



Article

Framework to Emulate Spacecraft Orbital Positioning Using GNSS Hardware in the Loop

David Forero ¹, Segundo Esteban ^{2,*}  and Óscar Rodríguez-Polo ¹ 

¹ Space Research Group, Polytechnic School, University of Alcalá, 28805 Alcalá de Henares, Spain; david.forero@edu.uah.es (D.F.); o.rodriguez@uah.es (Ó.R.-P.)

² Department of Computer Architecture and Automatic Control, Faculty of Physic Sciences, Complutense University of Madrid, 28040 Madrid, Spain

* Correspondence: sesteban@ucm.es

Abstract: The paper presents a framework to emulate spacecraft orbits using GNSS hardware in the loop that enables the evaluation of new orbital positioning algorithms. The framework software generates the spacecraft orbit and the GNSS signals, including the most common perturbations. These signals are modulated and transmitted by a software-defined radio and received by a commercial GPS receiver. The system is validated using a test orbit, where the GPS receiver accurately determines the spacecraft positions. Moreover, using raw data provided by the receiver, the spacecraft positions have also been determined by software for a low earth orbit, in which civil GPS receivers do not work.

Keywords: satellite orbital simulation; satellite orbital determination; global navigation satellite system (GNSS); software defined radio (SDR); hardware in the loop (HIL)

1. Introduction

A Global Navigation Satellite System (GNSS) signal emulator is a valuable tool for the development of space systems. Its use facilitates the design and validation of the orbital control system of terrestrial satellites, as well as the Guidance, Navigation and Control (GNC) system of different types of spacecraft, including those aimed at lunar exploration [1,2]. Its scope of application can also be extended to other areas such as aviation, robotics, or ground, maritime and river navigation, offering a wide range of possibilities for both research and education.

In this context, it is particularly important to have test environments available to analyse the performance of a GNC system of any kind in the early stages of development. This is the main objective of this paper. Specifically, having a reliable emulator allows us to test, at a very low cost, different estimation algorithms, or the tolerance of a system against computer attacks, as well as many other scenarios involving GNSS.

To ensure the robustness of a GNC system, the environment must be sufficiently versatile and representative, allowing the analysis of the functionality, reliability and efficiency of the system, taking into account all the elements and sources of error that may affect the test [3], as well as the different scenarios that reproduce all the operational conditions.

In order to achieve results with a larger comprehension of real scenarios, it is essential to include as much GNSS system hardware and software as possible in the test. The final objective is to be able to validate the expected system behaviour, rigorously checking the result of the estimation and control algorithms that take the position provided by the GNSS receiver as the primary source of information. Including the GNSS receiver in this type of validation testing is key, as its performance and behaviour will propagate throughout the entire system.

According to this, the paper shows not only the framework design but its adaptation to the problem of satellite orbit estimation using a configurable Software Defined Radio (SDR) GNSS transmitter and commercial GNSS receivers. Specifically, the mitigation of



Citation: Forero, D.; Esteban, S.; Rodríguez-Polo, Ó. Framework to Emulate Spacecraft Orbital Positioning Using GNSS Hardware in the Loop. *Sensors* **2023**, *23*, 885. <https://doi.org/10.3390/s23020885>

Academic Editor: Maorong Ge

Received: 11 November 2022

Revised: 7 January 2023

Accepted: 10 January 2023

Published: 12 January 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

perturbation introduced by the receivers hardware has been studied and a solution has been provided and evaluated by orbital estimation results. In this way, the framework has been used to identify and correct the error due to the receiver's local clock bias.

In the article, an analysis of the state of the art on the use of GNSS in aerospace applications is first carried out. Then, the structure of the developed Framework (FW) is presented, both its mathematical foundation, as well as the hardware and software involved. The following sections explain in detail the two main FW subsystems and their hardware and software components. Next, the system is experimentally validated using an orbital position estimation problem, where the perturbation introduced by the receivers hardware is characterised and mitigated. Finally the conclusions are presented.

2. Related Works

In the last two decades, different Hardware-In-the-Loop (HIL) solutions have been proposed to evaluate performance of GNSS units, satellite formation navigation and estimation algorithms.

In 2009, Jae-Ik Park et al. established an HIL simulation test-bed to evaluate estimation and navigation algorithms [4]. In this development, however, the estimation is carried out using the carrier phase as the GNSS observable, and not the pseudorange, as proposed in our paper. Furthermore, their environment uses a commercial signal simulator, and does not offer the versatility of an SDR-based transmitter, which can be configured by file and thus work with any GNSS constellation, including those future constellations that will possibly be deployed in other environments, such as Mars or the Moon, for which open and configurable analysis environments will be required.

Wang Liduana et al. present, in 2010, a software-based GPS measurement simulator on L1 frequency and coarse acquisition (C/A) code for a space-oriented navigation system [5]. The simulator, coded in MATLAB language, generated both C/A code and carrier phase measurements. This work simulates and processes only the GPS signal and therefore does not generate the physical spread spectrum RF signals corresponding to the full GNSS constellation. Furthermore, it does not use real receivers and therefore does not take into account disturbances associated with the receiver hardware.

In 2015, Wang et al. showed that time and position data from mobile devices can be easily spoofed using very low-cost and open source tools [6]. This work shares with our paper the approach of using an SDR transmitter and low-cost hardware to build the environment. However, the objective pursued is different in that it does not address the complete problem of accurate position estimation of the receiving satellite, but rather the ability of the environment to fool commercial GNSS receivers under normal operating conditions over the Earth's surface and at bounded velocities.

In 2016, Arul and Sudha designed a new and simple low-cost L1 GPS signal simulator to test and evaluate GPS receiver performance by software in a laboratory environment [7]. This simulation environment does not use HIL, so it does not facilitate the empirical evaluation of the effect of disturbances due to receiver electronics on the estimation of the orbital position and does not analyze the possible correction mechanisms to be applied.

In 2017, Peng developed a hardware testbed based on GNSS emulation for a group of spacecraft flying in low-altitude orbit with the ability to remotely detect the ionosphere [8]. This work addresses the same problem posed in this article: to have an environment that allows for emulating the reception of the GNSS signal and using it to determine the orbital position of satellites. However, as with the environment developed in [4], the use of a commercial GNSS signal simulator restricts its use, and does not allow the analysis of future GNSS constellations.

