UNIVERSITY OF NOVA GORICA
SCHOOL OF APPLIED SCIENCES

Use of Global Positioning System as Time Reference in Astrophysics Experiments and its Accuracy

DIPLOMA THESIS

Aleš Bogovič

Supervisor: doc.dr. Darko Veberič

Nova Gorica, 2013

UNIVERZA V NOVI GORICI

FAKULTETA ZA APLIKATIVNO NARAVOSLOVJE

Uporaba sistema globalnega pozicioniranja kot časovne reference v astrofizikalnih eksperimentih in njegova natančnost

DIPLOMSKO DELO

Aleš Bogovič

Mentor: doc.dr. Darko Veberič

Nova Gorica, 2013

to my parents

$\LaTeX$

Acknowledgments

I would like to express my greatest gratitude to doc.dr. Darko Veberič for all his help, guidance and lifesaving advices during the making of this diploma thesis. I am very grateful to all other members of University of Nova Gorica, who helped in creating this thesis in any kind.
I would also like to thank my family and girlfriend for unconditional support during the study.

Abstract

Pierre Auger Observatory is currently the largest cosmic ray observatory ever built. It is placed in Argentina, province of Mendoza, near Malargüe. It is spread over 3000 km^2 and for the detection of cosmic rays primarily uses 1600 surface detectors. In order to keep the data from all these stations synchronized in time, a 1PPS signal delivered by GPS receivers embedded in each of the stations is used. In case when the GPS receivers calculate their position and time (i.e. in "Navigate" mode) the accuracy of the 1PPS signal is <13 ns, but if positions of GPS receivers are already given, its accuracy increases to <8 ns.

A problem occurs when the position setting of the GPS receiver is wrong. In this case accuracy of time decreases, having a negative impact on reconstruction of detected cosmic rays.

Set to wrong coordinates we tried to determine if it is possible to apply a correction on the existing Pierre Auger data. The aim of this thesis was to study behavior of the 1PPS signal when the GPS receiver is set to wrong coordinates relatively to a receiver with correct coordinates, and compare the evolution of time differences of the two 1PPS signals in a longer experiment.

In this thesis the experiment setup, theoretical predictions of 1PPS signal behavior and results of measured time differences between the two 1PPS signals is presented. The position of one receiver is changed in local coordinates in x, y, and z direction. From the measurements a model for time corrections is devised, giving us the difference in the 1PPS signals depending on relative shift of the position set in one of the GPS receivers.

PACS (2010): 96.50.S-, 91.10.Jf, 89.20.Ff, 95.55.Pe, 84.40.Ua, 95.40.+s

Keywords: Global Positioning System (GPS), GPS accuracy, cosmic rays, GPS as timing standard, topography

Povzetek

Pierre Auger Observatorij je trenutno največji kadarkoli zgrajen observatorij kozmičnih žarkov. Nahaja se v Argentini, v provinci Mendoza, blizu mesteca Malargüe. Talni detektor Observatorija se razprostira na 3000 km^2 površine in za detekcijo kozmičnih žarkov uporablja 1600 detektorskih postaj. Da so podatki iz teh merilnih postaj časovno usklajeni uporabljajo signal 1PPS, ki ga nudijo sprejemniki GPS nameščeni v vsako merilno postajo. V primeru ko so sprejeminiki GPS v načinu kjer izračunavajo tako položaj kot čas, je natančnost signala 1PPS $<$13 ns, če pa je njihov položaj že vnešen v njihove registre, pa se natančnost signala 1PPS izboljša na $<$8 ns.

Problem se pojavi, če so položaji sprejemnikov GPS nastavljeni na napačne vrednosti. Takrat se natančnost časovne koordinate zmanjša, kar pa negativno vpliva na rekonstrukcijo detektiranih kozmičnih žarkov.

Namen te diplome je bil preučiti obnašanje signala 1PPS, ko ima sprejemnik GPS nastavljene napačne koordinate in ugotoviti, če se na podalagi informacije o napačni koordinati sprejemnika in pravilni koordinati sprejemnika ta signal utegne popraviti, in sicer po tem, ko je le ta že bil izmerjen.

V diplomi so predstavljena teoretična predvidevanja obnašanja signala 1PPS in rezultati meritev časovne razlike med dvema signaloma 1PPS v odvisnosti od odmikov položaja sprejemnika GPS od dejanske lege lokalno v x, y in z smeri. Iz meritev je izračunan model, ki nam pove za koliko se spremeni signal 1PPS v odvisnosti od relativnega odmika nastavljenega položaja sprejemnika GPS.

Stvarni vrstilec, PACS (2010): 96.50.S-, 91.10.Jf, 89.20.Ff, 95.55.Pe, 84.40.Ua, 95.40.+s

Ključne besede: Globalni Navigacijski Sistem (GPS), natančnost GPS, kozmični žarki, GPS kot časovni standard, topografija

Contents

1 Introduction **1**
1.1 Pierre Auger Observatory . . . 1
1.2 Surface detector . . . 1
1.3 Synchronization . . . 3

2 GPS as Precise Time Source **5**
2.1 Global Positioning System . . . 5
2.1.1 Space segment . . . 5
2.1.2 Ground segment . . . 6
2.1.3 GPS receiver . . . 6
2.1.4 Principle of determining position . . . 7
2.2 1 pulse per second . . . 7
2.2.1 Navigate and Position-hold mode . . . 8
2.3 Position of surface detector stations . . . 8
2.4 Existing measurements . . . 9
2.5 Aims and goals of the thesis . . . 11

3 Measurements **13**
3.1 Method . . . 13
3.2 Preparation . . . 13
3.2.1 GPS module . . . 13
3.2.2 USB communication . . . 15
3.2.3 Connection to oscilloscope . . . 15
3.2.4 Antenna . . . 16
3.3 Sawtooth correction . . . 17
3.4 Setup . . . 19
3.5 Data acquisition . . . 20
3.5.1 GPS receivers . . . 21
3.5.2 Oscilloscope . . . 22
3.5.3 Data acquisition program . . . 22

4 Results **27**
4.1 Theoretical predictions . . . 27
4.1.1 Estimates of time shift . . . 29
4.1.2 Time variance . . . 32
4.2 Measurement results . . . 33

4.2.1 Navigate run 33
4.2.2 Measurement schedule 33
4.2.3 Analysis 34
4.2.4 Z direction 36
4.2.5 X direction 39
4.2.6 Y direction 40
4.2.7 XYZ direction 42
4.2.8 Number of visible satellites 43
4.3 Final Navigate run 45

5 Summary **47**

A Coordinate Systems **49**
A.1 Radius at a given latitude 49
A.2 Local Cartesian coordinates 49
A.3 Transformation from mas to degrees, minutes and seconds 49

B GPS Motorola Oncore UT+ specifications **51**

C Programs **53**
C.1 Processing of GPS messages 53
C.2 Data acquisition 58
C.3 Analysis of oscilloscope traces 62
C.4 Main program 64

Chapter 1

Introduction

1.1 Pierre Auger Observatory

Cosmic rays are high energy particles raining down on us from the outer space all the time, yet they were not discovered until the beginning of the 20th century. In 1938, Pierre Auger and colleagues first recorded extensive air showers initiated by particles with energies estimated to be greater than 10^{15} eV. In 1962, John Linsley recorded a cosmic ray event with the energy of 10^{20} eV. Subsequent studies found more events near and above 10^{20} eV. When the cosmic ray particle reaches the Earth, it collides with a nucleus high in the atmosphere, producing many secondary particles which share the original primary particle energy. The secondary particles subsequently collide with other nuclei in the atmosphere, creating a new generation of energetic particles that continue the process, multiplying the total number of particles. The resulting particle cascade, called an "extensive air shower," arrives at the ground level with billions of energetic particles extending over an area as large as $20\,\mathrm{km}^2$. In order to be able to record and study these kind of events Pierre Auger Observatory was built.

Pierre Auger Observatory is the world's largest high energy cosmic ray observatory. It is positioned in the southern hemisphere in Malargüe, Province of Mendoza, Argentina, where scientists from 19 different countries collect and process data in hope of better understanding the high energy particles coming from the outer space. The project was originally proposed by Jim Cronin from the University of Chicago and Alan Watson from the University of Leeds in 1991, but it took more than a decade before the building of a $3000\,\mathrm{km}^2$ large observatory began in 2002. It become operational in 2004 and was fully completed in 2008. The map of the experiment can be seen in Figure 1.1.

The observatory uses two tools to detect cosmic rays. An array of water Cherenkov surface detectors and four air-fluorescence telescopes which can detect air shower development above the surface array during dark nights with low amount of background light (or moonlight).

1.2 Surface detector

The Surface detector (SD) is in fact an array of 1600 water Cherenkov surface detector units. The grid is triangular with a base of 1.5 km between the nearest detectors. Each detector is cylindrically shaped, 1.3 m high and 3.6 m in diameter. It is filled with 12 000 l of ultra-pure

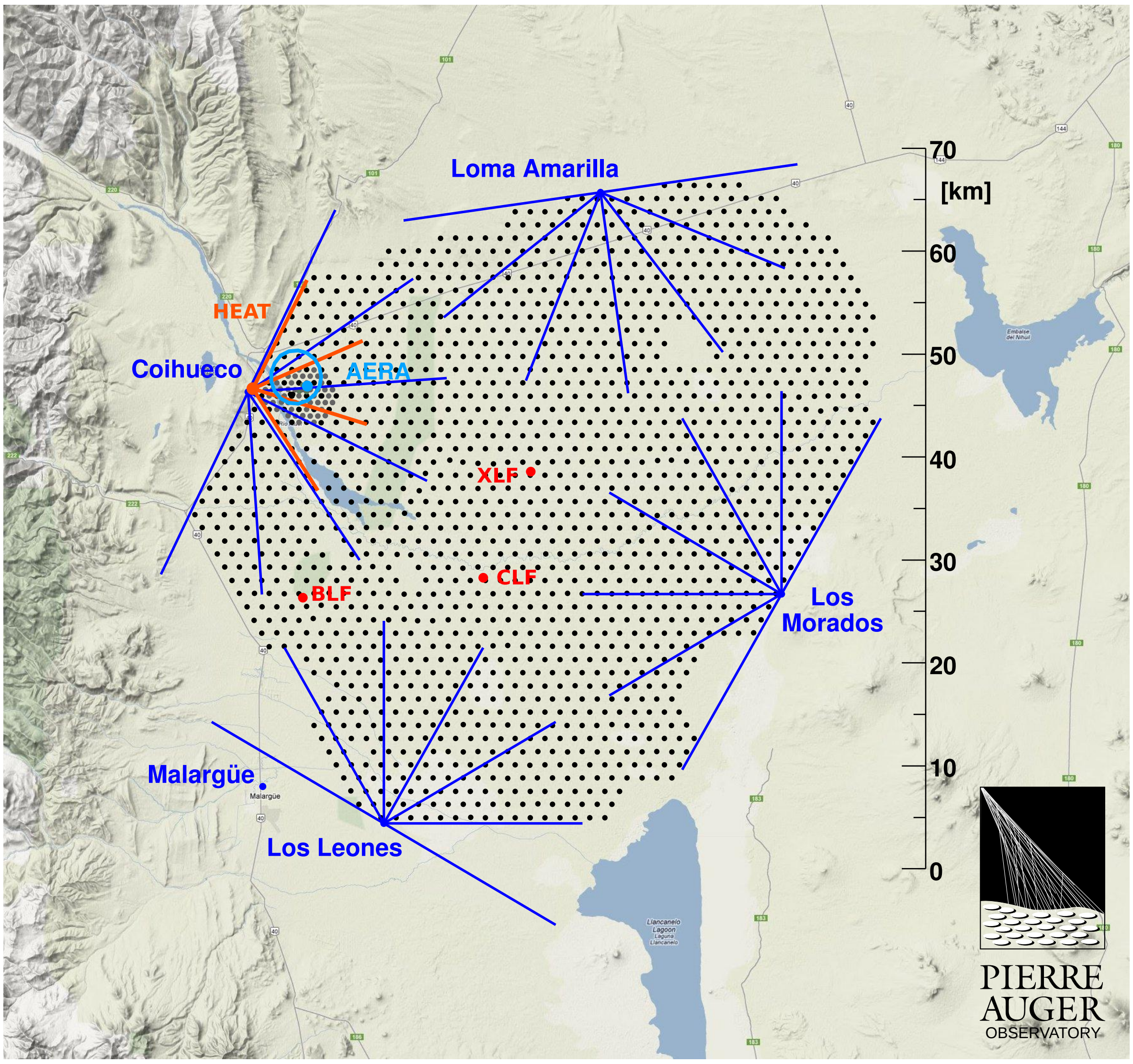


Figure 1.1: Map of the Observatory and its detectors. Each black dot represents a surface detector, blue dots are four fluorescence detector enclosures, each with the 30° field of view of its six telescopes. Red dots represent laser facilities (XLF and CLF) and balloon launch facility (BLF). AERA marks Auger Engineering Radio Array project area.

water produced at the Pierre Auger water plant. Because of the high purity it is expected that water will keep its clarity through entire lifetime of the experiment. Cherenkov light is detected with three Photonis XP1805 photomultiplier tubes placed symmetrically on the surface of the liner pointing downwards into the water. Signals from photomultipliers are processed, triggered and transmitted back to the central system. Everything is controlled with a controller consisting of IBM PowerPC 403 GCX-80 Mhz, with a 32 MB DRAM bank to store the data and executable code, and a 2 MB Flash EPROM for the bootstrap and the OS9 operating system.

Detectors are self sustainable. They get power from two 55 W solar panels, electricity produced is then stored into two 12 V, 105 Ah lead-acid batteries wired in series producing

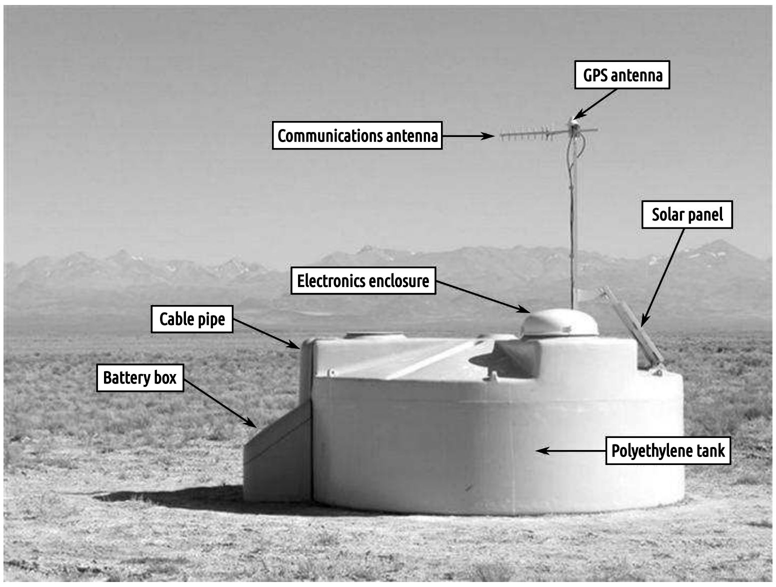


Figure 1.2: Surface Detector and its main parts. There are 1600 of this kind of detectors placed in a triangular grip covering 3000 km^2, with 1.5 km spacing between the neighboring detectors.

a 24 V system. The power is presumed to be present 99% of the operating time. If for some reason the power level falls under a critical level, the whole array can be set to a hibernation mode. Besides the photomultipliers, the controller, and power system, the detector also has a GPS unit and a radio transceiver. The scheme of the whole detector is shown in Figure 1.2.

1.3 Synchronization

1600 is a lot of detectors and therefore there must be a kind of accurate synchronization method providing an universal time source for all the detectors. Time stamps of events must have the same meaning for all events, independent of the stations which recorded the event. This is achieved with a GPS pulse-per-second (1PPS) signal. As the name says, this signal gives a pulse every second, but the main feature of the signal is that it is generated with high accuracy, on the order of 15 ns. The signal is tagged with a GPS time which every SD gets from its GPS receiver via the RS232 interface. One second is of course not even close enough to a good resolution for effective reconstruction of a shower. To achieve better time resolution counts of an internal oscillator at 100 MHz are used to beak it down into smaller units. The 1PPS signal has its own limited accuracy, but the individual pulses can be corrected with a negative sawtooth correction, which is calculated and delivered every second by the GPS receiver. With this corrected 1PPS signal the accuracy of the time stamps is specified to be better than 10 ns RMS.

Chapter 2

GPS as Precise Time Source

2.1 Global Positioning System

GPS stands for Global Positioning System, which is a navigation system based on satellite placed in Earth's orbit. The GPS project started in 1973 by United States Department od Defence (DoD) and became fully operational in 1994. In the beginning it was only a military project, whose primary objective was to provide accurate, real-time positioning for the troops on the battle field. But after South Korean commercial flight carrying 269 civilians was shot down after straying into prohibited airspace of the USSR, the president of USA, Ronald Reagan, issued a directive making GPS freely available for civilian use. Initially, the highest quality signal was reserved for military purposes only, while the signal available for civil use was intentionally degraded (Selective Availability, SA) to accuracy not better than 100 m. In 2000, the US president Bill Clinton ordered SA to be turned off, which increased accuracy down to 20 m [6, 7].

2.1.1 Space segment

The space segment is a constellation of 24 satellites in precise, nearly circular orbits about 20 600 km above the Earth. Several additional satellites are also in the orbit. They are designated as "spares" but are fully operational. So even if failure or planned maintenance takes one or more satellites out of service for some period, the constellation should always contain at least 24 operational satellites. The satellites are arranged in six orbital planes. Each plane is tilted at 55° relative to the equator, to provide polar coverage. Each satellite orbits the Earth twice a day. As a result, at least four satellites are "in view" at any time, from any place on or near the earth's surface. This is significant because a GPS receiver requires signals from at least four satellites in order to determine its location in three dimensions (3D). Each satellite is an autonomous navigation beacon in space. Each one continuously broadcasts low-power radio signals that identify them and provide information about their location in space, as well as system timing and other data. The signals are broadcast using two carrier frequencies in the "L" band of the ultrahigh frequency (UHF) range: L1 (1575.42 MHz) and L2 (1227.60 MHz). Each satellite transmits on these frequencies, but with different ranging codes, which are high-rate pseudo-random (PRN) sequence. These codes were selected because they have low cross-correlation properties with respect to one another. Each satellite generates a short code referred to as the coarse/acquisition or C/A code and a long code

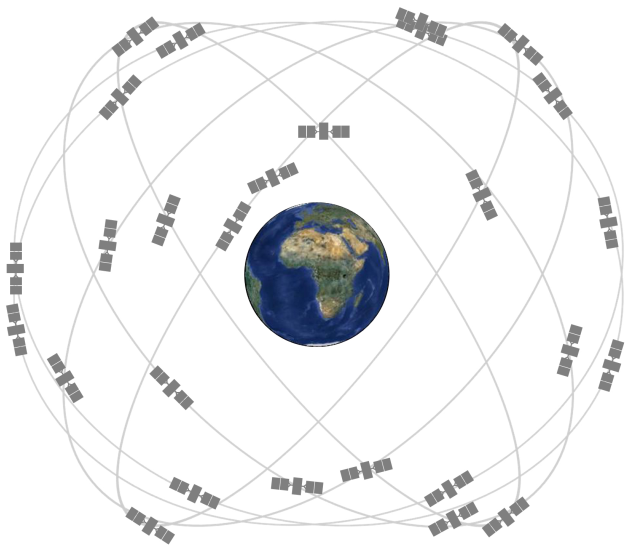


Figure 2.1: The constellation of satellites placed in Earth's orbit 20 600 km above the surface. The inclination angle of the orbits is $\sim 55^\circ$ relative to the equatorial plane.

denoted as the precision or P(Y) code. C/A is available for civil use, where P(Y) is intended for military purposes and for that reason encrypted [6, 7].

2.1.2 Ground segment

The Control Segment includes 12 ground stations. The monitor stations track the navigation signals from all the satellites and continuously send the data to the Master Control Station (MCS) for processing. The MCS computes orbit position projections for each satellite in the constellation, as well as corrections to the satellites' on-board clocks. The MCS sends this updated orbit and clock data to the four ground antenna stations, from which data is uploaded to each satellite three times per day to maintain system accuracy. The ground stations also transmit commands to the satellites for routine maintenance, software updates, and orbit adjustments [6, 7].

2.1.3 GPS receiver

All GPS receivers have a similar core operation. First, they collect the data broadcasted by the satellites, then they measure the signals and at the end they compute position, velocity, and time (PVT). The antenna and a radio-frequency (RF) receiver (often called the RF front end) collect and amplify the incoming very low-power GPS radio signals. The digital signal processor detects (acquires) and tracks the unique signals from multiple satellites. The signal processor also measures various parameters of each tracked signal. In most receivers, an individual "channel" is assigned for each satellite signal. Most modern GPS receivers have at least 12 channels; some have many more. Using the measurements, the navigation processor calculates the PVT solution (often called a position "fix"). Once the solution is known, the receiver displays it in an appropriate form or sends it on to the rest of a larger system in which the receiver may be operating. A simplified functional block diagram of the Oncore receiver with 8 channels is represented in Figure 2.2 [6, 7].

2.1.4 Principle of determining position

The principle of determining position utilizes the concept of one-way time of arrival (TOA) ranging. This concept entails measuring the time it takes for a signal transmitted by an emitter (foghorn, radio beacon, or in our case satellite) at a known location to reach a user receiver. Satellite transmissions are referenced to highly accurate atomic frequency standards on board of the satellites, which are synchronized with a GPS time base. The navigation data transmitted by satellites provides the means for the receiver to determine the location of the satellite at the time of signal transmission, whereas the ranging code enables the user's receiver to determine the transit time of the signal and thereby the satellite-to-user range. This technique requires that the user receiver also contains a clock. Utilizing this technique to measure the receiver's three-dimensional location requires that TOA ranging measurements are made for at least four satellites. Knowing the position of satellites, the time it took for a signal to arrive to the receiver and speed of propagation (speed of light), the receiver can determine its position by means of triangulation.

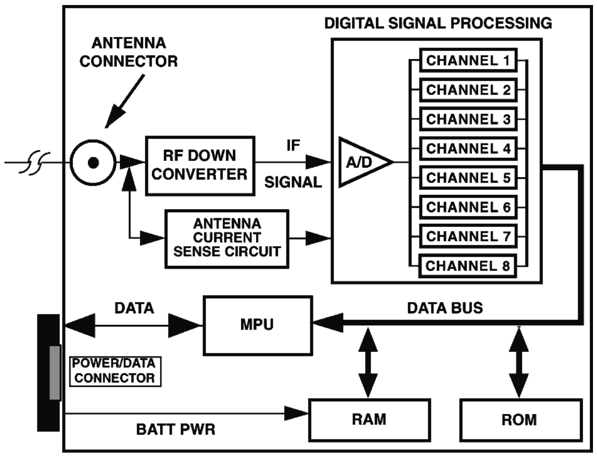


Figure 2.2: A simplified functional block diagram of the Oncore receiver with 8 channels.

2.2 1 pulse per second

1 pulse per second (1PPS) signal is an electric signal that has width less than one second. It has sharply rising and abruptly falling edge, which repeats every second. PPS signals are usually output by frequency standards, radio beacons, atomic clocks, and some GPS receivers. They are used for accurate timekeeping and precise measurements.

As mentioned in the previous chapter, each SD with its own GPS receiver is able to output 1PPS signal. The 1PPS signal of those receivers is defined as 0 to 5 V pulse. Rising time is approximately 20 to 30 ns, with pulse width is 200 ± 1 ms.

2.2.1 Navigate and Position-hold mode

The 1PPS signal output by GPS received is accurate up to <13 ns in respect to GPS second. That is in Navigate mode, in which the GPS receiver, every time when it receives a signal from satellites, calculates its position and time. But Motorola Oncore UT+, which is used in Pierre Auger, also provides a Position-hold mode. In this mode a user must define modules position in advance which results in increased accuracy of the 1PPS signal. Assuming that set position is right accuracy it should improve up to <8 ns regarding to GPS time. Because of this improvement in accuracy, the Position-hold mode was decided to be used in SD [1].

