



Communicating with AcubeSAT via a SatNOGS based platform

Sunday, 26 Oct 2025



ARISTOTLE
UNIVERSITY
OF THESSALONIKI

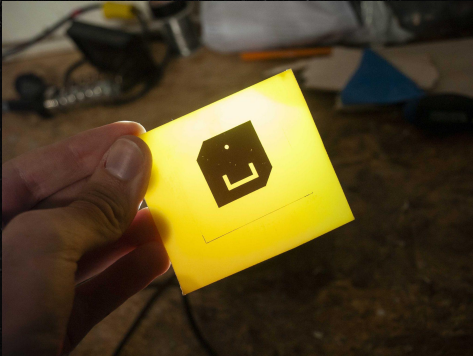


FLY YOUR SATELLITE!

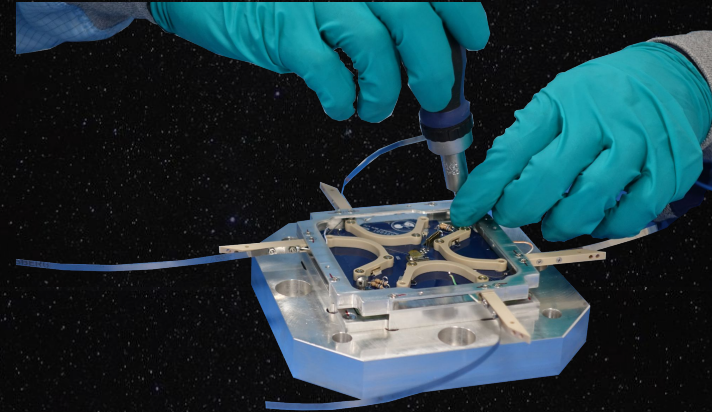
The Communications Subsystem (1)

The AcubeSAT communications subsystem consists of:

1. A near omni-directional UHF deployable turnstile antenna (in-house)
2. A circularly polarized S-band patch antenna (in-house)



Manufacturing of the S-band patch antenna



UHF turnstile antenna and its deployment mechanism

The Communications Subsystem (2)

The COMMS Board module consists of:

- In-house modified version of the SatNOGS COMMS hardware, by Libre Space Foundation
- In-house software design
- In-house FPGA IP Cores design

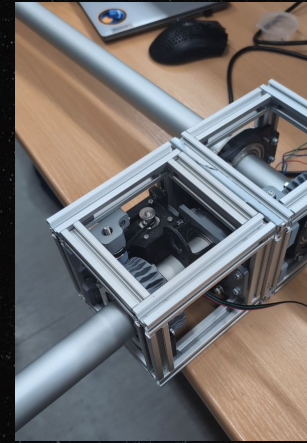


Manufactured COMMS Board

The Communications Subsystem (3)

The Ground Station segment includes:

- S-band 1m parabolic antenna operating at 2.4 -2.45 GHz
- UHF SatNOGS Helical Antenna v5 operating at 435-438 MHz
- The SatNOGS Rotator v3 alongside with the SatNOGS rotator controller PCB



Rotator Assembly



Helical Antenna

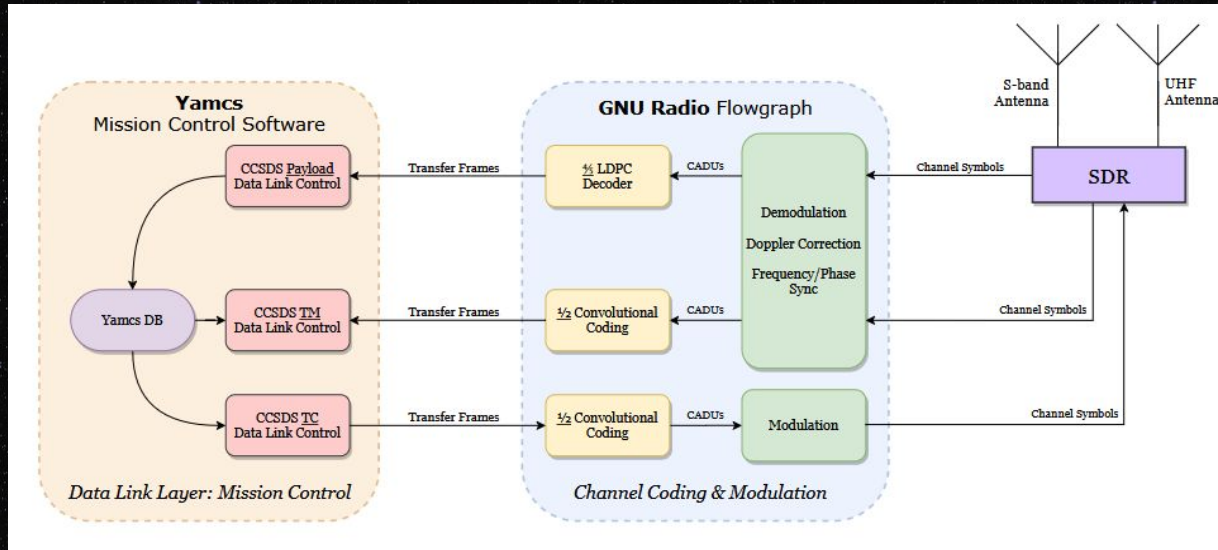


Parabolic Dish

Downlink/Uplink

For the UHF Downlink/Uplink GMSK mod/demod will be used alongside with BCH decoding/coding

For the S-band Downlink OQPSK mod/demod will be used alongside with LDPC decoding



Functional Architecture of the Ground Station

On-Board Modulation: FPGA (1)

Why FPGA for Telecommunications?