In 2018, Ebinuma developed a low-cost GPS simulator using software-defined radio, with which dry GPS attacks can be developed [9]. Finally, in 2020, Cao presented his PhD thesis on practical GPS spoofing attacks on consumer drones using the vulnerabilities of civilian GPS drones [10]. These two papers share the same environment approach based on the use of an SDR transmitter and commercial GPS receivers that has been used in our

work, but as in [6], they do not address the problem of satellite orbital position estimation, and therefore do not offer a solution to the problem of using commercial receivers due to compliance with the Wassenaar agreement.

3. Framework Structure

3.1. Theoretical and Mathematical Basis of the Framework

The main objective of the FW is to generate the signals of a GNSS system to be received by a spacecraft (S/C) in space-time coordinates (t, x, y, z) . GNSS positioning technology is based on the calculation by the receiver of the travel times of signals emitted from satellites in the GNSS constellation, whose position and time of emission are coded in the signal itself. To make this calculation possible, a common reference time, called GNSS time, has been defined, which has a constant offset to International Atomic Time (abbreviated TAI, from the French name Temps Atomique International), which in the equations we will denote by t . However, while the satellites of the GNSS constellations are equipped with redundant high-precision caesium or rubidium atomic clocks, the receiving spacecrafts use crystal oscillators that are not as accurate and sensitive to many disturbances, leading to various synchronisation problems. A modelling of these synchronisation problems can be found in [11,12]. Equation (1) shows the time of the GNSS constellation transmitters, which are synchronised with each other, as a function of their offset $\delta t^s(t)$. Equation (2) shows the receiver time as a function of its offset $\delta t_r(t)$.

$$t^s(t) = t + \delta t^s(t) \quad (1)$$

$$t_r(t) = t + \delta t_r(t) \quad (2)$$

The main observable is the direct measure of the travel time $t_r^s(t)$ of the $P(Y)$ signal transmitted from the phase centres of the GNSS satellite antennas to the receiver. This time, as it is stated in [13], can be calculated by Equation (3) that contains the error introduced by clock offsets at both the transmitter and receiver. While $P(Y)$ signal is encrypted, several techniques have been developed that allow observations to be made without the decryption hardware key [14]. In this case, there is a loss in the Signal/Noise ratio which results in a reduction in accuracy:

$$t_r^s(t) = t_r(t) - t^s(t - \tau_r^s(t)) \quad (3)$$

where $t_r(t)$ is the local time at which the signal is received, transmitted from the GNSS satellite at its local time $t^s(t - \tau_r^s(t))$, with $\tau_r^s(t)$ the true travel time. By replacing (1) and (2) in (3), Equation (4) is obtained.

$$t_r^s(t) = \tau_r^s(t) + \delta t_r(t) - \delta t^s(t - \tau_r^s(t)) \quad (4)$$

Multiplying by the speed of light, we have in (5) the first approximation of the observable distance between the GNSS satellite and the spacecraft, called the pseudorange and denoted by $P_r^s(t)$, where the term ρ_r^s in the equation represents the true distance:

$$P_r^s(t) = \rho_r^s + c \cdot (\delta t_r(t) - \delta t^s(t - \tau_r^s)) \quad (5)$$

This equation does not yet take into account perturbations due to atmospheric effects, multi-path instrumental delays, or other effects such as thermal noise, although, in the case of low-orbit space missions, the only atmospheric effect that should be considered would be the ionospheric effect. This effect, defined in (6), is due to the delay of paths on the electromagnetic waves caused by ions and free electrons in the ionosphere, and depends on the frequency of the waves [12]:

$$I_r^s(t, f) = \frac{40.3}{f^2} TEC_r^s(t) \quad (6)$$

where TEC_r^s represents the total electron density along the signal path, and f is the frequency of waves.

The thermal noise $\epsilon_{rp}^s(t)$ is modelled in the literature as a random variable with zero mean. The rest of the perturbations, such as the instrumental delay $b_{rp}^s(t)$, the multi-path $m_{rp}^s(t)$ and the system errors $S_{rp}^s(t)$, will be grouped in the term $M_{rp}^s(t)$ defined in Equation (7):

$$M_{rp}^s(t) = b_{rp}^s(t) + m_{rp}^s(t) + S_{rp}^s(t) \quad (7)$$

Adding these perturbations, it is possible to compute the pseudorange required to build the GNSS signal, $\rho_{tx}(t)$, as shown by Equation (8):

$$\rho_{tx}(t) = P_r^s(t) + I_r^s(t, f) + M_{rp}^s(t) + \epsilon_{rp}^s(t) \quad (8)$$

3.2. Framework Hardware and Software

The FW physically consists of several hardware blocks in charge of emulating the spacecraft GNSS signals. The GNSS signal is generated by a low-cost Software Defined Radio (SDR) (HackRF One [15]) and received by a GNSS receiver (ublox NEO M8T [16]). The SDR is controlled by a personal computer. A software block, developed in Python and C languages, computes and transfers the baseband signal file for each S/C. The GNSS receiver is controlled by the On-Board Computer (OBC), in this case, a RaspBerry-Pi. Figure 1 shows a high-level view of this FW.

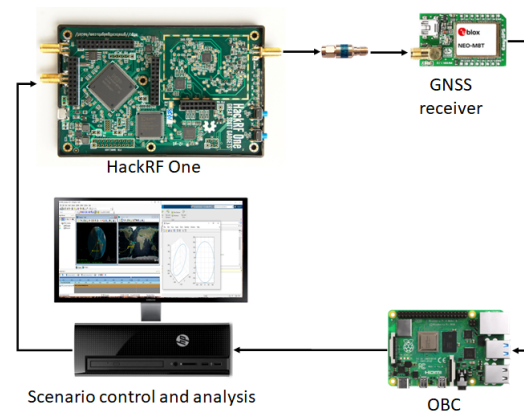


Figure 1. Hardware interconnection of GNSS emulation framework.

The block diagram in Figure 2 shows more in detail the different functional blocks involved in the prototype.

