leandvb: A lightweight software DVB-S demodulator

leandvb is a lightweight implementation of portions of the DVB-S standard in plain C++. It is developed primarily for receiving Digital Amateur TV, including HamTV from the International Space Station.

READ THE HYPERTEXT VERSION HERE:
http://www.pabr.org/radio/leandvb/leandvb.en.html

Revision History		
1.2	2016-09-03	Updated for github. More code rates. Demodulator status output. leansdrscan. leansdrcat.
1.1	2016-03-16	Troubleshooting hints. Notes on live MPEG decoding.
1.0	2016-02-19	Initial release.

Table of Contents

1. Motivations

DVB-S was developed in the 1990s for digital satellite TV broadcasts. In recent years it has been adopted by the amateur radio community for Digital Amateur TV (DATV). Inexpensive hardware-accelerated DVB-S receivers are available in the form of set-top-boxes, PCI cards and USB dongles. Unfortunately not all receivers are suitable for DATV for the following reasons:

- Their RF stage is typically optimized for an input signal in the 950 .. 2150 MHz range (downconverted from 10.700 .. 12.750 GHz by a satellite LNB), whereas DATV experiments are also conducted in the 70 cm and 13 cm bands (possibly even 2 m and 6 m). Frequency converters add unnecessary cost, complexity and noise.
- They are designed to provide a plug-and-play experience for non-technical users, whereas radio amateurs may prefer to have control over as many settings as possible in order to make the most of propagation conditions.
- Some receivers do not support the very low symbol rates that amateurs are required to use due to bandwidth constraints.
- Some prominent DATV transmitters, such as the ISS HamVideo system, generate a slightly non-compliant MPEG stream which some receivers fail to decode.

With custom drivers it is apparently possible to tune out-of-spec, support non standard symbol rates, control internal settings and retrieve raw MPEG packets: see http://www.vivadatv.org. But this approach only works with specific chipsets

Software-defined receivers avoid these problems entirely. SDR front-ends are now available for many frequency bands with various trade-offs between price, bandwidth and dynamic range. The popular RTL-SDR dongles cover 10 m, 6 m, 2 m, 70 cm and 23 cm.

A software DVB-S receiver also brings all the benefits of software-defined radio:

- One can capture raw I/Q samples from the air and demodulate them later. There are fewer opportunities for non-recoverable operator error during the recording phase. This is especially useful for transient signals such as those from the ISS.
- Demodulation can be attempted as many times as necessary with various combinations of parameters until the final error rate is minimized.
- Waveforms can be examined and pre-processed prior to demodulation, e.g. to eliminate spurious signals or correct unexpected frequency offsets. Computationally intensive algorithms can be used in an attempt to remove non-random noise.
- Coherent I/Q captures from multiple receivers could be combined to achieve array gain, diversity gain or better directivity.
- Perfectly identical signals can be injected into multiple software demodulators for benchmarking purposes.
- A SDR receiver doubles as a crude spectrum analyzer. This is useful for troubleshooting many problems, e.g. detecting in-band interference.
- Software demodulators are future-proof because all SDR front-ends have essentially the same interface: configure center frequency, sample rate and gain; receive raw I/Q data. Actually, as computer performance increases, a given software demodulator will become capable of supporting faster modulation rates or processing larger numbers of channels simultaneously.

2. Related work

gr-dvb [GRDVB] is a complete implementation of DVB-S for gnuradio. I used it extensively during earlier DATV experiments (see [SOFTDATV]).

gr-dvb has great value as a reference implementation and as an educational tool, but it is too slow to demodulate DATV signals from the ISS in real-time, even on a fast desktop PC. Besides, it requires the whole gnuradio infrastructure including Boost, SWIG and Python; this is impractical on embedded platforms.

By contrast, **leandvb** is written in plain C++ with barely any dependency, and is 10 times faster on a single core than **gr-dvb** running on two cores. The speed-up comes at the cost of receiver sensitivity; for some applications this is an acceptable trade-off. More specifically, **leandvb** does not currently implement the Viterbi algorithm. It performs deconvolution on hard symbols instead.

