Master Thesis

Design and Implementation of a Distributed Wireless Sensor Node Software for a Slope Movement Monitoring System

Stephan Gether, BSc

Institute for Technical Informatics
Graz University of Technology

Assessor: Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger

Advisor: Dipl.-Ing. Leander Bernd Hörmann, BSc.
Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger

Graz, March 2013

Kurzfassung

Drahtloses Sensornetzwerke (WSN) werden meistens für energieautarke Low-Power Anwendungen mit niedrigen Datenraten eingesetzt. Bei dem GeoWSN Projekt wird versucht, mit Hilfe eines WSNs Massenbewegungen zu erkennen und vorherzusagen. Da die Bewegungen mit GPS Receivern erkannt werden sollen, weichen die Anforderungen an das WSN von der Norm ab. Die Knoten sollen weiterhin energieautark funktionieren, der Energieverbrauch und der Datendurchsatz sind aber durch die Notwendigkeit eines kontinuierlichen Datenstroms von den GPS-Receivern zur Basis erhöht. Weiters muss auch eine hohe Ausfallsicherheit gewährleistet werden, damit die Positionsbestimmung zuverlässig funktioniert.

Diese Masterarbeit beschäftigt sich mit dem Design und der Implementierung einer Knotensoftware, die genau diesen Anforderungen entspricht. Hierfür wurde ein Netzwerk-Stack entwickelt der es ermöglicht große Datenmengen möglichst energieeffizient an die Basis zu übertragen. Dies wird durch ein intelligentes Multihop Routing Protokoll bewerkstelligt, das sich bei Netzausfall automatisch neu konfiguriert und nicht übertragene Daten gegebenenfalls erneut überträgt oder speichert.

Abstract

Wireless Sensor Networks (WSN) are usually used for energy-self-sufficient low-power applications with low data rates. The GeoWSN project is trying to recognize and predict a slope movement with a WSN. These movements should be detected by a GPS Receiver and so the requirements for the WSN differ to the norm. The energy usage and the data rates are increased by the need of a continuous data stream from the GPS receiver to the base station. This stream consists of GPS raw data and sensor data and is needed for an exact position determination. For these positioning algorithms, it is essential to get continuous and reliable data.

This master thesis addresses the design and implementation of a sensor node software that exactly matches these requirements. Therefore a network stack was developed, which allows energy-efficient high data rate transmissions of the GPS data to the base station. This is achieved through an intelligent multi hop routing protocol, that adapts itself to the given conditions, automatically retransmits lost packets and store data during a lost network connection.

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....

date (signature)

Acknowledgment

This master thesis would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study.

First and foremost i would like to thank my family, especially my parents for the support throughout all my studies. They have always supported me since my birth in all situations and gave me the financial means and the assistance to finish this study.

I also express my special gratitude to my girlfriend Katharina for her support and the patience in the last years.

This master thesis would not be possible without the callaboration with the consortium of the GeoWSN project consisting of Geolith Consult Hermann & Loizenbauer OEG, the Department of Emergency Management and National Defence of the federal state of Styria, the TeleConsult Austria GmbH and Graz University of Technology (Institute of Navigation and Institute for Technical Informatics).

I wishes to express my gratitude to my supervisors, Ass.Prof. Dipl.-Ing. Dr. techn. Christian Steger and Dipl.-Ing Leander Hörmann who was abundantly helpful and offered invaluable assistance, support and guidance. I also have to thank the whole Institute for Technical Informatics for the opportunity of this master thesis.

Graz, March 2013 Stephan Gether

Contents

List of Figures

List of Tables

List of Abbreviations

ack	Acknowledge
addr	Address
CCS	Code Composer Studio
CRC	Cyclic Redundancy Check
CSMA-CA	Carrier Sense Multiple Access/Collision Avoidance
DSP	Digital Signal Processing
FIFO	First In First Out Buffer
FPGA	Field Programmable Gate Array
GAF	Geographic Adaptiv Fidelity
GEAR	Geographic and Energy-Aware Routing Protocol
GeoWSN	Geologic Wireless Sensor Network
GNSS	Global Navigation Satellite System
GPRS	General Packet Radio Service
GPS	Global Positioning System
HWI	Hardware Interrupt
I2C	Inter-Integrated Circuit
IDE	Integrated Development Environment
IP	Internet Protocol
ISM	Industrial, Scientific and Medical Band
JTAG	Joint Test Action Group
LEACH	Low-Energy Adaptive Clustering Hierarchy Routing Protocol
MAC	Media Access Control

MECN	Minimum Energy Communication Network
MSG	Message
nbcast	Neighbor Broadcast
OR	Overlapped Regions
OS	Operating System
QoS	Quality of Service
RBF	Radio Basic Functions
ROM	Read Only Memory
RSSI	Received Signal Strength Indication
RTOS	Real Time Operating System
RX	Receive
SAR	Sequential Assignment Routing
SH	Sector Heads
SPI	Serial Peripheral Interface
SWI	Software Interrupt
TEEN	Threshold Sensitiv Energy Efficient Sensor Network
TX	Transmit
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network

Chapter 1

Introduction

Nowadays, the desire of a connected smart environment increases steadily. For this purpose techniques need to be created, which interact native between the environment and human beings, recognizes and responds automatically to events.

For environmental monitoring, Wireless Sensor Networks (WSNs) are generally accepted. These systems consist of some sensor nodes and at least one gateway node and are predestined for use in nature without infrastructure because they need no power grid or other wires. WSN nodes usually use energy harvesting for self-supply and a wireless channel for the communication between the nodes. Due to the fact that radio transmissions are limited in space, a multi hop routing is used for a range and coverage extension.

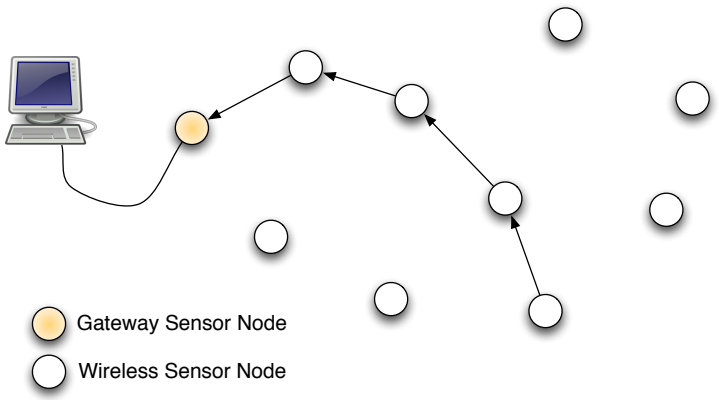


Figure 1.1: Typical Wireless Sensor Network.

This is shown in Figure 1.1. The sensor nodes transmit their messages not directly to the gateway node, but through other nodes in range. So it is possible to use low-power radio communication for long distances. Furthermore these nodes can be equipped with a lot

of sensors or actors. Thus, WSNs are very flexible and can be used for a wide range of applications.

1.1 Motivation

The GeoWSN project is motivated by the changing climate and the related geological events, which in recent years more and more frequently occurred; especially slope movements increased in Austria. Hence, the requirement of environmental monitoring is increasing. Until now, the most used monitoring system only can detect moving slopes, but they are not usable as early warning system. With the determined data of the GeoWSN, an exact model should be created, which allows predicting possible danger.

Also GPS equipped WSN systems are not common and thereby a lot of new design challenges arise. Especially the requirements on the network stack and the communication protocol, which are both part of this master thesis, make the development attractive.

1.2 GeoWSN

This master thesis is part of a national KIRAS project called GeoWSN[1], witch is a feasibility analysis of an early warning system for slope movements with low-cost GNSS (Global Navigation Satellite System) receivers. This should be realized by a WSN with a mutlihop routing protocol. Therefor a WSN node equipped with a GPS receiver and some sensors for measure temperature, humidity, acceleration and light are developed. The GPS raw data, the collected sensor data and the WSN status should be sent by the WSN to a base station, which consists of a node for communication and a server. All gathered data will be saved and analyzed on that server. More precisely this server has three main functions:

1. Storage of the raw data from the nodes
2. Calculation of the coordinates from the GPS raw data
3. Visualization of this data

The system overview is shown in Figure 1.2. The System is divided in different parts, which are developed by different master theses. The software design and implementation of the WSN nodes, is the part of this master thesis.

[1]Project grant GeoWSN (832344) by the Austrian Research Promotion Agency (FFG) under KIRAS PL3.3

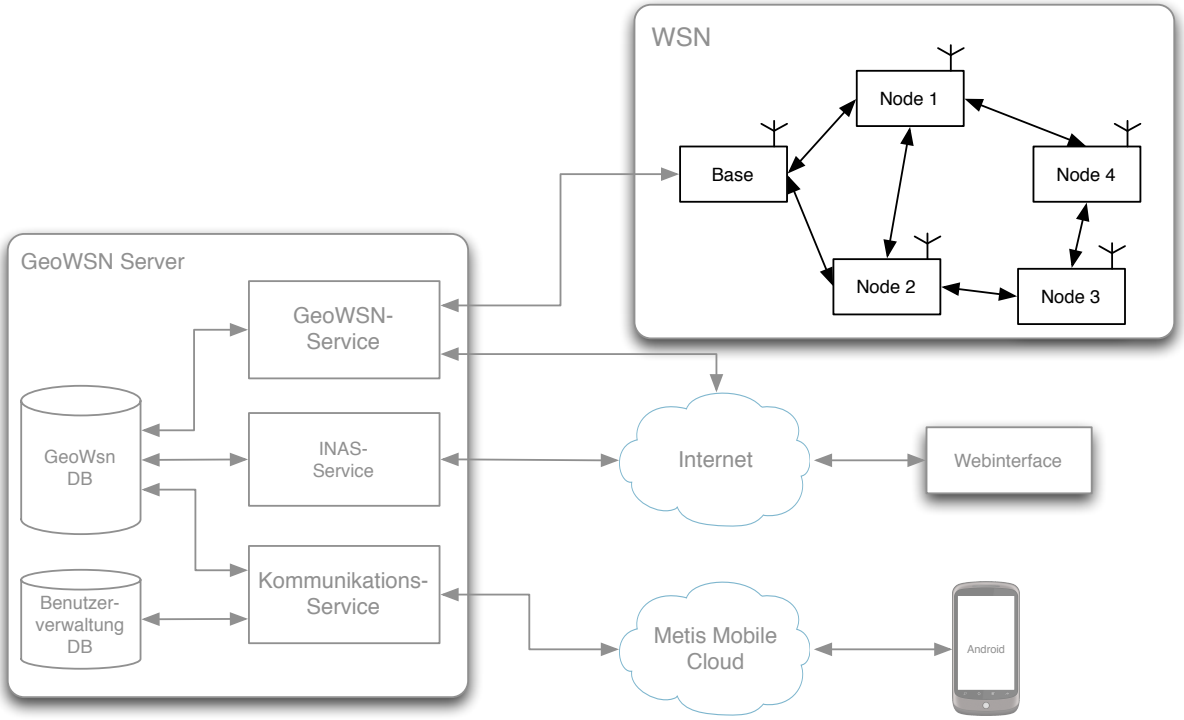


Figure 1.2: System overview GeoWSN.