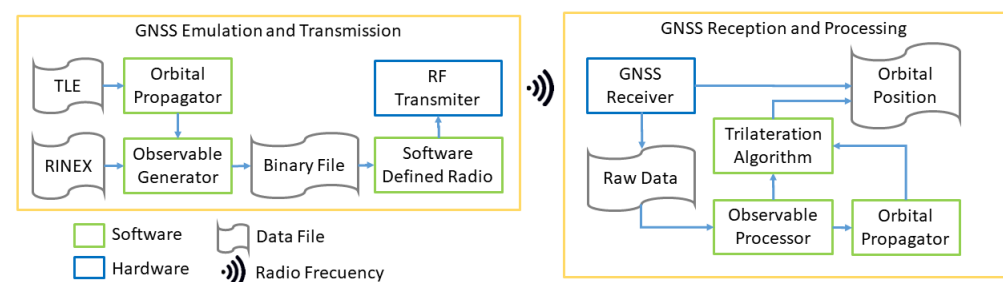


Figure 2. Functional diagram block of the framework.

As can be seen in the figure, one of the inputs to the functional model is a TLE (Two Line Elements) file. This file contains two lines of data encoding a list of orbital elements at a given epoch and is used by the FW to generate the spacecraft orbital trajectories using the SGP4 propagator [17].

The second input to the functional model is a navigation RINEX (Receiver Independent Exchange Format) file, which provides the GNSS constellation ephemeris data. The RINEX file is processed to generate the orbits of the constellation's satellites at different epochs.

Using the positions and velocities of the satellites relative to the spacecraft, it is possible to generate the observables to be captured by the GNSS receiver. These observables are encoded and modulated for radio frequency transmission using a SDR.

To finish the emulation chain, the radio signal is received and processed by the GNSS receiver in order to provide the orbital position. However, the receiver does not return any orbital position in a LEO orbit, due to the Wassenaar agreement. In this situation, it is necessary to work with raw data and process the observable by software.

4. GNSS Emulation and RF Transmission

As explained above, the GNSS emulator purpose is to generate, in real time, the RF signals that should be received by a spacecraft GNSS receiver. This block implements three software modules: the first two, named *Orbital Propagator* and *Observable Generator*, work together to generate the *Binary File* with the GNSS signal to be transmitted. The last module runs in a Software Defined Radio and configures the spread spectrum transmitter in charge of radiating this information. Each of these three software modules is explained in detail in the following subsections. One aspect to be taken into account is that orbital propagators normally work under the reference system known as Earth Center Inertial (ECI), while GNSS positioning algorithms normally use Earth Centered Earth Fixed (ECEF). For this reason, it will be necessary at several stages of the emulation to perform transformations between the two systems. In this FW, the transformations between the World Geodetic System (WGS 84), used by the GPS system, and the True Equator Mean Equinox (TEME) system, used by the SPG4 orbit propagator, have been implemented. In case of using other GNSS or propagators, appropriate transformations would have to be implemented.

4.1. Orbital Propagator

In order to emulate the GNSS signals that a spacecraft would receive during a mission, the first step is to calculate the spacecraft orbit. Since the FW has as its first reference scenario the GNSS emulation in near-Earth orbits, a Python implementation of the SGP4 propagator has been chosen to generate Low Earth Orbits (LEO). This propagator calculates the orbit of the spacecraft from the orbital parameters provided in a TLE file [18].

For the propagation of the GNSS satellite constellation, a higher precision propagator is implemented in the Observable Generator module, which includes multiple corrections parameterised in the RINEX files. Figure 3 shows an example representation of the propagation of the orbits of the spacecraft and a GPS constellation using ECI as a reference system.

4.2. Observable Generator

As it was mentioned before, this module is in charge of generating the GNSS baseband to be transmitted. It uses as inputs the navigation RINEX file together with the spacecraft orbit file, previously calculated by the propagator. The output provided by this module is a binary file with the baseband that must be transmitted by the radio and received by the spacecraft GNSS receiver.

In order to provide a realistic emulation, the Observable Generator integrates the ability to incorporate perturbations that can be enabled and disabled via software. As previously described, GNSS receivers suffer from perturbations due to tropospheric, ionospheric and relativistic effects, as well as inaccuracies in their internal clocks and instrumental delays. Most of the perturbations are modelled in the RINEX file, including: the clock synchronisation parameters (toc) a_0 , a_1 y a_2 , the instrumental delay manipulation tgd and the ionospheric parameters $(\alpha_1, \dots, \alpha_4)$ and $(\beta_1, \dots, \beta_4)$.

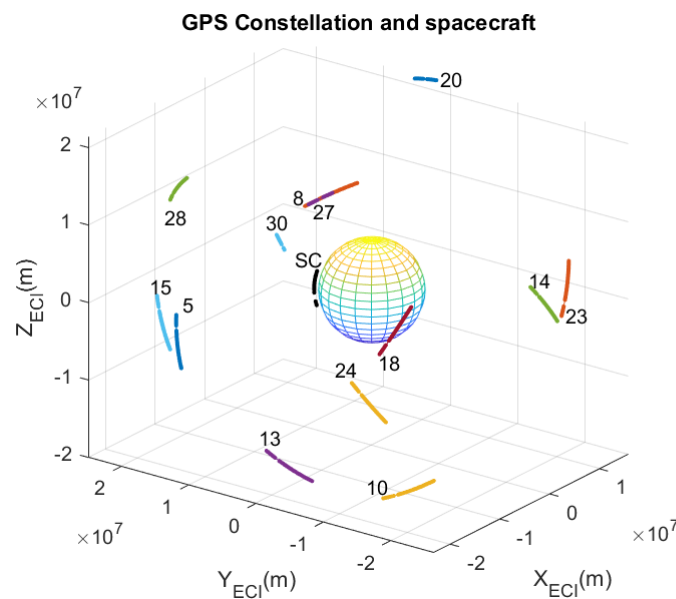


Figure 3. Orbital propagation of the S/C and GPS constellation.

