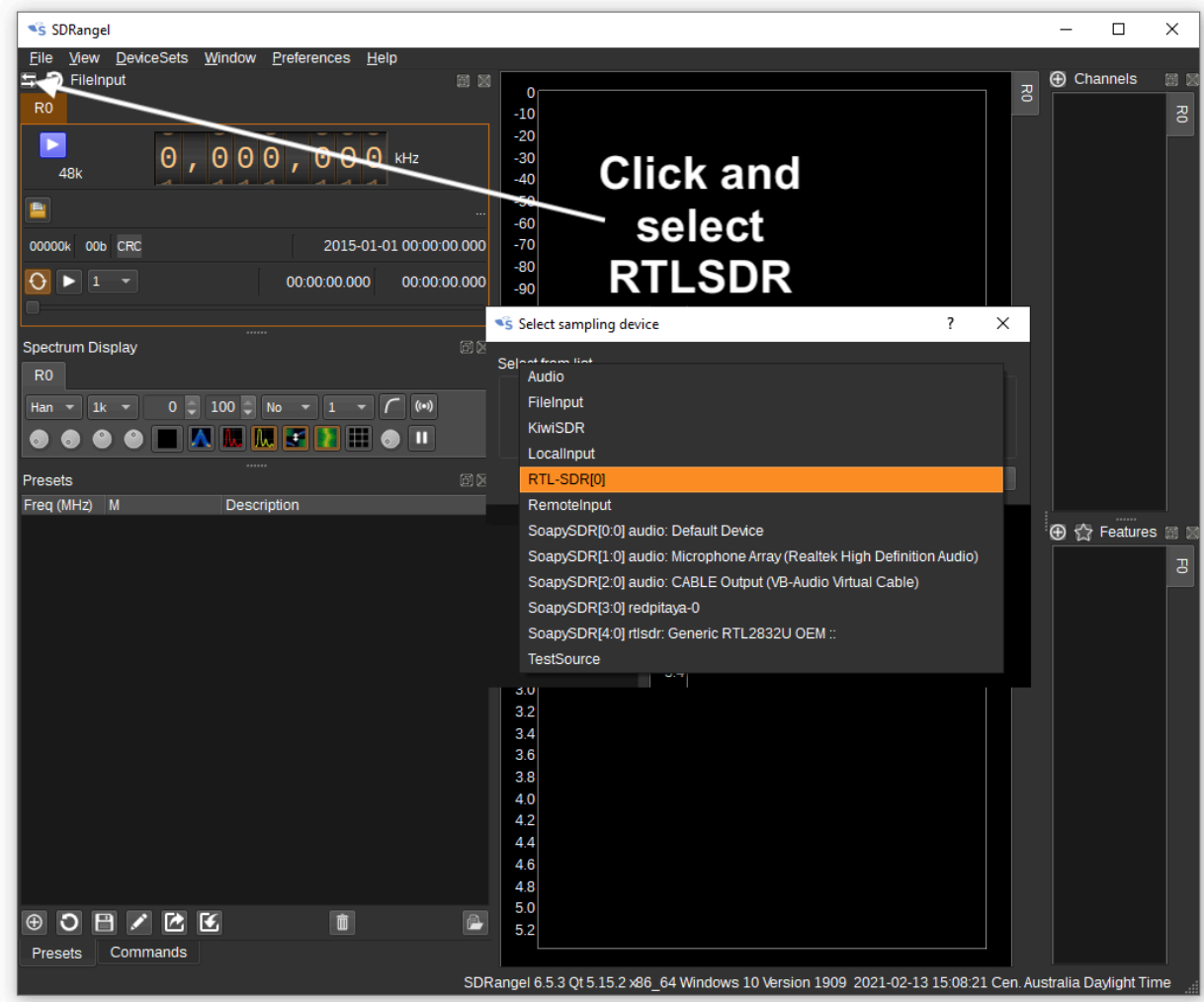
The latest version of SDRAngel for Windows (64-bit only!) can be downloaded from here: https://github.com/f4exb/sdrangel/releases (look for the win64.exe installer)

NOTE: The latest version (v6.6.0) supports displaying of Modulation Error Rate (MER) data.