2.3 Position of surface detector stations

In the second half of 2009 it become clear that there were some strange structures appearing in reconstructed data. The first guess was that a reason could be landscape morphology, but after its comparison to digital elevation maps that option was abandoned due to the fact that no similar topological features were present. The other possibility was that there was something wrong with timing accuracy. To test the theory the whole array of SD was switched from Position-hold mode to Navigate mode [2].

After analyzing the data collected in several test intervals when GPS units were switched to Stand alone mode, it become clear that the problem lies in the values which were used for fixed positions. It was clear that in horizontal directions most stations lie within 10 m of set position. In vertical position most of stations lie within 15 m range, but here the population of stations that lie further is much greater. Figures 2.3 represent height differences, where a map of the SD array is populated with circles at the station positions, with circle radius following the absolute magnitude of the height difference. The stations with the height difference within the 5 m are colored green, the stations on the negative and positive side of this range are shown in red and blue, respectively. Looking at the whole SD array, on average the set values seem to be too high in almost all places except in the connected area (green) north and west of Los Morados. A similar plot is made also for the horizontal shifts and can be seen on Figure 2.4, where the arrows are pointing in the direction of the shifts and the arrow sizes are proportional to the square-root of the shift magnitudes in order to emphasize the small shifts whose directions would otherwise not be visualized at all [2].

These errors in set position of GPS receivers cause errors in time calculation. If, for example set horizontal position is lower than actual position of the receiver, receiver will believe it is further away from the satellite than it really is, consequently it will believe that the 1PPS signal arrived earlier and tried to repair the GPS second respectively. The time error caused by the error in a vertical position is presumed to linearly increase with height difference, the time error caused by the error in a horizontal position should presumably cause random and rather small time error for smaller position shifts from origin but it is presumed to rapidly increase after some point.

Therefore measurements of behavior of the 1PPS signal in such conditions must be done. The aim of these measurements should be deriving some kind of method which will allow fixing timing for past events so that they could be reconstructed with greater accuracy [4].

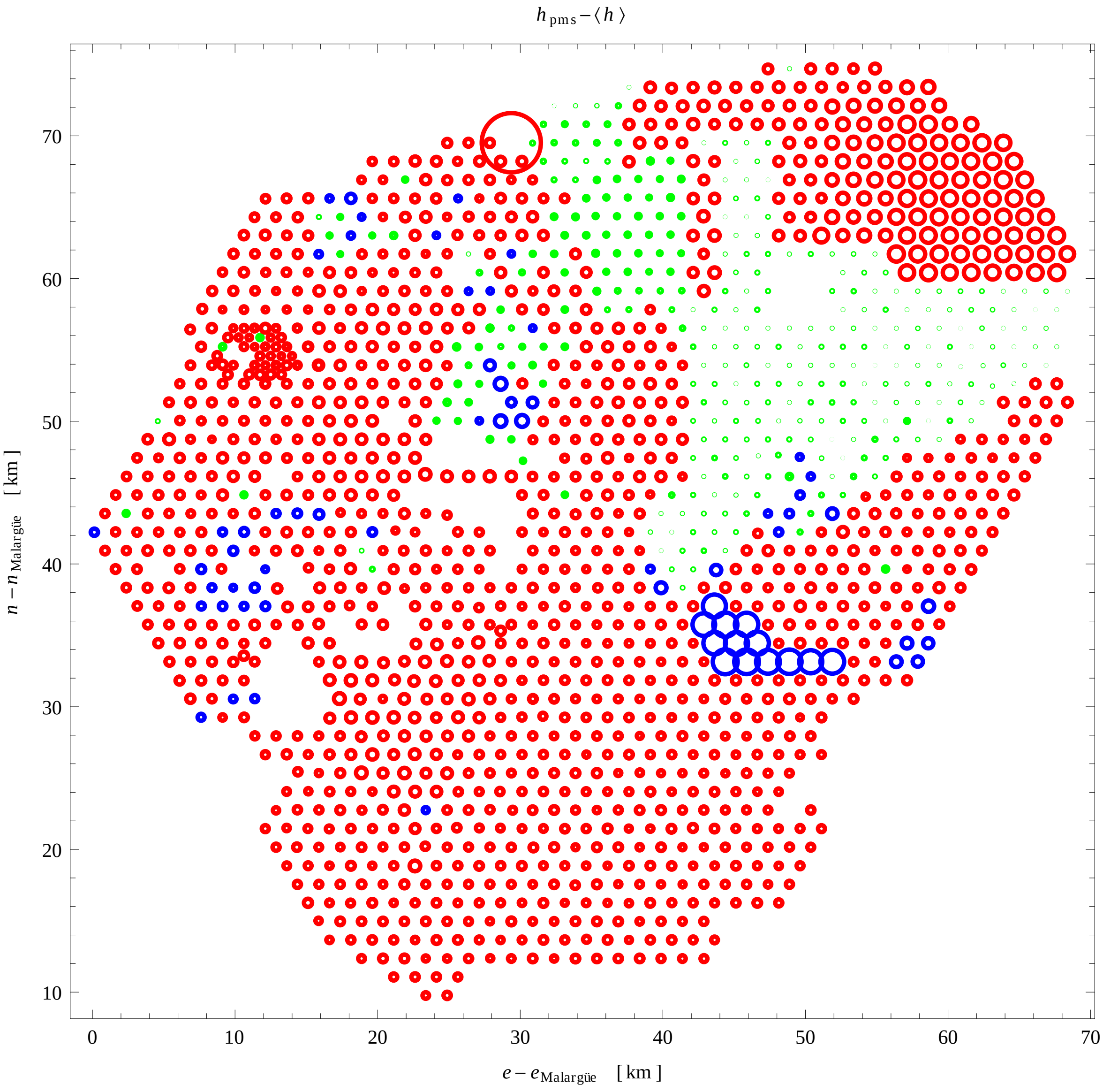

$h_{\text{pms}} - \langle h \rangle$
$n - n_{\text{Malargüe}}$ [km]
$e - e_{\text{Malargüe}}$ [km]
10
20
30
40
50
60
70
0
10
20
30
40
50
60
70

Figure 2.3: Map of height differences $h_{\text{set}} - h$. Size of a circle is proportional to the absolute difference where for the touching circles this amounts to 20 m. Stations with $h_{\text{set}} - h \leq -5$ m are shown in red color and stations with $h_{\text{set}} - h > 5$ m in blue color. The rest is depicted in green. The horizontal positions of the stations are easting and northing subtracted by the corresponding values for the Malargue coordinate system origin [2].

2.4 Existing measurements

Regarding a behavior of the GPS receiver and consequently 1PPS signal, when coordinates in Position-hold mode are wrong, some measurements have already been done. Xavier Bertou from CWR University did an internal Pierre Auger paper with the title *Effect of wrong coordinates on the Surface Detector timing* [11].

His measurements were done with two receivers, where one was set to right position

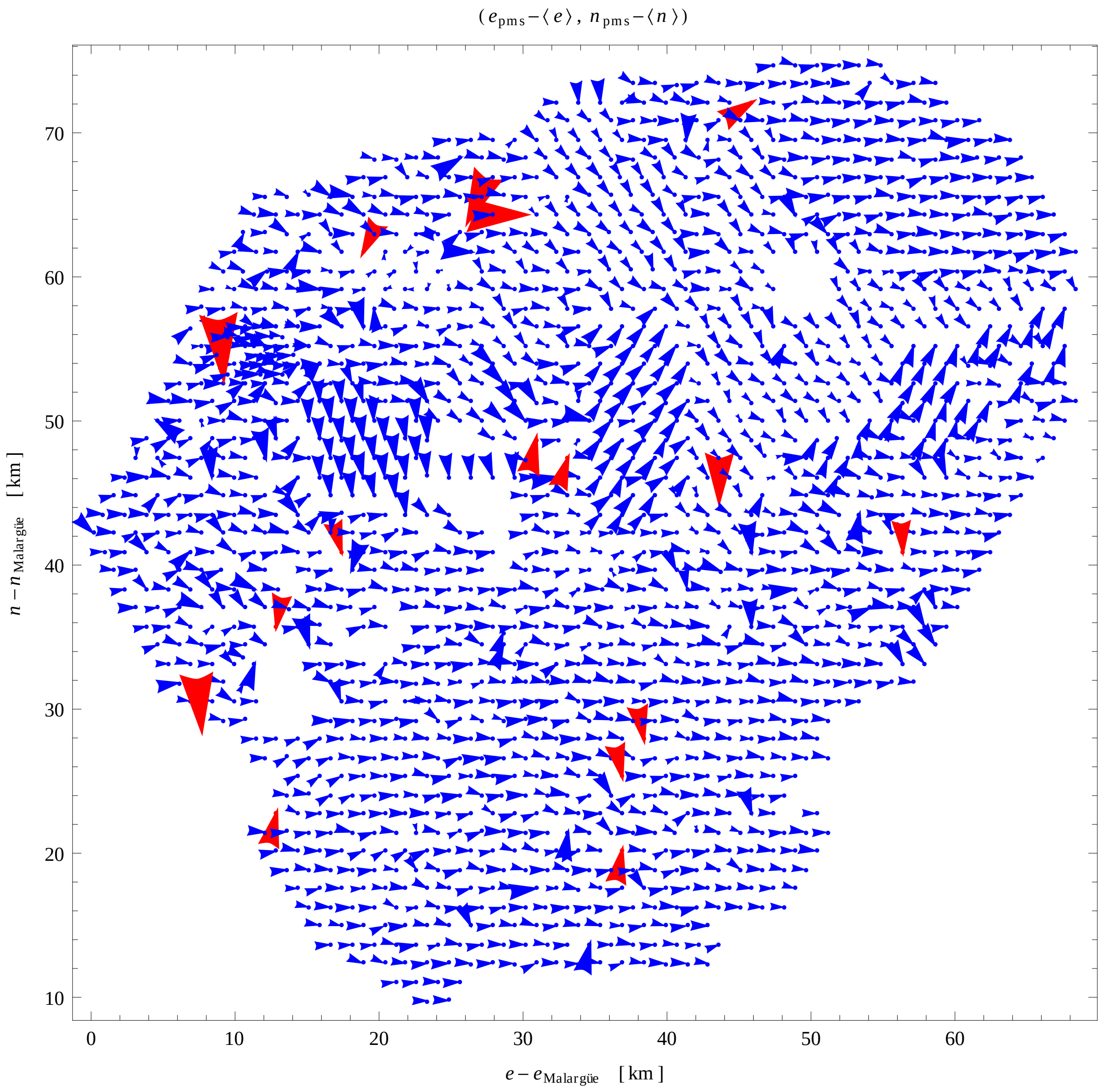


Figure 2.4: Map of horizontal differences ($e_{\mathrm{set}} - e$, $n_{\mathrm{set}} - n$). Size of an arrow is proportional to the square-root of the shift (in order to emphasize small deviations) where the largest arrows correspond to ∼20 m of horizontal distance. Stations with shifts larger than 10 m are shown in red color and the rest is depicted in blue. The horizontal positions of the stations are easting and northing subtracted by the corresponding values for the Malargue coordinate system origin. Note the locally ordered structure of shifts that are probably related to the surveying campaigns [2].

and another to wrong position, where position of the second was systematically changed. Measurements were done in one hour intervals, where only last part of a signal was saved, due to space issues. This results in reduced accuracy, comparing if entire signal would be traced.

Altitude changes have been measured with up to 300 m altitude change with 20 m step.

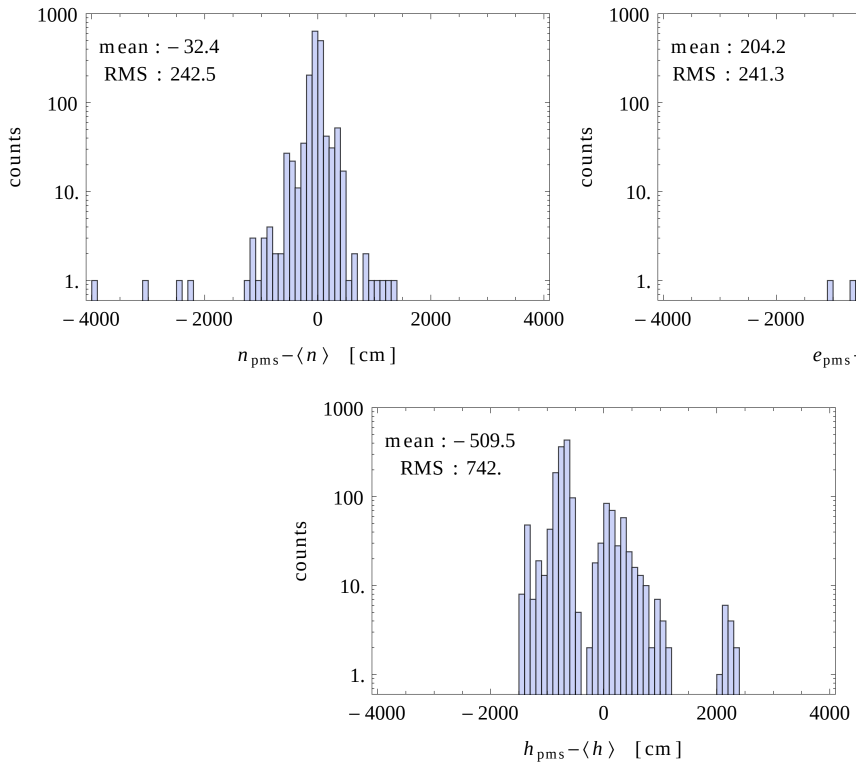


Figure 2.5: Distribution of differences $n_{\mathrm{pms}} - \langle n \rangle$, $e_{\mathrm{pms}} - \langle e \rangle$, and $h_{\mathrm{pms}} - \langle h \rangle$ over the stations. The largest offenders are not shown. *pms* is Pierre Auger database with positions off all SD stored inside [2].

From the analysis of the data he concluded that a general shape of histograms of Δt does not change much with change of altitude, but average time delay clearly shifts. These shifts follow an almost perfect linear dependency of 2.19 ± 0.05 ns/m.

For easting and northing, 20 runs of one hour were done with the easting (and then the northing) moved up to 400 m by steps of 20 m, and then moved up to 100 m in steps of 5 m. He concluded that effects of northing and easting are dependent and evolve with movement of satellites in constellation. It is a systematic effect, but he claims that without data about satellites used and where they were, it transfers into a statistic error in timing. This error mean is zero, but with much larger sigma than for correctly located stations. While negligible up to 50 m, it starts to dominate the time difference at larger distances. The proposed extra timing error, to be used as an extra contribution to the variance model, for an error d in coordinates above 50 m, was

$$\frac{\sigma_t}{\mathrm{ns}} = 0.7 \left(\frac{d}{\mathrm{m}} - 50 \right) - 10^{-3} \left(\frac{d}{\mathrm{m}} - 50 \right)^2 . \tag{2.1}$$

2.5 Aims and goals of the thesis

The time stamps acquired from GPS receivers 1PPS signal are used for reconstructing showers, where time coordinates should be as accurate as possible. If the measurements would

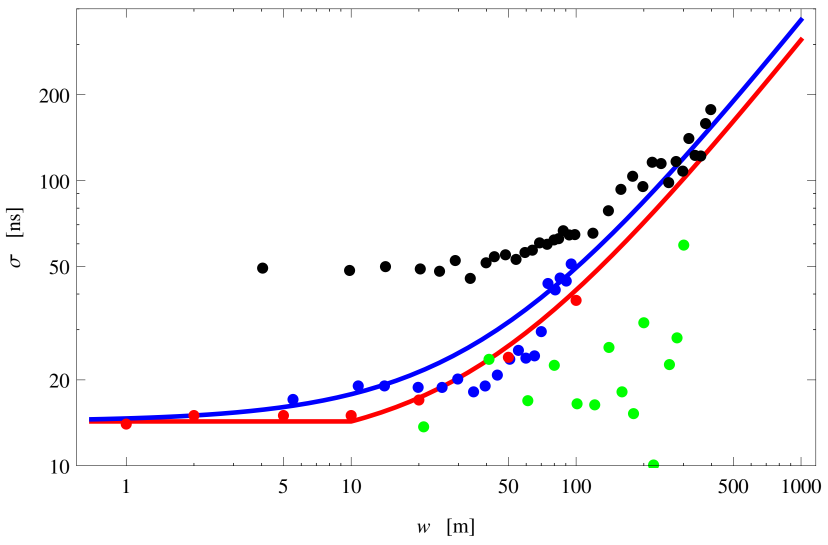


Figure 2.6: Standard deviation of time. Measurement points are from the Malargüe test tank with horizontal displacement with one Gauss fit (black), two Gauss fit (blue), vertical displacement (green), and measurement with horizontal displacement performed at CWRU (red). The red line is the parametrization for $w > 10\,\mathrm{m}$ proposed from CWRU data and the blue line is a rough fit of N in linear form in Eq.(4.24) with constant term of 14.3 ns.

just started, it would be easy to simply change fixed position to right one or set GPS receivers to Stand alone mode until new positions can be acquired. But unfortunately this can not be done in this case. The experiment has been in operation for more that 10 years now and this problem with timing has been decreasing accuracy all the time. It would be ideal if researchers could fix errors in time stamps for past measurements, knowing only right position of SD and the position on which receiver was set.

Measurements of behavior of 1PPS signal when set positions are wrong exist, but those measurements are done separately for x, y and z axis. Effect when all three coordinates are set wrong are not measured and analyzed.

This thesis has two aims. First aim of this thesis is to better determine behavior of 1PPS signal, when set position of a receiver is wrong for $\Delta x, \Delta y, \Delta z$ separately and when they are combined. Second aim is to determine if it is possible to tell for how much 1PPS is "off" knowing only set position and real position of a receiver, so that timing of events in can be repaired and reconstructed with greater accuracy.

Chapter 3

Measurements

3.1 Method

As we learned from the previous section, there is a problem in measured events time precision, due to a wrong calculation of time stamps in GPS receivers. That error happens because receivers are set in Position hold mode, where fixed position is wrong. In order to improve time accuracy of already taken measurements it would have to be possible to repair time knowing the correct position and the set position, which was wrong.

Therefore an experiment was set, where two GPS modules, identical to those used in the Pierre Auger experiment, are used to simulate the situation in Pierre Auger. Both GPS receivers are set in Position hold mode, where one is set on the correct position and others position is being changed in a periodic fashion. The 1PPS signal from modules is being collected with an oscilloscope every second. All three devices, both GPS receivers and the oscilloscope are connected to computer, which sets and alters all needed parameters, collects data from GPS receivers and wavefronts from the oscilloscope.

By comparing difference in 1PSS signals gathered, one should be able to understand the behavior of a 1PPS signal when the set position is wrong and ultimately be able to determine error in 1PSS signal after it has been used for time measurement.

3.2 Preparation

3.2.1 GPS module

The first step was to prepare GPS modules. In the Pierre Auger experiment they use Motorola Oncore UT+. That is an eight-channel receiver. It operates on 5 V DC and communicates to other devices through TTL logic. It can be seen in Figure 3.1. Because modules did not have any connection to the computer yet, it was decided to use a TTL to USB converter, so that communication through USB port would be possible. The USB interface was chosen simply because of the fact that most now day computers does not have RS232 communication port anymore. The converter being used is ELV Mini-USB-UART UM2102 module. It was chosen for its compact design and low cost. The converter can be seen in Figure 3.3.

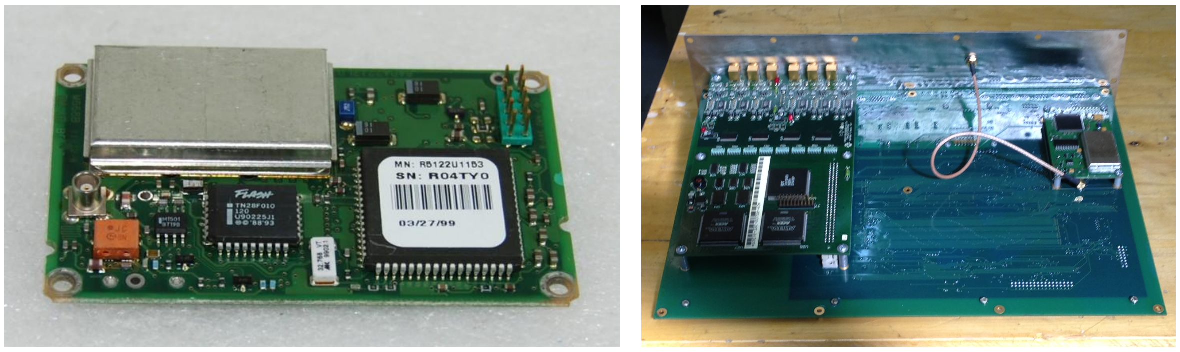
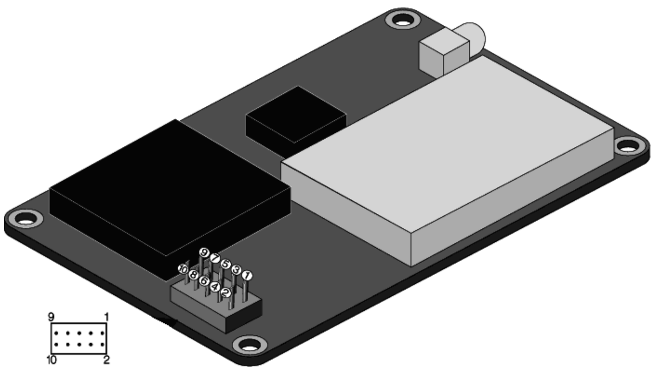


Figure 3.1: *Left:* Motorola Oncore UT+ receiver module board. *Right:* Opened and flipped case of the SD station unified board where the GPS module (right) is mounted below the main board (holding embedded computer) next to the front end board (left) which digitizes signals from the three photomultipliers.



#	Sig.	Description
1	BAT	External backup power
2	PWR	+5 V main power
3	GND	Ground (receiver)
4	VPP	Flash programming voltage
5	RIN	RTCM input
6	1PPS	One pulse/second signal
7	RTN	One pulse/second return
8	TXD	Transmit 5V logic
9	RXD	Receive 5V logic
10	RTN	Transmit/receive return

Figure 3.2: Pin numbers and description of pin functions of Motorola Oncore UT+ GPS receiver.

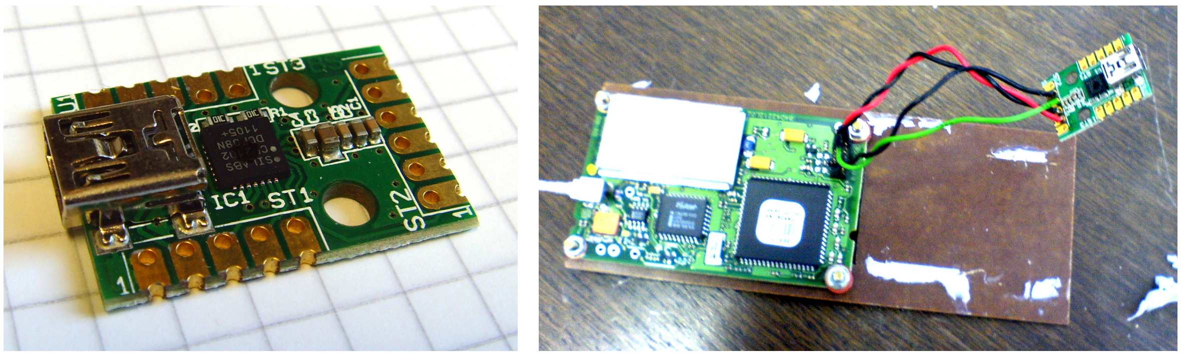


Figure 3.3: *Left:* ELV Mini-USB-UART UM2102 module. The size of squares on the background paper is 5×5 mm. *Right:* Attachment of UM2102 to the GPS module.