The pseudo-code Listing 1 shows the implementation of the GNSS Observable Generator. In the initial stages, the ReadRINEXFile and ReadOrbitFile functions read the inputs, information of GNSS constellation and the orbit of the spacecraft to be emulated. This is followed by the configuration of the active perturbations to be processed, the buffer allocation and the initialization of the channels. Then, a loop is entered which simulates the relative positions of the transmitting satellites and the spacecraft. In this loop, for each SC position of the orbit the GNSS positions are retrieved using GetGNSSEphemeris function, the visible satellites are determined in DetermineVisibleGNSSs function, and the observable signals are generated by ObservableGenerator function. AddPerturbations function adds modification to the observable signals for each active perturbation. Finally, GenerateBasebandSignal function generates the final signal, and it is added to the output binary file. This generator is implemented in C, and a Python wrapper has been built to integrate it into the simulator.

Listing 1. GNSS signal generator.

```

Initialization ()
ReadRINEXFile ()
ReadOrbitFile ()
ActivePerturbation ()
InitializeBuffer ()
InitializeChannels ()
While (not end S/C orbit)
{
    GetGNSSEphemeris ()
    DetermineVisibleGNSSs ()
    ObservableGenerator ()
    AddPerturbations ()
    GenerateBasebandSignal ()
    AppendToBinaryFile ()
}

```

Figure 4 compares the true path, $\rho_r^s(t)$, in the left graphic, and the pseudoranges used in the transmission, $\rho_{tx}(t)$, in the right graphic. Although, due to the scale, they look quite similar, and their differences can be seen in the graph below. According to Equation (8), these differences are mainly due to the travel time, the clock error of GPS transmitters

and the instrumental delay of the GPS transmitters, since atmospheric disturbances are negligible, and thermal noise has been disabled. This graph already shows that each pseudorange is perturbed in a different way. For the case of space orbits, the ionospheric and tropospheric effects can be discarded, but the rest of the effects cannot. In particular, instrumental delay, clock biases and the relativistic effect, due to the high orbital velocities, are quite relevant. The transmitted pseudoranges are very important in this simulator because they indirectly define the moment at which the radio signals should be transmitted to the S/C because the receiver is wired to the radio.

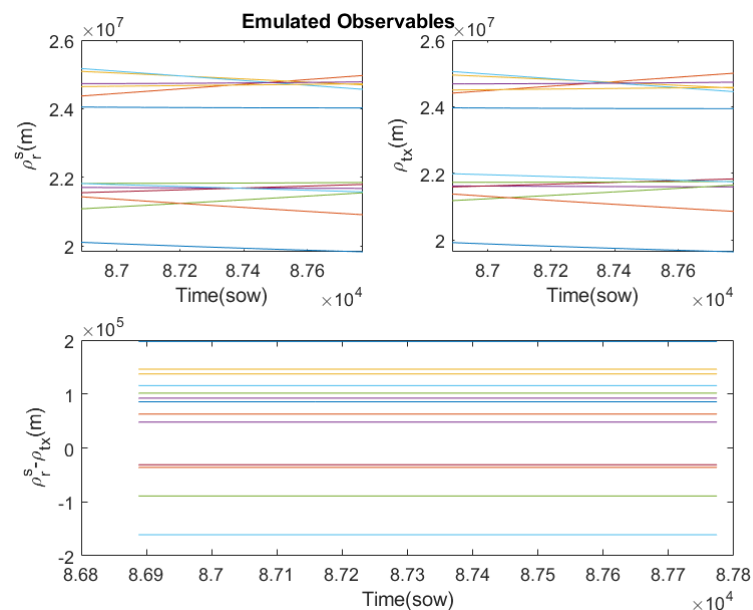


Figure 4. Difference between true path and pseudoranges used in the transmission for the GPS constellation (each color represents a different GPS emitter).

4.3. Software Defined Radio

Software defined radio (SDR) is a radio communication system where the typical hardware components of a radio system are implemented in software. In this project, GNU Radio software toolkit is used [19]. The software produces a signal identical to that transmitted by the GNSS constellation. This signal is generated in the digital domain and then converted into an analogue signal in the time domain. Finally, it is radiated by means of an antenna or a guided medium to the GNSS receiver. The SDR implemented in this project is a simple Direct Sequence Spread-Spectrum (DSSS) modulator represented in the block diagram in Figure 5.

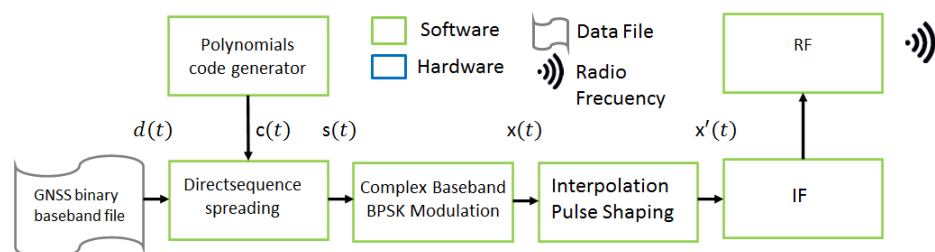


Figure 5. Functional block diagram of a simple DSSS modulator.

The elements involved in the DSSS modulator are:

- GNSS binary baseband file, with output $d(t)$, is the binary file with the observable encoding generated by the *Observable Generator* routine.

- Polynomials code generator, with output $c(t)$, implements a shift register with linear LFSR feedback to generate pairs of predefined sequences representing a six degree polynomial as a random source of the polynomial generator.
- Direct sequence spreading, with output $s(t)$, performs a spread spectrum modulation of the signals $d(t)$ and $c(t)$.
- Complex Baseband BPSK Modulation, with output $x(t)$, is a binary phase shift keying (BPSK) modulator at spread spectrum.
- Interpolation Pulse Shaping, with output $x'(t)$, interpolates each BPSK symbol with K samples according to a rectangular pulse to transform the baseband to an intermediate frequency.
- IF Stage converts the interpolated signal to an intermediate frequency analog signal.
- RF Transmit adds the carrier to convert the system into a radio transceiver.

4.4. Hardware Tx

The radio used in the FW is a HackRF One. It is a low-cost SDR peripheral with the capacity of transmitting or receiving radio signals, from 1 MHz to 6 GHz, designed to enable the testing and development of radio technologies [19]. The main characteristics of HackRF One are:

- Operating frequency 1 MHz a 6 GHz;
- Half-duplex transceiver;
- Twenty million samples per second;
- Eight bit of quadrature samples (I y Q);
- Receive and transmit gain configurable for software;
- The antenna connector is an SMA.