1.3 Outline

This master thesis is about the design and implementation of the wireless sensor network node software for the GeoWSN project. As previously mentioned in Section 1.2, the detection of a slope movement should perceived by GPS positioning. But to get a proper position with a low-cost GNSS Receiver, it is necessary to implement algorithms like Real Time Kinematic (RTK) [18] or Precise Point Positioning (PPP) [9]. These algorithms need the GPS raw data from the receiver to compute a precise position. The used concept of these algorithms are described in [22].

These data have to be transmitted together with sensed data like humidity, temperature or acceleration to the base node and further to the GeoWSN server. The following requirements should be met from the developed software:

- Near Real Time: The measured data should be transmitted as fast as possible to the GeoWSN Server, so the positioning algorithm remains up to date.
- Fault Tolerant: No measured data sets should be lost. In case of an error, the data sets must be cached in a flash memory and retransmitted to the base if possible.

- Continuous Measurement: For the positioning it is necessary to get continuous data sets.
- Energy efficient: It is necessary for a self powered WSN to use a power aware routing protocol to increase the lifetime of the whole system.
- Multihop Routing: To extend the wireless range, a multi hop routing protocol must be implemented.

The implementation takes place on a self developed board, driven by a Texas Instruments MSP430 micro controller. A Microchip MRF24J40 radio module is used for the wireless communication. For the development the Texas Instruments Code Composer Studio is used.

In **Chapter 2**, the related work is listed. First, some existing slope monitoring systems are described and afterwards different basic approaches for WSN routing protocols with some examples are given. Another relevant part of the related work is mentioned below; high data rate WSNs.

The design of the GeoWSN node software is described in **Chapter 3**. First, the limitations of a WSN and then some operating systems are shown. Afterwards, a system overview with requirements and the architecture of the node is given.

In the implementation **Chapter 4** the basic description of the node software is shown. Therefor, the radio communication, the measurement and also the node controller unit are described with different charts and diagrams. Also, the detailed communication protocols as well as the class description are given in this chapter.

In **Chapter 5**, the results of the master thesis are discussed. Therefor, the system lifetime enhancement and the fault-tolerance will be illuminated.

Chapter 6 gives a conclusion and makes the completion oft the thesis.

Chapter 2

Related Work

This section deals with previously done work in WSN-based slope monitoring and routing protocols for WSNs. First, an overview of existing Slope Monitoring Systems is given in Section 2.1. Then, Section 2.2 shows advantages and disadvantages of different routing protocols. Finally, Section 2.3 deals with basic approaches for high performance (high data rate) WSNs.

2.1 Slope Monitoring

In the last years, an increase of slope movements has occurred around the world. Particular regions with much precipitation in monsoon seasons [2] are affected, but also mountainous countries [31] perceive an increase. Because of the increasing danger for property and life, a real time monitoring for slopes is necessary. The development of such systems can be divided in three main groups:

Nonautomatic: This was the beginning of geological monitoring. Someone had to control the sensors which were attached to the slopes.

Automatically centralized: One sensor like GPS receivers [7] or automatic tachymeter [31] observe the whole slope.

Automatically distributed: The newest approach is to use a wireless sensor network to observe a slope.

In this section, the automatically distributed systems will be discussed due to the relevance to this master thesis.

2.1.1 Wireless Sensor Network and Slope Monitoring

The most studies in this field follow the same principle. A cheap wireless sensor node is equipped with accelerometer, tiltmeter, pressure sensor or odometer to recognize the movement of the slope. Some have also sensors to measure some known triggers for landslides as for example the ground water level or rain volume. These nodes use a low power wireless communication like Zigbee with low data rates. The systems are most of the time in sleeping mode to save power and wait for an event. This can be a rain shower, a small movement or an earthquake. If an event occurs, the sensor node start to send the measured data to the base and increase the measure interval to do not miss a landslide. This strategy is usually sufficient, because not all landslides move abruptly [2] and it saves a lot of power.

An interesting example is described in [28], which measures the slope movement only with a 3-axes accelerometer. The system is shown in Figure 2.1 and consists of a WSN, a base station with a computing unit, a data server and a web service to inform and alert people. Each WSN node continuously collects and analyzes the sensed acceleration data taken from the monitored slope. For the evaluation, the 3-axes values are calculated to a tilt value, which serves as a trigger for the more accurate calculation of the surface displacement velocity. This second calculation need more effort and a higher sampling rate, but it is required for an accurate prediction of the slope failure time. So it is possible to alert people before a landslide happens.

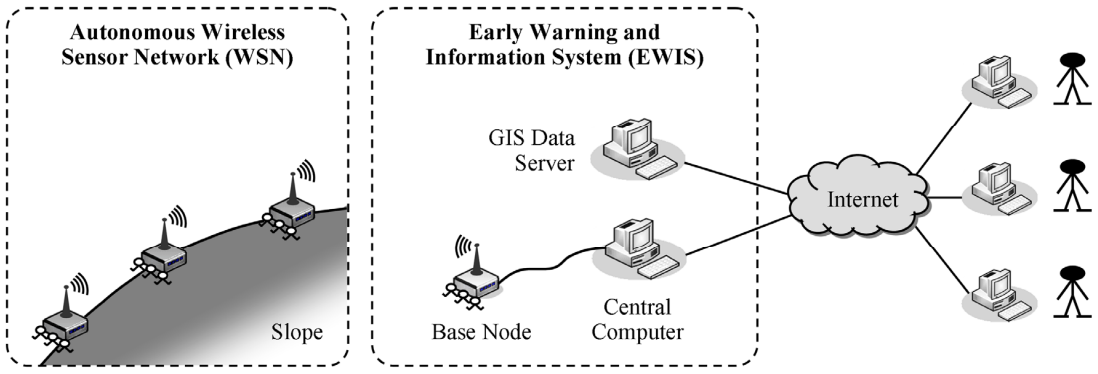


Figure 2.1: Overview slope monitoring system [28].

Another study [16] also uses tiltmeter, but in an other use case. Every WSN node has two tiltmeter and they are connected with a tension wire. These sensors are now placed so that they span a tension crack in the slope. So it is possible to recognize every movement

on the slope surface. The detected movement data is transmitted to a data log via a wireless communication network. The logged data is then send through GSM Network to the control room.

The last relevant study [2] implements only geophysical sensors. In other words, a soil moisture sensor, a pressure sensor and for the movement a flexible bend sensor. With the measured data sets it is possible to predict a slope failure accurately. These sensors are attached to a Crossbow MICAz Mote, a very common WSN node. This node communicates also within the 2.4 GHz ISM band, the same as Zigbee, and it is driven by a low power ATmega 128L processor. This node uses a real time operating system to run communication and sensor application simultaneously.
The prototype of the system was tested on a slope model to generate a data model chart. Based on these observations, different warning levels could assign to different sensor levels. These levels are used to enable appropriate actions to prevent human lives and properties.

2.1.2 GPS-based Slope Monitoring

GPS, Global Positioning System, is a satellite based positioning system. The accuracy of a low cost GPS receiver is too low for slope monitoring, because the occuring movements are less than one centimeter. Second level GPS receivers with a more suitable accuracy are very expensive. For that reason not so much slope monitoring systems with GPS are implemented yet. Another bad issue is the power consumption of the receiver and so the effort of an energy self-sufficient system is larger. In order to be able to build an economic GPS based slope monitoring system, there are a two approaches to overcome this challenge:

- Reduce the number of expensive components like GPS receiver.
- Use low-cost components in combination with several positioning algorithms for a better accuracy.

The first is used in a system described in [7]. The system is called "Application of Multi-antenna GPS Technology in Monitoring Stability of Slope" and consists of data receiving equipment, data processing units and analysis and management software, shown in Figure 2.2. The main concept is to use only on GPS receiving and a so called GPS multi-antenna control unit. This component has multiple antenna inputs and one output to the receiver. So it is possible to log the GPS signals from each antenna sequentially into the receiver. The time interval for every logging is configureable and depends on the application.

The maximum number of antennas are only limited through the antenna multiplexer. In this study 8 antennas are possible. These antennas are connected by good quality antenna cables and thereby distances up to 120 m between receiver and antenna are possible. With the use of signal amplifiers, which should compensate the signal loss, up to 300 meters are possible.

To get a better accuracy a differential GPS algorithm is used. For this a reference station with a known position is needed to reduce the atmospheric influences. Several tests on different terrains show that an accuracy of about 2 mm can be achieved under typical slope site conditions. So this study shows a possibility for an economical observation of a slope by a GPS driven system.

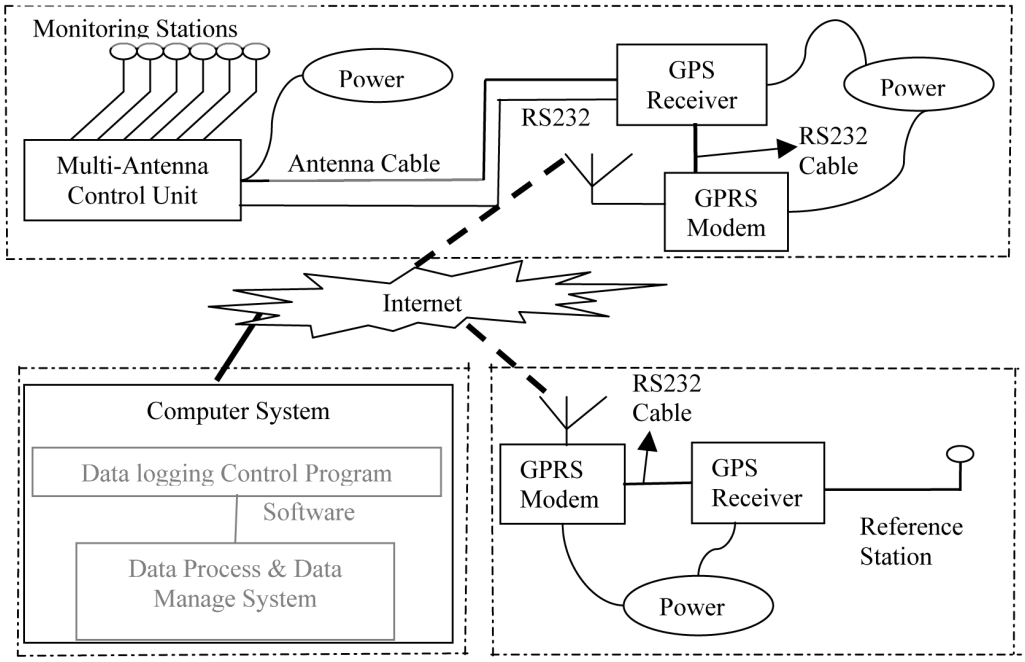


Figure 2.2: Overview multi-antenna GPS slope monitoring system [7].

