SATELLITE COMMUNICATIONS SYSTEMS

Simulation of a satellite receiver system for GNSS signals

Davide Tomasella (M.2037849) - davide.tomasella@studenti.unipd.it
Lorenzo Borsoi (M.2056099) - lorenzo.borsoi.1@studenti.unipd.it
Mattia Piana (M.2053041) - mattia.piana@studenti.unipd.it
Gabriele Andreetta (M.2053121) - gabriele.andreetta@studenti.unipd.it

University of Padova

Master's degree in ICT for Internet and Multimedia - Photonics
Master's degree in ICT for Internet and Multimedia - Telecommunications
Master's degree in Electronics Engineering

May 20, 2022

1 Project overview

Quick description

2 Development strategy

We organize the development of our project into different phases to allow the development of the core functionalities autonomously. In the followings, we will briefly describe our workflow.

1. **Initial considerations:** We discussed the aims of the project and our previous ideas about how a satellite receiver works and how we can realize its simulation. We define the working principle of our algorithm.
2. **Task division:** After a global understanding of the whole project, we split the algorithm into independent middle-level functionalities implemented with classes. We assigned each of them to a different person to start the development autonomously.
3. **High-level implementation:** We transcribe the simulator working principles into a high-level algorithm that calls the previously defined middle-level operations. We also decide the classes' input and output interfaces.
4. **Core functionalities' implementation:** Each class represents different middle-level functionalities and includes configuration, action, and low-level functions. We collaborated on the most challenging parts, developing and testing the code.
5. **Test of core functionalities:** We developed several test signals to cover different cases, and we tested the separated classes. We also evaluate how the configuration parameters impact the algorithms.

6. **Opinion sharing:** During the development, we carried out weekly meetings to present our advancement, discuss the implementation choices, and decide how to solve the problems and improve the simulator.

7. **Integration into high-level algorithm:** We then integrate the core functionalities into the high-level simulator to test the whole system and analyze how they interact with each other.

8. **Report writing:** We wrote the final report, collecting and summarizing the core functionalities and the main results of the simulator.

3 Simulation algorithm

We implemented the entire simulation code with MATLAB, using file interfaces to acquire the input signal and parameters and save the simulation results on top of the performance plots. The file `RECEIVER.m` contains the high-level algorithm, and it is also described in Fig. 1. Firstly, the read sampled signals are filtered to remove the high-frequency noise, and the acquisition procedure starts. Once the message synchronization pattern with the correct PRN is found, the simulation proceeds to the tracking phase, performed in parallel on multiple symbols. Here we compensate the Doppler envelope and filter the signal with a narrower bandwidth before the actual estimation of time and frequency shifts. The symbols are demodulated, and we repeat the tracking till the end of the message. Finally, we analyze the received message and save the simulation results.

The following sections describe the classes adopted to implement the core functionalities, focusing on their working principle and tunable parameters.

3.1 Requirements definition

We translate the requirements defined by the Scenario Manager into a list of parameters that can be provided to the simulation though a `JSON` file. In the following, we list them specifying the parameter name, type, and range, and its description:

- `fSampling`: `float`, 2-60 MHz, sampling frequency of the baseband signal (affected by Doppler shift)
- `quantizationBits`: `int`, 8/16/24/32, number of bit per I/Q sample in the binary file
- `scenarioDuration`: `float`, 1-60 s, duration of the simulated signal (determines the dimension of the `IQsamples` file)
- `SV_PRN_ID`: `int`, 1-50, PRN sequence of *Primary Codes for the E1-B Component* (see GALILEO OD SIS ICD)
- `CRCpolynomial`: `hex-string` (24 bit), generator polynomial G(x) for CRC value creation
- `SYNCpattern`: `bin-string` (10 bit), `0101100000`, fixed synchronization header
- `TAILpattern`: `bin-string` (6 bit) `000000`, fixed padding tail
- `SVIDlength`: `int`, 6, length of SV ID field
- `MIDlength`: `int`, 4, length of Message ID field
- `MBODYlength_ACK`: `int`, 30, length of Message Body field with ACK packets
- `CRClength`: `int`, 24, length of CRC sequence

- `nPRN_x_Symbol`: `int`, 1-10, number of PRN repetition for each symbol, determines the symbol period with the following parameters
- `nChip_x_PRN`: `int`, 4092, length of the PRN sequence of *Primary Codes for the E1-B Component* (see GALILEO OD SIS ICD)
- `chipRate`: `float`, $1.023\,\mathrm{MHz}$, chip rate of the PRN sequence
- `maxDoppler`: `float`, $100\,\mathrm{KHz}$, maximum Doppler frequency, useful to estimate the signal bandwidth and filter noise

In the same way, we also define `JSON` file containing the simulation output, in particular, the decoded message and the log information of the algorithm:

- `version`: `string`, version of the algorithm
- `ACQUISITION_OK`: `bool`, debug log of acquisition algorithm
- `TRACKING_OK`: `bool`, debug log of the tracking algorithm
- `DEMODULATION_OK`: `bool`, debug log of the message analysis
- `SV_ID`: `string 6char`, binary ID of target satellite (demodulated)
- `message_ID`: `string 4char`, message ID (demodulated)
- `message_body`: `string 30char`, message body (demodulated)
- `CRC`: `string 24char`, CRC of the message (demodulated)
- `ACKed`: `bool`, true if the computed CRC is equal to the received one
- `isACKmessage`: `bool`, true if the `message_body` contains ACK flag (first bit is 1)
- `estimatedDopplerStart`: `float`, estimated doppler frequency after acquisition procedure
- `estimatedDopplerEnd`: `float`, estimated Doppler frequency after message demodulation (tracking procedure)
- `estimatedDelay`: `float`, estimated time delay (from the start of the `IQsamples` file) during the acquisition procedure
- `estimatedPhase`: `float`, $0 - 2\,\pi$, estimated envelope phase (at `estimatedDelay`) during the acquisition procedure

3.2 Samples reading and management

3.3 Signal acquisition

3.4 Down-conversion and filtering

3.5 Signal tracking

3.6 Message decoding

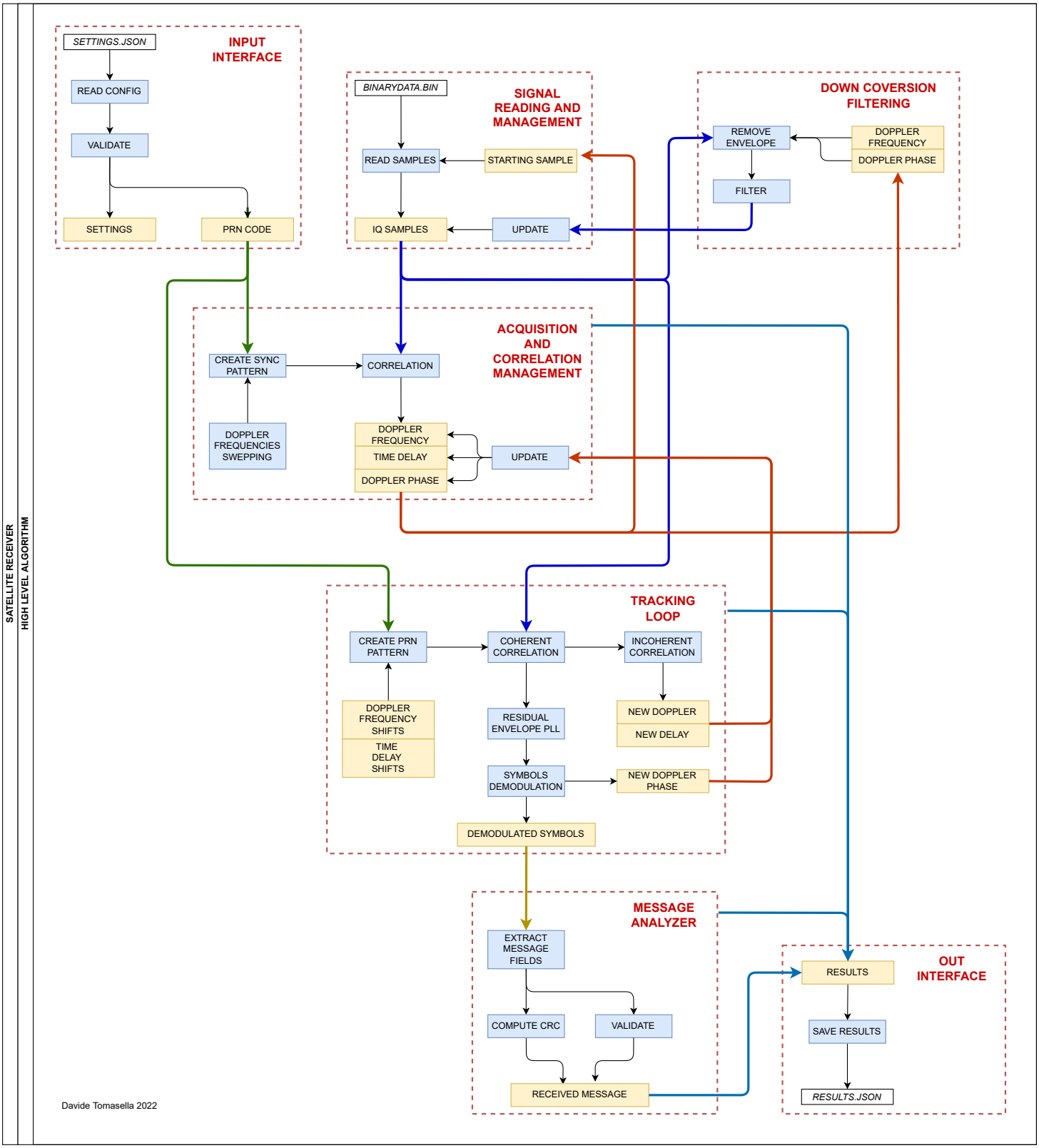

FIGURE 1: Scheme of the high-level steps of the simulation algorithm.

4 Testing and results

4.1 Test signals generation

5 Conclusions

References