STUDENTWORK

Final Report

GPS system for supervising rock movements phenomena

25.05.2011

Group:
Alan Klimala
Sławomir Zagórski

Final Report HO2–300 05/2011

http://prosjekt.hisf.no/~11git-gps/

HØGSKULEN i SOGN OG FJORDANE

www.hisf.no

TITLE: Final Report HO2-300	**DATE:** 25.05.2011	**REPORT NUMBER:** **3/2011**
PROJECT TITLE: GPS system for supervising rock movements phenomena	**ACCESS:** **URL:** http://prosjekt.hisf.no/~11git-gps/	**NUMBER OF PAGES:** **62**
GROUP: Alan Klimala Sławomir Zagórski	**SUPERVISORS:** Joar Sande Marcin Fojcik	
EMPLOYER: Sogn og Fjordane University College		

SUMMARY:
This project has been developed as a Senior Design Project. The main goal of this project has been to develop embedded system which is able to detect rock movements phenomena base on new proposed differential algorithm. In order to achieve this task there is a necessity to make analysis of using popular GPS system modules in cooperation with math algorithm and design efficient hardware supporting data gathering and processing them based on examined algorithm.

KEYWORDS:
H02-300, Senior Design Project – Final Report; GPS system for supervising rock movements phenomena

Foreword

This is the final report for the project that was developed as a Senior Design Project. The project aim was to prepare design of embedded system able to detect potential danger related to soil and rock instantaneous falls phenomena. It was made for the Sogn of Fjordane University College that was represented by Marcin Fojcik and Joar Sande. Project is also a part of the research programmes made in cooperation with Silesia University of Technology.

When we look at the project from the perspective of our whole studying period it was first time we have designed such complex system. On the one hand it was difficult task to implement proper differential math algorithm to designate correct distance between two points based on gathered GPS data without the need of being sure about the accuracy of the position`s measurements gathered in every data in set. On the other hand it was tough to design embedded system able to support gathering necessary data and efective enough to do complicated math computations bearing in mind low energy consumption.

We are very grateful to the Sogn of Fjordane University College for the opportunity to taking part in Senior Design Project as exchange students. Participation in this project allowed us to improve our experience in the real project lifetime cycle execution which will hopefully help us in professional work after studies.

Contents

1. Abbreviations and symbols....6
2. Resume....8
3. Introduction....9
3.1. Project aims....9
3.2. Division of work....9
4. Theory background....10
4.1. Global Positioning System....12
4.2. GPS errors sources....13
4.3. Modern low power embedded platforms....15
4.4. Wireless communication for embedded systems....16
5. Tools....18
5.1. Programming language....18
5.2. Integrated development environment....19
5.3. Source code management....20
5.4. PCB design software....20
5.5. Embedded development boards....21
6. Analysis of GPS system capabilities....23
6.1. Process of gathering data....23
6.2. Data processing....25
6.2.1. Data storing....26
6.2.2. Data parsing....28
6.2.3. Creation of data pairs....29
6.2.4. Distance computation....29
6.4. Tests....30
6.4.1 Distance A....31
6.4.2 Distance B....33
6.4.2 Distance C....37
6.5. Conclusions....39
6.6. Further development....39
7. Measurement wireless embedded system....41
7.1. XBee module breakout board....41
7.2. TS node device design....42
7.3. TS node firmware implementation....44
7.4. BS node device design....47
7.5. BS node firmware implementation....49
7.6. Conclusions....52
7.7. Further development....52
8. Organization....53
8.1. Order placement company....53
8.2. Project supervisors....53
8.3. Developers....54
9. Project administration....55
9.1. Carrying out project according to plan....55
9.2. Development Gantt diagram....55

9.3. Week schedule....56
9.4. Meeting schedule....57
9.5. Budget....58
11. List of figures....59
12. List of tables....60
13. List of appendixes....61
14. References....62

1. Abbreviations and symbols

APB	Advanced Peripheral Bus
ARM	Advanced RISC Machine Microcontrollers Architecture
Cortex-M3	ARM Processor Based on the ARMv7-M Architecture
CPU	Central Processing Unit
CRC	A cyclic redundancy check, an error-detecting code designed to detect accidental changes to raw computer data
CTS	Clear to Send serial communication signal used to synchronize data sending
DGPS	Differential Global Positioning System
EFM32G890	EnergyMicro's low power 32-bit microcontroller
Eclipse CDT	Fully functional and open source C and C++ Integrated Development Environment
ECTS	European Credit Transfer and Accumulation System
EUSART	Enhanced Universal Synchronous Asynchronous Receiver Transmitter communication module
Flash	Non-volatile computer storage chip that can be electrically erased and reprogrammed
GPS	The Global Positioning System
GUI	Graphical User Interface
HDOP	Horizontal Dilution of Precision, relative accuracy of horizontal position
HO2-300	Code of the Senior Design Project at the Sogn og Fjordane University College
IDE	Integrated Development Environment
MEO	Medium Earth orbit
Microchip	Microcontrollers and Analog Semiconductors Provider
NMEA	National Marine Electronics Association, also short for NMEA 0183
NMEA 0183	A combined electrical and data specification for communication between marine electronic devices
NOR Flash	Type of flash memmory in each cell has one end connected directly to ground, and the other end connected directly to a bit line.
Open OCD	Open source software that support on-chip debugging
OpenSource	Describes practices in production and development that promote access to the end product's source materials

PCB	Printed circuit board, used to mechanically support and electrically connect electronic components using conductive pathways, tracks or signal traces etched from copper sheets laminated onto a non-conductive substrate
PIC18F25K80	Microchip's low power 8-bit microcontroller
RAM	Random-access memory
RISC	Reduced Instruction Set Computing
SFUC	Sogn og Fjordane University College
SNR	Signal-to-noise ratio, a measure used in science and engineering to quantify how much a signal has been corrupted by noise
SourceryG++ Lite	GNU Toolchain for ARM processors development
SRAM	Static random-access memory
UART	Universal Asynchronous Receiver Transmitter communication module
VDOP	Vertical Dilution of Precision, relative accuracy of vertical position
ZigBee	Wireless Communication Protocol Using Low-power Digital Radios Based on the IEEE 802.15.4-2003 Standard

2. Resume

This report is a summary of work on an attempt to create a system allowing to detect the smallest changes in distance between two objects. The aim is to help the humanity in the detection of high risk situations, such as mudflows and rock slides, and thereby saves human lives.

The main objectives of this project is to use a standard GPS modules and place emphasis on the issue of energy consumption. In addition, the decisive factor in implementing such a system must be the cost of a single device, so it can be propagated as widely as possible.

The project was divided into two parts partially independent. First one aimed at the creation of hardware and second one at the analysis of collected data to create software that is able to detect these movements. The need for low power consumption forces the system to work as rarely as possible, and thus is not possible to apply statistical methods in the analysis of collected data.

Each of these parts had faced a big problem, which could not be overcome during the project execution phase. The hardware part problem was to not get needed equipment at the time, which made it impossible to create a firmware for the base station, while in the analytical part, it turned out that GPS modules have too little accuracy to be able to create a proper algorithm based on data from them.

However, despite this, we tried to get through this project as much information as we could to assist in the future studies. We hope that we succeeded, and that someone will benefit from our work. We hope that we succeeded, and that someone will benefit from our work, that system like this would be built.

3. Introduction

The project, named GPS system for supervising rock movements phenomena, is made as a subject of a Senior Design Project researches, with code H02-300, in the spring semester 2011. The order placement company is Sogn og Fjordane University College represented by Joar Sande and Marcin Fojcik. This project is a part of researches conducted by SFUC in cooperation with Silesian University of Technology, known as Geology Information Technology, to investigate the possibility of using modern technology to predict risks associated with geological phenomena.

3.1. Project aims

Main goals of this project is development of high power efficiency embedded system able to gather and process geolocation data and analysis of possibilty of use of differential algorithm, which is eventually a part of embedded system's firmware, to detect even the smallest change of position based on cheap and popular GPS modules.

3.2. Division of work

The complexity of the project required a deliberate dividing the work among the people involved in it to achieve satisfactory results within a specified time. The division was based on the skills possessed by each student at the initial stage. Whereas one of us already possessed the skills needed to use of mathematical analysis and statistical methods in order to find out some correlations between huges sets of time-varying data, the other already had some expirience with designing and programming embedded systems with limited resources. Because of that division of project seems to be obvious. This kind of parallel work provided highly efficent method of achieving objectives of our project without the need to gain large amounts of new knowledge in order to be able to implement each peace of whole complex system.

From now part of a project related to the statistical analysis we will be called "software part" and the part related to the embedded system design "hardware part", even though

one of its phases will focus on implementing the firmware.

4. Theory background

Using computer technologies is obvious, with regard to any kind of human activities. It seems to be obvious that one of these should be to ensure human safety. Modern technology gives us the means and opportunities to save property and even lives by developing systems for monitoring different sources of threat. We can collect huge sets of measurements of various physical parameters and try to find out some regularities and correlations between values in time in order to create forecast of many types of potential dangers such as sudden natural events. One of them is soil and rock instantaneous falls phenomena occurring particularly in mountainous areas. Because of natural terrain shape in Norway, there are a lot of areas where the risk of such threat occurrence is very high.

Gelogy specialists have already run some reaserches about predicting place and time of disaster by using some specially designed equipment. However, often the cost of purchasing the necessary equipment is high enough to effectively limit the scope of such systems.

In our opinion, technological progress and the related decline in prices of electronical parts are large enough to ask about the possibility of constructing a system that would be as efficient and reliable as already known expensive commercial solutions but remains cost-effective in manufacturing. This system should remain fully open to allow create improvements in its construction supported by the conclusions of the results of his work in real environment in the future.

We believe that this challenge is achievable and this project is an attempt to obtain such a system by exploring the possibility of using commercially available GPS system to detect changes in the position of objects relative to each other by using a differential algorithm to process data obtained from low-cost GPS modules. To accomplish this we must create a system that can be considered as a sensor network with one central master node (called further Base Station or just BS) and multiple slaves nodes (called further Terminal Stations or just TSs) staggered as far away from master as it is possible. Task of each TS node

would be the collection of GPS measurtements and transfer gathered data to a BS that should do the appropriate calculations in order to detect potential risks associated with the discovered displacement of the ground on which TS node has been initially placed.

Topology of considered nodes network is called point-to-multipoint and is depicted on Fig.1.

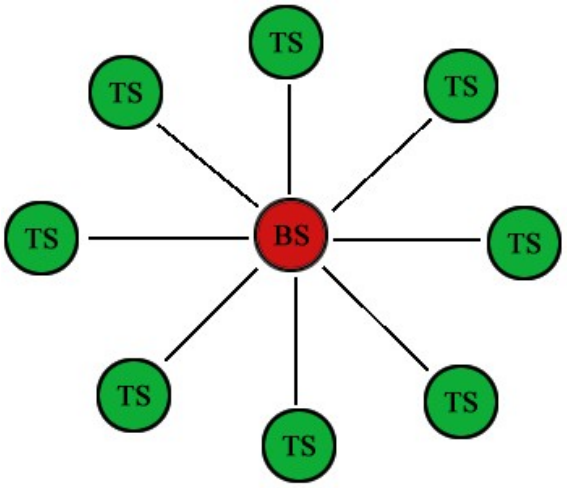


Figure 1: Proposed solution connections topology.

4.1. Global Positioning System

The Global Positioning System consists of 24 sattelites that are spaced around the Earth, their ground stations that monitor the status of system and user antennas and receivers.

GPS satellites fly in medium Earth orbit (MEO) at an altitude of approximately 20,200 km and they circles the Earth twice a day. MEO is presented at Figure 2 as green dash-dot line.

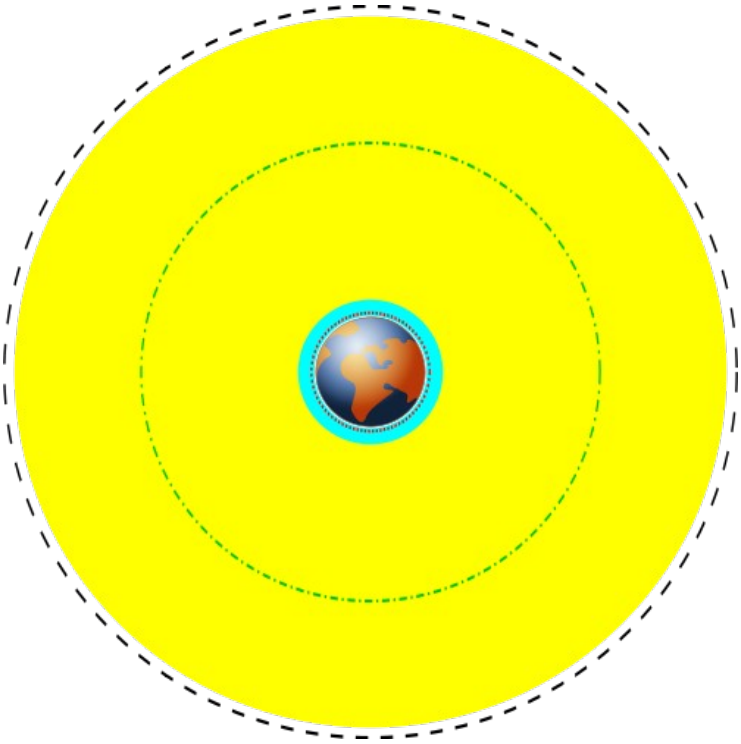


Figure 2: Orbits around the Earth [1]

At least four satellites are in view from virtually any point on the planet. This is ensured by 24-slot arrangement - the satellites in the GPS constellation are arranged into six orbital planes, each containing four primary satellites what can be seen at Figure 3.

In 2010, the Air Force announced plans to adjust the GPS constellation to implement an "Expanded 24" configuration *[3]*. When the changes are complete in late 2011, three of the orbital planes will be strengthened by one primary satellite each. Then GPS will effectively operate as 27-slot constellation what will improve coverage on most parts of the world.

