Design of a Time-Stamp subsystem for Geostationary satellites orbit determination and Earth observation

Author:
Josep Cerdà Roscar

Supervisor:
Antoni Broquetas Ibars

A Degree Thesis Submitted to the Faculty of the Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

In partial fulfilment of the requirements for the degree in Telecommunication Engineering

Barcelona, June 2018

Abstract

This project proposes and evaluates a solution to address the existing problem of time synchronization data in the context of precise orbit determination of geostationary satellites. For this reason, in this memory a review of the basics of an interferometer and the need of making this accurate time synchronization are done. Then, an introduction to time transference and the method that will be followed are explained as well as the proposal of a system design which fulfills the requirements. Furthermore, all the work that has been done in order to develop the time stamp system is expressed in this memory. Finally, the validation of this system using different clocks as time references and other experiments are explained.

Acknowledgments

First of all I have to show my gratitude to professor Antoni Broquetas Ibars who has guided me during this project and thanks to his advices this project has become true.

In addition, I have to thank Roger Martin Fuster, a former of the GEOSAR team and a PhD candidate, who has also guided me during this project and has given me a lot of advice. I would also like to thank all the employees of the laboratory of the department of Theory and Signal Communications. In particular, I would like to name Albert Martón who has helped me in the manufacturing of the hardware required.

Finally,I have to remark that this work has been financed by the Spanish Science, Research and Innovation Plan (MINECO) with Project Codes TIN2014-55413-C2-1/2-P and TEC2017-85244-C2-2/1-P.

Revision history and approval record

REVISION LIST

Revision	Date	Purpose
0	11/06/2018	Document creation
1	11/06/2018	Document revision
2	28/06/2018	Document revision
3	29/06/2018	Document revision
4	2/07/2018	Document revision

DOCUMENT DISTRIBUTION LIST

Name	E-mail
Josep Cerdà Roscar	josep.cerda.roscar@gmail.com
Antoni Broquetas Ibars	broquetas@tsc.upc.edu

Written by:		Reviewed and approved by:	
Date	11/06/2018	Date	11/06/2018
Name	Josep Cerdà Roscar	Name	Antoni Broquetas Ibars
Position	Project Author	Position	Project Supervisor

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Project motivation

Precise Orbit Determination of Geostationary telecommunication (GEO) satellites and forming radar images of the Earth surface based on Synthetic Aperture techniques both rely on accurate synchronization of the radar data collected by every antenna. To achieve this objective a Time Stamp subsystem must be designed in order to add the necessary time alignment information to the acquired data.

This project takes part of a research group, that is working on an interferometer for GEO satellites tracking, called GEOSAR. GEOSAR group also wants to use the data collected by the interferometer for processing of SAR images. The problem that appears when the interferometer is gathering data is that all data points must be marked by a time stamp defining the exact time in a standard format during which the data was acquired. High precisions when time stamping are needed in order to determine the orbit correctly. In order to create this Time Stamp subsystem, the project requirements are the following:

- Study the requirements of temporal radar synchronism in the context of GEOSAR data acquisition.
- In the error analysis realized in the context of developing the interferometer, it has been established that the time error requirement is in the order of 1 μs.
- Design a valid interface to high performance GPS receivers optimized for time/frequency

reference use.

- Propose a suitable protocol for inclusion of time stamp header at line level of the GEOSAR acquired data.
- Concept/Circuit validation and analysis.

1.2 Workplan

- **WP1.- Implementation of a permanent power supply system**
  - WP1.1.- Electrical design
  - WP1.2.- Optimal components research and validation
  - WP1.3.- Protoboard implementation
  - WP1.4.- PCB implementation
- **WP2.- Design and implementation of a time-stamp subsystem**
  - WP2.1.- Autonomous learning about data acquisition
  - WP2.2.- Design of the electrical system
  - WP2.3.- Components research
  - WP2.4.- Data acquisition of the GPS receiver
  - WP2.5.- Creation of a time-stamp software
  - WP2.6.- Integration of the whole system
- **WP3.- Data validation**
- **WP4.- Final Memory Writing**

The Gantt diagram can be found in Appendix D.

1.3 Memory structure

In this memory, a brief explanation about radio interferometry and orbit determination will be explained on Chapter 2 as well as the need of time synchronization on interferometry. Moreover, the time transference technique will be explained focusing on the

One-Way GPS time transfer, which will be the one that will be used in order to carry out this project.

Then, at Chapter 3, there will be explained all the steps that have been done so as to create the Time Stamp subsystem. In Chapter 4, the results and the validation of the system will be done. Finally, an approximate budget will be made in Chapter 5 and the conclusions will be exposed in Chapter 6.

Chapter 2

State of the art

In this chapter, the needed literature for understanding the problem statement as well as the adopted solution will be explained.

2.1 Interferometry for orbit determination

The main principle of interferometry is to combine signals generated from the same source at different locations in order to obtain information about the source. This techniques are carried out by interferometer devices. They use the superposition principle in order to obtain better output signal noise [1].

In order to use interferometry for orbit determination, the appropriate device would be an VLBI (Very Long Baseline Interferometer). It consists of a network of receivers, that would receive the signal and process it in order to obtain the phase difference between receivers. Once the data is acquired, it is processed with a complex correlation process where the phase difference is obtained. This phase will be later processed in order to determine the satellite position. The baseline separation determines the phase sensitivity to the satellite position and the ambiguity separation due to inherent cyclic nature of phase. However, this type of operation requires a coherent system across a whole continent which implies a complexity level and cost that are not affordable. For this reason, the GEOSAR team has implemented a low resolution pilot system, based on the VLBI principle but using a short baseline in order to ease the task of synchronizing

receivers [2]. As it is based in the VLBI principle, it has similar time synchronization problems that the ones that are tried to be solved.

In the process of acquisition in VLBI, the data is obtained by the receivers in different moment of time due to long distances between them. This delay can be modeled solving a geometry problem and should be corrected before the process of correlation [3].

$$L = B \cdot \cos(\beta) = c \cdot \tau_g \tag{2.1}$$

where,

- $\boldsymbol{L}$ Distance (m)
- $\boldsymbol{B}$ Baseline (m)
- $\boldsymbol{\beta}$ Angle between B and L
- $\boldsymbol{\tau_g}$ Geometric delay (s)
- $\boldsymbol{c}$ Speed of light (m/s)

Due to this delay and other related with the instrumentation and tropospheric and ionospheric delays, the signal acquisition must be very precise and clearly time marked. Therefore, the need to obtain independent and synchronized time references which can be used to coordinate the different interferometer nodes arises. The most frequently way to solve this problem is by using atomic clocks.

The atomic clocks are extremely precise clocks. They principle of operation is based on atomic physics. It uses the microwave signal that electrons in atoms emit when they change their energy levels [4, 5]. This atomic process allows to match the frequency of a crystal oscillator. They are the most accurate time and frequency standards. The most typical atomic clocks are the rubidium, the cesium and the hydrogen maser. In table 2.1 it can be seen the performance and stabilities that can be obtained with these clocks.

Table 2.1: **Atomic clock stabilities**

	Rubidium	Cesium	Hydrogen Maser
Stability at 1s	$5 \cdot 10^{-11} - 5 \cdot 10^{-12}$	$5 \cdot 10^{-11} - 5 \cdot 10^{-12}$	$\simeq 5 \cdot 10^{-14}$
Stability at 1 day	$\simeq 5 \cdot 10^{-12}$	$\simeq 5 \cdot 10^{-14}$	$\simeq 4 \cdot 10^{-15}$

In the low resolution pilot system, the delay time problem isn't as big as in the VLBI system because of the closer positioning of the receivers. However, the acquired data needs to be synchronized with an universal time reference as it needs in VLBI. In this case, the need for time synchronizing the data with such precision is because once the signal processing has been done, in order to determine the satellite orbit, there is the need to know at which precise moment the satellite was in that position. So the data needs to be synchronized with high precision and accuracy with a universal time reference. However, in both cases, there is a time transference problem that has to be solved.