The hardware of the radio is very simple, and Figure 6 shows a block diagram with the main components. The host computer runs the SDR code and produces digital signal samples at a sampling rate lower than the ADC/DAC rate. The FPGA uses the data sample stream from the host computer and performs high sample rate signal processing to make the resulting digital signal compatible with the ADC/DAC requirements. The high sample rate processing is performed in the FPGA, while the low sample rate processing is performed in the host computer running the SDR algorithms.

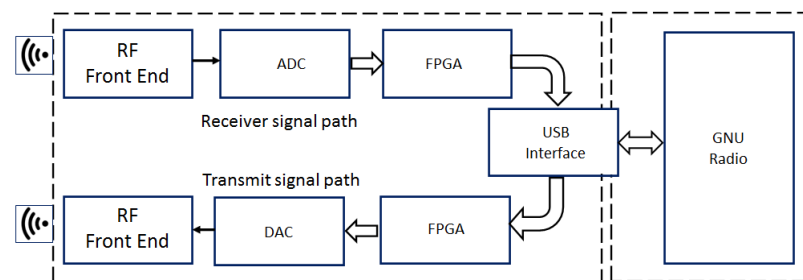


Figure 6. HackRf One main components: Radio Frequency interface (RF Front End), Analog to Digital Converter (ADC) and Digital to Analog Converter (DAC), Field-Programmable Gate Array (FPGA) and a USB interface connected to the GNU Radio.

5. GNSS Reception and Processing

At this stage, it is possible to work with commercial or ad hoc designed GNSS receivers. The *GNSS Receiver* hardware can either determine the position directly or provide the *Raw Data* to be processed by software blocks: *Observable Processor*, *Orbital Propagator* and *Trilateration Algorithm*.

5.1. GNSS Receiver

To validate the simulator, the commercial GNSS receiver NEO-M8T from Ublox [16] has been used. It is a medium performance receiver, compatible with the BeiDou, GLONASS,

Galileo and GPS constellations. This receiver is limited by the Wassenaar agreement, which restricts the operation of GNSS receivers to altitudes below 50,000 m and velocities up to 500 m/s. Because of this, it does not determine the position for space orbits. In this case, the receiver NEO-M8T provides the raw measurements shown in Table 1.

Table 1. Raw data of GPS receiver: reception time in Second Of Week (SOW), transmitter identifier (ID), the pseudorange measurement (ρ_{rx}), the phase carrier measurement (ϕ_r^s) and the Doppler measurement or zero crossings of the received frequency. The column $\rho_{proc}(m)$ is the processed pseudorange.

SOW $t_r(s)$	ID	$\rho_{rx}(m)$	$\phi_r^s(Hz)$	Doppler	$\rho_{proc}(m)$
432,381	G10	23,114,092.3772	107,533,978.113	−1009.187	23,114,092.3772
432,381	G24	22,062,373.2656	102,007,202.204	1776.915	22,160,204.6500
43,2381	G13	20,289,894.7344	926,927,48.9384	1532.4572	20,438,859.9648
432,381	G15	23,910,225.0839	111,717,657.7537	−1215.151	23,999,154.3198

5.2. Processing

To process the raw data, a software GNSS receiver with three stages has been implemented:

- *Observable Processor:* To estimate the true distances, $\rho_i^s(t)$, the corrections of the different perturbations are added to the pseudoranges measured by the receiver, $\rho_{rx}(t)$. These estimates are called processed pseudoranges, $\rho_{proc}(t)$, which are shown in Table 1. With these values, the travel times and the transmission times, $t^s(t)$, can now be estimated.
- *GNSS Orbital Propagator:* The GNSS satellites orbits are propagated using transmission times and ephemerides of the constellation. In Figure 3, the orbits of GPS satellites have some discontinuities when the receiver has not registered a signal for those GPS satellites.
- *Trilateration Algorithm:* The GNSS satellite positions and the processed pseudoranges are the inputs of a trilateration algorithm to calculate the position of the spacecraft. As a use case, an algebraic solution of trilateration problem has been implemented [20], but many others algorithms may be evaluated.

6. Framework Validation

Next, a step-by-step validation of the developed environment is carried out. For this purpose, the GPS constellation has been emulated and different orbits for the S/C have been evaluated using the developed FW.

6.1. Validation of GNSS Emulation and Transmission

The first stage is to validate the GNSS Emulation and Transmission stage. For this purpose, the GPS receiver has been used as a black-box to check whether it understands the signals emitted by the emulator and calculates the orbital positions correctly. In order for the receiver to determine the position, it is necessary to work with an orbit that complies with the Wassenaar agreement. Therefore, it has been necessary to introduce a small modification in the orbital propagator that allows us to modify two orbital parameters, the semi-major axis and the mean momentum of the orbit. Thanks to the fact that SPG4 is a Kepler propagator, the orbit, although lower and slower, follows geometrical laws and does not fall on the Earth's surface.

In this case, the FW generates the radio signals from the modified orbit of the S/C, and the GPS receiver is fed with these signals. After a few seconds of initialisation, the GPS receiver determines the S/C's position and orbital velocity. Although this receiver has Spoofing detection, it does not detect anything anomalous. Figure 7 shows the orbital positions propagated by the FW and the orbital position determined by the GPS receiver and their residuals. The GPS is using a Kalman filter to smooth the noise, but also intro-

duces a transitory behaviour. After the transitory, the residuals are within the technical specifications of the GPS receiver.

These results validate the GNSS emulation and transmission stage, as the signals are understood by a commercial receiver. Furthermore, it appears that the accuracy of the radio used is more than sufficient to generate synthetic GNSS signals.

