Monitor

for a GPS receiver

2007

Title:

Monitor for GPS

Project group:

Group number 1056

Project period:

From 20 of March to 28 of June

Participants:

M. Olivier BOURREL

Supervisor:

Professor Kai Borre

Number of copies: 6

ABSTRACT

Mosts questions on the Global Positioning System (GPS) are related to the actual status of the system. For a given position on the surface of the Earth, the actual number of visible satellites, their trajectories, the time of their rising and setting, the actual receiver position, the actual GMT, etc are the parameters of interest.

The MONITOR displays all these issues on a carefully designed graphical user interface. The establishment of the MONITOR is based on a complete understanding of how GPS works. This conceptual undertstanding is converted into a comprehensive computation.

PREFACE

This report presents the work of group 1056 of the GPS specialization at the Aalborg University, Denmark.

The report contains 5 chapters and 2 appendices. Chapter 1 introduces the general project matter. Chapter 2 gives problem formulation. Chapter 3 covers basic concepts about GPS technology, theory and specifications. Chapter 4 gives the system description and algorithms of the software. Chapter 5 covers the implementation of the GUI. Finally, a complete reference of literature and list of figures are provided in the appendices.

Olivier Bourrel

CONTENTS

1. INTRODUCTION 8

1.1. BACKGROUND 8

1.1 GPS 8

1.2 GUI 9

1.3 GPS RECEIVER: UBLOX – ANTARIS® 4 11

1.4 MATLAB 12

2. PROBLEM FORMULATION 13

2.1. ACTUAL PROBLEM WITH GPS 13

2.2. PROBLEM STATEMENT 13

2.3. HOW TO ESTABLISH A COMMUNICATION PROTOCOL WITH THE RECEIVER WITH AN APPROPRIATE AND SUITABLE SOFTWARE/PROGRAM? 14

2.4. EXPLORE THE DATA TYPES AND DATA FORMATS AVAILABLE IN THE RECEIVER, AND CHOOSE BETWEEN THESE, THE FORMAT FOR MEASUREMENT AND EPHEMERIS DATA 15

2.5. WHICH COMPUTATIONS ARE NEEDED, TO ESTABLISH RECEIVER POSITION? 15

2.6. HOW DO WE PRESENT THESE VISUALIZATIONS GRAPHICALLY? 15

3. THEORY AND SPECIFICATIONS 16

3.1. POSITIONING WITH GPS 16

3.2. GPS SIGNALS 16

3.3. A BASIC GPS RECEIVER 19

3.4. USER POSITION 20

3.5. BASIC EQUATIONS FOR FINDING USER POSITION 22

3.6. GPS ERRORS AND ACCURACY 23
3.6.1. User Equivalent Range Error (UERE) *23*
3.6.2. Dilution Of Precision (DOP) computation *24*
3.7. MEASUREMENT OF PSEUDORANGE 26
3.8. SIGNAL TO NOISE RATIO 28
3.9. C/A CODE GENERATOR 29
3.10. GPS DATA FORMAT 30
4. SYSTEM DESCRIPTION AND ALGORITHMS 32
4.1. OVERVIEW 32
4.2. THE SOFTWARE 33
4.2.1. The flow diagram *33*
4.2.2. Description *36*
4.3. SERIAL COMMUNICATION 37
4.3.1. Creating the serial port object *37*
4.3.2. Configuring the port setting *37*
4.3.3. Writing and reading data *38*
4.4. DATA FORMAT IN UBLOX® ANTARIS 4 40
4.4.1. UBX protocol key features *40*
4.4.2. UBX packet structure *40*
4.4.3. UBX Class IDs *41*
4.5. DETAILED DESCRIPTION 42
5. IMPLEMENTATION OF THE GUI 46
5.1. GRAPHICAL USER INTERFACE 46
5.1.1. Advantages of GUIs *46*
5.1.2. Start with the GUI *47*

5.1.3. Basic window *48*

5.1.4. The menus *50*

5.2. THE POSITION 51

5.3. SATELLITE POSITIONS 53

5.4. THE DOP VALUES IMPLEMENTATION IN THE GUI 54

5.5. SIGNAL TO NOISE RATIO (SNR) 55

5.5.1. The Signal Strength *55*

5.5.2. The SNR *57*

5.6. THE TIME GMT 58

6. CONCLUSION 59

6.1. OBJECTIVES 59

6.2. RESULTS 59

6.2.1. GPS *59*

6.2.2. Communicating with GPS receiver *59*

6.2.3. Programming in Matlab *60*

6.3. CONCLUSION AND PERSPECTIVES 60

6.3.1. Conclusion *60*

6.3.2. Perspectives *60*

7. APPENDIX A: LITERATURE 61

8. APPENDIX B: LIST OF FIGURES 63

1.Introduction

1.1. Background

Global Positioning System (GPS) technology has become an interesting field of study for the past decade, this is because there have been significant advances made in receiver and systems technologies. These advances have enhanced the effectiveness of satellite positioning methodologies. Today, these methodologies show potential usefulness beyond their surveying roots.

Moreover, processing techniques of GPS carrier signals have been greatly improved. Attempt in finding better and better ways of increasing the stability and accuracy of the carrier signal, has led to the opening up of new possibilities with improved GPS performance not originally envisioned.

1.1 GPS

The GPS is currently the only fully functional Global Navigation Satellite System (GNSS). More than two dozen GPS satellites are in medium Earth orbit, transmitting signals allowing GPS receivers to determine the receiver's location, speed and direction.

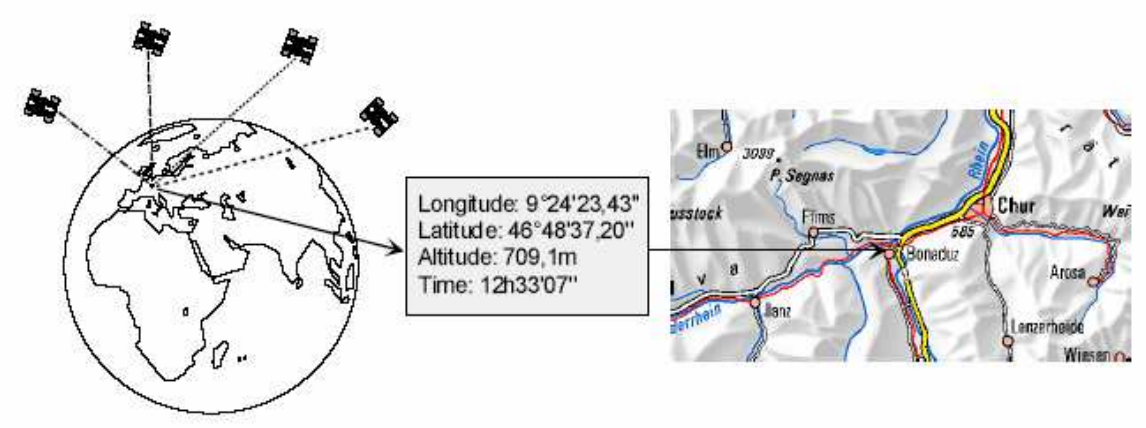

Figure 1.1: The basic function of GPS [17]

Since the first experimental satellite was launched in 1978, GPS has become an indispensable aid to navigation around the world, and an important tool for map-making and land surveying. GPS also provides a precise time reference used in many applications including scientific study of earthquakes, and synchronization of telecommunications networks.

Developed by the United States Department of Defense, it is officially named NAVSTAR GPS (NAVigation Satellite Timing And Ranging Global Positioning System). The satellite constellation is managed by the United States Air Force 50th Space Wing.

GPS receivers come in a variety of formats (see figure 1.2), from devices integrated into cars, phones, and watches, to dedicated devices such those shown here from manufacturers Trimble, Garmin and Leica (left to right).

Figure 1.2: Civilian GPS receiver for navigation [17]

1.2 GUI

A graphical user interface (GUI) is a human-computer interface (i.e., a way for humans to interact with computers) that uses windows, icons and menus and which can be manipulated by a mouse (and often to a limited extent by a keyboard as well).

GUIs stand in sharp contrast to command line interfaces (CLIs), which use only text and are accessed solely by a keyboard. The most familiar example of a CLI to many people is MS-DOS. Another example is Linux when it is used in console mode (i.e., the entire screen shows text only).

A window is a (usually) rectangular portion of the monitor screen that can display its contents (e.g., a program, icons, a text file or an image) seemingly independently of the rest of the display screen. A major feature is the ability for multiple windows to be open simultaneously.

Each window can display a different application, or each can display different files (e.g., text, image or spreadsheet files) that have been opened or created with a single application.

An icon is a small picture or symbol in a GUI that represents a program (or command), a file, a directory or a device (such as a hard disk or floppy). Icons are used both on the desktop and within application programs. Examples include small rectangles (to represent files), file folders (to represent directories), a trash can (to indicate a place to dispose of unwanted files and directories) and buttons on web browsers (for navigating to previous pages, for reloading the current page, etc.).

Commands are issued in the GUI by using a mouse, trackball or touchpad to first move a pointer on the screen to, or on top of, the icon, menu item or window of interest in order to select that object. Then, for example, icons and windows can be moved by dragging (moving the mouse with the held down) and objects or programs can be opened by clicking on their icons.

A GUI is a user interface based on graphics (icons and pictures and menus) instead of text using a mouse as well as a keyboard as an input device.

A GUI is used for making communication with a program or a machine easier for the user, and for displaying information which the user can easily relate.

A GUI for monitoring GPS observables and position is created by following these general rules:

- Simplicity
- Consistency
- Familiarity

Most GPS receivers are already equipped with capabilities for communicating graphically with the user. This project will consist of developping a GUI for the Ublox Antaris® 4 GPS receiver (see Figure 1.2).

The GUI will contain many windows in order to see the different information from the receiver:

- Position of GPS receiver
- Position of satellites in the sky
- Precision of position
- Position error
- Signal Strength
- Pseudoranges error
- ...

1.3 GPS receiver: Ublox – Antaris® 4

GPS receivers are becoming consumer products. In addition to their outdoor use (hiking, cross-country skiing, ballooning, flying, and sailing), receivers can be used in cars to relate the driver's location with traffic and weather information.

Any GPS receiver contains a computer that computes its own position by getting bearings from pseudoranges to at least four satellites. The result is provided in the form of a geographic position - longitude and latitude - to, for most receivers, within 20 meters.

Moreover, if the receiver is also equipped with a display screen that shows a map, the position can be shown on the map. And if a fourth satellite can be received, the receiver/computer can figure out the altitude as well as the geographic position.

If you are moving, your receiver may also be able to calculate your speed and direction of travel and give you estimated times of arrival to specified destinations.

This GPS receiver contains a u-center program which show information about the satellite constellation, signal to noise ratio, time etc (see figure 1.3):

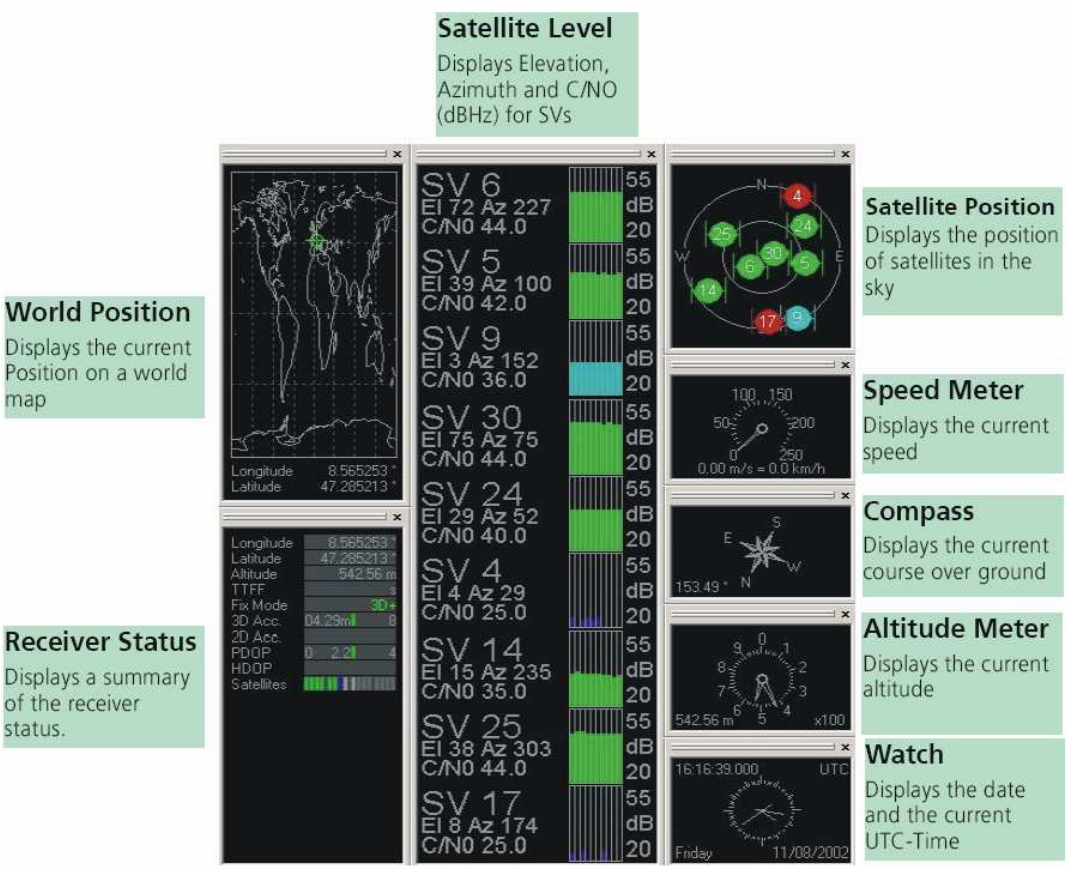

Figure 1.3: Start display after a successful computation of receiver position [12]

1.4 Matlab

It has chosen to use Matlab to design and implement the GUI in because Matlab is the software available to the writer.

It is a good solution to use Matlab because it has good capabilities in linear algebra which is very important in the programming GPS data. Besides the capabilities, Matlab also supports GUI development. And Matlab has everything needed for this project.

2. Problem Formulation

2.1. Actual problem with GPS

The functionality of conventional GPS receiver architectures is usually determined primarily by hardware with minimal configurability through software. Because the hardware dominates the design, upgrading a conventional receiver design in the worst case means completely abandoning the old design and starting over again. In upgrating a software GPS receiver design the vast majority of the new content is software and the rest is improvements in hardware component design.

