gr-satellites project update

Dr. Daniel Estévez

7 February 2021
FOSDEM

gr-satellites

gr-satellites is a GNU Radio out-of-tree module with a collection of telemetry decoders for Amateur satellites. It supports most satellites that transmit on the 145 and 435 MHz Amateur bands: 29 different protocols, and 177 satellites

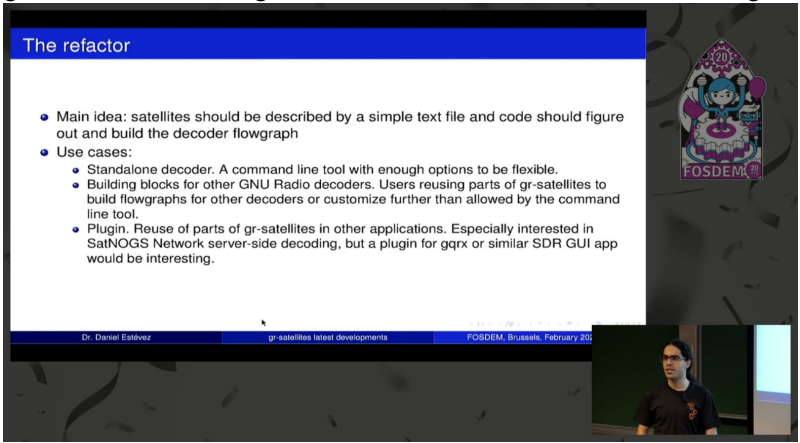
The project started around the end of 2015. The main goal of gr-satellites is to provide a solution so that anyone can decode any satellite that transmits on Amateur radio spectrum.

An additional project goal is education: as a learning resource and as support for novel satellite teams (example: university projects)

Last year at FOSDEM 2020

Lots of fun and waffles with the people from the Free Software Radio devrooom!

gr-satellites talk: large code refactor and future release of gr-satellites v3.0.0



gr-satellites throughout 2020

More time to work on projects due to the pandemic. But most of that time has gone into projects other than gr-satellites.

gr-satellites v3.0.0 released in June after 8 months of refactor work (on and off!)

Several releases during the year: v3.6.0 in December. These usually add support for recently launched satellites and/or include improvements suggested by users.

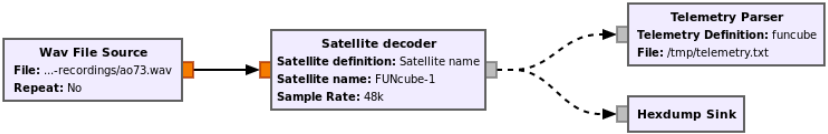
Highlights of gr-satellites v3

1. gr-satellites allows working at several levels of detail:
   - `gr_satellites` command line tool
   - Satellite decoder GNU Radio companion block
   - Component blocks
   - Low level blocks

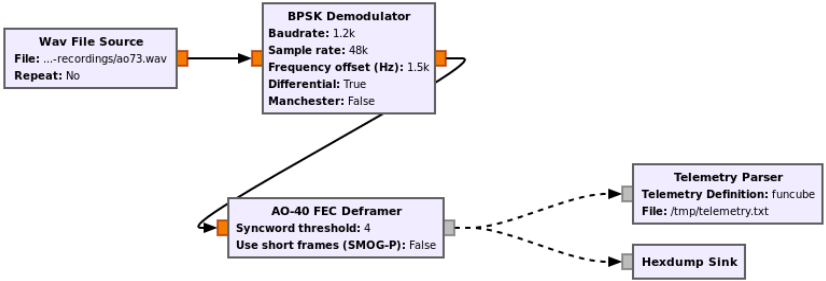
gr_satellites command line tool

```
$ gr_satellites FUNcube-1 --wavfile satellite-recordings/ao73.wav \
      --samp_rate 48e3
-> Packet from 1k2 BPSK downlink
Frame type WO10
----------------------------------------
Realtime telemetry:
----------------------------------------
Container:
    eps = Container:
        photovoltage = ListContainer:
            0
            0
            0
        photocurrent = 0
        batteryvoltage = 8140
...
```