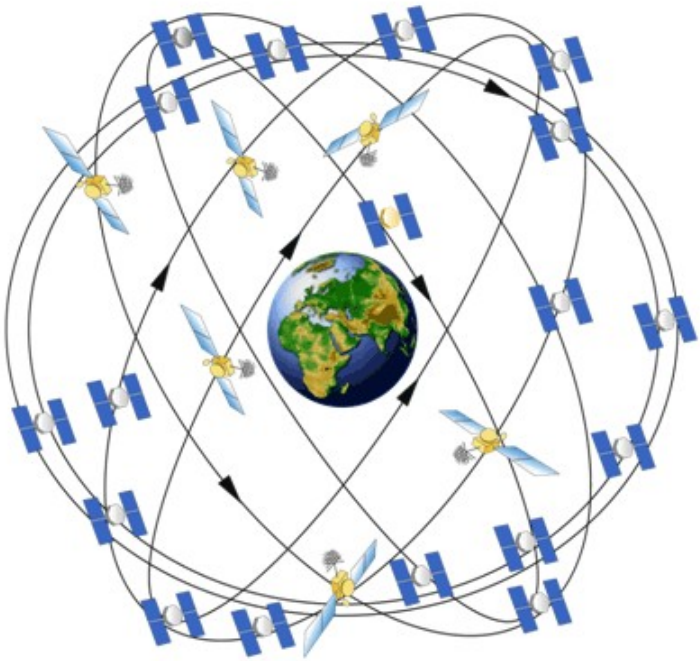


Figure 3: GPS 24-slot Constellation [2]

The basic GPS measurement is the range, which is the distance from the satellite to the antenna. In this case the math error correction is critical in making the position data accurate.

Differential GPS was to meet the needs of positioning and distance measuring applications that requires much higher accuracies than standard stand-alone GPS. In standard DGPS one or more user receivers are combined with one or more refernce receivers located at known locations. This system architecture provide considerable error correction.

4.2. GPS errors sources

The major sources of error affecting stand-alone GPS are:

- ✔ Emphemeris Error

  Transmitted data to determine the current position of setellites are based on the

current position of Earth, Sun, Moon and other planets. Incorrect or dellayed detection of their positions and influence of other celestial bodies and space debris results in error, called Emphemeris Error.

- ✔ Ionospheric and Tropospheric Propagation Delay

  Radio signals passing through the ionosphere and the troposphere, particularly at shallow angles, are refracted and slowed down during propagation.

- ✔ Satellite Clock Drift

  Satellite dilution of position precision causes Satellite Clock Drift effect.

- ✔ Multipath

  Ranging error is caused by radio signals reaching the receiver after reflection from Earth's surface.

- ✔ Receiver Noise

  Manufacturing limitations and tolerances accepted for the receivers are known like receiver noise.

Table 1 summarises the above error sources and their possible improvements by DGPS in estimation of magnitude.

Table 1: *Comparison of error sources in stand-alone GPS and DGPS [4]*

Error Source	Stand Alone (m)	DGPS (m)
Empheresis	5 - 20	0 - 1
Ionosphere	15 - 20	2 - 3
Troposphere	3 – 4	1
Satellite Clock	3	0
Multipath	2	2
Receiver Noise	2	2

As shown in above error comparison, DGPS significantly improves GPS measurement precision. Additionally multipath and receiver noise errors cannot be corrected by general version of this method.

If we assume that user receiver and reference receiver are stationed in relative close distance, the math errors, shown in above table, are significantly decreasing. In most cases they are so small that almost negligible.

4.3. Modern low power embedded platforms

If we look at the proposed solution we can see two types of nodes presented differ level of complexity of the tasks required to be performmed on each of them. TS node is the simplest element of system. The main work it should do is to grab data from GPS module using standard serial communication and send them wirelessly to the base station. To save energy consumption device have to persist in kind of stand-by mode as long as it is possible. Idea is to design interrup based awaking system for client station, which will pull device in operation mode when base station demand this. Based on this knowledge, we can safely assume that 8-bit microcontroller will be able to confidently meet this requirement, on condition that it has support for energy saving, at least 2KB of RAM for the collection of samples and two UART communication interfaces to handle communication with GPS module and wireless comunication module.

We chose PIC18F25K80 microcontroller produced by Microchip. This devices are high-computational units at an economical price offering a range of features that can significantly reduce power consumption during operation, like possibility to run microcontroller with its CPU core disabled but peripherals still active. This feature we can use to achive pernament stand by mode disturbed by measurement performance request sent by BS node. In addition, it has 3.6 KB of RAM and two EUSART interfaces, what was next determinant of good choice. More detailed overview of this device can be found in Appendix 1.

The situation is a little bit different when we consider the requirements associated with the BS node. It should be able to collect data from its own geolocation GPS module, receive the data collected in the TS nodes, process the collected data using an algorithm implemented in order to detect potential threats (which probably will require a significant resource of computing power) and synchronize operation of entire network. And most important requirement, all of these tasks should be handled with the minimum amount of

energy needed.

Although in the initial phase of the project we identified STM32F100 microcontroller as the best choice during the research we got to Norwegian company, EnergyMicro, homepage which proposed a system with very similar parameters while remaining more energy friendly solution. Therefore we changed our target platform to EFM32G890F128 microcontroler as a main unit of BS node.

This microcontroller is also based on ARM Cortex-M3 32-bit core that could operate at a frequency of up to 32 Mhz. It is equipped with 16 KB RAM, many communication interfaces and what is the most important has flexible energy management system with ability to operate with current consumption of 0,9 uA in Deep Sleep mode, 45uA in Sleep Mode and 180uA/MHz in Run Mode what is the best achievement in the nowadays market of microcontrollers. More detailed overview of this device can be found in Appendix 2.

In addition, in order to allow the use of an differential algorithm to detect the possible risks associated with changing the initial position of nodes relative to each other, GPS data should be downloaded synchronously on both devices, which makes a need for both types of nodes to have its own GPS receiver.

4.4. Wireless communication for embedded systems

When choosing a wireless communication technology we aim to find a solution that would fully support all low-level tasks related to communication, from establishing a connection, through packet transmission till the ability to work in point-to-multipoint topology. These aspects of communication stacks go far beyond the scope of our project, so we wanted to avoid as many implementation of mechanisms associated with these tasks as possible. So on the one hand it should be a mean of communication based on a multifunctional wireless modules but on the other hand, it is still limited by the requirement of energy efficiency. Our research has shown that the best solution, which merit consideration related to testing out the possibilities of its use in our project, will be usage of XBee-PRO wireless communication modules distributed by Digi company.

These modules use the IEEE 802.15.4 ZigBee networking protocol for fast point-to-

multipoint or peer-to-peer networking. They are designed for high-throughput applications requiring low latency and predictable communication timing. [7] By use of these modules it becomes possible to transfer data at distance of up to 1,6 km with 250 Kbps data rate. This solution handle with task like communication errors by using retries and acknowledgements, nodes adressing and point-to-multipoint network management.

These modules need a lot of current to transmit (215 mA) and receive (55mA) data, but we can configure each of them the way they will be in sleep mode in most of time to not waste energy. In sleep mode XBee-PRO module consumes less than 10uA of current and wake it up only when there is some data to send. In view of the available data rate we can safely assume that these devices will send data fast enough that we do not have to worry about the maximum current consumption.

More detailed information about this solution can be found in Appendix 5.

5. Tools

Every engineer knows that his work productivity is strongly dependent on the availability of tools he can use. Times where the designer had to start every project entirely from scratch are long gone and today a lot of tedious and repetitive tasks can be successfully handled automatically by the tools we use. Integrated environments are able to generate our skeletons of programs, make the necessary configuration using a simple wizard, step by step, asking us about preferences for a project or to generate documentation based on properly prepared comments in the code. For this reason, the time spent on analysis and selection of tools should not be treated as lost as the appropriate selection can effectively improve the performance of work at each stage of the project, from design through implementation, ending on tests.

There are many factors influencing the assessment of the suitability of the particular tool. First we should check whether the product can be used to create the software which will eventually run on a chosen platform. It is also advisable to check the ease of use and level of support for the developer during the implementation process.

However, it was one more thing important for us. In the course of the project we wanted to know and learn how to use the tools that we can successfully use in the future for our own commercial projects without having to purchase the appropriate licenses. For this reason, we tried to find as many tools that are distributed on OpenSource licenses as it was possible.

5.1. Programming language

Choice of programming language used in the project is closely linked with the choice of target platform of our project and availability of proper compilers for that platform. When talking about hardware part of project we know that our target devices are based on PIC and ARM microcontrollers, what effectively narrowed our possible choices to only two options - the C and Assembler - because only compilers of these languages are free and without any restrictions. Because C is a high level language that guarantees easy of use

and efficiency it seems to be the best possible choice.

On the other hand, thinking about the analysis of large amounts of data, immediately comes to mind MATLAB, so at the beginning it was the primary IDE and language for software part of project. Unfortunately, after a short time it turned out that MATLAB is not suitable for processing many strings - it was to slow to use it. So the next step was to find a better programming language. And so, C + + was chosen - because of its speed, and the availability of free cross-platform IDEs.

5.2. Integrated development environment

Considering the part of the project related to the design of embedded system, we concluded that it is not possible to obtain a single development environment for both the selected target platform. For this reason, the implementation process of firmware for the device with the PIC microcontroller on the board we used a program called MPLAB X developed by the manufacturer of this chip, Microchip company. It is a fully functional IDE that supports the process project preparation for a particular version of the system, syntax highlighting, very well navigation between files of the project and the great support for the process of debugging software running on the target device.

For developing software for the platform EFM32 we decided to prepare our own IDE based on the free items. The role of an integrated environment serves the well-known and free software called Eclipse CDT. Toolchain needed to compile and link our implementations is SourceryG++ which is free to use in Lite version without any limitations. Last thing is software supported on-chip debugging process. To this task we engaged open source software called OpenOCD. In this way, we obtained fully functional and free platform for creating software designed to work with ARM microcontrollers.

For data analysis Qt Creator was chosen as IDE. Because it is free of charges, cross-platform and has integrated GUI editor. Additionally, it is used by many developers and companies from the IT industry, which provides us the the safety of regular updates.

5.3. Source code management

It is recommended to use version control system during the work on a project including code writing. According to wikipedia.org:

"Revision control, also known as version control or source control (and an aspect of software configuration management or SCM), is the management of changes to documents, programs, and other information stored as computer files. It is most commonly used in software development, where a team of people may change the same files. Changes are usually identified by a number or letter code, termed the 'revision number', 'revision level', or simply 'revision'. For example, an initial set of files is 'revision 1'. When the first change is made, the resulting set is 'revision 2', and so on. Each revision is associated with a timestamp and the person making the change. Revisions can be compared, restored, and with some types of files, merged."

Despite the fact that the division of labor has meant that each of us worked on a separate area code, ability to restore the state of code before the change, which proved to be only worsen the work of the whole system, with a few simple steps has been very helpful. By installing VisualSVN software on it and TortoiseSVN software on our private computers we got fully functional version control system. VisualSVN acts as a version control server management software and TortoiseSVN is its client.

5.4. PCB design software

In the initial phase of the project to create schemas and PCBs we used software called CadSoft Eagle. It has a free license for some usage, but this version has several limitations. In the meantime we have discovered a new, completely free tool called KiCad, which once again showed that free tools can successfully compete with commercial solutions. Designing a system using KiCad found to be very simple and intuitive process. And one more time we have gained expertise in the use of software, which can be used commercially in our future projects.

5.5. Embedded development boards

In order to carry out tests of various possible solutions that can identify problems that can not be solved by using a specific platform or to show which requirements should be modified in order to achieve the best possible results developers often use the development board in the early phase of the prototype device design. So it was in this case.

To work with TS node prototype we used EVBeasyPIC development board showed in the Figure 4. This board provides a variety of peripherals and gives access to every PIN of MCU connected to this board. In order to be able to debug applications running on microcontroller we used ICD2 debugger clone provided by the same company that we had EVBeasyPIC board from. This debugger is shown in the Figure 5.

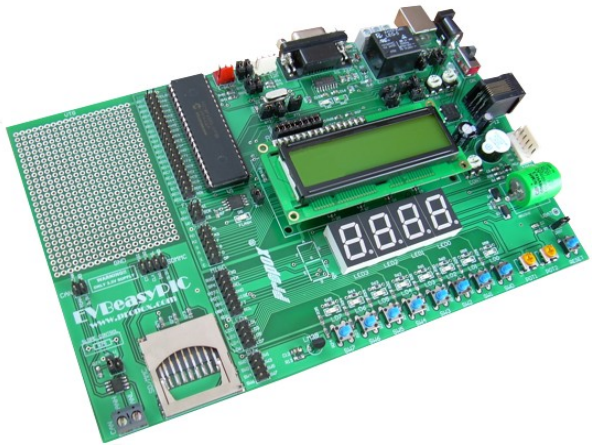


Figure 4: EVBeasyPIC Development Board [5]

Creating a prototype of BS node was performed based on a two kind of development boards provided by a company Micro Energy called EFM32 Gecko Starter Kit and EFM32 Gecko Development Kit (Figure 7). First board was used as a part of the final solution due to lack of capacity to perform PCB needed to service EFM32G890 microcontroller in packages that was offered by Energy Micro company. Second board was used on prototyping stage of project. Both boards have built-in on-chip debugger interface and

support for tracking power consumption when combined with software from EnergyMicro.

Figure 5: ICD2 clone debugger [6]

Figure 6: EFM32 Gecko Development Kit [7]

6. Analysis of GPS system capabilities

One of the elements of the project was to develop software for BS, which would allow us to detect even the slightest movement of one of the TSs. Unfortunately, in the middle of the project it turned out that used by us modules do not allow to achieve the required accuracy with low power consumption. But it was too late to change the approach and start from scratch, so we decided to continue research in order to get as much as possible from it and help future researchers. In view of the fact that it was not yet late phase of the project, we resigned on implementation for the BS and instead, also due to lack of memory limitations (as is the case with flash memory on the device), we wrote the code for the PC to be able to gather and analyze more data. The process is shown below.