Time critical DSP tasks

Parallel processing

Pipelining

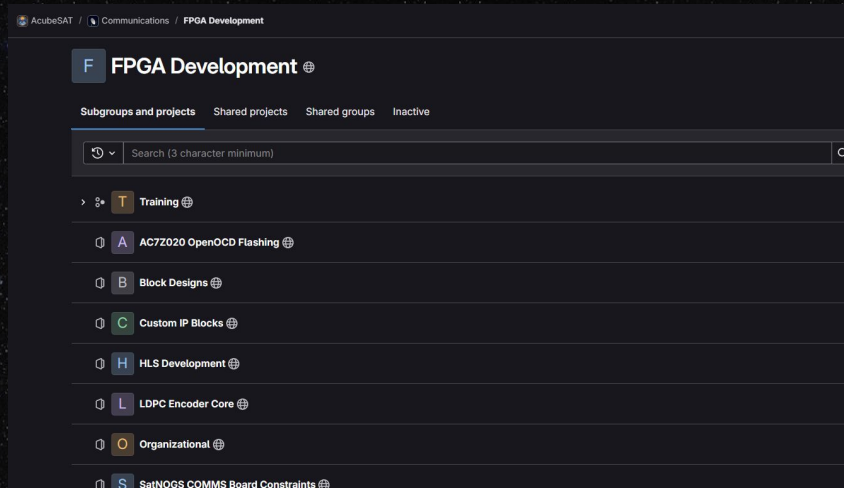
Pipelining Reconfigurable design



Manufactured COMMS Board
during functional tests

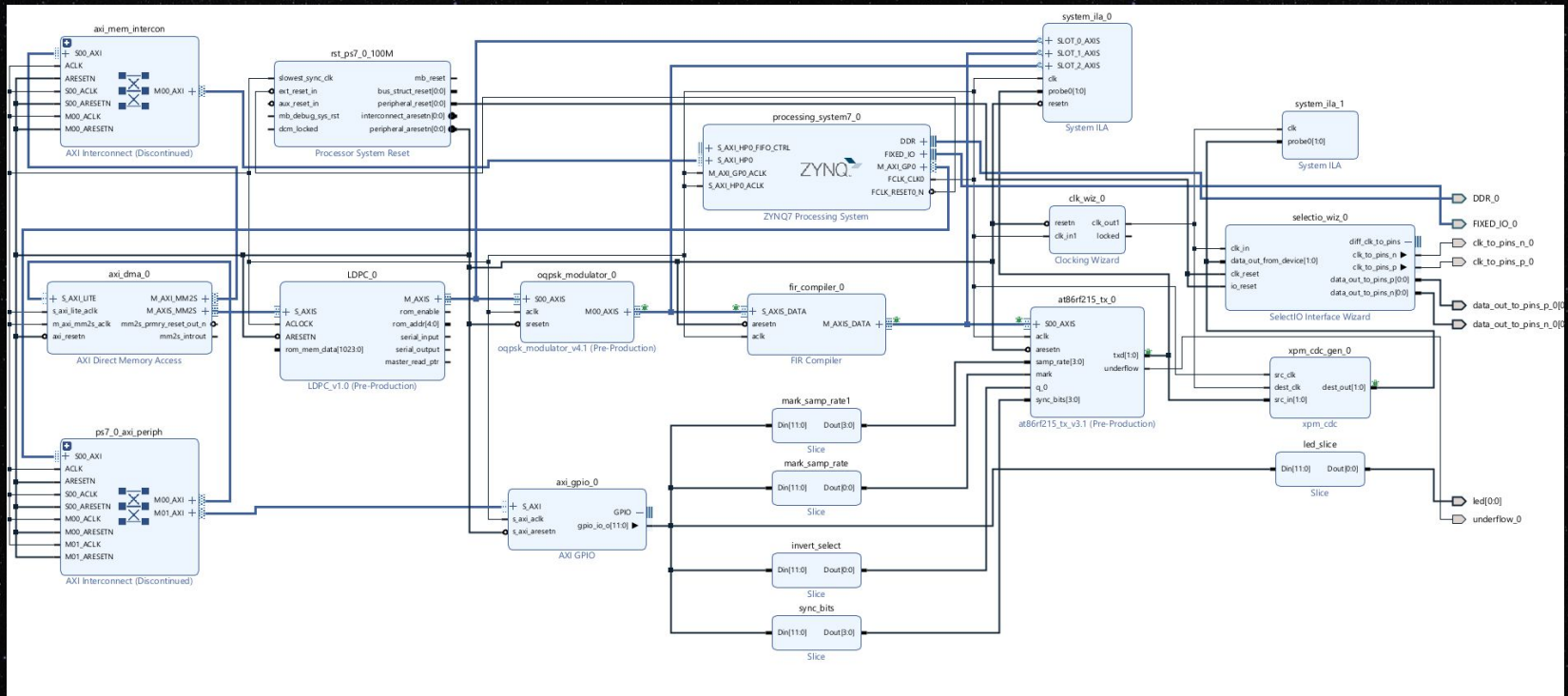
On-Board Modulation: FPGA (2)

- Tools used: Vivado / Vitis by Xilinx AMD
- Open source code in GitLab:
<https://gitlab.com/acubesat/comms/fpga-development>