2.2 Time transfer

Time transfer was born as a technique used for comparing different frequency standards. This methods allowed the creation of atomic clocks as well as the creation and the distribution of the time standards. It allowed the worldwide unification of time with the establishment of the International Atomic Time (TAI), the primary time regulation based on multiple atomic clocks spread over the world. It also was the basis for the creation of the Universal Coordinated Time (UTC), which was derived from TAI.

In this section, the main time transfer technique used for developing the time stamp system will be explained.

2.2.1 One-way GPS Time Transfer

The one-way time transfer technique consists on sending a signal to one or several receiver, which will use the transmitted signal as the reference for calibration. It's a simple method which requires having well calibrated the receivers, knowing the transmission channel and the propagation delay well dimensioned. If the propagation delay is known, this method can be mathematically modeled with this simple model:

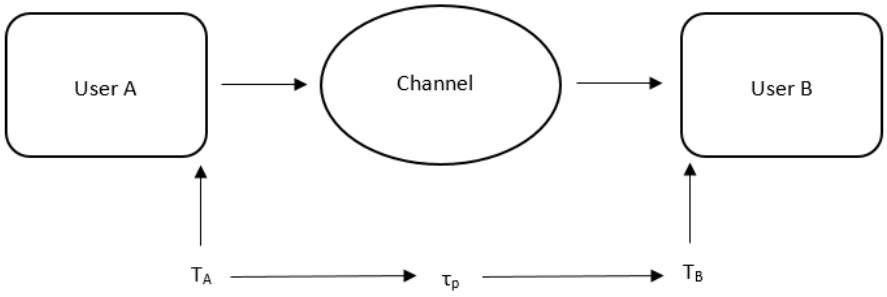

Figure 2.1: One-way time transfer method

$$\tau = \tau_p \tag{2.2}$$

$$T_A = T_B - \tau \tag{2.3}$$

where,

$\boldsymbol{T_A}$	Temporal mark in the transmission moment in A.
$\boldsymbol{T_B}$	Temporal mark in the reception moment in B.
$\boldsymbol{\tau_p}$	Delay of the tranmission between A and B.
$\boldsymbol{\tau}$	Temporal adjustment that needs to be done to the clock in B in order to synchronize it with the clock in A

There are several technologies that use this time transfer technique. However, the most common and worldwide used is the one One-Way GPS (Global Positioning System) time transfer technique.

The one-way GPS technique uses the signals obtained from a GPS receiver as the reference for calibration. The signals that come from the GPS are used in real time and no post processing of the measurement results is required [6]. In the one-way GPS time transfer technique, a user can access a globally available common time reference, UTC, by using only one receiver and taking advantage of the use of the GPS navigation message.

The GPS has always been known as a versatile, global tool for positioning and location. However, has also become the primary system for distributing time and frequency [7]. GPS satellites are fixtures in telecommunications networks and calibration and testing laboratories.

GPS satellites are controlled and operated by the United States Department of Defense (DoD). Some of their features are described in table 2.2.

Table 2.2: **GPS satellite features**

Satellite number	24 (4 in each of the 6 orbits)
Height	20200 Km
Tilt	55º from the equatorial plane
Period	11 h 58 min
Carrier frequencies	L1 = 1575,42 MHz L2 = 1227,6 MHz

The most interesting feature that isn't mentioned here is that every GPS satellite contains between three and four atomic clocks inside, which are coordinated by ground stations and referenced to UTC. This is what makes GPS as a primary tool for time distribution.

There are several types of GPS receivers used in time and frequency metrology. However, most share several common features. Some of this common features are

- They can track from 8 to 12 satellites in order to provide time and frequency signals derived from an average of all satellites in view
- They can provide time-of-day and date information in a computer readable format.
- Most provide a 1 Pulse Per Second (PPS) electrical output that can be easily synchronized to within 100 ns of UTC by entering a delay constant that compensates for the antenna, antenna cable and receiver delays.

Their time-of-day and date information is the one that will be used in order to create the high precision time stamp system. Although there are other GPS Time Transfer techniques that offer better levels of accuracies, as it can be seen in table 2.3, one-way Time Transfer technique is able to offer the accuracies needed in this project and its main advantage is that it doesn't require station-to-station communications between users and

other ground receiver system like GPS Common View [8]. For this reason, this technique is the one that fits better with the requirements and specifications of the desired Time Stamp system.

Table 2.3: **Typical Uncertainities of GPS Measurment Techniques**

Technique	**Timing Uncertainty**	**Frequency Uncertainty**
One-way	$< 20ns$	$2 \cdot 10^{-13}$
Single-Channel Common View	$\simeq 10ns$	$\simeq 1 \cdot 10^{-13}$
Multi-Channel Common-View	$< 5ns$	$< 5 \cdot 10^{-14}$
Carrier-Phase Common-View	$< 500ps$	$< 5 \cdot 10^{-15}$

Chapter 3

Methodology/project development

In this chapter, the process realized to make the time stamp subsystem is going to be explained as well as its integration inside the whole system.

In order to design the system, the first thing that was done was studying and analyzing some previous projects related with the creation of a GPSDO, a device that uses the GPS signals in order to control and synchronize the frequency of an oscillator [9, 10]

The GPS receiver used in the GPSDO was a LEA-M8T, from the brand U-Blox. Looking at its documentation carefully [11], it was seen that this GPS receiver was able to time stamp with high precision following some steps, so it was decided that it would be used in order to make the time stamp system.

3.1 High Precision Time Stamps

As commented before, the LEA-M8T can provide high precision time stamps which are synchronized with UTC time. In this section it will be explained how this receiver can deliver this time stamps.

This kind of GPS receivers developed by U-Blox have its own message protocol, called UBX. They also use the typical GPS protocol (NMEA), which is complemented with the UBX protocol. In order to acquire time stamps, the UBX protocol is the one that will be used because it has a message called UBX-TIM-TM2, which has information for high

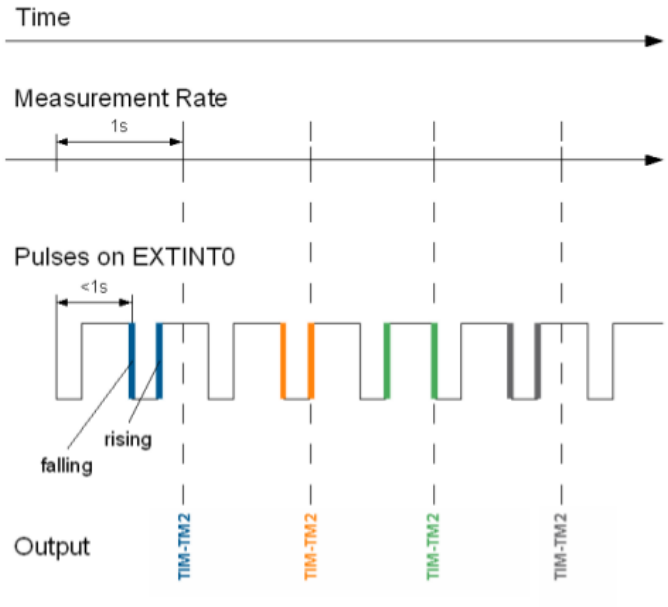

Figure 3.1: UBX Time Stamp messages chronogram

precision time stamping.

The LEA-M8T, will output a time stamp message if this steps are followed:

- Enabling the UBX-TIM-TM2 message (by default they aren't enabled).
- If the Time Stamps have to be synchronized with UTC, a previous configuration messaged called UBX-CFG-TP5 has to be sent to the GPS receiver.
- A rising or falling flank has to be detected in one of the external interruption channels that the LEA-M8T has.

If these steps are followed, then the GPS receiver will output the Time Stamp message, as it can be observed in the figure 3.1

3.2 System proposal

Once it is known how to obtain high precision time stamps, the system proposal was made. Our system proposal was the following:

(Esquemàtic amb les part del sistema i com quedaria integrat en l'interferòmetre).