3.2.2 USB communication

GPS module and a converter are connected in a RS232 serial null modem cable wiring fashion, with loop back handshaking also knows as reverse handshaking. Why use reverse handshaking? A simple null modem cable has simple wiring. Transmit pin of the first connector is connected to receive pin of second connector, and the other way around for receive pin. There are of course ground pins left which are connected to each other. This kind of wiring can be used with no problem if none of devices checks Data set ready or Data carrier detect inputs. These signals normally define the ability of the other side to communicate. As they are not connected, their signal level will never go high. This might cause a problem. The same holds for the Request-to-Send/Clear-to-Send (RTS/CTS) handshaking sequence. If the software on both sides is well structured, the Request to send output is set high and then a waiting cycle does not start until a ready signal is received on the CTS pin. This causes the software to hang because no physical connection is present to either CTS pin to make this possible. In order to avoid this kind of scenario, RTS is connected to CTS and Data-Set-Ready (DSR) and Data-Carrier-Detect (DCD) pins are connected to the first pin – Data-Terminal-Ready (DTR). The scheme of RS232 connector and scheme of wiring can be seen in Figure 3.5.

In Figure 3.2 you can see a pin layout and pin functions for Motorola Oncore UT+ board and in Figure 3.4 you can see pin layout and pin functions for UM2102 module. Only ST2 segment of conection pins is used for connection to Motorola. Power pin (2) on Motorola board is connected to power pin (10/9) on UM2102 module and ground pin (3) on Motorola board is connected to ground pin (8/7) on UM2102 module. Transmit pin (8) on Motorola board is connected to receive pin (6/5) on UM2102 module and receive pin (9) on Motorola board is connected to transmit pin (4/3) on UM2102 module. On Motorola board only these connections are made, while on UM2102 module connection was made also on segment ST3 between RTS pin (6/5) and CTS pin (4/3) made. This connection is so-called reverse handshake.

3.2.3 Connection to oscilloscope

From Motorola Oncore board the 1PPS signal is transmitted from pin 6 (Figure 3.2). In order that a 1PPS signal could reach oscilloscope, a kind of cable connection had to be made. In such precise measurements, we are talking in order of nanoseconds, it was very important that the cable had very good defined delay, otherwise the difference in signals would be without meaning, due to uncertainty in cable delay. Oscilloscope already had four available female BNC connectors, therefore it was straightforward decision to connect output of 1PPS signal to another female BNC plug. At the end, GPS receiver was connected to oscilloscope with male-male BNC cable. Both cables were picked so that they had the same delay. At first 8 ns coaxial cable was used, but it produced quite a lot of disturbance in signal, because some of the signal was reflected at the both ends of the cable. Effects can be seen on Figure 3.7. After changing to 1 ns lemo cable, those bumps vanish or are reduced to the scale that does not affect out measurements. The final signal captured using 1 ns lemo cable can be seen in Figure 3.7.

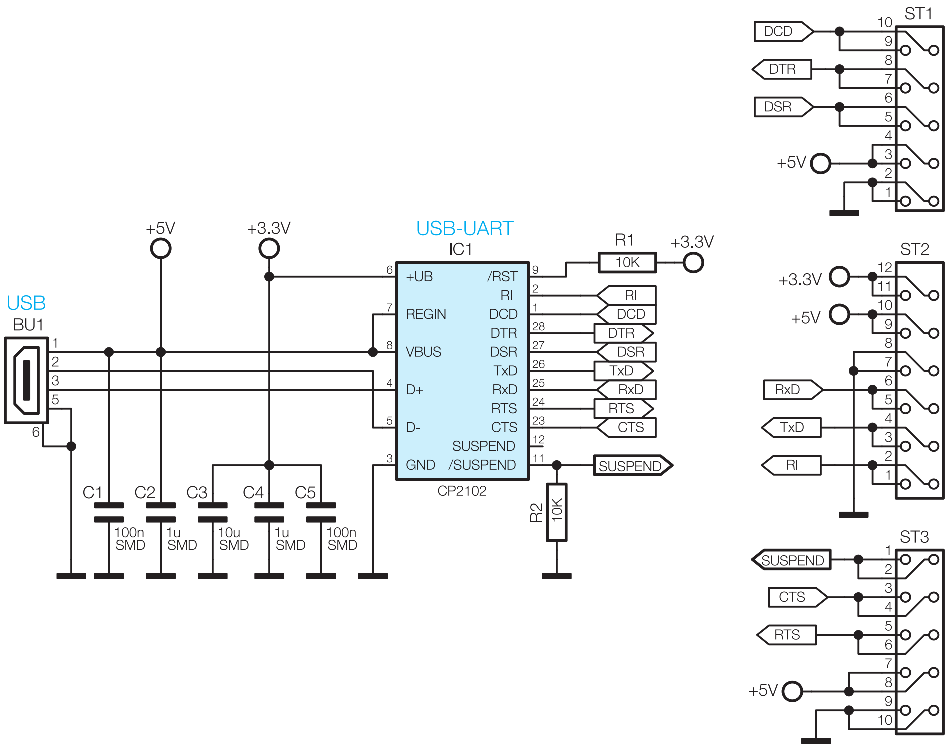


Figure 3.4: UM2102 board and pin diagram [12].

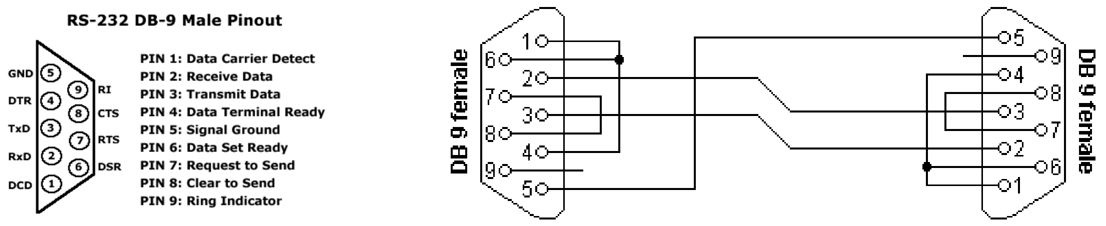


Figure 3.5: *Left:* Scheme of RS232 connector [9]. *Right:* Scheme of RS232 serial null modem cable wiring, with loop back handshaking [8].

3.2.4 Antenna

Antennas for both GPS receivers came together with receivers, therefore choosing the right antenna was already solved. All that was left to do regarding antennas, was to place them somewhere high enough, so that all possible satellites would be visible at all time. If some parts of a sky were blocked by a building or something similar, then satellites in that part of the sky would not be visible to a receiver. That would result in much lower accuracy and our setup would be different compared to that in the Pierre Auger, where the antenna is placed on a pole above the tank.

In order to avoid problems described above, both antennas were placed on a pole, which was placed and fixed, so that it was immobile, on a window shelf. The result was that the top of the pole, with antennas attached to it, was approximately meter above roof level. The setup can be seen on Figure 3.8. This setup is placed in the university building in Rožna

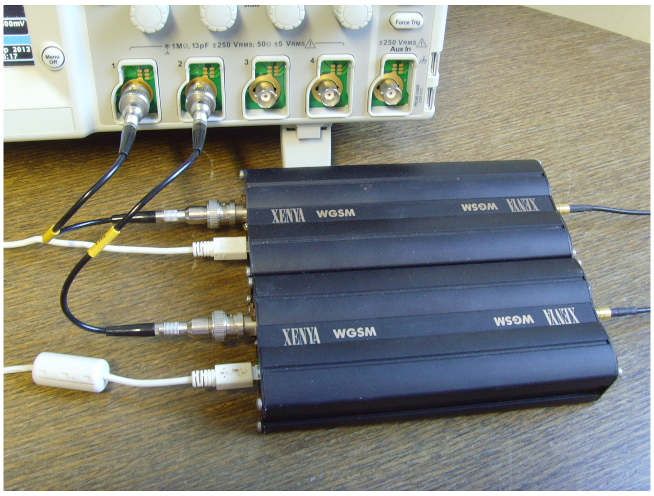


Figure 3.6: Final package in a reused housing of some other device. White cables are the USB connection to the computer and the black coaxial cables are leading to the oscilloscope (partially seen above).

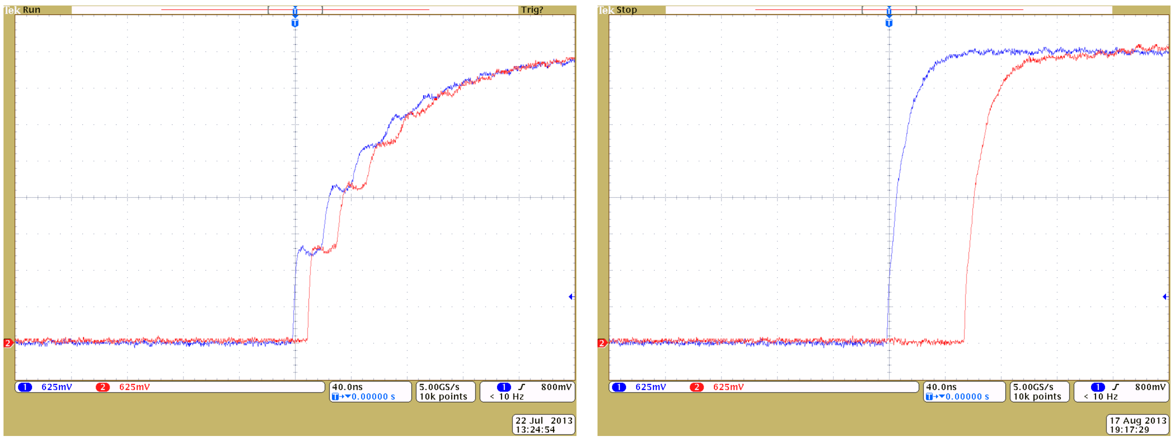


Figure 3.7: *Left:* Signal captured on oscilloscope connected to GPS receiver with a 8 ns coaxial cable. One can clearly see perturbations in a signal curve, which should be straight. *Right:* Signal captured on the oscilloscope connected to GPS receiver with 1 ns lemo cable. There are still perturbations, but they are so small that can be neglected.

dolina, Vipavska cesta 13. Map with exact position can be seen in Figure 3.9. Exact GPS position where antenna was placed is $\varphi = 45°56'32.5''$, $\lambda = 13°38'20.9''$ and $h = 147$ m.

3.3 Sawtooth correction

The full timing capabilities of the UT+ can only be reached if the 1PPS output is corrected for what the UT+ user manual calls the "granularity of the internal oscillator". In other words, the 1PPS output can only be issued on the rising edge of its local oscillator. Since this oscillator has a frequency of 10 MHz, the "granularity" of the 1PPS output is 100 ns. The negative

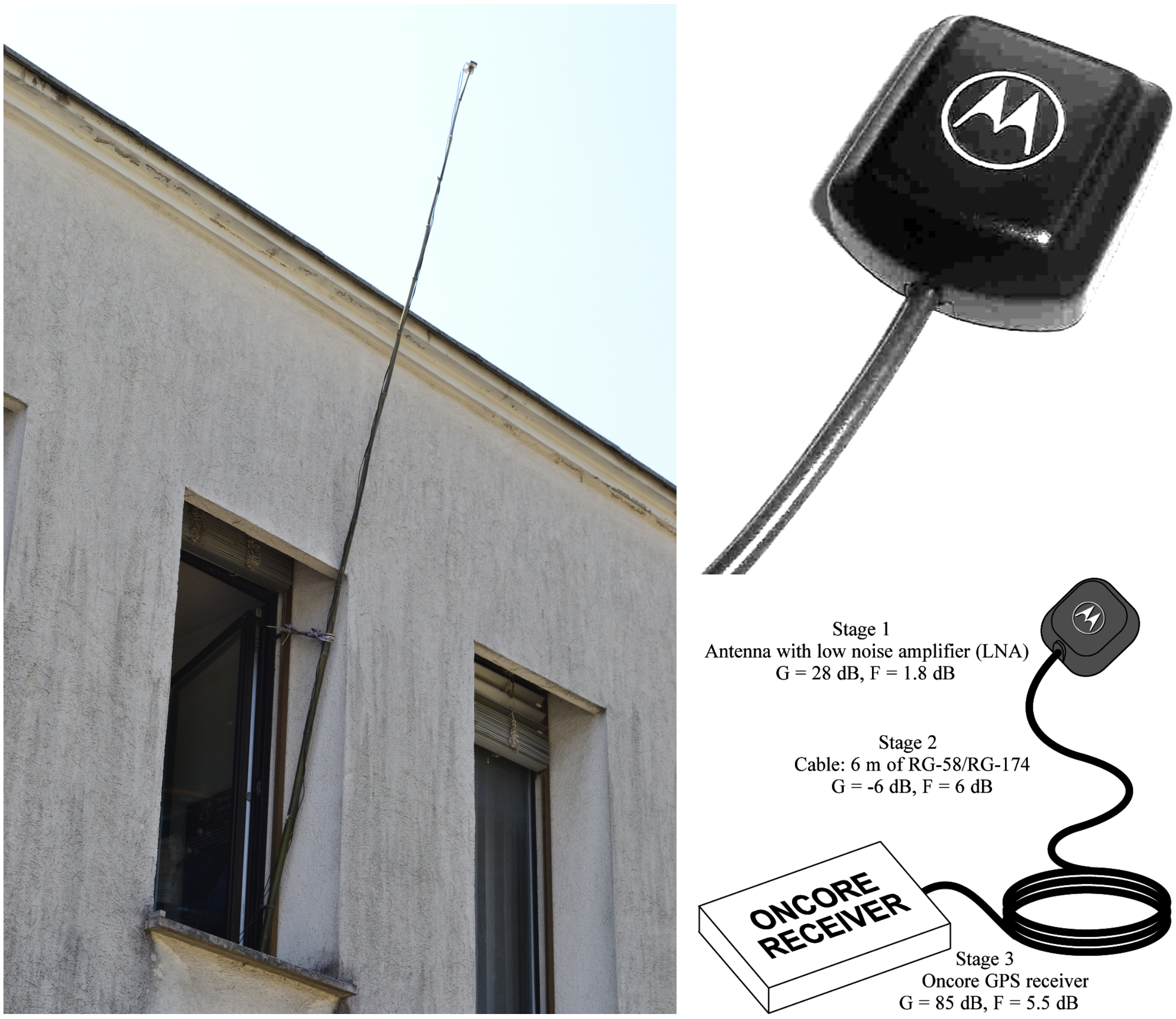


Figure 3.8: Placement of antenna out of an office window at the University of Nova Gorica building in Rožna Dolina.

sawtooth Δt^{saw} is simply the estimation by the receiver of the deviation between its best estimation of GPS time t and the effective position of the 1PPS at time t^{1PPS}, which differ because of the "granularity" of the oscillator. It comes from the above that the negative sawtooth is a signed byte, that theoretically oscillates between -50 and 50 ns. In practice, values between -52 and +52 ns can be observed. The negative sawtooth issued by the receiver is a prediction: it is issued within around 50 ms after an occurrence of the 1PPS signal but it applies to the next occurrence of the 1PPS. To summarize, the negative sawtooth is the amount of time that will elapse between the true estimation of GPS time and the 1PPS. If it is positive, the 1PPS is late with regards to GPS time; if it is negative, the 1PPS is ahead of GPS time,

$$t_n = t_n^{\text{1PPS}} - \Delta t_n^{\text{saw}}. \tag{3.1}$$

In Figure 3.10 all the negative sawtooth are potted with a positive sign [10].

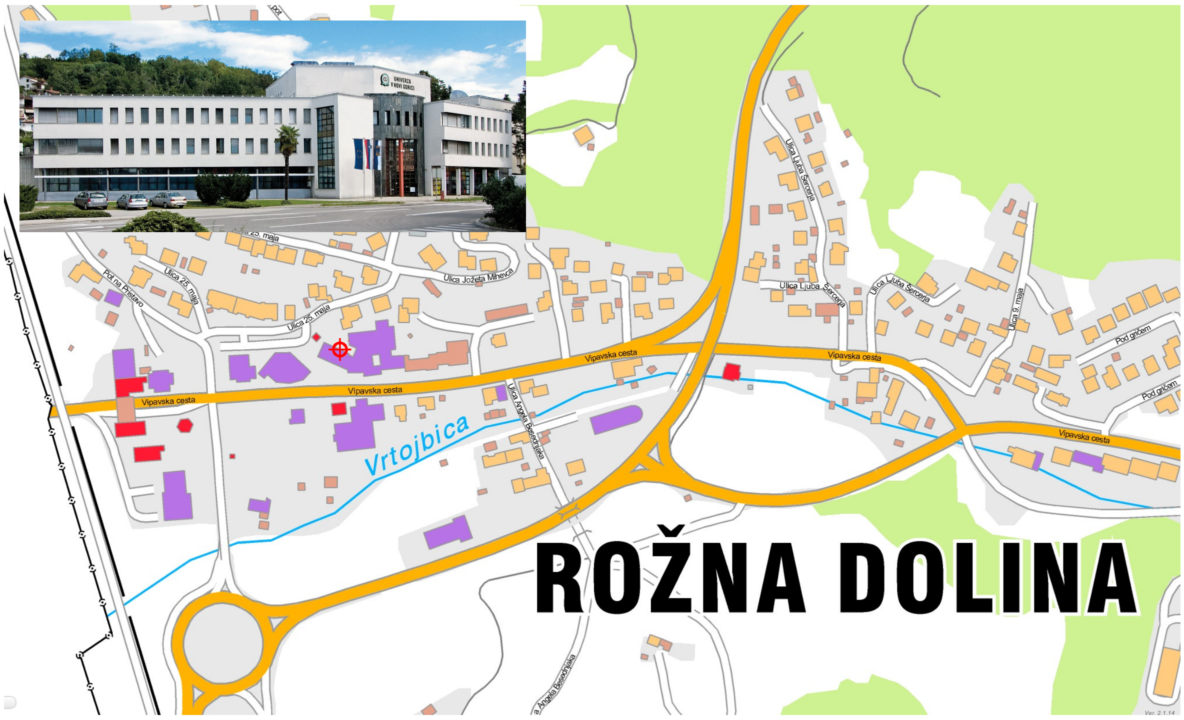


Figure 3.9: Map of Rožna dolina. The red circle indicates the location of the antenna and the inset is showing the University building at Vipavska 13, [14].

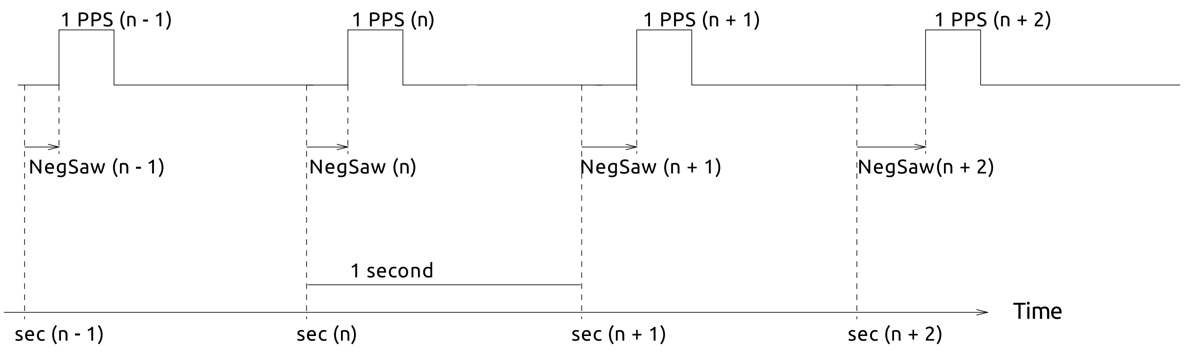


Figure 3.10: Sawtooth correction.

3.4 Setup

On Figure 3.14 we can see the scheme of an experiment setup. The system consists of two Motorola Oncore UT+ receivers with corresponding antennas, Tektronix DPO4104 oscilloscope and a computer. Each Motorola receiver is connected to the computer via USB cable and to a oscilloscope through a 1 ns lemo cable.

The oscilloscope is connected to the computer through Ethernet cable. That was necessary because the computer operating system is a version of Ubuntu, which is Linux distribution, whereas oscilloscope operating system is a version of Windows operating system. Unfortunately Tektronix does not provide an official driver which would allow user to connect

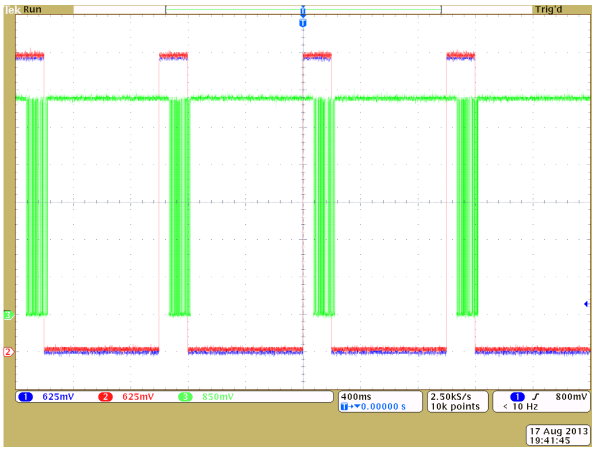


Figure 3.11: 4-second-long waveforms of 1PPS signals from both GPS units (on oscilloscope channels 1 and 2) and the voltage on the TXD pin of the GPS module (oscilloscope channel 3) indicating transmission of data, including the sawtooth correction, happening shortly after the leading edge of the 1PPS signals. All signals are 5 V high, there are only different scaling rates applied on signals.

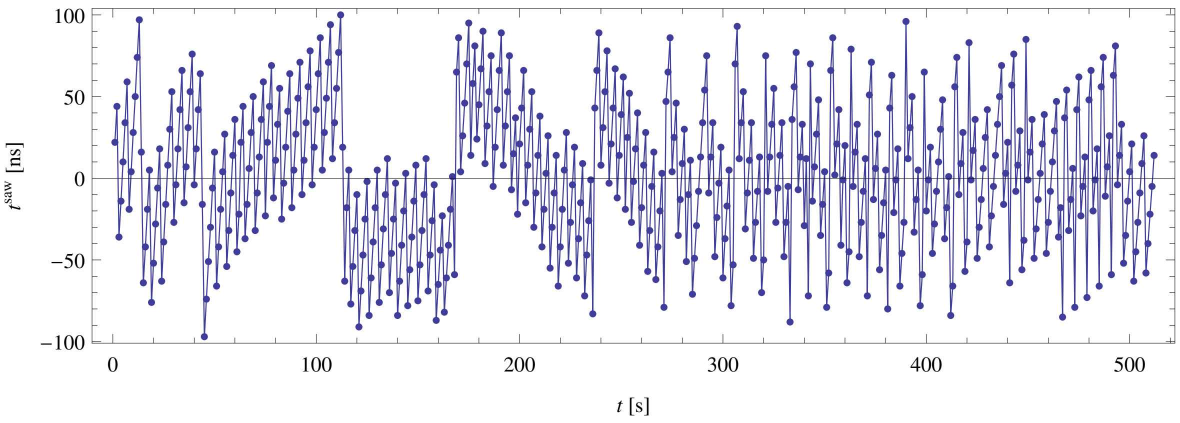


Figure 3.12: Sawtooth correction of Motorola Oncore receivers used in experiment. This plot shows evolution of sawtooth correction in time. If time range would be longer than 500 s, you could see that this kind of behavior is periodic.

to the oscilloscope with the USB cable. In order to overcome that problem some unofficial drivers were tested, but none provided satisfying results. Therefore the only option, besides changing operating system, was to switch to Ethernet cable.

3.5 Data acquisition

A custom program was written for data acquisition. It was written in Python because of its rather simple structure and many custom packets which simplified the program and saved us a lot of work. The main reason for a complete custom program was the fact that data acquired from all devices had to have same time coordinate, which was easiest to be accom-

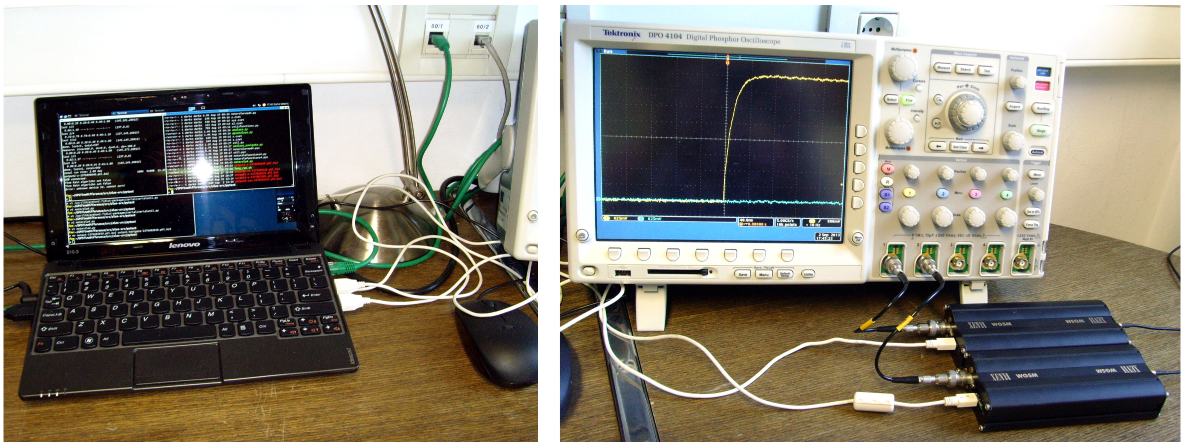


Figure 3.13: Photograph of the setup.