A development of new positioning receivers becomes problemetic due to rapid changes in positioning technology. There is a need for advanced receiver developments with respect to their flexibility against possible changes in the structure of future signals, thus influencing the architecture concept of the receiver.

A software GPS receiver system in its entirety is costly to implement but the technology provides versability and flexibility which are far greater than that of any other receiver design technology. Therefore a software receiver is still attractive from an economic standpoint because the particular mix of services can be altered rapidly and the value of the investment not lost due to changing demand. It is not unrealistic that software design approach can become an industry standard for GPS receivers. Knowledge and experience in software receiver design possibly will become a necessity for an engineer involved in this area.
Thus taking such kind of project is an interesting and challenging task.

2.2. Problem statement

A three month period assigned for the project sets a certain limitation for achieving the desired task. Thus, the main project task is segregated into subtasks in order to analyse the total project. The main project is reformulated and divided.

So, in realizing the pre analysis of this project and working with the Ublox Antaris® 4, the following problems/questions have been defined.

• How to establish a communication protocol with the receiver with an appropriate and suitable software/program?

• Explore the data types and data formats available in the receiver, and choose between these, the format for measurement and ephemeris data.

• Which computations are needed, to establish receiver position?

• How do we present these visualizations graphically?

2.3. How to establish a communication protocol with the receiver with an appropriate and suitable software/program?

The software GPS receiver consists of a RF front-end, an ADC, and a software GPS program that runs on PC.

All the other processing including signal acquisition, tracking, data decoding, and solving position are all implemented in software using signal processing techniques.

The aim of the software GPS receiver is to transferre all the hardware signal processing into a set of software algorithms running on a suitable hardware platform, including a digital front-end that is in charge of digitizing the incoming signal and feeding the software with the data.
The software GPS receivers process the signal in software giving great flexibility and evidently reduced processing speed (software is muche slower than hardware).
The signal processing function is the core of any GPS receiver, performing the following basic functions:

Signal acquisition: It is the signal synchronization process detecting the presence of the satellite and giving estimates of the signal parameters.

Signal tracking: One of the more important function is the tracking process. It is to demodulate all the datas from the incoming GPS signal and to detect the data type.

Data decoding and pseudorange estimation: When the signal data is received it is possible to decode the GPS signal data and to estimate the distance between the receiver and the satellites (this distance is the pseudorange).

Position solution: The position solution is an estimate of the receiver position. To estimate this position, there are two solutions. The receiver computes a position; alternatively we also compute the position by the use of our Matlab code.

2.4. Explore the data types and data formats available in the receiver, and choose between these, the format for measurement and ephemeris data

In this receiver, the data types and data formats available are contained in the UBX Binary Protocol. All data we need to do this project are available from this protocol. But some are useless for measurement and ephemeris data and we have to select what we need for the Graphical User Interface.

2.5. Which computations are needed, to establish receiver position?

Knowing the position of satellites from the UBX Binary Protocol, the user position and GPS time can be computed from at least four satellites.

Thus the receiver position can be calculated by different methods. For instance by using least-squares method or by Kalman filtering algorithm. But using different algorithms the position computation can give better results in terms of accuracy and precision.

2.6. How do we present these visualizations graphically?

We will present these visualizations graphically by using the GUI of Matlab. GUI means Graphical User Interface and will serve to show much information to know how the GPS technology works. Graphically, we will present graphics where we will find the satelittes available in the sky, the dilution of precision, and position of the GPS receiver and actual GMT.

3.Theory and Specifications

This chapter contains basic theoretical knowledge required to understand the development of the software-based receiver. In this chapter, I describe the basic specifications.

3.1.Positioning with GPS

Distances between satellites and receiver are not directly measured but need to be computed. Typically the distances are derived from two fundamental GPS measurements:

Pseudorange measurement: it is the basic GPS observable that all types of receiver can make. It utilize the C/A and/or P codes modulated onto the carrier signals. The measurement records the actual time taken from the relevant code to travel from the satellite to the receiver. This time is then multiplied by the speed of light to convert it into a distance. At a certain instant in time, codes (C/A codes, for example) are generated within the satellite and the receiver. The satellite's code is then transmitted and is picked up by the receiver. The receiver compares the state of the incoming code with its own code - the difference is the travel time.

Carrier phase measurement: it can be thought of as the same as the pseudorange measurement in that it is a measurement of the time taken for a signal to travel from the satellite to the receiver. This time, the L1 and L2 carrier signals are utilized. Since these have wave lengths of 19 and 24 cm respectively, the distance is some multiple (the integer ambiguity) of 19 or 24 cm plus the observed quantity. The receiver does, however, give the user a helping hand to allow for the ambiguities to be computed. When the receiver locks onto a satellite signal, it assigns some arbitrary value to the ambiguity. From then on, the receiver counts how many complete cycles have occurred since lock on. In other words, the integer ambiguity needs to be computed only when each satellite is locked on by the receiver - subsequent measurements are correct relative to the initial one.

3.2.GPS signals

Positioning with GPS is based on radio signals transmitted from the satellites. The signals transmitted are a mixture of codes and data messages that are needed to determine a distance between a satellite and the receiver and to separate signals from different satellites. The GPS signal can be referred to as ranging signal.

The GPS ranging signal is broadcast at two frequencies.

A primary signal:

L1 = 1575,42 MHz = 154 x 10,23 MHz

A secondary signal:

L1 = 1227,6 MHz = 120 x 10,23 MHz

And mathematically, the signals coming from satellites can be written as:

$$S_{L1}(t) = A_{p1}P_s D(t)\cos(2\pi f_1 t + \phi_1) + A_{c1}C_s D(t)\sin(2\pi f_1 + \phi_1)$$
$$S_{L2}(t) = A_{p2}P_s D(t)\cos(2\pi f_2 t + \phi_2)$$

Where

$S_{L1}(t)$ and $S_{L2}(t)$	- the two different GPS signals on L1 and L2
f_1 and f_2	- correspond to L1 and L2 carrier frequencies
P_s, C_s	- the binary spreading codes for satellite s
A_{p1}, A_{c1} and A_{p2}	- signal amplitudes of P_s and C_s codes on L1 and L2
$D(t)$	- the 50 Hz data message broadcast by satellites
ϕ_1 and ϕ_2	- L1 and L2 carrier phases
t	- time

The codes *Ps* and *Cs* are known as precision or P-code and coarse/acquisition or C/A-code respectively.

Most civilian users only use the L1 frequency C/A code. C/A code is a short Pseudo Random Noise (PRN) code broadcast at a bit (or chipping) rate of 1.023 MHz. This is the principle civilian ranging signal, and it is always broadcast unencrypted. The use of this signal is called the standard positioning service (SPS). At present the C/A code is available only on L1.

The chipping rate of the C/A code is selected so that the period of the code corresponds exactly to 1 ms for time-keeping purposes. The C/A code consists of a sequence of +1 and -1, which repeats itself every millisecond. The codes are modulated onto the carrier by the binary phase shift keying (BPSK) modulation.

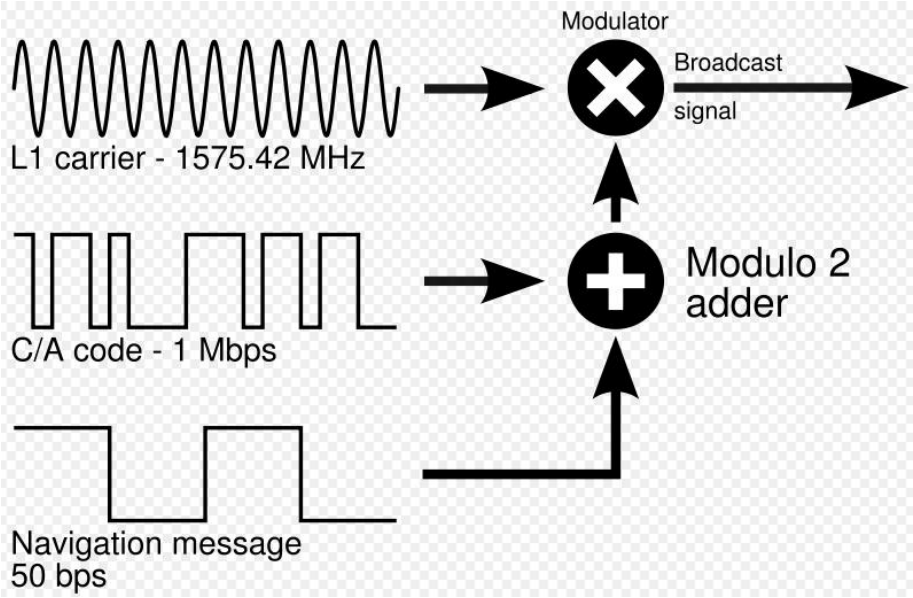

Figure 3.1: GPS broadcast signal [17]

Both carriers carry the broadcast satellite message *D(t)*, as low frequency (50 Hz) stream of data designed to inform the user about the health and position satellite. This data can be decoded by receiver and used for positioning in real or post-process time.

The PRN signals are used for two purposes, to separate different signals and to determine the propagation delay. The PRN codes are unique for each satellite and have special correlation properties: the cross correlation is almost zero for all time shifts for different codes, and the autocorrelation is high at the zero time shift and close to zero for all ather time shifts.

This property makes it possible for a receiver to acquire signals from different satellites and to determine the propagation time by correlating the received signal with an internally generated copy of the transmitted signal. The time shift for the highest correlation is a measure of the propagation time.

3.3.A basic GPS receiver

The basic GPS receiver is shown in figure 3.2. The signals transmitted from the GPS satellites are received from the antenna. Through the radio frequency (RF) chain the input signal is amplified to a proper amplitude and the frequency is converted to a desired output frequency. An analog-to-digital converter (ADC) is used to digitize the output signal. The antenna, RF chain, and ADC are the hardware used in the receiver.

After the signal is digitized, software is used to process it. Acquisition means to find the signal of a certain satellite. The tracking program is used to find the phase transition of the navigation data. In a conventional receiver, the acquisition and tracking are perform by hardware.

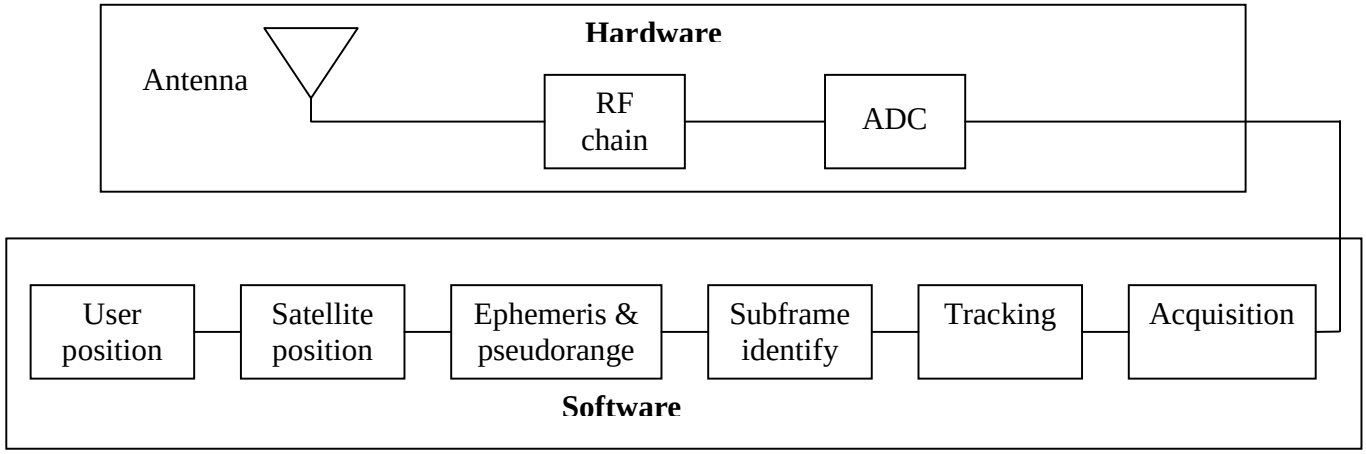

Figure 3.2: A fundamental GPS receiver [1]

From the navigation data phase transition the subframes and navigation data can be obtained. Ephemeris data can be obtained from the navigation data. The pseudoranges are estimated from a correlation process based on the fronts of the navigation data. The ephemeris data are used to obtain the satellite positions. Finally, the user position can be calculated for the satellite positions and the pseudoranges. Both hardware used to collect digitized data and the software used to find the user position will be discussed in the report.

3.4. User Position

Before determining the user position, we have to understand the term pseudorange. This word is the association of pseudo and range. That is to say we can articulate pseudorange as an equation:

$$pseudo\text{-}range = range + receiver\ clock\ offset$$

The receiver clock has an offset and is shown in this equation:

$$b = c \cdot dt_u$$

where

b	- the receiver clock offset
dt_u	- the clock error
c	- the speed of light

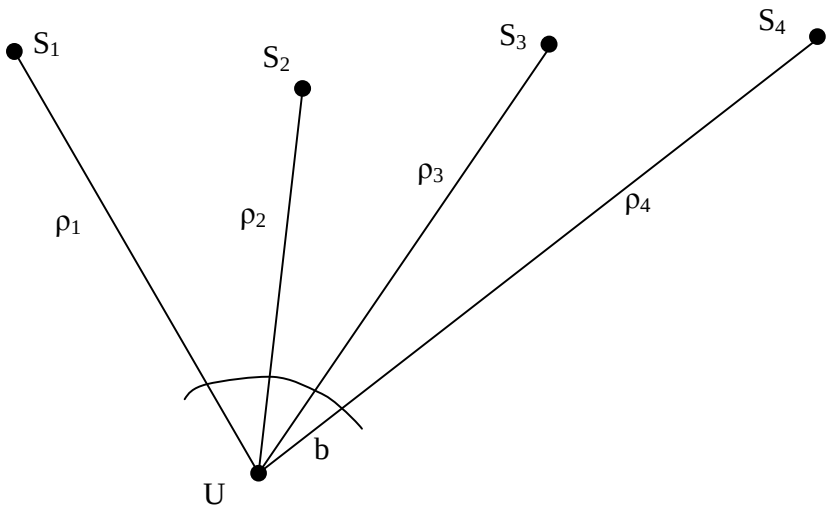

Figure 3.3: Explanation of receiver clock offset

The position of a certain point in space can be found from distances measured from this point to some known positions in space. Let us use some examples to illustrate this point. In figure 3.3, the user position is on the x-axis (this is a one-dimensional case). If the satellite position S_1 and the distance to the satellite x_1 are both known, the user position can be at two

places, either to the left or right of S_1. In order to determine the user position, the distance to another satellite with known position must be measured. In this fugure, the positions of S_2 and x_2 uniquely determine the user position U.