- A micro controller that is able to communicate with the LEA-M8T and parsing the messages specified previously. Moreover, it has to be able to send the time stamps to the interferometer subsystem that will store the collected data.
- A power supply board for the GPS receiver as well as the micro controller.

3.2.1 Power Supply Board

As commented previously, the GPS receiver used was part of a GPSDO project. In that project, the receiver was soldered into a small board made to fit with the GPSDO design. Therefore, in order to take advantage of the old design, a Power Supply board was created with the objective of giving stability and power supplying to the GPS receiver.

Looking at the electrical specifications of the LEA-M8T[12], the supply voltage needed was 3 V and with a maximum current consumption of 67 mA. For this reason, the power supply board has been designed taking into account this specifications as well as to fit well with the GPS receiver board. The board design as well as its schemetic are represented in figures 3.2 and 3.3.

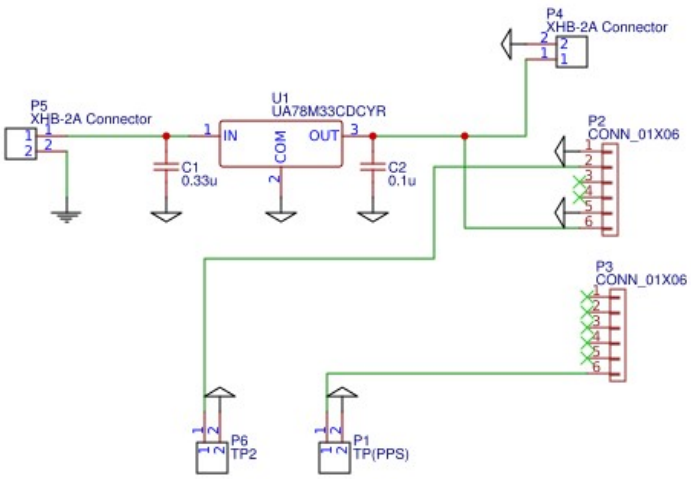

Figure 3.2: Power Supply schematic design

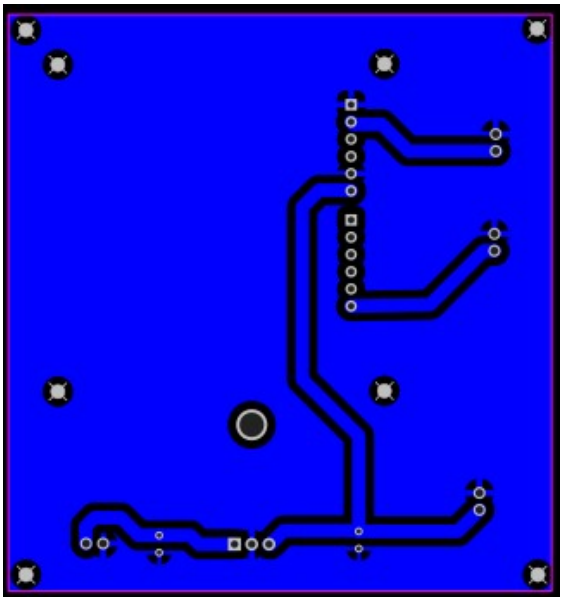

Figure 3.3: Power Supply PCB design

As it can be observed, this board has 4 holes in the middle in order to support the GPS receiver board. The connexions between the two boards are made using the 6 pin strips there are.

The typical supply voltage as said before is 3V. However, voltages between 2,7 and 3,6 V are well accepted by the LEA-M8T. As 3,3 V voltage regulators are more common and easy to find, it was decided to use a UA78M33C, which outputs 3.3V if a voltage between 5,3 and 25 V is inputed [13].

Finally, some polarized connectors have been placed in order to supply the board with an external power supply and to obtain some important signals of the LEA-M8T, which are the time pulses. In figure 3.4 the final implementation can be observed and how it fits with the GPS receiver board.

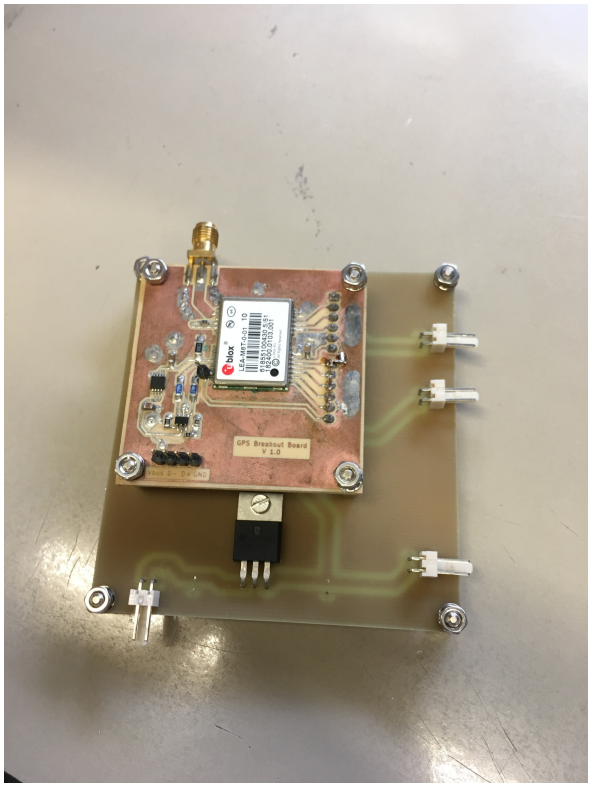

Figure 3.4: Final implementation with the GPS receiver board

3.2.2 Communication with LEA-M8T

In this section all the steps followed in order to receive, read and send the UBX-TIM-TM2 message will be explained. First of all, it had to be decided the micro controller which would be communicating with the LEA-M8T. The micro controller that was chosen was the Atmel ATMega32U4 [14] because it is available on an evaluation board of the firm SparkFun. This evaluation board is called an Arduino ProMicro [15] because it has the same functionality than an Arduino board. This board was chosen due to some factors:

- Its size, which would be important when integrating the system into the interferometer.
- The voltage logic levels of the GPS recevier, which are the same as the supply voltage.
- Allows two Serial communications at the same time.

Then, the type of communication used in order to receive and send information between both devices was established. This GPS receiver allows different types of communication (UART, SPI, USB, DDC). It was decided that an UART communication would be established. The UART communication was selected because it allows error checking using the parity bits, data transmission with only two wires and it doesn't need a clock signal for synchronization. Its configuration is listed below:

- Baud rate: 19200.
- Databits: 8
- Stopbits: 1
- Parity: Yes
- Bit order: Most Significant Bit First

It was decided that the baud rate of 19200 would be a suitable rate so the ProMicro would have the enough time for reading the delivered data. Then, in order to control communication errors the parity bits were activated.

3.2.2.1 LEA-M8T configuration

In order to establish the communication between the ProMicro board and the LEA-M8T, this one had to be well configured to establish an UART communication as well as providing the correct and desired time stamp message.

U-Blox has developed a computer software that allows users to communicate in an easy and visual way with its GPS receivers via USB called U-Center. As the board designed in the GPSDO project for the LEA-M8T allows the connection via USB, the device was configured by this way.

U-center is a really powerful tool that can be used for many purposes, which are explained in its manual [16]. However, in this project it will only be used for configuring the GPS receiver.

When connecting the device to the computer, in U-Center it has to be indicated in which COM port the receiver is connected.

Then, in order to configure the device, the UBX protocol has a list of configuration messages which will be used to configure the LEA-M8T in a correct way. In U-Center, the configuration menu allows to send these configuration messages.

The first thing to configure was allowing the UART communication with the features mentioned before. Therefore, once the configuration menu is opened, the UBX-CFG-PRT with its fields filled as in figure 3.5 has to be sent. It can be appreciated that in this message the type of protocol that will be outputted is chosen, and that only the UBX protocol is enabled in this configuration.