6.1. Process of gathering data

Data were collected by simple device shown at Figure 7. It consist of GPS module with antenna and RS232 to USB converter.

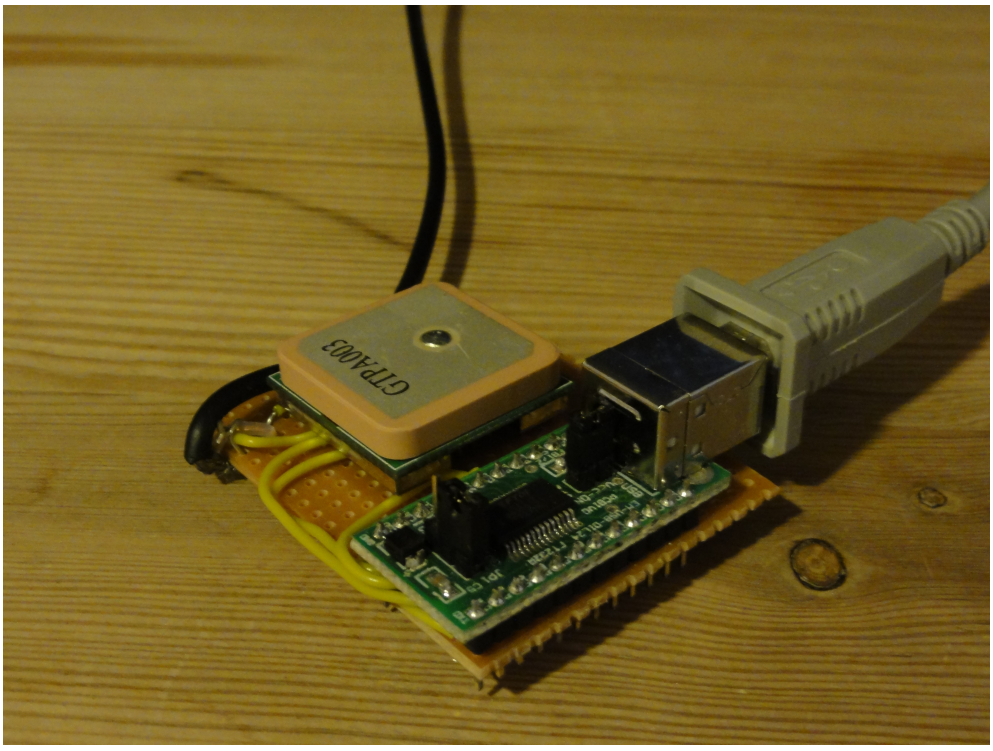


Figure 7: Device used to communicate with GPS module

It was connected to PC by USB cable and communication was obtained by use of terminal aplication RealTerm. Beside of configuring GPS module it allowed us capturing GPS data from module, which is shown at Figure 8.

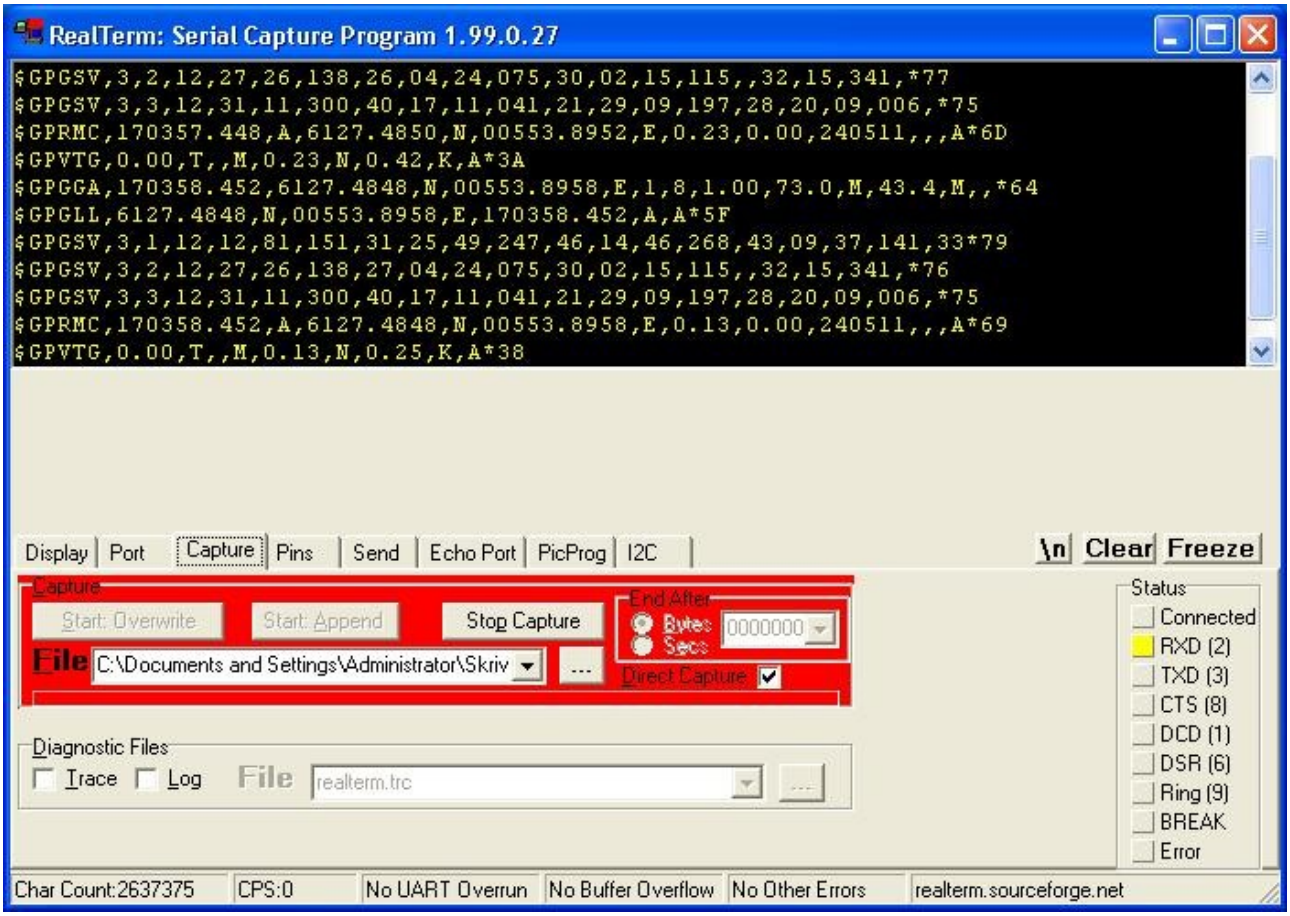


Figure 8: RealTerm during capturing data

Data were being collected at the various distances between receivers. At small distances, errors were too large to discuss them (even several hundred percent). Distances around 200 meters were promising and were within the real scope of the system, so only they will be presented in the following chapters.

6.2. Data processing

Data gathered from GPS module are in the form of NMEA 0183 sentences. The use of the available commands for our GPS module is shown in Table 2.

Table 2: Selected NMEA 0183 sentences

NMEA 0183 command	Description	Data fields
GLL	Geographic Position, Latitude, Longitude and time	latitude, longitude, status, CRC
RMC	Recommended minimum specific GPS/Transit data	time of fix, status, latitude, longitude, speed in knots, track made good in degrees True, UT date, magnetic variation degrees, CRC
VTG	Track Made Good and Ground Speed	Track made good, fixed text 'T', empty field, empty field, speed over ground in knots, speed over ground in kilometers/hour, CRC
GGA	Global Positioning System Fix Data	time, latitude, longitude, quality of fix, number of satellites in view, HDOP, altitude, height of geoid above WGS84 ellipsoid, empty field, empty field, CRC
GSA	GPS DOP and active satellites	mode (M - manual, A - automatic), fix type (1 - fix not available, 2 - 2D, 3 - 3D), IDs of satellites in position fix (up to 12), PDOP, HDOP, VDOP
GSV	GPS Satellites in view	total number of messages of this type in this cycle, message number, total number of satellites in view, setellite data (statellite id number, elevation in degrees, azimuth, SNR; repeated up to 4 times), CRC

Example of gathered sequence of sentences is shown at Figure 9.

```
$GPGGA,193925.000,6127.4772,N,00553.8896,E,1,9,1.10,97.9,M,43.4,M,,*6C
$GPGLL,6127.4772,N,00553.8896,E,193925.000,A,A*54
$GPGSA,A,3,27,18,15,09,25,17,12,22,14,,,,1.95,1.10,1.61*0F
$GPGSV,3,1,12,09,80,171,36,27,66,132,43,12,48,221,43,17,38,063,38*73
$GPGSV,3,2,12,14,28,311,41,15,25,179,25,22,21,280,35,24,12,006,*75
$GPGSV,3,3,12,18,11,250,36,25,09,235,45,28,03,060,,11,03,001,*77
$GPRMC,193925.000,A,6127.4772,N,00553.8896,E,0.00,155.83,250311,,,A*6D
$GPVTG,155.83,T,,M,0.00,N,0.00,K,A*37
```

Figure 9: Example of NMEA 0183 sentences

Simultaneously to the collection of data, *NMEAparser* class has been written - it allows to parse these data. All types and functions described in this chapter are part of this class.

6.2.1. Data storing

For storing data there was defined type *NMEAdataList*:

```
typedef QList<NMEAdata> NMEAdataList;,
```

which is *QList* (type of list provided by Qt environment) of *NMEAdata* structures:

```
struct NMEAdata
{
    unsigned char sentences;
    GPGGA gpgga;
    GPGSA gpgsa;
    GPGLL gpgll;
    GPRMC gprmc;
    GPVTG gpvtg;
};
```

This structure consists of structures for every used NMEA 0183 sententes and field *sentences* – status bit field of already filled sentences fields. Structures for sentences are shown below:

```
struct GPGGA
{
    unsigned int satellitesNum;
    double fixTime; // in seconds
    unsigned int fixQuality;
    double latitude;
    double longitude;
    double altitude;
    double hdop; // horizontal ditution of precision
    double hog; // height of geoid (mean sea level) above WGS84 ellipsoid
};
struct GPGSA
{
    bool mode0; // M - Manual, forced to operate in 2D or 3D,
                // A - Automatic, 3D/2D
    bool mode1; // 1 - Fix not available, 2 - 2D, 3 - 3D
    unsigned char satelllitesIDs[14]; // IDs of satellites used in fix
    double pdop; // PDOP
    double hdop; // HDOP
    double vdop; // VDOP
};
struct GPGLL
```

```
{
    double latitude;
    double longitude;
    double time; // in seconds
    bool valid;
};
struct GPRMC
{
    double fixTime; // in seconds
    bool status;
    double latitude;
    double longitude;
    double speed;
    double trackAngle;
    double date;
};
struct GPVTG
{
    double ttmg; // true track made good
    double gSpeedN; // ground speed in knots
    double gSpeedK; // ground speed in kilometers/hour
};
```

For paired data (described in 6.2.3) type NMEAdataList was defined:

```
struct NMEAdataPairList
{
    NMEAdataList data0;
    NMEAdataList data1;
};
```

And for storing parsed data structure *NMEAresultData was created*:

```
struct NMEAresultData
{
    double date;
    double time; // in seconds
    double latitude0;
    double longitude0;
    double latitude1;
    double longitude1;
    double distance; // in m
    unsigned int satellitesNum0;
    unsigned int satellitesNum1;
    double hdop0;
    double hdop1;
};
```

6.2.2. Data parsing

Process of loading and parsing data is done by function *loadData*:

```
/*
 * Loads NMEA sentences data from file and returns them in created list.
 *
 * @param input file name, bit field - which sentences should be parsed,
 *        minimum number of satelltes
 * @return list of parsed NMEA sentences
 *
 */
NMEAdataList loadData(QString,unsigned char,unsigned char);
```

This function, depending on the analyzed sentence name, calls one of the following methods:

```
/*
 * Parses GGA sentence
 *
 * @param GGA sentence
 * @return structure filled witch sentence data
 *
 */
GPGGA parseGPGGA(QString);
/*
 * Parses GSA sentence
 *
 * @param GSA sentence
 * @return structure filled witch sentence data
 *
 */
GPGGA parseGPGSA(QString);
/*
 * Parses GLL sentence
 *
 * @param GLL sentence
 * @return structure filled witch sentence data
 *
 */
GPGLL parseGPGLL(QString);
/*
 * Parses RMC sentence
 *
 * @param RMC sentence
 * @return structure filled witch sentence data
 *
 */
GPRMC parseGPRMC(QString);
/*
 * Parses VTG sentence
 *
```

```
 * @param VTG sentence
 * @return structure filled witch sentence data
 *
 */
```

6.2.3. Creation of data pairs

For distance computation there is need of paired data. Function *makeDataPairs* checks two source lists and returns one of data related by having the same time stamps.

```
/*
 * Creates pairs of data.
 *
 * @param two source lists of NMEA sentences
 * @return list of paired NMEA sentences
 *
 */
NMEAdataPairList makeDataPairs(NMEAdataList,NMEAdataList);
```

6.2.4. Distance computation

When we have pairs of data finally we can compute distance between two points – they are corresponding elements of *NMEAdataPairList*. Function that give us that opportunity to do so is name *computeDistance*:

```
/*
 * Computes distance between two points for all list of pairs.
 *
 * @param two source lists of NMEA sentences
 * @return list of paired NMEA sentences
 *
 */
NMEAresultDataList computeDistance(NMEAdataPairList);
```

It uses for distance computing function called *distanceVincentysInverse:*

```
/*
 * Computes distance between two points.
 * Implements Vincentys inverse algorithm.
 *
 * @param latitude of first point, longitude of first point,
 *        latitude of second point, longitude of second point,
 * @return distance in meters
 *
 */
double distanceVincentysInverse(double,double,double,double);
```

This function implements Vincentys inverse algorithm described in *[9]*.

