gr-satellites latest developments

Dr. Daniel Estévez

2 February 2020
FOSDEM 2020, Brussels
What is gr-satellites?

- A GNU Radio out-of-tree module with a collection of telemetry decoders for Amateur satellites
- Input: IQ RF samples (from SDR, conventional radio or recording)
- Output: packets in hex or parsed telemetry values
- Project goal: providing an open-source solution for decoding every satellite that transmits on Amateur bands
- Started in 2015 and has essentially been a one man’s project, but I’m always eager to collaborate with others
Currently, there are three branches and major version numbers:

- **maint-3.7** (versions 1.*). GNU Radio 3.7. No new satellites added since October 2019.
- **maint-3.8** (versions 2.*). GNU Radio 3.8. New satellites are being added here.
- **next** (future versions 3.*). GNU Radio 3.8. Large refactor of the code base. This talk is about the work in this branch.
Architecture before the refactor

- Each satellite has its own flowgraph
- Basic information about each flowgraph is included in the README
- The flowgraph contains the telemetry decoder (from IQ to PDUs) and telemetry parsers, image decoders and telemetry submitters as appropriate
- No GUI
- Some configuration parameters. Designed to run as a .py script from the terminal.
- Input is real-time real (not IQ) samples at 48ksp by UDP
- Output gets printed to the terminal, or passed on via sockets or files
Problems with this architecture

- Lots of repetition (e.g. FSK demodulator appears in many flowgraphs). Difficult to maintain.
- At the same time very flexible and not very flexible about input and output formats, and general behaviour: the user could modify whatever they like, but doing so in all the flowgraphs is cumbersome.
- Adding new satellites involves copying the flowgraph of a similar satellite and modifying it.
Main idea: satellites should be described by a simple text file and code should figure out and build the decoder flowgraph.

Use cases:

- Standalone decoder. A command line tool with enough options to be flexible.
- Building blocks for other GNU Radio decoders. Users reusing parts of gr-satellites to build flowgraphs for other decoders or customize further than allowed by the command line tool.
- Plugin. Reuse of parts of gr-satellites in other applications. Especially interested in SatNOGS Network server-side decoding, but a plugin for gqrx or similar SDR GUI app would be interesting.
Key elements for the refactor

- **SatYAML files.** A YAML file describing the satellite: basic information and specifications and protocols about its transmitters.
- **Components.** High-level elements of the decoding chain. These are hierarchical flowgraphs joining simpler low-level blocks.

The standalone decoder (command line tool) reads the SatYAML file and figures out what components to put together.

It is also possible to use components and/or lower level blocks to create flowgraphs in GNU Radio companion.
Components

The decoding chain is divided into the following high-level components:

- **Demodulators**: Convert RF samples into a stream of (soft) symbols
- **Decoders**: Convert a symbol stream into frames (GNU Radio PDUs). They perform frame boundary detection, FEC decoding and CRC checking (roughly, the physical layer)
- **Transports**: Implement upper layer network protocols, performing defragmentation if needed. Examples: a KISS stream embedded into the frames, or CCSDS Space Data Link frames that contain Space Packets.
- **Datasinks**: Do something useful with the data. Examples: telemetry decoder, telemetry submitter to SatNOGS DB, file receiver, write packets to file...
The same flowgraph using components

AFSK Demodulator
Baudrate: 4.8k
Sample rate: 48k
AF carrier: 3.6k
Deviation: -1.2k

GOMspace U482C Deframer
Syncword threshold: 4

Telemetry Submit
Server: SatNOGS DB

Telemetry Parse
Telemetry Definition: au03
Output: Standard output (stdout)

Hexdump Sink
- Describe the protocols used by the satellites in a component-centric way
- It is not easy to describe the protocols in an accurate enough way to choose a matching decoder. There are many variants, parameters and ad-hoc things.
- Rather than trying to allow a very general description, I reckon that most of the deframers used by the satellites are best described as ad-hoc
Modulations and framings supported

modulations = ['AFSK', 'FSK', 'BPSK', 'BPSK Manchester', 'DBPSK', 'DBPSK Manchester']

Using SatYAML in GNU Radio companion

**Satellite decoder**
- **Satellite definition**: Satellite name
- **Satellite name**: AU03
- **Sample Rate**: 48k

**Telemetry Submit**
- **Server**: SatNOGS DB

**Telemetry Parser**
- **Telemetry Definition**: au03
- **Output**: Standard output (stdout)

**Hexdump Sink**
The standalone command line decoder

$ gr_satellites AU03
Need to specify exactly one of the following input sources: {--wavfile, --udp, --kiss_in}

usage: gr_satellites satellite [-h] [--wavfile WAVFILE]
[--samp_rate SAMP_RATE] [--udp]
[--udp_ip UDP_IP] [--udp_port UDP_PORT]
[--kiss_in KISS_IN] [--iq]
[--input_gain INPUT_GAIN] [--kiss_out KISS_OUT]
[--kiss_append] [--hexdump]
[--telemetry_output TELEMETRY_OUTPUT]
[--clock_offset_limit CLOCK_OFFSET_LIMIT]
[--gain_mu GAIN_MU] [--deviation DEVIATION]
[--syncword_threshold SYNCWORD_THRESHOLD]
[--verbose_fec]

The command line decoder figures out the console options depending on which components the satellite to decode uses (e.g., we have a `verbose_fec` option for AU03, but not a `verbose_crc`).
Telemetry parsers

- Telemetry parsing is done using construct
- A new telemetry definition can be added by writing the construct `Struct` that corresponds to the packets transmitted by the satellite
- This telemetry definition can then be used in SatYAML files and with the “Telemetry Parser” datasink block
- Currently there are 20 telemetry definitions in gr-satellites, but there are many satellites without a parser. You can help by adding support for your favourite satellites
gr-satellites v3 includes a new generic framework to reassemble files transmitted in chunks

Images are displayed in real-time as they are received

A new decoder can be implemented by deriving from the `FileReceiver` or `ImageReceiver` class and implementing the elements of the protocol that are not already covered

Currently there is support for LilacSat-1, D-SAT, K2SAT, 1KUNS-PF, SMOG-P and Świątowid using this framework
Current status and roadmap

- Most of the features available in gr-satellites v2 are now ported to the next branch.
- A series of four alphas have been released to showcase the new architecture and functionality.
- Currently testing and tweaking the demodulator performance. A new alpha will be released after this.
- Then, probably v3.0.0 will be released. This needs documentation and unit tests.
- There are still many possible improvements and features allowed by the new architecture. These will appear in later v3.* versions.
- What about integration with SatNOGS Network? (Thread open in Libre Space forums since 2018, but not much progress done. Hopefully the new architecture will make things easier)
- Follow future work in my blog http://destevez.net or Twitter @ea4gpz