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

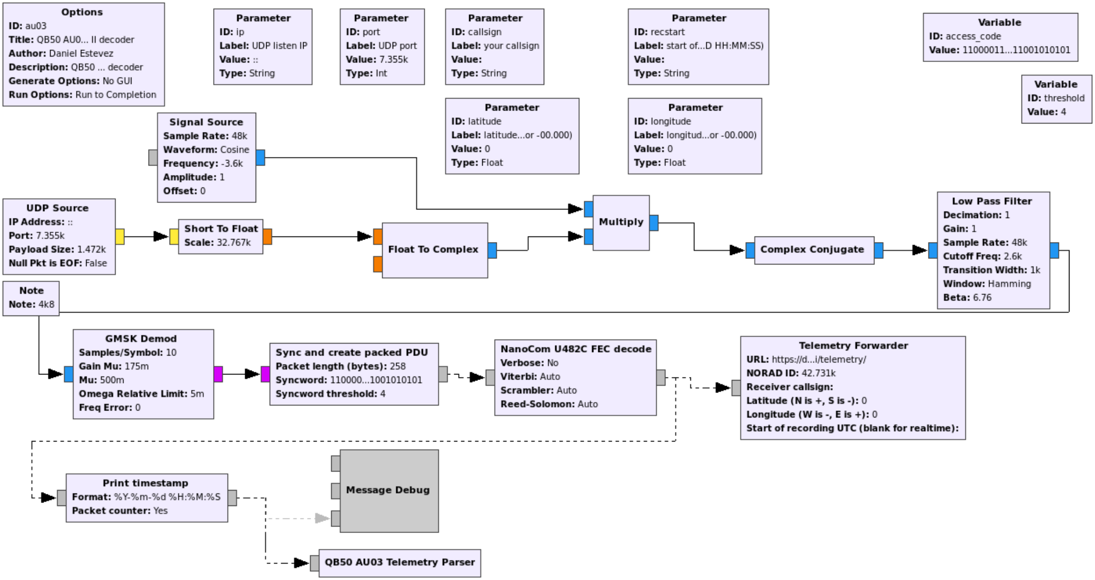
gr-satellites versioning

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 48ksps streamed by UDP
- Output gets printed to the terminal, or passed on via sockets or files



Options
ID: au03
Title: QB50 AU0... II decoder
Author: Daniel Estevez
Description: QB50 ... decoder
Generate Options: No GUI
Run Options: Run to Completion

Parameter
ID: ip
Label: UDP listen IP
Value: ::
Type: String

Parameter
ID: port
Label: UDP port
Value: 7.355k
Type: Int

Parameter
ID: callsign
Label: your callsign
Value:
Type: String

Parameter
ID: recstart
Label: start of...D HH:MM:SS)
Value:
Type: String

Variable
ID: access_code
Value: 11000011...11001010101

Variable
ID: threshold
Value: 4

Parameter
ID: latitude
Label: latitude...or -00.000)
Value: 0
Type: Float

Parameter
ID: longitude
Label: longitud...or -00.000)
Value: 0
Type: Float

Signal Source
Sample Rate: 48k
Waveform: Cosine
Frequency: -3.6k
Amplitude: 1
Offset: 0

UDP Source
IP Address: ::
Port: 7.355k
Payload Size: 1.472k
Null Pkt is EOF: False

Short To Float
Scale: 32.767k

Float To Complex

Multiply

Complex Conjugate

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 48k
Cutoff Freq: 2.6k
Transition Width: 1k
Window: Hamming
Beta: 6.76

Note
Note: 4k8

GMSK Demod
Samples/Symbol: 10
Gain Mu: 175m
Mu: 500m
Omega Relative Limit: 5m
Freq Error: 0

Sync and create packed PDU
Packet length (bytes): 258
Syncword: 110000...1001010101
Syncword threshold: 4

NanoCom U482C FEC decode
Verbose: No
Viterbi: Auto
Scrambler: Auto
Reed-Solomon: Auto

Telemetry Forwarder
URL: https://d...i/telemetry/
NORAD ID: 42.731k
Receiver callsign:
Latitude (N is +, S is -): 0
Longitude (W is -, E is +): 0
Start of recording UTC (blank for realtime):

Print timestamp
Format: %Y-%m-%d %H:%M:%S
Packet counter: Yes

Message Debug

QB50 AU03 Telemetry Parser

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.