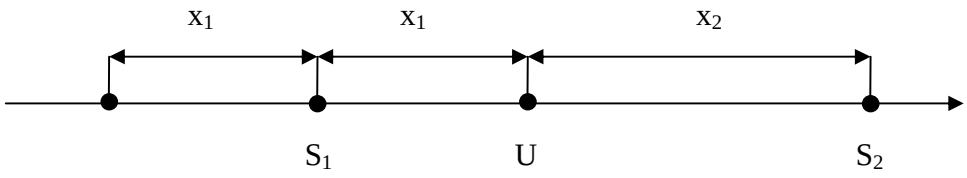

Figure 3.4: One-dimensional user position

Figure 3.4 shows a two-dimensional case. So as to determine the user position, three satellites and three distances are required. The trace of a point with constant distance to a fixed point is a circle in two-dimensional case. Two satellites and two distances give two possible solutions because two circles intersect at two points. A third circle is needed to uniquely determine the user position.

For similar reasons one might decide that in a three-dimensional case four satellites and four distances are needed. The equal-distance trace to a fixed point is a sphere in a three-dimensional case. Two spheres intersect to make a circle. This circle intersects another sphere to produce two points. In order to determine which point is the user position, one more satellite is needed.

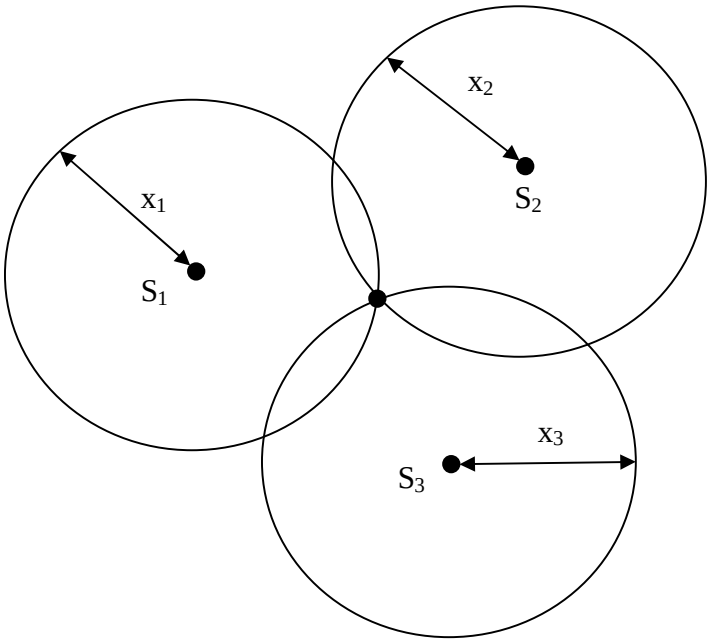

Figure 3.5: Two-dimensional user position

In GPS, the position of the satellite is known from the ephemeris data transmitted by the satellite. The receiver measures the distances from the receiver to the satellites. Therefore, the position of the receiver can be determined.

3.5. Basic equations for finding user position

In this section the basic equations for determining the user position will be presented. Assume that the distance measured is accurate and under this condition three satellites are sufficient. In those equations, there are three known points at location r_1 or (x_1, y_1, z_1), r_2 or (x_2, y_2, z_2) and r_3 or (x_3, y_3, z_3), and an unknown point at r_u or (x_u, y_u, z_u). If the distance between the three known points to the unknown point can be measured as

$$\rho_1 = \sqrt{(x_1 - x_u)^2 + (y_1 - y_u)^2 + (z_1 - z_u)^2} + b$$

$$\rho_2 = \sqrt{(x_2 - x_u)^2 + (y_2 - y_u)^2 + (z_2 - z_u)^2} + b$$

$$\rho_3 = \sqrt{(x_3 - x_u)^2 + (y_3 - y_u)^2 + (z_3 - z_u)^2} + b$$

$$\rho_4 = \sqrt{(x_4 - x_u)^2 + (y_4 - y_u)^2 + (z_4 - z_u)^2} + b$$

With *b* the receiver clock offset seen previously.

Because there are three unknowns and three equations, the values of xu, yu and zu can be determined from these questions. Theoretically, there should be two sets of solutions as they are second-order equations. These equations can be solved relatively easily with linearization and an iterative approach. The solution of these equations will be discussed later.

In GPS operation, the positions of the satellites are given. This information can be obtained from the data transmitted from the satellites and will be discussed. The distances from the user (the unknown position) to the satellites must be measured simultaneously at a certain time instance.

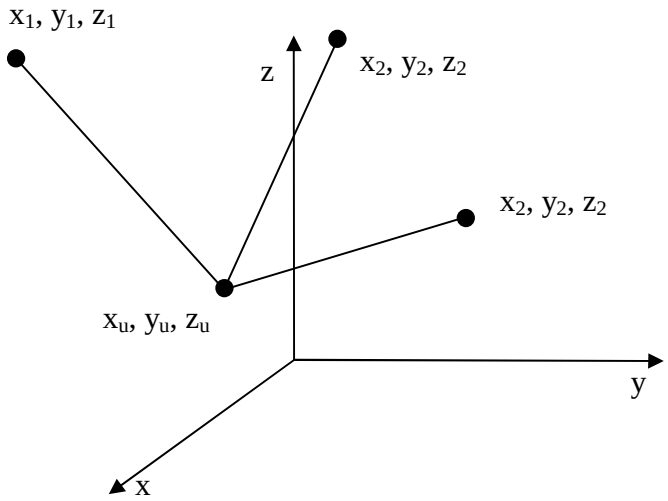

Figure 3.6: Use three known positions to find one unknown position

Each satellite transmits a signal with a time reference associated with it. By measuring the time of the signal traveling from the satellite to the user the distance between the user and the satellite can be found.

3.6.GPS errors and accuracy

3.6.1. User Equivalent Range Error (UERE)

The position calculated by a GPS receiver requires the current time, the position of the satellite and the measured delay of the received signal.

To measure the delay, the receiver compares the bit sequence received from the satellite with an internally generated version. By comparing the rising and trailing edges of the bit transitions, modern electronics can measure signal offset to within about 1% of a bit time, or approximately 10 nanoseconds for the C/A code. Since GPS signals propagate nearly at the speed of light, this represents an error of about 3 meters. This is the minimum error possible using only the GPS C/A signal.

Sources of User Equivalent Range Errors (UERE)

Source	Effect
Ionospheric effects	± 5 meter
Ephemeris errors	± 2.5 meter
Satellite clock errors	± 2 meter
Multipath distortion	± 1 meter
Tropospheric effects	± 0.5 meter
Numerical errors	± 1 meter or less

3.6.2. Dilution Of Precision (DOP) computation

Several external sources introduce errors into a position estimated by a GPS receiver.

One important factor in determining positional accuracy is the constellation, or geometry, of the group of satellites from which signals are being received. DOP only depends on the position of the satellites: how many satellites you can see, how high they are in the sky, and the bearing towards them. This is often refered to as the geometry.

An indicator of the quality of the geometry of the satellite constellation is the Dilution of Precision or DOP. The computed position can vary depending on which satellites are used for the measurement. Different satellite geometries can magnify or lessen the position error. A greater angle between the satellites lowers the DOP, and provides a better measurement. A higher DOP indicates poor satellite geometry, and an inferior measurement cofiguration, or in other words: the lower the value the greater the confidence in the solution.

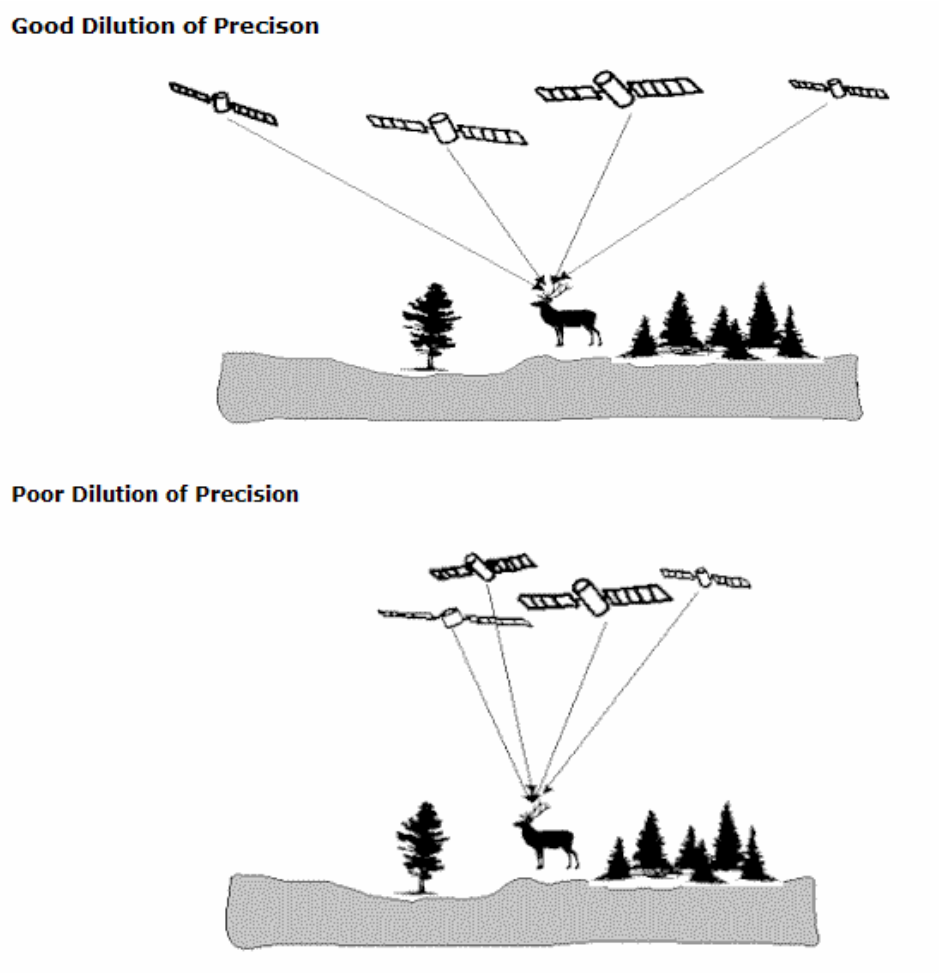

Figure 3.7: Different Dilution Of Precision

DOP is often divided up into components. These componets are used because the accuracy of the GPS system varies. The satellites move, so the geometry varies with time, but it is very predictable.

The geometrical dilution of precision (GDOP) is computed from the geometric relationships between the receiver position and the positions of the satellites the receiver is using for navigation.

- Geometric DOP (GDOP). Three position coordinates plus clock offset in the solution.

$$GDOP = \frac{1}{\sigma}\sqrt{\sigma^2{}_x + \sigma^2{}_y + \sigma^2{}_z + \sigma^2{}_b}$$

Where

σ	-the standard deviation derived from the least-square procedure
$\sigma_x \sigma_y \sigma_z$	- the measured rms errors of the user position in the xyz directions
σ_b	- the measured rms user clock error expressed in distance

- Position DOP (PDOP). Three coordinates.

$$PDOP = \frac{1}{\sigma}\sqrt{\sigma^2{}_x + \sigma^2{}_y + \sigma^2{}_z}$$

- Horizontal DOP (HDOP). Two horizontal coordinates.

$$GDOP = \frac{1}{\sigma}\sqrt{\sigma^2{}_x + \sigma^2{}_y}$$

- Vertical DOP (VDOP). Height only.

$$VDOP = \frac{\sigma_z}{\sigma}$$

- Time DOP (TDOP). Clock offset only.

$$TDOP = \frac{\sigma_b}{\sigma}$$

3.7. Measurement of pseudorange

The fundamental observation quantity in GPS is the propagation time of the signal between the satellite and the receiver. The distance is measured on the basis of the propagation time multiplied by the speed of light.

A GPS receiver basically makes two kinds of measurements: pseudorange and carrier phase.

Time of travel Δt is measured and then the range to satellite is calculated as:

$$\rho = c \cdot \Delta t + b$$

where

ρ - the distance measurement (pseudorange)
c - the speed of light, which in the GPS is set to 299792458 m/s²
Δt - measured travel time
b - the receiver clock offset

Each terminal, transmitter (satellite) and receiver is controlled by a separate clock. The satellite clock generates the signal. The receiver clock detects when the signal arrives. But both clocks must keep the same time, because an error in the synchronization of the two clocks of 1 µs creates an error in range of 300 meters. But since it is pratically impossible to keep the two clocks perfectly synchronized physically, it is usually done mathematically. In general each clock will run at its own rate, keeping its own time. However, if the relationship between the two time bases defined by the clocks is known, then they can still be synchronized. So the problem is to find the relationship between the two time bases. GPS works only bacause we can asume that all GPS satellite clocks are synchronized. A user can then assume that all the simultaneous GPS ranges measured by their receiver are related to the same clock at the satelitte and of the range measurement.

The difference in two time frames introduces a bias into the measurement. Pseudorange technique requires accurate time information. Pseudorange is the time shift required to line up a replica of the code generated in the receiver with the received code from the satellite multiplied by the speed of light. Ideally the time shift is the difference between the time of signal reception (measured in the receiver time frame) and the time of emission (measured in the satellite time frame).

Evry satellite sends a signal at a certain time *tsi*. The receiver will receive the signal at a later time *tu*. The distance between the user and the satellite *i* is

$$\rho_{iT} = c(t_u - t_{si})$$

where

ρ - the true value of pseudorange from user to satellite *i*
c - the speed of light
tu - the true time of reception
tsi - the true time of transmission from satellite *i*

From a practical point of view it is difficult, if not impossible, to obtain the correct time from the satellite or the user. The actual satellite clock time *t'si* and actual user clock time *t'u* are related to the true time as

$$t'_{si} = t_{si} + \Delta_{bi}$$
$$t'_u = t_u + b_{ut}$$

where

Δb_i - the satellite clock error

b_{ut} - the user clock bias error

Besides the clock error, there are other factors affecting the pseudorange measurement. The measured pseudorange ρ_i as

$$\rho_i = \rho_{iT} + \Delta T_i + \Delta I_i - c(\Delta b_i - b_{ut}) + e$$

where

ΔT_i -the tropospheric error

ΔI_i - the ionospheric error

e - other biases

But some of these errors can be corrected: for example, the tropospheric delay can be modeled and the ionospheric error can be corrected in a two-frequency receiver. The errors will cause inaccuracy of the user position. However, the user clock error cannot be corrected through received information. Thus, it will remain as an unknown.