6.4. Tests

During the project has been collected 4 726 948 of sequences, what considering one sequence per second gives us approximately 1300 hours of data. In more than 90% cases there was from 7 to 10 satellites in view, which is shown at Figure 10 and in Table 3.

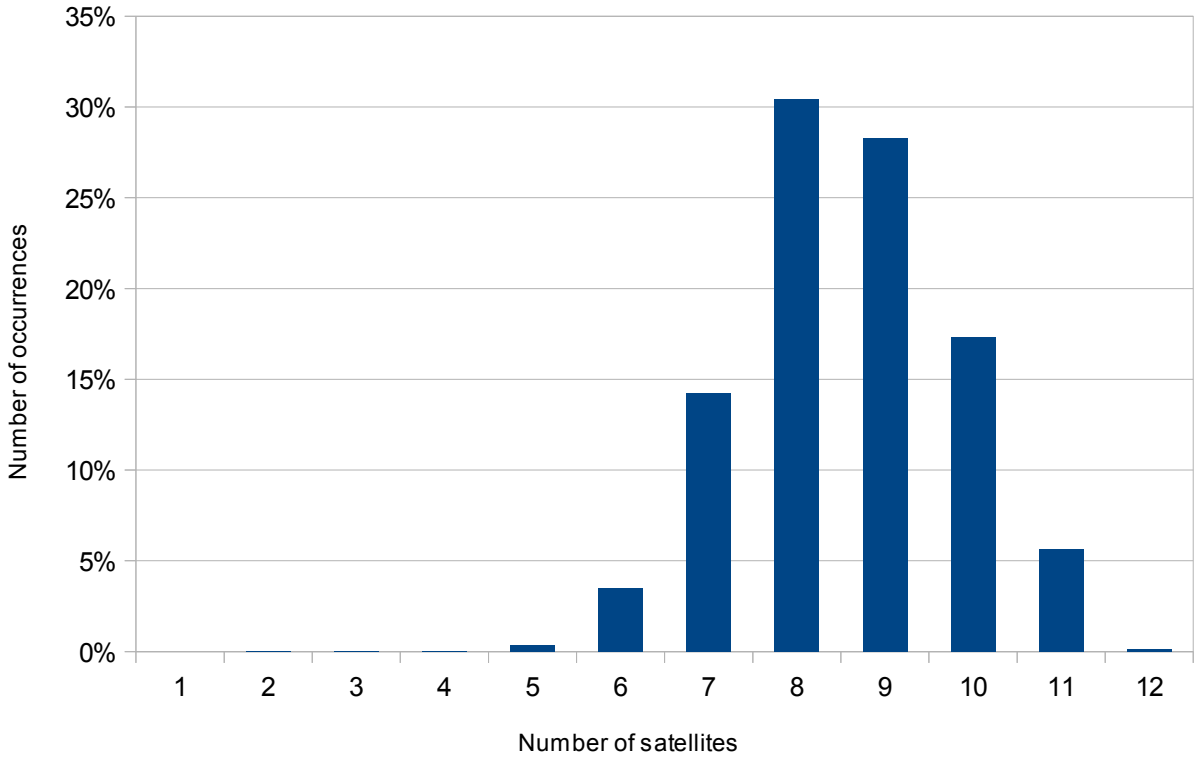


Figure 10: Number of satellites in view

Table 3: Number of satellites in view

Number of satellites	Number of occurences [%]
1	0,00000
2	0,00002
3	0,00186
4	0,02767
5	0,35408
6	3,49661
7	14,25474
8	30,44248
9	28,26100
10	17,33469
11	5,66410
12	0,16275

6.4.1 Distance A

Distance A means approximately 200 meters. Approximately because of lack of proper measurements. Results from 16.04.2011 are shown at Figure 11 to Figure 14.

Sample pairs count: 83522

	Latitude 1	Longitude 1	Latitude 2	Longitude 2	Distance [m]
Min:	61,45680	5,90151	61,45760	5,89793	177,63700
Max:	61,45810	5,90258	61,45830	5,89850	242,31300
Mean:	61,45729	5,90182	61,45788	5,89823	202,05576
Standard deviation:	0,00006	0,00008	0,00005	0,00007	5,02875

Figure 11: Tests summary for distance A from 16.04.2011

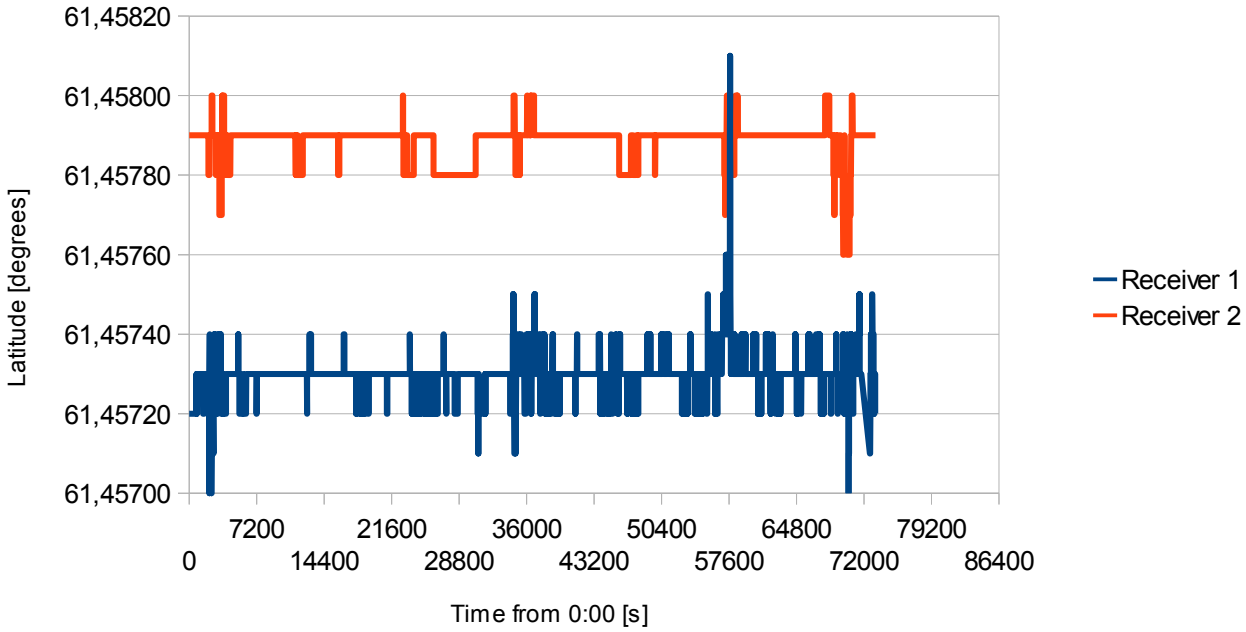


Figure 12: Variations in latitude for distance A from 16.04.2011

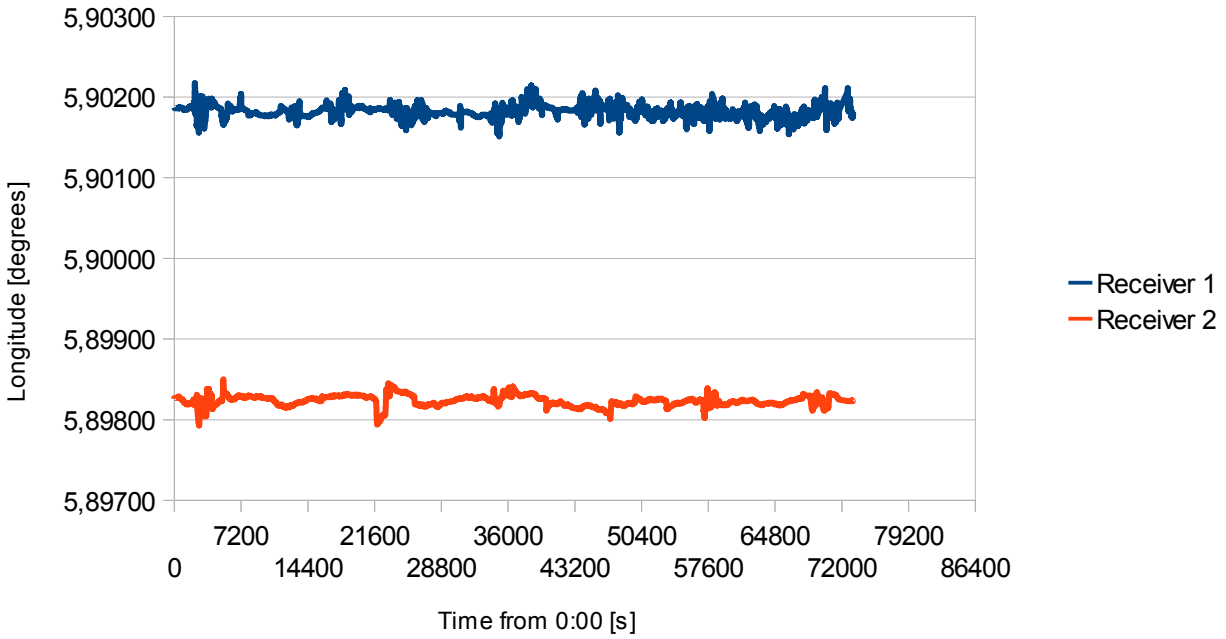


Figure 13: Variations in longitude for distance A from 16.04.2011

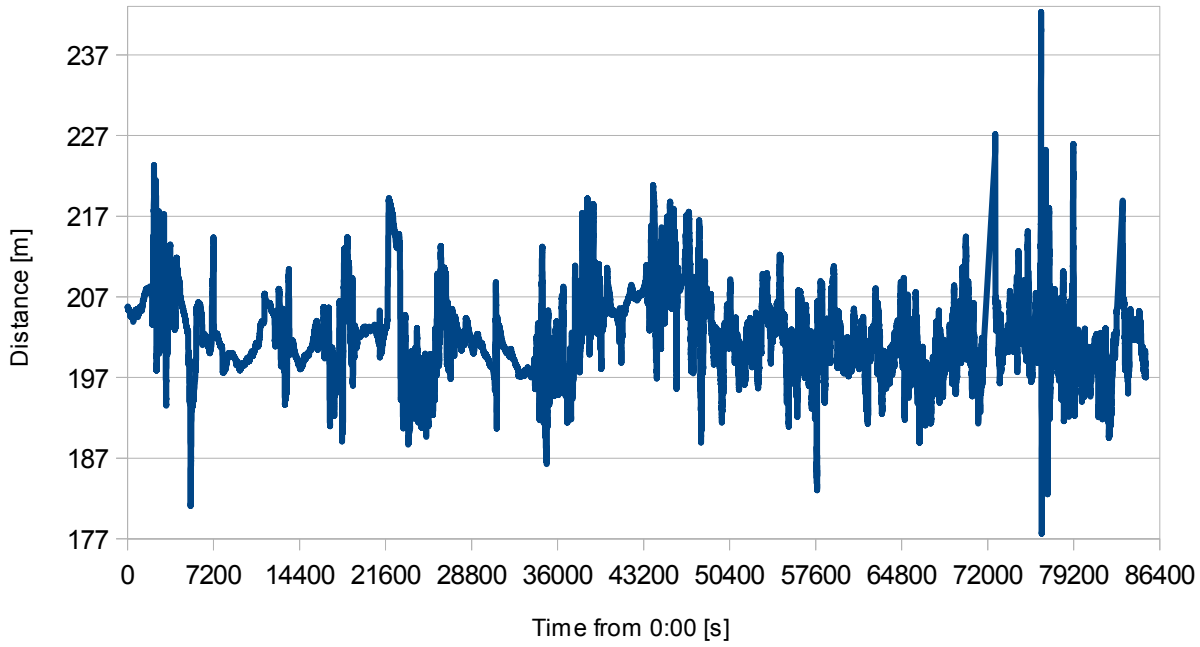


Figure 14: Variations in computed distance for distance A from 16.04.2011

6.4.2 Distance B

Distance B means 60 cm more than distance A. Results from 28.04.2011 are shown at Figure 15 to Figure 18.