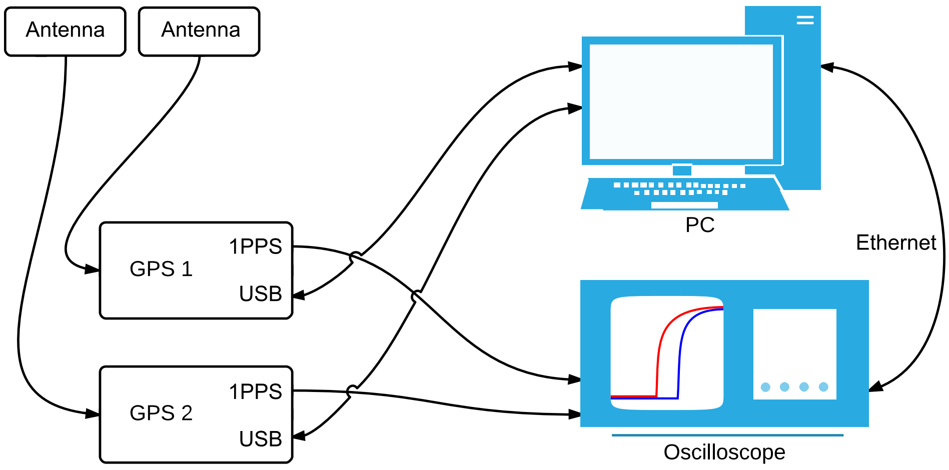


Figure 3.14: Scheme of setup.

plished with one program handling acquisition from all devices.

3.5.1 GPS receivers

Motorola Oncore UT+ uses binary interface, which means that all commands sent to the device and all data received from it must be in a binary format. There is also possibility of NMEA output, which returns characters. Even though that kind of output is easier to process, you do not need to decode data, the command range is limited and does not covers all needed options. For that the reason binary format was used all the time.

Each command begins with `@@`, followed by two letters, first the upper case, second he lower case, which define the command. Then come variables of the command and a check-sum. The command ends with the control characters: a carriage return (ASCII code 13) and a new line (ASCII 10). Example of a command:

```
@@Ea 01 24\r\n
```

Even though GPS receivers were connected to the computer through USB, RS232 standard still has to be used for communication. Due to that `pySerial`, a python module, was used. This module encapsulates access to serial ports in python environment. Not only 1PPS signal but other information as well were gathered from receiver, such as position of receiver, GPS time of receiver, visible satellite position and signal strength and 1PPS correction. Most of those information are not directly connected with primary goal, but they are collected in Pierre Auger. So in case some interesting relations would been found during analysis, that relation could be included in the final correction equation.

3.5.2 Oscilloscope

As mentioned before, th oscilloscope was accessed through Ethernet cable not USB as originally planned. Due to that VXI-11 standard was used for communicating. VXI-11 is an international standard for control of equipment over standard TCP/IP network. You may consider VXI-11 as network based GP-IB. Although GP-IB is still used widely in the field, most modern measurement devices are equipped with Ethernet interface and support VXI1 standard over Ethernet. Using Ethernet based VXI11, measurement system can be setup using a standard PC and this equipment without additional hardware, such as GPIB interface boards or boxes [3].

The observed value was voltage in respect to time. The coordinate system is divided into smaller “boxes”, where the oscilloscope allows user to specify dimensions of unit “box”. In this case height was set to 625 mV so that the signal was exactly 8 “boxes” high and the length was set to 40 ns. Trigger was set on the first waveform, with value of 800 mV. The second waveform was untriggered. For each waveform 10 000 points were acquired, where point width is 0.2 ns. We can conclude that the biggest observable time difference is 1000 ns. Any time difference higher than 1000 ns can not be resolved and therefore that kind of measurement would be rejected in the analysis. 10 000 is quite a big number if we keep in mind that two waveforms and data from two GPS receivers had to be acquired and stored in second, every second. So in order to reduce time of acquisition, the oscilloscope was set to output data in the binary format. That way the size of waveform was reduced. Of course that data will have to be decoded, but once data is acquired the time needed for decoding is not the biggest concern. Typical oscilloscope screen-shot can be seen in Figure 3.15.

3.5.3 Data acquisition program

When the program is started, it first connects to GPS receivers and set up GPS receivers for measurements. In this case output time is set to GPS time and cutoff angle is set to 15°. The cutoff angle defines an angle above the horizon, below which satellites are ignored even if they are still visible and received. This is done to avoid interference problems caused by buildings, trees, multi-path effects, and atmospheric errors at low angles. 0° equals to horizon, 90° means that a satellite is right above the receiver. The program also sets all output messages and sets initial predefined position of a receiver. That position was determined by calculating mean value of latitude, longitude and height, after that data was collected for two days. The receiver identity is checked for the purposes of consistency. Receiver cases are labeled and are always connected to the same port in the oscilloscope. It was decided that the first receiver is triggered and is set to the right position at all times, while the second receiver is not triggered and his position is being changed.

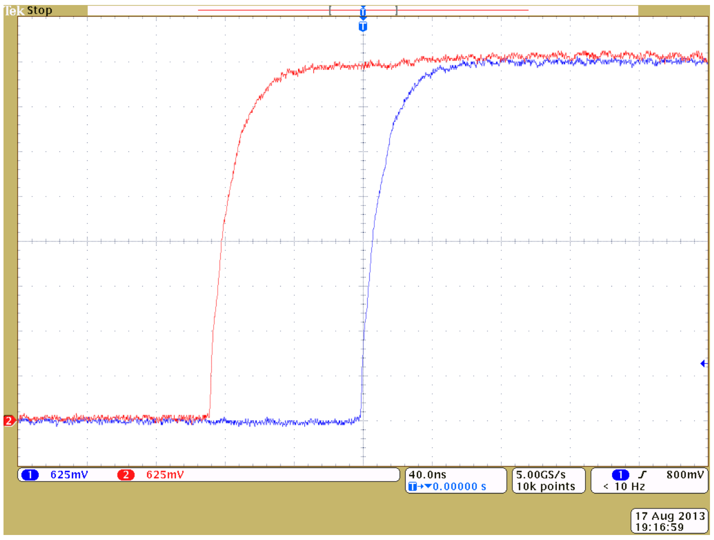


Figure 3.15: Typical output of the oscilloscope, where curves represent the 1PPS signal. Blue curve is from the first GPS receiver whose position is fixed at all times. Red curve is from the second GPS receiver, whose position is altered.

When receivers are set, the program connects to the oscilloscope. Settings for oscilloscope are written in a separate file. The reason for the separate file is that there are 555 different parameters to set and it would be messy to have them in the program file. Most important settings were already discussed in previous section and the others are mostly cosmetical nature therefore I am not going to waste time on them. The output file is also opened and prepared for writing. After that program reads the file with positions which will be used in that particular run.

The acquisition is made so that each position (where by position I mean wrong position which is set to second receiver) is being set for a particular amount of time, after which new position is set. The number of different positions measured on one run is limited, because after the last position measurements continue with the first one. The set of positions is being measured for 2 days, with 10 minutes measuring each position. Of course in this way one position is not measured for the entire constellation, but only a limited part of constellation, which affects accuracy of final result. But if one wanted to include only all constellation effects, one position would have to be measured for the entire year. That would require a completely different approach, not to mention it would be much more financially demanding. On the other hand the time for bachelor thesis is limited to six months, so that kind of compromise had to be made.

Each position is being measured as follows. The first new position is set to the second receiver. After that program sends a command for waveform acquisition to the oscilloscope. Unfortunately this has to be done every time. It would be much better if the oscilloscope was set to send waveform every time when it gets new trigger, but it is not possible. GPS receiver are set to send data every second, so after waveform command have been sent, you just wait for the data from receivers and the oscilloscope. Because in general one can not know which device will return data first, the select function is used. This function allows non blocking reading. In practice this means that a user provides a list of devices which

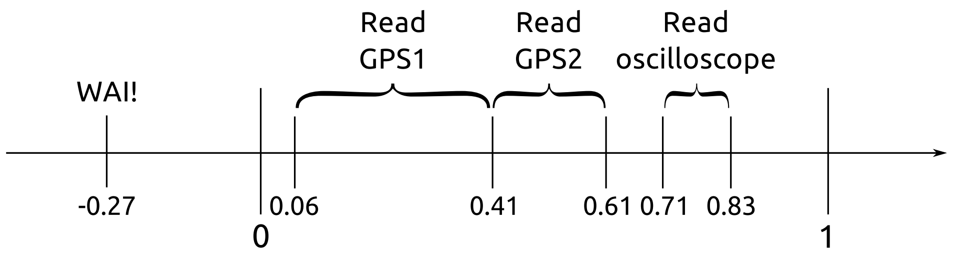


Figure 3.16: Timeline of data acquisition. `WAI!` is a command send to the oscilloscope which makes it wait until new waveforms are available before sending them. The first GPS message is not being read exactly at full second (i.e. at 0) since it takes some time to process the signals received from satellites. The next second is denoted by 1.

needs to be read and the function tells you when a device and which device can be read. This function is packed in a loop, so the program waits until either of them has something to tell, read data and then waits until new data is ready. This way you do not lose any data while waiting for another device to transmit data. The program is written is such way that all read data correspond to the same signal, but unfortunately one can not be sure until the analysis. If data is not completely synchronized, it does not completely corrupt it, but it makes its analyzing harder.

In principle all devices must be in a select loop, but after testing it was determined that the program runs more stable if in select function are only GPS receivers and the oscilloscope is read outside the function. So in practice reading goes as follows. When the program enters the loop for the first time, the command for waveforms acquisition is sent to the oscilloscope. That command works in such way, that it waits until a new trigger happens and after that waveforms are sent to the computer. So back to the loop. Waveform command is sent, now the program enters the select loop and waits until GPS receivers get new signal from satellites and return data. That should correspond to a new waveform. When data is available program reads it. When data from both receivers is acquired, waveforms are read. When reading is completed a new waveform command is sent to oscilloscope and all data is written to the file. The loop goes back to waiting for new data from receivers. The time line of data acquisition as it happens in real program can be seen in Figure 3.16. This loop is repeated for 10 minutes, after that the position is changed and the whole thing repeats.

After two days the file with measurements, a new set of positions is loaded from the file and everything repeats until all set of positions in file are measured.

Data is stored in files using Pickle, a Python module which implements a fundamental, but powerful algorithm for serializing and de-serializing a Python object structure. Basically it allows user to store Python object in the file and then retrieve those same objects back from file. Each second a Python dictionary with time, Δx, Δy, Δz, information retrieved from both GPS receivers and two waveforms is stored. This kind of a dictionary is approximately 20 kB big. Final file size would be in order of 2.6 GB. To save some space `bzip2` file compressor was used. That way size reduced approximately for a factor 3, to a dictionary size $\sim$7 kB and final file size to $\sim$900 Mb for each run.

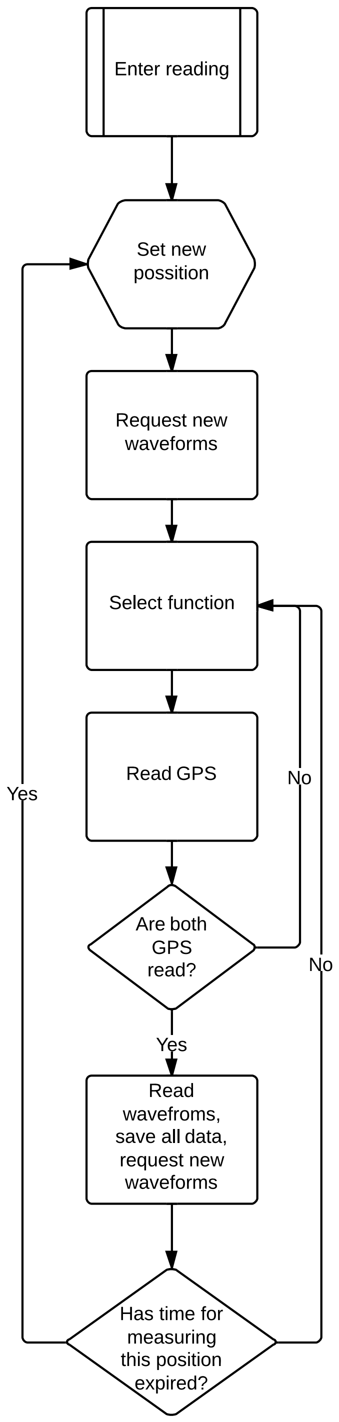


Figure 3.17: Schematic representation of reading loop. Before program enters this loop, it set up both GPS receivers and oscilloscope, reads file with positions and open output file.

Chapter 4

Results

4.1 Theoretical predictions

GPS receiver calculates its position based on its distance to at least four satellites. Distance to satellites can not be obtained directly, but is calculated based on the knowledge of speed of transmitted signal (speed of light c) and time it took for the signal to arrive from the satellite to the receiver

$$d = c\,\Delta t. \tag{4.1}$$

When it calculates distance to the first satellite, it knows that it is located somewhere on a sphere with radius equal to the distance to the satellite. When distance to the second satellite is added sphere reduces to a circle. With third satellite receiver gets down to two points. In some cases this is enough to determine position, due to the fact that one point has impossible position, but in general there has to be fourth satellite to pinpoint receiver location. Scheme of this procedure can be seen in Figure 4.1.

When the receiver is in Position-hold mode only time is fitted ant therefore only one satellite is enough to calculate time precisely. At a certain time a satellite broadcasts a message tagged with a time t_s. At the time of the reception, the receiver will try to adjust the internal clock to show the time of the message, corrected for the time the message took to reach the receiver,

$$t = t_\mathrm{s} + d/c, \tag{4.2}$$

where d is the distance between the satellite and the receiver, and c is the speed of light. If more satellites are tracked at the same time, this target time becomes an average over the time estimates for different satellites,

$$\langle t \rangle = \langle t_\mathrm{s} + d/c \rangle, \tag{4.3}$$

over all satellites in sight.

When the GPS unit is set to the fixed-position mode with position which deviates from the true position by vector $\vec{w}$, the unit will make a wrong time synchronization

$$t' = t_\mathrm{s} + d'/c, \tag{4.4}$$

where the wrong vector $\vec{d'}$ to the satellite is shifted by $\vec{w}$ and thus $\vec{d} = \vec{w} + \vec{d'}$. The shift of the estimated time Δt corresponding to the position shift $\vec{w}$ can be obtained by subtracting Eqs. (4.2) and (4.4),

$$\Delta t = t - t' = (d - d')/c. \tag{4.5}$$

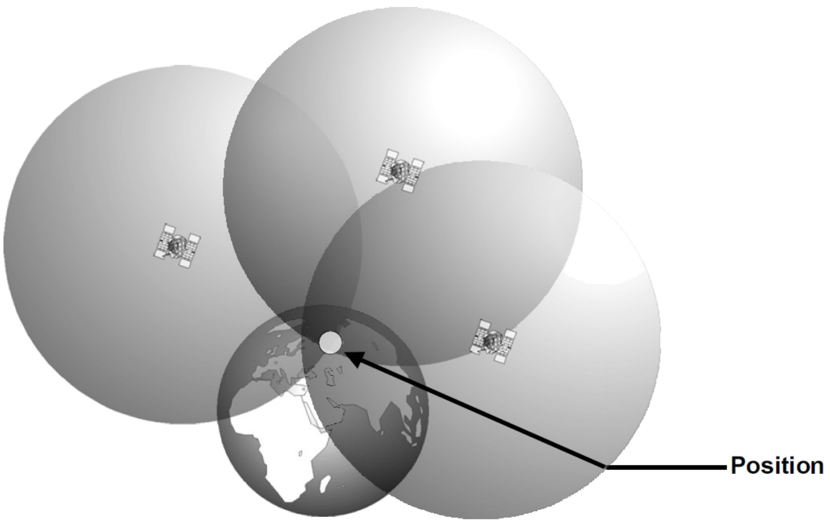


Figure 4.1: GPS receiver must have at least four satellites in view in order to be able to calculate its position precisely. From this picture you may get the impression that three satellites would be enough, but you must keep in mind that there is another intersection of those spheres that is not visible in this figure [13].

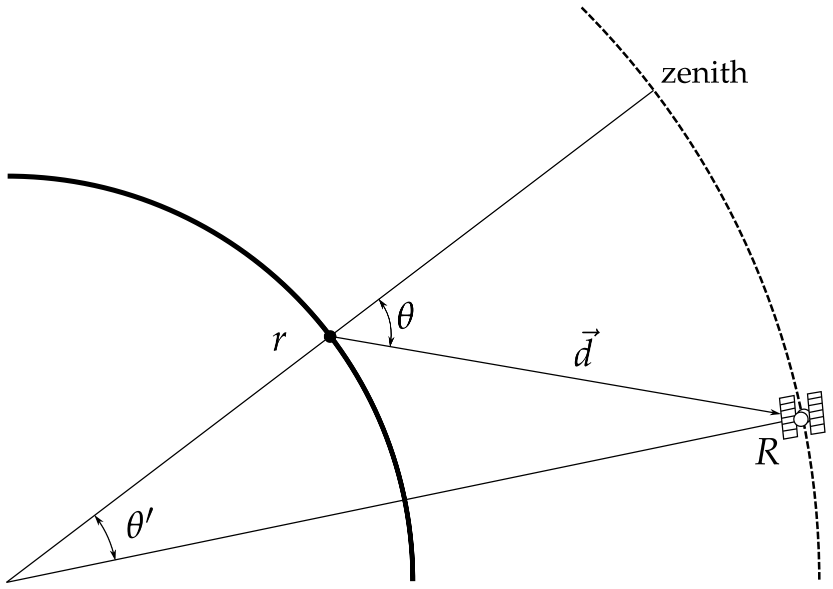


Figure 4.2: Scheme of geometry. R is a distance from the Earth center to a satellite, r is Earth radius. R is in the range of ~26 600 km and $r \approx 6400$ km.

For many satellites in view Δt becomes average over all time differences for satellites in view.

$$c\langle \Delta t \rangle = \langle d - d' \rangle. \tag{4.6}$$

The time shift is thus

$$c\Delta t = d - |\vec{d} - \vec{w}| = d - \sqrt{d^2 - 2\vec{w} \cdot \vec{d} + w^2}, \tag{4.7}$$

Satellites orbit around Earth on 26 600 km, where w is usually in order of 10 m or in rare cases in order 100 m. Base on that $w/d \ll 1$ can be safely assumed. This way Equation (4.7) can be expanded to first order,

$$c\Delta t \approx \vec{w} \cdot \frac{\vec{d}}{d}. \tag{4.8}$$

The mean time shift will thus depend on a very simple quantity related to the distribution of satellites in the sky,

$$c\langle \Delta t \rangle = \vec{w} \cdot \langle \hat{d} \rangle = \vec{w} \cdot \vec{\xi}, \tag{4.9}$$

where $\hat{d} = \vec{d}/d$, and $\vec{\xi} = \langle \hat{d} \rangle$ is the time shift coefficient. The first measurements of the time shift coefficient were done by Xavier Bertou in 2008 [11] by changing the position parameters of the test station at the Auger main campus. The vertical component was found to be $\xi_z = 2.19\,\mathrm{ns/m} \times c = 0.657$.

In the next Sections we will try to estimate the value of the time shift coefficient under different assumptions of satellite distribution.

4.1.1 Estimates of time shift

Uniform distribution on the local sky

Let us assume that satellites are uniformly distributed over the local sky. We can immediately see that this distribution is axisymmetric around the zenith direction and that the time shift coefficient has only one component,

$$\vec{\xi} = (0, 0, \langle d_z/d \rangle). \tag{4.10}$$

Value of ξ is therefore a length of vector $\vec{\xi}$. Since $\vec{\xi}$ has only one component, its length is simply this component, $\langle d_z/d \rangle$, which is obtained as

$$\langle d_z/d \rangle = \langle \cos\theta \rangle = \frac{\int \cos\theta \,\mathrm{d}\cos\theta}{\int \mathrm{d}\cos\theta} = \tfrac{1}{2}(1 + \cos\theta_{\max}), \tag{4.11}$$

(angles and distances are the same as in Figure 4.2).

For the default setting a $\theta_{\max} = 75°$ is used in Pierre Auger Observatory, for which we get $\xi_z = 0.63$ and $\xi_z/c = 2.1\,\mathrm{ns/m}$.

Uniform distribution on GPS sphere

Let us assume the satellites are uniformly distributed over the visible portion of the sky. We can see that this distribution has similar properties as the previous one, it is again axisimetric and therefore time shift coefficient has only one component,

$$\vec{\xi} = (0, 0, \langle d_z/d \rangle). \tag{4.12}$$

Since $d_z = R\cos\theta' - r$ and $d = R\sqrt{1 - 2\rho\cos\theta' + \rho^2}$ where $\rho = r/R$, the mean of this component can be written as

$$\langle d_z/d \rangle = \frac{\int d_z/d \,\mathrm{d}\cos\theta'}{\int \mathrm{d}\cos\theta'}., \tag{4.13}$$

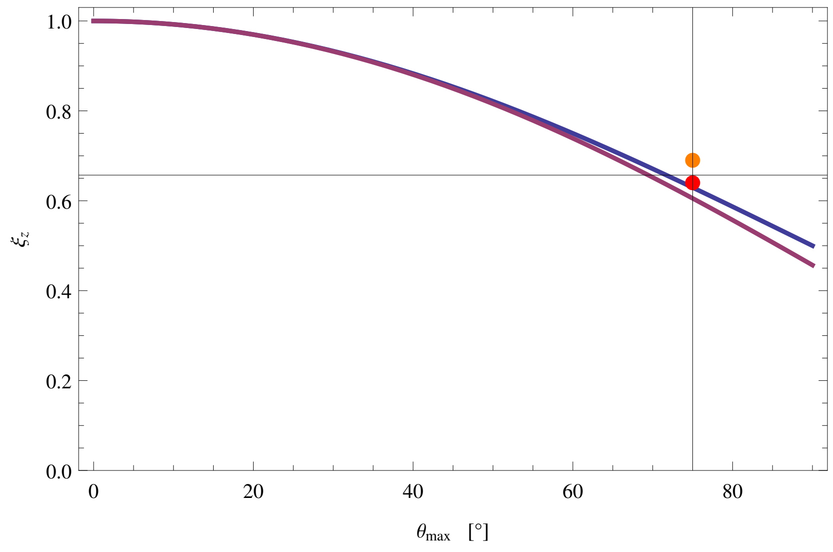


Figure 4.3: Time shift coefficient ξ_z for different zenith angle cuts and two approximations, uniform local sky coverage (blue) and uniform coverage of the sphere with radius of 26 600 km (purple) and two Monte Carlo simulations, with rotation of the Earth and satellite orbiting around the Earth unconnected (red) and connected (orange). The axis origin is set to the default zenith angle cut $\theta_{\max} = 75°$and the Auger measurement $\xi_z/c = 2.19\,\text{ns/m}$.

