Online GPS for PermaSense WSN

SEMESTERTHESIS

by
Patrice Guillet

Advisors:
Bernhard Buchli
Dr. Jan Beutel

Professor:
Prof. Dr. Lothar Thiele

Computer Engineering and Networks Laboratory
Department of Information Technology and Electrical Engineering
ETH Zürich

October 15, 2010

Author:	Patrice Guillet, `pguillet@ee.ethz.ch`
Advisors:	Ben Buchli, `bbuchli@ethz.ch`
	Dr. Jan Beutel, `beutel@tik.ee.ethz.ch`
Professor:	Prof. Dr. Lothar Thiele, `thiele@tik.ee.ethz.ch`

Handout:	August 2010
Due:	October 2010

Computer Engineering and Networks Laboratory
Department of Information Technology and Electrical Engineering
ETH Zürich

Acknowledgement

I would like to thank my advisor, Ben Buchli, for his support. He helped me a lot with this project and gave me many useful hints and ideas. We had many interesting discussions which brought the project forward. I am also thankful to my second advisor Dr. Jan Beutel, who could answer all my questions about the PermaSense project and explained a lot of it to me. I would also like to thank Mustafa Yuecel, Roman Lim, Tonio Gsell and Matthias Keller who are all part of the Computer Engineering Group and are experts in their areas. Special thanks go to Dr. Philippe Limpach from the Institute of Geodesy and Photogrammetry at ETH Zürich, who processed my data and gave me a lot of important input. Last but not least, I owe my thanks to Professor Thiele, who is providing all the infrastructure to do this research and is also my master tutor.

Abstract

In this semester thesis we built a prototype sensor node which is able to monitor rock movements in an alpine environment with differential GPS. Low-cost single frequency GPS receivers use the satellite-based GPS navigation system for positioning. Data gathered from at least two nodes is post-processed with methods of differential GPS to reach an accuracy in the centimeter range. The system works reliably and accurately and is fit for use in an extreme alpine environment. The GPS messages can be transmitted over wireless LAN to a backend server and are stored on an internal SD card if no connection is possible. We analysed the required GPS sampling frequency and sampling time as well as the power consumption. Duty-cycling was evaluated and implemented to limit power consumption of the prototype system.

Contents

Tables

Figures

1 Introduction

The goal of this project was to build a prototype sensor node that enables online monitoring of relative movement of alpine rock formations using differential GPS. The project was completed as part of the PermaSense[1] project. For this purpose two prototype sensor nodes, equipped with low-cost single frequency GPS receivers were built and installed on the ETZ rooftop. The data gathered by these nodes can be post-processed with differential GPS methods to extract accurate positioning and movement information. The goal was to achieve high solution accuracy while minimizing power consumption, bandwidth usage and overall system cost. To achieve this, sampling frequency and sampling time of the GPS receiver, duty-cycling of the prototype system and the power save modes of the GPS receiver were evaluated. Postion and movement in the centimeter resolution were achieved while reducing system power consumption considerably.

In the first chapter of this document the PermaSense project itself, basics of wireless sensor networks and the fundamentals of satellite navigation and differential GPS will be briefly discussed. Chapter 2 then gives an overview of related work. Chapter 3 explains the experimental setup with a description of hardware and software used for this project. Experiments conducted and the results, including analysis thereof will be shown in chapter 4. Finally, a short description of future work and concluding remarks can be found in chapters 5 and 6, respectively.

1.1 The PermaSense Project

The PermaSense project is a joint computer science and geoscience project to measure permafrost related parameters in the Swiss Alps, and conducted by ETH Zürich, Uni Basel and Uni Zürich. The goal is to provide an inexpensive solution in the form of wireless sensor networks that provide continuous datastreams from inaccessible sites for geophysical research. The extreme alpine environment with large temperature changes and massive wind, snow, and ice impacts presents huge

[1]http://www.permasense.ch

challenges for sensor node development and deployment. Despite the adverse conditions, the PermaSense team has currently two successful PermaSense wireless sensor network deployments, one at the Jungfraujoch and a second one at the Matterhorn. An additional deployment for monitoring glacier rock movement is planned to be installed on the Dirru glacier within weeks of writing this document. Further information about the PermaSense project can be obtained from the PermaSense project website[2]

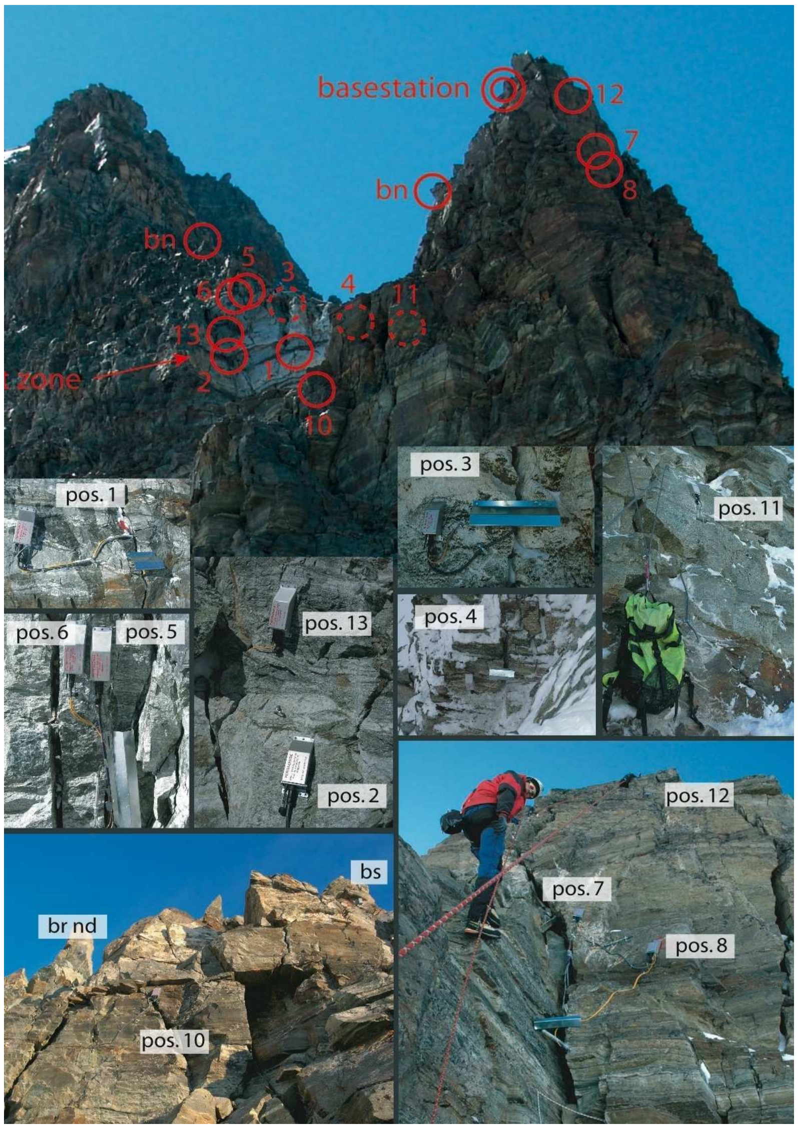

Figure 1-1
PermaSense Wireless Sensor Network deployment at the Matterhorn

[2]http://www.permasense.ch

1.2 Wireless Sensor Networks

A Wireless Sensor Network (WSN) consists of small devices, called sensor nodes, that are equipped with possibly multiple, differing sensors, with which varying parameters from the node's surroundings can be collected. The data these sensors gather is either processed on the nodes directly or sent to a centralized sink node for storage and/or processing. The data transmission between nodes, and to the data collection sink is generally provided by a low-power wireless radio link. Because reliability is an important factor and deployed WSNs are often not easily reached, the challenge is to build a network that enables multi-hop communication so that every node can communicate reliably with every other node (including sink) within the network, without generating unnecessary traffic. A multi-hop, tree-like network structure is thus used to guarantee that communication prevails even if intermediate nodes are not reachable due to node failure, temporary disconnection from the network, or environmental obstacles. Due to remote deployment sites without access to reliable energy sources WSN nodes are usually battery powered and expected to last multiple years without human interaction. Low-power design is thus essential, further adding to design and deployment challenges.