Thus, the pseudoranges are used for computing the position of the receiver in code measurements. So, concretely the pseudorange from the receiver to a satellite is given by the equation:

$$r_u^s = \sqrt{(x_u - x^s)^2 + (y_u - y^s)^2 + (z_u - z^s)^2} + c \cdot dt_u$$

where

r_u^s - the pseudorange

x_u, y_u, z_u - the receiver position

x^s, y^s, z^s - the satellite position

c - the speed of light

dt_u - the error in the receiver clock

3.8. Signal to noise ratio

The signal to noise ratio (SNR) is given as the power of the receive signal relative to the power density of the noise in the signal. If the value of the SNR is high, then the received signal is stronger than the noise in the signal and a better range estimation is possible. The correlation of the received signal with the receiver generated signal will be better if there are less noise in the received signal.

In theory, to compute the SNR, it will be necessary to compute the power of the signal and the power of the noise.

3.9. C/A Code Generator

The satellites also broadcast two forms of clock information, the Coarse / Acquisition code, or C/A which is freely available to the public, and the restricted Precise code, or P-code, usually reserved for military applications.

The C/A code is a 1,023 bit long pseudo-random code broadcast at 1.023 MHz, repeating every millisecond. Each satellite sends a distinct C/A code, which allows it to be uniquely identified.

The P-code is a similar code broadcast at 10.23 MHz, but it repeats only once a week. In normal operation, the so-called "anti-spoofing mode", the P code is first encrypted into the Y-code, or P(Y), which can only be decrypted by units with a valid decryption key.

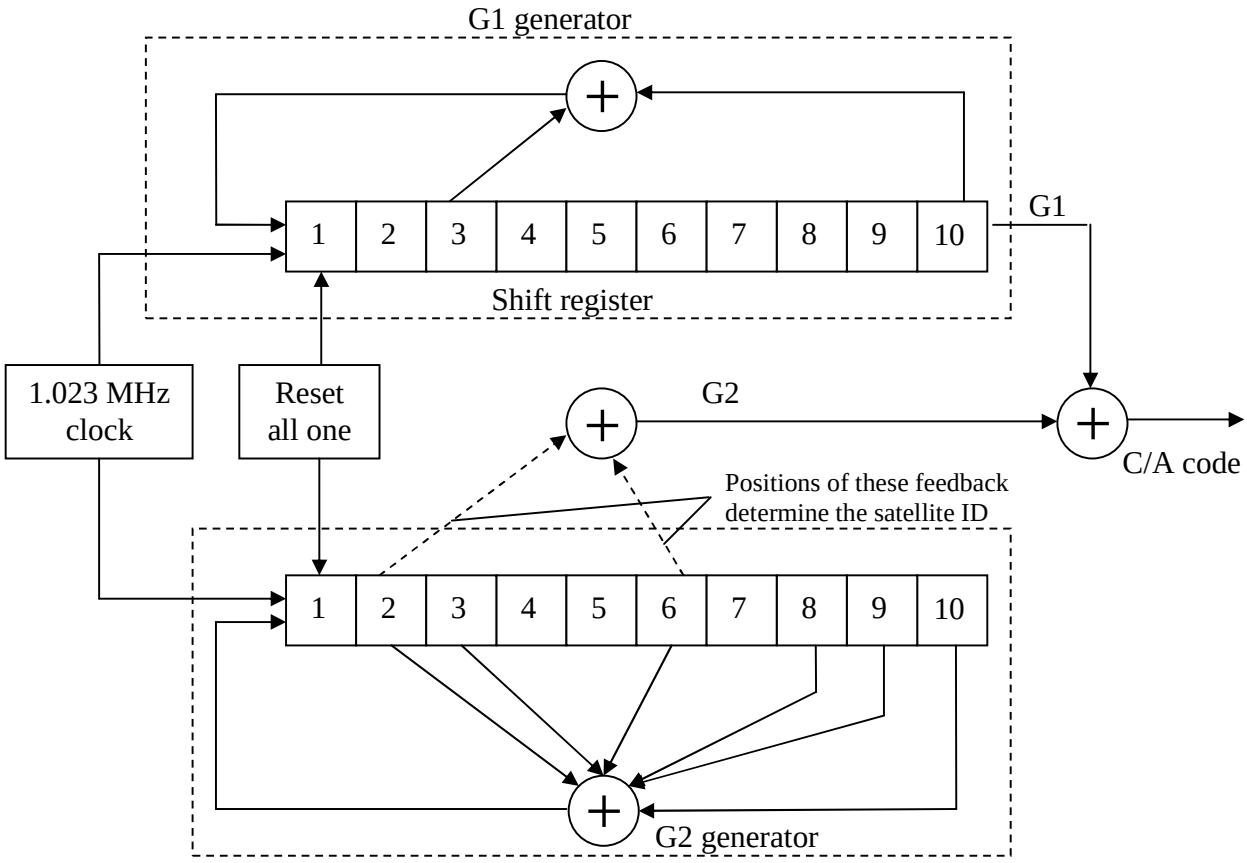

Figure 3.8: C/A code generator [1]

The C/A code signals can be generated from the product of two 1023 bit PRN sequences, G1 and G2.

Both G1 and G2 are generated by 10 stage linear shift registers which are driven by a 1023 MHz clock. The generated sequence length is 1023 bits. The C/A codes sequence is the result of modulo-2 adding of the outputs of G1 and G2. The output of G1 is from the last bit of the register, while the G2's output is generated from two bits which are referred to as code phase selection through another modulo-2 adder. Selection of these bits is defined by the satellite number. The implementation of such code generator is straight forward according to the diagram shown in figure 3.7.

3.10. GPS Data Format

Each GPS satellite provides data required to support the position determination process. Figure 3.8 provides an overview of the data contents and structure within the navigation message. The data includes information required to determine the following:

- Satellite time of transmission
- Satellite position
- Satellite health
- Satellite clock correction
- Propagation delay effects
- Time transfer to UTC
- Constellation status

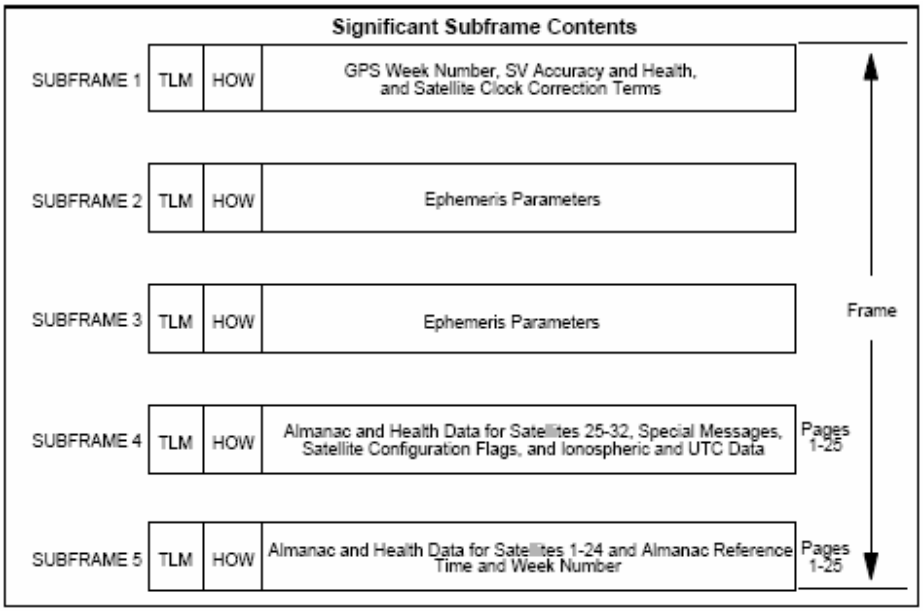

Figure 3.9: Navigation Message Content and Format Overview [5]

The message structure utilizes a basic format of a 1500 bit long frame made up of five subframes, each subframe being 300 bits long. Subframes 4 and 5 are subcommutated 25 times each, so that a complete data message will require the transmission of 25 full frames.

The 25 versions of subframes 4 and 5 are referred to as pages 1 through 25 of each subframe. Each subframe will consist of ten words, each 30 bits long; the MSB of all words is transmitted first.

Each subframe and/or page of a subframe starts with a Telemetry (TLM) word and a Handover word (HOW) pair. The TLM word is transmitted first, immediately followed by the HOW. The latter is followed by eight data words. Each word in each frame contains parity.

At end/start of week (a) the cyclic paging to subframes 1 through 5 will restart with subframe 1 regardless of which subframe was last transmitted prior to end/start of week, and (b) the cycling of the 25 pages of subframes 4 and 5 will restart with page 1 of each of the subframes, regardless of which page was the last to be transmitted prior to the end/start of week. All upload and page cutovers will occur on frame boundaries (i.e., modulo 30 seconds relative to end/start of week); accordingly, new data in subframes 4 and 5 may start to be transmitted with any of the 25 pages of these subframes.

4. System Description and Algorithms

This chapter gives a detailed description of the various parts that make up the system. It begins with the description of how we were able to establish a communication with the receiver, and then a description of the Matlab functions used in data handling and processing.

4.1. Overview

The following section gives a general overview of the components of the system used as an implementation of the solution as stated in the problem formulation

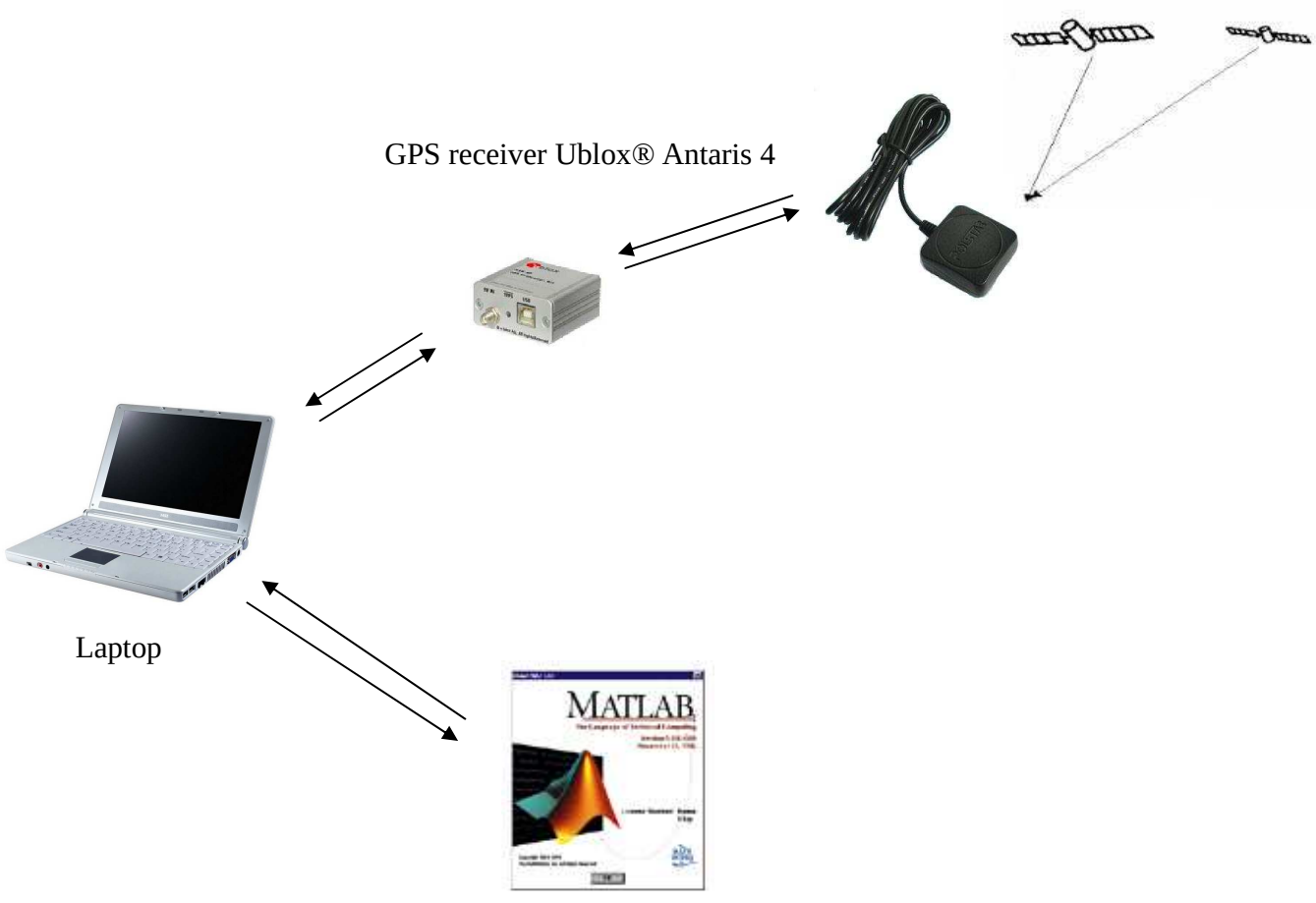

Figure 4.1: Overview of the setup

4.2. The Software

This is the main part of the code, written for the GUI. It consists of several functions and a main loop. The main aim of the function is to maintain the GUI and to acquire new data from the GPS receiver.

4.2.1. The flow diagram

The following general functionality has to be provided by the developed software:

- Establish communication from Matlab workspace to the Ublox Antaris 4 GPS Receiver.
- Handle binary data
- Process data in Real-Time

The function is described in a flow diagram shown in Figure 4.2.

The function starts by declaring and initializing a set of variables and the serial communication. It sets up communication with the receiver and optimizes the serial port settings for a faster communication mode.

After the initialization of the serial communication, the programm calls the *Antaris_send* function. This function has only input variables. It sets up the receiver in order to “ask” to the receiver what parameters we need in the sequel.

Then, the *Antaris_CRC* function is used in this function to compute the data parity. This function needs as input a message in a form of byte array and has as output two CRC bytes.