Another GPS based system is developed by the ETH Zürich [3] and is further designed as WSN. They use a low cost wireless sensor node equipped with a low cost L1 GPS receiver.

Basically, the system shown before and the actual one are similar, except of the GPS data processing. The first one uses one receiver and a cable based analog signal and the second one use one GPS receiver per antenna and node, and a wireless digital link. Thus, the second design is more flexible than the first, but the data flow is more complex and the costs are higher. The accuracy is also in a sub centimeter range by using a differential GPS technology. A disadvantage is the power supply of the system, because there are only batteries used and the lifetime of the system is calculated for about 60 days. This

restricts the common use, but it shows the possibility of a GPS based WSN with low cost components for slope monitoring.

All in all, WSN is excellent suitable for environmental monitoring, particularly for slope movements trough the flexibility and expandability. Further it has been shown, that it is possible to build a system consisting of low-cost GPS components that have an accuracy that is sufficient for observing slow moving landslides.

2.2 Wireless Sensor Network Routing Protocols

Ever since distributed wireless sensors have been invented, there is a requirement of a communication algorithm that supports addressing and data transmission from a sensor (producer) to a base (sink). This is the task of a routing protocol.

In [1] a deeper look to routing issues of WSNs is given. So the most common routing protocols and techniques are classified into four major categories. The next four subsections will give a short overview.

2.2.1 Data-centric Protocols

This routing protocol type is different to normal address based routing where addressable network members manage the routing in the network layer of the communication stack. In data-centric routing the sink sends requests to regions or nodes and wait for data. To get the required data, it is necessary to adopt attribute-based naming to specify the property of data. In this section, some of these protocols are shown.

Flooding is the simplest protocol, therefor only queries or data broadcasts have to be sent. Every node in range that receives a broadcast, works in the same way and sends it till the hop count expire or the message reaches the target. This protocol is very simple to implement, but very energy and time inefficient, because very much overhead and redundancy arises.

SPIN [12] uses high-level meta data to name information. The key feature of SPIN is to send this meta data with so called ADV messages to each neighbor. This advertisement is now evaluated and if the data are needed, a requirement message is sent to the node. After the sink gets the data, a new ADV message is sent to each neighbor.

It works, as mentioned before, without addressing, every node knows only the data

it needs. This advertisement mechanism cannot guarantee the delivery of data, because if the sink and the producer are far away and the nodes between them are not interested in the data, the data will not be transmitted. SPIN is not a good choice for systems which need a high transmission safety.

Direct Diffusion, Energy-aware: Direct Diffusion [15] is one of the first data-centric routing algorithms. The idea is to diffuse the data through the network by using a key-value pair for naming. If a sink needs data from a node, a query with key-value pairs is generated. The query can contain locations, names, intervals, durations and several gradients. The sink now broadcasts this query and every node that receives the broadcast saves the key-value pairs for later use and broadcasts the query again. If the current node has the required data, the data will directly transmitted to the sink. This is achieved by the saved gradients from the query, which describe the direct route back to the sink. The functionality is also shown in Figure 2.3. Through direct diffusion, it is possible to save a lot of energy because of less overhead and redundancy. Moreover this routing protocol is - in contrary to SPIN - suitable for applications that need a high transmission reliability.

The energy-aware routing protocol described in [37] is a modified Direct Diffusion protocol. This algorithm uses an other path reinforcement algorithm. Not only the shortest path and maximum data rate is included into the algorithm, also the energy balance over the network and the communication costs are used for route calculation. The advantage of this procedure is, that not all messages transmitted through the shortest path and the lifetime of the whole network will increase. There are also other variations of the Direct Diffusion protocol. One is called Rumor routing [5], where the back gradient is selected randomly. Another modification is the Gradient-based routing, where, in comparison to Direct Diffusion, each node saves the gradients. So

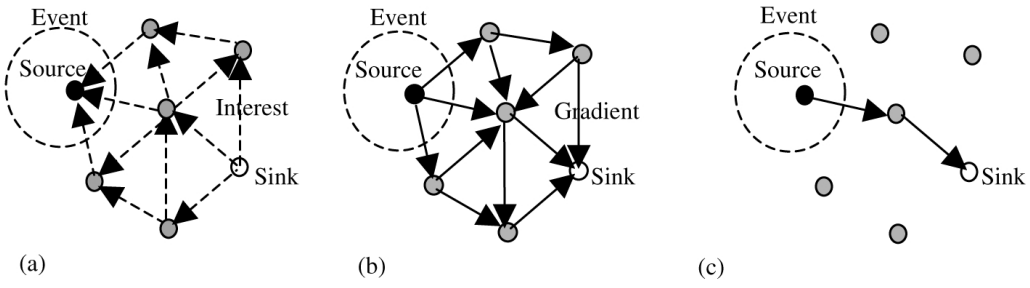


Figure 2.3: Direct Diffusion: (a) interest broadcast, (b) find shortest path with gradients, (c) transmit data through direct path [1].

every node knows every time the shortest way to the sink.

ACQUIRE [27] is a mechanism for efficient querying in a sensor network. With this mechanism, a sink sends an active query into the WSN. Each node that gets the query, tries to partially resolve the query with their local information. The query is forwarding as long as it is resolved and a complete response can be sent directly to the sink. Thus, it is possible to get complex data sets provided by many nodes with only one query. The network can therefore be seen as a distributed database.

2.2.2 Hierarchical Protocols

The main aim of these protocols is to split a network in smaller clusters and perform a data fusion to reduce the amount of data and hops, to save energy. Each cluster determines a head, typically the node with the highest energy level, which will cache the data, reduce the redundancy and send the sets to the sink. This protocol is used for example for smart buildings. Every room corresponds to a cluster and the communication between two rooms only takes place over the cluster heads.

LEACH is a low-energy adaptive clustering hierarchy routing protocol and is described in [11]. It is a very common used protocol and builds the cluster by the received signal strength. A head is established randomly to balance the energy level of all nodes. The head operates as gateway to the other clusters and the sink. So only one node needs to communicate over a long distance, which will need a lot of power. The optimal count of heads should be 5% of the total count of nodes. LEACH achieves with this clustering a reduction of energy consumption of the whole network over a factor of 7 compared to direct transmission.

PEGASIS is an improvement of the LEACH protocol [21]. The cluster-building algorithm is changed, so a node sends information only to its nearest neighbor. A cluster looks like a chain of nodes and the gathered sensor information are transmitted from node to node. One node takes over the head of the chain and sends, same as in LEACH, the data to the sink. This is shown in Figure 2.4. Node C_0 sends the own data to C_1, C_1 aggregates the received data with its own and forwards them to C_2. Then node C_2 sends the token to C_3 and further to C_4, then C_4 returns the data to C_3 and C_3 aggregates the data. Last C_3 sends the data to C_2 and this node transmits the whole merged data to the base node. This modification has an energy consumption improvement of about 100% to 300% and is possible through smaller overhead and fewer management messages as in LEACH. The disadvantages are a higher delay contingent on the multi hop routing in a chain and the bottleneck problem of the single head.

$$C_0 \rightarrow C_1 \rightarrow C_2 \rightleftarrows C_3 \rightleftarrows C_4$$
$$\downarrow$$
Base Station

Figure 2.4: Chaining in PEGASIS, redrawn from [21].

TEEN is a Threshold sensitive Energy Efficient sensor Network protocol [23]. The functionality is basically the same as in LEACH. First, the network will be grouped in clusters by distances and then one head will be chosen. This will be done in multiple levels, depending on the distances between the clusters and is pictured in Figure 2.5. But different from LEACH, the head sends thresholds to the clusters, a hard and a soft threshold for sensed attributes. Until the hard threshold is not reached, the node is in sleep state. As soon as the hard threshold is reached, the node starts sending the sensed information to the head and the head forwards the data to the sink. Afterwards, new data will only be sent, if the new value differs more than the soft threshold pretends from the old value. Thus, the transmitted messages and the energy consumption are reduced by longer sleep intervals. Simultaneously, the network remains active through the never sleeping heads. So it is preferred for monitoring systems, but not well for systems where continuous information are needed.
An extension of TEEN is APTEEN, described in [24]. Thereby AP stands for Adaptive and it means that it not only supports threshold triggered events like TEEN,

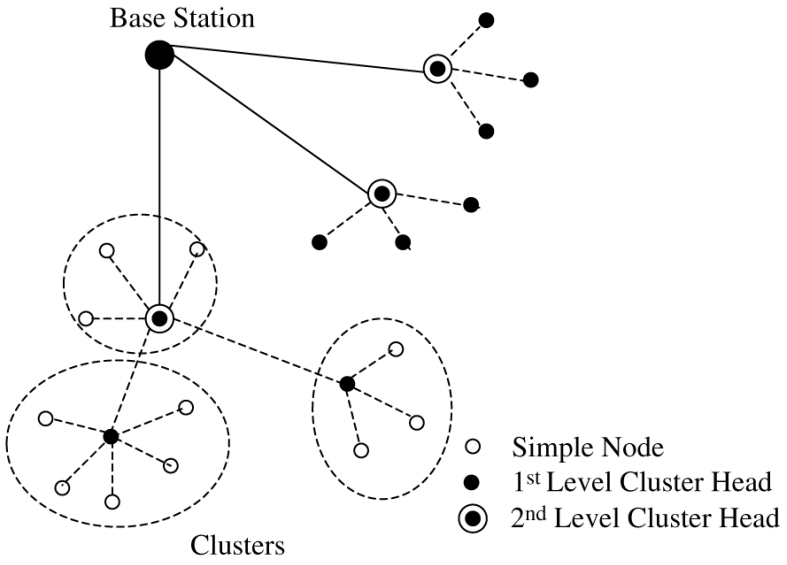


Figure 2.5: Hierarchical cluster in teen inherited [1].

but also the continuous information gathering. TEEN has the best performance in network lifetime and energy consumption. The performance of APTEEN is still good and lines up between TEEN and LEACH.