1.3 Satellite Navigation

A Global Navigation Satellite System (GNSS), of which the American NAVSTAR system (commonly called GPS) is the most well known, can be used to determine position and time at any point on earth using an appropriate satellite signal receiver.
The GPS system consists of three segments. First, the space segment includes all operational satellites that continuously transmit GPS information. Second, the control segment consists of all the ground stations, which collectively monitor and control the space segment. Finally, the user segment contains all GPS receivers, which are found in cell phones and other consumer electronics and in high end devices for military purposes alike. A GNSS system user expects from it to accurately and quickly determine her position, but many other use cases have been described, such as for tracking property, locating closest medical first responders, and even for surveying and map making.
The basic principles of satellite navigation are easily understood, but the overall process is very complex and makes use of advanced technology and a fair amount of data crunching. The basic procedure to determine the location of a GPS receiver is as described next.
A satellite in the earth's orbit sends a message with a time stamp on a known frequency. The receiver, listening on the predetermined frequency can therefore lock onto the signal and decode the message from the carrier wave. It then compares the message's time stamp with its own clock to measure the time difference $\Delta\tau$. To extract the satellite signal from multiple satellites the receiver uses code correlation techniques to distinguish individual satellites. An internal replica of the incoming signal is generated and aligned with the received satellite signal. The receiver shifts the replica code to match the incoming code from the satellite. When the codes match, the satellite signal is compressed back into the original carrier frequency band. Receivers typically use phase-locked-loop techniques to synchronize

the receiver's internally generated code and carrier with the received satellite signal. A code tracking loop is used to track the signals while a carrier tracking loop is used to track the carrier frequency. The two tracking loops work together in an interactive process, aiding each other, in order to acquire and track the satellite signals. A generic GPS receiver tracking system is illustrated in Figure 1-2 [9].
Next the process to determine location using the GPS signals is described.

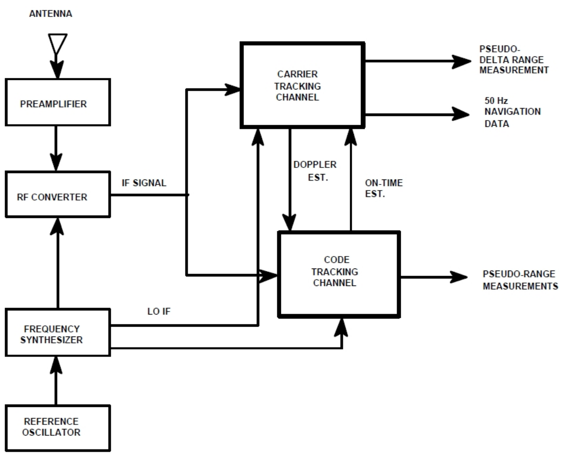

Figure 1-2
Generic GPS receiver tracking system [9]

1.3.1 GPS Positioning

In a one dimensional scenario (see Figure 1-3) we can calculate the distance from the satellite to the receiver, also called pseudorange (PSR), with the time difference $\Delta\tau$ and the known velocity of the signal (roughly speed of light, i.e. c=300'000 km/s), as shown in Equation (1.1).

$$PSR = \Delta\tau \cdot c \tag{1.1}$$

Accurate knowledge of time is very important in satellite navigation systems. A time error of $1\mu s$ in Equation (1.1) will, due to the fast signal speed, lead to a positioning error of 300m. Although every satellite is equipped with four atomic clocks, including an atomic clock in every receiver is clearly not feasible. Therefore, to eliminate the time shift, the signal from an additional satellite is required.

Referring to Figure 1-4 and Equation (1.2), the position can now be calculated with the two signal travel times $\Delta\tau_1$ and $\Delta\tau_2$, the distance between the senders A and the speed of light c.

$$PSR = \frac{(\Delta\tau_1 - \Delta\tau_2) \cdot c + A}{2} \tag{1.2}$$

Analogously, to determine a position in three dimensional space, the pseudoranges from three satellites are used to find the intersection at the receiver's pseudoposi-

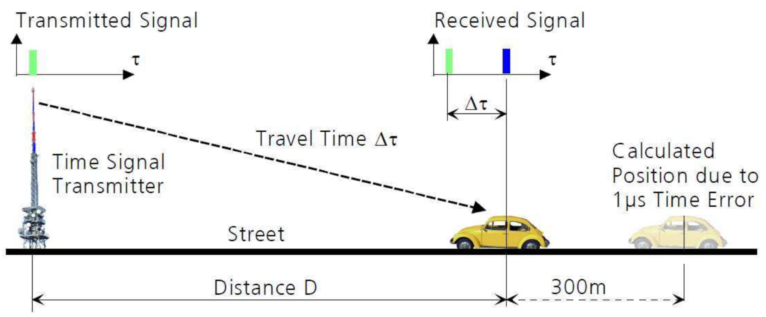

Figure 1-3
One-dimensional example of distance measuring by signal travel time [15]

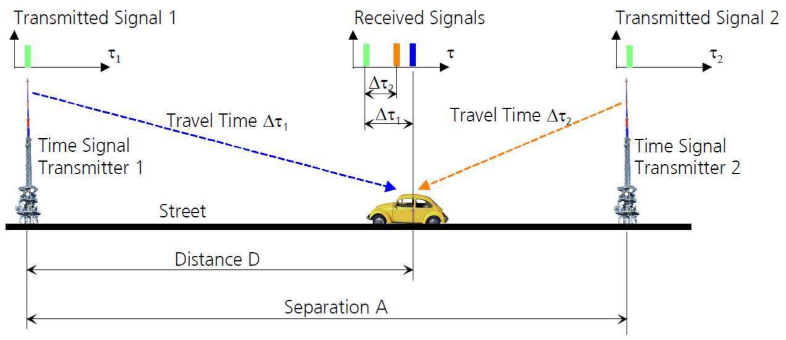

Figure 1-4
Clock error compensation with an additional signal [15]

tion. To account for the time shift, the signal from a fourth satellite is required. The pseudorange is now calculated as shown in Equation (1.3).

$$PSR = R + \Delta t_0 \cdot c \tag{1.3}$$

R denotes the real distance between the receiver and the satellite, c is the signal speed and Δt_0 the difference between the satellite clock and the receiver clock. In a cartesian coordinate system with i=4, Equation (1.3) is expanded to Equation (1.4).

$$PSR_i = \sqrt{(X_{sat\_i} - X_{rec})^2 + (Y_{sat\_i} - Y_{rec})^2 + (Z_{sat\_i} - Z_{rec})^2} + \Delta t_0 \cdot c \tag{1.4}$$

Using four satellites, Equation (1.4) leads to four non-linear equations in four variables. This set of non-linear equations can be solved with an iterative Taylor Series approach [15].

1.3.2 Errors in GPS measurements

In section 1.3.1 the basic process to find a position using GPS signals was explained. The solution accuracy depends on many factors. Table 1-1 shows several possible sources of error in GPS measurements.

The data transmitted by the satellites contains ephemeris[3] and clock data with limited accuracy. The ionosphere has a big effect on the travel time of the GPS signals. In vacuum these signals travel at the speed of light, but the ionized gas molecules in the ionosphere slow down the electromagnetic waves. Because the ionization levels vary, the signal delay due to ionization is not constant. The density of the troposphere also has an effect on the signal travel speed. Increased density or humidity retards the speed of the satellite signals. Reflections of the GPS signals lead to multipath errors. GPS receiver measurement noise and internal time delays introduces further errors. Last but not least the actual satellite constellation also has an effect on accuracy. If all visible satellites span a small volume (i.e. are densly spaced), the overlapping range-error areas are larger and the position can be determined less accurately.