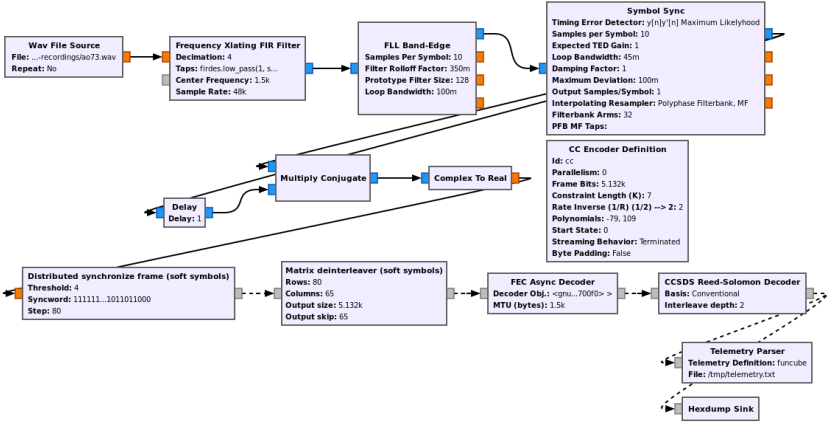
Satellite decoder block



Component blocks

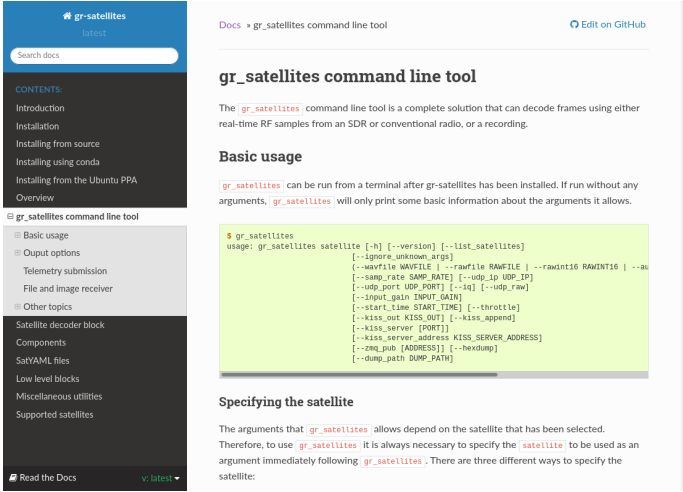


Low level blocks



Highlights of gr-satellites v3

2. User documentation; comprehensive, written in Sphinx, hosted on readthedocs.io



gr-satellites
latest

Search docs

CONTENTS:

Read the Docs v: latest

Docs » gr_satellites command line tool

Edit on GitHub

gr_satellites command line tool

The `gr_satellites` command line tool is a complete solution that can decode frames using either real-time RF samples from an SDR or conventional radio, or a recording.

Basic usage

`gr_satellites` can be run from a terminal after gr-satellites has been installed. If run without any arguments, `gr_satellites` will only print some basic information about the arguments it allows.

```
$ gr_satellites
usage: gr_satellites satellite [-h] [--version] [--list_satellites]
                               [--ignore_unknown_args]
                               {--wavfile WAVFILE | --rawfile RAWFILE | --rawint16 RAWINT16 | --au
                               [--samp_rate SAMP_RATE] [--udp_ip UDP_IP]
                               [--udp_port UDP_PORT] [--iq] [--udp_raw]
                               [--input_gain INPUT_GAIN]
                               [--start_time START_TIME] [--throttle]
                               [--kiss_out KISS_OUT] [--kiss_append]
                               [--kiss_server [PORT]]
                               [--kiss_server_address KISS_SERVER_ADDRESS]
                               [--zmq_pub [ADDRESS]] [--hexdump]
                               [--dump_path DUMP_PATH]
```

Specifying the satellite

The arguments that `gr_satellites` allows depend on the satellite that has been selected. Therefore, to use `gr_satellites` it is always necessary to specify the `satellite` to be used as an argument immediately following `gr_satellites`. There are three different ways to specify the satellite:

Highlights of gr-satellites v3

3. SatYAML: a YAML-based format to describe satellite protocols and metadata

```yaml
name: AO-73
alternative_names:
  - FUNcube-1
norad: 39444
telemetry_servers:
  - FUNcube
data:
  &tlm Telemetry:
    telemetry: funcube
transmitters:
  1k2 BPSK downlink:
    frequency: 145.935e+6
    modulation: DBPSK
    baudrate: 1200
    framing: AO-40 FEC
    data:
    - *tlm
```

SatYAML and SatNOGS Transmission Schema

SatYAML allows gr-satellites to construct the decoders required for each satellite. It solves this problem well, but it doesn't strive to be very general or backwards-compatible.

It solves a problem similar to SatNOGS' future Transmission Schema (satnogs-db issue #317). SatYAML gives useful lessons learned about possible difficulties (for example: too many ad-hoc protocols used by cubesats!)

Currently the metadata in SatNOGS DB is very poor regarding protocols (I wasn't able to pull out the list of satellites that use AX.25!). The Transmission Schema strives to solve this, but it's been in development for 1.5 years.

If the future Transmission Schema provides all the information that gr-satellites needs, then SatYAML may not be needed in the future.

Multiplatform support

- Conda package (Linux, OS X, Windows) since August, thanks to Ryan Volz and Petrus Hyvönen
- Debian package since October, thanks to Maitland Bottoms
- Ubuntu PPA since December
- MacPorts, thanks to Michael Dickens and ra1nb0w
- Other Linux distributions: Arch AUR