2.2.3 Location-based Protocols

Wireless sensor networks normally include a power-aware routing protocol. Thus, the most protocols need to know location information of the nodes to determine the distances and furthermore the power consumption. Location-based protocols go one step further and address a region by their location. So it is possible to send a query to a selected location and to get the requested information. Also, a flooding-based route discovery is not required and saves energy. The following protocols described are all power-aware:

MECN Minimum Energy Communication Network is described in [20] and is originally designed for addressed networks like IP. The aim of the protocol is to determine the minimum power topology for a network. Therefor, for every node a relay region is set. This is a region around a node which includes all neighbor nodes, through which transmitting is more energy efficient than direct transmitting. For a calculation, the two-dimensional location of each node needs to be known. This algorithm is also used by moving nodes, therefor the positionings have to be updated for every node by GPS.

GAF means Geographic Adaptive Fidelity [32] and is primarily designed for ad-hoc networks, but it can be adapted to wireless sensor networks because it implements an energy-aware location based routing algorithm. The basic idea of GAF is to place a virtual grid on the network. As shown in Figure 2.6, every node needs to associate itself to one point of the grid. If more than one node belongs to one point, it is redundant and only the nearest node is staying active, the other nodes have to enter

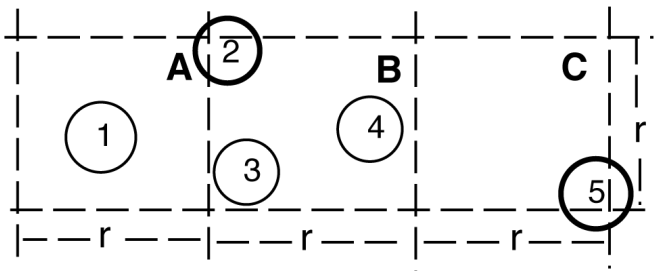


Figure 2.6: Example of virtual grid in GAF [1].

sleep mode. Thus, the whole network still remains active and all needless nodes can be shut down. An adaption of the GAF is described in [35] and named GAF&Co. Therefor, the virtual grids of GAF are replaced by hierarchical hexagonal cells to reduce local minimums in WSNs.

GEAR is a Geographic and Energy-Aware Routing protocol and is described in [36]. The main aim of the algorithm is to reduce the broadcasts of data-centered routing - like direct diffusion - by restricting the region. In contrast to direct diffusion the number of interests and the energy costs involved can be reduced significantly.

2.2.4 QoS-aware Protocols

These protocols use additional Quality of Service indicators like end-to-end delay or link reliability. In this section, a few of these protocols will be presented.

Maximum lifetime energy routing is explained in [4]. The main aim of this kind of routing is to find the maximum lifetime of the network. So a transmit path is selected by the lowest link cost. These are calculated in this study with two different functions:

$$c_{ij} = \frac{1}{E_i - e_{ij}} \qquad \text{and} \qquad c_{ij} = \frac{e_{ij}}{E_i}$$

where e_{ij} is the energy needed for transmitting a message from node i to node j and E_i is the energy level of node i. The routing path is now determined by the minimum cost path with Bellman-Fords shortest path algorithm.

By a simulation, a comparison with the minimum transmitted energy (MTE) algorithm was done and both cost functions have a higher lifetime of the network as MTE. This algorithm can easily be changed to the minimum cost forwarding protocol [33] by adding other factors, like end-to-end delay or throughput to the cost function.

Sequential Assignment Routing (SAR) which is mentioned in [29], uses QoS parameter in the routing algorithm to find a path with a high fault tolerance and achieves energy efficiency. The basic idea is a routing table with multi path entries. So during the path finding time slot, a tree of links to every node will be created. To route a message from a node to the sink, the tree is used to find multiple paths. Then one of these is chosen according to energy resources, priority of the message and QoS on this path. The fault tolerance is given by an automatic path update, if a

failure occurs. Simulations have shown that SAR is more energy efficient than routing protocols which only use a minimum-energy metric and don't include a packet priority.

SPEED is a QoS routing algorithm that provides a soft real-time guarantee and is described in [10]. It uses geographic forwarding to find a routing path. Therefor, neighborhood information of every node is required. Furthermore, the packet minimum speed of every link is determinated. So it is possible to estimate the end-to-end delay by dividing the distance through the speed of every needed link. Thus, a path can be chosen, which fulfills the real-time requirements.

2.3 High Data Rates in Wireless Sensor Networks

Now, it is common to use ultra low power transceivers in WSNs for wireless communication, because of the limited resources. But also the bandwidth and transmission range are decision criterias for a transceiver. So every application needs to find the best tradeoff between performance and power consumption.

But not only transceivers are the communication bottleneck in a wireless sensor network, equally the routing protocol and the network topology are important for a good performance. For example, the end-to-end delay from a node to the sink increases with every node and hop. So the routing protocol is very important to find the optimal path trough the network. Besides, an intelligent routing maybe allows simultaneous transmission over several paths or to open a data stream for big data packages. Hence, it is important to estimate the required data throughput first. Then, the maximum throughput of an end-to-end link has to be calculated. With this information, it is possible to design a WSN exactly for the requirements of an application.

Now to achieve a higher throughput, one or both of the following things have to be upgraded:

Hardware: That means a higher-performance transceiver hardware with higher data rate, most achieved through a higher modulation [25] [13] or the use of more than one transceiver [17].

Software: An intelligent routing protocol [30] is able to use the capacity of the whole network and to improve the end-to-end delay.

Through these improvements, a higher data rate and thus a lot of new applications like audio or video streaming [26] are possible. These are shown in the next subsections.

2.3.1 High Data Rate Transceiver

Since the demand for low power consumption and high data rate transceivers increases, a lot of research work is done. A lot of these chips are now available and it is not surprising that a variety of applications have been developed already.

In [13], a few high data rate radio interfaces are shown. Also an application is mentioned, a WSN should replace the complex cabling of an inflight entertainment system in an airplane. Thereby, the Smart Video Display Unit, shown in Figure 2.7, the Passenger Control Unit and the Seat electronic box have to interact with each other. The problem with this is, that the display and the control unit are not in the same seat. With a wireless link between these units, the cabling can be simplified. If the transmission only handles audio and control commands, a data rate of about 3 Mbits/s would be sufficient. But this is still much higher than Zigbee or other standards provides. Further, if also video content should transmit over the radio link, the needed data rate is abundantly higher.

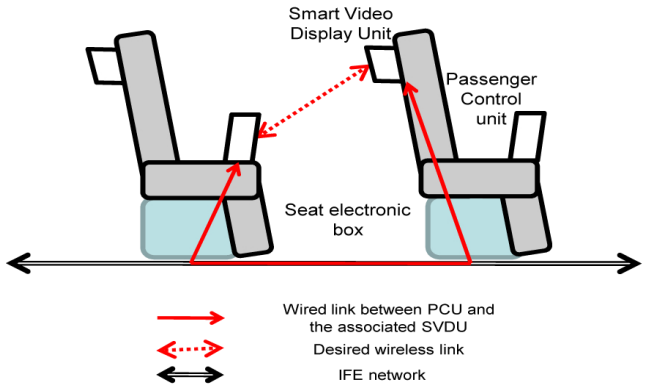


Figure 2.7: In flight entertainment system with WSN [13].

So if this application is being realized, a new transceiver with higher data rate has to be developed. In the paper, transceivers with different modulations are implemented on an FPGA. According to the paper, the Frequency Division Multiple Access is the best modulation scheme for WSNs, because of the low energy consumption in comparison to others.

Another relevant approach to increase the data rate of a WSN is described in [17]. Thereby, a multi radio platform, shown in Figure 2.8, is implemented. A FPGA is used as flexible controller for the four transmitters, which are connected on the four sides of the platform. Every receiver has a data rate of 1 Mbits/s and 83 frequency channels. So it is possible to implement a grid, where every node has a connection to its four direct neighbors by using unique radio channels. This is shown in Figure 2.9. Thus, the node can communicate simultaneously with several nodes, so it is possible to implement an ultra low

latency routing, or optimize the routing for high data rates. The use of multiple radios close to each other brings also disadvantages. For example the antenna design has to be accurate to avoid inferences between the modules. Also, a multiprocessor architecture was implemented on the FPGA. Here, for every radio a processor was responsible. So, a direct message forwarding with ultra low latency is possible.

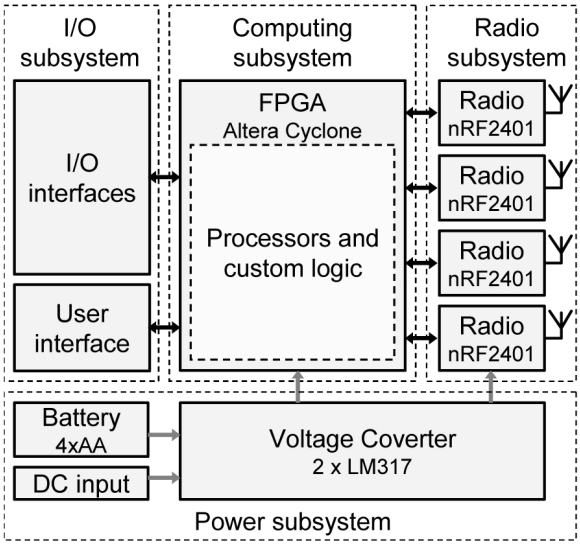


Figure 2.8: System overview of the multi radio platform [17].

For testing the platform, two different routing algorithms have been implemented. These are described in the following section:

Ultra low latency WSN routing have the aim to minimize the delay for an end-to-end multi hop path. In normal WSNs, a hop delay is very significant in comparison to the transmission time. Hence, a high performance multi hop WSN is very difficult to realize. In Figure 2.9 can be seen that if every link has a unique radio channel, simultaneous receiving and sending is possible. After the header has been received, the message can be forwarded. So the hop delay is only the receiving time of the header. Thus, only the header length is decisive for the hop delay.

High data rate routing is an extension to the ultra low latency routing. Thereby, like demonstrated in Figure 2.9, one data package is divided in four paths (the black thick arrows), and so a maximal data rate of 3,3 Mbits/s is reachable [17]. Therefore the source and destination have to use all their radios simultaneously, the intermediate nodes react in the same way as it is described in ultra low latency routing with two radio modules.