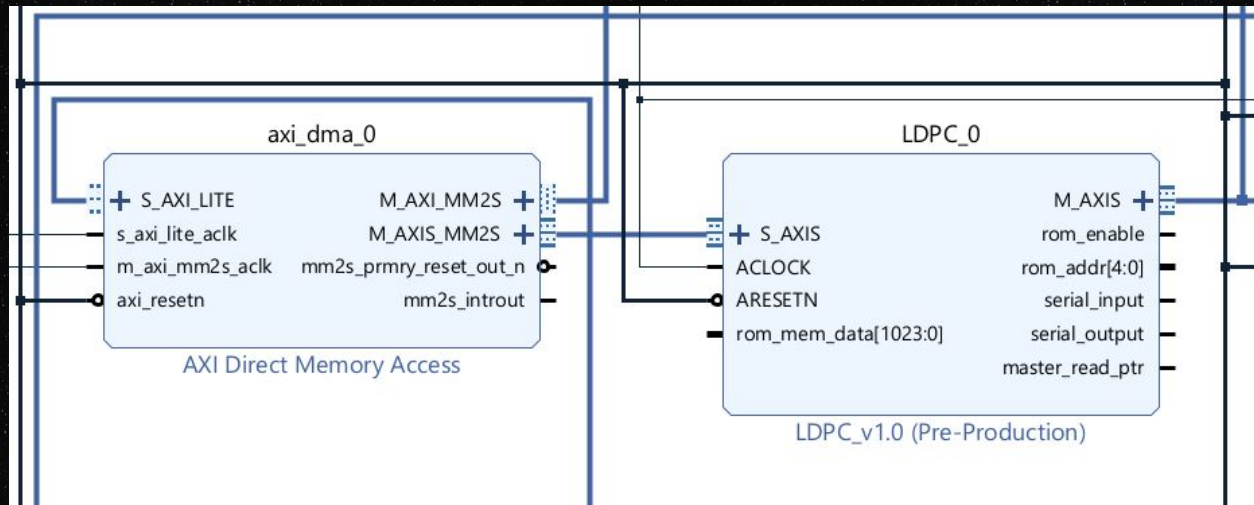
On-Board Modulation: FPGA (3)

- The S-band modulation chain:



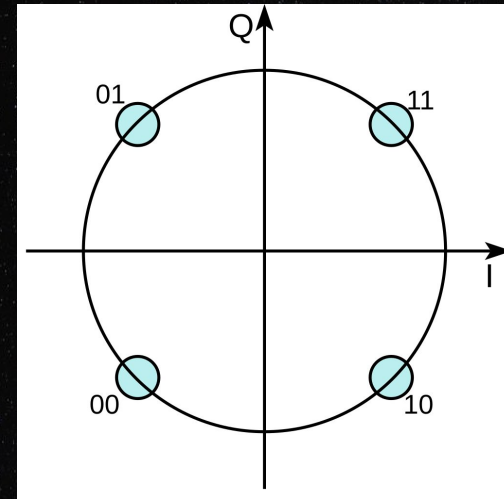
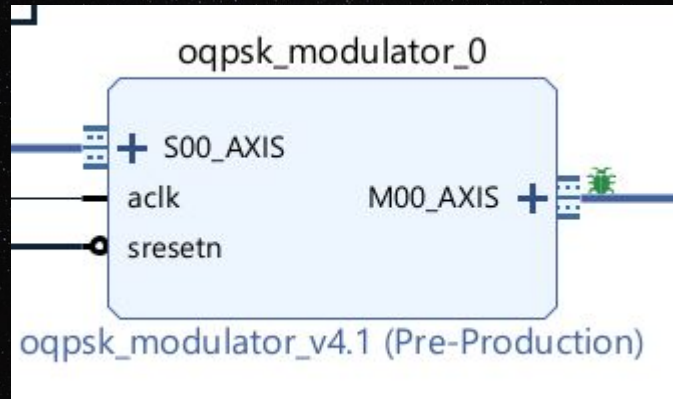
On-Board Modulation: FPGA (4)

- Receiving image data from MCU using SPI
- Passing them into the Programmable Logic (PL) using a Direct Memory Access (DMA) core starting with the LDPC encoder