3. Features

- DVB-S, QPSK, code rates 1/2, 2/3, 3/4, 5/6, 7/8.
- Detection of conjugated I/Q signals (for downconverters with L.O. above the signal).
- Tested at low sampling rate (2 Msymbols/s at 2.4 Msamples/s).
- Automatic filtering of "birdies" (useful with low-cost SDR hardware).
- Default input format is unsigned 8-bit I/Q (compatible with the **rtl_sdr** command-line utility).
- Also supports 32-bit float I/Q format (compatible with **gqrx**).

4. Limitations and known problems

- No band-pass filtering. In crowded bands, consider reducing the sampling rate so that only the signal of interest is captured.
- Demodulation only.
- Presumably **leandvb** does not meet the error correction requirements of the DVB-S standard. Therefore it cannot formally claim DVB-S compliance.

5. Installation

leandvb requires only a C++ compiler and standard libraries. It is developed for Linux and may compile on other UNIX-like and embedded platforms as well.

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

6. Usage

```
Usage: leandvb [options]  < IQ  > TS
Demodulate DVB-S I/Q on stdin, output MPEG packets on stdout

Input options:
  --u8          Input format is 8-bit unsigned (rtl_sdr, default)
  --f32         Input format is 32-bit float (gqrx)
  -f HZ         Input sample rate (default: 2.4e6)
  --loop        Repeat (stdin must be a file)

Preprocessing options:
  --anf N        Number of birdies to remove (default: 1)
  --derotate HZ  For use with --fd-pp, otherwise use --tune
  --fd-pp NUM    Dump preprocessed IQ data to file descriptor

DVB-S options:
  --sr HZ        Symbol rate (default: 2e6)
  --tune HZ      Bias frequency for demodulation
  --cr N/D       Code rate 1/2 .. 7/8 (default: 1/2)

UI options:
  -h             Print this message
  -v             Output info during startup
  -d             Output debugging info
  --fd-info NUM  Print demodulator status to file descriptor
  --gui          Show constellation and spectrum
  --duration S   Width of timeline plot (default: 60)
  --linger       Keep GUI running after EOF

Testing options:
  --awgn STDDEV  Add white gaussian noise (slow)
```

Note: The **--gui** option is only available on platforms with X11.

6.1. Off-line demodulation

For HamTV signals at 2395 MHz downconverted to 397 MHz by a MMDS LNB with 1998 MHz LO:

```
$ rtl_sdr  -f 397000000  -s 2400000  capture.iq
$ ./leandvb  --gui  < capture.iq  > capture.ts
$ ffplay capture.ts
```

6.2. Real-time demodulation

Conceptually, a live receiver and decoder would be started as follows:

```
$ rtl_sdr  -f 397000000  -s 2400000  -  |  ./leandvb  --gui  |  vlc -
```

However, this ignores known problems with real-time multimedia streaming:

- Some MPEG players insist on scanning their input stream prior to decoding. The resulting latency can sometimes be reduced by specifying formats and codecs manually on the command-line.
- Most MPEG players will pace video playback by using the playback rate of the sound card as a time base, regardless of the actual input data rate. This will unavoidably cause buffer under-runs (leading to audio artifacts) or overruns (causing **leandvb** to stall due to a blocked output pipe and lose synchronisation). This can sometimes be mitigated by manually specifying a frame rate slightly higher that the real value.

For interactive applications such as videoconferencing, sources of latency must be understood as well.

- MPEG has provisions for transmitting frames out-of-order, which implies latency. Encoders should be configured to not use backward prediction, i.e. B-frames.
- DVB-S interleaving causes an unavoidable latency of 2244 bytes, i.e. 300 ms at 64 Kbit/s.
- **leandvb** tends to process data in fixed-size chunks (but the resulting latency should be less than that caused by interleaving).

- Most MPEG decoders will buffer data to ensure a smooth playback.