Error cause	Approximate error
Ephemeris data	1.5m
Satellite clocks	1.5m
Effect of the ionosphere	3.0m
Effect of the troposphere	0.7m
Multipath reception	1.0m
Effect of the receiver	0.5m
Total RMS value	4m

Table 1-1: GPS error causes [15]

1.3.3 Differential GPS

For this project, solution accuracy well below the values shown in Table 1-1 is required. For this purpose differential GPS (DGPS) was employed, and its basic operation is briefly explained in this section.
A first approach to DGPS consists of measuring the signal travel time and use a correction factor from a reference station. The reference station with an accurately known position receives GPS signals and compares the computed pseudorange with its own location. The pseudorange error can then be used for all other deployed receivers, either by direct communication or during post-processing. Post-processing is conducted offline by specialised software with the data of all receivers from a certain period. The correction factors from the reference station are valid for receivers within a range of up to 200km. These factors are applicable because the signals read by the reference station and deployed receivers travel approximately through the same media, and are hence affected by it in a similar way. With signal travel time delay measurement an accuracy of approximately 1 meter can be achieved. For more accuracy carrier phase analysis can be included, which is briefly explained next.

The publicly available GPS signal is a periodic electromagnetic wave with a wavelength of approximately 19cm. There is a large number of complete wave cycles N

[3]A mathematical description of a satellites orbit

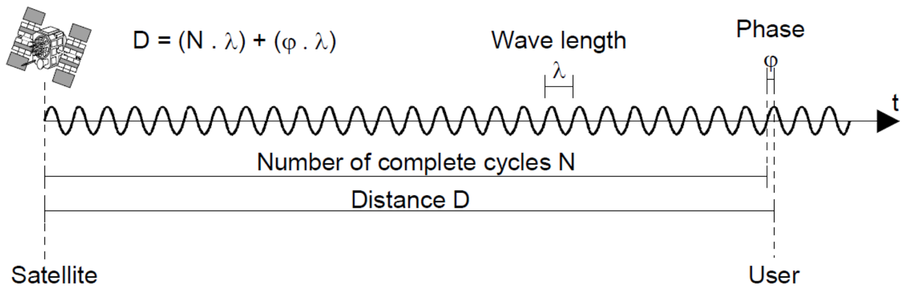

Figure 1-5
Principle of carrier phase measurement [15]

and one partial cycle (phase) φ (see Figure 1-5). The carrier phase measurement is thus ambiguous; it looks the same every 19 centimeters. By observing different satellites at different times and comparing the measured data with the reference station, the position can be calculated to an accuracy of a few milimeters. A double-differencing technique is used to remove the satellite and receiver clock errors. This is inherently a relative positioning technique, therefore the user receiver must know the reference station location to determine its absolute position [9]. If there is a longer pause in sampling, track of complete cycles N is lost. It is thus essential to have continous measurements over a long time to avoid cycle slips.

2 Related Work

Most work in the field of environmental tracking with GPS has been conducted with much more expensive dual frequency GPS receivers. Work in this area contains for example tracking of earthquakes [5]. GPS receivers are used as seismometers, but sampling frequency is an issue because of the higher frequency of seismic waves. Some work has been conducted using DGPS with low-cost GPS receivers [3]. The author studied various approaches for high accuracy point positioning using single frequency GPS receivers. [1], on the other hand, compared different receivers and also concluded that a sampling interval of 1s or 30s makes no difference in accuracy. A preliminary experiment with low cost GPS receivers for glacier tracking was in 2003 [4]. This thesis showed problems with accuracy of the solution and reliability of the nodes. DGPS was not used and sampling was rather short (2-5 minutes per day), which lead to inaccurate results. GPS integration in WSNs has also been used for localizing sensor nodes [11]. This paper proposed that the deployer of sensor nodes be equipped with GPS and the deployment of the nodes could thus be localized more precisely. [6] used GPS collars for cattle to monitor the behaviour in their environment. These collars communicated over a WSN. Finally, the preliminary work for this thesis was completed as a semester thesis at ETH Zürich [7].

3

Experimental Setup

This chapter briefly discusses the setup for the experiments conducted (see Chapter 4), and explains both the hardware and software components of the GPS enabled node. Our experimental setup consists of two Permasense CoreStation each equipped with a u-blox LEA-6T GPS module. They are connected to a GSN backend server via WLAN. A connection to a Basestation over the Dozer [2] network is possible but not used (see Figure 3-1)

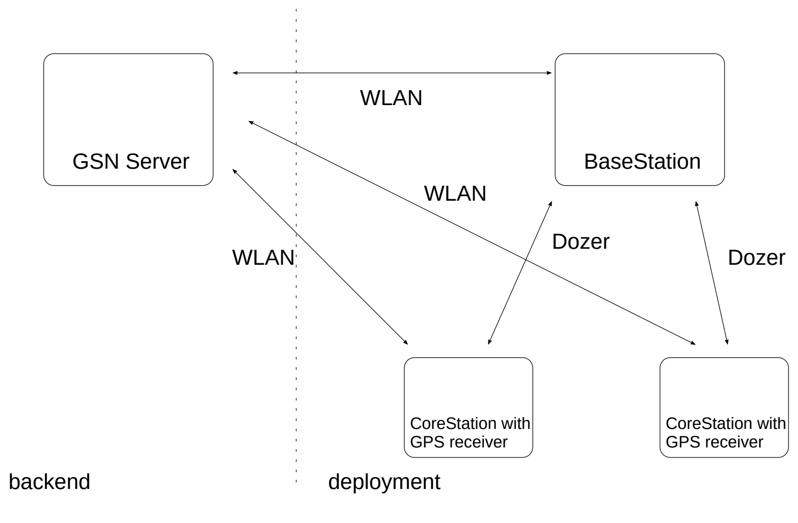

Figure 3-1
The experimental setup

3.1 Hardware

The Permasense CoreStations are equipped with the components described in this section. The components are packed in a waterproof metal box, with all connections to the outside of the box sealed waterproof and EMP protected. The CoreStations accept an input Voltage between 5-18 V, and are powered by a 12V battery. Figure

3-2 provides a block diagram of the hardware components, which are individually discussed in the following subsections.

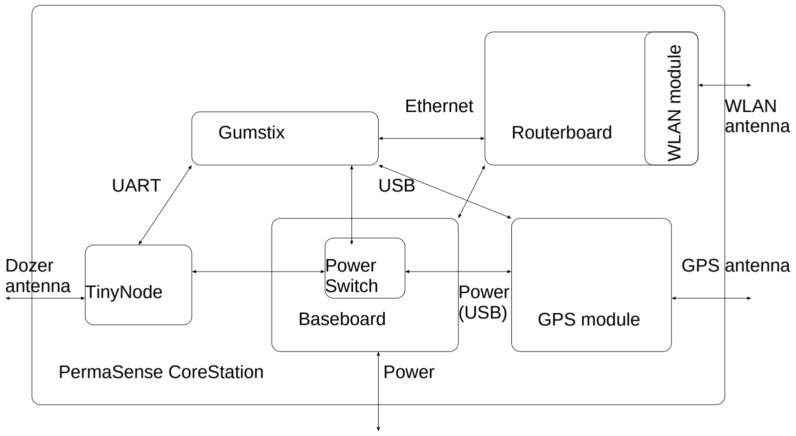

Figure 3-2
Block diagram of the CoreStation hardware. Connections not labeled are power supplies