Sample pairs count: 80327

	Latitude 1	Longitude 1	Latitude 2	Longitude 2	Distance [m]
Min:	61,45710	5,90166	61,45790	5,89795	188,79200
Max:	61,45770	5,90194	61,45820	5,89845	221,24600
Mean:	61,45730	5,90183	61,45804	5,89827	207,99218
Standard deviation:	0,00004	0,00004	0,00005	0,00005	3,20598

Figure 15: Tests summary for distance B from 28.04.2011

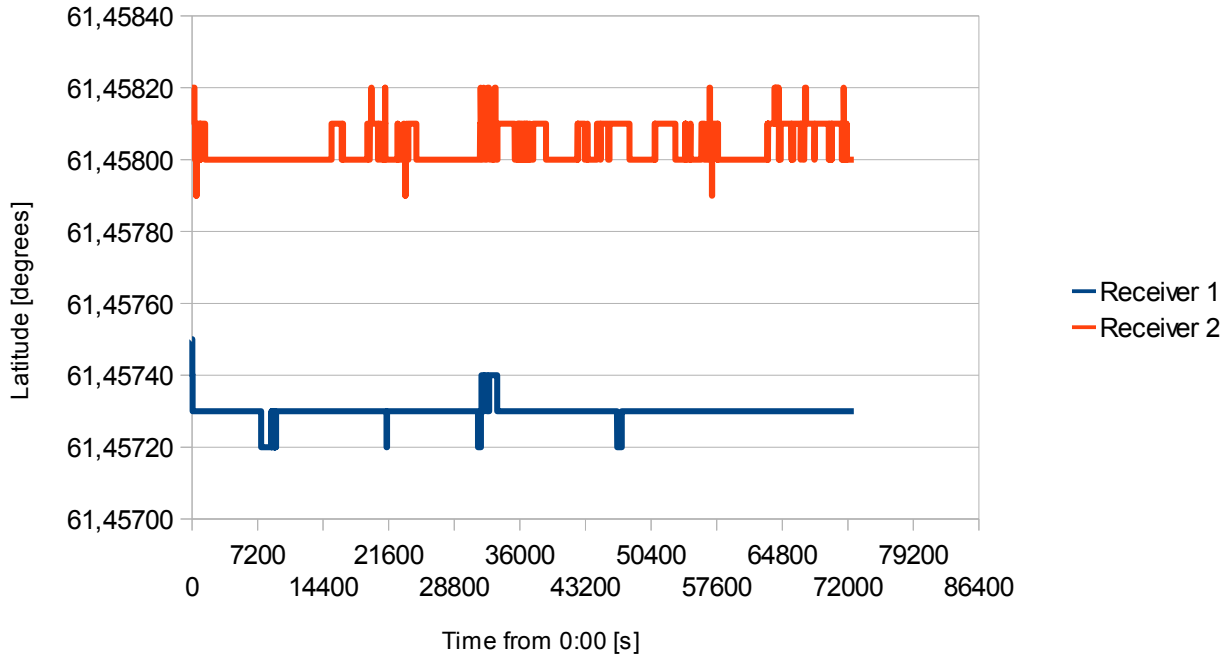


Figure 16: Variations in latitude for distance B from 28.04.2011

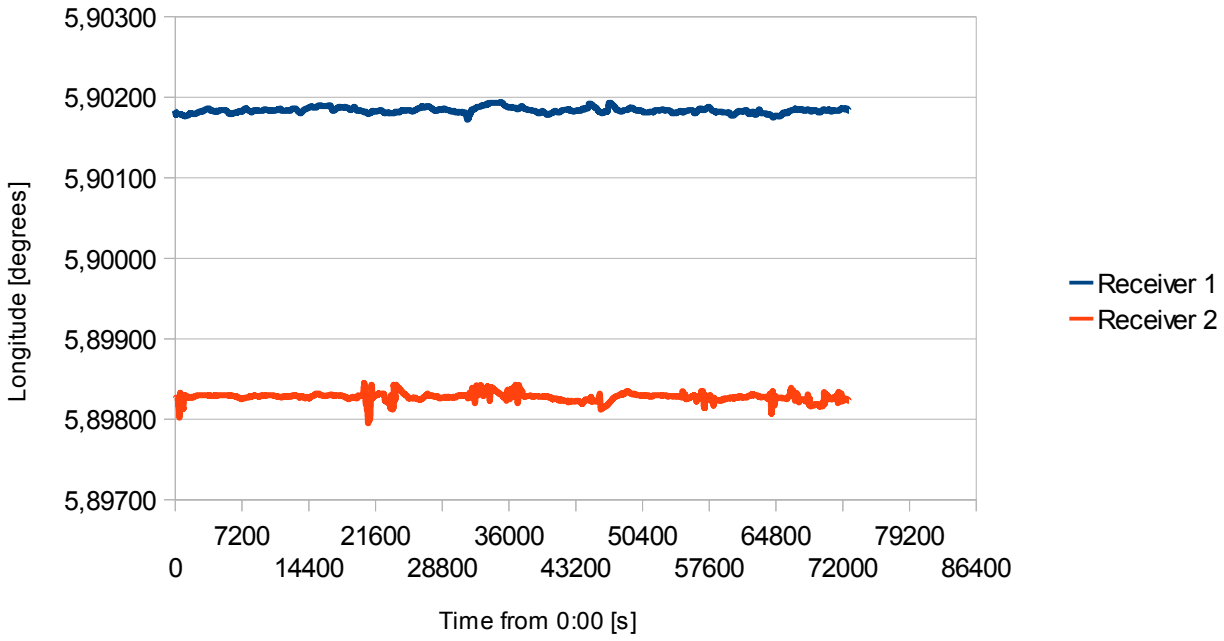


Figure 17: Variations in longitude for distance B from 28.04.2011

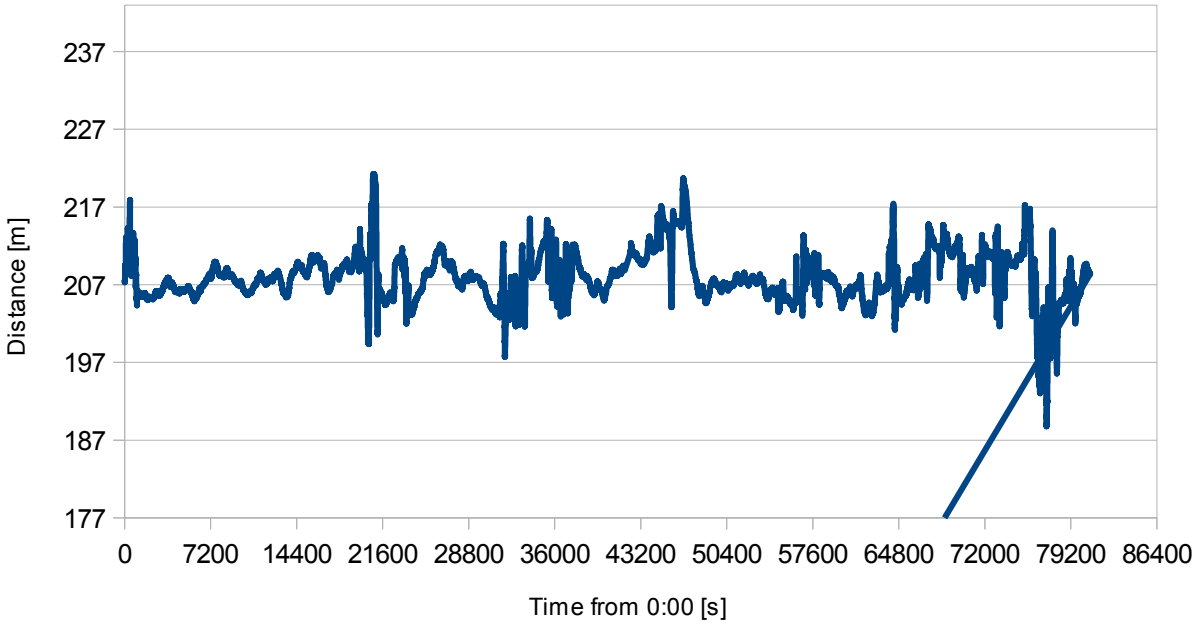


Figure 18: Variations in computed distance for distance B from 28.04.2011

Results from 30.04.2011 are shown at Figure 19 to Figure 22.

Sample pairs count: 77805

	Latitude 1	Longitude 1	Latitude 2	Longitude 2	Distance [m]
Min:	61,45710	5,90164	61,45790	5,89802	174,21900
Max:	61,45760	5,90199	61,45830	5,89903	218,26200
Mean:	61,45729	5,90184	61,45804	5,89829	207,84716
Standard deviation:	0,00004	0,00005	0,00005	0,00004	2,95583

Figure 19: Tests summary for distance B from 30.04.2011

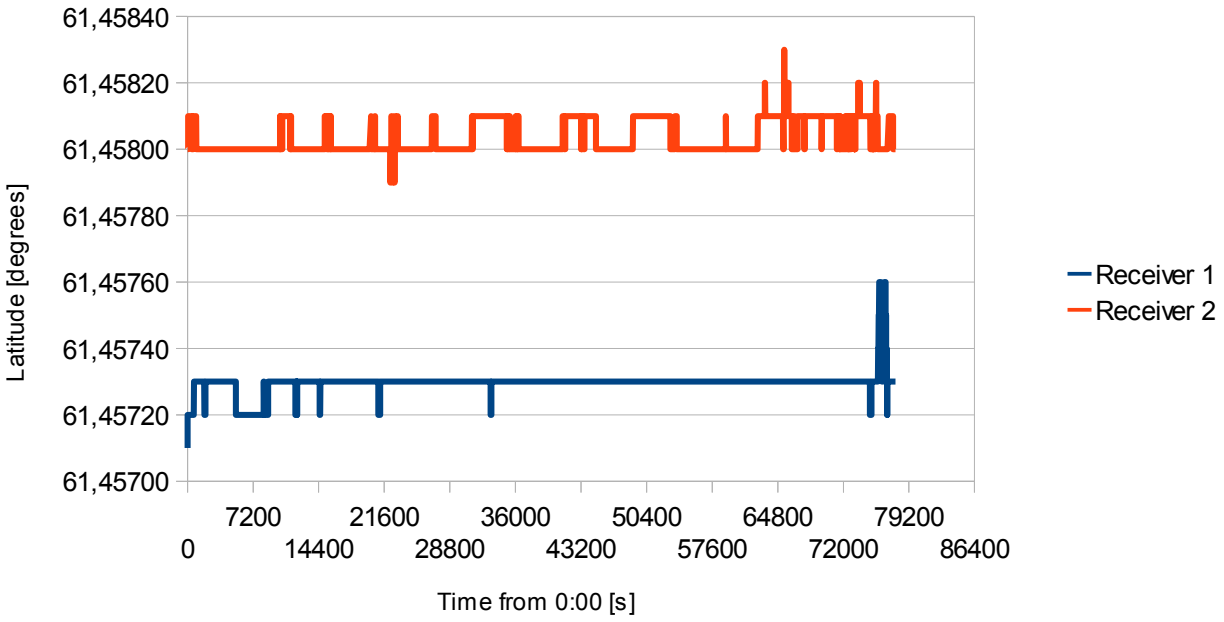


Figure 20: Variations in latitude for distance B from 30.04.2011

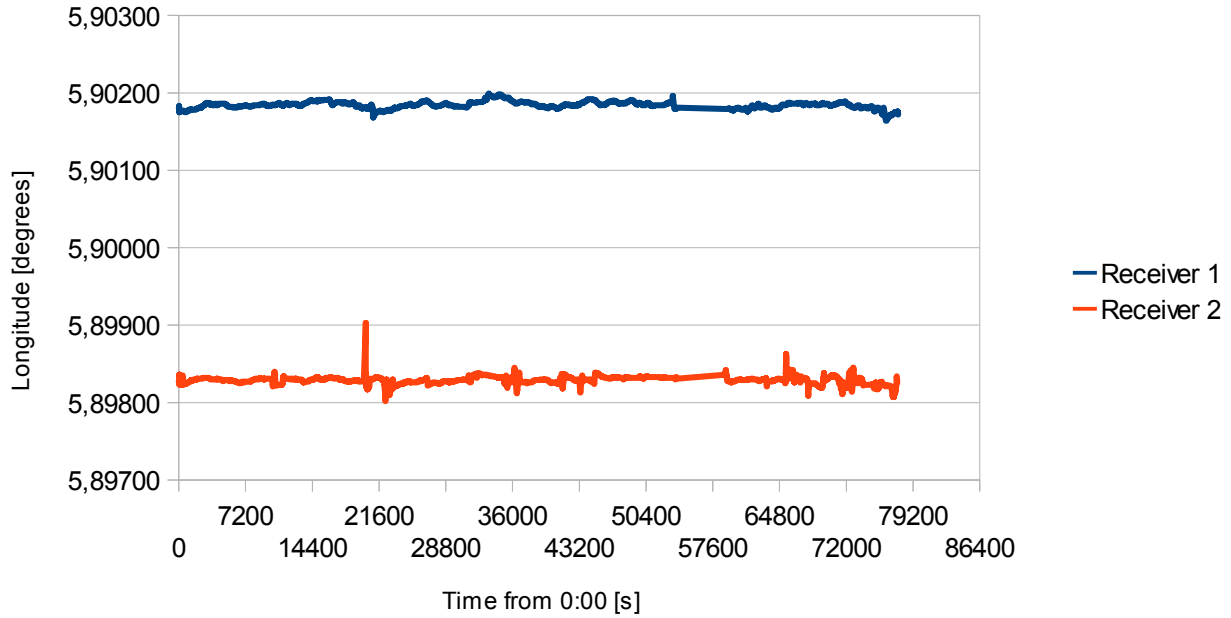


Figure 21: Variations in longitude for distance B from 30.04.2011

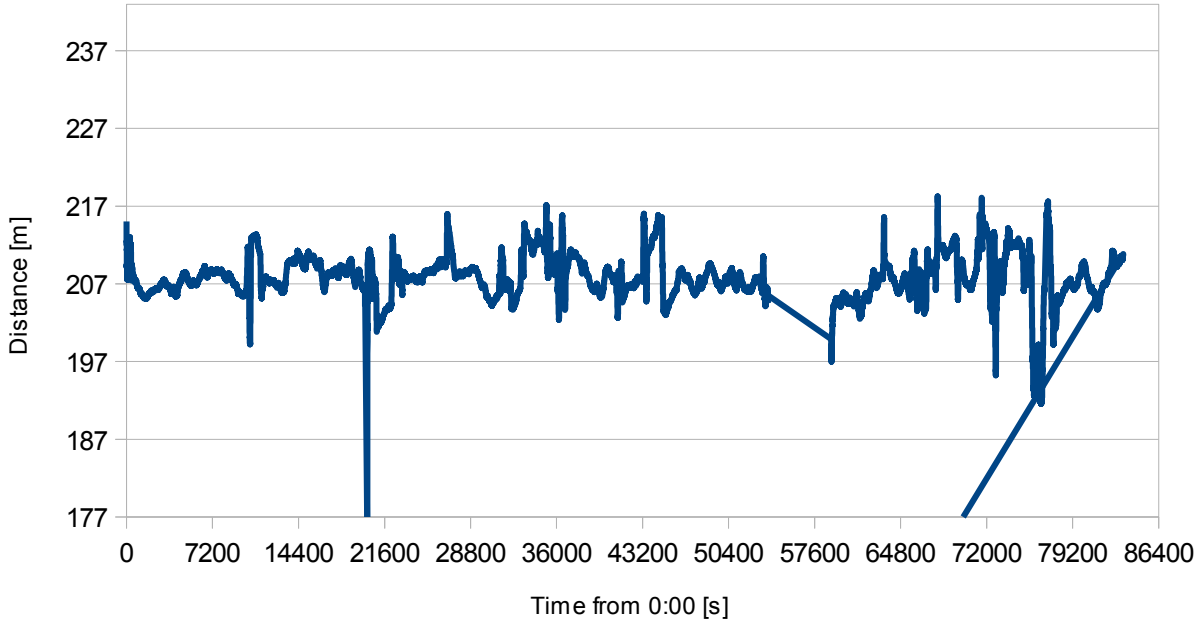


Figure 22: Variations in computed distance for distance B from 30.04.2011

6.4.2 Distance C

Distance C means 1 m more than distance B. Results from 07.05.2011 are shown at Figure 23 to Figure 26.