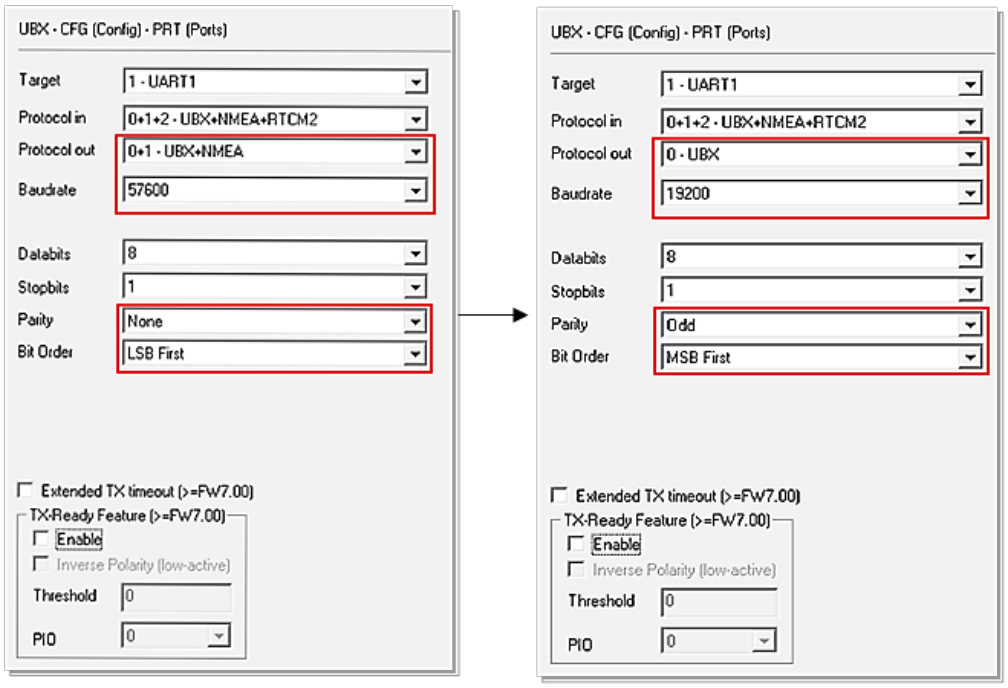

Figure 3.5: UBX-CFG-PRT Message

Then, the next thing to configure is enabling the message UBX-TIM-TM2 to be outputted. So, the configuration message that allows this is the UBX-CFG-MSG, as it showed in figure 3.6.

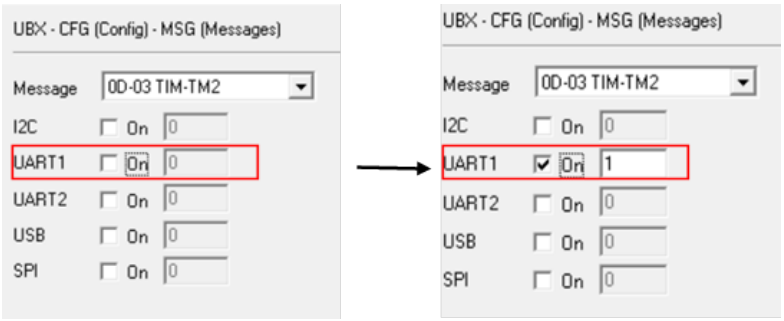

Figure 3.6: UBX-CFG-MSG Message

Now the GPS receiver will be configured so when a rising or falling edge arrives onto one of the two external interruption channels that it has, it will output the UBX-TIM-TM2 message by the UART pins. However, nothing related with the time reference synchronization has been mentioned here. This is due to the UBX-CFG-TP5 by default is configured with UTC.

Finally, the LEA-M8T has a flash memory that can store the actual configuration inside, so when it is turned off, at restarting the same configuration will be applied. In order to store the actual configuration the message UBX-CFG-CFG has to be sent, as showed in figure 3.7.

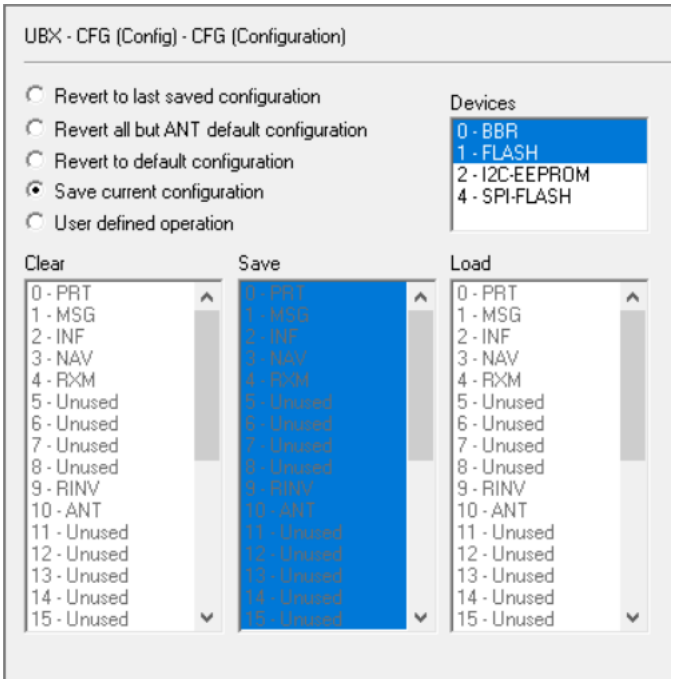

Figure 3.7: UBX-CFG-CFG Message

3.2.2.2 Parsing the time stamp messages

Once the LEA-M8T has been configured and it is ready to output the UBX-TIM-TM2 messages, it will be needed some software function for parsing the time stamp message that it will be provided.

The UBX-TIM-TM2 message has the structure showed at figure 3.8. Knowing this structure and how the UBX protocol works (figure 3.9 a software program, which would be loaded to the micro controller, that detects when there is a new UBX-TIM-TM2 message and parses it was created.

31.21.6 UBX-TIM-TM2 (0x0D 0x03)

31.21.6.1 Time mark data

Message	**TIM-TM2**					
Description	**Time mark data**					
Firmware	Supported on: • u-blox 8 / u-blox M8 from protocol version 15 up to version 23.01					
Type	Periodic/Polled					
Comment	This message contains information for high precision time stamping / pulse counting. The delay figures and timebase given in CFG-TP5 are also applied to the time results output in this message.					
	Header	*Class*	*ID*	*Length (Bytes)*	*Payload*	*Checksum*
Message Structure	0xB5 0x62	0x0D	0x03	28	*see below*	CK_A CK_B

Payload Contents:

Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Description*
0	U1	-	ch	-	Channel (i.e. EXTINT) upon which the pulse was measured
1	X1	-	flags	-	Bitmask (see graphic below)
2	U2	-	count	-	rising edge counter.
4	U2	-	wnR	-	week number of last rising edge
6	U2	-	wnF	-	week number of last falling edge
8	U4	-	towMsR	ms	tow of rising edge
12	U4	-	towSubMsR	ns	millisecond fraction of tow of rising edge in nanoseconds
16	U4	-	towMsF	ms	tow of falling edge
20	U4	-	towSubMsF	ns	millisecond fraction of tow of falling edge in nanoseconds
24	U4	-	accEst	ns	Accuracy estimate