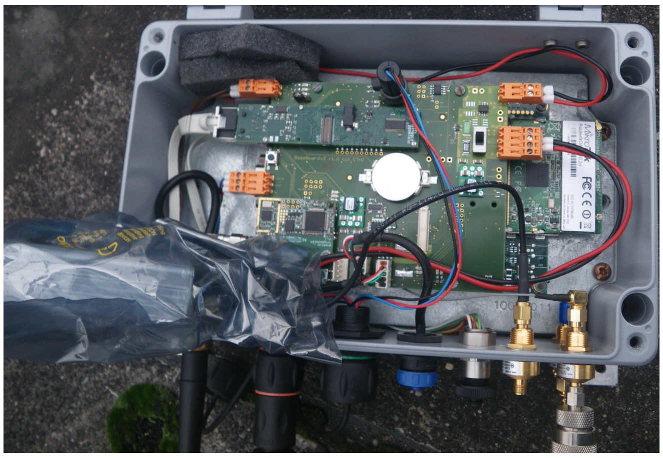

Figure 3-3
The PermaSense CoreStation with the GPS evalutation kit

3.1.1 Baseboard

The BaseBoard v2 is a custom ETH printed circuit board. It is the basis of the CoreStation connecting all the components. It features a unregulated power input with 5 - 18 Volt. Onboard generated voltages are 3.3V, 4.2V and 5 V. A backup battery ensures the power supply of a real time clock (RTC). The connectors include one TinyNode connector, one gumstix verdex connector and three USB connectors. Although available, the GSM connector was not used for this project. The power supply of the BaseBoard can be turned off manually or by an attached TinyNode.

3.1.2 Gumstix PC

The heart of the CoreStation is the gumstix verdex pro XL6P embedded computer[1] running an Ångström[2] linux distribution. In the CoreStation, the gumstix is extended with a netpro-vx ethernet interface. This 10/100baseT interface is connected to the Mikrotik router, which provides much more reliable wireless network access than an optional WLAN module on the gumstix board. The verdex pro XL6P contains a 600MHz Marvell PXA270 processor and has 128MB of RAM and 32MB of flash memory. It also features a MicroSD card slot for additional flash memory, which is used for logging data when no network connection exists. The gumstix is the heart of the system and runs the PSBacklog (described in Section 3.2.2) software, which controls logging and communication with the sensors and the centralized sink node or backend server.

3.1.3 Mikrotik WLAN

Communication from the CoreStation to the GSN backend server is done via a Mikrotik Routerboard 433[3] combined with a wireless LAN card. The Mikrotik RB433 comes with a Atheros 300Mhz CPU, three LAN and three miniPCI connections. It is combined with a Mikrotik R52 miniPCI card for wireless LAN. It supports 802.11 a/b/g in the 2GHz and the 5 GHz band, of which the latter was used for this project.

3.1.4 TinyNode

The Shockfish TinyNode 184[4] can communicate over the Dozer [2] network and is in control of the power supply of the system. The TinyNode is a 30x40 mm small, low power radio device with a 16MHz TI MSP430 microcontroller and a Semtech SX1211 ultra-low power wireless transceiver. Communication with the gumstix is possible via an UART connection. The TinyNode can cut the power from the other components on the baseboard.

3.1.5 GPS Module

The GPS signals are received with a u-blox LEA-6T evaluation kit interfaced and powered via USB. The LEA-6T GPS module allows for precision time measurements and can give out RAW data which is needed by the post-processing software. An external active patch antenna is used to receive the GPS signals. UART and DDC interfaces are available, but we communicate with the GPS module over USB. The protocol used is the u-blox proprietary UBX protocol. The details of the UBX protocol can be found in the u-blox protocol specification [14]. The LEA-6T can also be configured to deliver complete navigation solutions.

[1]http://www.gumstix.com
[2]http://www.angstrom-distribution.org/
[3]http://www.mikrotik.com/
[4]http://www.tinynode.com/

Parameter	Used Setting	Description
priority	20	Priority for sending to GSN
gps_device	/dev/ttyACM	Interface of the GPS module
poll_interval	30	Time between GPS measurements
measurement_time	10800	GPS sampling time
quality_threshold	6	Threshold for good signals
signal_threshold	25	Threshold for good signals

Table 3-1: [GPSPlugin_options] section of backlog.cfg

3.2 Software

In this section the system software is discussed. All node software is executed on the gumstix verdex running an Ångström linux operating systemFigure 3-4 shows a high level overview of the software stack. In the following subsections, the software is described in a bottom-up approach from data aquisition to post-processing.

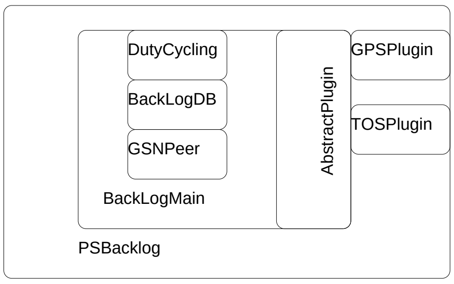

Figure 3-4
Block diagram of the PSBacklog

3.2.1 GPSPlugin

For reliable data acquisition a software that is able to configure and read the u-blox LEA-6T GPS module is needed. Data should be delivered periodically from the GPS receiver and handled by the software. Also, the connection to the module should be reliable. We use a plugin for PSBacklog. The plugin reads its parameters during the Init phase (see Section 3.2.1.1) from the backlog.cfg file (see Table 3-1).

3.2.1.1 Init

To initialize the GPSPlugin and the GPS device, the plugin specific parameters are first read from the configuration file. Then, the serial port is opened to configure the device. The device parameters are polled and compared to the configuration parameters and only overwritten when differing. The parameters we can set are:

These parameters can be set with UBX-CFG messages. The exact format of the messages can be determined with the u-blox protocol specification [14].

Parameter	Used Setting	Description
Measurement rate	GPS_TIME and value from poll_interval	Period between measurments in milliseconds. Time must be set to GPS_TIME
Port protocols	UBX only on USB	Accepted protocols (NMEA/UBX) on I/O ports
Message format	RXM-RAW on USB with rate 1	Message type on I/O port with update rate
Power Mode	0	0: max. performance, 1: power save mode, 4 eco
Power Save Mode Settings	5,1,0,0,0	Update period needs to be ≤ 5s for power optimized tracking
Save configuration	Save current configuration	Clear, save or reset configuration to flash memory

Table 3-2: Configuration of the GPS module

3.2.1.2 Run

When the GPSPlugin is running, it is waiting for messages from the module. Once a message arrives on the serial port the header is analyzed and, if matching the expected header, the message is read. The checksum is calculated and compared with the checksum provided by the receiver. This is done for a measurement time specified in the backlog.cfg. There are two possibilities for getting RAW messages. The originally implemented method used a poll. By sending the message header of the RAW message with an empty dataload, a RAW message can be polled from the receiver. By setting the measurement rate and the update rate in a configuration message, no polling is needed but the messages can be read from the serial port at the desired rate.

3.2.2 PSBacklog

PSBacklog is a custom software for the PermaSense project developed by the PermaSense team. PSBacklog can process data from sensors by sending them to GSN or storing it in a local database on a SD card if no connection is possible. If a connection to GSN is established it writes the data stored in the local database to GSN and deletes it from the database.
The AbstractPlugin routine describes an API to include plugins into the PSBacklog (see Figure 3-4). Those plugins are controlled completely by PSBacklog. If PSBacklog wants to shut down it calls the isBusy routine of each plugin. This will return if the plugin still need time to complete its task. To avoid being blocked by a malfunctioning plugin, a maximum execution time can be set. This implementation was not very practical for our purpose, because the backlog went to shutdown mode first, before it checked the plugins.
The new PSBacklog is plugin-dependent. Similar to the isBusy routine each plugin has a howLong routine. When called, each plugin returns a value in seconds speci-

fying how much time it will still need to execute its task. The PSBacklog then waits for the maximum of the returned values before it runs the shutdown routine. Only when all plugins have finished, the PSBacklog goes to shutdown mode.
Some of the plugins are not designed to run in duty-cycle mode or for a long time. It is thus important to only include the plugins we need. With the new duty-cycling mode it is possible to shut down the gumstix and the components and turn them back on again with the help of the TinyNode. In addition to the power saving achieved, we get the advantage that the gumstix is rebooted to a known state regularly.