Download and run the installer. You may also get a Windows Defender notification, just click ‘More Info’ and then ‘Run Anyway’. You should end up with a shortcut in your start menu.

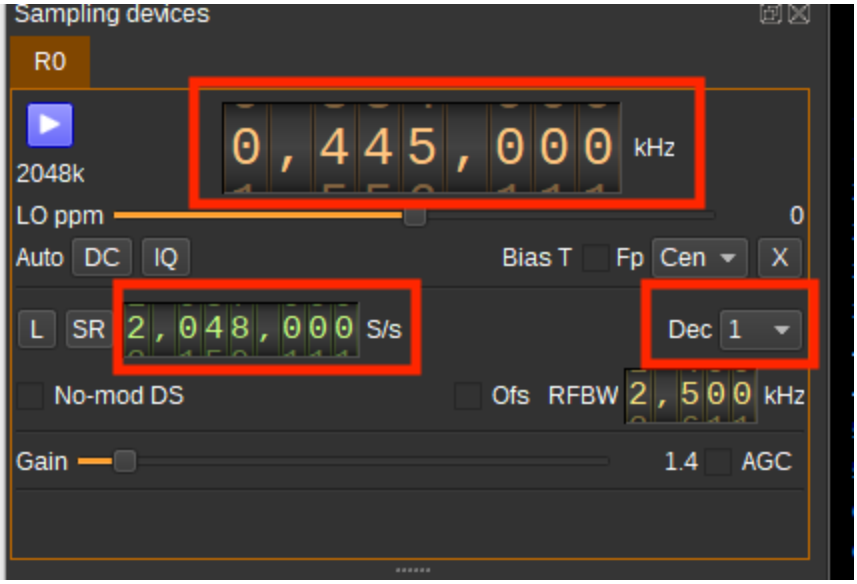
Setting up to Receive DVB-S

1. Open SDRAngel from the start menu shortcut. It can take quite a while to load, and sometimes will even just hang on the splash screen. If this happens, just try loading it again. You may also get a Windows Defender notification, just click ‘More Info’ and then ‘Run Anyway’.
2. When you first startup SDRAngel, you’ll presented with this screen:

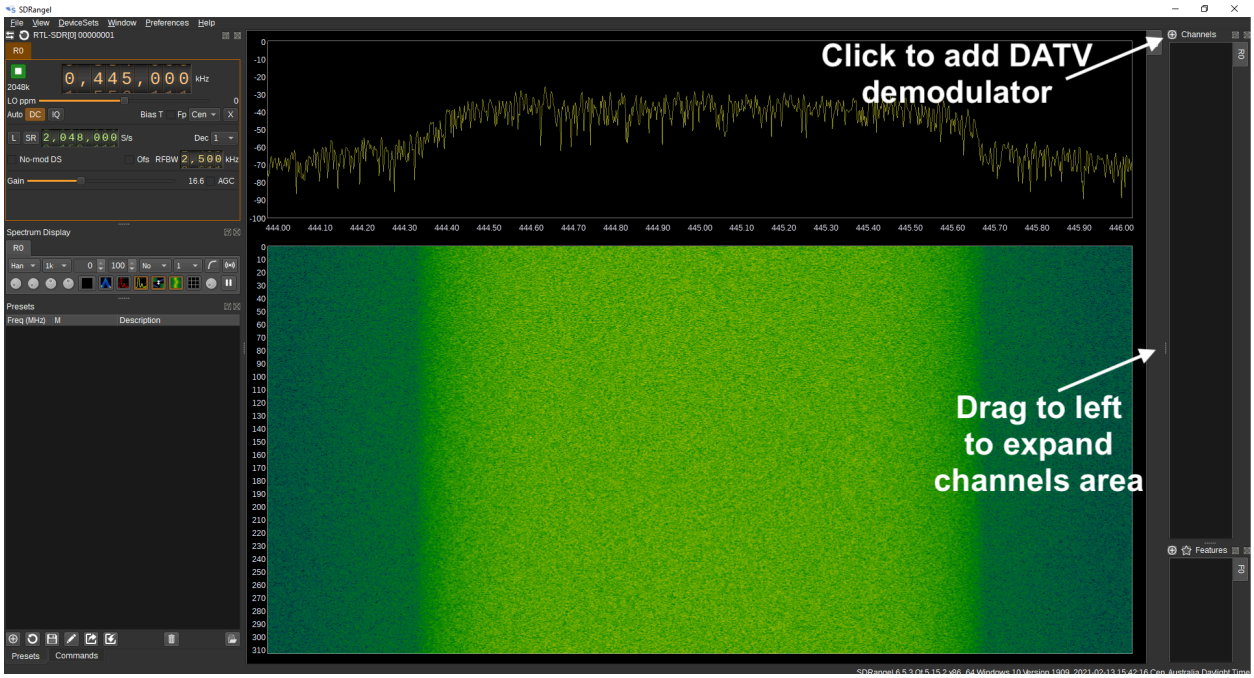


3. Select the indicated icon, and choose the RTLSDR entry from the list. If you don’t see a RTLSDR entry, you might not have installed the drivers correctly.

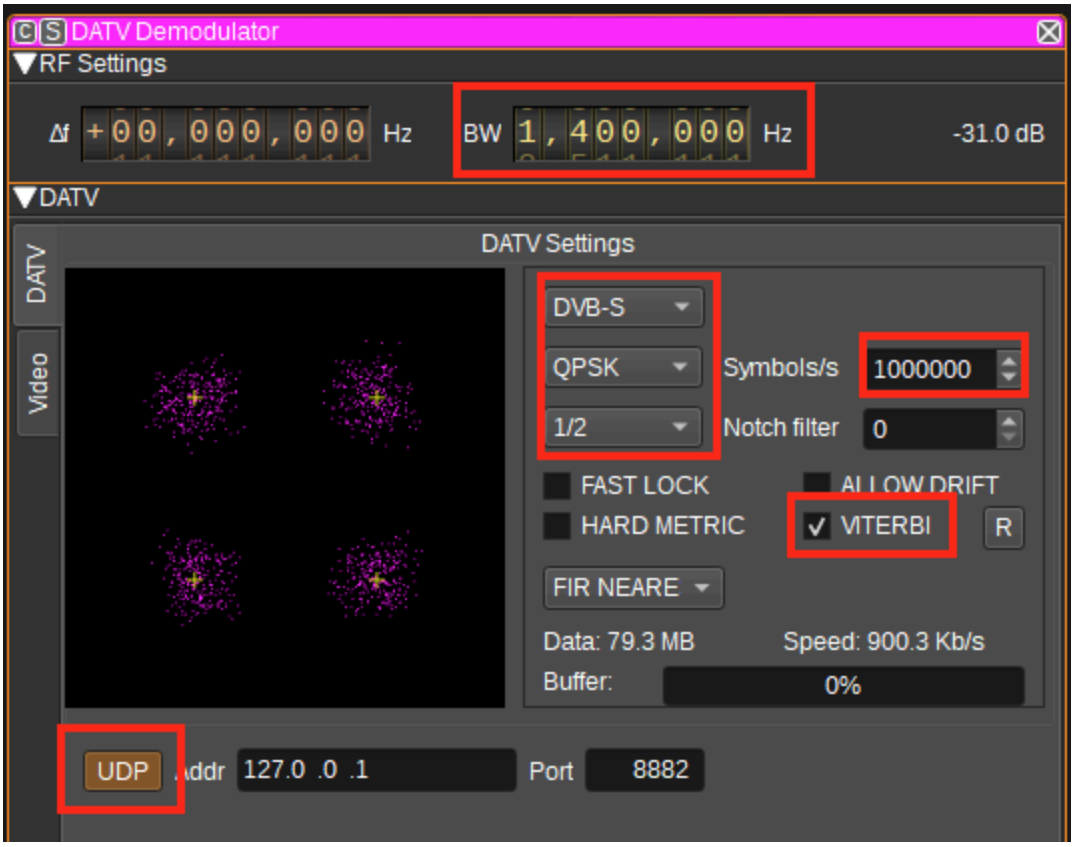
4. Now, we need to set up the RTLSDR Input device.