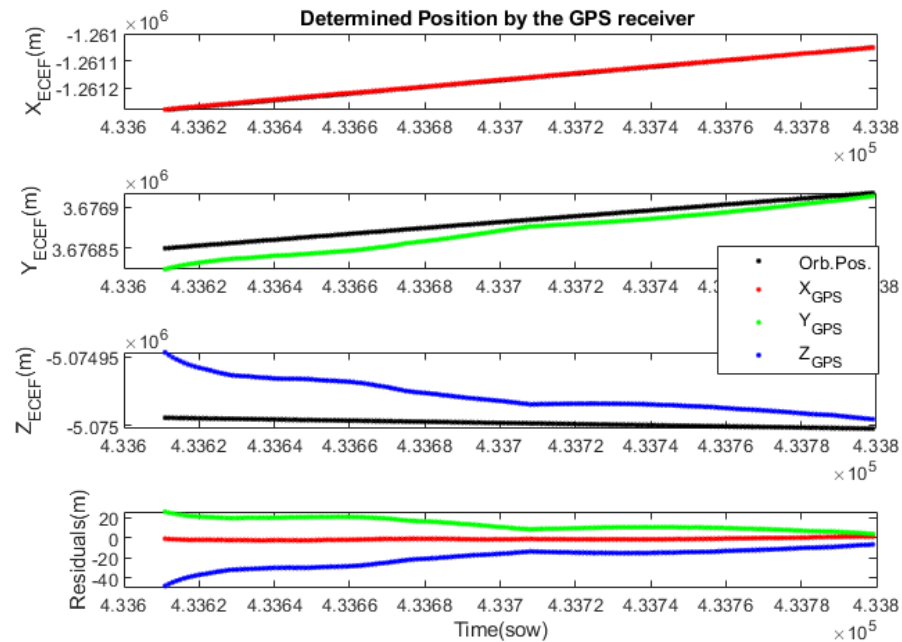


Figure 7. Position determined by the GPS receiver for the modified orbit. The ECEF coordinates of the propagated orbit are shown in black and those determined by the GPS in red, green and blue. The last graph shows the evolution of the residuals or errors.

Figure 8 left shows the difference between the pseudoranges used to transmit, ρ_{tx} , and those recorded by the GPS, ρ_{rx} . Theoretically, these should be coincident, but instrumental errors cause them to differ. It can be seen that the error evolves over time, but coincides for all GPS transmitters, as the lines overlap. This indicates that this error is mainly due to the GPS receiver, and specifically, it has been identified as an instrumental error associated with the Local Clock Bias (CB) of the receiver, which suffers from a drift. The GPS receiver estimates the CB, through its Trilateration algorithm. Figure 8 right shows the CB estimated by the GPS receiver, CB_{gps} , and the pseudorange errors expressed in seconds, CB_{pr} . It can be seen that there is a difference of 8 ms, which can be interpreted as an instrumental time delay of the GPS receiver.

This FW allows for characterising the instrumental delay of a GPS receiver, which is very difficult to do in flight because the GPS excitation signal is not available. In this process, the synchronisation of the signals from the transmitting stage and the signals from the receiving stage is critical. To solve this problem, the simulator time stamps each signal. This delay is critical for space applications, as the orbital velocities are very high. In the case studied, an instrumental delay of 8 ms has been detected, which would be an error of the order of 50 m under the conditions of a LEO orbit, with an orbital velocity of about 21,000 km/h.

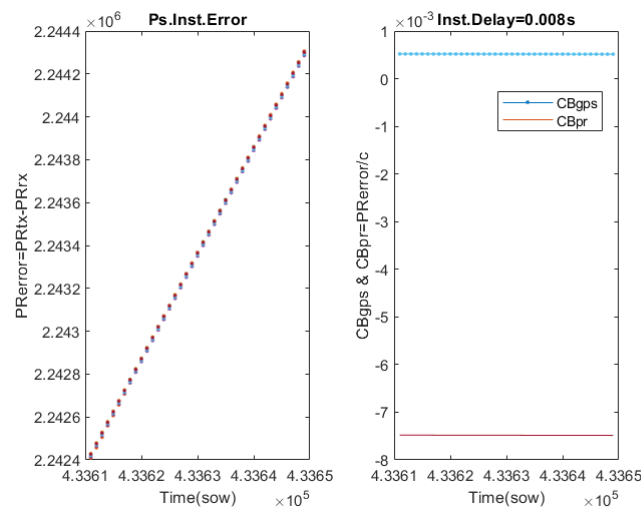


Figure 8. Estimation of the instrumental delay of the GPS receiver. The left plot shows the error between the pseudoranges used by the radio to transmit and those recorded by the GPS (each color represents a different GPS emitter). The right plot shows the Clock Bias estimated by the GPS and those calculated from the error between the pseudoranges.

6.2. Validation of GNSS Reception and Processing

To check that the FW allows the evaluation of other determination or trilateration algorithms, it is necessary to work at a lower level, using the raw signals recorded by the GPS receiver. The results presented below correspond to the LEO orbit shown above in Figure 3. For this orbit, the GPS receiver we are using does not determine the position, but provides raw signals from the observables, which are stored by the FW in the *RawData* file.

From the raw signals of the observables, the first step is to process the received pseudoranges to estimate the true distances, called the processed pseudoranges. This can be achieved using the formalism described in Section 3.1. In this observable processing stage, it is also possible to cancel those disturbances that are parameterised in the RINEX files. Figure 9 shows in the first two plots the raw pseudoranges, ρ_{rx} , and the processed ones, ρ_{proc} . Due to the scale they look similar, but they actually show significant differences. Figure 9 bottom shows the difference between the true distances and the processed pseudoranges. Although the thermal noise disturbance has not been included, a negligible noise can be observed which is due to the transmission-reception channel used. Thanks to all the corrections made, an acceptable error in the processed pseudoranges of less than 20 m is achieved. Ideally, these errors should be practically zero, but instruments are not perfect and make measurement errors. Because the signals from the GPS transmitters arrive at different times at the receiver, the temporal resolution of the receiver affects them differently. Due to the high orbital velocities, small temporal errors translate into errors of the order of metres. The developed FW has again highlighted the limitations of the GPS receiver used.

The emission times are calculated from the processed pseudoranges and the reception times. The orbital propagator then calculates the positions of the GPS transmitters. Then, a trilateration algorithm is applied to determine the orbital position of the S/C and its local clock bias. In this case, an extended version of the [20] algorithm has been implemented, which also estimates the clock bias.

Figure 10 shows the non filtered residuals of the S/C determined coordinates with respect to the propagated ones. The bottom graph shows the distance error associated with the determined local clock bias. Since the instrumental delay of the GPS receiver has been eliminated, the distance error associated with the clock bias is quite low. The noise is mainly due to receiver hardware limitations and transmission channel noise.