To request and download data from the receiver, any function is used. We only read what the receiver sends.

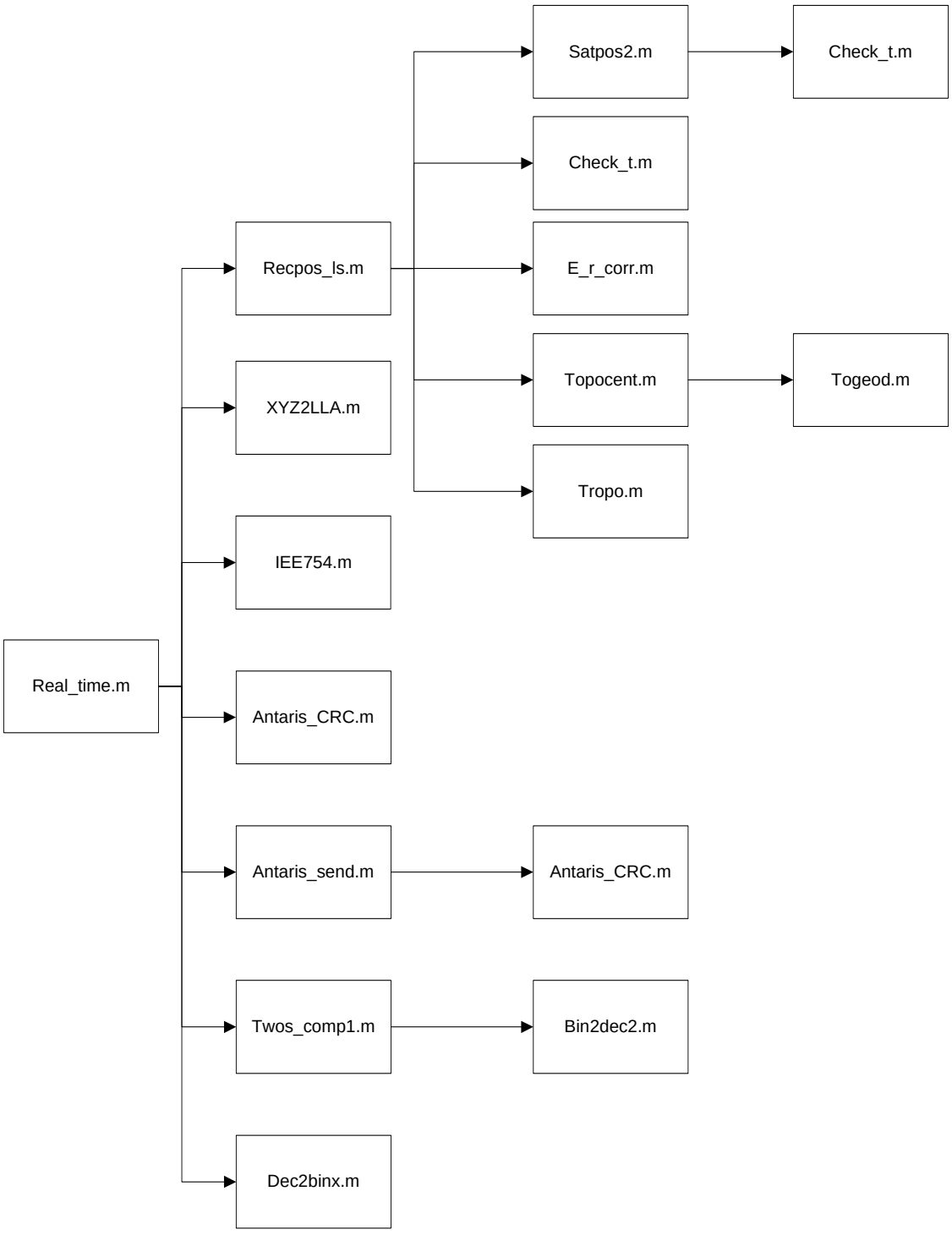

Figure 4.2: Relation diagram of the m-files in the system

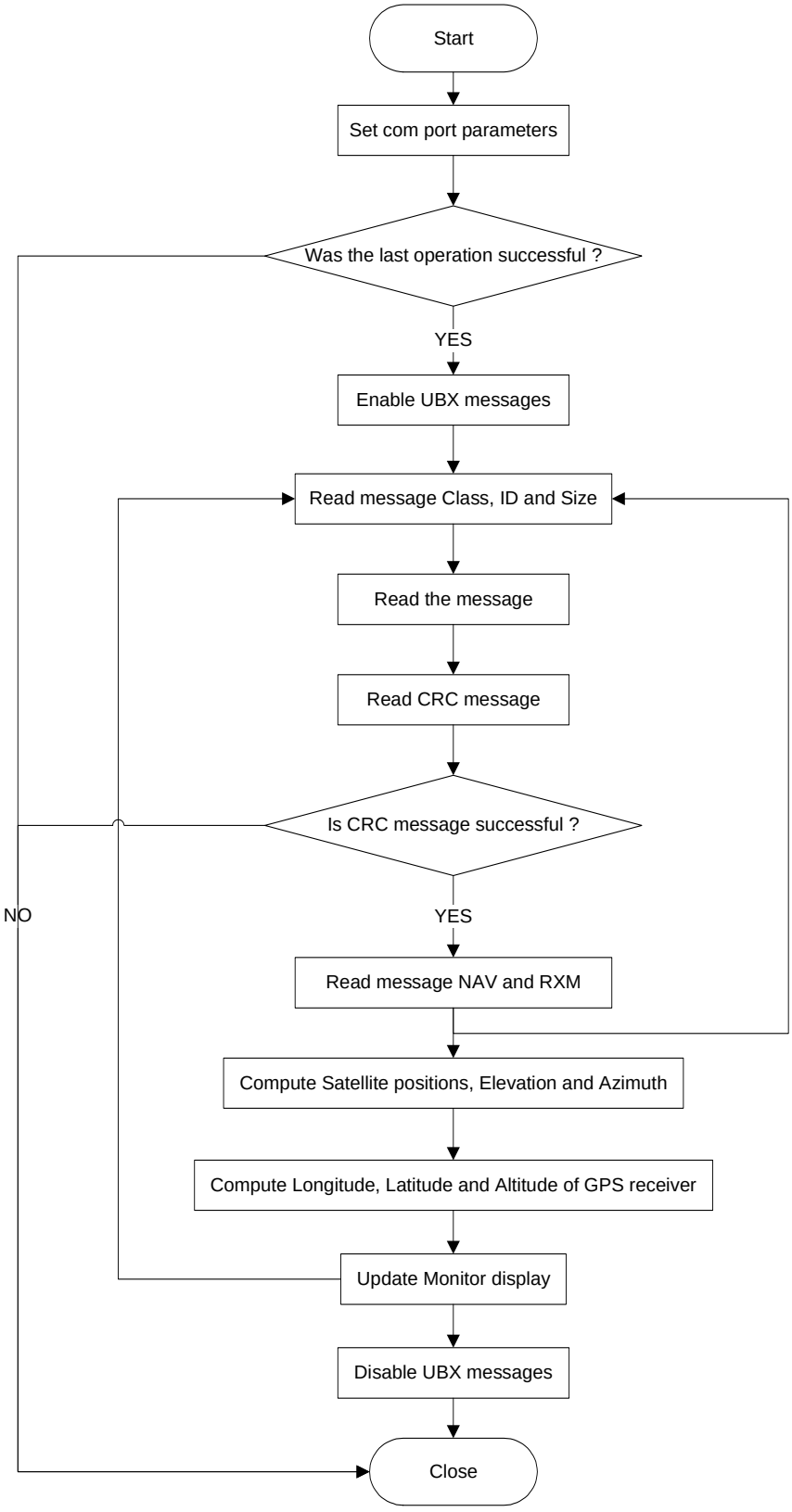

Figure 4.3: flow chart of the overall program

4.2.2. Description

The *Real_time* function is used to write to receiver and to receive all the data we need to make the GUI.

The first lines of the function are to initialize a connection with the GPS receiver through the serial port (COM). There is no input to this function. The ouput is:

- the ephemerides
- the longitude
- the latitude
- the altitude
- the parameters of measurement (RXM)
- the parameters of navigation (NAV)
- the coordinates X, Y and Z
- the Azimuth
- the Elevation
- the PRN number
- the position of satelittes

The function starts by detecting the serial port of communication in COM 5. The detection would be done by asking the user to input the serial port number to which the receiver is connected; there will be a possibility of amelioration. If the GPS receiver is not connected to the computer or an error appeared before, the function creates an error:

Can not open serial port, try again.

Then, if there were no errors and the detections was successful, the function detects the serial port settings of the GPS receiver. It does so by changing the parameters and querying the receiver.
It is necessary to set the communication property:

- Baud rate
- Data bits
- Parity
- StopBits
- Terminator

As we have already told, if those parameters are not the same between the serial port object and the device, the data are not readable or writable.

Next the main function collects all variables needed for the single position calculation of the GPS receiver. So the function will read all the measurements, ephemeris and position from the receiver.

To get the measurement data (RXM), we have to read this class which is marked by the class ID. The receiver download the UBX message to the serial buffer. To extract the values from the binary string the message needs to be parsed. And as the format of the message is known, so the buffer is read by chunks of bytes.

After parsing all the message, the variables must be converted from the binary string of bytes to a decimal number. This convertion is done with the twos_comp1 function which converts two's complement binary into decimal.

4.3.Serial communication

Matlab was used in writing and reading data to the receiver through the serial port. In doing so the following steps were taken:

- Serial port object was created in Matlab workspace.
- Communication settings were configured.
- Data were written and read from the receiver.

4.3.1. Creating the serial port object

We can create a serial port object with the *serial* function. *serial* requires the name of the serial port connected to the device as an input argument.
Once the serial port object is created, its properties are automatically assigned. But, we can also configure property values during object creation.

For example, to create a serial port object associated with the serial port COM5:

```
S = serial('COM5');
```

4.3.2. Configuring the port setting

Before read and write data, the serial port object and the device must have identical communication settings. If they are not identical, then we could not successfully read and write data.

Baud Rate	9600
Data Bits	8
Parity	None
StopBits	1
Terminator	CR/LF

The Matlab Command set was used in setting the communication properties. The same command was used in setting the time out (this is the length of time elapsed before an error message is output if no data is transferred).
The receiver was also set to the same values.

4.3.3. Writing and reading data

Before reading data, the program has to send a message to the GPS receiver in order to take only the information we need. Consequently the Matlab command *fwrite* was used to write into the receiver. But a function is necessary to send messages to the Antaris GPS receiver.

***Antaris_send* function**

Functionality:

This function will make the messages to be send to the Antaris GPS receiver.

Operation:

This function needs mainly to input the message class to be sent and the message ID to be sent in decimal.

fwrite(comm,Antaris_send(2,16,1,1));

So this line sends to the Antaris GPS receiver the message class 2 and the message ID 16. These parameters correspond to the RXM-RAW 0x02 0x10. This message contains much information needed to be able to compute satellite position, Longitude, Latitude or Altitude ...
The receiver will return messages of type RXM-RAW as defined Figure 4.4.

Message		**RXM-RAW**				
Description		**Raw Measurement Data**				
Type		Periodic/Polled				
Comment		This message contains all information needed to be able to generate a RINEX file.				
Message Structure		*Header*	*ID*	*Length*	*Payload*	*Checksum*
		0xB5 0x62	0x02 0x10	8 + NSV * 24	8 + NSV * 24 Bytes	CK_A CK_B
Payload Contents:						
Byte Offset	*Number Format*	*Scaling*	*Name*	*Unit*	*Purpose / Comment*	
0	I4	-	ITOW	ms	Measurement integer millisecond GPS time of week (Receiver Time)	
4	I2	-	Week	weeks	Measurement GPS week number (Receiver Time)	
6	U1	-	NSV	-	# of satellites following	
7	U1	-	RES1	-	Reserved	

Start of repeated block (NSV times}

8 + N*24	R8	-	CPMes	cycles	Carrier phase measurement [L1 cycles]
16 + N*24	R8	-	PRMes	m	Pseudorange measurement [m]
24 + N*24	R4	-	DOMes	Hz	Doppler measurement [Hz]
28 + N*24	U1	-	SV	-	Space Vehicle Number
29 + N*24	I1	-	MesQI	-	Nav Measurements Quality Indicator: >=4 : PR+DO OK >=5 : PR+DO+CP OK <6 : likely loss of carrier lock in previous interval
30 + N*24	I1	-	CNO	dbHz	Signal strength C/No. (dbHz)
31 + N*24	U1	-	LLI	-	Loss of lock indicator (RINEX definition)

Figure 4.4: Messages in Class RXM output status and result data in from the Receiver Manager [14]

Now we send all the messages we want we can read the messages the Antaris GPS receiver send. For example to read the GPS time of the week in seconds, we need to use this line:

*time = twos_comp1(dec2binx([msg(4) msg(3) msg(2) msg(1)]),1) *10^-3;*

So to have the GPS time of the week, the program has to read the bits 1 to 4. These bits contain binary data; we have to compute in decimal to be comprehensible. So a function is necessary:

***twos_comp1* function**

Functionality:

Function converts two's complement binary into decimal

Operation:

sk = twos_comp1(bin, sign)

This function needs in input a signed or unsigned binary number (bin) and the User longitude (sign):

0 - unsigned
1 - signed

And this function returns the decimal number (sk).

Consequently after having computed the decimal number of the message, we have just to multiply by 10^-3 so as to have the GPS time as seconds of week.

So we have to do the same procedure for all classes we want to read.