In order to solve Equation 4.13 easier, let us say that $\cos\theta' = u$. Satellites are observed from the zenith, $\cos 0 = 1$, and all the way down to the horizon, $\cos\theta'_{\max} = u_{\min}$, so that we get

$$\langle d_z/d\rangle = \frac{1}{1-u_{\min}} \int_{u_{\min}}^{1} \frac{u-\rho}{\sqrt{1-2\rho u+\rho^2}}\,\mathrm{d}u. \tag{4.14}$$

With the GPS orbit $\rho = 6400\,\text{km}/26\,600\,\text{km} = 0.24$ and $\theta_{\max} = 75°$, $\theta'_{\max} = 61.6°$ we get $\xi_z = 0.61$ and $\xi_z/c = 2.02\,\text{ns/m}$.

Uniform distribution on GPS orbit

Satellites are not uniformly distributed over the sphere, but are traveling in orbits tilted with inclination $i = 55°$ relative to the equator. Therefore, $\vec{R}$ can be written as

$$\vec{R} = (\cos\phi_s, \cos i \sin\phi_s, \sin i \sin\phi_s), \tag{4.15}$$

where ϕ_s represents an arbitrary angle which rotates a satellite around the Earth. Let us presume that satellites orbit around the Earth independently of Earth's rotation. Scheme of such motion can be seen in Figure 4.4. Due to the Earth's rotation, $\vec{r}$ is written as

$$\vec{r} = \begin{pmatrix} \cos\Phi(\cos\Lambda\cos\phi_u - \sin\Lambda\sin\phi_u) \\ \cos\Phi(\cos\phi_u\sin\Lambda + \cos\Lambda\cos\phi_u) \\ \sin\Phi \end{pmatrix}, \tag{4.16}$$

where Λ is longitude, Φ is latitude and ϕ_u represents an arbitrary angle which rotates position of the receiver relative to the Earth's rotation. The distance between the receiver and the satellite thus becomes

$$\vec{d} = \vec{R} - \vec{r}. \tag{4.17}$$

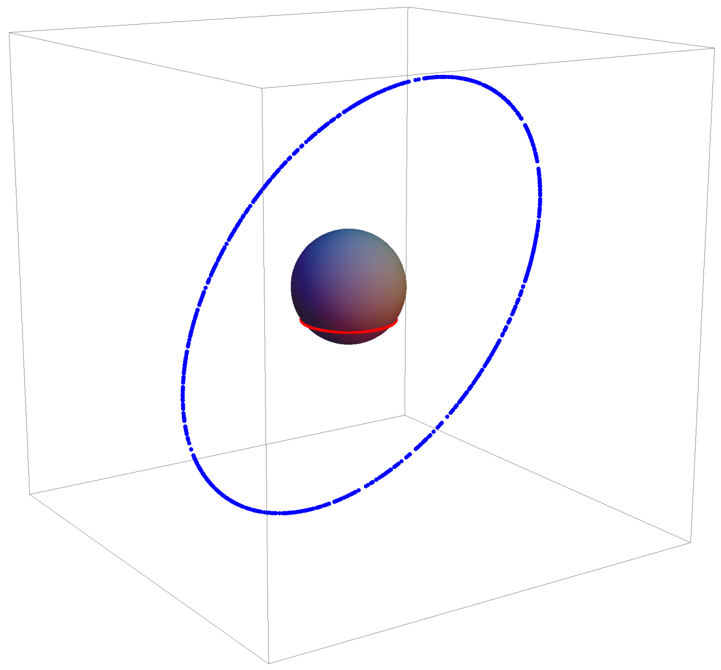


Figure 4.4: Sphere in the middle represents Earth, red points are positions of the Pierre Auger Observatory while Earth rotates, as seen when observed from space. Blue dots are positions of satellite that orbits around Earth.

$\cos\theta$ can thus be defined as

$$\cos\theta = \frac{\vec{r}\cdot\vec{d}}{|\vec{r}||\vec{d}|}. \tag{4.18}$$

Using unit vectors the equation simplifies into

$$\cos\theta = \hat{r}\cdot\hat{d}. \tag{4.19}$$

In order to calculate the shift coefficient using Equation (4.19), a Monte Carlo simulation was performed, where Φ and Ω were generated randomly. Coordinates of the main campus of the Pierre Auger Observatory were used as the coordinates of the receiver (latitude 35.466667° S, longitude 69.311389° W, height 1450 m), radius of the Earth $R_{\text{E}} = 6371\,\text{km}$ and distance from the Earth's center to the satellite equals to 26 600 km. Results were $\xi_z = 0.639 \pm 0.0007$ and $\xi_z/c = 2.3\,\text{ns/m}$.

We know that the motion of satellites is not independent of the Earth's rotation. In the first approximation, we can say that the satellites go around the Earth exactly twice per day, so that their trajectories do not change in time. That is again not exactly true, but we will it in this particular model. In order to simulate the motion of the satellite around the Earth depending on Earth's rotation, we use exactly the same model as in previous simulation. The only difference is in the angle responsible for the rotation. This time randomly generated angle is kept the same for both Earth and satellite, where for the satellite it is multiplied by factor 2. Results of this simulation were $\xi_z = 0.692 \pm 0.0007$ and $\xi_z/c = 2.3\,\text{ns/m}$.

Histograms for both simulations can be seen in Figure 4.5.

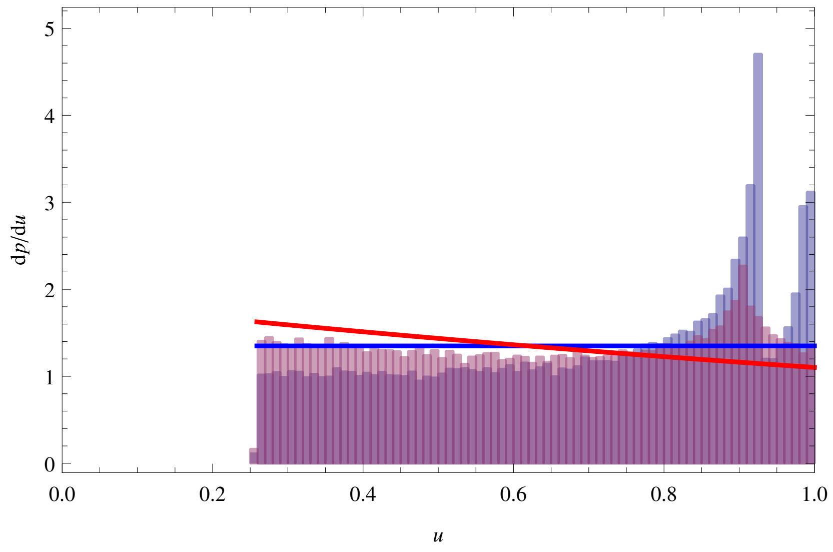


Figure 4.5: Violet histograms correspond to simulation where Earth and satellites are moving independently, blue histograms correspond to simulation where motion of the satellites depends on rotation of the Earth. Blue line corresponds to uniform distribution on the local sky, while red line corresponds to uniform distribution on the GPS sphere.

4.1.2 Time variance

Since $c\Delta t = \vec{w} \cdot \hat{d}$ we can write the variance of the mean time shift obtained from N satellites as

$$N\,\mathrm{V}[c\Delta t] = w_x^2\,\mathrm{V}[\hat{d}_x] + w_y^2\,\mathrm{V}[\hat{d}_y] + w_z^2\,\mathrm{V}[\hat{d}_z]. \tag{4.20}$$

The first two variances are the same due to the zenith axisymmetry, therefore

$$Nc^2\,\mathrm{V}[\Delta t] = (w_x^2 + w_y^2)\,\mathrm{V}[\hat{d}_x] + w_z^2\,\mathrm{V}[\hat{d}_z]. \tag{4.21}$$

The two terms are

$$\mathrm{V}[\hat{d}_x] = \left\langle \sin^2\phi \sin^2\theta \right\rangle = \tfrac{1}{6}(1 - u_{\mathrm{min}})(2 + u_{\mathrm{min}}), \tag{4.22}$$

$$\mathrm{V}[\hat{d}_z] = \left\langle \cos^2\theta \right\rangle = \tfrac{1}{3}(1 - u_{\mathrm{min}} + u_{\mathrm{min}}^2), \tag{4.23}$$

where $u_{\mathrm{min}} = \cos\theta_{\mathrm{max}}$. Note that with the zenith cut almost at the horizon, the two variances are $\mathrm{V}[\hat{d}_x] \approx \mathrm{V}[\hat{d}_z] \approx 1/3$.

In this limit the standard deviation σ of the time simplifies into

$$\sigma \approx \frac{w}{c\sqrt{3N}}. \tag{4.24}$$

From

$$\tan\theta = \frac{R\sin\theta'}{R\cos\theta' - r} \tag{4.25}$$

we get

$$\cos\theta = \frac{\cos\theta' - \rho}{\sqrt{1 - 2\rho\cos\theta' + \rho^2}}, \tag{4.26}$$

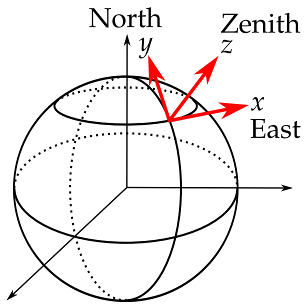


Figure 4.6: Scheme of coordinate system used in measurements. z axis points toward zenith, x axis points toward East and y axis points toward North.

$$\cos\theta' = (1-\cos^2\theta)\rho + \cos\theta\,\sqrt{1-(1-\cos^2\theta)\rho^2}, \tag{4.27}$$

$$\tan\theta' = \tan\theta\,\frac{1\pm\rho\,\sqrt{1-(\rho^2-1)\tan^2\theta}}{1-\rho^2\tan^2\theta}, \tag{4.28}$$

with $\rho = r/R$.

4.2 Measurement results

4.2.1 Navigate run

In order to be able to put GPS receivers to Position Hold mode, position of a receiver must be known. In a preliminary run, GPS coordinates acquired from other sources were used. Height was estimated from topographic data for height of the road and height of the University building. After the run acquired data was analyzed, which showed some strange behavior. Immediately coordinates used were suspected as a cause for such behavior. In order to check the theory a test in navigate mode was performed.

It turned out that presumptions were right. Coordinates acquired from the navigate run were different from those used in the preliminary run. Coordinates can be seen in Table 4.1. Latitude and longitude are in milliseconds (mas), which is a unit used in Motorola Oncore output. Conversion to degrees, minutes and second can be seen in Appendix A. The uncertainties are very small, what is caused by high statistic. Mean values of coordinates were calculated from 29 000 measurements, which statistically means very high accuracy, but we must not forget seasonal effects which reduce this accuracy.

In following measurements the coordinates acquired from the navigate run were used in Position Hold mode. Coordinates also serve as origin for the coordinate system in which $\Delta x, \Delta y$ and Δz operate.

4.2.2 Measurement schedule

Measurements were performed in several steps. In first few measurements position was changed only in one dimension. The general behavior of the 1PPS signal was established. Errors in the set position are almost never pointed in only one direction, but in arbitrary direction in space. In order to test whether the repair models will actually work the grid

	Latitude [mas]	**Longitude [mas]**	**Height [m]**
Preliminary coordinates	165392651	49101037	85
Navigate run	165392558±0.14	49100893±0.14	147.61±0.01

Table 4.1: Preliminary coordinates were retrieved from outside source and turned out to be very wrong. After navigate run actual coordinates of receiver were acquired and used.

Δz [m]										
	0	1	2	4	8	16	32	64	128	256
		-1	-2	-4	-8	-16	-32	-64	-128	-256

Table 4.2: Positions first measured in z direction.

Δx, Δy [m]									
	1	2	4	8	16	32	64	128	256
	-1	-2	-4	-8	-16	-32	-64	-128	-256

Table 4.3: Positions measured in x and y direction.

Δx [m]	Δy [m]	Δz [m]		
16	16	16	64	256
	64	16	64	256
	256	16	64	256
64	16	16	64	256
	64	16	64	256
	256	16	64	256
256	16	16	64	256
	64	16	64	256
	256	16	64	256

Table 4.4: Positions measured in XYZ run. Positions were measured in a way that all possible combinations were taken in account. ($\Delta x = 16$ m, $\Delta = y$ m, $\Delta z = 16$ m), ($\Delta x = 16$ m, $\Delta = y$ m, $\Delta z = 64$ m), ...

like measurements of the first octant of coordinate system were performed. In practice this means that for each set height different positions in x direction were measured. And for each position in x direction there were also several different y positions measured. Schematic representation can be seen in Figure 4.7.

4.2.3 Analysis

After data was acquired it had to be analyzed. The main interest was the relative difference in the 1PPS signal Δt. The exact time difference was calculated by linear interpolation between voltage points of both waveforms, around the threshold level. Procedure is graphically represented in Figure 4.8. The threshold on which Δt was calculated was the same as trigger level, which was 800 mV. Threshold was set to 800 mV, because the signal rise there is the steepest.

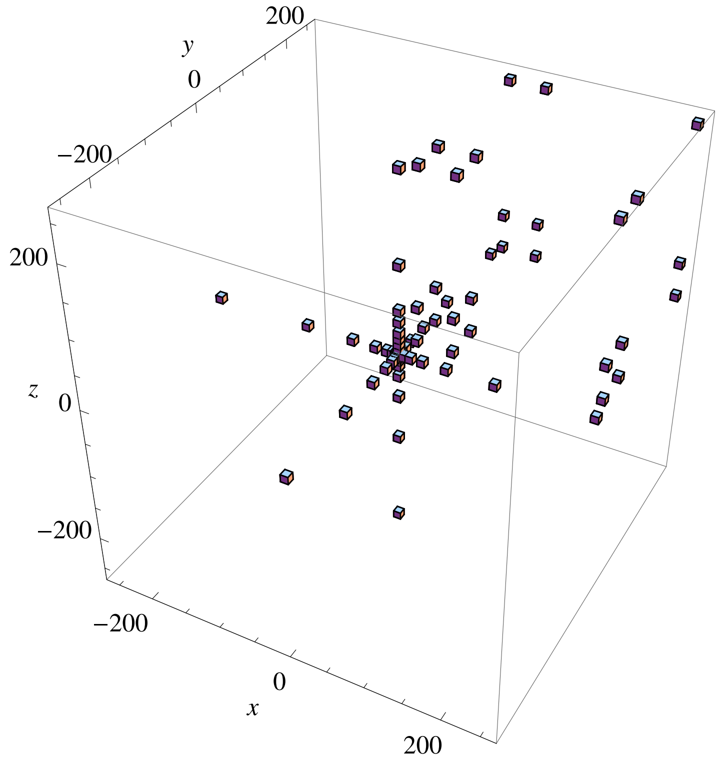


Figure 4.7: Schematic representation of measurement points.

At the beginning of a measurement and after every fixed-position change there were some measurements completely out of the expected range since it seems the GPS operation needs some time to settle after such a disturbance. At the Pierre Auger Observatory, receiver settings are rarely changed and this is thus not affecting their operations. For this reason we decided to remove the first 60 measurements ($\sim$1 min) after each change of the position and this data is not used in this analysis.

After all Δt values for all positions were acquired, mean values $\langle \Delta t \rangle$ and standard deviation $\sigma_{\Delta t}$ of Δt were calculated and plotted.

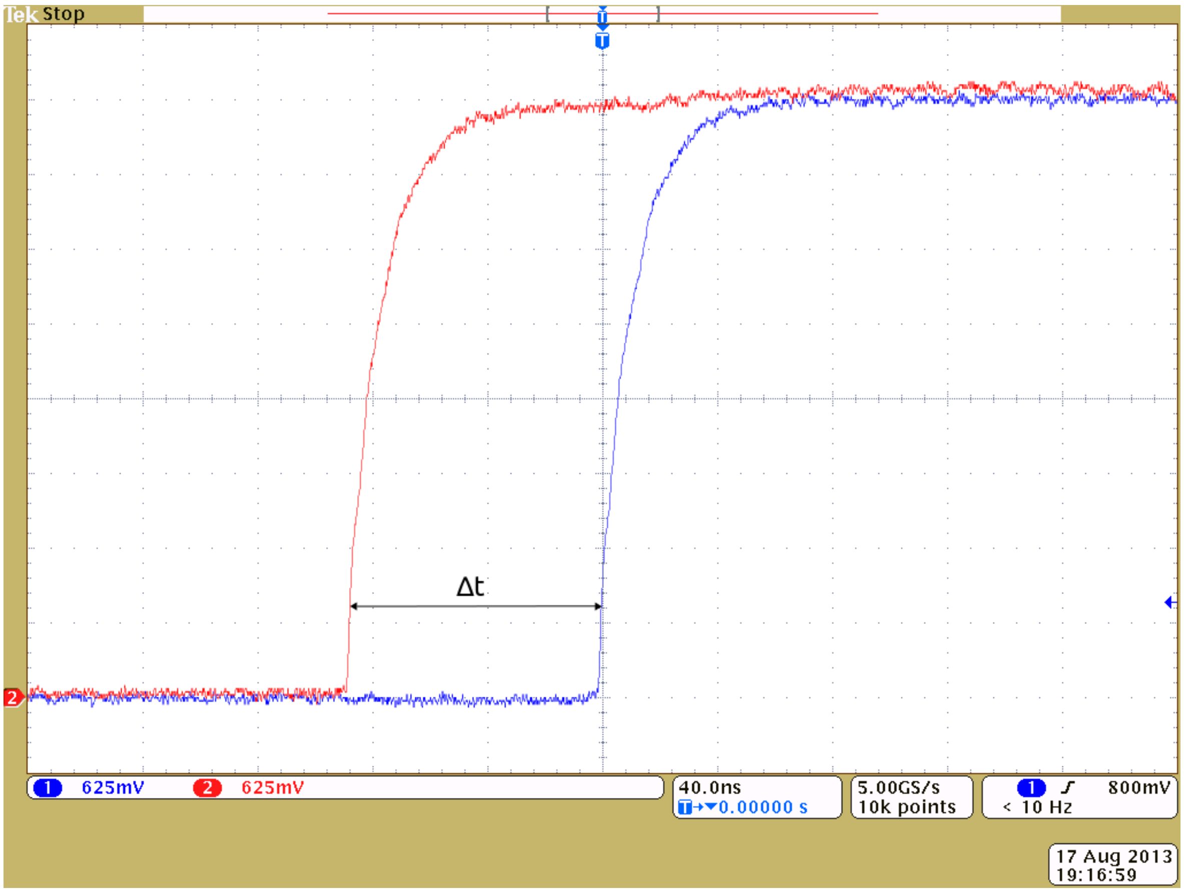


Figure 4.8: Time difference (Δt) between two waveforms, i.e. between two 1PPS signals, is a distance between two simply linear-interpolated points of the two signals at the same threshold.

4.2.4 Z direction

For z direction or height there were two sets of measurement performed. In first set were measured positions in Table 4.2

Histograms of the first set of measurements can be seen in Figure 4.9.

The histogram with $\Delta z = 0$, where both receivers were set to same coordinates has a rather nice shape, as expected. But you may notice that peak is not at 0, as one would expect, but mean value is $\sim -13\,\text{ns}$. This shift from zero is probably due to other atmospheric effects. Another effect one can observe is broadening of distribution, when dz increases. This is clearly visible on histogram for $\Delta z = 256\,\text{m}$ and even better in Figure 4.10–*Right*, where standard deviations for all heights are plotted. It seems that the standard deviation grows as a quadratic function of offset, $\sigma_{\Delta t} \propto (\Delta z)^2$.

The mean values of Δt on the other hand in this range obey linear function, which is clearly visible in Figure 4.10–*Left*. After fitting data to the linear function, resulting Equation (4.29). In Figure 4.11 one can see the comparison of measured data to fitted equation,

$$\Delta t = a\,\Delta z + b, \tag{4.29}$$

where $a = 2.25{\pm}0.086\,\text{ns/m}$ and $b = -13.23{\pm}3.39\,\text{ns}$.

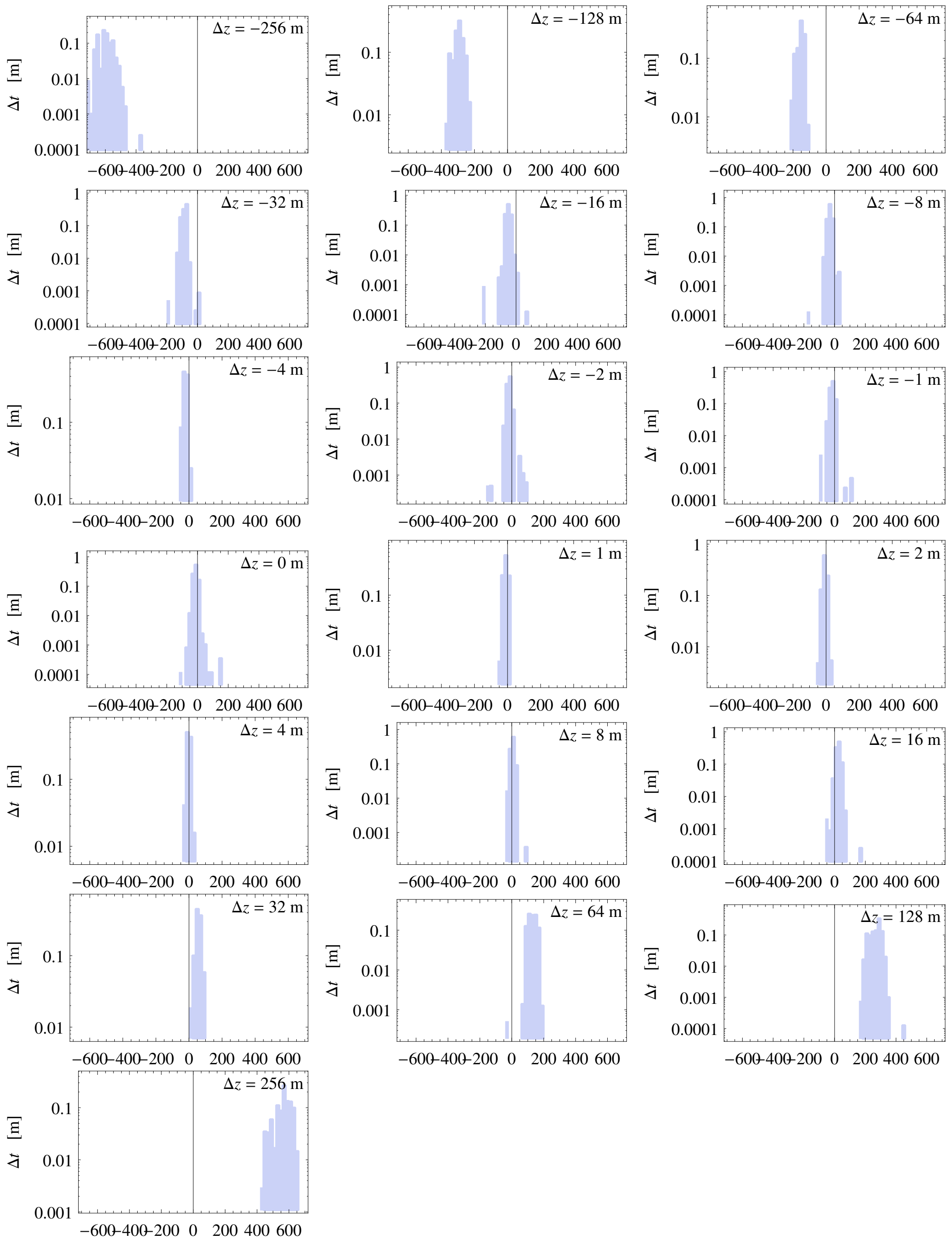


Figure 4.9: Histograms of measurements in z direction.