Sample pairs count: 72171

	Latitude 1	Longitude 1	Latitude 2	Longitude 2	Distance [m]
Min:	61,45720	5,90156	61,45760	5,89788	192,73900
Max:	61,45750	5,90197	61,45830	5,89851	230,81900
Mean:	61,45729	5,90183	61,45804	5,89827	208,20787
Standard deviation:	0,00004	0,00005	0,00006	0,00005	2,93198

Figure 23: Tests summary for distance C from 07.05.2011

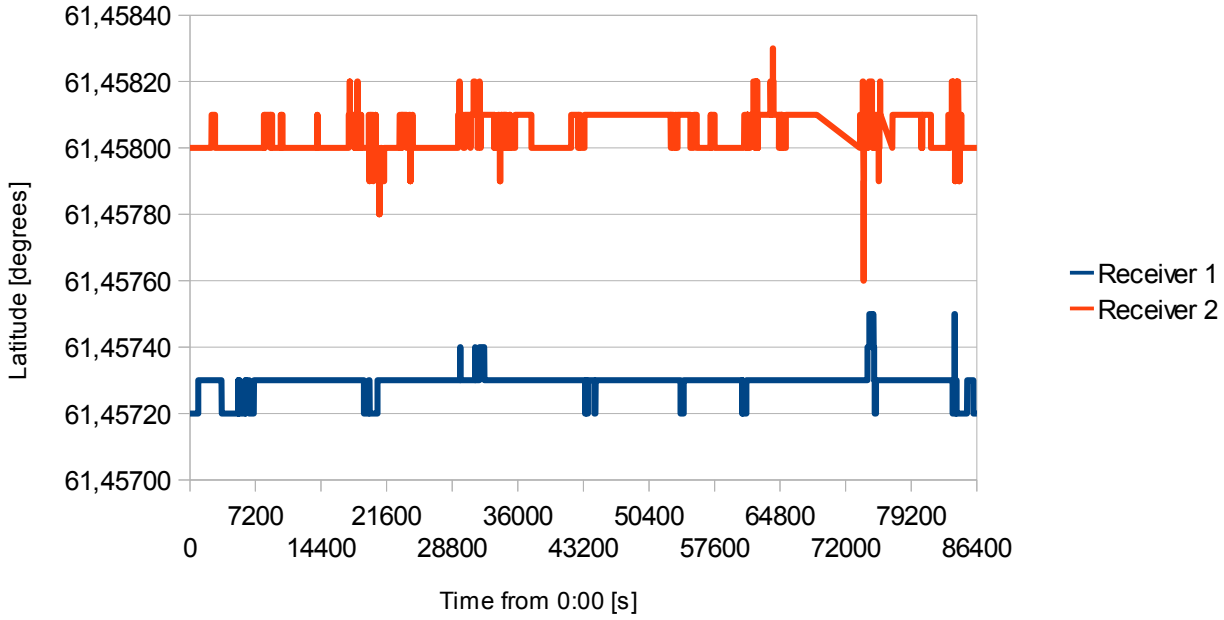


Figure 24: Variations in latitude for distance C from 07.05.2011

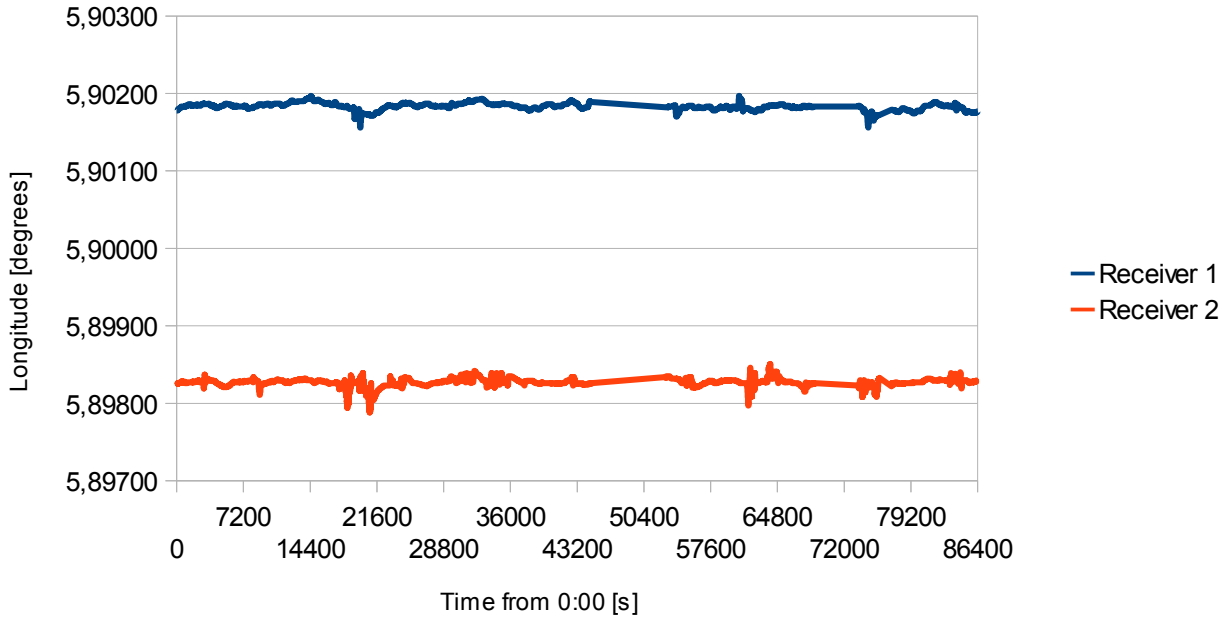


Figure 25: Variations in longitude for distance C from 07.05.2011

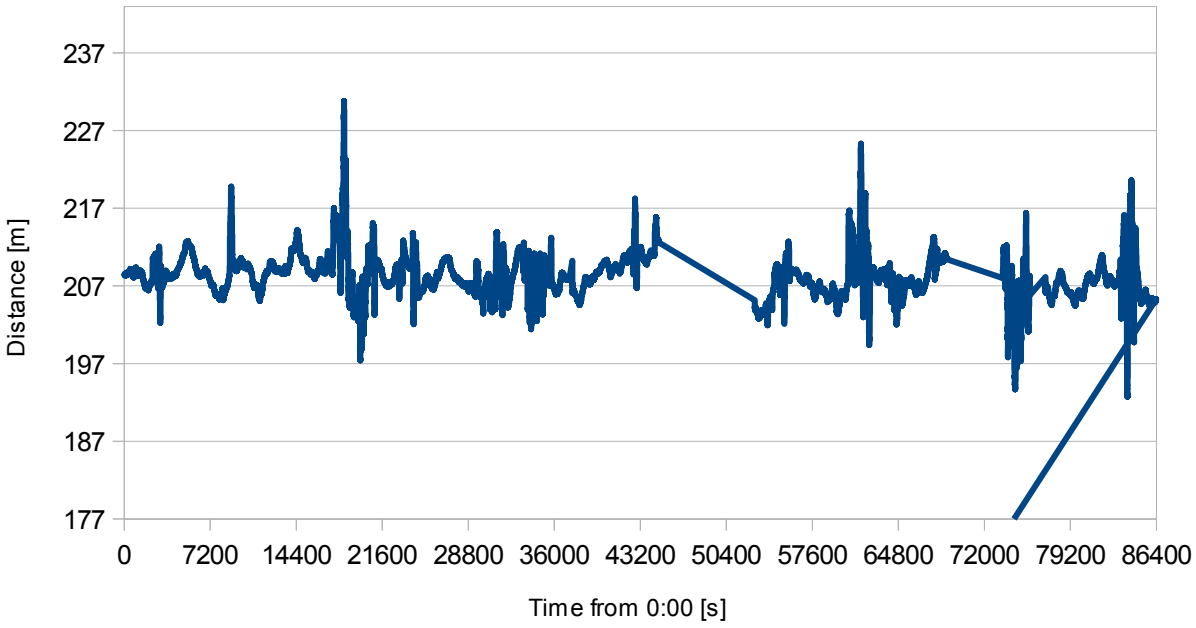


Figure 26: Variations in computed distance for distance C from 07.05.2011

6.5. Conclusions

Our main goal was to detect the slightest variation in the distance, considering as much as possible power consumption. For this part of project, this means that our devices could be turn on only for a short while during a day.

With this in mind, visible at Figures from 11 to 26 variations in longitude and latitude, and resulting of that, in computed distance makes impossible to detect all the differences in distance between devices, not giving too much false positives.

6.6. Further development

For further consideration we see two possible ways.

The first one assumes the use of GPS modules from the same class. In this case there is need to reconsider reconstruction of the whole system in terms of energy consumption, because of need to gathering data not only periodically but as much 24/7/365 as possible.

That conclusion is borne at Figures from 11 to 26, where are visible from higher perspective correlations between longitudes and latitudes of two devices and comparable means for distance B in two different days.

On the other hand the second one assumes the same philosophy of system in terms of power consumption, which makes it necessary to replace of GPS modules (created for tracking, not for distance measurement) with something much more accurate. Here we see possibility of use GPS chips which normally are part of standard GPS modules. This way there is no need for big changes in the whole system, but unfortunatelly there is need to check if this GPS chips are capable of that accurate data gathering, and then creation of the whole software layer - capable of calculation of needed data from gathered raw data.

7. Measurement wireless embedded system

Description of work related to the preparation of the system can be divided into four main parts. Two of them are closely related to the design of equipment that formed the hardware base of each node's device and two more with the implementation of the software controlling the operation of each of these devices. In this section we will try to show every step required to close each part, from conception through design up to implementation of solutions.

Worthy of note is the fact that producing the target PCB we commissioned an external company.

7.1. XBee module breakout board

To be able to manipulate the communications module position in any future case we decided that each of them will be connected to the nodes by means of flexible wires. For this reason it was necessary to design breakout board for XBee module, allowing the statement of such a connection. Schematic of this board is shown in Figure 27 and result PCB in Figure 28.

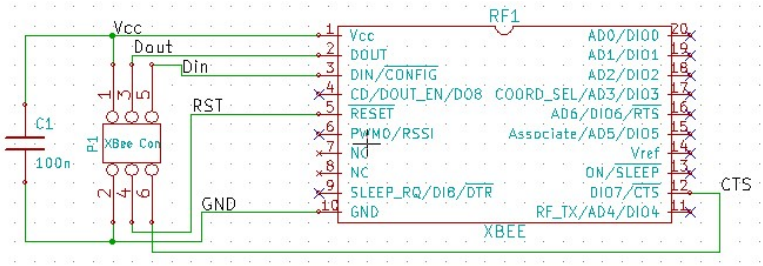


Figure 27: XBee breakout board schematic.

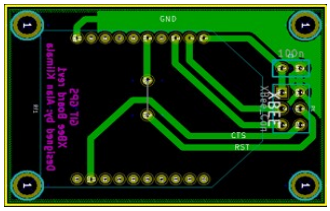


Figure 28: XBee breakout board PCB.

7.2. TS node device design

By analyzing the requirements of the node we can easily identify a modules that creates the target device, concluding that each Terminal Station should be equipped with a communication module, a module to collect GPS data and the microcontroller, which will have the task to supervise the work of the device. With keeping in mind solutions we have proposed in chapter 4 we can easily prepare a general block diagram of the device as showed in figure 29.

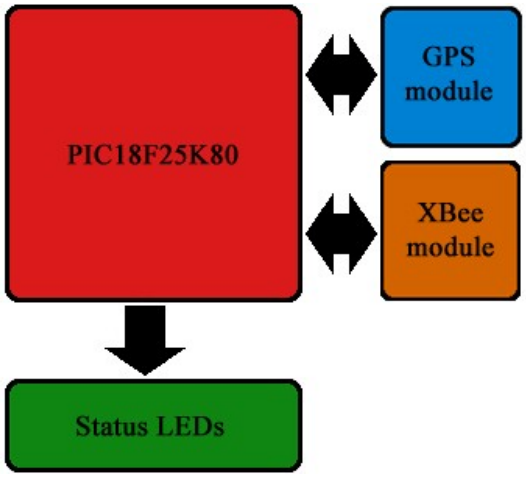


Figure 29: General TS device block diagram

As you can see the main element of a microcontroller, which has a bidirectional connection with a GPS module and a module for wireless communication. In addition, there is a module that contains four LEDs to inform about the status of operation. Both GPS and XBee module communicate with their environment via interface U(S)ART, and therefore they should be connected to the microcontroller pins that are assigned to this type of communication. Fortunately, microcontroller we have chosen has two independent USART interfaces, thus making the connection with each of these three elements was a relatively simple task. In addition the microcontroller is connected to the reset input pins of each module and to CTS pin of XBee module. Status LEDs should be connected to the microcontroller output and lit by setting the value of output to logical 1 (true).

The whole device should be powered by batteries with a nominal voltage of about 3.6 V (these are the most available batteries) but to work with power equal to 3.3 V. The proper and stable voltage should ensure used voltage regulators manufactured by the Microchip company called MCP1801. Additionally the device is normally working with a clock of 8MHz frequency provided by cristal resonator.