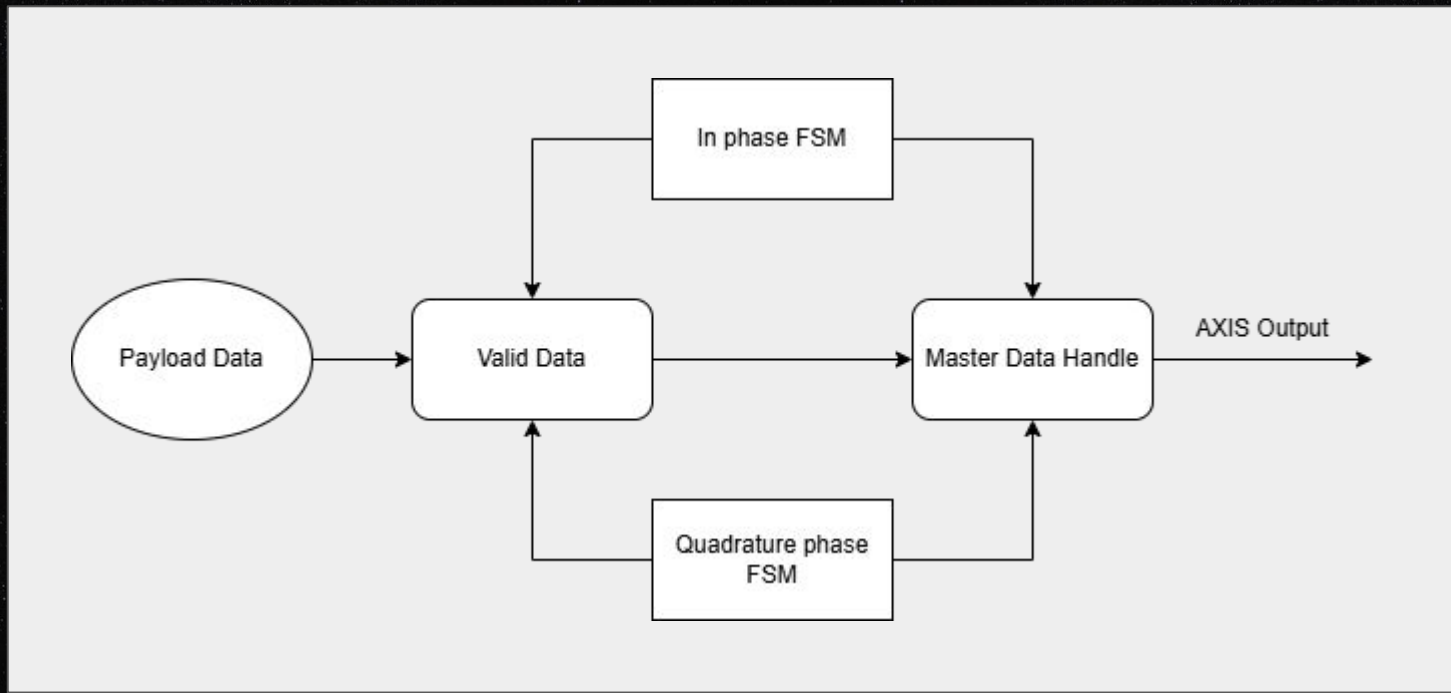


OQPSK

- Recommended by [CCSDS-401.0-B-29-S](#)
- High efficiency & Robustness



OQPSK Core Overview



OQPSK - Datapath (1)

Valid Data : Separation of the input into I/Q components

```
logic [C_S00_AXIS_TDATA_WIDTH / 2 - 1 : 0] inphase_extracted;
logic [C_S00_AXIS_TDATA_WIDTH / 2 - 1 : 0] quadrature_extracted;
always_comb begin
    for (int i = 0, int j = 0, int k = 0; i < C_S00_AXIS_TDATA_WIDTH; i++) begin
        if (i % 2 == 0) begin
            quadrature_extracted[j++] = s00_axis_tdata[i];
        end
        else begin
            inphase_extracted[k++] = s00_axis_tdata[i];
        end
    end
end
end
```

OQPSK - Datapath (2)

- Master Data Handler: Matching I/Q bits to the corresponding amplitudes

```
localparam SqrtTwoOverTwoPos = 16'b0101101000010010; // Is +0.70709228515625 approximately +sqrt(2)/2.
localparam SqrtTwoOverTwoNeg = 16'b1010010111101110; // Is -0.70709228515625 approximately -sqrt(2)/2.

always_comb begin
    if (inphase_state == SEND_OFFSET) begin
        m00_axis_tdata[31:16] = 0;
    end
    else begin
        m00_axis_tdata[31:16] = inphase_valid_data[inphase_data_counter] ? SqrtTwoOverTwoPos : SqrtTwoOverTwoNeg;
    end
end
```

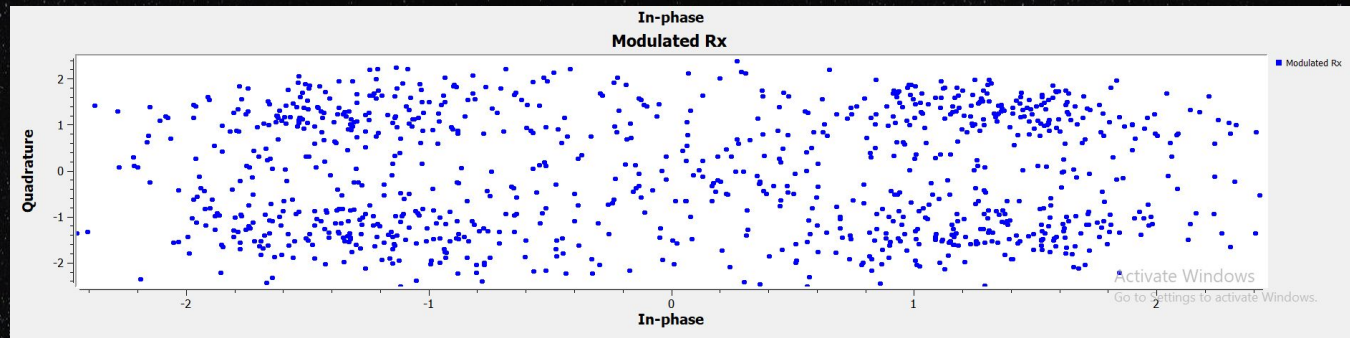
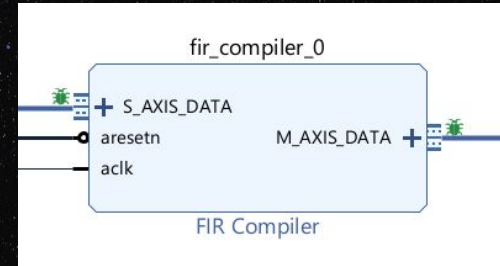
OQPSK State Machines