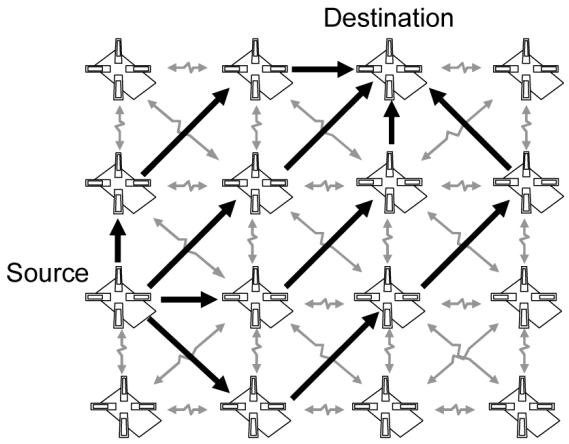


Figure 2.9: Mesh network topology [17].

This study is innovative for a fixed WSN, but for a self-reconfigurable system it is limited in application, because the routing with four radios and multiple frequencies is very complex. Every node has to know its location, the neighbors locations and the frequencies which are used in its wireless range. Also the hardware is complex, expensive and furthermore, the power consumption from four radios is significantly higher.

2.3.2 Intelligent Multi Path Routing for High Performance WSNs

As mentioned before, a lot of future applications for WSNs need a higher data rate. In large wireless sensor networks, it is possible to use a multi path routing algorithm. In [30], a WSN based high data rate streaming with multi path routing is shown. Thereby, one data package is divided into smaller messages, which send simultaneously data on several paths to the sink. So, the data throughput of the network approximately increases by the number of paths. But also problems comes with this algorithms:

- Interference range is about twice the communication range.
- Location of source and destination node has to be known.
- Every node needs to know every route to avoid collisions.

To master the greatest problem, an algorithm called “Interference-Zone Marking” is used. Thereby, the network is divided in sectors shown in Figure 2.10. First, the Sector Heads (SH) along the main path from source to destination have to be determined. Secondary, every SH node send a Broadcast to get the Overlapped Regions (OR) between the sectors and to find possible secondary paths. Then, if all paths are found, the source starts with sending the packet over the main path. If the first packet leaves the interference sector, the

source transmits the second packet over a secondary path, and so on. Thus it is possible to get a fault tolerant and high data rate WSN for applications like FTP.

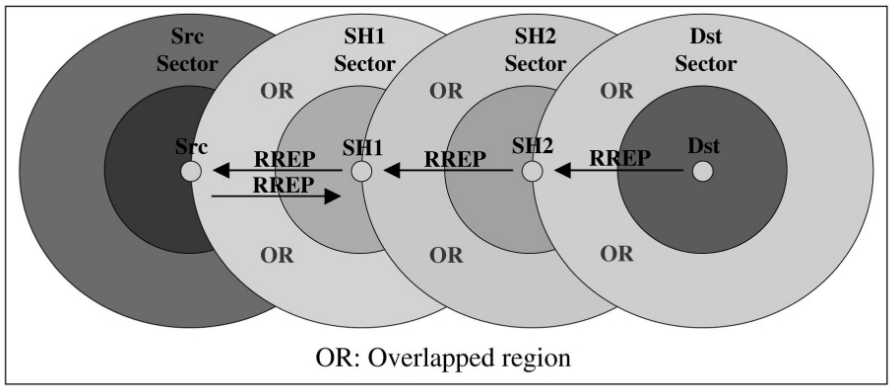


Figure 2.10: Interference sector marking [30].

2.4 Summary

For this Chapter, a lot of related work has been analyzed. First, some slope monitoring systems are discussed. They are distinct in different classes. Then, wireless sensor network based monitoring systems are shown. Also, GPS based systems are mentioned, but it was found, that GPS equipped WSN based systems are not often used. The project from Section 2.1.2 shows that it is possible to get the needed accuracy for slope monitoring with more then one geographic distant GPS antennas and/or receivers.

Then, a lot of routing protocols for WSNs are discussed, but a proper routing protocol for this project hasn't been found. Also some relevant approaches are shown, which went down more or less in the developed routing protocol. Thus, the number of node is low, the location based and hierarchical algorithms are rather not feasible. But especially the QoS and energy-aware algorithms were relevant for this project.

Also high data rate WSNs are discussed. It was found, that it is nowadays not realizable to develop a real high data rate WSN with common transceiver and passable costs. But for the data rates of the GeoWSN a normal 2.4 GHz transceiver seems be enough. Also, with an intelligent routing protocol and low overhead transmissions it is possible to increase the data throughput of a WSN.

Chapter 3

Design

This section is about the design of the WSN node and the resulting challenges. First in Section 3.1 the problems, which can happen by designing a WSN system are mentioned. After this in Section 3.2 the used operating system is compared to two others and the reason for choosing SYS/BIOS is shown. Finally, Section 3.3 gives a short overview of the requirements of the system, the designed architecture of the software and the developed routing protocol.

3.1 Limitations of a WSN

Apart from the standard issues from wireless systems like energy supply and radio channel failures, WSNs have due their topology additional restrictions. In this section some major problems are mentioned which have to consider in the design of a WSN.

3.1.1 Energy Hole Problem

Due self powered wireless nodes have limited energy resources, energy efficiency is one big goal in designing a Wireless Sensor Network. So the energy save mechanisms pervade all layers, from the physical to the application layer. In ad-hoc wireless networks it is sufficient to consider every individual node to maximize the network lifetime. Due the multihop network topology of a wireless sensor network, the lifetime of the network depends on the whole network. Lifetime in a sensor network generally means the period of time in which the network can maintain the required functionality. So a WSN who exceeds this period do not fulfill the task. This is unacceptable for an early warning system.

To understand how the lifetime of a WSN is composed, the topology of the network has to be analyzed. Due to multihop routing, a message is passed through the network from node to node until the sink is reached. So it is possible to divide a network in hop rings

arranged around the sink node. This is shown in Figure 3.1. Thereby, a ring conforms the wireless range of a node. So a direct transmission only inside ring 0 is possible. From ring 1 the sink can be reached with one hop in ring 0, from ring 2 with two hops and so on and so forth. Thus, the relayed traffic in nodes close by the sink is much higher than the traffic at nodes further away.

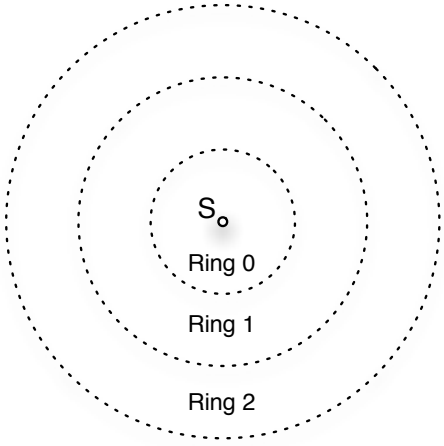


Figure 3.1: Energy hole around the sink node.

The traffic load of a ring can be calculated by the following formulas. Thereby, ring 0 have to overcome the whole network traffic, ring 1 only have to menage the traffic outside of ring 0 and so on and so forth.

$$Load_{ring0} = \frac{total\ traffic\ in\ the\ network}{num\ of\ nodes\ in\ ring\ 0} \tag{3.1}$$

$$Load_{ring1} = \frac{total\ traffic\ from\ outside\ ring\ 0}{num\ of\ nodes\ in\ ring\ 1} \tag{3.2}$$

From that can be deduced that the traffic in a normal distributed network near by the sink is about three times higher [19] than in the outer rings. As well the energy consumption of a WSN node is closely linked to the wireless network traffic and thereby the nodes in ring 0 have a higher power consumption. If one or more nodes from ring 0 runs out of energy the whole network is not longer functional and an energy hole occur.

So it is necessary to avoid energy holes for maximizing the WSN lifetime. Therefor an intelligent routing protocol has to analyze the energy level of each node and recalculate the routes if necessary.

3.1.2 Local Minimum Problem

The local minimum problem [34] is primarily caused by wrong deployment of sensor nodes. If so called routing holes as shown in Figure 3.2 arises, a local minimum occurs. That means that from Node A no neighbor is closer to the sink than A itself, but A has no direct connection to the sink. By using a geographic routing like greedy forwarding, packets get stuck at node A and do not get forwarded to the sink. So it is necessary to use a routing algorithm which recognizes such routing holes to avoid a Local Minimum. Also by properly placed nodes without routing holes, the risk of a local minimum decrease.

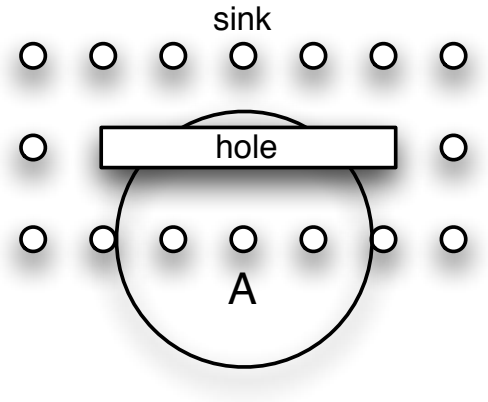


Figure 3.2: Local minimum problem.

3.1.3 End-to-End Delay

Another problem by designing a WSN is the end-to-end delay of a message caused by the multihop topology of a WSN. A simplified composition to estimate the end-to-end delay is shown in Figure 3.3 and Formula 3.3.

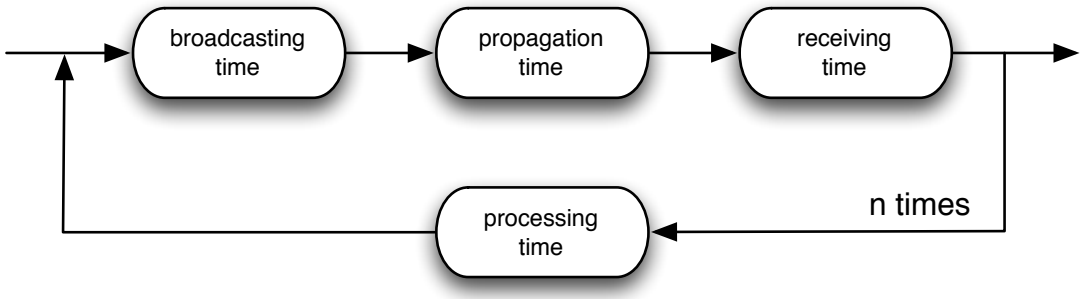


Figure 3.3: Composition of end-to-end delay.

$$t_{delay} = (t_{broadcast} + t_{propagation} + t_{receiving}) * n + t_{processing} * (n-1) \quad (3.3)$$
$$where\ n = number\ of\ hops$$

Thereby, the propagation time compared to broadcast and receiving time is very short. Propagation time means the time that an electromagnetic wave needs from the sender to the receiver. The broadcast and receive time is strongly coupled with the modulation of the wireless channel and the message size. The processing time depends on how deep the multihop protocol is integrated in the network stack and the computational power. Furthermore, the end-to-end delay increases proportional to the number of hops, because every hop node has to process and forward the received message until the sink is reached.