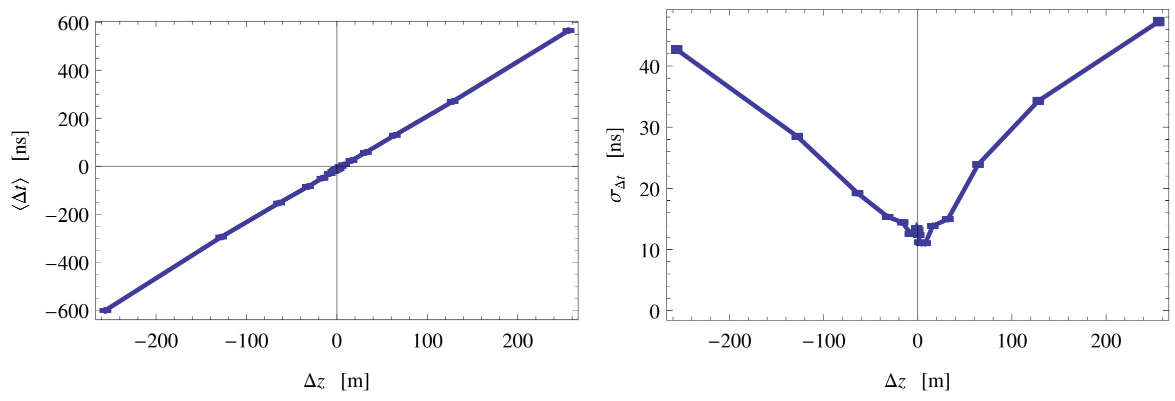


Figure 4.10: *Left:* Mean values of measurements, $\langle \Delta t(\Delta z) \rangle$. *Right:* Standard deviation of these measurements, $\sigma_{\Delta t(\Delta z)}$.

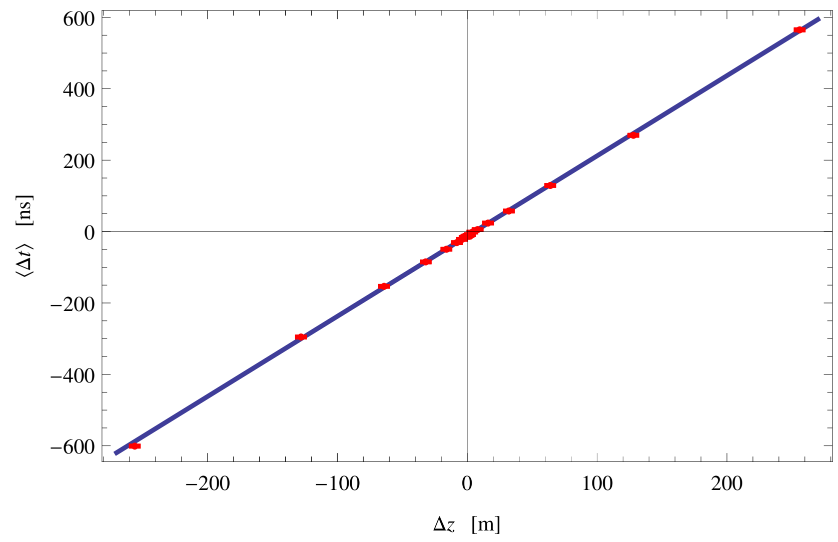


Figure 4.11: Fitted equation 4.29 (blue) in comparison to measured values (red).

4.2.5 X direction

For x direction, there was only one set of measurements. Positions measured can be seen in Table 4.3. This time there was no measurement where both receivers would be set on the same position, because that position was already measured in z direction and there was no need for repeating the measurement.

Histograms of Δt at different positions can be seen in Figure 4.13. In this case it was expected for the mean vales to be near 0 again. But as it was the case in measurements of z direction, the mean value was below 0 in range from -33 ns to -3 ns. Surprisingly the value of -3 ns was measured for $\Delta x = -256$ m and -33 ns for 256 m. Mean values can be seen in Figure 4.12–*Left*.

Standard deviation on the other hand behaves exactly as expected. It increases with absolute distance to the right position. This can be seen on histograms (Figure 4.13), where rather nice Gaussian shape degenerates to almost random distribution. For smaller Δx standard deviation behaves as quadratic function, where at larger shifts it behaves more like linear function. This behavior is better seen in Figure 4.12–*Right*.

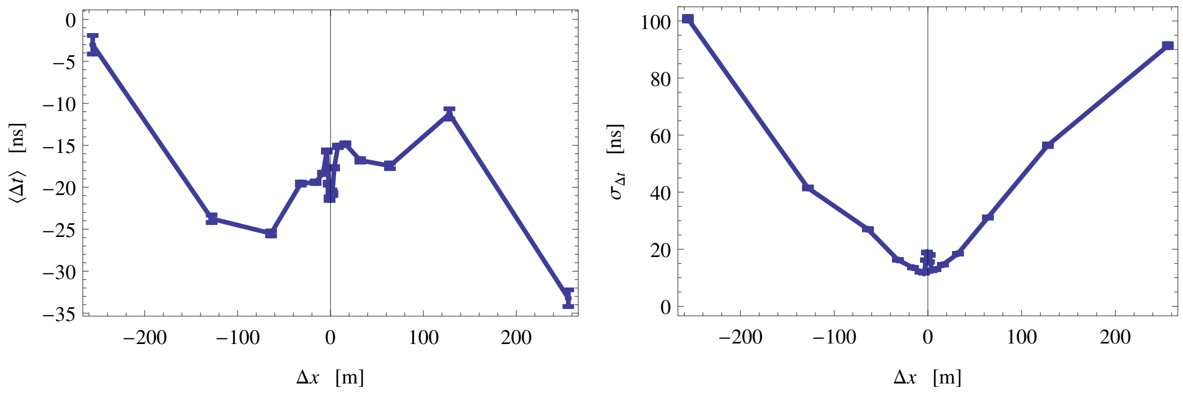


Figure 4.12: *Left:* Mean values of measurements, $\langle \Delta t(\Delta x) \rangle$. *Right:* Standard deviation of these measurements, $\sigma_{\Delta t(\Delta x)}$.

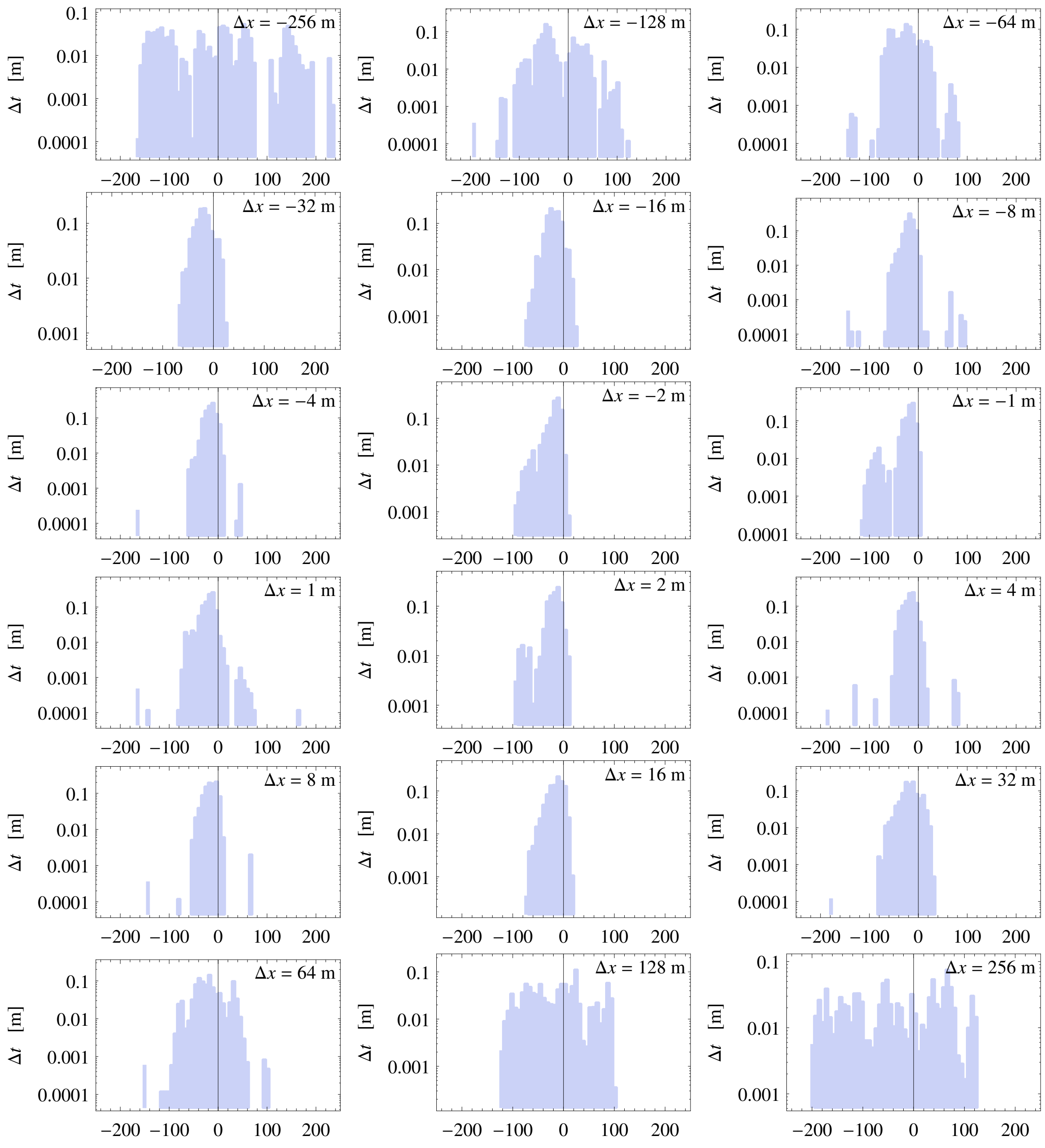


Figure 4.13: Histograms of measurements in x direction.

4.2.6 Y direction

For y direction, there was again only one set of measurements. Positions were the same as in measurement of x direction and can be seen in Table 4.3.

Histograms can be seen in Figure 4.14. The behavior is similar to that in x direction. The mean vale for $\Delta y = -256$ m is 42 ns, all other positions have negative mean values, whereas the minimum is -67 ns at $\Delta y = 256$ m. Again for small Δ Gaussian shape is observed, which broadens to very random like distribution.

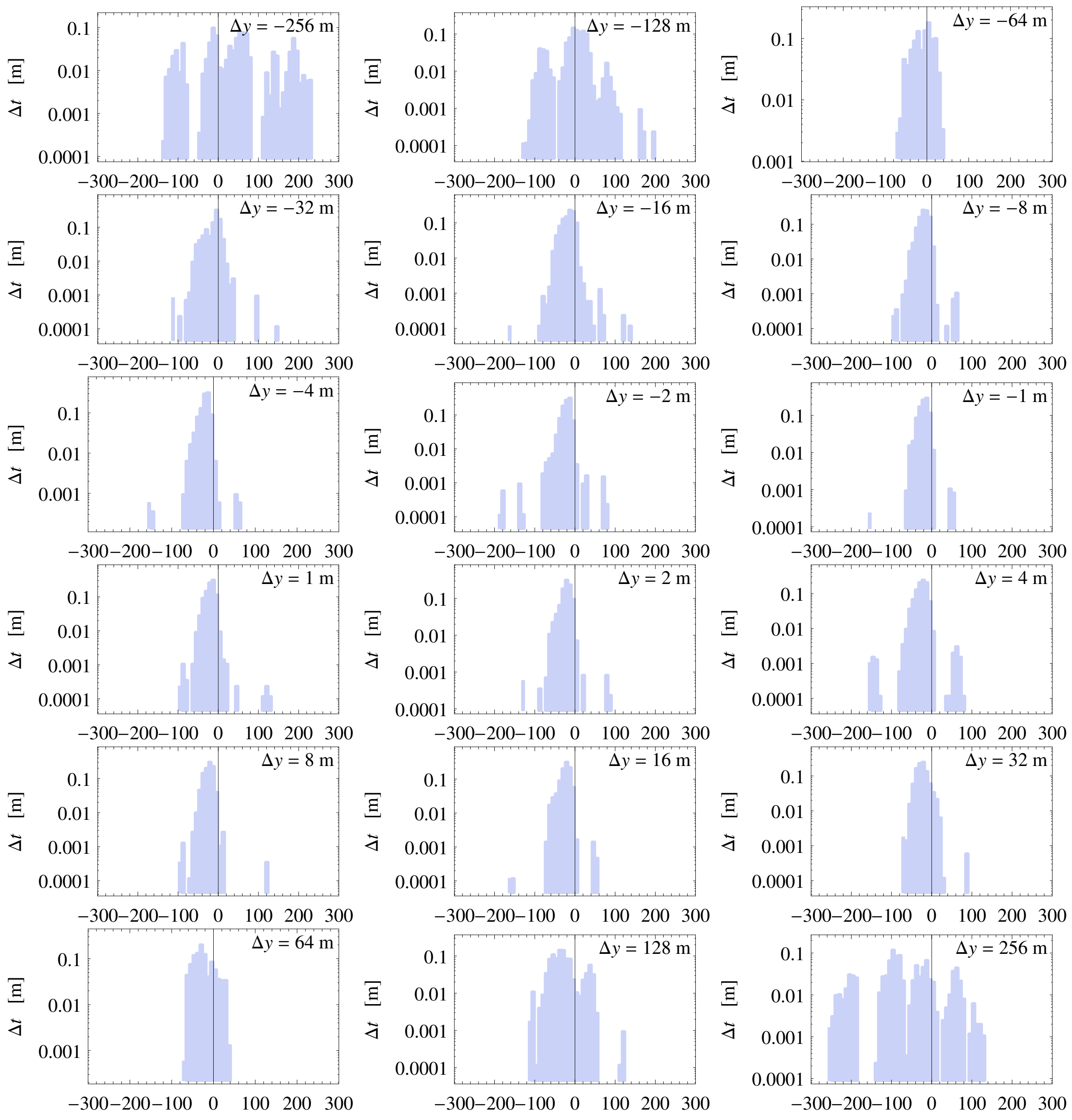


Figure 4.14: Histograms of measurements in y direction.

Standard deviation has practically the same shape as one for Δx, as expected. For smaller Δ it seems to behave similarly to a quadratic function, whereas for larger Δ it seems to behave more linearly.

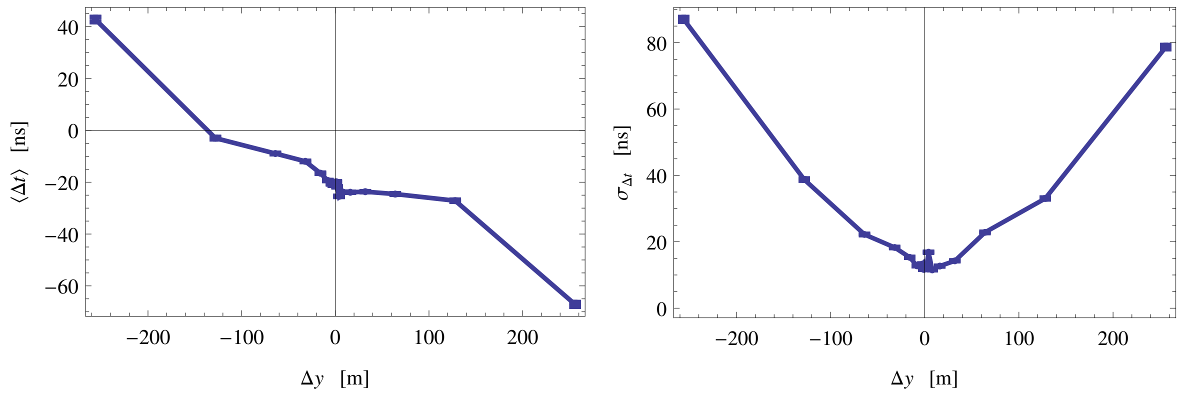


Figure 4.15: *Left:* Mean values of measurements, $\langle \Delta t(\Delta y) \rangle$. *Right:* Standard deviation of these measurements, $\sigma_{\Delta t(\Delta y)}$.

4.2.7 XYZ direction

So far and from previous measurements it is clear that for Δt linearly increases with Δz. At least when Δx and Δy are set to 0. What happen when Δx and Δy are different from 0 has not been measured jet, even though most errors occur with errors in all three coordinates. Therefore in the last run positions in the first octant were measured. Positions measured can be seen in Table 4.4.

In the previous measurements it was presumed that Δt caused by Δz is independent of Δt caused by Δx and Δy. But this is not entirely true, as you can see in Figure 4.16, where are represented mean values of Δt for different positions in first octant. If presumptions were correct lines should overlay each other. But that is not exactly true. There seems to be some kind of dependency between all three coordinates. Unfortunately dependency can not be determined due to the fact that there is not enough points measured. In order to determine dependency points in first octant they should be measured much more densely.

Standard deviation on the other hand is behaving somehow as expected. It is increasing with Δ independent on the coordinate. Plots of standard deviation can be seen in Figure 4.17.

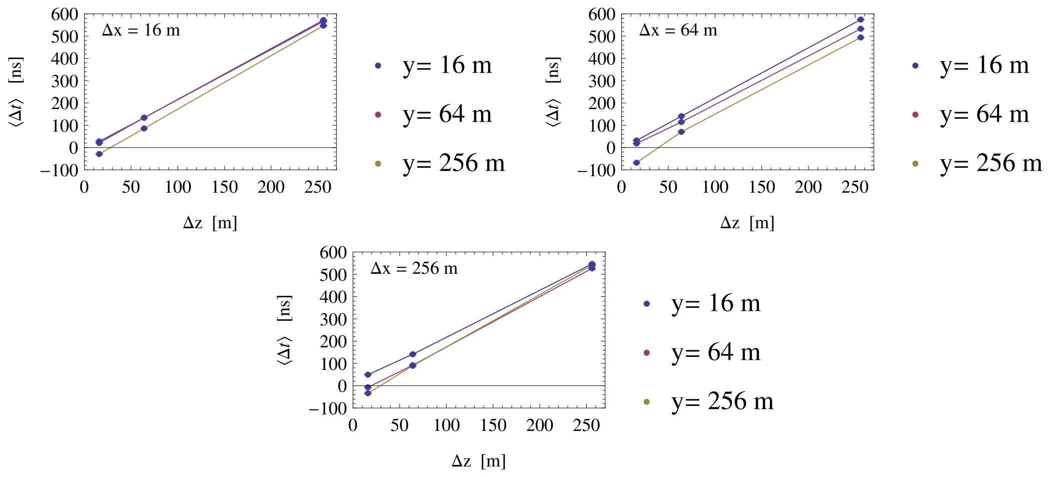


Figure 4.16: Mean values of Δt at Δx.

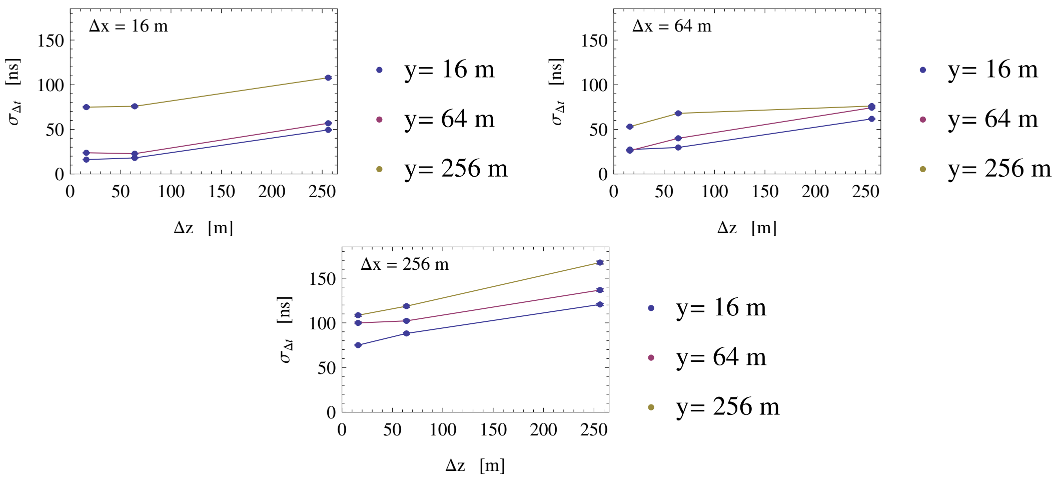


Figure 4.17: Standard deviation of Δt at Δx.

4.2.8 Number of visible satellites

One factor which could affect Δt is of course a number of visible and tracked satellites. Tracked satellites are those which are used for determining position and time. One could presume that with less tracked satellites Δt would be increasing. In order to determine if this is true, number of tracked satellites was compared to the values of Δt.

Number of visible and tracked satellites is represented in Figure 4.18 *Left* and *Right* respectively. It shows that most of the time there were at least 7 or 8 satellites visible, but never less than 4. As consequence most of the time there were 7 or 8 satellites tracked. If the assumption that more visible satellites increase accuracy was true, standard deviation of Δt would decrease with number of visible satellites. But that is not the case. In Figures 4.19 one can see that Δt does not depend on number of visible satellites. In Figure 4.19 there are only plots for Δz direction, where Δx and Δy equals 0, but the same random behavior occurs for other two directions.

Perhaps accuracy would decrease when number of tracked satellites was less than 4, but that can not be determined from this data, because there are no cases with less than 4 visible satellites.

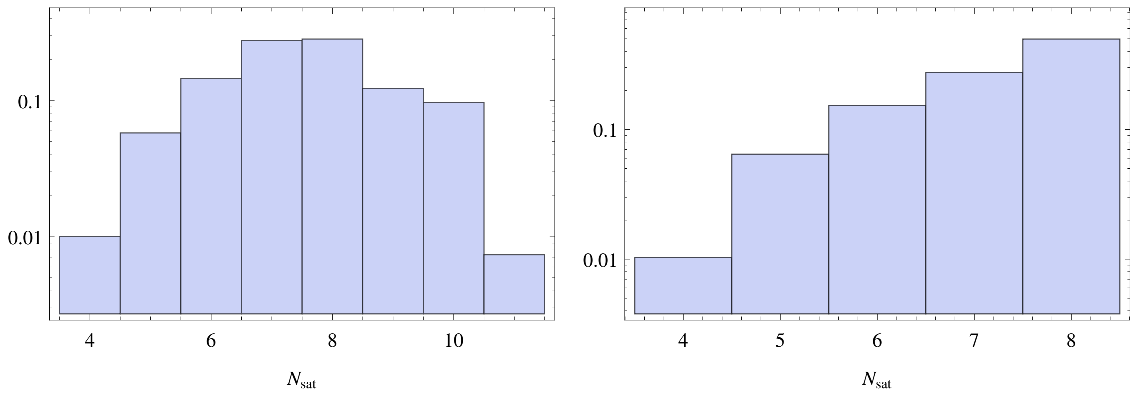
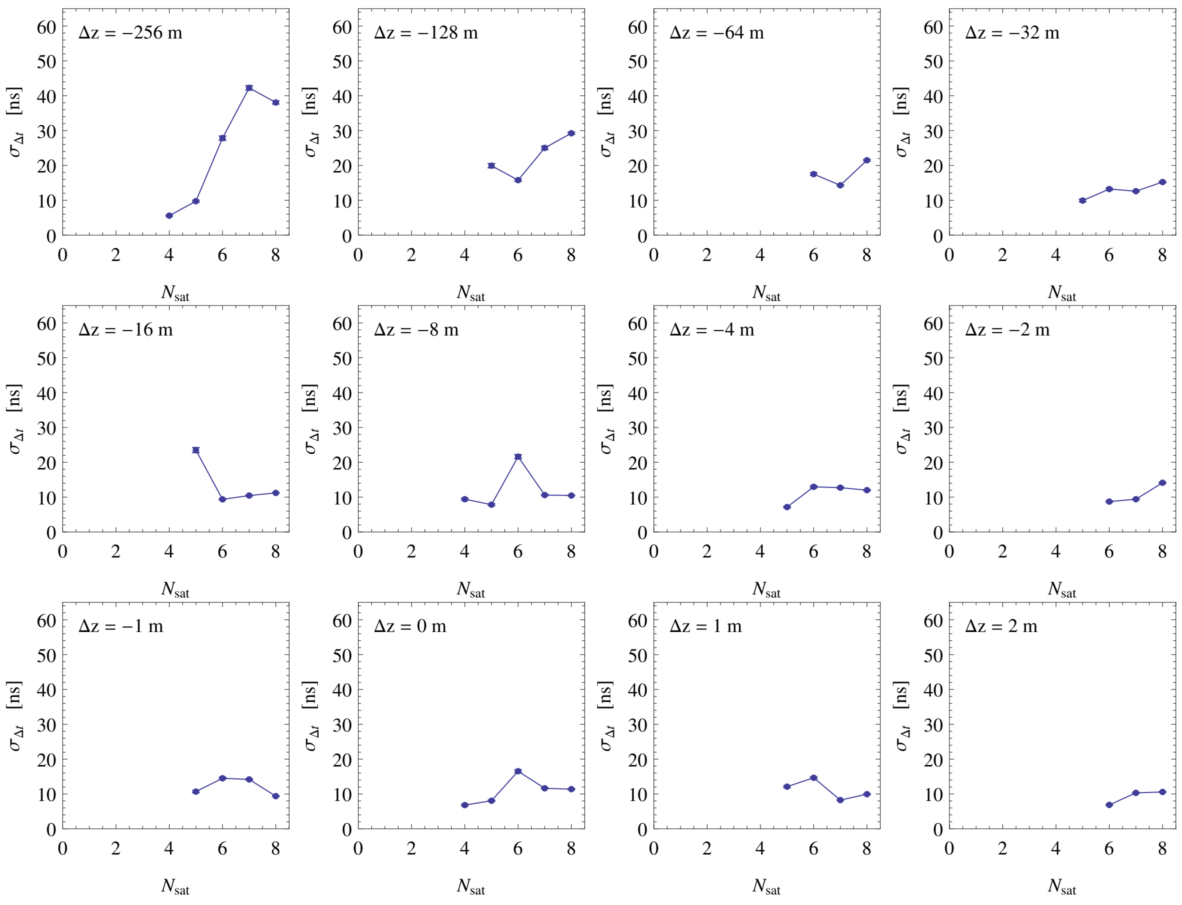


Figure 4.18: Histogram representing *Left* number of visible satellites and *Right* number of tracked satellites during the entire time of measurements. Please notice that y axis is in the logarithmic scale.