Continuous integration

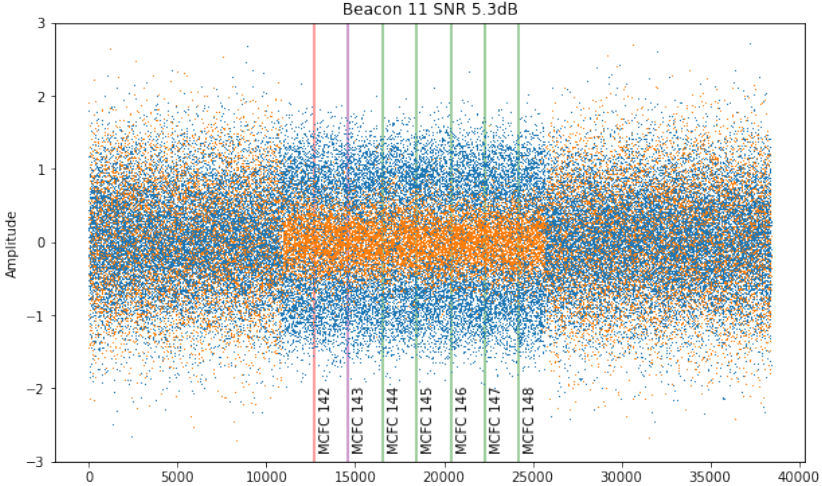
Since May, continuous integration with Buildbot for the code and docs, and Github actions for `clang-format`

Based on the GNU Radio build server. Huge thanks to all the team for making their Buildbot configurations available.

Not very many unit tests, but slowly increasing

Detailed internal state output

Symbols and other variables of the decoder are dumped to files. Can be used to benchmark and optimize decoders, for debugging, or for education.



Around gr-satellites

- Roza Chatzigeorgiou's B.Sc. thesis (Technical Univ. Crete), "Experimental study of satellite signal processing with software defined radios". A study about how to improve LilacSat-2's demodulator and decoder.
- "ESA OPS-SAT Mission: Powered by GNU Radio" talk by Tom Mladenov at GRCon20.
- Decoding deep-space probes: my workshop at GRCon20, lots of activity and talks regarding Tianwen-1, and recent activity with Chang'e 5.

More community engagement

Leading a small open-source project can sometimes feel like releasing your code out in the void

Previously, I handled many gr-satellites questions and support by email

Now I'm encouraging this discussion to happen in public, in Github issues. This is an idea I took from Kate Temkin.

This has caused an increase in community engagement

In the issues we have good technical discussion between users, new features been suggested, bugs detected and solved, troubleshooting...

150 issues opened in 2020, versus 65 in 2016-2019

An example of a good issue: #196

Jan van Gils asked why a particular SatNOGS recording gave no decodes, and how to optimize decoder parameters for this recording. I replied expanding on what the docs say about plotting symbols.

Closed **JY1-Sat isn't being decoded** #196

janvgils opened this issue on 17 Nov 2020 · 10 comments

As an example of usage, I start `ipython` and then run `gr_satellites` like:

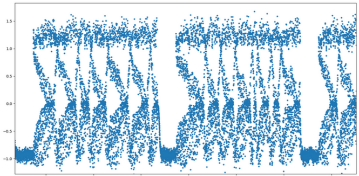
```
In [1]: !gr_satellites JY1-Sat --wavfile satnogs_3154621_2020-11-17T11-39-53.wav --samp_rate 48e3 --f_offset 11.
```

(the `!` is used to run shell commands from `ipython` rather than Python code). I'm using `--f_offset 11.5e3` because the signal is really at 11.5kHz rather than 12kHz, as one can check with Audacity.

This produces several files in the current folder with the internal signals of the demodulator. In this case of a DBPSK signal, the most interesting is the clock recovery output, but since this is a differential signal and there is no carrier phase recovery, using differential decoding before plotting is helpful. I do

```
In [2]:  x = np.fromfile('clock_recovery_out.c64', dtype='complex64'); plt.plot((x[1:]*np.conjugate(x[:-1])).rea
```

to read the file, and plot the differentially-decoded symbols. I zoom in on the graph and see something like this:



→ Transfer issue

Delete issue

Collaboration with satellite teams

Satellite teams that use gr-satellites as a base for their decoder typically would fork off with no plans to merge back to upstream

This is not the best. It leads to unmaintained code, and often to re-inventing the wheel.

With gr-satellites v3, I've written a "letter to satellite teams", inviting them to collaborate early on with the goal of getting their requirements into upstream

Writing a "presentation" as a Github issue explaining what protocols the satellite will use is encouraged. This serves to determine what extra functionality is needed in gr-satellites and how best to implement it.

Issues for satellite teams support

3 Open 0 Closed Author Label Projects Milestones Assignee Sort

Support for EIRSAT-1 satellite teams 6
#182 opened on 19 Oct 2020 by Fergal95

Support for Team Anant satellite teams 25
#141 opened on 23 Jul 2020 by ShayanMajumder

Support for BeliefSat-1 satellite teams 21
#112 opened on 4 Jun 2020 by mbokade

Contributions from the community

VZLUSAT-2 and FossaSat-1B/-2 decoders contributed by Jan Gromeš. This led to good discussion about how to add some more general blocks, including a future generic CRC check block.

Integration with SatNOGS by Daniel Ekman `https://github.com/kng/satnogs_gr-satellites`. Running gr-satellites in the SatNOGS post-observation script, collecting decodes, and sending as observation decodes.