3.2.2.1 DutyCycling

The initial setup used cron for duty cycling. Jobs to execute were saved in a schedule file formatted as a crontab. These schedules could only be uploaded via GSN. For every task in the schedule the gumstix is started if it is not already running. The content of the job does not matter, hence a dummy script can be executed as an alarm clock. Shutdown signals and wake-up timers for the TinyNode are set by the PSBacklog. When a shutdown signal is received the TinyNode cuts the power to all components. After a timer has run out it reconnects the power causing the gumstix to boot. The first problem with this approach was that for short execution intervals it was possible that the PSBacklog was in shutdown mode while the next job should have been executed. After the PSBacklog finished shutdown and gave the order to the TinyNode to cut the power, the TinyNode had already fired its wake up signal and would therefore not send another wake up signal. The gumstix thus went to sleep forever (or at least until the next service window, which caused the node to wake up at a user definable time). This problem was solved quickly by setting the next wake up timer at the end of the shutdown mode and giving a minimum time avoiding negative timers we got with missed wake up times.
The ScheduleHandler turned out to cause kernel panics when the duty cycle mode was activated. The suspicion was that there exists a problem with the job execution. The new DutyCyclingClass written does not execute any jobs, but only schedules wake ups. It can be configured via the PSBacklog configuration file (see Table 3-3). This very reduced class does not make any connection to GSN and cannot read crontab-like schedules. However, kernel panics disappeard completely with the DutyCyclingClass. An additional feature introduced with this class is the ability to duty-cycle the wireless network interface, a component that draws significant current. Turning the Mikrotik router and the wifi off when possible leads to substantial power savings. These results are described in section 4.2.4.

3.2.3 GSN

The backend is implemented with a GSN[5] server. GSN is a streaming-data database developed at EPFL. GSN is a software middleware designed to facilitate the deployment and programming of sensor networks. In functions as a connection between the sensor network and the internet. With so called virtual sensors in an XML format we can make received data visible in a web interface. GSN stores all received data in SQL databases dependent on the message type.

[5]http://sourceforge.net/apps/trac/gsn/

parameter	used setting	description
plugin_dependent_mode	1	Let plugins determine runtime if enabled
uptime_minutes	10	minimal uptime
downtime_minutes	1260	Time between shutdown and boot
hard_shutdown_offset_minutes	1	Time between shutdown signal and power cut
approximate_startup_seconds	90	Amount of time estimated for startup
wlan_duty_cycling	1	Enable WLAN duty-cycling
wlan_uptime_minutes	10	Time WLAN is on
wlan_downtime_minutes	160	Time WLAN is off

Table 3-3: Entries in the [dutycycling] section of backlog.cfg

3.2.4 Bernese

Bernese has been developed by a team at the University of Berne[6]. It is able to calculate and compensate the effects of ionospheric disturbances and inaccurate satellite data for an accurate solution. Bernese provides solving of carrier phase measurements. Out of the GSN database we can extract a cvs file with our GPS data. This file has to be converted to a RINEX file, which serves as input for the Bernese GPS Software. This software, which can also be used for GPS data post processing, computes a solution for the location and movement of the observed agent. My data has been analyzed with Bernese by Dr. Philippe Limpach.

[6]http://www.bernese.unibe.ch

4 Results

4.1 Experiments

To obtain data that permits evaluation of solution accuracy an experiment with the setup described in Chapter 3 was conducted on October 7, 2010. The duty-cycle was set to plugin-dependent active time. Sampling time was set to 4 hours, with a sampling interval of 30s followed by 3 hours downtime. The two CoreStations with their GPS antennas were placed on the roof of the ETZ building at ETH Zürich about 25 meters apart from each other (see Figure 4-1). At 10:20am the GPS antenna of the rover node was moved 5 centimeters in the horizontal plane. At 4:20pm the GPS antenna was moved again by 10 centimeters in the horizontal plane and about 10 centimeters in elevation. The data gathered was collected in the GSN backend and sent for analysis to Dr. Philippe Limpach, who post-processed the data with the Bernese GPS software. Also the data of the entire week the system was running was analyzed (see Figure 4-4).

Current consumption of the setup and individual components was done in the lab with a power analyzer. A voltage of 12V was supplied and the current consumption of both the entire CoreStation in full performance mode, and the CoreStation with disabled wireless was measured. The GPS module's current draw with a supply voltage of 5V, running it in full performance mode, eco mode, full power save mode, and power optimized tracking mode was also measured.

4.2 Analysis

4.2.1 GPS Sampling

4.2.1.1 Sampling frequency

Several studies [1][3][8] showed that the GPS sampling frequency does not need to be as high as possible. For accuracy, a sampling period of 1 second and a period of 30 seconds yielded the same results. Mostly these results were obtained by measuring

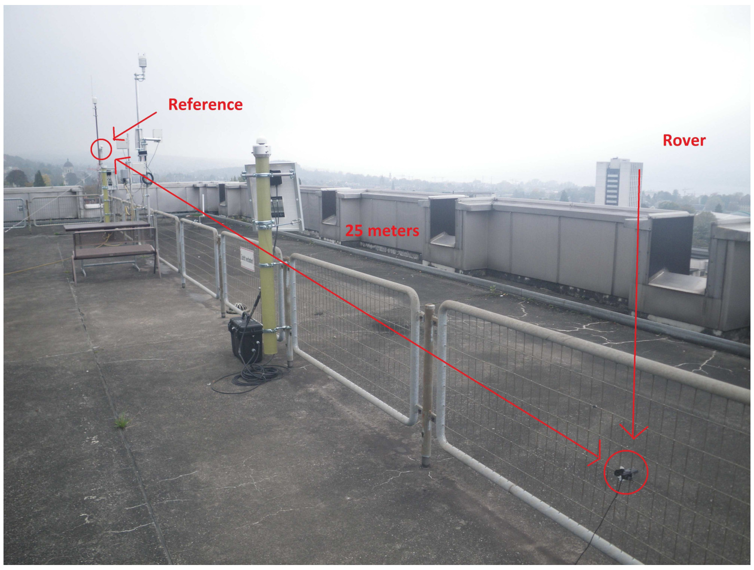

Figure 4-1
The experimental setup on the ETZ rooftop

in a 1 second interval and downsampling the data to an interval of 30 seconds. The measurements in this project were made with a period of 30 seconds.

4.2.1.2 Sampling time

The main goal was to get accurate positioning and movement solutions, therefore continuous tracking of the GPS satellites for carrier phase analysis was required. Experiments showed that a continuous sampling time of at least 2 hours is necessary to get accurate results, but a sampling time of 4 hours was chosen for the final implementation as a trade-off between accuracy and power consumption. A longer sampling time does not give significantly better results as can be seen in Figure 4-2.

4.2.2 Solution Accuracy

With the sampling frequency set to $\frac{1}{30}Hz$ and the sampling time set to 4 hours, as definend before in section 4.2.1, the goal of an accuracy in the centimeter range was reached. The movements from the experiment are clearly recognizable. Experiments by Dr. Philippe Limpach [8] showed that a short baseline is essential to improve accuracy. It is thus recommended to have a reference station close by, but of course outside the moving test site.

4.2.3 Power Consumption

Power is an important issue no only of the PermaSense project, but in general for WSNs. Because the nodes are not easy accessible they have to be designed for a long runtime. For the prototype system it was clear, that too much power was consumed. An estimate with manufacturer provided values of the systems current consumption lead to the values shown in table 4-1.