Figure 3.8: UBX-TIM-TM2 message structure

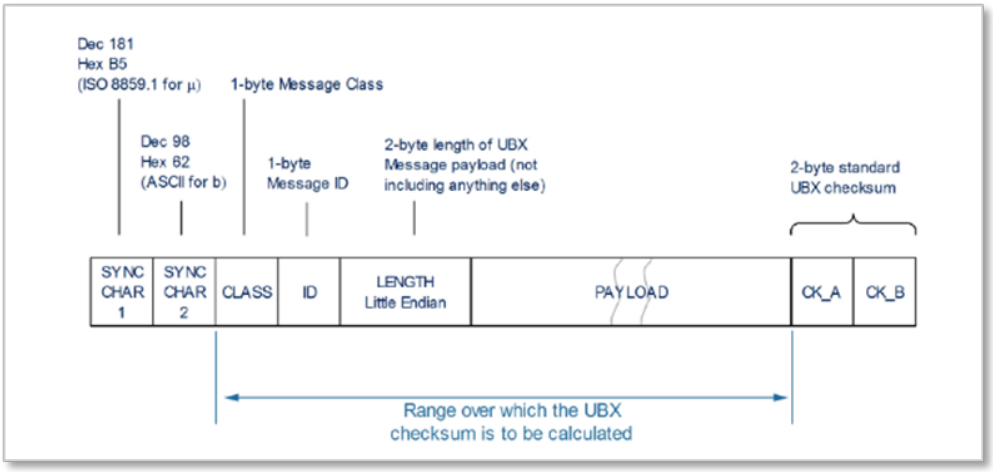

Figure 3.9: UBX Protocol Structure

This program can be found in Appendix A and it does the following:

- There is an struct with the same variables there are in the message UBX-TIM-TM2.

- The function that reads and parses the data arriving from the GPS receiver is the *processGPS()*. Its function is:
  - It checks whether the first bytes match with the UBX header.
  - If they match, the it places every byte in the correct memory position, in order to complete every variable of the truct.
  - It calls the *calculateChecksum()* function, which calculates a priori which will be the checksum of the message.
  - Finally, if the calculated checksum matchs with the received one, it returns true.
- Every *loop()* in the ProMicro board software, there is a call to the *processGPS()* function. If it has returned true, a new Time Stamp message has been read. Then, the desired variables of this message, which are the time of the week and the fraction time of the week of the detected rising edge are processed and sent to the data storing subsystem of the interferometer.

When loading this program to the Arduino, connecting it to the LEA-M8T, as showed in figure 3.10 and watching the results in the Serial monitor, it can be observed that the messages are well parsed as showed in figure 3.11.

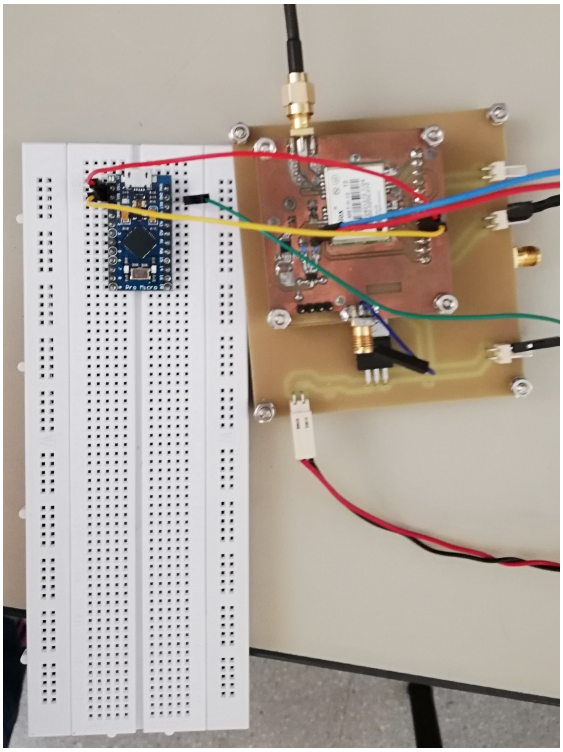

Figure 3.10: Connections between ProMicro and LEA-M8T

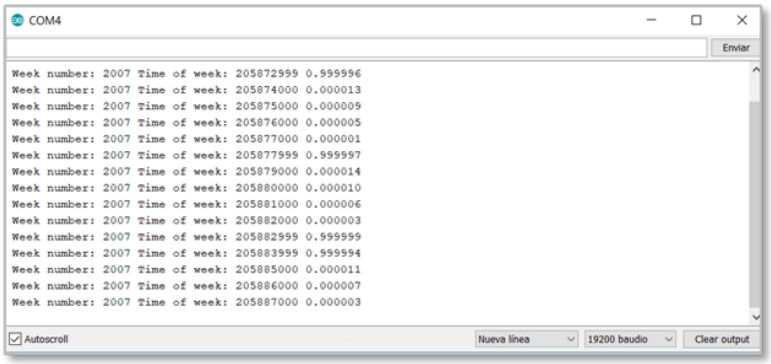

Figure 3.11: Arduino Serial Monitor timestamps

3.3 System integration

Once the designed system had been developed, and its correct functioning was proved, it was time to integrate it into the interferometer. For this reason, the integration design that was proposed is shown below as well as a block diagram showing how it would be integrated.

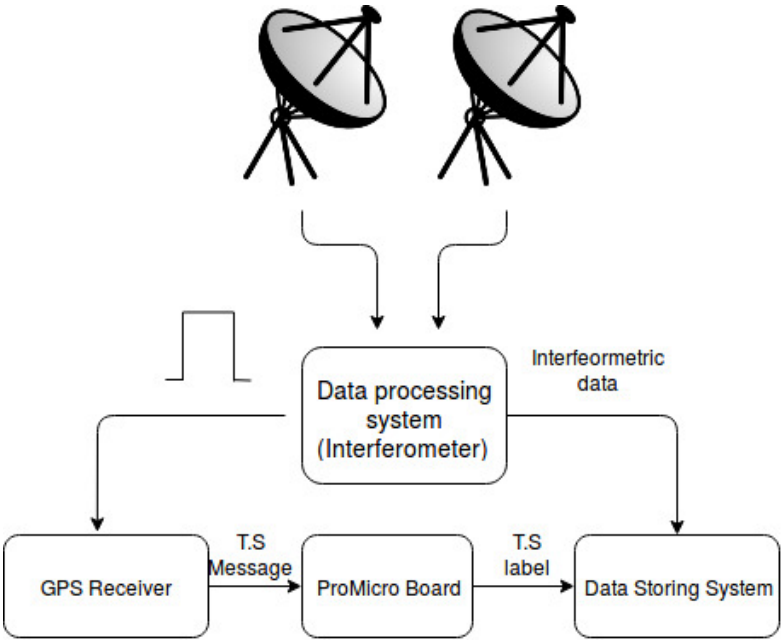

Figure 3.12: Block diagram of the system integration into the interferometer

- The interferometer collects data during some timeand samples it at a 1kHz rate. For this reason, it has been accorded that every 1000 of samples, a time stamp label will be requested.
- The interferometer subsystem which has the collected samples, has to generate a pulse which will be the one that will go to the interruption channel of the GPS receiver.
- Then, once the pulse has been registered and time stamped by the GPS receiver, it will be sent to the ProMicro board which will process the time stamp message.
- Finally, the ProMicro board, using a Serial connection to send the time stamp label, will send this information to the interferometer subsystem which will store

the gathered data to time stamp.

This integration was proposed because it was thought that it would be an strong system due to the fact that when the interferometer subsystem wants to time stamp a collection of collected data, it sends directly the pulse to the GPS receiver interruption channel. By this way, it is guaranteed that the pulse will be time stamped in the precise moment it was generated, and it won't suffer time delays. Moreover, the communication between the ProMicro board and the interferometer subsystem was decided that would be via USB because it would allow connecting the ProMicro into the interferometer system without any complications, allowing to use the same software program which was created to monitor the time stamps into the computer and that is in Appendix A.

Chapter 4

Results

Once the system has been created, its precision and accuracy had to be tested. For this reason, several experiments have been done in order to know whether the precisions we could acquire are the ones needed for the time stamp system.

In order to analyze the precisions of this system we have used different clocks as pulse generators connected to the interruption channel. By doing this, the precision and the accuracy of the clock references has been analyzed as well as the level of precision of the time stamps we have.