This script was successfully used to display live video from the ISS and simultaneously record I/Q samples:

```
$ while true; do
   rtl_sdr  -g 22  -f 397000000  -s 2400000  -  |  tee $(date +%Y%m%d_%H%M%S%z.iq)  |
  done
```

Explicit caching of the transport stream may also help the decoder identify the MPEG stream, e.g.:

```
$ ...  |  mplayer -cache 128 -
```

leansdrcat can be used to play back a file at a constant bit rate. This is useful for testing whether a MPEG decoder can handle real-time streams.

```
$ ./leansdrcat  --cbr $((2400000*16))  < capture.iq  |  ./leandvb  --gui  |  mplayer -
```

6.3. Automatic detection of modulation settings

The DVB-S symbol rate and code rate must be supplied to **leandvb** on the command-line. A companion tool, **leansdrscan**, can help detect these parameters automatically.

The following command will spawn **leandvb** with six combinations of symbol rates (250 kS/s, 500 kS/s, 1000 kS/s) and code rates (1/2, 7/8) until MPEG packets are obtained.

```
$ ./leansdrscan  -v  --rewind  ./leandvb  -f 1024000  --sr 250000,500000  --cr 1/2,7/8
```

For live streams the **--rewind** option is not needed. Besides, the **--tune** option could be used to scan multiple narrow-band channels within a wide-band capture:

```
$ rtl_sdr ...  |  ./leansdrscan  -v  ./leandvb  -f 2400e3  --tune -500e3,0,500e3  --sr
```

Note: This may not work on crowded bands because **leandvb** does not perform bandpass filtering yet.

6.4. Integration with other software

leandvb can provide status information on an auxiliary output channel specified by **--fd-info**.

```
$ rtl_sdr ...  |  ./leansdrscan  ./leandvb --fd-info 2 -f 1024e3 --sr 250e3,500e3 --cr
CR 1/2
SR 250000.000000
LOCK 0
FREQ 4047
SS 66.660774
MER 19.8
LOCK 1
FREQ 4062
SS 65.118179
MER 18.7
FREQ 4047
...
```

This should allow **leandvb** to be used as a plugin for other applications. For example, **leandvb_tui.sh** displays the status information on one line:

```
$ rtl_sdr ...  |  ./leandvb_tui.sh  ./leansdrscan  ./leandvb  --fd-info 2  -f 1024e3  -
[SS  68] [Offset   +4047 Hz] [MER 18.7 dB] [LOCKED] [SR  250000 Hz] [FEC 1/2]
```

7. Performance

7.1. Throughput

CPU usage depends on the input sample rate, on the symbol rate and to a lesser extent on the error rate. Note that the **--gui** option increases CPU load.

leandvb can demodulate 2 Msymbols/s digitized at 2.4 Msamples/s in real-time on a 1.33 GHz Intel Z3745 tablet. This makes it possible to receive HamTV from the ISS with a low-cost portable setup, which was the original goal of the project.

leandvb is also known to work on Raspberry Pi platforms, but at slower symbols rates.

Table 1. Benchmark between platforms

Dataset	**samprate/symbrate/dur**	**Intel Core i7**	**Intel Atom Z3745**	**Raspberry Pi 2 Model B**
gqrx_20151211_204746	2.4M / 2M / 23	2588 KiB in 4 s	2588 KiB in 13 s	2639 KiB in 50 s

7.2. Latency

TBD

7.3. Sensitivity

leandvb can demodulate a HamTV signal captured during an overhead pass of the ISS with a linearly-polarized 24 dBi dish, MMDS downconverter and RTL-SDR R820T2 dongle.

leandvb successfully received the ARISS HamTV contact on 2016-05-09 at 67° maximum elevation with a shoulder-mounted dish (see [HAMPADS]).

Demodulation was extremely poor during another pass at 47° elevation with wet weather.

Table 2. Benchmark between demodulators

Dataset	**samprate/symbrate/dur**	**leandvb**	**gr-dvb without Viterbi**	**gr-dvb**
gqrx_20151211_204746	2.4M / 2M / 23	2588 KiB in 4 s	2579 KiB in 8 s (17 s total)	2648 KiB in 37 s (58 s total)

8. Notes on security

Any software component that processes third-party data is a security risk. This rule applies to web servers and software-defined radio receivers alike. In theory, if software-defined reception of sophisticated digital modes becomes popular, a malicious transmission on a HF band could target software vulnerabilities in a large number of receivers worldwide.