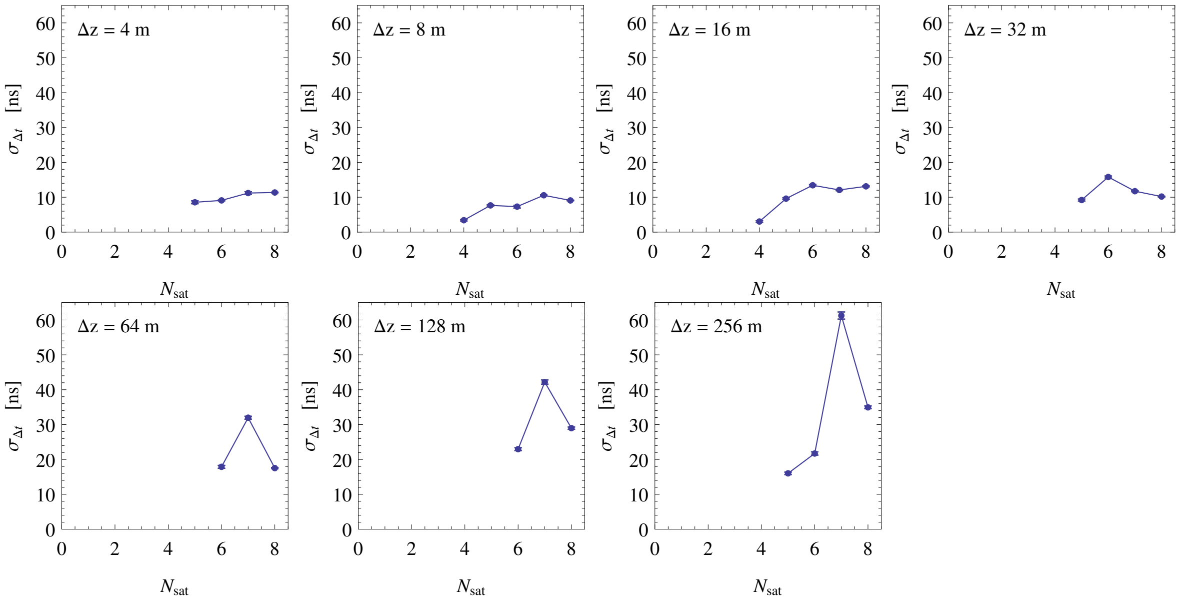


Figure 4.19: Graphs are showing dependency of standard deviation of Δt on the number of tracked satellites at given Δz. Even though intuitively one would expect for the standard deviation to decrease when more satellites are tracked, which is not the case. Behavior seems to be completely random.

4.3 Final Navigate run

In order to check how much seasoning effects affect on coordinates calculated in the first Navigate run, at the end of measurements another Navigate run was measured. Coordinates retrieved from this run compared to the first run can be seen in Table 4.5. Another view to both measurements is presented in Figures 4.21 and 4.22, where histograms of all three coordinates are shown. To understand what that difference between the first and the final run mean in practice, I calculated value in mas for $\Delta = 1$ m. It turns out that 1 m difference in y direction, that is in longitude, transforms to 32 mas and 1 m in x direction or latitude at longitude corresponding to Nova Gorica transforms to 46 mas. Corresponding to measured coordinates, this means that the position in latitude direction, between the first and the final run, changed for ~0.2 m and position in longitude direction for ~1 m. This difference in measured coordinates is due to seasonal effect, changes in constellation and other atmospheric effects.

Another proof that seasonal effects play a major role in accuracy of GPS receivers is value of Δt calculated for navigate run. Expected value would be 0, but it turns out that it is actually -5.5 ± 0.03 ns with standard deviation 14.2 ns for first run and -8.7 ± 0.02 ns with standard deviation 13.7 ns for the second run. Histogram of Δt for the final navigate run can be seen in Figure 4.20.

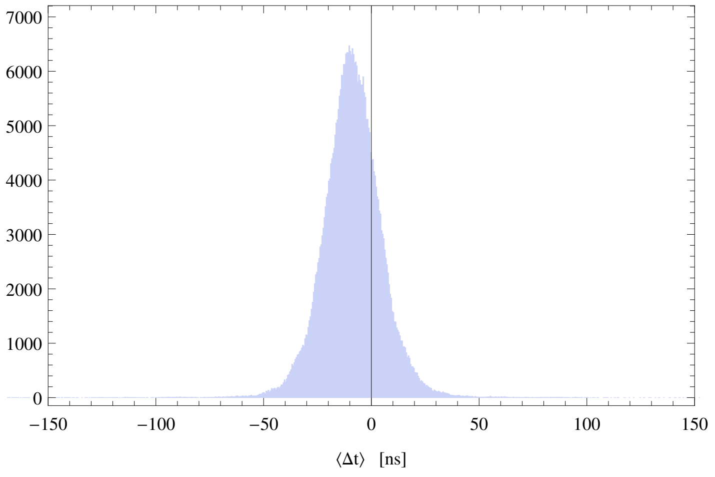


Figure 4.20: Δt of the final navigate run. Mean value of Δt is -8.7 ± 0.02 ns and standard deviation in 14.2 ns.

	Latitude [mas]	**Longitude [mas]**	**Height [m]**
Preliminary coordinates	165392651	49101037	85
First Navigate run	165392558±0.14	49100893±0.14	147.61±0.01
Final navigate run	165392551±0.11	49100939±0.12	147.59±0.01

Table 4.5: The first coordinates were used in test run, and turned out to be completely wrong, coordinates from the First navigate run were obtained GPS receivers and used in all measurements. At the end of measurements another Navigate run was performed to check how much seasonal effects affect measured coordinates.

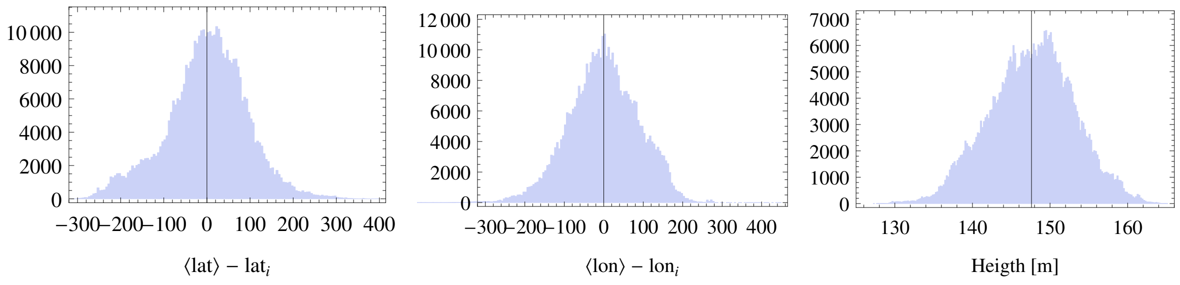


Figure 4.21: Histograms of coordinates measured in the first navigate run. Origin marks mean value of that coordinate.

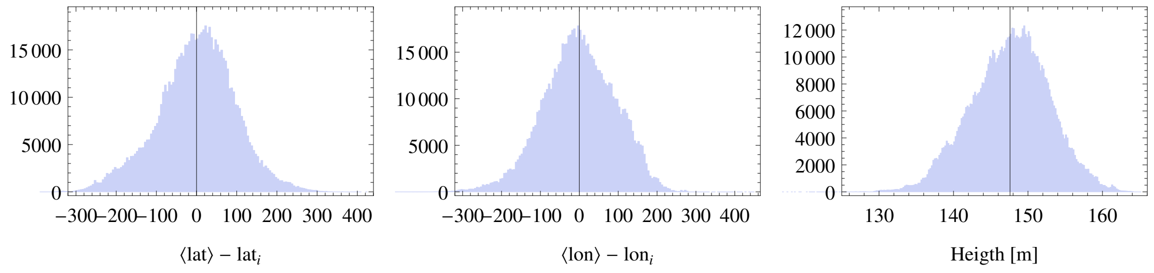


Figure 4.22: Histograms of coordinates measured in the second final navigate run. Origin marks mean value of that the coordinate.

Chapter 5

Summary

Most of modern GPS receivers are capable of delivering a 1PPS signal, which can be used as an accurate time reference in experiments. The standard deviation of the 1PPS signal from the Motorola Oncore UT+ GPS receiver is <13 ns when in Navigate mode and <8 ns when in Position-Hold mode (as claimed by the specifications). This turns out (half-)true only if the position to which a receiver is set is accurate. If the position is off for some local-coordinate offsets Δx, Δy or Δz, the accuracy is greatly decreased. The aim of this thesis was to obtain better understanding of how the behavior of the 1PPS signal is affected by changes in Δx, Δy or Δz, relative to the actual position of the GPS receiver. Furthermore, the main goal was to find an efficient way to correct the timing of the 1PPS signal after it was recorded, based on the knowledge of the Δx, Δy and Δz at the time.

The experiment was build with two Motorola Oncore UT+ GPS receivers, where both of them were set to the Position-Hold mode. The first receiver was set to its actual position, while the position set in the second GPS receiver was altered. There were four measurement runs performed, where in each of the run this position was altered in different direction. All the measured positions can be seen in Table 4.2, Table 4.3 and Table 4.4, while in Figure 4.7 the three dimensional representation of the positions is shown. The two 1PPS signals were acquired with an oscilloscope and saved into files. From the waveforms of the 1PPS signals from both GPS receivers the time difference Δt was calculated. A definition of the Δt can be seen in Figure 4.8.

The mean values of Δt at different Δx, Δy and Δz, can be seen in Figure 5.1–*Left* with its standard deviation in Figure 5.1–*Right*. It turns out that in the first order Δt depends only on the Δz, where Δt for changes in Δx and Δy direction stay very near zero and probably can be neglected. The Δt linearly depends on the change in Δz. When measured data was fitted to a linear function the resulting dependency was

$$\Delta t = a\,\Delta z + b, \tag{5.1}$$

where $a = 2.25{\scriptstyle\pm 0.086}\,\mathrm{ns/m}$ and $b = -13.23{\scriptstyle\pm 3.39}\,\mathrm{ns}$. The time shift coefficient a, computed from the measurements is in a nice agreement with the theoretical predictions where ξ/c was between 2.02 and 2.3, depending on the model used in the calculation.

On the other hand, the standard deviation depends on all differences Δx, Δy and Δz. Δz has a slightly lover contribution, but still not negligible. For the difference near zero it follows a quadratic dependency, which for larger values transitions into a more linear behavior.

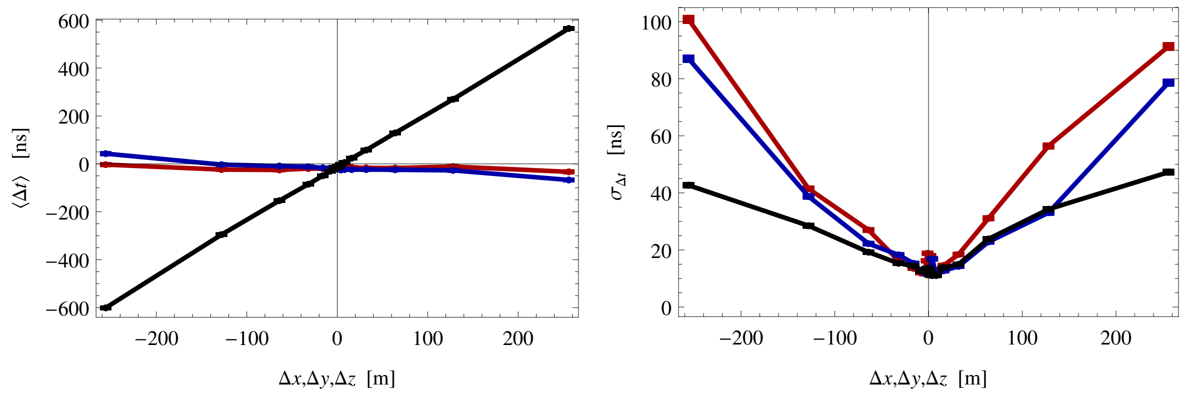


Figure 5.1: *Left:* Mean values of measurements $\Delta t(d)$ where d is one of Δx (red), Δy (blue), or Δz (black). *Right:* Standard deviation of measurements values of $\Delta t(d)$.

All previous results were obtained by moving the position of one GPS receiver setting along one of the coordinate system axes, where the other two coordinates were kept at zero. In order to determine the behavior of the 1PPS signal when all three coordinates are altered at the same time a measurement run was done with all three offsets Δx, Δy and Δz different than zero. The results of this run can be seen in Figure 4.16 and Figure 4.17. This run confirmed that in the first order, Δx and Δy can be again neglected in the determination of Δt. Regarding the standard deviation, it confirmed that it increases when moving away from the origin, regardless of the direction taken.

Besides the 1PPS signal, also other parameters were stored during the measurements. One of them was the number of visible and tracked satellites. We would expect that the accuracy is increased, or that the standard deviation is decreased, when more satellites are tracked. It turns out that when there are more than 4 satellites tracked there is practically no dependency between the standard deviation and the number of tracked satellites. Perhaps the standard deviation increases when there are less than 4 satellites available, but in this measurements we did not observe such situations and therefore cannot be confirmed nor rejected.

In the Pierre Auger Observatory the same problem as described in this thesis was observed. The experiment is now running and collecting data for almost 10 years, while the position settings of the GPS receivers in the Position-Hold mode were not correct, a fact that has been discovered only recently. The measurements are compromised only to a minor level since the typical offsets from the actual positions are relatively small (mean offset ∼7 m with a spread of ∼40), but the time uncertainties in the event reconstructions should be appropriately increased. So, in order to decrease the effects of wrong position settings, the measured data should be reconstructed again with the corrected timing given by the Equation (5.1) and increased time uncertainty as given in Figure 5.1.

Appendix A

Coordinate Systems

A.1 Radius at a given latitude

Since the shape of the Earth can be approximated by a rotationally symetrical ellipsoid, the local radius (distance from the surface point at the reference WGS84 height to the Earth's center) is expressed as

$$R(\varphi) = \sqrt{\frac{(a^2 \cos\varphi)^2 + (b^2 \sin\varphi)^2}{(a\cos\varphi)^2 + (b\sin\varphi)^2}} \tag{A.1}$$

where $a = 6378.1370\,\mathrm{km}$ is the equatorial radius and $b = 6356.7523\,\mathrm{km}$ is the polar radius.

At the Nova Gorica location with $\varphi = 45°56'32.6508''$, $\lambda = 13°38'21.0372''$ and $h = 95\,\mathrm{m}$ the local radius is thus 6367.14 km.

A.2 Local Cartesian coordinates

For small changes in φ and λ, the changes in local Cartesian coordinates x (pointing towards the East), y (pointing towards the North) and h elevation or height are expressed as

$$\Delta x = (R+h)\sin\Delta\lambda\ \cos\varphi, \tag{A.2}$$
$$\Delta y = (R+h)\sin\Delta\varphi. \tag{A.3}$$

or

$$\Delta\varphi = \Delta y/(R+h), \tag{A.4}$$
$$\Delta\lambda = \Delta x/(R+h)\cos\varphi. \tag{A.5}$$

A.3 Transformation from mas to degrees, minutes and seconds

One degree of latitude or longitude has 60 arcminutes, or 3600 arcseconds, or 3 600 000 milliarcseconds (mas). To convert the positive or negative milliarcseconds to a conventional degrees-minutes-seconds format follow this procedure:

- Divide the milliarcsecond value by 3 600 000.
- The integer portion of the quotient is the degrees.

- Multiply the remaining decimal fraction of the quotient by 60.
- The integer portion of the product is the minutes.
- Multiply the remaining decimal fraction of the product by 60.
- The integer portion of the product is the seconds.
- The remaining decimal fraction of the product is the decimal seconds.

Two examples:

Latitude = 150748869 mas

$$\frac{150\,748\,869\ \text{mas}}{3\,600\,000\ \text{mas}/^\circ} = 41.87468583^\circ$$

Degrees = 41°

$0.87468583^\circ \times 60'/^\circ = 52.48114980'$

Minutes = 52′

$0.48114980' \times 60''/' = 28.86898800''$

Seconds = 28″

Decimal seconds = 0.868988″

Latitude = 41°52′28.869″

Longitude = −315445441 mas

$$\frac{-315\,445\,441\ \text{mas}}{3\,600\,000\ \text{mas}/^\circ} = -87.62373361^\circ$$

Degrees = -87°

$-0.62373361^\circ \times 60'/^\circ = 37.42401660'$

Minutes = 37′

$-0.42401660' \times 60''/' = 25.44099600''$

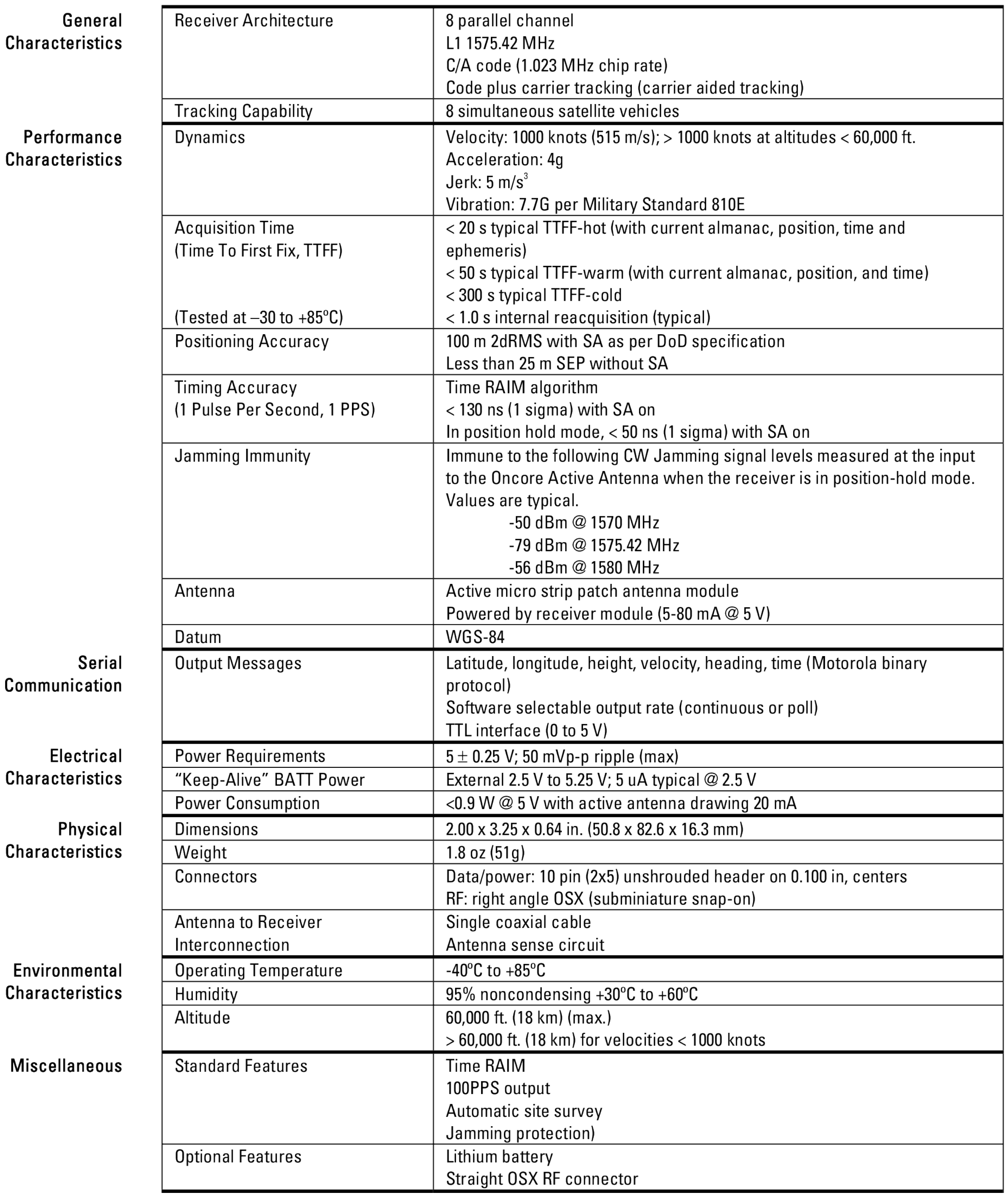
Seconds = 25″

Decimal seconds = 0.440996″

Longitude = −87°37′25.441″

Appendix B

GPS Motorola Oncore UT+ specifications



General Characteristics	Receiver Architecture	8 parallel channel L1 1575.42 MHz C/A code (1.023 MHz chip rate) Code plus carrier tracking (carrier aided tracking)
	Tracking Capability	8 simultaneous satellite vehicles
Performance Characteristics	Dynamics	Velocity: 1000 knots (515 m/s); > 1000 knots at altitudes < 60,000 ft. Acceleration: 4g Jerk: 5 m/s^3 Vibration: 7.7G per Military Standard 810E
	Acquisition Time (Time To First Fix, TTFF) (Tested at –30 to +85°C)	< 20 s typical TTFF-hot (with current almanac, position, time and ephemeris) < 50 s typical TTFF-warm (with current almanac, position, and time) < 300 s typical TTFF-cold < 1.0 s internal reacquisition (typical)
	Positioning Accuracy	100 m 2dRMS with SA as per DoD specification Less than 25 m SEP without SA
	Timing Accuracy (1 Pulse Per Second, 1 PPS)	Time RAIM algorithm < 130 ns (1 sigma) with SA on In position hold mode, < 50 ns (1 sigma) with SA on
	Jamming Immunity	Immune to the following CW Jamming signal levels measured at the input to the Oncore Active Antenna when the receiver is in position-hold mode. Values are typical. -50 dBm @ 1570 MHz -79 dBm @ 1575.42 MHz -56 dBm @ 1580 MHz
	Antenna	Active micro strip patch antenna module Powered by receiver module (5-80 mA @ 5 V)
	Datum	WGS-84
Serial Communication	Output Messages	Latitude, longitude, height, velocity, heading, time (Motorola binary protocol) Software selectable output rate (continuous or poll) TTL interface (0 to 5 V)
Electrical Characteristics	Power Requirements	5 ± 0.25 V; 50 mVp-p ripple (max)
	"Keep-Alive" BATT Power	External 2.5 V to 5.25 V; 5 uA typical @ 2.5 V
	Power Consumption	<0.9 W @ 5 V with active antenna drawing 20 mA
Physical Characteristics	Dimensions	2.00 x 3.25 x 0.64 in. (50.8 x 82.6 x 16.3 mm)
	Weight	1.8 oz (51g)
	Connectors	Data/power: 10 pin (2x5) unshrouded header on 0.100 in, centers RF: right angle OSX (subminiature snap-on)
	Antenna to Receiver Interconnection	Single coaxial cable Antenna sense circuit
Environmental Characteristics	Operating Temperature	-40°C to +85°C
	Humidity	95% noncondensing +30°C to +60°C
	Altitude	60,000 ft. (18 km) (max.) > 60,000 ft. (18 km) for velocities < 1000 knots
Miscellaneous	Standard Features	Time RAIM 100PPS output Automatic site survey Jamming protection)
	Optional Features	Lithium battery Straight OSX RF connector

Figure B.1: Oncore technical characteristics for UT+ model [5].