Detailed schematic of made connections is presented in Appendix 6.

After preparing the schematic we could proceed to generate the PCB, resulting in board design as set out in Figure 30.

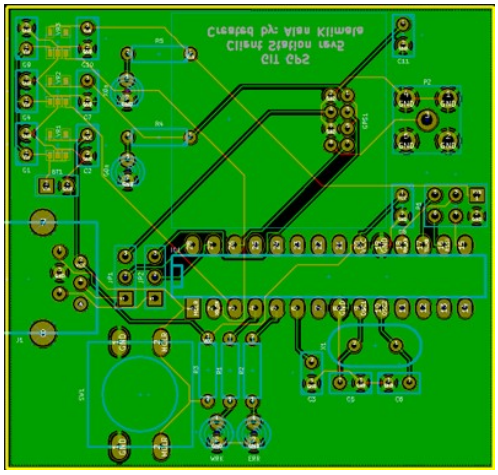


Figure 30: Terminal Station PCB

After receiving the PCB from the factory we proceeded to solder all components and ready result is as in the Figure 31 (GPS module is mounted on the other side and is not visible in the figure).

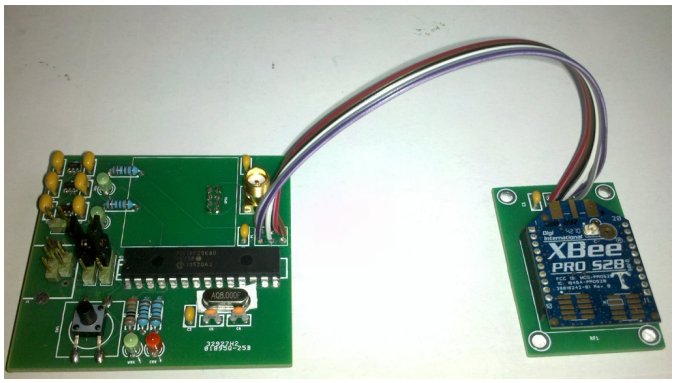


Figure 31: Ready device with communication module connected to it

7.3. TS node firmware implementation

In considering the tasks should be performed by this node, we can conclude that these operations are sequential and repetitive. When we see such relations we should immediately come to mind that this is a clear example of the need to use design pattern called State Machine. When applying software to act as a state machine we need to identify the operating states in which device can exist according to the task it is actually performing and predict the reasons for which there should be a change in its current state made. After the analysis we distinguished these states and identified the conditions for their change. We will not be in this chapter thoroughly analyze the application source code, because it is too complex and long to put it in our report. The documentation prepared for this firmware is good enough that if necessary it is able to refer to it in order to obtain information on how some particular functionality was achieved.

In return, we will explain operation of the unit, step by step from its launch until the beginning of a cyclical work in order to clarify what actions are performed by the device depending on what the current state of the device is. Figure 32 shows statecharts diagram of devices' firmware.

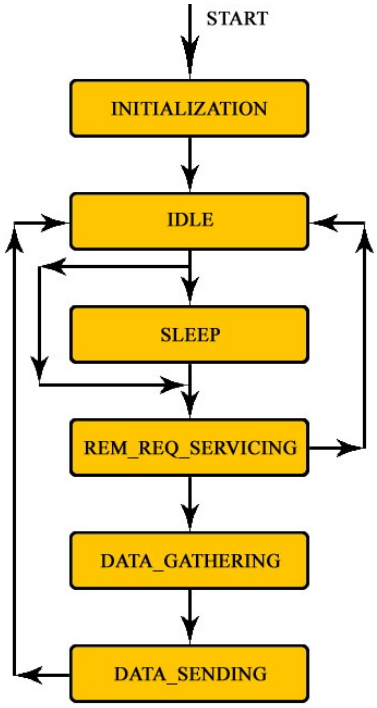


Figure 32: TS node firmware workflow diagram

As we can see the device starts its work from the entry into the state called INITIALIZATION. In fact there is no such state defined in software, because the actions that we understand under the term of initialization are executed only once, after the device is switched on and before it enters the cyclic work mode. At this point, the device makes the necessary configuration, such as initialization of peripherals, disabling unnecessary features, and also connection to the base station in order to make its registration. This registration is based on a 16-bit device address given at the stage of compiling and burning firmware at the station. This address is used by both the XBee module to a making a links, as well as by the Base Station to remember the appearance of a new node that collects data to be used in future. While every part of device is now switched on, the GPS module remains to be switched off in order to save energy.

When these are completed the unit goes into IDLE state, which verifies the fact of existence some waiting to service remote requests of execution of some activities sent by the Base Station. Initially, there will be no such requests so the device immediately moves to another state called SLEEP. However, if there will be some data waiting for service in the receive buffer of the communication interface connected to the XBee module, the device will be switched to REM_REQ_SERVICING state.

SLEEP state was designed in order to meet maximizing of energy efficiency requirement. In this state microcontroller has its core off while communication interface connected to XBee module is still active. Thanks to that the microcontroller needs now even down to 900 nA what is great result. In order to achieve this kind of microcontroller's operation the SLEEP mode service routine is responsible for calling functions that must perform the configuration which puts microcontroller in power saving mode by changing the clock source to internal clock of 250 Khz frequency and calling microcontroller's assembler command called the same as next state - SLEEP. Before that there is also neccessity to reconfigure USART communication interface connected to XBee module in order to be able to still work at 9600 baud rate although main clock frequency has changed. When everything is done microcontroller starts to work energy energy saving. To change this state XBee module has to get some request from Base Station, that will be pushed to microcontroller's USART interface thus triggering peripheral interrupt that wake core of

mcu up. Microcontroller doesn't change now its clock configuration in order to check if incomming data is really a request of performing some actions. If it is, SLEEP mode service routine calls function that is responsible for configuring device back to high-performance running mode and changes actual state to REM_REQ_SERVICING.

In this state the device receives and analyzes a command sent by the base station. At this point there are two possibilities for further operation. If the Base Station requested to perform some configuration of TS node appropriate steps are taken and after end of particular functions execution the device returns to state IDLE which tasks have already been discussed. However, if the Base Station requires measurements of the current position of node device is put in the next state called DATA_GATHERING.

The task of this state is to initialize the GPS module by switching it on and waiting an appropriate amount of time required for the proper operation of the module. After that the device collects data in number defined at the stage of burning firmware on the station, due to limitations of available memmory to store gathered samples. Once a sufficient number of samples are collected device goes to the state called DATA_SENDING.

Entering this state the device is designed to switch off the GPS module and then handle the transmission process of the collected data to the BaseStation. Because of the size limitation of one data packet sent through the network based on the ZigBee protocol, routines services this state od device has to make appropriate division of data to be transferred and the sequentially sending it to the BS node. When every data is transfered successfully device goes into the IDLE state thus ending one cycle of its work.

In this way we can outline the way of implementing firmware that controls the node. In order to learn details about specific solutions, that are closely related to hardware support, it is needed to refer to the documentation included with the project files of this firmware.

7.4. BS node device design

By analyzing the requirements of this node we can notice that Base Station should have the same modules as Terminal Station has. These modules are a communication module, a module to collect GPS data and the microcontroller, which will have the task to supervise the work of the device. Now we can once again present general block diagram of device as in Figure 33.

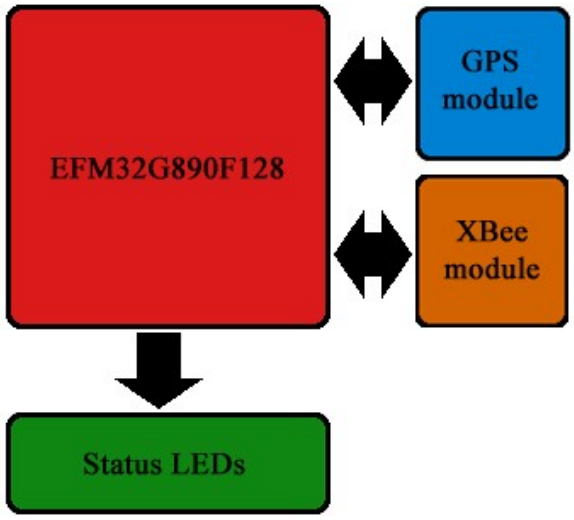


Figure 33: General TS device block diagram

However, this time we encountered a problem that prevents the creation of unit fully integrated on one PCB. Enclosures of microcontroller we could get from distributors were too small to be able to hand solder this element so we decided to use smaller of our EFM32 development board – EFM32 Gecko Starter Kit – as a part of final device. Because of this we designed PCB which allowed the connection of the GPS module and a connection to a communication module board through flexible wires, and also included a status LEDs. This board can then be combined with EFM32 development board thus yielding a fully functional device.

In addition, we designed this board the way that allows of connecting an external flash

memory in case the need to store gathered data in future project development.

As with the TS node device this device should be powered by batteries with a nominal voltage of about 3.6 V but to work with power equal to 3.3 V. Once again MCP1801 voltages regulators were used. Additionally this device is normally working with a clock of 32 Mhz frequency

Detailed schematic of made connections is presented in Appendix 7.

After preparing the schematic we could proceed to generate the PCB, resulting in board design as set out in Figure 34.

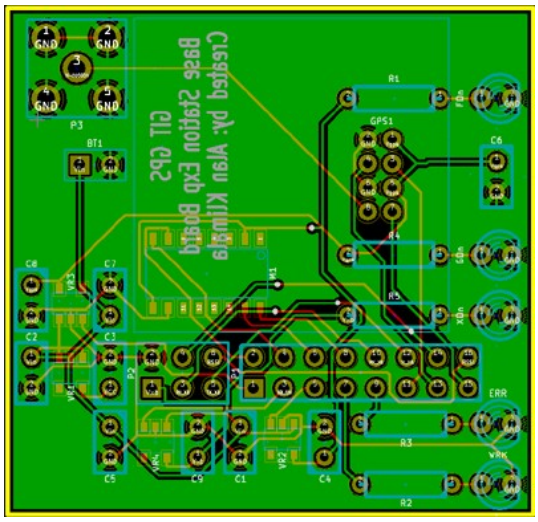


Figure 34: Base Station PCB

With ready PCB we could solder components on the board and result is presented in Figure 35. Once again GPS module is mounted on the back side and is not visible in this figure.

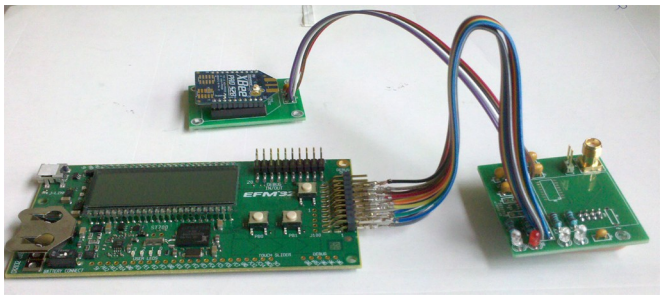


Figure 35: Ready device.

7.5. BS node firmware implementation

In contrast to the TS node firmware implementation of the BS node proved to be much more complex task than initially assumed. Because of the number of tasks that were assigned to this node, it was necessary to consider to adapt some kind of a real-time operation system as a base for the entire firmware implemented for this device. The reason is that the Base Station should be able to collect GPS data, process them with additional data gathered by TS nodes using special differential algorithm, querry TS nodes for geolocation data gathering and synchronize working of entire mesurement system. In the worst case each of these tasks will be expected to be serviced at the same time what brougth us to a conclusion that these requirements claim a solution that is capable of multitasking.

After the reasearches we decided to use an open source project called FreeRTOS. This project is scale-able real time kernel designed specifically for small embedded systems. It can be configured in preemptive or cooperative mode of work. It has portable source code almost entirely written in C. Supports inter-tasks communication and synchronization by using solutions like queues, binary semaphores, counting semaphores, recursive semaphores or mutexes and their priority inheritance. From project's webpage [10] we can also get information that it is designed to be small, simple and easy to use. While the first of these features is real truth, the next two features proved to be not as real as we could expect them to be. After a period of delight at the possibilities of what gave the use of this system in our device, based mainly on analysis of opportunities arising from the project documentation, it came time of painstaking and intricate problems that had to be solved in order to adapt this system to the requirements which have to achieved by our end solution. And this is a major cause of failure in this field of project driving, because we we have not had everything done on time. On the one hand tools nedded to start working on this part of project, EFM32 development boards, came to us later than we had expected and on the other hand the use of this system requires depth knowledge of the rules for its operation and ways to adapt it to work on a specific device and expansions of its functionality closely associated with the tasks to be carried out by it.

Therefore, first we explain what were the terms of our assumptions of how the software controlling work of the BS node should work, and then explain what objectives we have already achieved.

Firstly, this system had to be ported to our target platform. Although the official website of the project proposed ready-made solution for the platform we have chosen, there was no source code for the development environment we wanted to use in our work due to the fact that it was fully open source IDE. To achieve this we had to learn how to create scripts for the linker tool in order to properly configure the deployment of the program in the memory space of our microcontroller. The next step was to familiarize ourselves creation of a Makefiles to manage the way the target binary image of our firmware is built when using GNU C utility as a compiler.

After that we planned to implement dedicated software connected with FreeRTOS in order to meet every functional requirement that had been identified. The idea was to write four different tasks routines the way that each of them would be able to handle with one of the functionality requirements and by proper assigning priorities to these tasks ensure the reliability of the whole system.

First task should be responsible for processing data in order to identify potential risks arising from the data collected. It was expected to have the highest priority among all performed tasks due to the need of using as high computational power as it was possible in order to achieve fast operation of the prepared differential algorithm based on huge sets of data.

Second task should manage the schedule for collecting data and decide how often a particular TS node should be queried about the samples in conjunction with the results of detecting some threat due to recent computations performed on the data from this node.

Third task was expected to be responsible for nodes network management. Registering new nodes taking care of communication with each node existed in the network.