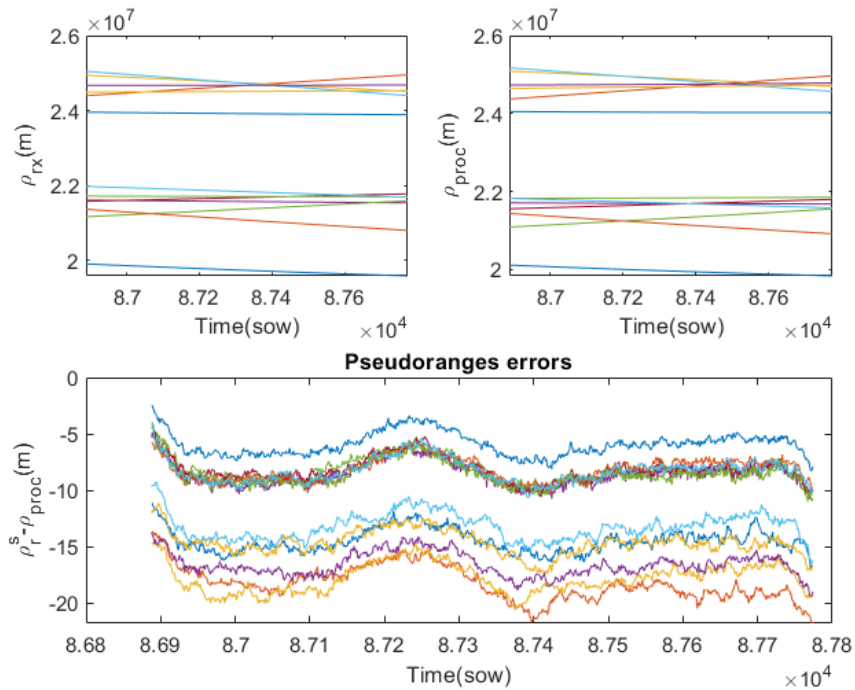


Figure 9. Received pseudoranges and processed pseudoranges and their errors (each color represents a different GPS emitter). **Top left** plot shows the pseudoranges measured by the receiver. **Top right** plot shows the processed pseudoranges. **Bottom** plot shows the difference between true distances and the processed pseudoranges.

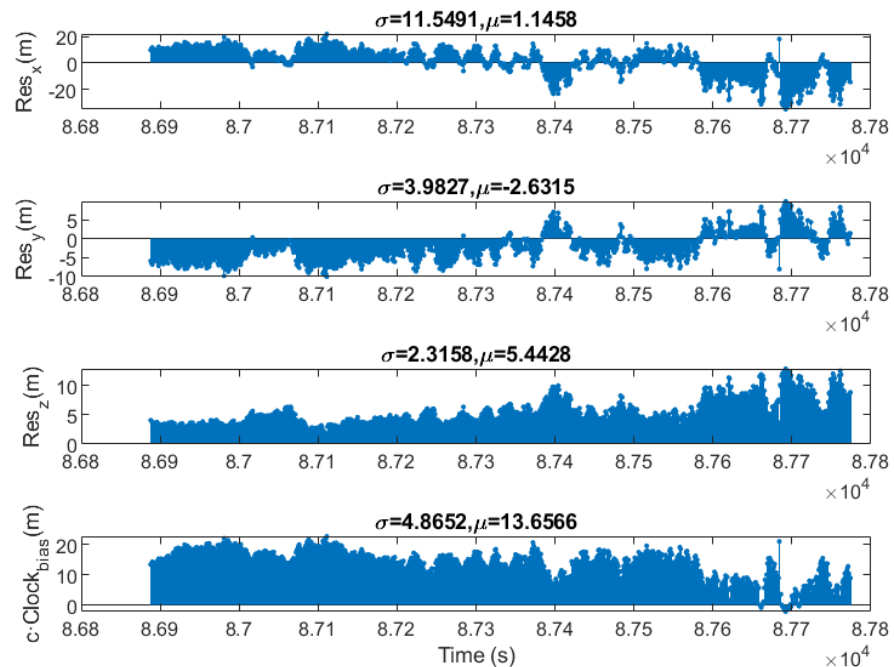


Figure 10. Residuals of the determined position by software for a LEO Orbit.

7. Conclusions

This paper presents a low-cost framework for hardware-in-the-loop orbit simulation. The framework allows validation of the integrated behaviour of the hardware and software involved in orbit position determination. It includes the simulation of perturbations due to ionospheric effects, tropospheric effects, thermal noise and instrumental delay of emitter and receiver, providing a realistic recreation of aerospace and hardware environments.

The framework uses as input information the spacecraft's orbital parameters, provided through a TLE file, and the GNSS constellation ephemeris data, provided through the RINEX file. From this information, the system calculates the GNSS constellation signals to be received by the spacecraft and, using a software-defined radio, transmits them to a GNSS receiver. The position determined by the GNSS receiver can be evaluated directly, thus employing the receiver as a black-box that directly calculates the position of the spacecraft during the simulation. In addition, a second way of using the framework gives access to the raw observables from the GNSS receiver and uses them to evaluate a particular determination algorithm.

Initial validation of all stages of the framework has been completed using black-box mode on the NEO-M8 commercial GNSS receiver connected in baseband to the software defined radio. The results obtained show that the framework works as expected with position determination accuracy in the order of the technical specifications of the receiver. Subsequently, tests have been completed using, this time, the raw observables provided by the receiver, thus validating the orbital position determination algorithm. The results show that the developed framework, although low-cost, allows the evaluation of orbital positioning systems at different levels. The environment is useful for verifying the performance of commercial GNSS receivers and checking the quality of their estimates. In this sense, it has been possible to verify that the NEO-M8 GNSS receiver, complying with the Wassenaar agreement, does not directly offer position determination outside the permitted limits, although it does allow access to the raw observables.

The framework provides a very useful tool for evaluating small satellite missions that require precise orbital position determination, such as those formed by satellite constellations, where in-flight formation and collision warning systems need to be managed. At orbital velocities, instrumentation errors significantly influence the position determined. Furthermore, how measurement errors are propagated to the determined position depends very much on the algorithm used and the configuration of the selected GPS satellites. The framework presented in this paper is ideal for qualifying the robustness of new algorithms against instrumental errors and external disturbances.