- Two Finite State Machines (FSM): one for each component
- Each handling the state of the component transmission
- Switching from SEND to OFFSET in one component and reverse in the other ensures our desired time delay of the Quadrature component

```
typedef enum logic [1:0] {  
    SEND_DATA = 2'b00,  
    SEND_OFFSET = 2'b01,  
    IDLE = 2'b10,  
    RESET = 2'b11  
} transmission_state_t;
```

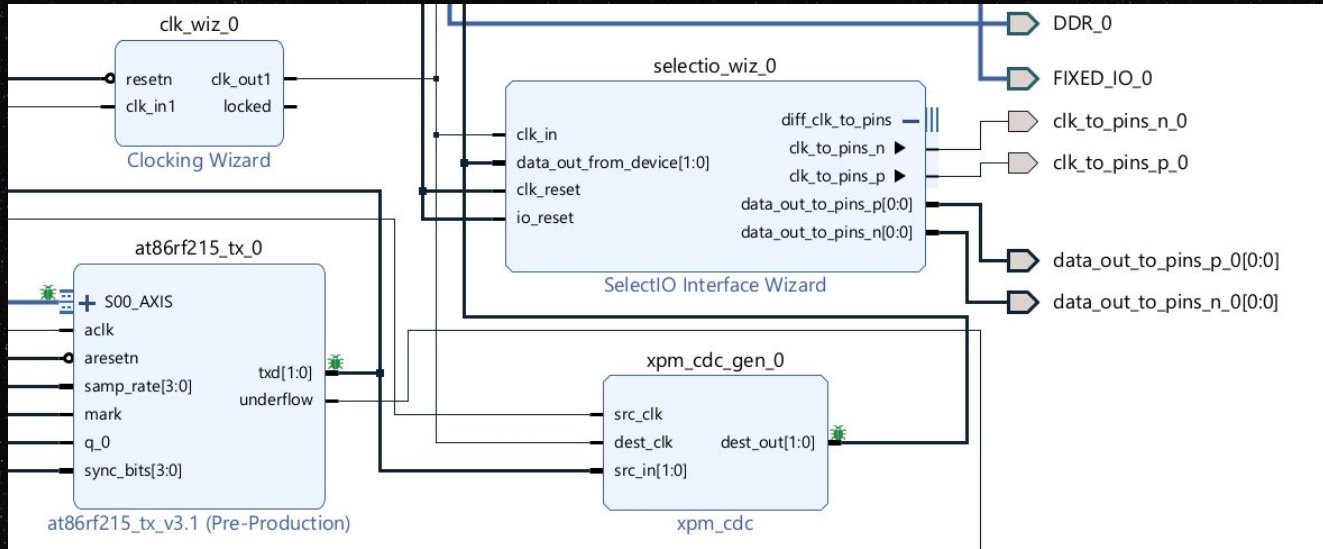
Root Raised Cosine Filter & ISI

Last DSP task is sample filtering using a Xilinx Library core to mitigate Inter-Symbol Interference (ISI)