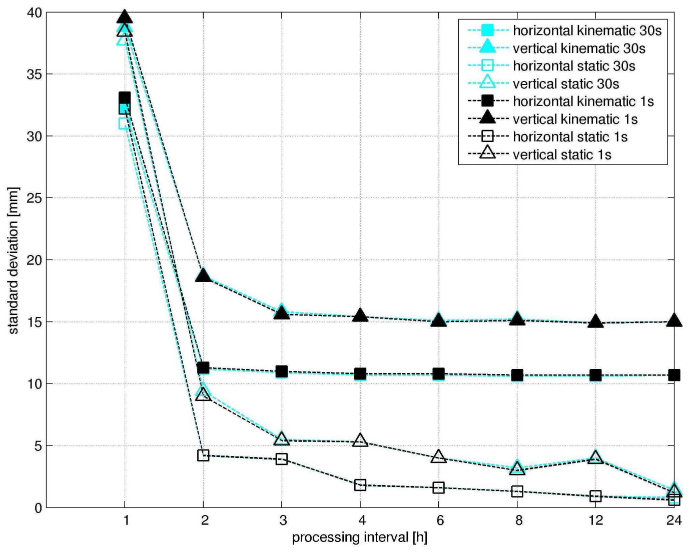

Figure 4-2
Positioning error dependent on sampling frequency and sampling time [8]

component	current consumption
Baseboard	50mA
Gumstix verdex	100-300mA (idle - full load)
netpro-vx	100mA
Routerboard	150mA
TinyNode	$< 1\mu$A -25mA
GPS module	50mA
Total	550mA

Table 4-1: Estimated current consumption of the CoreStation components

Measurements with the power analyzer confirmed that the whole system at peak performance consumed 535mA. When turning off the ethernet part of the gumstix and the routerboard with the WLAN module, the current consumption was reduced to about 250mA. Average current consumption of a 50% duty-cycle with duty-cycled WLAN was 120mA. The GPS measurements itself need only about 10% of the total current draw. However with this system the GPS is powered over the Baseboard USB and the GPS measurements are read by the gumstix computer. So, these components have to be on while we measure. The gumstix has about 60 seconds to boot. Hence if we cannot increase our GPS sampling period well over 2 minutes, it will not be worth shutting down the gumstix. We will now split up the power discussion in three parts. Entire system, GPS and wireless.

4.2.3.1 System

With the duty-cycling mode it is possible to cut the entire system from power and only run the TinyNode. The power consumption of the TinyNode in sleep mode is