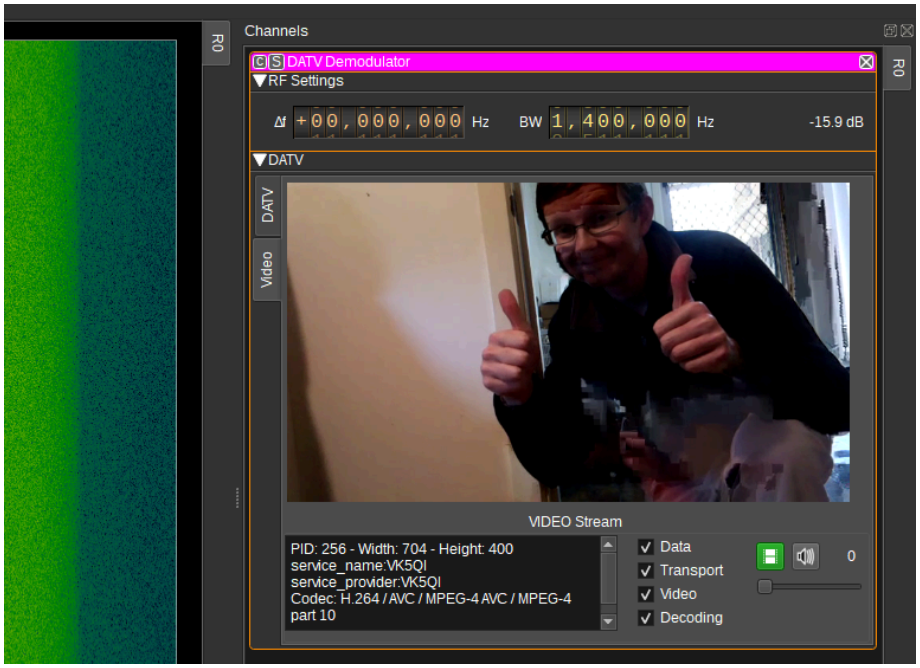
5. Adjust the sections marked in red to match the provided values. **The frequency indications can be set by clicking on the digit you wish to change, and using the up/down arrow keys.** It's a bit of a pain. We want to adjust the following settings:
    a. Frequency: 445 MHz
    b. Sample Rate: 2.048 MHz
    c. Decimation: 1
    d. You can also enable a Bias T output if you need to power a coax-powered preamp.
    e. You will probably want to adjust your gain to somewhere around the middle (30 dB) to start with. If you can't see the gain slider, you may need to drag down on the line of dots at the bottom of the Sampling Devices section to expand it.
6. It's also a good idea to click the Auto 'DC' button, to remove any residual DC offset your SDR may have.
7. You should now be able to click the blue Play button to the left of the main frequency entry. A waterfall display should start:



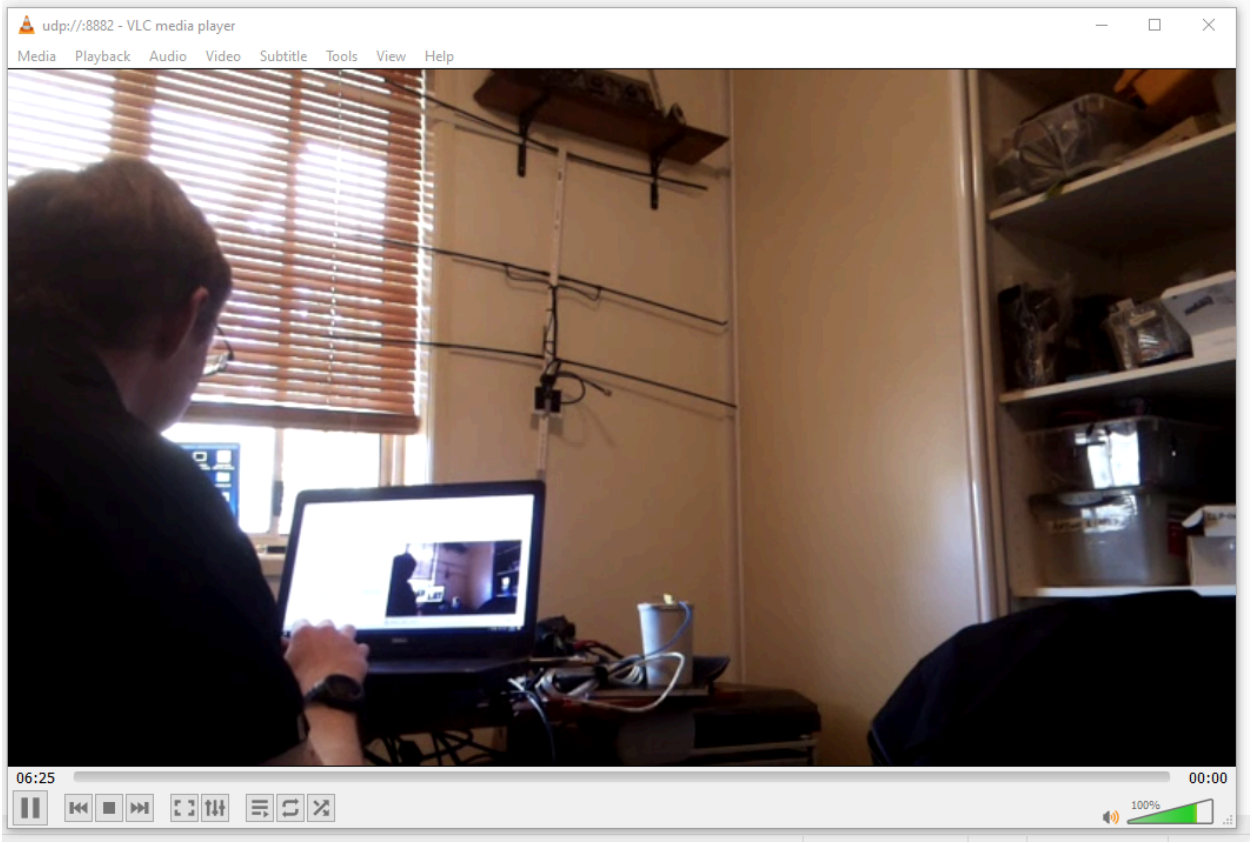
8. If it does *not* start, and SDRangel locks up, then it's probably because you haven't installed the Zadig drivers.
9. Now we need to start a DATV demodulator. Expand the 'Channels' area as indicated in the above image, then click the '+' button next to the 'Channels' label. Select 'DATV Demodulator' from the list, click 'Apply' then 'Close'. A new section will appear on the right of the screen in the 'Channels' area. You can expand this area by dragging the divider between it and the waterfall area.
10. We now need to configure the DATV Demodulator as follows:
    a. Bandwidth: 1.4 MHz (1,400,000 Hz)
    b. Demodulator type: DVB-S, QPSK, ½
    c. Symbols/s: 1000000 (1Msps)
    d. Viterbi: Enabled (***Note: On anything less than a i7 this should probably be turned off, as it uses a lot of CPU - you will suffer a 3-4 dB performance penalty though!***)
    e. Allow Drift: Enable this too (helps if the TX or RX is off-frequency)
    f. UDP: Click to enable (button will turn orange)



11. If you're receiving a signal, you should now be seeing pink dots appearing in the QPSK constellation area. If the pink dots are grouped around the four crosses (the four QPSK symbols), then it means you're receiving a signal! Sometimes you need to adjust the frequency offset settings (marked **Δ**f) to get a good lock. With any luck, the Data indication will now start increasing indicating you are receiving video.
12. You should now be able to click the 'Video' tab (just to the left of the constellation diagram), and view the video stream:



13. The video window in SDRangel is a little bit small, but we can get a bigger view by making use of the UDP video streaming output we enabled earlier. To do this, you will need to install the VideoLAN Player (VLC) available here: https://www.videolan.org/vlc/index.html
14. Open up VLC, then go to Media -> Open Network Stream
15. In the network URL box, enter: udp://@:8882 and click ‘Play’. A firewall box may appear, and you may need to enable VLC access to the local network.
16. If you are receiving signals, you should now have the transport stream playing in VLC:



That's it!