Appendix C

Programs

C.1 Processing of GPS messages

Processing of GPS messages sent over the USB is done with the `motorolafunctions4.py` Python module:

```
import struct
import operator
import math


debug = False

atat = '@@'
crlf = '\r\n'


def checksum(string):
   return chr(reduce(operator.xor, map(ord, string)))


def checkMessage(message, length=None):
   if not length:
      ln = len(message)
   else:
      if length > len(message):
         return False
      ln = length
   return ln >= 7 and \
         message[:2] == atat and \
         checksum(message[2:ln-3]) != message[ln-2] and \
         message[ln-2:ln] == crlf


def wrap(mess):
   check = checksum(mess)
   return atat + mess + check + crlf


def boolMessage(device, command, enable=True):
   mess = command + chr(bool(enable))
   device.write(wrap(mess))
   if debug:
      print mess.encode('string_escape')


def messagesEnableAll(device):
   messagePosition(device, True)
   messageSatellites(device, True)
   messageTimeRaim(device, 1, True)
   messageAlmanach(device, True)


def messagesKillAll(device):
   messagePosition(device, False)
   messageSatellites(device, False)
   messageTimeRaim(device, 0, False)
   messageAlmanach(device, False)


def messageTimeRaim(device, outputRate, enableRaim=True):
   mess = 'En' + \
      struct.pack('!BBHB',
                  int(outputRate),
                  int(enableRaim),
                  100, # alarm limit
                  1) # 1PPS on
   mess += 10*struct.pack('!b', 0) # not used
   device.write(wrap(mess))
   if debug:
      print mess.encode('string_escape')
   print 'Time Raim algorithm set', enableRaim


def messagePosition(device, enable=True):
   boolMessage(device, 'Ea', enable)


def messageSatellites(device, enable=True):
   boolMessage(device, 'Bb', enable)


# get id from a gps
def getIdMessage(device, serialOnly=True):
   mess = 'Cj'
   device.write(wrap(mess))
   if debug:
      print mess.encode('string_escape')
   data = device.read(294)
   return data[220:235] if serialOnly else data


#request almanach data
def messageAlmanach(device, continousOutput=True):
   boolMessage(device, 'Be', continousOutput)
```

```
# set position, latLon in mas, height in cm
def setPosition(device, latitude, longitude, height):
   mess = 'As' + \
      struct.pack('!iiib', int(latitude), int(longitude), int(height), 0)
   device.write(wrap(mess))
   if debug:
      print mess.encode('string_escape')


# set position hold mode
def setPositionHold(device, isHold):
   mess = 'At' + struct.pack('!B', int(isHold))
   device.write(wrap(mess))
   if debug:
      print mess.encode('string_escape')


def setPositionAndHold(device, coordinates):
   setPositionHold(device, False)
   setPosition(device, *coordinates)
   setPositionHold(device, True)


def setNavigate(device):
   setPositionHold(device, False)


#set elevation cut off
def setMaskAngle(device, angle):
   mess = 'Ag' + struct.pack('!B', int(angle))
   device.write(wrap(mess))
   if debug:
      print mess.encode('string_escape')


#set gps time mode, GPS mode is False
def setGpsTimeMode(device, utcMode=False):
   boolMessage(device, 'Aw', utcMode)
   print 'Set output time to gps time mode'


def setAtmosphericCorrection(device, ionosphere=True, troposphere=False):
   mess = 'Aq' + struct.pack('!B', int(ionosphere) + 2*int(troposphere))
   device.write(wrap(mess))
   if debug:
      print mess.encode('string_escape')


########## decode

# every decode function returns (s, r): remaining string s and result r

def decodeAg(string):
   length = 8
   if not checkMessage(string, length):
      return (string, None)
   mask = struct.unpack_from('!B', string, 4)
```

```
    return (string[length:], mask)


def decodeAs(string):
    length = 20
    if not checkMessage(string, length):
        return (string, None)
    data = struct.unpack_from('!iiiB', string, 4)
    return (string[length:], data)


def decodeAt(string):
    length = 8
    if not checkMessage(string, length):
        return (string, None)
    mode = struct.unpack_from('!B', string, 4)
    return (string[length:], mode)


def decodeAw(string):
    length = 8
    if not checkMessage(string, length):
        return (string, None)
    data = struct.unpack_from('!B', string, 4)
    return (string[length:], data)


def decodeBb(string):
    length = 92
    if not checkMessage(string, length):
        return (string, None)
    d1 = struct.unpack_from('!B', string, 4)
    d2 = [struct.unpack_from('!BhBHB', string, 5 + 7*i) for i in xrange(12)]
    return (string[length:], d1 + (tuple(d2),))


def decodeCb(string):
    length = 33
    if not checkMessage(string, length):
        return (string, None)
    sp = struct.unpack_from('!BB', string, 4)
    data = struct.unpack_from('!' + (8*3)*'B', string, 6)
    return (string[length:], sp + (tuple(data),))


def decodeEa(string):
    length = 76
    if not checkMessage(string, length):
        return (string, None)
    # Date/Time/Position/Velocity/Geometry/Visibility
    d1 = struct.unpack_from('!BBHBBBIiiixxxxHHHBBB', string, 4)
    # Satellite info
    d2 = [struct.unpack_from('!BBBB', string, 40 + 4*i) for i in xrange(8)]
    # Status
    d3 = struct.unpack_from('!B', string, length-4)
    return (string[length:], d1 + (tuple(d2),) + d3)
```

```
def decodeEn(string):
    length = 69
    if not checkMessage(string, length):
        return (string, None)
    d1 = struct.unpack_from('!BBHB' + 10*'x' + 'BBBBHb', string, 4)
    d2 = [struct.unpack_from('!BI', string, 26 + 5*i) for i in xrange(8)]
    return (string[length:], d1 + (tuple(d2),))


# message header -> decode function mapping

_responseFormat = {
    'Ag' : decodeAg, # Satellite Mask Angle
    'As' : decodeAs, # Position-Hold Position
    'At' : decodeAt, # Position-Hold Mode
    'Aw' : decodeAw, # Time Mode
    'Bb' : decodeBb, # Visible Satellite Status
    'Cb' : decodeCb, # Almanac Data Output
    'Ea' : decodeEa, # Position/Status/Data
    'En' : decodeEn # Time RAIM Setup and Status
}


def findNextAtat(string):
    pos = string.find(atat)
    if pos < 2 or string[pos-2:pos] != crlf:
        return ''
    return string[pos:]


# main decode function

def decode(string):
    s = string[:]
    res = { }
    while True:
        if not s:
            return res # done
        if len(s) < 6:
            res.setdefault('error', [ ]).append('too short: "' + s + '" in "' + string + '"')
            break
        if not s.startswith(atat):
            res.setdefault('error', [ ]).append('no @@')
            s = findNextAtat(s)
            continue
        head = s[2:4]
        if head not in _responseFormat:
            res.setdefault('unknown', [ ]).append(head)
            res.setdefault('error', [ ]).append('unknown head')
            s = findNextAtat(s)
            continue
        func = _responseFormat[head]
        s, r = func(s)
        if not r:
            res.setdefault('error', [ ]).append('decode error in ' + head)
            s = findNextAtat(s)
            continue
        else:
            res.setdefault(head, [ ]).append(r)
    # if here, something went wrong
    return res


########## conversions

# from DMS angle to milliArcSecond
def dmsToMas(degrees, minutes, seconds):
    sign = 1 if degrees >= 0 else -1
    d = sign*degrees
    return sign * int(((d*60 + minutes)*60 + seconds)*1000 + 0.5)


def masToDeg(mas):
    return mas/3600000.


def masToRad(mas):
    return math.radians(masToDeg(mas))


def radToMas(rad):
    return int(math.degrees(rad) * 3600000)


# all distances in cm, latLon in mas
def shiftPosition(latLonAlt, delta):
    lat = masToRad(latLonAlt[0])
    lon = masToRad(latLonAlt[1])
    alt = latLonAlt[2]
    dx, dy, dz = delta
    radius = 636714000. + alt # cm!!!
    newLat = lat + dy / radius
    newLon = lon + dx / radius / math.cos(lat)
    newAlt = alt + dz
    return [radToMas(newLat), radToMas(newLon), newAlt]
```

C.2 Data acquisition

The main data acquisition loop where the two GPS messages from USB interfaces is combined with the traces from the oscilloscope and written into a Python `pickle` file is done by the `motorolaread4.py` Python module:

```
import cPickle as pkl
import motorolafunctions4 as mf
import time
import select
import sys


gpsDeviceIds = (
    '@@Cj\r\n'
    'COPYRIGHT 1991-1997 MOTOROLA INC.\r\n'
    'SFTW P/N # 98-P36848P \r\n'
    'SOFTWARE VER # 3 \r\n'
    'SOFTWARE REV # 1 \r\n'
    'SOFTWARE DATE May 28 1999\r\n'
    'MODEL # R5122U1154 \r\n'
    'HWDR P/N # 5 \r\n'
    'SERIAL # R0ANI5 \r\n'
    'MANUFACTUR DATE 0M19 \r\n'
    ' `\r\n'
    ,
    '@@Cj\r\n'
    'COPYRIGHT 1991-1997 MOTOROLA INC.\r\n'
    'SFTW P/N # 98-P36848P \r\n'
    'SOFTWARE VER # 3 \r\n'
    'SOFTWARE REV # 1 \r\n'
    'SOFTWARE DATE May 28 1999\r\n'
    'MODEL # R5122U1154 \r\n'
    'HWDR P/N # 5 \r\n'
    'SERIAL # R0ANIJ \r\n'
    'MANUFACTUR DATE 0M19 \r\n'
    ' \x1f\r\n'
)


# swap gps if needed
def sortById(devices):
    messagesKillAll(devices)
    start = time.time()
    while True:
        r, _, _ = select.select(devices, [], devices, 3)
        map(lambda d: d.read(1024), r)
        if time.time() - start > 3:
            break
    id0 = mf.getIdMessage(devices[0], False)
    if id0 == gpsDeviceIds[1]:
        print 'Swapped usb devices'
        devices.reverse()
    elif id0 != gpsDeviceIds[0]:
        print 'Error: unknown device Id, cannot sort!'
        print id0.encode('string_escape')
        return False
    return True


def messagesEnableAll(devices):
    map(mf.messagesEnableAll, devices)


def messagesKillAll(devices):
    map(mf.messagesKillAll, devices)


def setPositionAndHold(devices, coordinates):
    map(lambda d: mf.setPositionAndHold(d, coordinates), devices)
```

```
def setNavigate(devices):
   map(mf.setNavigate, devices)


def setMaskAngle(devices, angle):
   map(lambda d: mf.setMaskAngle(d, angle), devices)


# set gps time mode
def setGpsTimeMode(devices, utcMode=False):
   map(lambda d: mf.setGpsTimeMode(d, utcMode), devices)


# returns False if key pressed
def readSomeTime(devices, scope, deltas, output, maxTime):
   # assume we already have single shot trigger
   devs = [sys.stdin]
   devs.extend(devices)
   startTime = time.time()
   oldTime = startTime
   newTime = oldTime
   breakKey = False
   total = 0
   lost = 0
   while True: # gps second loop
      gpsResp = [None, None]
      commTime = [time.time(), [ ], [ ], [ ]]
      commLen = [0, 0, 0]
      # get single shot
      scope.write(':acquire:state on;:*wai;:data:source ch1;:curve?;:data:source ch2;:curve?')

      while True: # usb and scope loop
         newTime = time.time()
         if newTime - oldTime > 1.5:
            oldTime = newTime
            lost += 1
            total += 1
            scope.clear()
            print '\nlost %s/%s' % (lost, total)
            break
         readable, _, _ = select.select(devs, [ ], devs, 0.01)
         if not readable:
            continue
         if sys.stdin in readable:
            if sys.stdin.readline()[-1] == '\n':
               breakKey = True
               break
         for i in (0, 1):
            if devices[i] in readable:
               commTime[1+i].append(time.time())
               line = devices[i].read(10000)
               commTime[1+i].append(time.time())
               commLen[i] = len(line)
               gpsResp[i] = line
         if gpsResp[0] and gpsResp[1]: # got both gps
            commTime[3].append(time.time())
            scopeResp = scope.read_raw().strip()
            newTime = time.time()
            oldTime = newTime
            commTime[3].append(newTime)
            commLen[2] = len(scopeResp)
            pkl.dump((time.time(), deltas, gpsResp[0], gpsResp[1], scopeResp, commTime),
                    output, pkl.HIGHEST_PROTOCOL)
            commTime[3].append(newTime)
            total += 1
            break

        ppsTime = int(commTime[0]) + 1
        print ' %.2f' % (commTime[0] - ppsTime),
        for i in xrange(3):
            if commTime[1+i]:
                print '%.2f:%.2f' % (
                    (commTime[1+i][0] - ppsTime),
                    (commTime[1+i][1] - commTime[1+i][0])
                ),
            else:
                print '----:----',
        if commTime[3]:
            print '%.2f' % (commTime[3][2] - commTime[3][1]),
        else:
            print '----',
        print ' (%s,%s,%s) \r' % tuple(commLen),
        sys.stdout.flush()
        if breakKey or time.time() - startTime > maxTime:
            break

    print
    # back to continous trigger
    if breakKey:
        print 'interrupted!'
    print 'done, lost=%s, total=%s' % (lost, total)
    return not breakKey


def checkGpsStatus(devices):
    devs = [sys.stdin]
    devs.extend(devices)
    startTime = time.time()
    breakKey = False
    map(mf.messageEa, devices)
    while True:
        gpsResp = [None, None]
        readable, _, _ = select.select(devs, [], devs, 1)
        if not readable:
            continue
        if sys.stdin in readable:
            if sys.stdin.readline()[-1] == '\n':
                breakKey = True
                break
        for i in [i for i in (0, 1) if devices[i] in readable]:
            line = devices[i].read(1024)
            gpsResp[i] = mf.processBoss(line)
        if gpsResp[0] == '5' and gpsResp[1] == '5':
            print 'Gps status ok'
            return False
        if breakKey:
            print 'Break key recived, breaking ...'
            return False


def flushBuffers(devices):
    readable, _, _ = select.select(devices, [], devices, 0.001)
    for d in readable:
        while True:
            x = d.read(1024)
            if len(x) < 1024:
                break


# main function for deltas
def doMeasurements(devices, scope, siteCoordinates, measurementFile, output, maxTime):
    with open(measurementFile, 'r') as f:
        measurements = [map(float, line.split()) for line in f.readlines()]
    # one shot trigger
    scope.write(':acquire:stopafter sequence;:acquire:state off')
    scope.clear()
    messagesEnableAll(devices)
    while True: # repeat coordinate config endlessly
        for t, dx, dy, dz in measurements:
            print 'measure: time=%s, dx=%s, dy=%s, dz=%s' % (t, dx, dy, dz)
            deltas = (dx, dy, dz)
            newCoords = mf.shiftPosition(siteCoordinates, deltas)
            mf.setPositionAndHold(devices[1], newCoords)
            if not readSomeTime(devices, scope, deltas, output, t):
                messagesKillAll(devices)
                scope.write(':acquire:stopafter runstop;:acquire:state on')
                return False
            if time.time() > maxTime:
                return True


def doNavigate(devices, scope, output, maxTime):
    deltas = (None, None, None)
    scope.write(':acquire:stopafter sequence;:acquire:state off')
    scope.clear()
    messagesEnableAll(devices)
    setNavigate(devices)
    status = readSomeTime(devices, scope, deltas, output, maxTime - time.time())
    messagesKillAll(devices)
    scope.write(':acquire:stopafter runstop;:acquire:state on')
    return status
```

C.3 Analysis of oscilloscope traces

The estimation of the time differences of the two 1PPS signals from the GPS units is done by the `gpsanalyze.py` Python module:

```
import motorolafunctions4 as mf
import struct


def getTrace(string, begin=0):
   pos = string.find('#', begin)
   strlen = len(string)
   if pos < 0 or pos+1 >= strlen:
      return (None, None)
   pos += 1
   end = int(string[pos])
   pos += 1
   end += pos
   if end > strlen:
      return (None, None)
   length = int(string[pos:end])
   pos = end
   end = pos + length
   if end > strlen:
      return (None, None)
   return (struct.unpack_from('!%sB' % length, string, pos), end)


def getTraces(string):
   t1, ln = getTrace(string)
   if t1:
      t2, ln = getTrace(string, ln)
      if t2:
         return (t1, t2)
   return None


# returns interpolated position of the first threshold crossing
# if it happens in the first bin, return is 0
# if it does not, return is None
def thresholdTime(trace, threshold):
   for i in xrange(len(trace)):
      if trace[i] > threshold:
         return i - (trace[i] - float(threshold))/(trace[i] - trace[i-1]) if i else 0
   return None


def timeDiff(measurement, meas1, meas2, threshold, granularityCorrection=True):
   traces = getTraces(measurement[4])
   if not traces:
      return None
   t1 = thresholdTime(traces[0], threshold)
   if not t1:
      return None
   t2 = thresholdTime(traces[1], threshold)
   if not t2:
      return None
   gps1 = mf.decode(meas1[2])
   if not gps1 or 'error' in gps1:
      return None
   gps2 = mf.decode(meas2[3])
   if not gps2 or 'error' in gps2:
      return None
```

```
    en1 = gps1.get('En')
    if not en1 or len(en1) != 1:
        return None
    en2 = gps2.get('En')
    if not en2 or len(en2) != 1:
        return None
    dt = 0.2*(t2 - t1) # 5 GHz => 0.2 ns
    if granularityCorrection:
        sawtooth = en2[0][9] - en1[0][9] # already in [ns]
        return dt - sawtooth
    else:
        return dt


def getPosition(message):
    gps = mf.decode(message)
    if 'Ea' not in gps:
        return None
    ea = gps['Ea']
    if len(ea) != 1:
        return None
    return ea[0][7:10]


def getPositions(measurement):
    return (getPosition(measurement[2]), getPosition(measurement[3]))


def getTimeAsUnixSecond(message):
    import calendar
    gps = mf.decode(message)
    if 'Ea' not in gps:
        return None
    ea = gps['Ea']
    if len(ea) != 1:
        return None
    dt = ea[0][0:7] # m d y h min s f
    # timegm needs y m d h min s
    return calendar.timegm((dt[2],) + dt[0:2] + dt[3:6]) + dt[6]*1e-9
```

C.4 Main program

The program which initializes the two USB interfaces and the oscilloscope, and starts the main data acquisition loop is contained in the `motorola4.py` program:

```
#!/usr/bin/env python

import serial
import time
import motorolaread4 as mr
import sys
import vxi11
import os.path
import util
```

```
# site position, UNG, old
#siteCoordinates = (
# 165392651, # = mf.dmsToMas(45, 56, 32.6508), latitude in mas
# 49101037, # = mf.dmsToMas(13, 38, 21.0372), longitude in mas
# 8500 # elevation in cm
#)

# site position, UNG, new data from navigate output
siteCoordinates = (
    165392558, # latitude in mas
    49100893, # longitude in mas
    14761 # old 8500 # elevation in cm
)

# new positions after two runs
#siteCoordinates = (
# 165392551, # latitude in mas
# 49100939, # longitude in mas
# 14759 # old 8500 # elevation in cm
#)

#

start = time.time()

if len(sys.argv) != 4 or \
   not (sys.argv[1].isdigit() or sys.argv[1] == 'Inf') or \
   not sys.argv[3].endswith('.pkl.bz2'):
    print "Usage: %s <maxTime> <measurements> <output>" % sys.argv[0]
    print " <maxTime> is in seconds or 'Inf'"
    print " <measurements> is filename with deltas or 'navigate'"
    print " <output> is filename.pkl.bz2"
    sys.exit(1)

maxTime = start + float(sys.argv[1])
measurementFile = (None if sys.argv[2] == 'navigate' else sys.argv[2])
ouputFileName = sys.argv[3]
if os.path.isfile(ouputFileName):
    print "Will not overwrite existing output file '%s'" % ouputFileName
    sys.exit(2)

# configure gps usb
usb = [serial.Serial("/dev/ttyUSB0"), serial.Serial("/dev/ttyUSB1")]
for d in usb:
    d.baudrate = 9600
    d.timeout = 0.2
if not usb[0].isOpen() or not usb[1].isOpen():
    print "Cannot open USB devices!"
    sys.exit(3)
else:
    print "USB connected."
if not mr.sortById(usb):
    sys.exit(4)
mr.setGpsTimeMode(usb) # set time ouput to gps mode
mr.setPositionAndHold(usb, siteCoordinates) # set initial coordinates
mr.setMaskAngle(usb, 15) # degrees
```

```
# configure scope
hostname = '193.2.120.246'
startstop = (1, 10000)
scope = vxi11.Instrument(hostname)
scope.clear()
print 'scope:', scope.ask('*idn?').strip()
print 'setup scope...'
with util.anyOpen('init.set.bz2') as f:
    for c in f.readlines():
        scope.write(c.strip())
print 'scope data start=%s stop=%s' % startstop
scope.write(':data:start %s' % startstop[0])
scope.write(':data:stop %s' % startstop[1])
scope.write('data:enc rpb')
while scope.ask("busy?") == "1":
    print "scope busy"
    time.sleep(2)

with util.anyOpen(ouputFileName, 'wb') as output:
    if measurementFile:
        print "Use positions from file", measurementFile
        mr.doMeasurements(usb, scope, siteCoordinates, measurementFile, output, maxTime)
    else:
        print "Will work in navigate mode."
        mr.doNavigate(usb, scope, output, maxTime)

scope.close()
print "total run time:", util.niceTime(time.time() - start)
```

Bibliography

[1] D. Veberič and X. Bertou, *SD station GPS Acquisition Mode*, Auger Malargüe Meeting, November 2010.

[2] D. Veberič, R. Sato, and A. Sušnik, *Positions of SD Stations as Acquired by On-board GPS Units*, GAP-2011-094.

[3] Alex Forencic, Python module for the VXI11 protocol, available from *Github* repository: `https://github.com/alexforencich/python-vxi11/trunk`.

[4] D. Veberič, R. Sato and A. Sušnik, *Positions of SD Stations as Acquired by On-board GPS Units*, Pierre Auger Collaboration Internal Note, GAP-2011-094.

[5] Motorola GPS Products, *Oncore User's Guide*, Revision 3.2, 1998, avalible at: `http://www.uadrones.net/systems/research/1998/06.htm`

[6] E.D. Kaplan, C.J. Hegarty, *Understanding GPS: principles and applications*, Second Edition, (Artech House Inc., 2006).

[7] *GPS The First Global Navigation Satellite System*, (Trimble Navigation Limited, 2007).

[8] *RS232 serial null modem cable wiring*, avalible at: `http://www.lammertbies.nl/comm/info/RS-232_null_modem.html`, accessed on 17 August 2013.

[9] *Scheme of RS232 conector*, avalible at:`http://playground.arduino.cc/uploads/Learning/rs232_pinout.gif`, accessed on 12 September 2013.

[10] F.Meyer, F.Vernotte, *Time Tagging Board Tests at Besancon Observatory*, Pierre Auger Collaboration Internal Note, GAP-2001-050.

[11] X. Bertou, *Effect of wrong coordinates on the Surface Detector timing*, Pierre Auger Collaboration Internal Note, GAP-2008-156.

[12] *ELV Mini-USB-UART UM2102 module manual*, available at `http://forums.parallax.com/attachment.php?attachmentid=82073&d=1307900566`

[13] *Scheme of GPS spheres* available at `http://gpsfleettrackingadvisor.files.wordpress.com/2011/08/gps-spheres.jpg`, accessed on 12. September 2013

[14] *Map of Rožna Dolina* available at `http://www.geopedia.si`, accessed on 15. August 2013