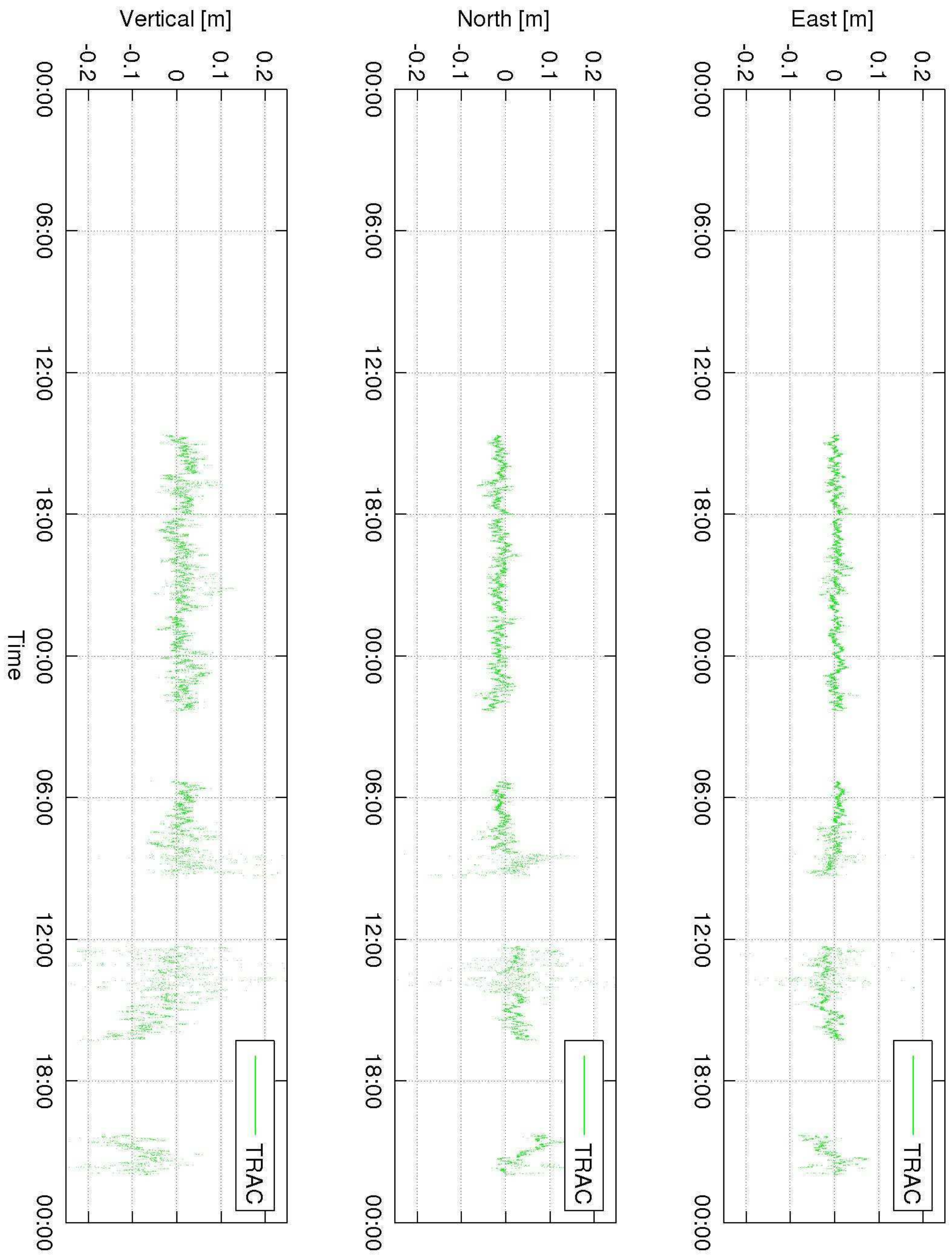

Figure 4-3
Positioning data of ETZ rooftop setup for 1 day

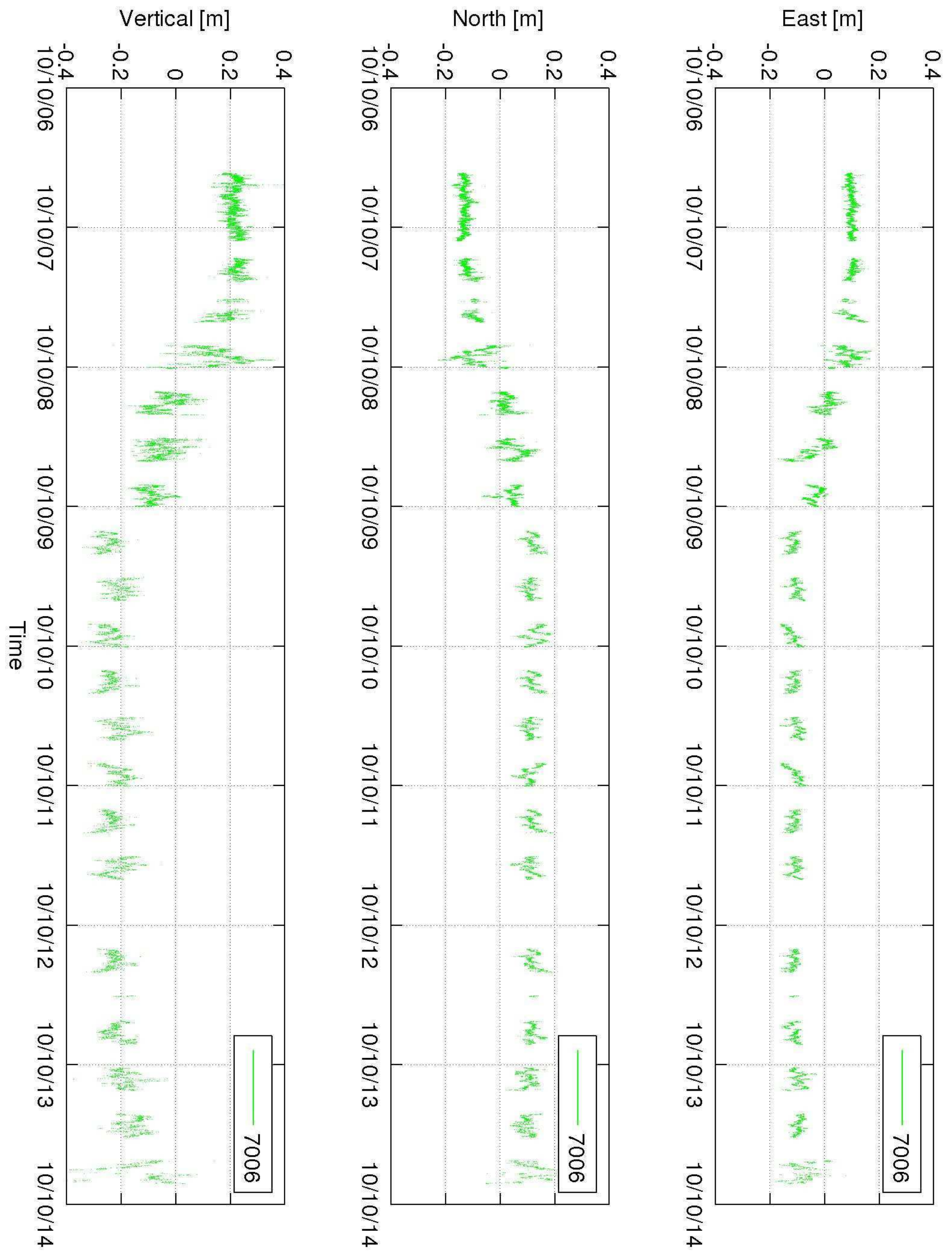

Figure 4-4
Positioning data of ETZ rooftop setup for 1 week

below 1μA and thus neglectable. The advantage of having the TinyNode powered at all times is that the system remains reachable over the Dozer network. Because the power supply over the baseboard still draws some current, we achieve a current consumption of a few milliampère in sleep mode. As explained in Section 3.2.2.1, the PSBacklog sets the timers on the TinyNode for cutting the power and repowering. All components boot automatically when connected to power.

4.2.3.2 GPS

The u-blox LEA-6T GPS module has three power modes: Maximum performance, eco mode, and power save mode. According to the specifications, the difference between maximum performance mode and eco mode is that in eco mode the acquisition engine is turned off as soon as a sufficient number of satellites are being tracked. New satellites are still acquired though. Our power analysis showed no visible difference between the two modes and was hence uninteresting.
The power save mode has a variety of parameters that can be adjusted. For details the u-blox protocol specifications [14] can be consulted. In this mode there still are yet two more modes. One is power optimized tracking (POT), and the other shuts down the module completely for a user defined amount of time. POT is used when the update period is less or equal to 5 seconds. In the power optimized tracking mode, the GPS receiver still tracks signals but does not acquire new signals or download data, which significantly reduces current drawn. To go into POT the signal strength must be above 30db/Hz. Because we use the carrier frequency shift for differential GPS we have to track the satellites continuously to avoid cycle slips. Update periods above 5 seconds with shutting down the receiver in between are therefore not an option. Still, the power analysis showed that when not downloading data, an update period of 5 seconds uses ≈55mA for 1 second and ≈26mA for 4 seconds (see Figure 4-5. This leads to significant savings on power, because in the other available power modes the receiver requires approximately 55mA continuously. Unfortunately it is not possible at the moment to make use of the power save modes over USB. The release notes of the u-blox 6 firmware version 6.02 state that USB disconnects can occur when entering backup mode, and it was found that this also occured with the power optimized tracking mode.

4.2.3.3 Wireless

Refering again to Table 4-1, a power-hungry component is the wireless network interface. Turning off the Mikrotik router with wifi card, the current consumption was reduced by 280mA. This is about half of the total current draw by the system. Out of that realization, the idea was born to duty cycle the wireless part. As seen before in Section 4.2.3, the gumstix has to be running during GPS measurements. If the gumstix has no connection to the GSN server, the measurements are saved in the local database on the SD card. If a connection can be established, the entries in the local database are transmitted to GSN. Further details are given in the next section.

4.2.4 Duty-cycling

In this section the power cycling possibilities of the system are explained.

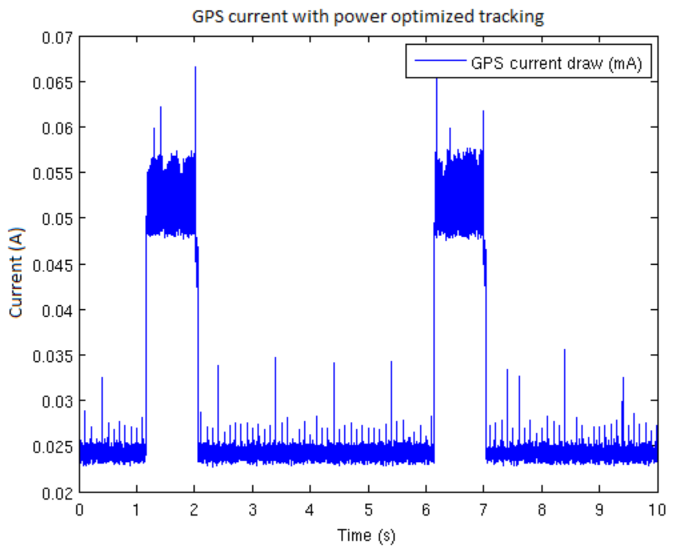

Figure 4-5
Current consumption of the GPS module with power optimized tracking

4.2.4.1 System

The sampling analysis showed that a continuous sampling time of 3 hours is required. If one such sampling period per day is assumed, the duty-cycle results in

$$DC = \frac{3h}{24h} = 12.5\% \tag{4.1}$$

4.2.4.2 WLAN

One GPS RAW message consists of 8 Bytes plus 24 Bytes for each tracked satellite. Depending on the sampling frequency the transmission time is

$$T_{WLAN} = \frac{data\, per\, sample \cdot number\, of\, samples}{transmission\, rate} = \frac{(8 + SV \cdot 24) \cdot (T_D \cdot f_s)}{TX} \tag{4.2}$$

where SV is the number of visible satellites, T_D the sampling time, f_s the sampling frequency, and TX the transmission rate. The measurements showed that on average there are about 10 satellites visible. Experience from the system deployed on the Matterhorn shows that a throughput between 40kB/s to 1 MB/s can be expected. This results in a average packet size of $8 + 10 \cdot 24 = 248Bytes$. For a sampling time of 3 hours and a sampling frequency of $\frac{1}{30}Hz$ this results in a transmission time of

$$T_{WLAN} = \frac{248Bytes \cdot 360}{40kB/s} = 2.2s \tag{4.3}$$

If we assume one 3 hour sampling period per day we get a duty-cycle percentage of

$$DC_{WLAN} = \frac{2.2s}{24 \cdot 3600s} = 0.0025\% \tag{4.4}$$

Component	Max. Performance	Active Time (3h/day)	Duty-cycling Performance
Baseboard	50mA	12.5%	6.25mA
Gumstix verdex	175mA	12.5%	21.88mA
net-vx	100mA	0.69%	0.69mA
Routerboard with WLAN module	190mA	0.69%	1.31mA
GPS	60mA	12.5%	7.5mA
Total	550mA		37.63mA

Table 4-2: Duty-cycling performance of the CoreStation setup

Wlan dutycycling thus has a huge effect on overall power consumption. However, in order to guarantee service access to the node the WLAN active time was increased to 10 minutes. With Equation (4.4) this still gives a duty-cycle of 0.69%.