4.4. Data Format in Ublox® Antaris 4

4.4.1. UBX protocol key features

The u-blox GPS Receivers use a u-blox proprietary protocol to transmit GPS data to a host computer using asynchronous RS232 ports. This proprietary protocol has the following key features:

- Compact. 8 Bit Binary Data is used
- Checksum Protected, using a low-overhead checksum algorithm
- Modular, using a 2-stage Message Identifier (Class- and Message ID)

4.4.2. UBX packet structure

A basic UBX Packet looks as follows:

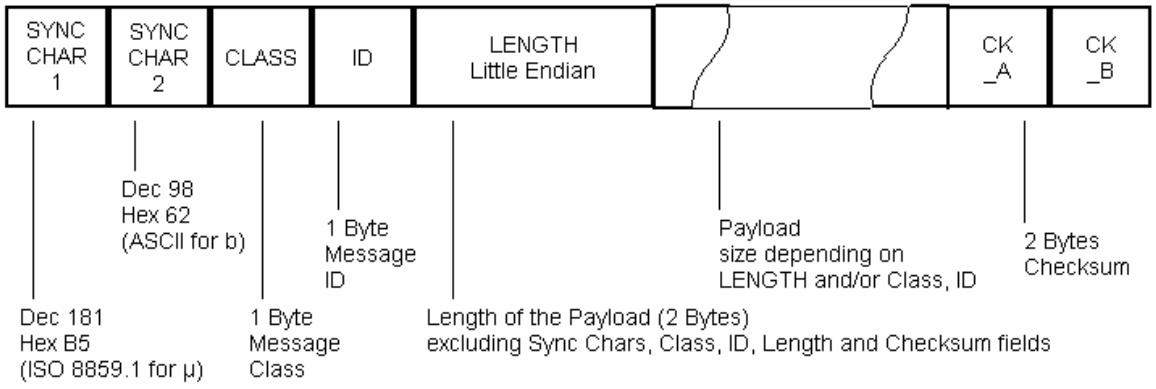

Figure 4.5: UBX protocol framing [12]

- Every Message starts with 2 Bytes: 0xB5 0x62
- A 1 Byte Class Field follows. The Class defines the basic subset of the message

- A 1 Byte ID Field defines the message that is to follow
- A 2 Byte Length Field is following. Length is defined as being the length of the payload, only. It does not include Sync Chars, Length Field, Class, ID or CRC fields. The number format of the length field is an unsigned 16-Bit integer in Little Endian Format.
- The Payload is a variable length field.
- CK_A and CK_B is a 16 Bit checksum whose calculation is defined below.

4.4.3. UBX Class IDs

A Class is a grouping of messages which are related to each other. The following table gives the short names, description and Class ID Definitions.

- **NAV** 0x01 Navigation Results: Position, Speed, Time, Acc, Heading, DOP, SVs used

- **RXM** 0x02 Receiver Manager Messages: Pseudo Ranges, avg C/N0, Channel Status

- **INF** 0x04 Informative Messages: Printf-Style Messages, with Ids such as Error, Warning, Notice

- **ACK** 0x05 Ack/Nack Messages as replies to CFG Input Messages

- **CFG** 0x06 Configuration Input Messages: Set Dynamic Model, Set DOP Mask

- **UPD** 0x09 Firmware Update Messages

- **MON** 0x0A Monitor: Stack Usage, Comunication Status, CPU Load, IPC Status, Task Status

- **AID** 0x0B Navigation Aiding: Ephemeris, Almanac feeds

- **TIM** 0x0D Timing: Timepulse Output, Timemark Results

- **USR** 0x4x SCK Customer Messages

But only the NAVigation measurements and the RXM measurements are required to establish the monitor’s functionality described in the problem formulation.

4.5.Detailed description

To calculate the receiver position using the least-squares method from measurement data from one epoch, quite a few things need to be computed. The m-file recpo_ls therefore calls a number of other m-files during its execution. All the m-files in this computation are made by Kai Borre. The functions will be described briefly, in the following.

Recpo_ls function description

Functionality:

Calculate receiver position from one epoch using least-squares method.

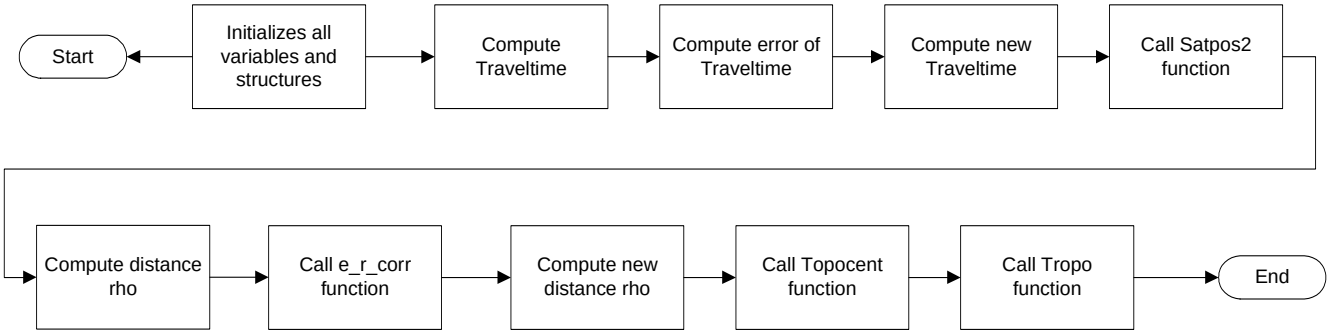

Figure 4.6: The flow diagram of *Recpo_ls* function

The aim of this funtion is the computation of the receiver position from pseudoranges. The input of this function are the pseudoranges, the ephemerides, the number of satellites and the time. The output of this function are the position of the receiver, the Geometric Dilution of Precision (GDOP), the Elevation and Azimuth and the PRN number of satellites.

The first necessary step is to assign the right ephemeris to each measurement.

The second thing needed, is a time correction. The time available in the data set is the receiver time when data is received. This needs to be converted to the time of the data transmission from the satellite. The correction is therefore a function of the signal travel time. Another correction is also needed at this point. That is the error introduced as a function of the earth's rotation. The correction for earth rotation is made by the m-file *e_r_corr*. A time check by *Check_t*.m is also engaged.

e_r_corr function description

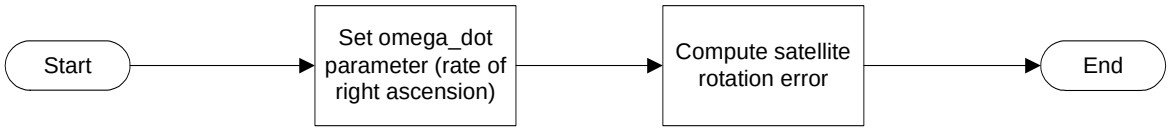

Figure 4.7: The flow diagram of e_*r_corr* function

When the time is corrected, all satellite positions XYZ in a ECEF system, at time of signal transmission, can be calculated, from the information in the ephemerides. This is done by the m-file *satpos2*

Satpos2 function description

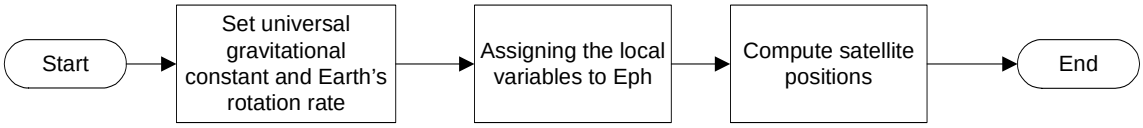

Figure 4.8: The flow diagram of *Satpos2* function

The *Satpos2* function computes the X, Y, Z coordinates at a time t and for given ephemeris. The Satpos2 function takes the ephemeris for a specific satellite, the epoch time as input arguments. As output it returns the coordinates of the satellite for which ephemerides were input, in ECEF format (i.e. X, Y, Z).

The flow diagram for the *Satpos2* function is shown figure 4.8.
The *Satpos2* function starts by declaring constants which in the GPS have specific values. An example, the universal gravitational constant and the Earth's rotation rate are needed for this computation.

After the declaration of constants, the local variables we need from the computation are assigned to Eph.

When the satellite positions are known, their azimuth, elevation and distance can be calculated. This is done in the m-file *topocent*

***topocent* function description**

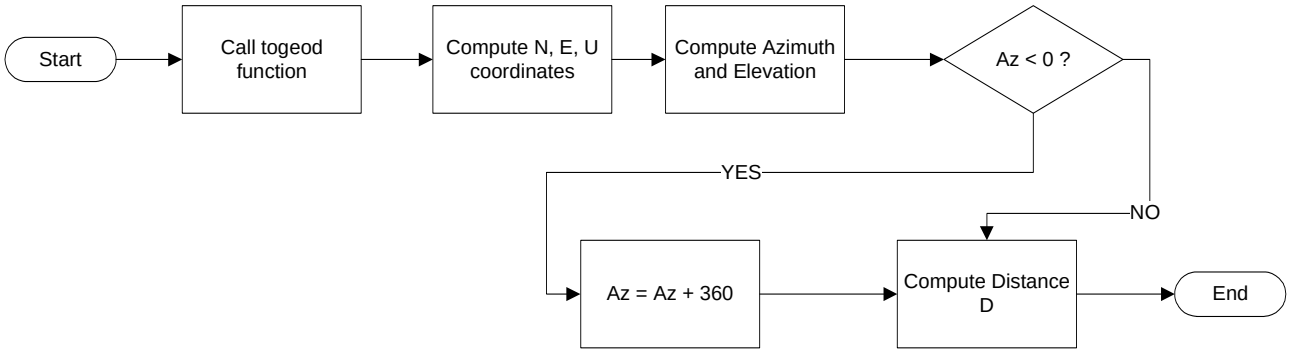

Figure 4.9: The flow diagram of *topocent* function

With information about elevation it is possible to compute a correction for the delay in the troposphere. This is done in the m-file *tropo*, which uses a model of a standard troposphere. No meteorological measurements are made; all parameters are therefore either standard values or set to zero. There will be made no correction for the delay in the ionosphere. It is therefore also set to zero.

***tropo* function description**

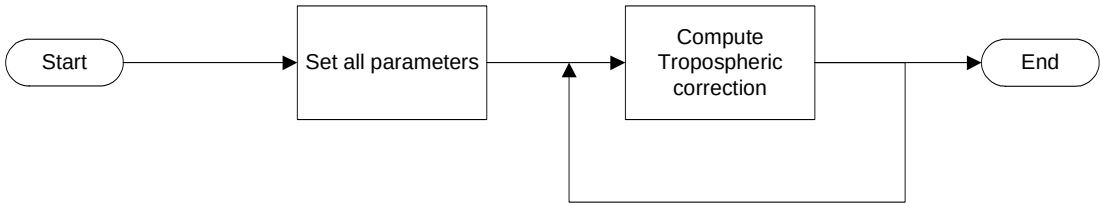

Figure 4.10: The flow diagram of *tropo* function

XYZ2LLA function description

Then, in order to show the position of the GPS receiver to the user, it was chosen to compute the Latitude, Longitude and Altitude from the coordinates of the receiver X, Y and Z.

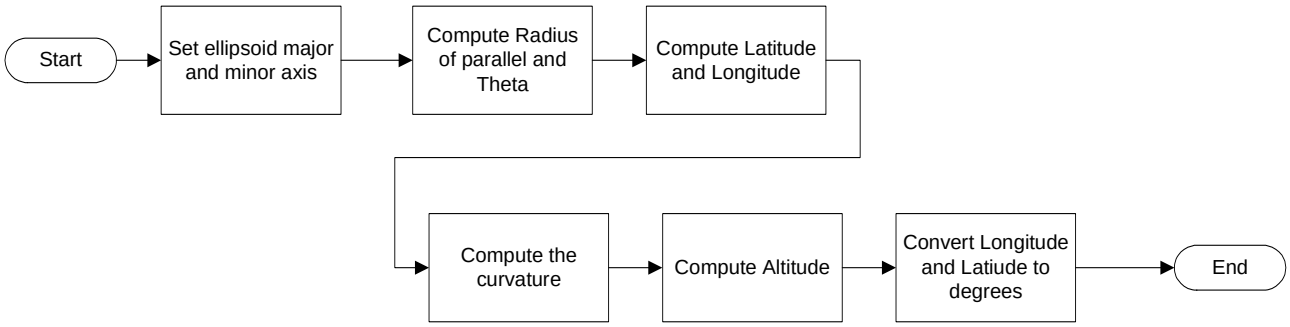

Figure 4.11: The flow diagram of *XYZ2LLA* function

5.Implementation of the GUI

This chapter describes the way the graphical part of the GUI was designed and implemented and which considerations were taken into this process.

5.1.Graphical User Interface

5.1.1. Advantages of GUIs

A major advantage of GUIs is that they make computer operation more intuitive, and thus easier to learn and use. For example, it is much easier for a new user to move a file from one directory to another by dragging its icon with the mouse than by having to remember and type seemingly arcane commands to accomplish the same task.

Adding to this intuitiveness of operation is the fact that GUIs generally provide users with immediate, visual feedback about the effect of each action. For example, when a user deletes an icon representing a file, the icon immediately disappears, confirming that the file has been deleted (or at least sent to the trash can). This contrasts with the situation for a CLI (command line interface), in which the user types a delete command (inclusive of the name of the file to be deleted) but receives no automatic feedback indicating that the file has actually been removed.

But the GUI has become much more than a mere convenience. It has also become the standard in human-computer interaction, and it has influenced the work of a generation of computer users. Moreover, it has led to the development of new types of applications and entire new industries. An example is desktop publishing, which has revolutionized (and partly wiped out) the traditional printing and typesetting industry.

Despite the great convenience of the GUI however, system administrators and other advanced users tend to prefer the CLI for many operations because it is frequently more convenient and generally more powerful1. On Unix-like operating systems, GUIs are actually just attractive, convenient coverings for command line programs (i.e., programs which operate from a CLI), and they rely on them for their operation.

One of the great attractions of Unix-like operating systems is that they have maintained their CLI capabilities while continuing to improve their GUIs, thereby allowing advanced users

to harness the full power of the computer while simultaneously making it easier for beginners and intermediate users. In consequence, the newer versions of Microsoft Windows (such as 2000 and XP) have downgraded their CLIs to marginal roles.

5.1.2. Start with the GUI

At the beginning, the user have no choice and nothing to do to run the graphical User Interface. It was chosen to an “observable” GUI without any interactions between the user and the interface because this GUI is GUI to show to a new user how GPS works.

Thus, the establishment of the monitor is based on a complete understanding of how GPS works. And this conceptual understanding is converted into a comprehensive computation.

The user will see all the important values required to understand the GPS technology. It will be shown the actual status of the system in real time.

It has been chosen to do an information message at the beginning, when the user run the program, like the one shown in figure 5.1. The message allows the user to come back if he has clicked in error in the menu *Start.*

Figure 5.1: The information message at the lauching of GUI

Thus, by the same way, it has been chosen to show information about the GUI itself and some information about the creator of the GUI and the copyright by clicking on the *Help* menu following by the *About* menu

The information of *About* menu is shown figure 5.2.

Figure 5.2: The information message by clicking on *About* menu

5.1.3. Basic window

The basic window is shown in figure 5.3.