Last, the fourth task was intended to be the Idle task, that should take care of putting device in appropiate power saving mode each time the sheduler of FreeRTOS kernel switch to this task because the absence of other tasks waiting to be serviced by

microcontroller unit.

The objectives that we have achieved is succesfully porting chosen real-time operating system to our target device in order to use this project with free and open source IDE we were using. To gain this goal we prepared appropiate Makefile, linker script and modified startup code for EFM32 microcontroller. Sources of our port can now be used in some further reaserches connected with using FreeRTOS on such target platform.

Next and last thing we have done was implementing small example od Idle task, that puts microcontroller in energy saving mode. The whole software is now very simple and based on ready-made examples introduced on FreeRTOS webpage, but can be base for future implementations as an example how to use hardware support offered by chosen microcontroller in order to implement energy efficient software using real-time operating system.

We strongly believe that if we had more time all the assumptions would be made, because the knowledge we have gained during research into the possibilities and use of this solution seems to be enough for this moment to cope with such challenge. If we had such knowledge at the beginning of the project work, it should be much easier and faster to design the software that would meet all the criteria.

7.6. Conclusions

To summarize this part of the project, we can safely say that most of the assumed by us objectives have been achieved. From now on, anyone who wants to further develop the possibilities of using GPS devices to create a cheap and efficient system sensors network will have equipment that can be used for this purpose. We were able to successfully design and construct devices which can be considered as a sufficient base of hardware to perform measurements related to the implementation of an new, better differential algorithm when working, for example, with improved GPS modules that may predict a better results.

With this in mind it can be concluded that this part of the project was finished with a success, although there have not been completed firmware for the base node. The amount of work, the conclusions that flow from it and achieved results exceeding expectations that could be initially taken bearing in mind amount of time we had to execute such a complex system.

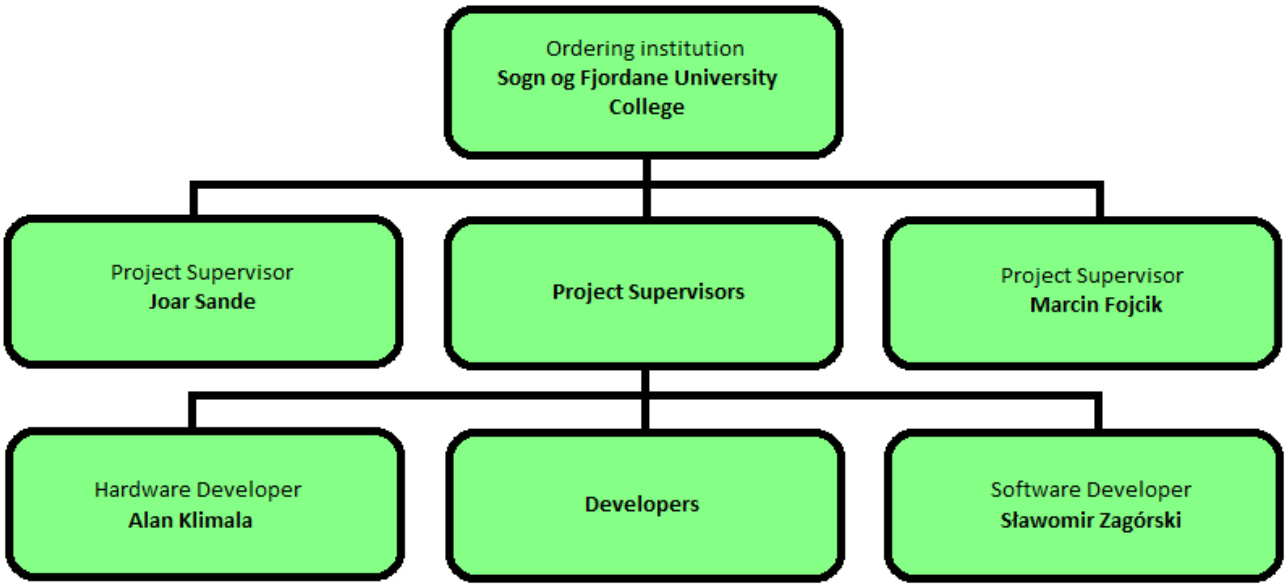
7.7. Further development

At this point we can propose two direction of system's development. For sure there should be implemented fully functional firmware for BS node based on our project. Three routines should be prepared in order to achieve three missing requirements just as it was mentioned in chapter 7.5. by discussing tasks that should be implemented in firmware.

There is also possibility to try to develop FreeRTOS in order to achieve more flexibility in EFM32 microcontroller's power saving mode support. For now everything is done in an additional application's source code rather than in core of the operational system. there is a possibility of greater integration of energy saving functionality with the FreeRTOS core, which, however, significantly exceeded the scope of this work and is a task complex enought that it can safely be considered as a topic for a separate project.

8. Organization

Organization of our project can be divided into three main group – ordering institution, supervisors group and developers group.



8.1. Order placement company

Project is being made for the Sogn og Fjordane University College that is represented by Marcin Fojcik and Joar Sande. The final result is prepared to be a part of the research programmes made in cooperation with Silesian University of Technology.

8.2. Project supervisors

Project has two supervising persons: Marcin Fojcik and Joar Sande. They are employees of Sogn og Fjordane University College.

Name	E-mail	Telephone
Marcin Fojcik	Marcin.Fojcik@hisf.no	57 72 26 70
Joar Sande	Joar.Sande@hisf.no	57 72 26 29

8.3. Developers

By developers we understand two persons: Alan Klimala and Sławomir Zagórski. They are students of Silesian University of Technology in Poland who came to Sogn og Fjordane University College on STF exchange programme.

Name	E-mail
Alan Klimala	Alan.Klimala@stud.hisf.no
Sławomir Zagórski	Slawomir.Zagorski@stud.hisf.no

9. Project administration

The project has been realized as Senior Design Project at Sogn og Fjordane University College. The subject code is HO2-300 and it has value of 20 ECTS credits. Each member of the project group spent approximately 500 hours making a project from 01.02.2011 to 06.06.2011.

9.1. Carrying out project according to plan

Unfortunatelly due to external problems, we had to change our development plan. Long shipment time of parts delayed the preparation of base and terminal stations but insufficient capabilities of GPS modules has eliminated the need of them in software part. In addition to them we created devices that was able to download GPS data from modules into our PCs. Thanks to this operation we obtained the possibility of a parallel researches on power consumption and GPS modules capabilities. In consequence, despite the many difficulties we managed to achieve success, not in producing fully functional system, but in gathering lots of data usefull for further researches. As we were supposed, we spent on the project about 500 hours per person.

9.2. Development Gantt diagram

In our schedule we took into consideration all the project development phases and we assigned the time periods to every one of them and additionally we defined time dependencies between project development phases. In critical situations, this allowed us not to fall apart and continue project execution. Schedule is shown on Gantt Diagram (Appendix 3) and Resource Plan (Appendix 4).

9.3. Week schedule

Table below presents project group schedule of hours spended at project in Sogn og Fjordane University College. Working week (9:00 – 14:30) begins on Monday and ends on Friday.

Table 4: Week schedule

Time	Monday	Tuesday	Wednesday	Thursday	Friday
8:30 – 9:30					
9:30 – 10:30	Project	Project	Project	Project	Project
10:30 – 11:30					
11:30 – 12:30	Meeting				
12:30 – 13:30	Project				
13:30 – 14:30					
14:30 – 15:30					
15:30 – 16:30					

9.4. Meeting schedule

Some differences occured between meetings schedule prepared in the preplaning phase of the project and actual meetings times. Due to mostly technical issues during project execution we decided to make only two, long general meetings with both supervisors and the rest of meetings reclassify to technical meetings only with one supervisor - Marcin Fojcik. General meetings took place at 25.02.2011 and 23.05.2011 and the technical meetings between. Table below presents actual meetings schedule. All meetings took place in Linus.

Table 5: Meetings schedule

Date	Time
25.02.2011	11:15 – 12:35
07.03.2011	11:15 – 12:00
14.03.2011	11:15 – 12:00
21.03.2011	11:15 – 12:00
28.03.2011	11:15 – 12:00
04.04.2011	11:15 – 12:00
11.04.2011	11:15 – 12:00
18.04.2011	11:15 – 12:00
25.04.2011	11:15 – 12:00
02.05.2011	11:15 – 12:00
09.05.2011	11:15 – 12:00
16.05.2011	11:15 – 12:00
25.05.2011	10:30 – 11:45

9.5. Budget

Table 6: *Budget*

Expense	Pieces	Piece cost	Comment
Developers Tools			
Computer for SVN Server	1	0,00 NOK	Provided by SFUC
Computer monitor	2	0,00 NOK	Provided by SFUC
Notebook	2	0,00 NOK	Provided by Developers
Power supply		0,00 NOK	Provided by SFUC
EVBeasyPIC evaluation board	1	300,00 NOK	Provided by Developers
PIC Cable ICD2 Debugger	1	300,00 NOK	Provided by Developers
STM32 III Cortex-M3 evaluation board	2	1 000,00 NOK	Provided by Developers
STM32-Discovery Cortex-M3 evaluation board	2	100,00 NOK	Provided by Developers
Segger J-Link ARM Debugger	1	2 000,00 NOK	Provided by Developers
Open Workbench Logic Sniffer	2	380,00 NOK	Provided by Developers
FTDI Interface Module	2	200,00 NOK	Provided by Developers
Bus Pirate v3	1	150,00 NOK	Provided by Developers
EFM32DVK Developer Kit	1	1 720,00 NOK	Bought at Avnet Memec
EFM32STK Developer Kit	1	370,00 NOK	Bought at Avnet Memec
Developer's workshop		0,00 NOK	Provided by SFUC
Hardware Parts			
PIC18F25K80 Microcontroller	6	0,00 NOK	Sample from producer
STM32F100RBT6 Microcontroller	2	60,00 NOK	
MaxStream Xbee Module	2	500,00 NOK	Provided by Developers
Passive elements		0,00 NOK	Provided by SFUC
PCB Production	1	424,00 NOK	Bought at SeedStudio
Travel and accommodation			
Tests in Linus		0,00 NOK	
Tests in open field close to student houses		0,00 NOK	
TOTAL		9 744,00 NOK	

11. List of figures

Figure 1: Proposed solution connections topology....11
Figure 2: Orbits around the Earth [1]....12
Figure 3: GPS 24-slot Constellation [2]....13
Figure 4: EVBeasyPIC Development Board [5]....21
Figure 5: ICD2 clone debugger [6]....22
Figure 6: EFM32 Gecko Development Kit [7]....22
Figure 7: Device used to communicate with GPS module....23
Figure 8: RealTerm during capturing data....24
Figure 9: Example of NMEA 0183 sentences....25
Figure 10: Number of satellites in view....30
Figure 11: Tests summary for distance A from 16.04.2011....31
Figure 12: Variations in latitude for distance A from 16.04.2011....32
Figure 13: Variations in longitude for distance A from 16.04.2011....32
Figure 14: Variations in computed distance for distance A from 16.04.2011....33
Figure 15: Tests summary for distance B from 28.04.2011....33
Figure 16: Variations in latitude for distance B from 28.04.2011....34
Figure 17: Variations in longitude for distance B from 28.04.2011....34
Figure 18: Variations in computed distance for distance B from 28.04.2011....35
Figure 19: Tests summary for distance B from 30.04.2011....35
Figure 20: Variations in latitude for distance B from 30.04.2011....36
Figure 21: Variations in longitude for distance B from 30.04.2011....36
Figure 22: Variations in computed distance for distance B from 30.04.2011....37
Figure 23: Tests summary for distance C from 07.05.2011....37
Figure 24: Variations in latitude for distance C from 07.05.2011....38
Figure 25: Variations in longitude for distance C from 07.05.2011....38
Figure 26: Variations in computed distance for distance C from 07.05.2011....39
Figure 27: XBee breakout board schematic....41
Figure 28: XBee breakout board PCB....41
Figure 29: General TS device block diagram....42
Figure 30: Terminal Station PCB....43
Figure 31: Ready device with communication module connected to it....43
Figure 32: TS node firmware workflow diagram....44
Figure 33: General TS device block diagram....47
Figure 34: Base Station PCB....48
Figure 35: Ready device....48

12. List of tables

Table 1: Comparison of error sources in stand-alone GPS and DGPS [4].....14
Table 2: Selected NMEA 0183 sentences.....25
Table 3: Number of satellites in view.....31
Table 4: Week schedule.....56
Table 5: Meetings schedule.....57
Table 6: Budget.....58

13. List of appendixes

Appendix 1: PIC18F25K80 datasheet

Appendix 2: EFM32G890F128 datasheet

Appendix 3: Gantt Diagram

Appendix 4: Resource Plan

Appendix 5: XBee PRO modules datasheet

Appendix 6: Terminal Station schematic

Appendix 7: Base Station schematic

Appendix 8: Preliminary report

14. References

[1] - Orbits around earth scale diagram, Mike1024, Wikimedia Commons, 2008, http://en.wikipedia.org/wiki/File:Orbits_around_earth_scale_diagram.svg

[2] - GPS Constellation, gps.gov, http://www.gps.gov/multimedia/images/constellation.gif

[3] - STRATCOM/AFSPC to improve global GPS coverage, Air Force , http://www.afspc.af.mil/pressreleasearchive/story.asp?id=123184576

[4] - Differential Global Positioning System (DGPS) for Flight Testing , AC/323(SCI-135)TP/189 , NATO RTO, October 2008

[5] – Propox company e-store, http://www.propox.com/products/t_180.html

[6] – Propox company e-store, http://www.propox.com/products/t_207.html

[7] – EnergyMicro web page http://www.energymicro.com/tools/efm32-gecko-development-kit

[8] – Digi company web page, http://tnij.org/xbee

[9] - Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations, T. Vincenty, Survey Review XXIII, 176, April 1975

[10] - FreeRTOS project's webpage, http://www.freertos.org/