Receiving DVB-S Signals - Linux + LeanSDR

If you're running a Linux platform, we can receive the DVB-S signal using a RTLSDR and the LeanSDR software stack. This guide assumes you are running Ubuntu 20.04. It will probably work on other Ubuntu/Debian-based distributions, but has only been tested on Ubuntu 20.04.

Installing Dependencies

You'll need to install a few dependencies to be able to compile and run the decoder. Open a terminal and run (note that the $ indicates the command is to be entered into a terminal, so should not be typed in):

```
$ sudo apt-get install build-essential rtl-sdr git libfftw3-dev libiio-dev libx11-dev mplayer
```

You can now download and compile the leansdr utility by running:

```
$ git clone http://github.com/projecthorus/leansdr.git
$ cd leansdr/src/apps
$ make
```

There may be some warnings about libiio not being found, these can be ignored.

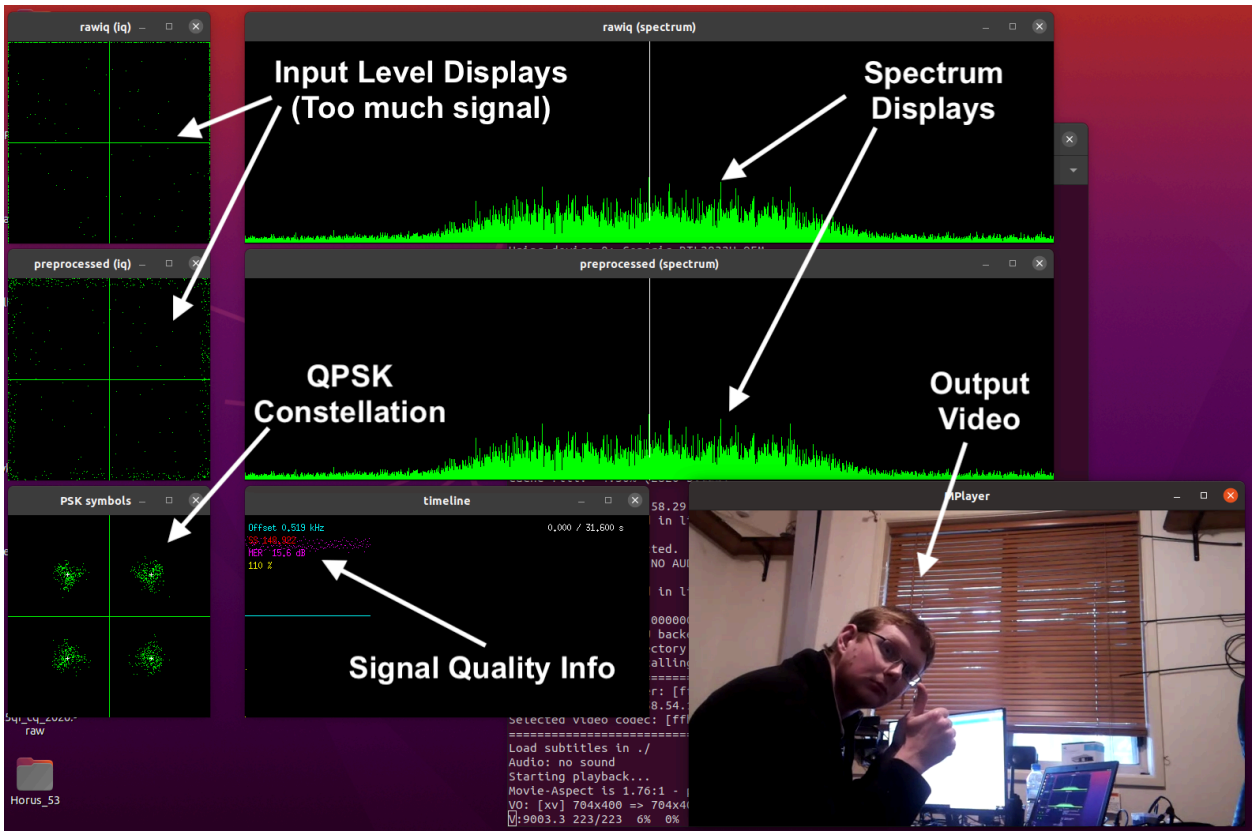
Running the Decoder

Open a terminal and navigate to the `leansdr/src/apps` directory that we were in before. Start the decoder by running:

```
$ ./start_horus.sh
```