Furthermore, the modularity and adaptability of the developed environment make it easy to extend its use to other scenarios, such as redundant positioning systems that combine beacons fixed on the Earth's surface with information from the GNSS satellite constellations themselves. They also enable the simulation of the deployment of future navigation systems for the Moon or Mars, or even to provide coverage for deep space exploration missions.

Future work includes extending the capabilities of the framework to work in real time, and developing a GNSS receiver using a software-defined radio. This receiver will overcome the restrictions of commercial receivers, both in orbit height and spacecraft speed, and thus provide the positioning of the spacecraft in real environments, without the need to process raw observables. In addition, to complete the capabilities of the framework, it is planned to include simulation parameters related to space weather, such as radiation and pressure perturbations, and to develop other orbit determination algorithms.

Author Contributions: Conceptualization, methodology, resources and writing, D.F., S.E. and Ó.R.-P.; software D.F.; validation and formal analysis, S.E.; investigation D.F.; supervision Ó.R.-P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This research has been carried out in the laboratories of the Universidad Complutense de Madrid and the Universidad de Alcalá, for which the authors would like to thank both institutions for the use of their institutional resources.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADC	Analog to Digital Converter
BDS	BeiDou Navigation Satellite System
DAC	Digital to Analog Converter
DSSS	Direct-Sequence Spread Spectrum
ECEF	Earth-Centered, Earth-Fixed
ECI	Earth-Centered Inertial
FPGA	Field-Programmable Gate Array
FW	Framework
GNSS	Global Navigation Satellite System
GNC	Guidance, Navigation and Control
HIL	Hardware In the Loop
LEO	Low Earth Orbit
RF	Radio Frequency
RINEX	Receiver INdependent EXchange
S/C	Spacecraft abbreviated
SDR	Signal/Software Defined Radio
SPG4	Simplified General Perturbations Satellite Orbit Model 4
TEC	Total Electron Content
TLE	Two Line Element
TDI	Travelling Ionospheric Disturbances

References

1. Capuano, V.; Botteron, C.; Leclère, J.; Tian, J.; Wang, Y.; Farine, P.A. Feasibility study of GNSS as navigation system to reach the Moon. *Acta Astronaut.* **2015**, *116*, 186–201. [CrossRef]
2. Palmerini, G.B.; Sabatini, M.; Perrotta, G. En route to the Moon using GNSS signals. *Acta Astronaut.* **2009**, *64*, 467–483. [CrossRef]
3. Peng, Y.; Scales, W.A.; Edwards, T.R. GPS-based satellite formation flight simulation and applications to ionospheric remote sensing. *Navigation* **2020**, *67*, 3–21. [CrossRef]
4. Park, J.I.; Park, H.E.; Park, S.Y.; Choi, K.H. Hardware-in-the-loop simulations of GPS-based navigation and control for satellite formation flying. *Adv. Space Res.* **2010**, *46*, 1451–1465. [CrossRef]
5. Liduan, W.; Chuanrun, Z.; Wenrui, J.; Xingqun, Z. Enhanced GPS measurements simulation for space-oriented navigation system design. *Chin. J. Aeronaut.* **2010**, *23*, 438–446. [CrossRef]
6. Wang, K.; Chen, S.; Pan, A. Time and position spoofing with open source projects. *Black Hat Eur.* **2015**, *148*, 1–8.
7. Elango, G.A.; Sudha, G. Design of complete software GPS signal simulator with low complexity and precise multipath channel model. *J. Electr. Syst. Inf. Technol.* **2016**, *3*, 161–180. [CrossRef]
8. Peng, Y. GNSS-based Spacecraft Formation Flying Simulation and Ionospheric Remote Sensing Applications. Ph.D. Thesis, Virginia Tech, Blacksburg, VA, USA, 2017.
9. Ebinuma, T. GPS-SDR-SIM. Available online: <https://github.com/osqzss/gps-sdr-sim> (accessed on 9 January 2023).
10. Cao, J. Practical GPS Spoofing Attacks on Consumer Drones. Ph.D. Thesis, University of Hawai'i at Manoa, Honolulu, HI, USA, 2020.
11. Parkinson, B.W.; Stansell, T.; Beard, R.; Gromov, K. A history of satellite navigation. *Navigation* **1995**, *42*, 109–164. [CrossRef]
12. Hofmann-Wellenhof, B.; Lichtenegger, H.; Wasle, E. *GNSS—Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and More*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2007.
13. Husty, G. *Global Positioning System*; Delft University Press: Delft, The Netherlands, 2000.
14. Woo, E.L.; Brodie, G.H. *Uss kinkaid (dd 965) Hull Fouling Standardization Trials*; Technical Report; NNaval Surface Warfare Center Carderock Division: Bethesda, MD, USA, 1992.
15. Rodríguez de Haro, J. Análisis Software y Hardware del SDR HackRF One. Master's Thesis, Universidad de Granada, Granada, Spain, 2017. Available online: <http://hdl.handle.net/10481/48019> (accessed on 9 January 2023).
16. NEO-LEA/M8T. *U-blox M8 Concurrent GNSS Timing Modules*, R06; 2022. Available online: https://content.u-blox.com/sites/default/files/documents/NEO-LEA-M8T-FW3_DataSheet_UBX-15025193.pdf (accessed on 9 January 2023).
17. Vallado, D.; Crawford, P. SGP4 orbit determination. In Proceedings of the AIAA/AAS Astrodynamics Specialist Conference and Exhibit, Honolulu, HI, USA, 18–21 August 2008; p. 6770.
18. Brouwer, D. Solution of the problem of artificial satellite theory without drag. *Astron. J.* **1959**, *64*, 378–397. [CrossRef]

19. gnuradio.org. GNU Usage Manual. 2022. Available online: https://wiki.gnuradio.org/index.php?title=Usage_Manual. (accessed on 9 January 2023).
20. Norrdine, A. An Algebraic Solution to the Multilateration Problem. In Proceedings of the 15th International Conference on Indoor Positioning and Indoor Navigation, Sydney, Australia, 13–15 November 2012. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.