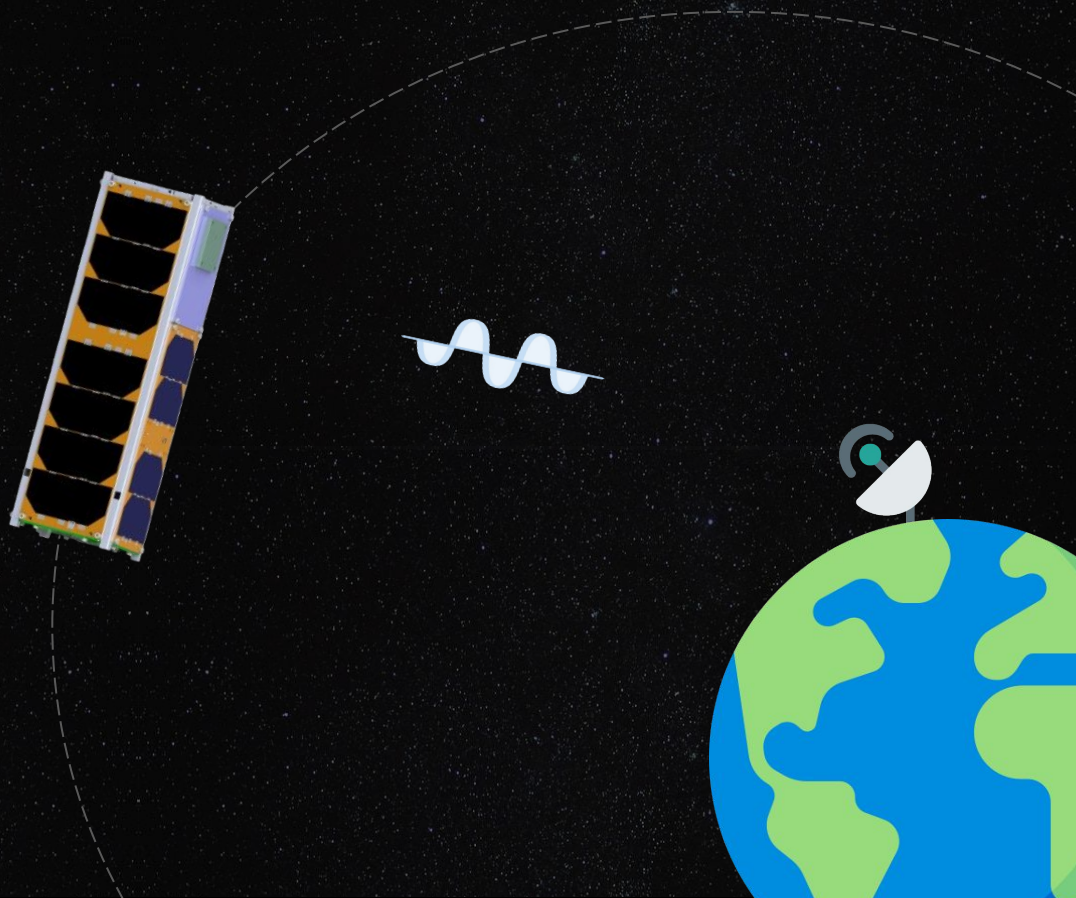


LVDS Transceiver

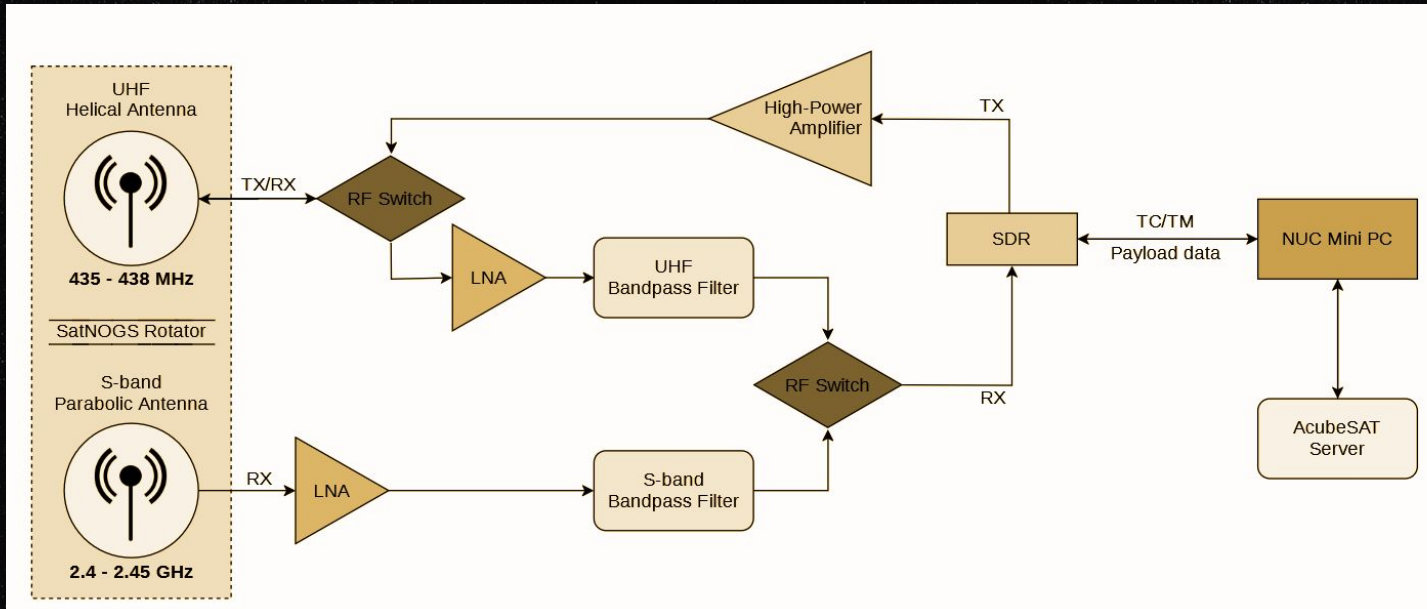
- Transceiver driver leading the filtered output onto the LVDS interface
- Higher clock domain needed for this



AcubeSAT Transmitting Image Data



OQPSK Demodulation



Block Diagram of the Ground Station

OQPSK Demodulation

Frequency & Phase Lock

Error Correction & Channel Coding

Doppler Compensation

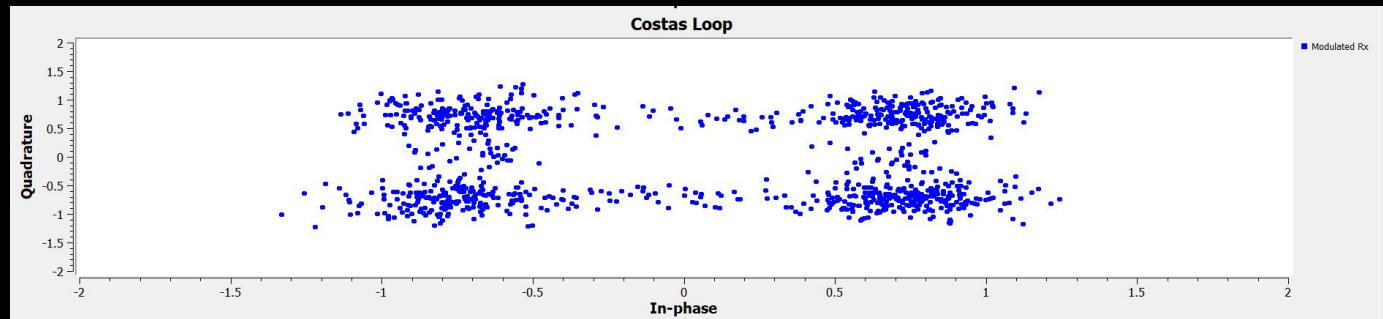


Open-source, free software toolkit that provides signal processing blocks used for our signal processing radio.