The merged results can be seen in Table 4-2

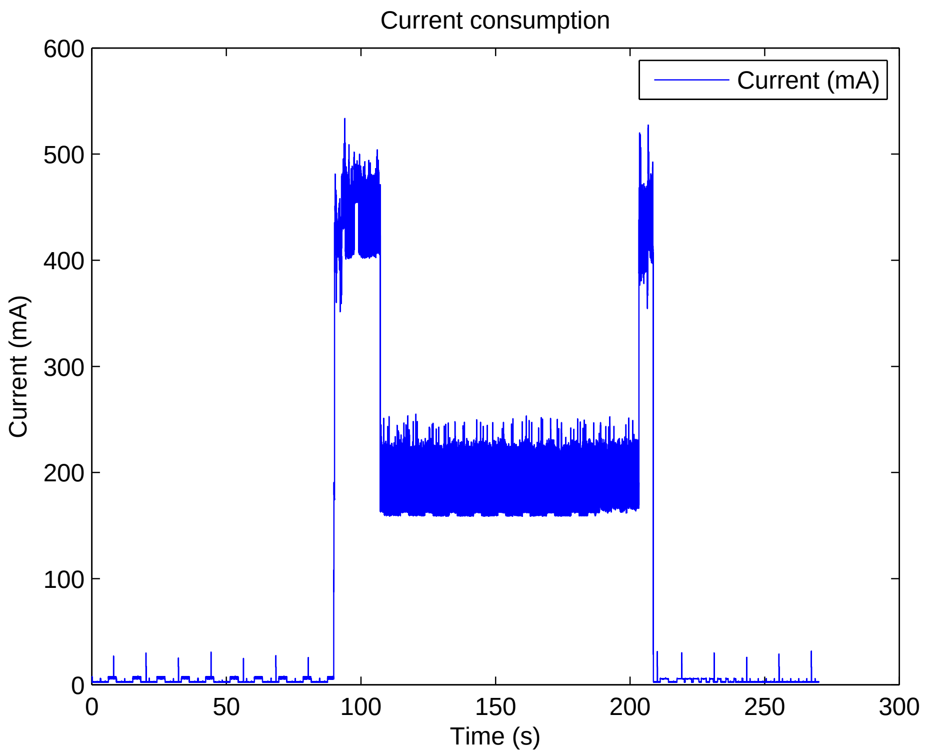

Figure 4-6
Current consumption of the CoreStation in full Duty-Cycle-Mode

5

Future Work

In this thesis a prototype GPS enabled WSN node was developed. A system to deploy will have to be much more power efficient. A possible future development towards a real- or near real-time monitoring system is difficult with the setup we have. This is especially because of the Bernese GPS software, which is a blackbox, and the correlation of the data from the different nodes. The processor on the node is not optimal from a low-power perspective. A dedicated hardware block, e.g. FGPA would be preferable for data processing. In the PSBacklog software it should be possible to schedule the plugins as jobs, now the plugins just start when PSBacklog is started.

6

Conclusion

In this project a system was built that is capable of monitoring movements of rock formations in alpine environments with low-cost GPS receivers. Differential GPS with carrier phase measurements was used to achieve an accuracy in the centimeter range. GPS sampling frequency and sampling time requirements are given. Further duty-cycling for power saving was evaluated. Duty-cycling was implemented for the wireless communication interface and for the entire system except the TinyNode. It was thus possible to reduce power consumption to 7% of the maximum performance consumption and still achieve the accuracy goals. The prototype sensor nodes work accurately, reliably, and substantial power savings have been achieved.

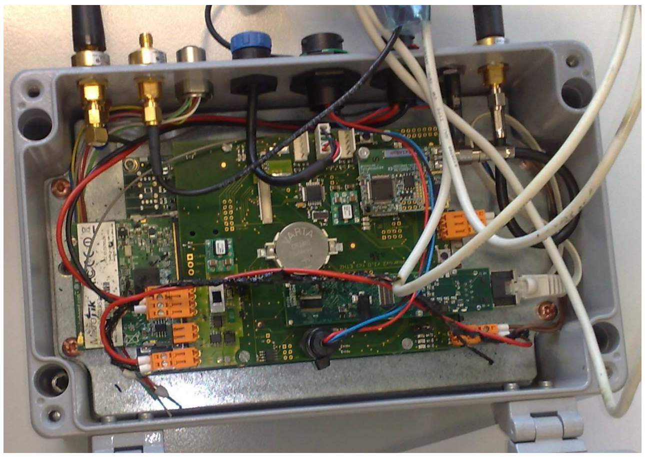

Figure 6-1
CoreStation after short circuit

This project yielded some interesting engineering challenges for me. I had to become familiar with the PermaSense project, understand the existing system and come up and realize ideas to advance the project. As usual in engineering, I also had to cope with set backs, like buggy software or faulty hardware (see Figure 6-1). Despite all the problems this project was really interesting and challenging, and an important part of my engineering education.

Bibliography

[1] Beran, T.; et al.: *High-Accuracy Point Positioning with Low-Cost GPS Receivers: How Good Can It Get?*,
University of New Brunswick, 2005

[2] Burri, N.; von Rickenbach, P.; Wattenhofer, R.: *Dozer: Ultra-Low Power Data Gathering in Sensor Networks*,
ETH Zürich, IPSN'07 2007

[3] Choy, S.L.: *An Investigation into the Accuracy of Single Frequency Precise Point Positioning (PPP)*,
RMIT, 2009

[4] Eso, R.A.: *Preliminary Results of a Low-Cost GPS-based Glacier Monitoring System: Trapridge Glacier, Yukon Territory, Canada*,
University of British Columbia, 2004

[5] Ge, L.; et al.: *GPS Seismometers with up to 20Hz Sampling Rate*,
University of New South Wales, 1999

[6] Handcock, R.N.; et al.: *Monitoring Animal Behaviour and Environmental Interactions Using Wireless Sensor Networks, GPS Collars and Satellite Remote Sensing*,
CSIRO, 2009

[7] Hunziker, J.: *Online GPS für Permafrost Monitoring*
`ftp://ftp.tik.ee.ethz.ch/pub/students/2009-HS/SA-2009-19.pdf`,
2009

[8] Limpach, P.; Grimm, D., *Rock glacier monitoring with low-cost GPS receivers*
in Abstract Volume 7th Swiss Geoscience Meeting, November 2009,
Neuchatel, Switzerland, 2009

[9] NAVCEN: *NAVSTAR GPS User Equipment Introduction*,
`http://www.navcen.uscg.gov/pubs/gps/gpsuser/gpsuser.pdf`, 1996

[10] Oetiker, T; et al.: *The Not So Short Introduction to LaTeX 2_ε*,
`http://tobi.oetiker.ch/lshort/lshort.pdf`, 2010

[11] Stoleru, R.; He, T.; Stankovic, J.A.: *Walking GPS: A Practical Solution for Localization in Manually Deployed Wireless Sensor Networks*, University of Virginia, LCN'04, 2004

[12] U-BLOX: *LEA 6 u-blox 6 GPS modules Data Sheet*, `http://www.u-blox.com/images/downloads/Product_Docs/LEA-6_DataSheet_%28GPS.G6-HW-09004%29.pdf`, 2010

[13] U-BLOX: *u-blox 6 Firmware Version 6.02 Release Notes* `http://www.u-blox.com/images/downloads/Product_Docs/u-blox6_Fw6.02_Release_Note%28GPS.G6-SW-10003%29.pdf`, 2009

[14] U-BLOX: *u-blox 5/6 Receiver Description (Module) including Protocol Specification* `http://www.u-blox.com/images/downloads/Product_Docs/u-blox5_u-blox6_RxDescrProtocolSpec_%28GPS-SW-09017%29.pdf`, 2010

[15] Zogg, J.M.: *GPS Compendium: Essentials of Satellite Navigation*, `http://www.u-blox.com/images/downloads/Product_Docs/GPS_Compendium%28GPS-X-02007%29.pdf`, 2009