The refactor

- 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.

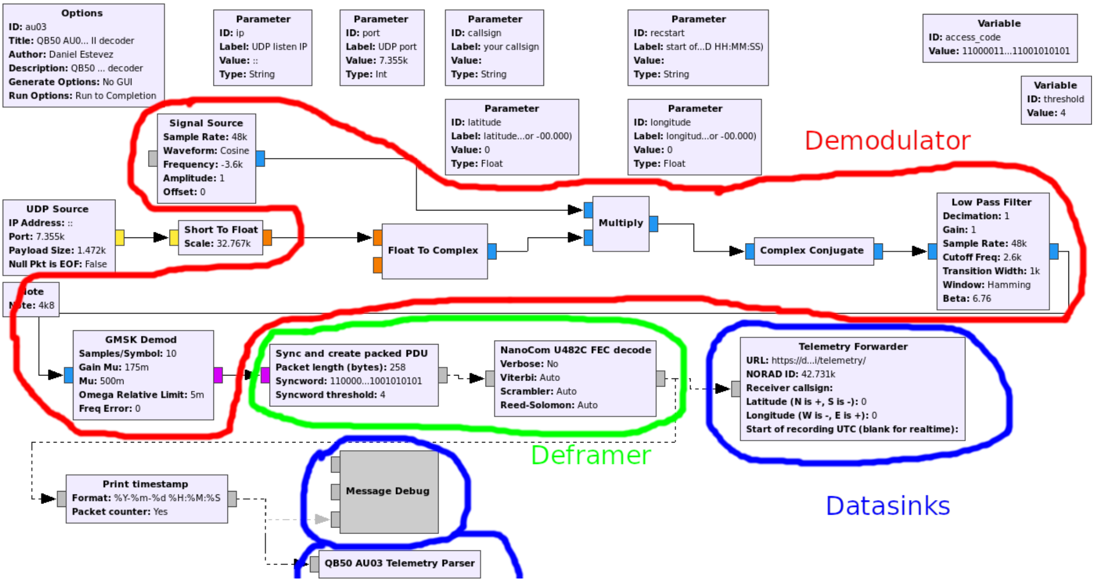
The standalone decoder (command line tool) reads the SatYAML file and figures out what components to put together.

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...



Options
ID: au03
Title: QB50 AU0... II decoder
Author: Daniel Estevez
Description: QB50 ... decoder
Generate Options: No GUI
Run Options: Run to Completion

Parameter
ID: ip
Label: UDP listen IP
Value: ::
Type: String

Parameter
ID: port
Label: UDP port
Value: 7.355k
Type: Int

Parameter
ID: callsign
Label: your callsign
Value:
Type: String

Parameter
ID: recstart
Label: start of...D HH:MM:SS)
Value:
Type: String

Variable
ID: access_code
Value: 11000011...11001010101

Variable
ID: threshold
Value: 4

Parameter
ID: latitude
Label: latitude...or -00.000)
Value: 0
Type: Float

Parameter
ID: longitude
Label: longitud...or -00.000)
Value: 0
Type: Float

Signal Source
Sample Rate: 48k
Waveform: Cosine
Frequency: -3.6k
Amplitude: 1
Offset: 0

UDP Source
IP Address: ::
Port: 7.355k
Payload Size: 1.472k
Null Pkt is EOF: False

Short To Float
Scale: 32.767k

Float To Complex

Multiply

Complex Conjugate

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 48k
Cutoff Freq: 2.6k
Transition Width: 1k
Window: Hamming
Beta: 6.76