OQPSK Demodulation

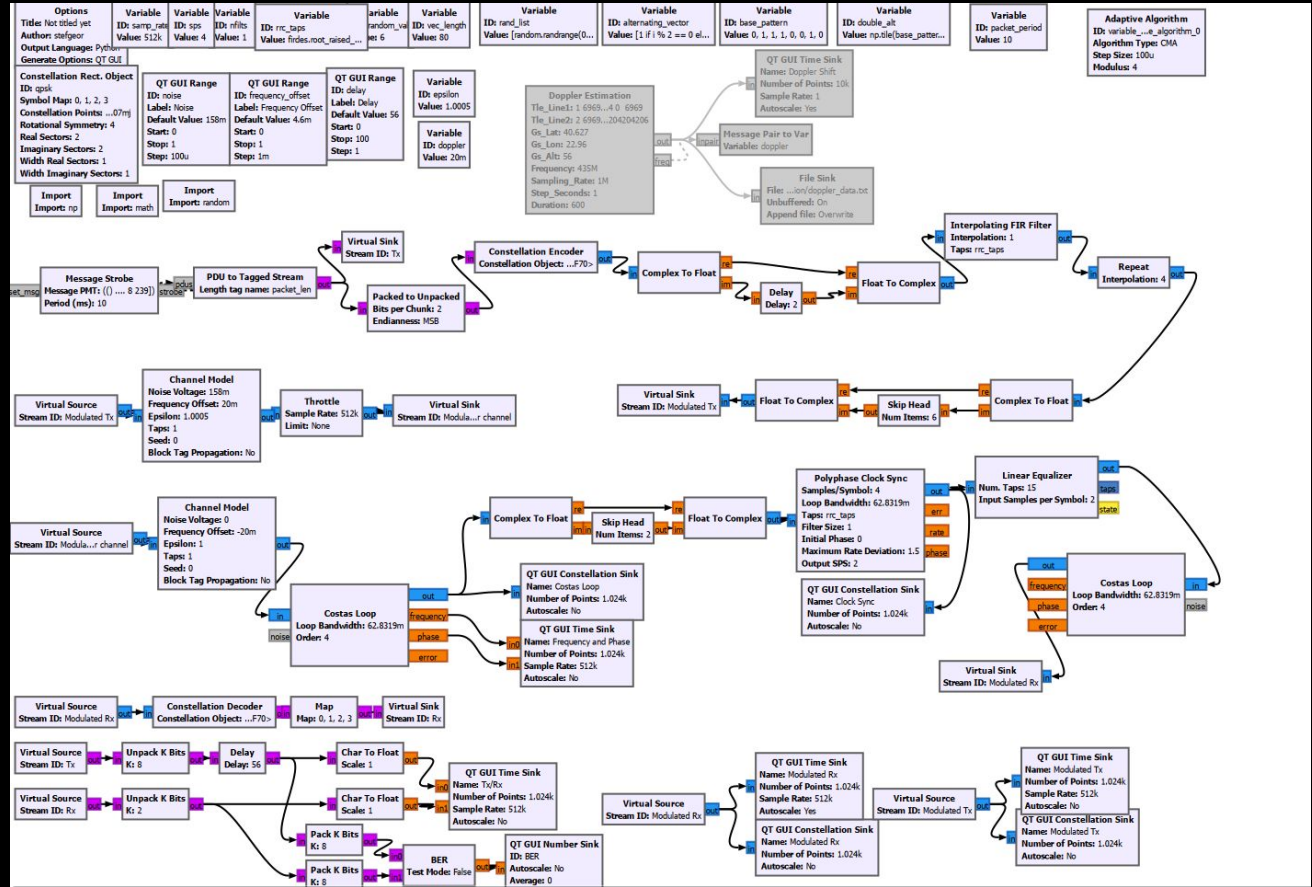
Doppler Compensation

- dynamically calculate the frequency offset based on the data given from the TLE file
- subtract the frequency from the demodulation process, by entering an opposite frequency offset in the Channel Model
- insert the signal through a Costas Loop, to compensate for the noise inserted by the channel



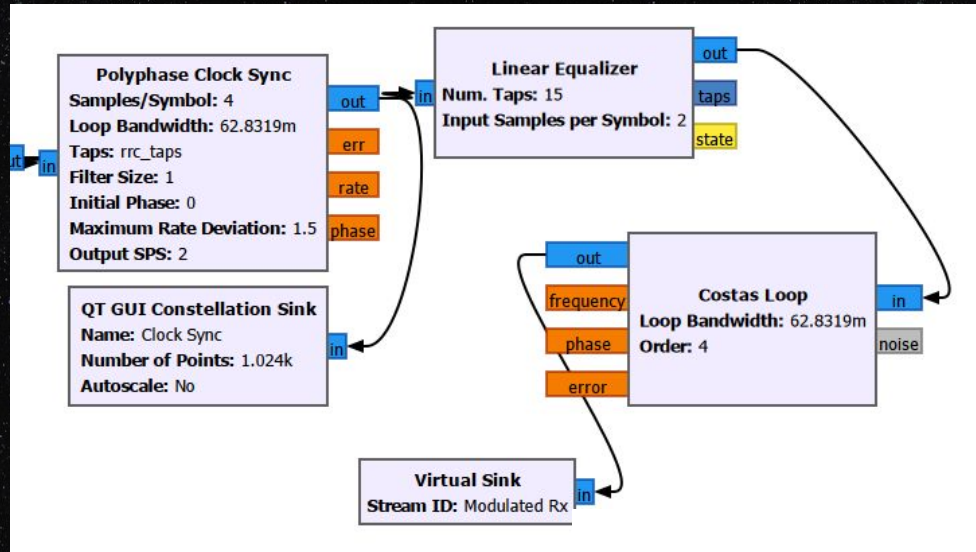
Flowgraph of the QPSK modulation/demo

ulation (SDR)