If you are running on a fast machine (i5 or better), I would recommend running:
`$ ./start_horus_viterbi.sh`

This enables Viterbi forward-error correction, which will improve demodulator performance, but does us a *lot* more CPU (95% of a core on my i5 laptop). It should give something like 3-4 dB decode improvement, so use it if you can! Watch out that
If all is working, a bunch of windows will immediately pop up, as follows:



The image above has annotations showing what each window represents. In this example we are receiving a reasonably clean (not perfect) signal, as evidenced by the constellation diagram showing four fairly clean regions.
The spectrum displays do not auto-scale like many other receiving software does, and so you may not see much in them if the signal is weak! The key window to look at to judge signal quality is the constellation diagram window - if you are seeing four distinct ‘blobs’ then you are likely receiving a signal!

Note that the video window (mplayer) will only show up once you are receiving a valid signal, and it may take a few seconds to appear even then.

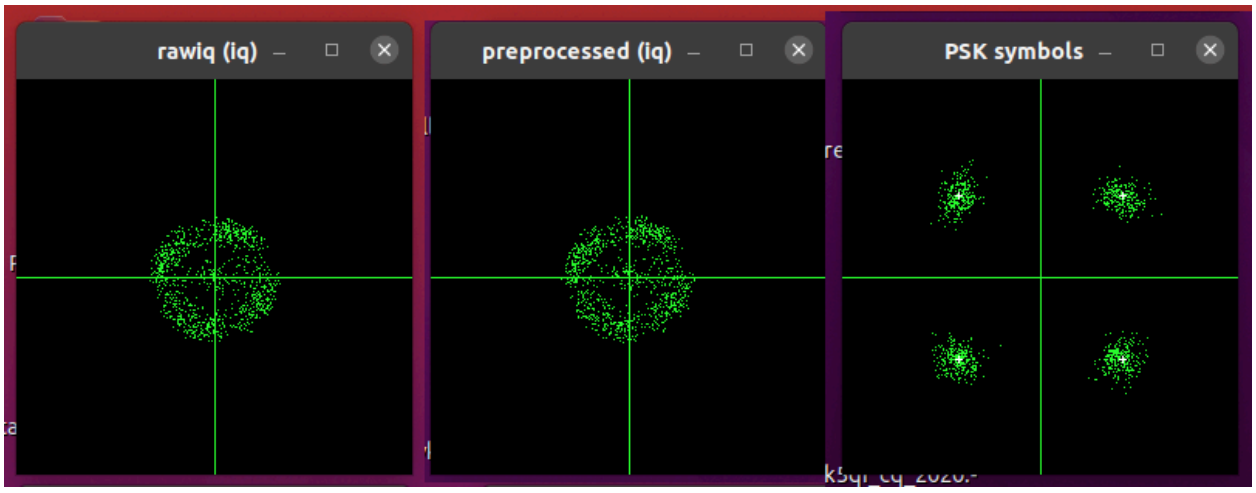
You can stop the decoder by hitting Ctrl+C in the terminal window. You might need to press Ctrl+C a few times for it to completely exit.

In this case the signal level is actually a little bit too high, and if we reduce the gain slightly we can improve the decoder performance. To modify the gain settings (and other settings), you need to edit that start_horus.sh file we ran earlier. We can edit this in a terminal by running:

```
$ nano start_horus.sh
```

(or nano start_horus_viterbi.sh if you are using that version)

Look for a line containing `GAIN=0` . The 0 value indicates the use of the RTLSDR’s automatic gain control (AGC), which should be sufficient in most cases.
If necessary, the gain can be adjusted by changing this value to something between 1 and 49. Save the file by hitting Ctrl+X, then Y. Restart the decoder by running `./start_horus.sh`



The above image shows the input level displays and constellation diagrams with a lower gain setting. The constellation regions now look a little cleaner.