This affects particularly on time-sensitive applications, because it is very complex to predict the message duration time. So a soft or even a hard real-time WSN is coupled with a lot of information overhead, which is needed for the estimation and the synchronization. For this reason it was decided to design only a near realtime WSN that matches the requirements for GPS based slope monitoring. Thereby, the overhead is reduced by an approximately estimation of the end-to-end delay, which should be sufficient for a large number of time-sensitive applications.

3.1.4 Hidden and Exposed Station Problem

These are well known general problems of wireless networks and do not only concern WSNs. The problem thereby is that the common media access method CSMA/CD not perfectly works because every station uses the same wireless channel. The hidden station problem is shown in Figure 3.4. It occurs if two nodes (A and C) simultaneously try to send a message to one node (B) and both sending nodes think the channel is free because

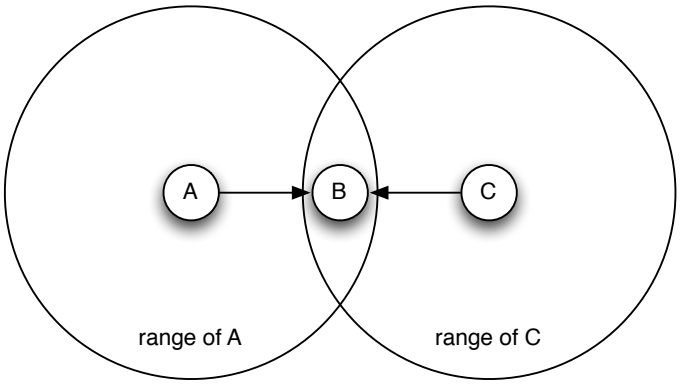


Figure 3.4: Hidden station problem.

they are not in range of each other. Thus, a collision occurs and one or both data messages were lost. So node (B) have to recognize this and have to inform the other nodes about this collision.

By the exposed station problem it is exactly the other way and it is shown in Figure 3.5. Two nodes (B and C) want to send a message to different nodes (A and D) but B and C can't send simultaneously because one node thinks the other node occupies the channel. Since A is not in range from C and D is not in range from B the simultaneous transmission would be no problem.

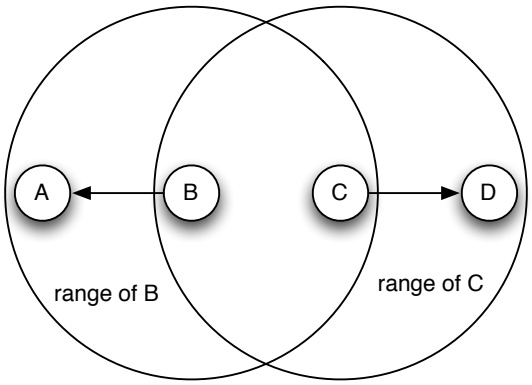


Figure 3.5: Exposed station problem.

These problems often happen during broadcasts, because of simultaneously forwarding. The problems are easily avoidable by preventing transmissions at the same time with slotted media access or other synchronization mechanisms and flow control signals like RTS/CTS. This mechanism is called CSMA/CA and is included in the IEEE 802.11 protocol.

3.2 Operating System

First, in this section, three Operating Systems (OSs) are presented and then a comparison shows which OS is most suitable for the GeoWSN nodes. All three OSs share that they evolved for microprocessors and their constrained resources. That means, they are useable for WSN note-class devices with about 10 to 100 kilobytes of code ROM and smaller than 20 kilobytes of RAM. Also efficient multitasking and a high flexibility are important features.

3.2.1 SYS/BIOS

This operating system[1] is developed by Texas Instruments, uses an advanced real-time kernel and is useable for a large range of DSPs, ARMs and micro controllers. SYS/BIOS is for free use in combination with microchips from Texas Instruments. The OS can be installed on a Windows computer and is automatically integrated in the Code Composer Studio. This development environment is described in more detail in Chapter 4.1.

Moreover it comes with an extensive logging, tracing and monitoring tools called XDCtools for debugging. This separated software component is also provided by Texas Instruments and it is the underlying tool needed by SYS/BIOS. It provides the graphical configuration window and sets up the OS for using with a specific device. Some features of SYS/BIOS are listed in the following:

- Fast multitasking performance
- Low latency for critical interrupts
- Dynamic memory management services offering both variable-sized and fixed-sized block allocation
- Easy configurable with graphical user interface
- Hardware and software interrupts
- Preemptive priority scheduling
- Up to 32 priority levels
- Multiple synchronisation mechanisms (Semaphore, FIFOs, Events, Gates, Queues...)
- Automatic power save mode

3.2.2 Contiki OS

Contiki OS is an open source operating system developed by Adam Dunkels at the Swedish Institute of Computer Science [8]. It is an OS special designed for Wireless Sensor Networks and it is runnable at systems with constrained resources. Additionally, it implements IP network connectivity for easily connect things to the Internet. The task scheduling is event based, but also threaded long running tasks are possible to run as kernel function. So the advantages of both systems are realized and have an effect on the performance.

[1] http://www.ti.com/tool/sysbios&DCMP=sysbios&HQS=Other+OT+sysbios, Feb 2013

Contiki is thereby delivered with an Ubuntu VMware image, which includes all needed software, a developing environment, debugging tools and even a simulator. So the installation and start-up with the system is relatively easy. A short overview of the features of Contiki are listened in the following:

- Event based kernel
- Preemptive multithreading task possible
- Over-the-air programming
- Highly portable
- Small code size
- Deep integrated communication support
- Direkt IP addressing

3.2.3 TinyOS

As well as Contiki, TinyOS is an operating system developed for WSNs. But it is not an OS in the traditional sense; it is more a programming framework, which generates an application specific OS by combining given components to a runnable binary. So a lot of functionality needed for a WSN application is already implemented in components. For example the MAC component includes the radio driver and protocols, so only high-level functions have to use in applications. Since the abstraction level is so high, the adaption effort to another platform is also high.

- Only event based
- Nonpreemptive scheduling
- Very low code size
- Abstract hardware with components
- Including radio MAC layer as component
- Programmed in nesC

3.2.4 Comparison of OSs

As shown in Table 3.1 Contiki or TinyOS are better solutions for standard WSN applications. But for the GeoWSN projects it is important to use a real-time system and have support for the specific hardware that was chosen. Also the network functionality included to TinyOS or Contiki could not be used because the special requirements, which are given, can't met. As well the seamless integration of SYS/BIOS to the Texas Instruments environment and also the low power functionality together with a MSP430 have simplified the decision to use SYS/BIOS as operating system. All in all for this project SYS/BIOS is the right operating system.

Table 3.1: RTOS comparison.

	SYS/BIOS	Contiki OS	TinyOS
Open source	no but its free to use with MSP430	yes	yes
Designed for WSN	no	yes	yes
Event-driven	no	yes	yes
Multitasking	yes	yes	yes
Preemptive	yes	yes	no
Realtime	yes	no	no
Power save mode	yes	yes	yes
Linked	static & dynamic	dynamic	static
Integrate networking	no	yes	yes
Support for MRF24J40	yes	yes	no
Portability	low	very high	high
Ported for MSP430	yes	yes	yes
Programming Language	C/C++	C	nesC

3.3 System Overview

This section gives a short overview of the software components of the wireless sensor network node used for the GeoWSN system. First the requirements on a GeoWSN node are mentioned and second the architecture of the designed software that matches those requirements is shown.

3.3.1 Requirements on a GeoWSN Node

At the beginning of the GeoWSN project some requirements for the WSN was defined to ensure the correct functionality of the GPS positioning and further of the slope monitoring and warning system.

Near Real Time: As mentioned before in Section 3.1.3, it involves considerable effort to develop a real-time wireless sensor network. Thus, the system should only pass a near real-time system. Therefore an end-to-end delay estimation algorithm is implemented to calculate the timing of the software.

Fault Tolerant: The system should be used as a social warning and monitoring system, so its the main goal to make the system failure-resistant. A complex system like this is susceptible for a lot of error sources. So it is important to recognize even at the design phase these potential errors and minimize those. Especially the wireless communication is prone to errors. So the network stack must support mechanisms to counteract this. Also the measurements should be error-free in order do not distort the data analysis.

Continuous Measurement: Also the measurement should be done periodically, because it is very important to remain the data consistent. The positioning algorithms need an ongoing data stream to achieve the required accuracy. If the data is lost for a long time the algorithm lost their accuracy and need a long time to get back to work properly.

Energy Efficient: Wireless devices have the downside that no power grid is permanent useable. So batteries or energy harvesting is necessary to guarantee a stable energy supply. Therefore the power consumption has to be minimized. It is very important to consider the energy issue deep into the design phase. Every decision made have to be a tradeoff between energy aware and functionality. Especially functions with high power consumption like wireless sending and receiving have to optimize in power consumption and usage.

Multihop Routing: The main goal of a wireless sensor network is to provide an independent, widespread wireless platform for information gathering. To achieve a higher coverage, a multihop routing protocol is used for wireless communication. The implemented protocol uses some parameters to calculate the best path through the network in order to deliver a message correctly.

These requirements are implemented in the node software which is shown in Figure 3.6 and is described in the next section.

3.3.2 Node Architecture

The node software consists of several parts and is shown in Figure 3.6. The main components are the radio and the measurement unit. These are responsible for the wireless communication and the information gathering. The software is split into tasks, which are

running on SysBIOS. The tasks are synchronized by semaphores and controlled by the node controller. The software also automatically adapts the functionality to the use as a base station or measurement node.

To get a near real-time behavior, the implemented application runs on a real-time operating system. Thus, it is possible to realize almost parallel tasks. These tasks are ordered and scheduled by priority. As shown in Figure 3.6 the Radiostack is driven by two tasks, to guarantee a fast radio communication response. Also the Measurement and the GPS Module have a separated task, because the GPS receiver works as a communication master and the system have to respond immediately.