The time stamps processed by the ProMicro board, are sent to a computer via the Serial port. When they arrive, using a Matlab script created to capture the data arriving to the Serial Port of the computer, showed in Appendix B, the data with the Time Stamps is stored into a text file in order to be lately processed.

In every experiment we have captured 360 time stamp samples. In order to analyze the precisions and the accuracy it has been created an other Matlab script,showed in Appendix C, which creates an array of the subtraction between consecutive time stamps. Once this array is created, the mean time between samples and the standard deviation are represented.

4.1 ProMicro clock as time reference

The first clock analyzed in order to see its precision and exactitude in the time stamp system was the ProMicro board clock. In this experiment, the objective was to analyze whether the same ProMicro could be used at the same time for reading the time stamps arriving from the GPS receiver and as the pulse generator for times tamping. In this order, in the *loop()* function, a PPS signal was emulated in order to analyze its exactitude and precision. The results are shown in figure 4.1

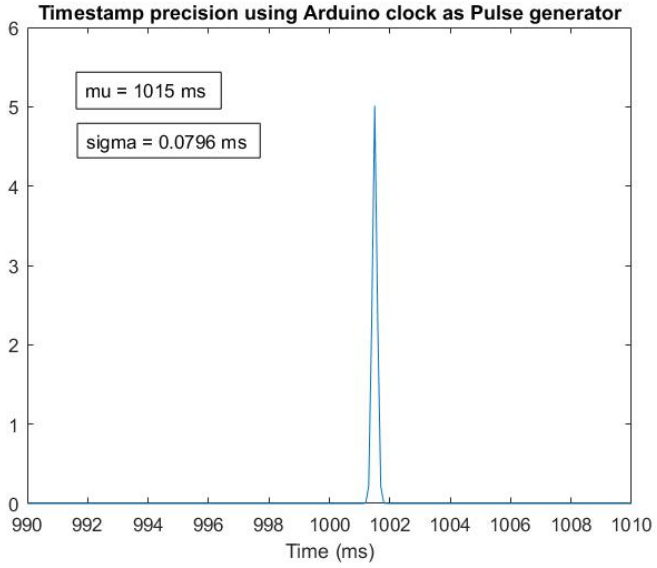

Figure 4.1: Precision and accuracy of the Arduino clock as the pulse generator

As it can be observed, the time precision of this clock is 79.6 μs. It is thought that this level of precision between samples is due to the fact that the micro controller time counter is dealing with different tasks and it isn't related to the precision of the internal clock it has. Furthermore, the mean time between samples is shifted 15 ms because of the processing of that time stamp messages. Therefore, we can conclude that using the ProMicro board for both tasks wouldn't be a good idea as it has been seen that it would provoke undesired time drifts.

4.2 PPS from LEA-M8T

Then, the PPS clock of the same GPS receiver, which is an electrical output that most GPS receivers have and consists on a one pulse per second signal, was analyzed. This experiment was made to ensure that the system created had the precision and exactitude desired in the specifications and that are supposed using this system. The results are exposed in figure 4.2

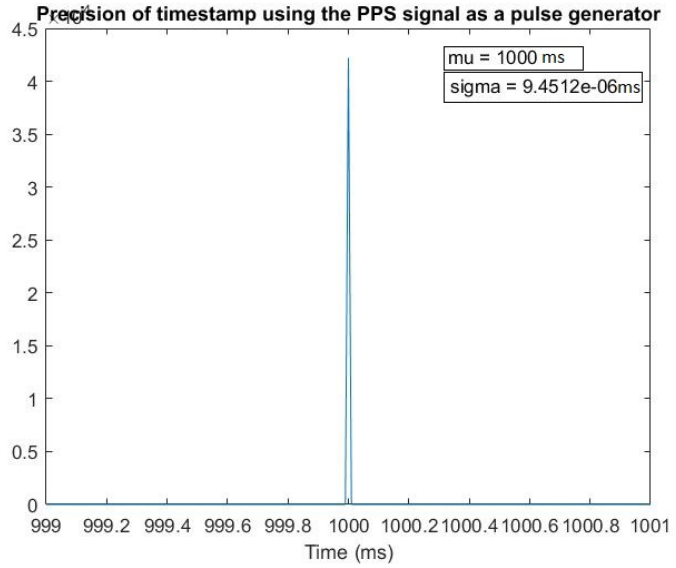

Figure 4.2: Precision and exactitude of the PPS of the LEA-M8T

Figure 4.2 corroborates what it was expected. It can be seen that there aren't practically time drifts between consecutive Time Stamps. The precision of this system, which is evaluated by doing the standard deviation of the samples taken, is of 9.4512 nanoseconds, which is even better than the required by the specifications.

4.3 PPS from 2nd LEA-M8T as time reference

Finally, the last clock used was the PPS from a second LEA-M8T there was in the laboratory. This was a form to confirm the results obtained in the previous experiment as well as to validate the time stamp high precisions. Moreover, this validation is more

decisive because here the pulse generator is an external clock which isn't synchronized with the device, as it was in the previous experiment. The results obtained of the experiments are summarized in figure 4.3.

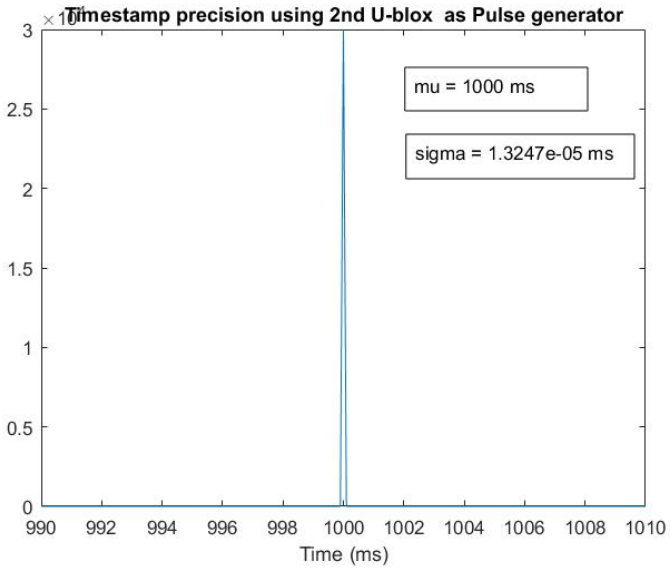

Figure 4.3: Precision and accuracy of the PPS of an external GPS receiver (LEA-M8T too)

As we can observe in the graphic above, using the PPS of a second GPS receiver, the mean time between samples is exactly one second and its deviation is about 13 ns. Therefore, it can be affirmed that the precision of the PPS signal as a pulse generator is very high and fits the specifications of the system really well.

4.4 Detection of small drifts

Finally, in order to validate the high precision of the system when time stamping, it was verified that it could detect delays or drifts in the order of nanoseconds. Therefore, to confirm this, a coaxial cable of 7.2 m has been connected between the PPS signal of the LEA-M8T and the interruption pin as it can be seen in figure 4.4. Then, supposing that the time stamps given when connecting the PPS signal of the same GPS receiver to the interruption channel will be exactly every second, which is practically true as it has

been demonstrated in the experiment before connecting the coaxial cable, and knowing a priori the delay that the coaxial cable will introduce, it can be verified whether the theoretical delay coincides with the practical one.