As I said before, there is no control area in the basic window because the basic user is not expected to have enough knowledge of GPS positioning.

Having a control area in this window would only make the window more confusing to navigate in which eventually would lead to more confusion for the basic user and making it harder for the user to extract the information needed.

The window will have the very minimum information about GPS positioning to understand it. Most information shown in the window is graphical. There is only little numerical information provided.

As it was discussed before, the basic user is assumed to have little or no understanding about GPS. It was chosen to make the information that is easy to understand like the position of the user and the sky view which have a bigger part of the window than the information about the difference between position computed and position measured or the Dilution Of Precision.

Every graphical user interface needs to get the attention of the user. So as to reach this aim, it was chosen to use colors in the skyview and colors for the different graphics like precision of position or the Signal to Noise Ratio (SNR). The sky view does not have the same array of colors but it was chosen to show the position of the different satellites with their PRN number, which is their identification number.

The text in the window has been grouped by colors to help the user to distinguish between titles, text labels and numerical information from the GPS receiver.

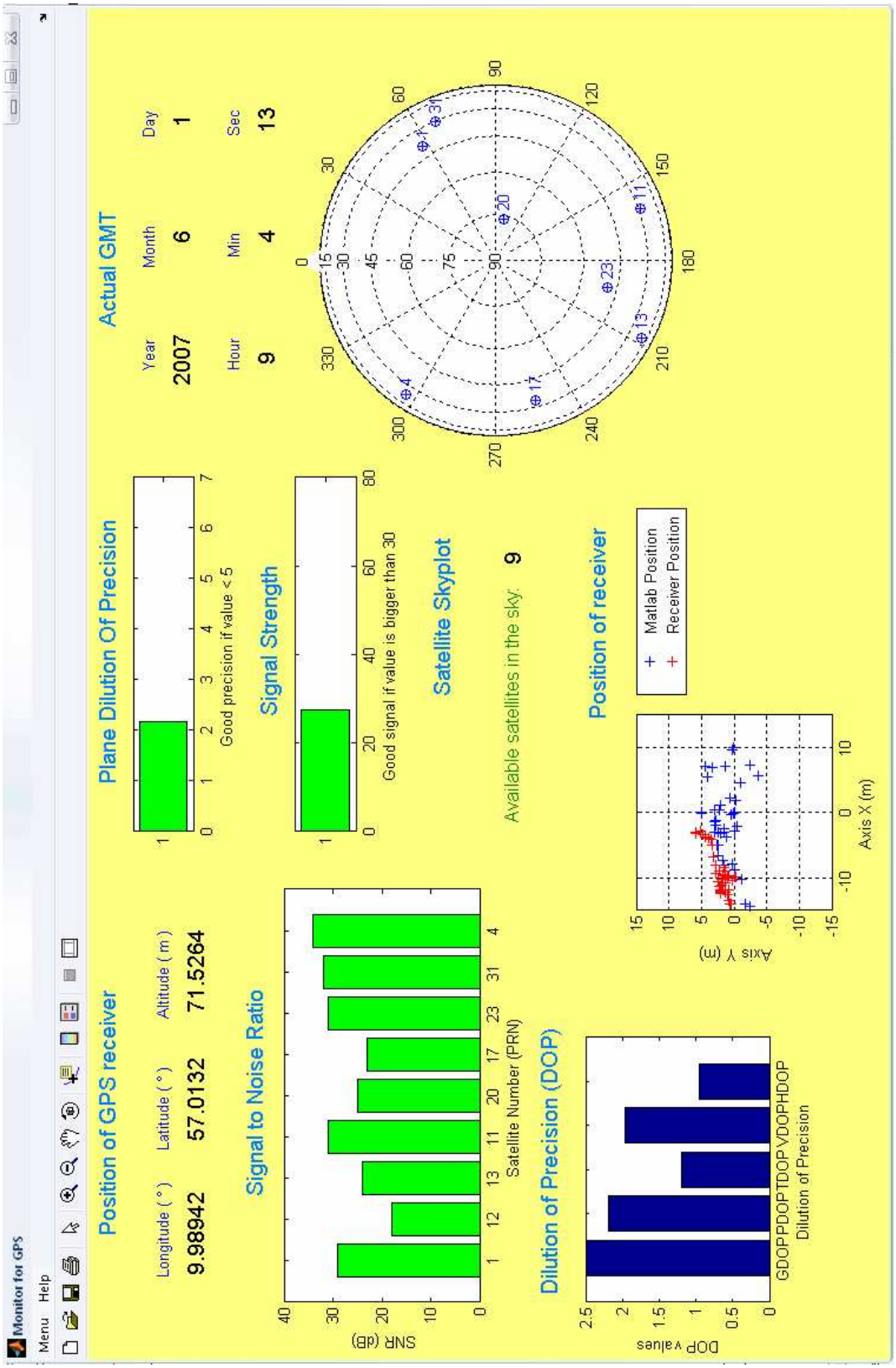

Figure 5.3: The basic graphical user interface window

5.1.4. The menus

The menus are positioned in the upper left side of the window in the way as lots of menus in the windows based systems. It would be a good idea to move this menu to another place, but Matlab does not provide this feature.

The structure of the basic menu is shown in Figure 5.4.

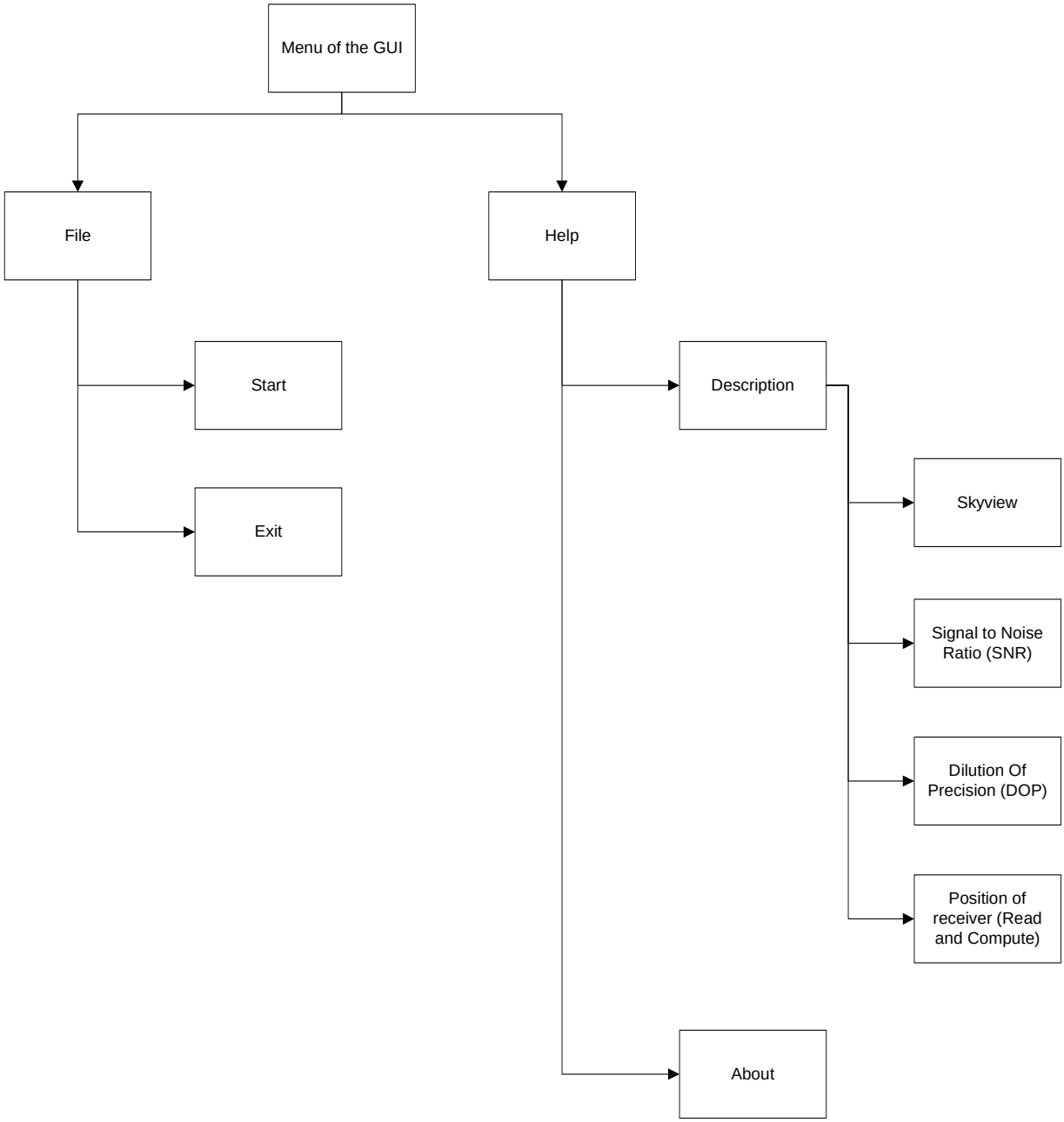

Figure 5.4: The structure of the GUI window menu

The menu consists of two groups of choice:

- *File*
- *Help*

Those groups are very common in graphical user interfaces. Thus, *File* menu allows the user to:

- Start the program in real time
- Quit the program

And the *Help* menu allows to the user to have some descriptions of this interface:

- Description:
  - of Skyview
  - of the Signal to Noise Ratio
  - of the Dilution Of Precision
  - of the position of receiver (Read and Compute)
- About

So the last menu, *About*, is dedicated to some information about the GUI itself, some credits about the creators of the GUI and the copyright statement.

5.2. The position

The most important information for the basic user is to know the position of the receiver.

The position is quantified by two pieces of information:

- the first is about the three text boxes which give the basic user information about longitude, latitude and altitude.
- the second is to show on a graphic the difference between the receiver position and the Matlab position.

Most people know that a position on the Earth can be determined in relation to the Equator and the Greenwich Meridian and consists of three components:

- Longitude
- Latitude
- Altitude

The height is usually given as a number of meters above the mean sea level. And Longitude and Latitude can be given in degrees, minutes and seconds or in decimal degrees.

Thus, to make the information in the three text boxes above the map easy to read and interpret by the user, it was chosen to show the longitude and the latitude in decimal degrees and the height is shown in meters above the geoid which is used as a representation of the mean sea level.

Figure 5.5: The three text boxes showing the coordinates of the user position

Then, it was chosen to show the difference between the position computed and the position measured from the GPS receiver.

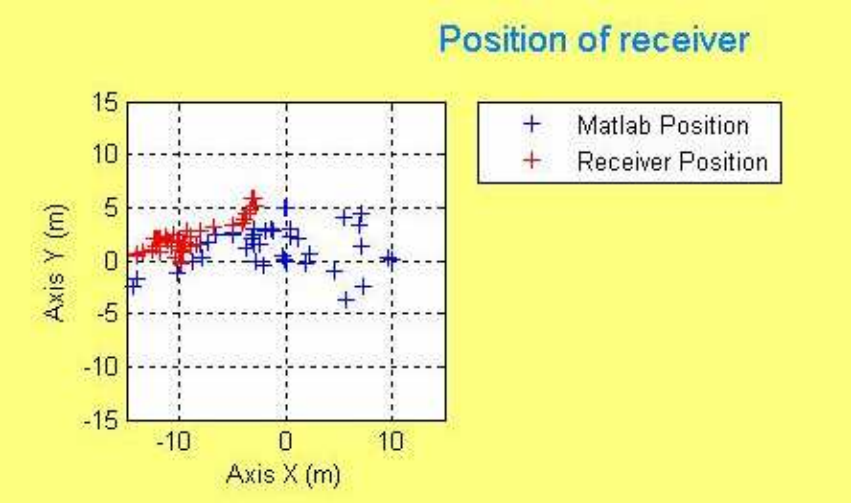

Figure 5.6: This graph shows the receiver position and the Matlab position.

5.3.Satellite positions

In order that the user can better understand the GPS, the satellite positions will be shown by using a skyview. A skyview is essentially a polar plot, where the length of the vector drawn in the polar plot is equal to the zenith distance of the satellite measured in angular units.

To make a skyview for the basic user, it is necessary to do an understandable polar plot. A polar plot with satellite positions is not something that a basic user understands very easily.

Consequently, the best way to make such a polar plot understandable is to relate it to a compass which is known by most people. So the polar plot will be circular but instead of showing N for north direction and E for east direction, it will show 0 degree for north direction in order to be a little bit more precise. This would make the basic user understand that he is in the "middle" of the plot and that he is being informed about the direction to the satellites compared to his own position.

And it was chosen to show the PRN number of each satellite near their position in order to help the basic user to know which satellites are above the horizon.

The skyview with the position of the different available satellites is shown figure 5.7.

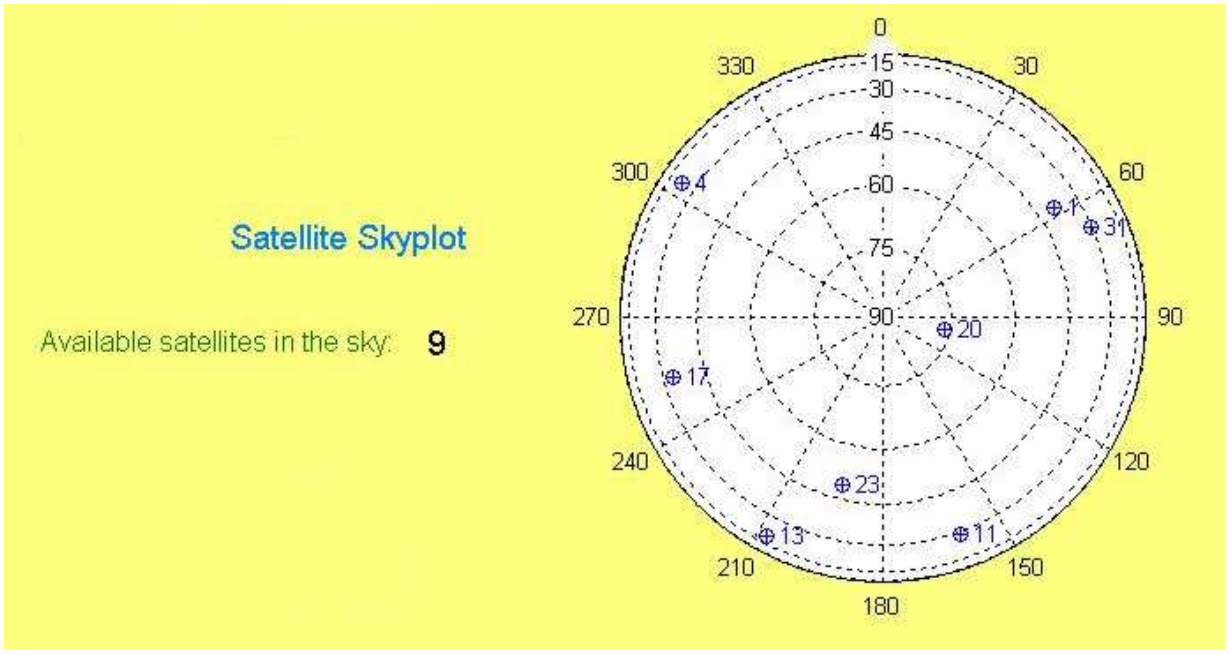

Figure 5.7: The sky view as it appears to the lay user

5.4. The DOP values implementation in the GUI

First, the PDOP value is used as an indicator of what precision to expect from the positioning.