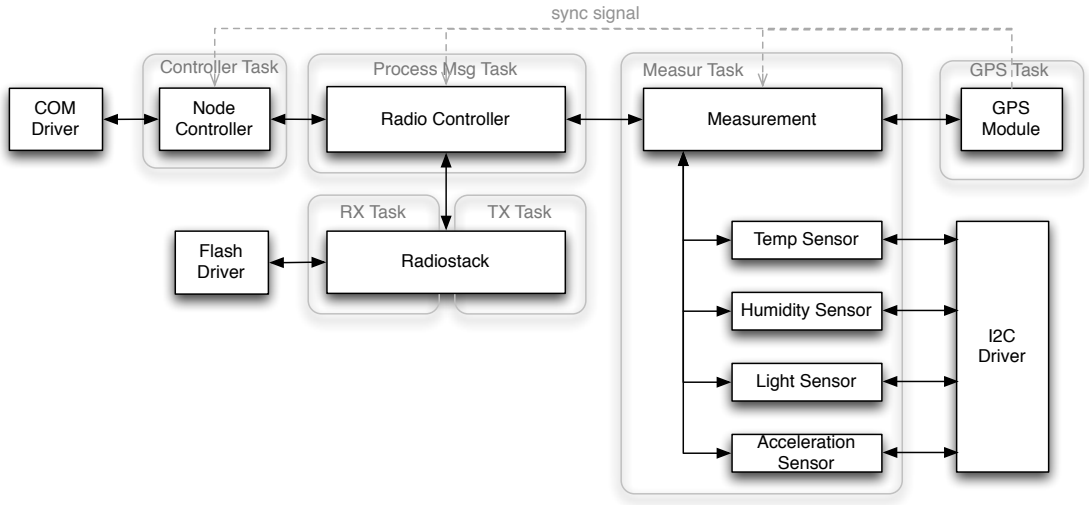


Figure 3.6: Functional node software architecture.

To minimize task synchronization failures and consequential wireless communication indifferences between nodes, the GPS sensor module provides a synchronization signal. This signal is generated by the atomic clock into the GPS satellites and has an accuracy of 1 millisecond. So all nodes are always synchronous and the probability of failures decreases. In the following the components are briefly described:

The **Controller** is, as the name describes, the control module of the node. It performs various functions depending on the use case. If the node is in use as base station, the controller additional starts the route discovery and manages the communication over the serial interface to the server. Otherwise, if it is in use as normal node, the controller only monitors the measurement and radio functions. The remaining modules are event driven and react on specified interrupts or events.

The **Radio Controller** is one of the event driven modules. It manages the received messages from other nodes and reacts with answer messages or events.

The **Radiostack** can be seen as a closed system which provides functions to send and receive messages. The whole network and routing functionality is hidden behind this module. So all the network layers are transparent and a direct connection between nodes is possible. To send a message only the destination address has to be known. Through the integration of a flash memory, messages who can't send through connection errors can be buffered and retransmitted later. The detailed description can be seen in Section 4.2.

The **Measurement** controls the continuous measurements and provides the measured data for processing. Clocked by the sync signal this module acquire information from the various sensors and buffer them until the data message is sent to the base station. A defined sensor interface facilitates the extensibility of the module and new sensors can be integrated easily by adding a new driver.

The **GPS Module** is the control unit of the GPS receiver. By a serial interface the GPS module is able to configure the receiver and receive the GPS raw data, which are essential for the positioning system. Then, the received data is buffered until the data message is transmitted to the base.

Also various **Sensors** are equipped on the node. So it's necessary to implement sensor drivers to communicate with them. Due all sensors are connected by one I2C bus, every sensor driver implements the I2C driver as deepest layer and provides a defined interface for communication.

Some additional **Drivers**, for example the seven segment display or the flash driver, are also implemented into the system to enable the entire functionality of the node hardware.

3.3.3 Routing Algorithm

In Chapter 2 a lot of routing algorithms are described, but none of those were directly useable for the GeoWSN. So a new routing protocol has to be designed, which meets all requirements. Therefor the following routing requirements have been defined:

1. Route discovery in one iteration.
2. Slightly control messages.
3. Longest possible network lifetime.
4. Short end-to-end delay.

To handle those needs, different mechanisms are merged to a routing protocol. At first a superframe is used for network coordination. The structure of the superframe is shown in Figure 3.7. The beacon normally is implemented as a network message to synchronize the nodes, but here the sync signal from the GPS receiver will be used. This is easier, more precise and decreases the network traffic. The whole superframe has a length of 2 seconds because of the 2 seconds transmission interval and is divided in short slots. First, during the GPS slot the GPS raw data is read from the receiver. After this, the base starts with sending so called nbcast (neighbor broadcast) to all its neighbors during the routing slot. These broadcasts contain the cost needed to send a MSG to the base. All nodes that received a nbcasts, transmit an own nbcast, but with new calculated costs. Thus, only one iteration is needed to build up the routing paths.

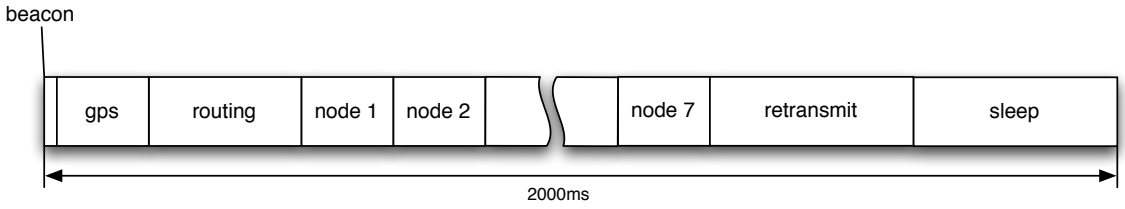


Figure 3.7: Superframe.

The next slots are for the nodes to transmit their data messages to the base. Every node has a fixed slot in which the MSG has to be transmitted. If it is not possible at that time, the message will be saved for a later retransmission. After the node slots follow the retransmission slot; in this time slot registered nodes can retransmit their saved messages. Thus, the needed control-messages are minimized, because of the simple route discovery and the slotted time management. Also the transmission errors are minimized, because the disturbance from other nodes is not possible.

A long network lifetime is achieved through the cost function, whereby a node chooses always the cheapest path to the base. For this cost calculation the energy level of the node and the RSSI between the linked nodes are used. The formula calculates the link cost between two nodes (i-j). This cost summarized, results in the path costs between the node and the base. Thus, a node with low energy will not used as hop node and the lifetime of the node and the whole network will be increased.

$$c_{ij} = \frac{1}{(E_j + rssi_{ij})} \tag{3.4}$$

Thereby E_j is the energy level of the receiving node and $rssi_{ij}$ is the received RSSI between node i and j. The RSSI level and the energy level are scaled before to get an appreciable cost value. The only disadvantage of this algorithm is, that every node only knows the way to the base station, but for the acknowledge message it is necessary to know the paths back to the nodes. So every data message memorizes the passed nodes and so the acknowledge can be sent back.

To get a short end-to-end delay the forwarding mechanism takes place on a low level layer of the network stack. Thus the forward time and also the end-to-end delay will be minimized.

3.3.4 Real-time Scheduling

SYS/BIOS is a priority based real-time scheduling operating system. That is necessary to meet the requirements on the software. As shown in Figure 3.8 there are different threads with different priority ranges, higher priority threads can interrupt lower priority threads.

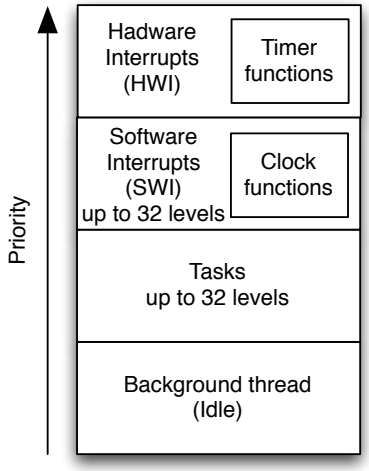


Figure 3.8: Thread priorities.

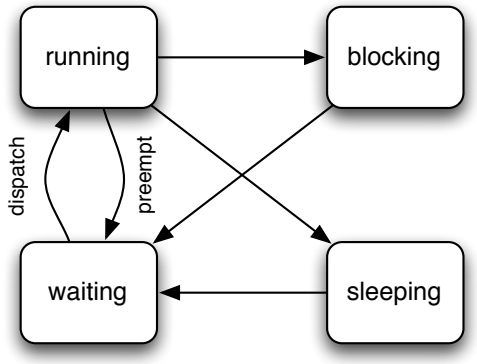


Figure 3.9: Thread states.

In this interrupt driven real-time scheduling technology, hardware interrupts have the highest priority and the SW interrupts the second highest. Those are followed by the tasks. Furthermore, the idle task has the lowest one. SWIs and tasks have additional 32 priority levels to choose. Thereby, it is difficult to assign the priorities to the tasks, because a compromise between thread switches, responsibility and efficiency have to be chosen. So the most time-sensitive tasks, like receiving from radio module, have to get the highest priority. Additionally, the program flow has to be considered, to reduce useless task switches. Figure 3.9 shows the possible thread states, so if tasks preempt each other, the execution time increases because of the thread switching overhead. The priority levels of the tasks are used for the GeoWSN node software as described in Table 3.2.

Table 3.2: Task priorities.

Task name	Priority	Stack size (byte)	Description
rxTask	6	512	Receive task
txTask	5	768	Transmssion task
mainTask	5	1024	Node controller task
gpsTask	4	1024	GPS receive task
measureTask	3	512	Task for sensor measurement
processMSGTask	2	768	Message processing task

3.4 Summary

The design of the node software has been discussed in this chapter. First, the limitations of a WSN and the associated design problems have been mentioned. Especially the energy hole problem and the end-to-end delay are important for this project. Afterwards three operating systems has been presented and discussed. Therefor, the operating systems have been compared by their major features and SYS/BIOS have been chosen, because of its close connection to the used micro controller.

Then all requirements on the GeoWSN have been mentioned. The system should be fault tolerant, near real-time and energy efficient. Further, a continuous measurement and a multihop communication should be guaranteed. The designed node architecture, which meets these requirements, has been described afterward. Therefor the main components and functions of the node software have been shown.

The routing algorithm has been a special part of the design phase, because a lot of possibilities have been considered. Using the “the simplest is the best” theme, a new power-aware algorithm has been designed, with small overhead and quick route discovery.

But not the routing alone, also the communication flow and protocols were kept simple by using a superframe based design.

Another design aspect has been discussed with the real-time scheduling. For the development of a real-time system it has been mentioned, that it is important to know the connections between threads. Hence, it has been possible to find right priorities for the different tasks to optimize the program flow.

Chapter 4

Implementation

In this chapter, the detailed implementation of the GeoWSN node software is shown. First, the development environment with the used tools and hardware is explained in Section 4.1. Afterwards, the developed software with its various functions is described. Also a detailed communication protocol description and flow- and sequence charts are shown in this chapter. The last subsection is about the developed simulation tool, which was used to generate the results of this master thesis.