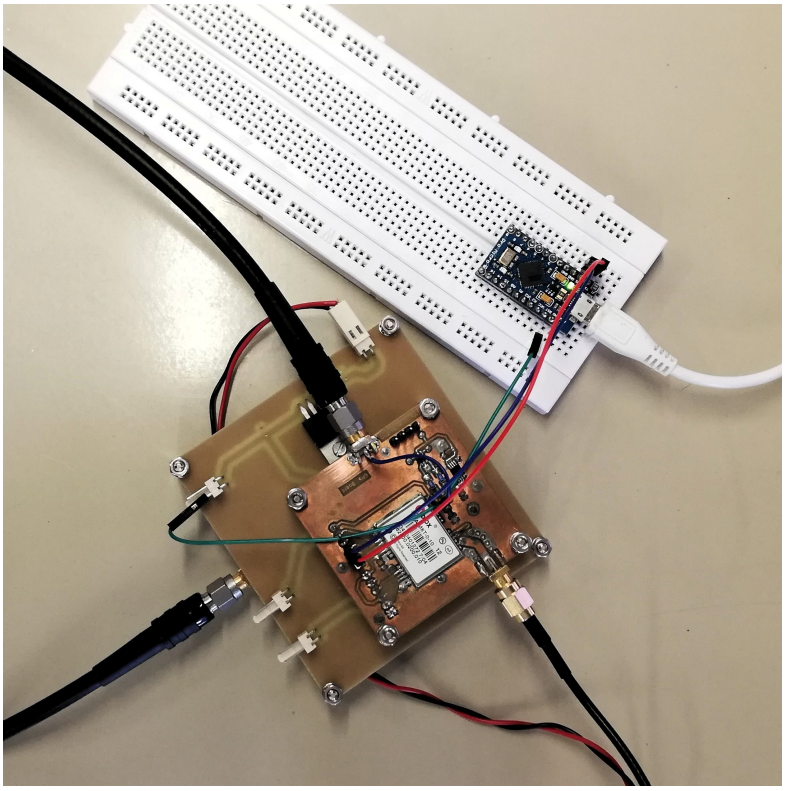

Figure 4.4: Coaxial connexion between the PPS signal and the external interruption channel

In order to calculate the theoretical delay, it has to be known which is the dielectric constant of the type of coaxial cable we are using as well as its longitude. The coaxial cable used is an RG-223 and its longitude is 7.2 m. Then, looking at the datasheet [17], it is seen that the dielectric constant of this type of cable is $\varepsilon_r = 2.29568$. Therefore, the delay will be:

$$t = \frac{d}{v} = \frac{d * \sqrt{\varepsilon_r}}{c} = \frac{7.2 * \sqrt{2.29568}}{3 * 10^8} = 36.3636ns$$

Then, using the Matlab script in order to capture the time stamps, the mean of the

nanoseconds which every time stamp has is made. The calculated mean when taking 360 samples is the following:

$$\mu = 45.483ns$$

This is the mean of the nanoseconds delay per Time Stamp. It can be observed that this mean is higher than the delay a priori. However, this is a logical thing knowing that the time precision of the system is about 10 ns, which it can also be observed in this calculation.

Chapter 5

Budget

In this section an estimation of the costs that would report making this project on a company or in a professional environment will be done. This project has been carried out in the laboratory of the Department of Theory of Signal and Communications of the ETSETB. it has to be remarked that without the resources they have there, it would be more difficult to complete this project. It has to be remarked that the cost of the board of the GPS receiver and all the components it has, has been taken of the previous Final Degree Project [9] and it has been taken into account at the prototype costs.

Table 5.1: **Project Budget**

Prototype Costs

Name	Price x Unit	Units	Total
GPS Board			
GPS receiver (LEA-M8T)	89,00 €	1	89,00 €
Antenna(ANT555)	15,00 €	1	15,00 €
Voltage Regulator(ADP3335)	2,92 €	1	2,92 €
ESD Protection(USBLC6-2)	0,10 €	1	0,10 €
Capacitors	0,01 €	10	1 €
Board Connectors	0,40 €	1	0,40 €
Power Supply Board			
Voltage Regulator(UA78M33C)	0,681 €	1	0,681 €
Capacitors	0,01 €	2	0,02 €
Board Connectors	0,40 €	4	1,60 €
System rest			
Arduino Pro Micro 3,3/8MHz	17,111 €	1	17,111 €
Coaxial Cable (RG-223)	1,72 €/meter	7,2 m	12,384 €
Software			
Matlab 2016 license	-	-	119,00 €
Working Costs			
Name	Price x Hour	Hours	Total
Junior Engineer	8,00 €	25 h/week · 21 weeks = 525 h	4200 €
		Total	**4459.216**

Chapter 6

Conclusions and future development

The main goal of this project was having a precise Time Stamp subsystem in order to have the required time alignment in the data acquired from the interferometer.

In the Introduction the accuracy specifications of the system were stated. It had to be a Time Stamp system synchronized with UTC which required accuracies below 1 us. It was seen in the State of the Art that if the Time Stamp had to had such accuracy levels, the best way to make the Time Stamp system was by GPS receivers, due to two factors:

- Each GPS satellite has multiple atomic clocks inside, so they GPS signals can contain really accurate time data.
- The capability of having a time reference and a really accurate Time Stamp system at an economic prize.

Then, in the Methodology section, the steps followed in order to build this system are explained. Once the system was created, it needed to be validated some way. Due to the difficulty of validating a time system which is more accurate that all the time references available in the laboratory, there have been some experiments involving a second GPS receiver or the same PPS signal because these signals are the best time references it can be found in the laboratory in order to validate the system. In my opinion, as it can be seen in the Results chapter, the main goal of this project that was having a Time Stamp system with precisions under the microsecond has been accomplished. However, there is still future work that it can be done in order to improve the system quality.

In order to keep improving the system, the antenna and the RF delays can be measured and calculated. This, as it has been explained in the Introduction, isn't a matter for Orbit Determination, but would be helpful because this delays will make that the position of the GEO satellite will be determined with a few meters of delay, which then by signal processing can be corrected. Nevertheless, having this delays measured, the GPS receiver would take them into account when giving the time synchronized with UTC and it wouldn't be necessary post signal processing.

Finally this project allowed me learning new topics related with time synchronization and GPS receivers and I think it has been a satisfactory final degree thesis in terms of learning and working.

Appendix A

Arduino Parsing Code

```
//Header of the UBX messages
const unsigned char UBX_HEADER[] = { 0xB5, 0x62 };

//This is the timestamp message structure
struct TIM_TM2 {
  unsigned char cls;          //Class message
  unsigned char id;           //ID of the message
  unsigned short len;         //Payload length
  unsigned char ch;           //Chanel where pulse was detected
  unsigned char flags;        //Bitmask
  unsigned short count;       //Rising edge counter
  unsigned short wnR;         //Week number of last rising edge
  unsigned short wnF;         //Week number of last falling edge
  unsigned long towMsR;       //Time of Week of rising edge (ms)
  unsigned long towSubMsR;    //Millisecond fraction of tow of rising edge in nanoseconds
  unsigned long towMsF;       //Time of Week of last rising edge (ms)
  unsigned long towSubMsF;    //Millisecond fraction of tow of falling edge in nanoseconds
  unsigned long accEst;       //Accuracy estimate
};

TIM_TM2 tim_tm2;

//8-bit Fletcher algorithm for checksum calculation
void calcChecksum(unsigned char* CK) {
  memset(CK, 0, 2);
  for (int i = 0; i < (int)sizeof(TIM_TM2); i++) {
    CK[0] += ((unsigned char*)(&tim_tm2))[i];
    CK[1] += CK[0];
  }
}
```

Figure A.1: Arduino parsing code(Part 1)

```
bool processGPS() {
  static int fpos = 0;
  static unsigned char checksum[2];
  const int payloadSize = sizeof(TIM_TM2);

  while ( Serial1.available() > 0 ) {
    byte c = Serial1.read();
    if ( fpos < 2 ) {
      // For the first two bytes we are simply looking for a match with the UBX header bytes (0xB5,0x62)
      if ( c == UBX_HEADER[fpos] )
        fpos++;
      else
        fpos = 0; // Reset to beginning state.
    }
    else {
      // If we come here then fpos >= 2, which means we have found a match with the UBX_HEADER
      // and we are now reading in the bytes that make up the payload.

      // Place the incoming byte into the ubxMessage struct. The position is fpos-2 because
      // the struct does not include the initial two-byte header (UBX_HEADER).
      if ( (fpos-2) < payloadSize )
        ((unsigned char*)(&tim_tm2))[fpos-2] = c;

      fpos++;

      if ( fpos == (payloadSize+2) ) {
        // All payload bytes have now been received, so we can calculate the
        // expected checksum value to compare with the next two incoming bytes.
        calcChecksum(checksum);
      }
      else if ( fpos == (payloadSize+3) ) {
        // First byte after the payload, ie. first byte of the checksum.
        // Does it match the first byte of the checksum we calculated?
        if ( c != checksum[0] ) {
         // Checksum doesn't match, reset to beginning state and try again.
          fpos = 0;
        }
      }
```