Note
Note: 4k8

GMSK Demod
Samples/Symbol: 10
Gain Mu: 175m
Mu: 500m
Omega Relative Limit: 5m
Freq Error: 0

Sync and create packed PDU
Packet length (bytes): 258
Syncword: 110000...1001010101
Syncword threshold: 4

NanoCom U482C FEC decode
Verbose: No
Viterbi: Auto
Scrambler: Auto
Reed-Solomon: Auto

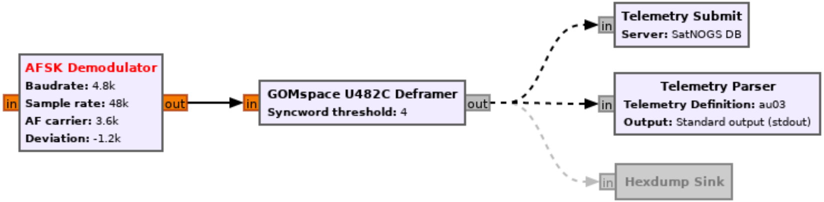
Telemetry Forwarder
URL: https://d...i/telemetry/
NORAD ID: 42.731k
Receiver callsign:
Latitude (N is +, S is -): 0
Longitude (W is -, E is +): 0
Start of recording UTC (blank for realtime):

Print timestamp
Format: %Y-%m-%d %H:%M:%S
Packet counter: Yes

Message Debug

QB50 AU03 Telemetry Parser

The same flowgraph using components



SatYAML

- 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

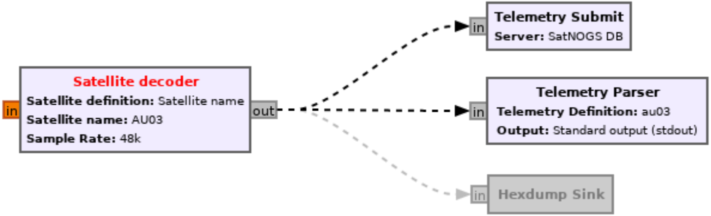
```
name: AU03
alternative_names:
  - QB50 AU03
  - i-INSPIRE II
norad: 42731
data:
  &tlm Telemetry:
    telemetry: au03
transmitters:
  4k8 AFSK downlink:
    frequency: 436.330e+6
    modulation: AFSK
    baudrate: 4800
    af_carrier: 3600
    deviation: -1200
    framing: U482C
    data:
    - *tlm
```

Modulations and framings supported

```
modulations = [’AFSK’, ’FSK’, ’BPSK’, ’BPSK Manchester’,
  ’DBPSK’, ’DBPSK Manchester’]
```

```
framings = [’AX.25’, ’AX.25 G3RUH’, ’AX100 ASM+Golay’, ’AX100 Reed Solomon’,
  ’3CAT-1’, ’Astrocast FX.25 NRZ-I’, ’Astrocast FX.25 NRZ’, ’Astrocast 9k6’,
  ’AO-40 FEC’, ’AO-40 FEC short’, ’AO-40 uncoded’, ’TT-64’, ’ESEO’, ’Lucky-7’,
  ’Reaktor Hello World’, ’S-NET’, ’Swiatowid’, ’NuSat’, ’K2SAT’,
  ’CCSDS Reed-Solomon’, ’CCSDS Reed-Solomon dual’, ’CCSDS Reed-Solomon differentia
  ’CCSDS Reed-Solomon dual differential’, ’CCSDS Concatenated’, ’CCSDS Concatenate
  ’CCSDS Concatenated differential’, ’CCSDS Concatenated dual differential’,
  ’LilacSat-1’, ’AAUSAT-4’, ’NGHam’, ’NGHam no Reed Solomon’, ’SMOG-P RA’,
  ’SMOG-P Signalling’, ’OPS-SAT’, ’U482C’]
```

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

File/image receiver

- 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 Światowid 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 fuctionality
- 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