OQPSK Demodulation

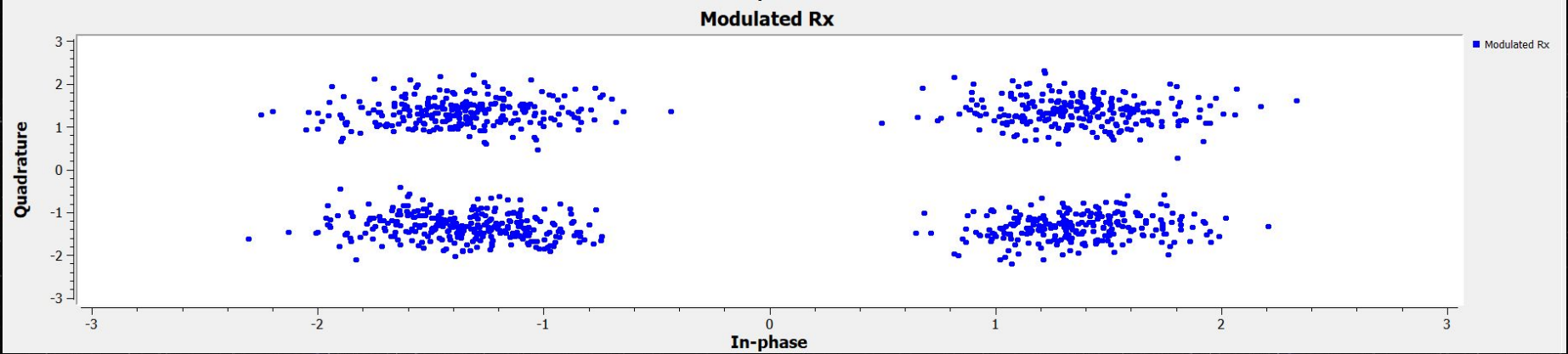
Polyphase Clock Sync:
Time synchronization while maximizing SNR and minimizing ISI



Linear Equalizer:
Inserts some additional noise

2nd Costas Loop:
avoid any further rotation in the final constellation

OQPSK Demodulation

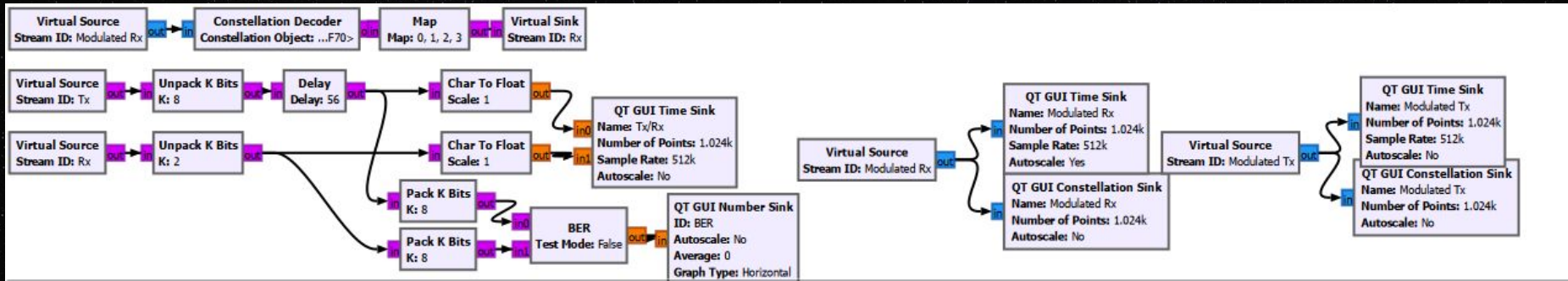


Final Constellation

OQPSK Mapping

After finalizing the constellation, we transfer the cleaned constellation points into binary data and measure BER performance by:

- obtaining the synchronized complex symbols from the constellation
- mapping back into bits
- comparing them to the transmitted ones



Next Steps



Verification of
the IP Cores
presented



End to end
testing for the
UHF and S-band



Finalization of
the GSMK Rx/Tx
chain



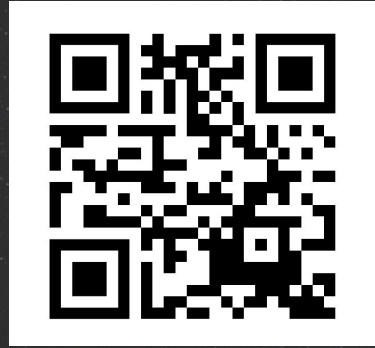
COMMS Board
integrated in the
flatsat

Thank you!

Read more about us:

gitlab.com/acubesat

acubesat.spacedot.gr



Contact us:

k.gkaripis@spacedot.gr

i.petkou@spacedot.gr