As a mitigation strategy, it is a good idea to run **leandvb** with minimal privilege, under SELinux, in a virtual machine, on dedicated hardware, etc.

9. Troubleshooting

This section lists symptoms, diagnostics and workarounds for common problems.

9.1. Configuration

9.2. Gain

9.3. Signal

10. Implementation details

Figure 8, “ DVB-S demodulation flow-graph ” illustrates the demodulation process.

The auto notch filter block performs a Fourier transform about once per second to identify spurious signals. Then it tracks their phase and amplitude in order to filter them.

Subsequent blocks follow the DVB-S standard closely. See [DVBS] for details.

The QPSK demodulator performs timing recovery and interpolation. It outputs soft symbols because DVB-S encourages Viterbi decoding, but the current implementation thresholds them immediately into 2-bit QPSK symbols.

The deconvolution block reverses the inner (convolutional) coding and puncturing, but the current implementation does not correct errors.

The synchronizer handles conjugation, constellation rotation (0, 90, 180 or 270°), alignment of puncturing patterns and alignment of byte boundaries. Synchronization is achieved when the stream contains MPEG start codes every 204 bytes. One in eight start codes is inverted.

The deinterleaver reverses the byte shuffling that is performed during modulation. (This is done to spread burst noise over a large number of packets so that Reed-Solomon can be more effective.) By design, start codes are not shuffled.

The Reed-Solomon block uses the last 16 bytes of each 204-byte packet to detect and correct errors.

The derandomizer reverses the bitwise obfuscation that is performed during modulation. (This is done for "energy dispersal", i.e. to flatten the RF spectrum.) It also de-inverts the inverted start codes.

11. Development

See https://github.com/pabr/leansdr.

12. Future work

- Better interpolation
- Sensitivity benchmarks with simulated noise.
- Multi-threading
- SIMD optimizations on x86 and ARM
- Viterbi. Dynamic demodulation trade-offs for optimal performance on all platforms
- Allow **leansdrscan** to run several demodulator in parallel.

- DVB-S2
- Modulator (for testing purposes; there are already several software-defined DATV modulators)
- GQRX plugin
- Android app.

13. Intellectual property issues

leandvb is developed primarily for non-commercial research and experimentation in the context of amateur radio. Irrespective of opinions about the legitimacy of software patents, anyone contemplating other uses should probably seek legal advice about the following issues.

The DVB-S standard was originally published in 1994 as ETS 300 421. There is hope that any applicable patents have expired or will expire shortly. However, even after a standard has become patent-free, some implementation details could still be subject to IP rights.

A complete software DVB-S receiver would typically include a MPEG-2 decoder. The MPEG-2 standard was finalized around 1995 and published in 1996. This article [http://www.osnews.com/story/24954/US_Patent_Expiration_for_MP3_MPEG-2_H_264/] estimates that it will become patent-free in 2018.

Related standards such as DVB-T (1997), H.264 (2003), DVB-S2 (2005), DVB-T2 (2008) and HEVC (2013) are probably covered by patents.

14. Acknowledgements

Edmund Tse published a reference implementation of DVB-S for gnuradio as early as 2010 ([GRDVB]). This provided a proof-of-concept and initial motivation for this project.

Early users F5OEO and K4KDR contributed I/Q recordings from a variety of modulators, transmission channels and receivers. This helped debug and fine-tune **leandvb**.

Bibliography

[DVBS] *Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for 11/12 GHz satellite services* . 1997. ETSI. https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=5316.

[GRDVB] *Software Radio for Digital Satellite Television* . 2010. Edmund Tse. http://www.edmundtse.com/wp-content/uploads/2009/04/treatise.pdf.

[SOFTDATV] *SDR reception of Digital Amateur TV from the ISS* . http://www.pabr.org/radio/softdatv/softdatv.en.html .

[HAMPADS] *HAMPADS: HAM-Portable Affordable Dish for Satellites* . http://www.pabr.org/radio/hampads/hampads.en.html .