Figure A.2: Arduino parsing code (Part 2)

```
      ,
      else if ( fpos == (payloadSize+4) ) {
        // Second byte after the payload, ie. second byte of the checksum.
        // Does it match the second byte of the checksum we calculated?
        fpos = 0;  // We will reset the state regardless of whether the checksum matches.
        if ( c == checksum[1] ) {
          // Checksum matches, we have a valid message.
          return true;
        }
      }
      else if ( fpos > (payloadSize+4) ) {
        // We have now read more bytes than both the expected payload and checksum
        // together, so something went wrong. Reset to beginning state and try again.
        fpos = 0;
      }
    }
  }  return false;

}
void setup() {
 Serial.begin(9600);
 Serial1.begin(19200; //This will be the Serial used to communicate with the LEA-M8T
}
void loop() {
   if ( processGPS() ) {
      double nanos = (double)tim_tm2.towSubMsR / 1e6;
      double timestamp = tim_tm2.towMsR + nanos; //The timestamp is unified in a unique variable
      Serial.print(timestamp, 6); //The timestamp is printed with 6 decimals of precision (nanoseconds)
      Serial.println();
   }
}
```

Figure A.3: Arduino parsing code (Part 3)

Appendix B

Matlab Script for reading Time Stamp Data

```
%2 column data reading script
clear all
clc
delete(instrfindall);
arduino = serial('COM4', 'BaudRate', 9600);
fopen(arduino);

for i=1:60
    serial = fscanf(arduino);
    flushinput(arduino);
    t = strsplit(serial, '\t');
    milis(i) = str2double(t(1));
    nanos(i) = str2double(t(2));
end

fclose(arduino);

fid = fopen('PPS&Coax3.6.txt', 'w');
formatSpec = '%f %f\n';
fprintf(fid, formatSpec, milis, nanos);
```

Figure B.1: Matlab Script for reading Time Stamp Data

Appendix C

Calculation and representation of Time Stamps precisions

```
%Precision calculation and representation
tow = milis + nanos;
for i=1:59
    z(i) = tow(i+1)-tow(i);
end

mu = mean(z);
sigma = std(z);
x = 990:0.1:1010;
f1 = fd_normal(x, mu, sigma);
plot(x, f1);
xlabel('Time (ms)');
title('Timestamp precision using PPS connected with 7.2m of coaxial cable');
```

Figure C.1: Calculation and representation of Time Stamps precisions

Appendix D

Gantt Diagram

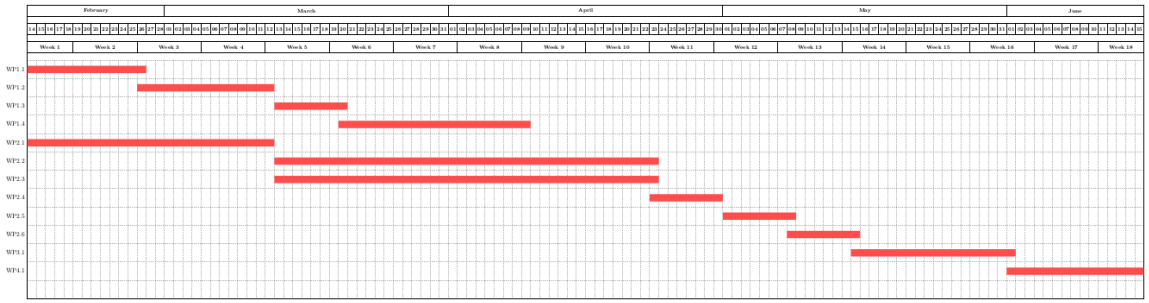

Figure D.1: Gantt diagram

Glossary

DDC Display Data Channel

DoD Department of Defense

GEOSAR Geostationary Synthetic Aperture Radar

GPS Global Positioning System

GPSDO GPS Disciplined Oscillator

NMEA National Marine Electronics Association

PCB Printed Circuit Board

PPS Pulse Per Second

SPI Serial Peripheral Interface

TAI International Atomic Time

UART Universal Asynchronous Receiver-Transmitter

USB Universal Serial Bus

USNO United States Naval Observatory

UTC Coordinated Universal Time

Bibliography

[1] A. R. Thompson, J. M. Moran, G. W. Swenson, *et al.*, *Interferometry and synthesis in radio astronomy*. Springer, 1986.

[2] R. M. Fuster, *Analysis, design and implementation of a compact interferometer for geostationary orbital tracking*. Final degree thesis, Universitat Politècnica de Catalunya, 2016.

[3] A. Niell, A. Whitney, B. Petrachenko, W. Schlüter, N. Vandenberg, H. Hase, Y. Koyama, C. Ma, H. Schuh, G. Tuccari, *et al.*, "Vlbi2010: Current and future requirements for geodetic vlbi systems," *International VLBI Service for Geodesy and Astrometry*, pp. 13–40, 2005.

[4] C. Audoin and J. Vanier, "Atomic frequency standards and clocks," *Journal of Physics E: Scientific Instruments*, vol. 9, no. 9, p. 697, 1976.

[5] L. L. Lewis, "An introduction to frequency standards," *Proceedings of the IEEE*, vol. 79, no. 7, pp. 927–935, 1991.

[6] M. A. Lombardi, L. M. Nelson, A. N. Novick, and V. S. Zhang, "Time and frequency measurements using the global positioning system," *Cal Lab: International Journal of Metrology*, vol. 8, no. 3, pp. 26–33, 2001.

[7] W. Lewandowski and C. Thomas, "Gps time transfer," *Proceedings of the IEEE*, vol. 79, no. 7, pp. 991–1000, 1991.

[8] A. Gifford, S. Pace, and J. McNeff, "One-way gps time transfer 2000," tech. rep., NATIONAL INST OF STANDARDS AND TECHNOLOGY BOULDER CO, 2000.

[9] F. C. Permañé, *Design and characterization of an ultra-stable GNSS disciplined reference oscillator*. Final degree thesis, Universitat Politècnica de Catalunya, 2014.

[10] R. S. Hernández, *Diseño e implementación de un oscilador disciplinado por GPS para sincronismo en un VLBI.* Final degree thesis, Universitat Politècnica de Catalunya, 2015.

[11] *Ublox LEA-M8T Protocol Specification.* Available at `https://www.u-blox.com/sites/default/files/NEO-LEA-M8T-FW3_DataSheet_(UBX-15025193).pdf`.

[12] *Ublox LEA-M8T Datasheet.* Available at `https://www.u-blox.com/sites/default/files/NEO-LEA-M8T-FW3_DataSheet_(UBX-15025193).pdf`.

[13] *UA78M33C Datasheet.* Available at `http://www.ti.com/lit/ds/symlink/ua78m.pdf`.

[14] *Atmel ATMega32U4 Datasheet.* Available at `https://cdn.sparkfun.com/datasheets/Dev/Arduino/Boards/ATMega32U4.pdf`.

[15] *Arduino ProMicro 3.3/8MHz Datasheet.* Available at `https://cdn.sparkfun.com/datasheets/Dev/Arduino/Boards/ProMicro8MHzv1.pdf`.

[16] *U-Center User Guide.* Available at `https://www.u-blox.com/sites/default/files/u-center_UserGuide_(UBX-13005250).pdf`.

[17] *RG-223 Coaxial Cable Datasheet.* Available at `https://www.pasternack.com/images/ProductPDF/RG223-U.pdf`.