In the interface, the PDOP value will be shown as a horizontal bar graph, going from zero to seven. As the estimated precision of the position is better the lower the PDOP gets the basic user needs to be informed about this problem. Otherwise the basic user would most likely expect the estimated precision of the position to be better the higher the value is. Therefore a text under the graph will explain what a good value is (i.e. less than five).

In addition, when the values is under the value four, the bar will turn green. And if the value is between 4 and 6, the bar will turn yellow while if it gets larger than 6, the bar will turn red. This should make the indication of the precision of position easier to resd and intuitive for the user as yellow is the normal color used for warnings and red is the normal color used for severe warnings.

The graph of the precision of position is shown figure 5.8.

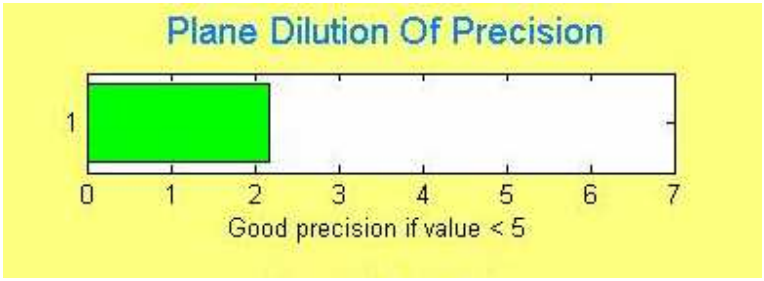

Figure 5.8: The basic user window graph for the Precision of Position

Then, in another graphic, the DOP observables will be presented:

- GDOP
- PDOP
- TDOP
- VDOP
- HDOP

To represent all values, it was decided to use a simple histogram.

The graph of DOP values is shown in figure 5.9.

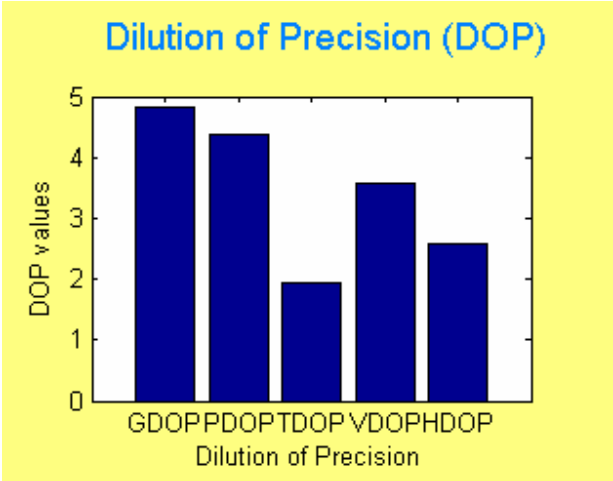

Figure 5.9: The graph of DOP values

5.5. Signal to Noise Ratio (SNR)

In this interface, the Signal to Noise Ratio will be illustrated by two graphics.
The value that relates to the SNR is called “signal strength” and will be shown as a bar going from 0 to 80.
The SNR will be represented in a histogram showing the SNR, in decibels, for each satellite symbolized by its identification, the PRN number.

5.5.1. The Signal Strength

The first thing to take into consideration was to tell the user the meaning of SNR. Consequently, it was necessary to create another name than “Signal to Noise Ratio” in order to the user would be able to understand the phrase “Signal to Noise Ratio”.

Therefore, it was chosen to call SNR :”Signal Strength” which is not really a good term but which helps to the meanig of this expression.

In the graphical representation, it was chosen to make a horizontal bar that shift colors, depending on how good the signal strength is.

Having chosen the representation, it was chosen to use the same colors that many bars to show the strength:

Green: representing a good signal

Yellow: representing a somewhat bad signal

Red: representing a bad signal

The good thing about this changing of colors is the basic user can see immediately the signal strength without especially knowing the value of the signal. Therefore when the bar gets into the yellow or red fields there will be “noise” in the signal compared to when the bar is in the green field.

An example of the signal strength bar is shown in figure 5.10:

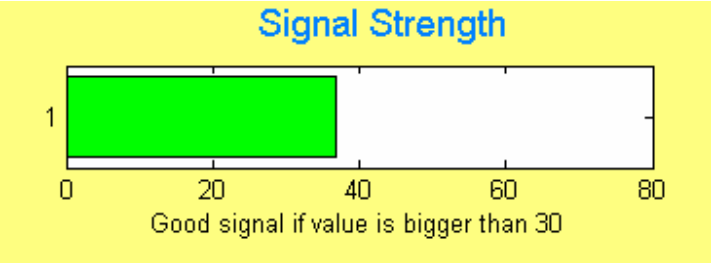

Figure 5.10: The signal strength graph

5.5.2. The SNR

In addition to the signal strength, it was chosen to show the SNR for each satellites. The SNR is represented with a histogram. The SNR shows the reception of the signal for each satellites which we can locate by their PRN number.

The SNR graph is shown in figure 5.11.

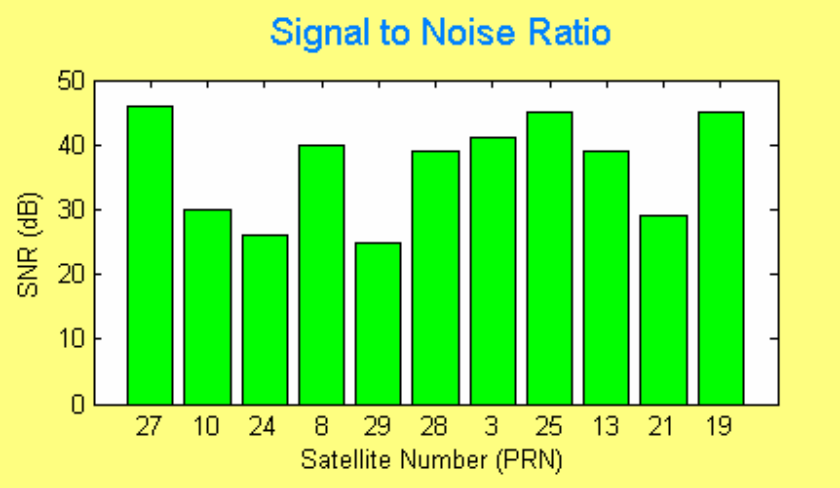

Figure 5.11: The bar diagram of the SNR for each satellite

5.6. The Time GMT

Greenwich Mean Time (GMT) is referring to mean solar time at the Royal Observatory, Greenwich in England. It is often used to refer to Coordinated Universal Time (UTC) when this is viewed as a time zone, although strictly UTC is an atomic time scale which only approximates GMT in the old sense. It is also used to refer to Universal Time (UT), which is the astronomical concept that directly replaced the original GMT.

Local	2007-05-20 11:40:42	Sunday	day 140	timezone UTC+2
UTC	2007-05-20 09:40:42	Sunday	day 140	MJD 54240.40326
GPS	2007-05-20 09:40:42	Week 1428	34856 s	cycle 1 week 0404 day 0

Figure 5.12: Example of clock the 2007-05-20 at 11:40:42

Figure 5.13: The actual GMT in the GUI the 2007-05-19 at 17:41:46

6.Conclusion

6.1.Objectives

According t the introduction, the main objectives were to design and implement a observing graphical user interface for monitoring system for the Ublox Antaris 4 GPS receiver using the Matlab programming environment.

The real-time version of the prototype GPS receiver was developed. And this monitor provides to the user a complete understanding of how GPS works. Thus the algorithms for the monitor system and the GUI design were created.

6.2.Results

Consequently, I have gained through a period of three months knowledge about many subjects in this project.

6.2.1. GPS

I gained an understanding of the basic GPS theory and the algorithms for the GPS. This understanding was primary gained during the description of the theory and the programming of functions directly related to the process of converting raw GPS data and computing useful information from this data.

6.2.2. Communicating with GPS receiver

The Ublox Antaris 4 GPS receiver has some particumarities regarding communication with it. I gained some knowledge in the field of the way binary data is converted to normal data representation in order to interpret the information coming from the receiver.

I have obtained an understanding of how to communicate via Matlab with an Ublox Antaris 4 GPS receiver. This knowledge can be generalized into an overall knowledge of how to communicate with most GPS receivers that allow communication with a computer.

6.2.3. Programming in Matlab

With this project, I gained an understanding of how to program in the Matlab environment and specially with the communication serie Input/Output and the design of a Graphical User Interface.

6.3. Conclusion and perspectives

6.3.1. Conclusion

The overall conclusion is that I succeed in making a workin graphical user interface for a monitoring software to the Ublox Antaris 4 GPS receiver. I used the algorithms of the position calculation, principals of the designing the GUI and the programming environment of Matlab. Nevertheless, there is still much more to learn and improve in my software.

6.3.2. Perspectives

It would be a mistake to confine this work in the Matlab environment. Beacause it would be very interesting to combine the work process with a high level programming language in order to make better improvements for this graphical user interface.

In addition, another important improvement would be to make the application a stand alone application. Therefore it is very impeding for the user of having Matlab installed in the computer so as to run the monitor.

And several functions like changing the colors of the graph, the size of the GUI, the place of the graph, the size of the main window allowing better customization of the GUI would increase the attractiveness of an application.

7.APPENDIX A: Literature

[1] James Bao-Yen Tsui, “Fundamentals of Global Positioning System receivers”, Wiley, 2005

[2] Stephen J. Chapman, “MATLAB® Programming for Engineers”, Brooks/Cole, 2002

[3] Eva Pärt-Enander and Anders Sjöberg, “The MATLAB® Handbook 5”, Prentice Hall, 1999

[4] David Wells, “Guide to GPS positioning”, Canadian GPS Associates 1986, 1987

[5] Global positioning system, Standard positioning service, Signal specification
http://www.navcen.uscg.gov/pubs/gps/sigspec/gpssps1.pdf

[6] Designing GUI Matlab
http://www.intelligent-systems.info/classes/ee509/gui.htm

[7] The MathWorks INC – Creating a GUI with GUIDE Demonstration
http://www.mathworks.com/products/demos/shipping/matlab/CreatingaGUIwithGUIDE_viewlet_swf.html

[8] Using the Serial Port with a Matlab GUI
http://cnx.org/content/m12062/latest/

[9] Scilab 4.1: Help Matlab Function
http://www.scilab.org/product/man-eng-scilab-4.1/index.html

[10] DMAP: UTM Grid Zones of the World
http://www.dmap.co.uk/utmworld.htm

[11] GPS, UTC and TAI Clocks
http://www.leapsecond.com/java/gpsclock.htm

[12] Antaris® 4 Protocol Ublox
www.u-blox.com

[13] Protocol Specification
http://biobug.org/techref/gps-ps1e/GPS.G1-X-00005.pdf

[14] U-blox: Support, Tools and Help
http://www.u-blox.com/customersupport/index.html

[15] GPS Basics, Introduction to the system, Application overview
http://telecom.tlab.ch/~zogg/Dateien/GPS_basics_u_blox_en.pdf

[16] Navstar Global Positioning, Interface Specification
http://www.navcen.uscg.gov/gps/geninfo/IS-GPS-200D.pdf

[17] Wikipedia Global Positioning System
http://en.wikipedia.org/wiki/Global_Positioning_System

8.APPENDIX B: List of figures

Figure 1.1: The basic function of GPS [17] 8

Figure 1.2: Civilian GPS receiver for navigation [17] 9

Figure 1.3: Start display after a successful computation of receiver position [12] 11

Figure 3.1: GPS broadcast signal [17] 18

Figure 3.2: A fundamental GPS receiver [1] 19

Figure 3.3: Explanation of receiver clock offset 20

Figure 3.4: One-dimensional user position 21

Figure 3.5: Two-dimensional user position 22

Figure 3.6: Use three known positions to find one unknown position 23

Figure 3.7: Different Dilution Of Precision 25

Figure 3.8: C/A code generator [1] 29

Figure 3.9: Navigation Message Content and Format Overview [5] 30

Figure 4.1: Overview of the setup 32

Figure 4.2: Relation diagram of the m-files in the system 34

Figure 4.3: flow chart of the overall program 35

Figure 4.4: Messages in Class RXM output status and result data in from the Receiver Manager [14] 39

Figure 4.5: UBX protocol framing [12] 40

Figure 4.6: The flow diagram of *Recpo_ls* function 42

Figure 4.7: The flow diagram of e_*r_corr* function 43

Figure 4.8: The flow diagram of *Satpos2* function 43

Figure 4.9: The flow diagram of *topocent* function 44

Figure 4.10: The flow diagram of *tropo* function 44

Figure 4.11: The flow diagram of *XYZ2LLA* function 45

Figure 5.1: The information message at the lauching of GUI....47

Figure 5.2: The information message by clicking on *About* menu 48

Figure 5.3: The basic graphical user interface window 49

Figure 5.4: The structure of the GUI window menu 50

Figure 5.5: The three text boxes showing the coordinates of the user position 52

Figure 5.6: This graph shows the receiver position and the Matlab position....52

Figure 5.7: The sky view as it appears to the lay user 53

Figure 5.8: The basic user window graph for the Precision of Position 54

Figure 5.9: The graph of DOP values 55

Figure 5.10: The signal strength graph....56

Figure 5.11: The bar diagram of the SNR for each satellite 57

Figure 5.12: Example of clock the 2007-05-20 at 11:40:42 58

Figure 5.13: The actual GMT in the GUI the 2007-05-19 at 17:41:46....58