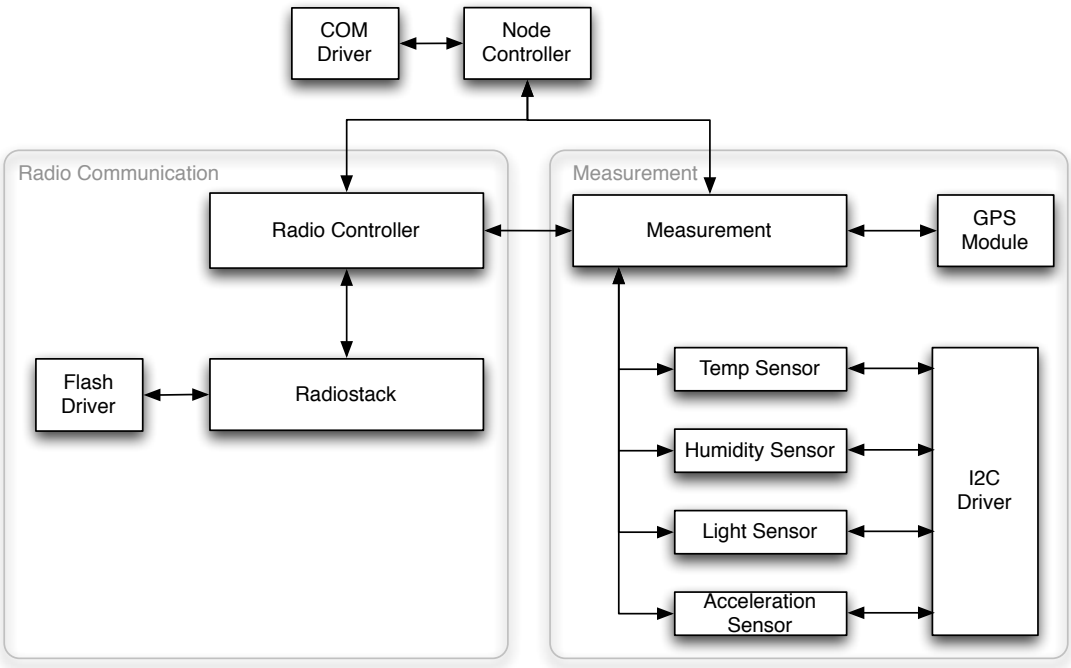


Figure 4.1: Node software components.

In Figure 4.1 the main software modules are shown. These two modules, Radio Communication and Measurement, are also described in the following sections.

4.1 Development Environment

The Implementation takes place on a self-developed wireless sensor node, which is driven by a MSP430 micro controller from Texas Instruments. The design of the developed hardware is presented in [6] and the main details are described in the next subsection. The software is written in C++ by using the Code Composer Studio, which is also used for programming the node. For debugging, the XDCtool with their runtime monitoring system was used. This tool are also described in the following section.

4.1.1 Hardware

The GeoWSN sensor node, shown in Figure 4.2, consists of several micro chips and interfaces. It is powered by a 16 bit MSP430f5438a microprocessor, which have an ultra low power consumption. It is equipped with a 16 MHz oscillator to deal with the large amounts of data produced by the GPS receiver. The chip has 256 kilobyte of program flash and uses 16 kilobyte SRAM.

Figure 4.2: GeoWSN node [6].

On the right upper side of Figure 4.2 the wireless radio module is fixed. It is a Microchip MRF24J40 RF transceiver that uses the 2.4 GHz ISM Band for communication. It is compatible to the IEEE 802.15.4 standard and supports different wireless networking protocols like ZigBee or MiWi. It has a simple four-wire SPI interface for controlling and configuring. The maximal data rate is in normal mode 250 kbps and in turbo mode 625 kbps. Also some MAC layer features are available in hardware. So CSMA-CA, automatic

acknowledgement response and automatic retransmit are included. These mechanisms are needed for a stable wireless communication and all used by the developed WSN.

There is also a temperature and acceleration sensor on the board as well as interfaces for external sensors like illumination and humidity. All sensors are connected by an I2C bus to the micro controller. The external U-Blox GPS receiver is connected by a serial UART interface to the MSP430. It is possible to configure the receiver and read the positioning raw data over this connection. To power the GPS receiver, a standard USB jack is mounted on the board. Furthermore, a seven-segment display, push buttons and connectors for extension boards are also included. So the base station node has a RS232 board on the extension connector. Also a GPRS board should be available in the future to transmit the data directly to the server.

For programming the MSP430, a JTAG connector is also on the node. Through this, it is possible to connect the node by a MSP430 USB-Debugging-Interface with the Code Composer Studio.

4.1.2 Code Composer Studio and XDCtools

The Code Composer Studio (CCS) is an integrated development environment (IDE) from Texas Instruments. It includes compilers for each Texas Instruments embedded device and is a plug-in of the Eclipse platform. So all editor features of Eclipse can be used in CCS and the programmed software can be immediately compiled and deployed on a device. Also full functional real-time debugging directly on the chip is possible.

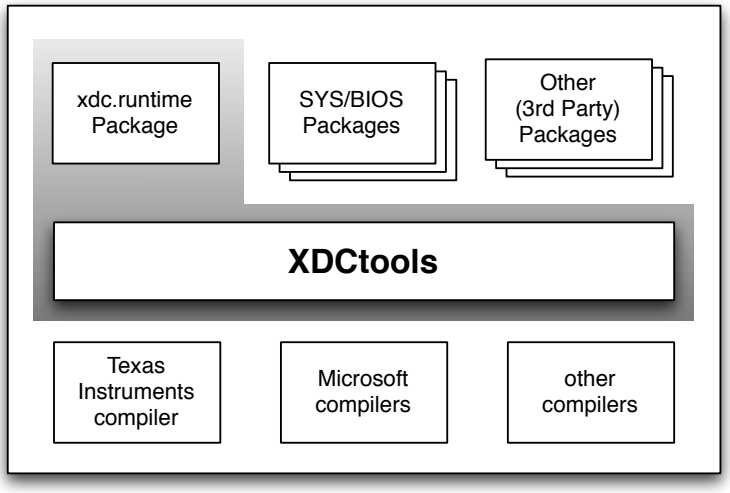


Figure 4.3: SYS/BIOS as a set of packages.

The XDCtools are necessary for the use of SYS/BIOS with the CCS environment. With this tool, SYS/BIOS can be configured and compiled to an operating system. In Figure 4.3

the structure of XDCtools and SYS/BIOS is shown. Thereby, SYS/BIOS can be seen as a set of packages around of the XDCtools. The upper layers are the xdc.runtime package, SYS/BIOS package as well as other user packages. On the other side the compilers, whether Texas Instruments or other compiler, translate the XDCtools output to machine code. Also during debugging the XDCtool are useful with various real-time analysis tools including in the xdc.runtime package. So a live thread switching, CPU load, stack size, blocking diagram, and so on can be viewed.

4.2 Radio Communication

The radio communication is one major part of the implementation of the GeoWSN node software. Figure 4.4 shows the main components of the network stack. The network stack

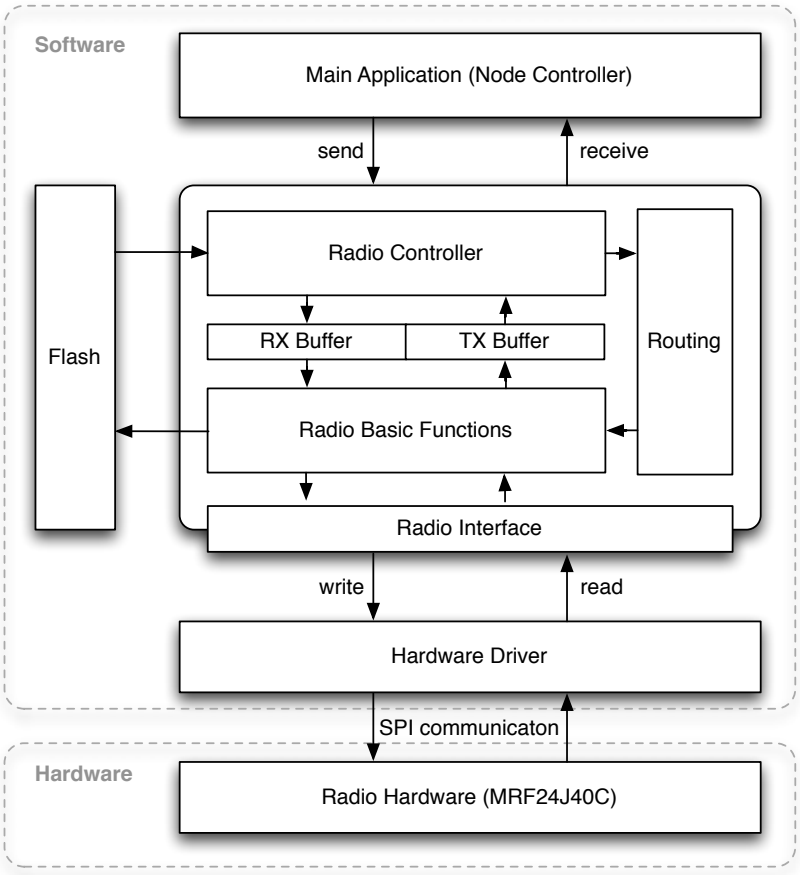


Figure 4.4: Communication modul overview.

abstracts the radio interface as high-level functions like sending and receiving. Therefor different components are used to fulfill the requirements of the network stack. These

components can be associated with layers of the OSI model. The simplified WSN OSI model is shown in Figure 4.5.

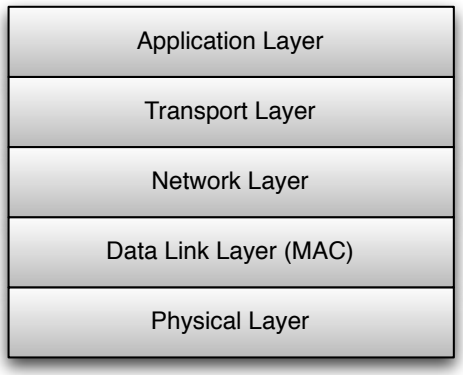


Figure 4.5: WSN OSI model.

The Physical and the Data Link Layer are implemented by the radio hardware, which is used in this project. The Network Layer is represented by the radio controller, the Transport Layer is implemented by the routing component and the Application Layer is represented by the Node Controller.

The routing is implemented as described in Section 3.3.3 and communicates with the radio controller and the Radio Basic Functions (RBF). If a neighbor broadcast is received, an interrupt starts the RX task and this task reads the MSG from the radio hardware and writes the message into the RX buffer. The radio controller analyzes the message and updates, if necessary, the routing table. The send message function in the RBF have to get the next hop address from the routing component if a data message should be sent.

The RX and TX buffer are used for synchronization between the tasks and the buffering of messages. These are needed, because of the hardware-related tasks (RX and TX) have a higher priority as the normal tasks. The radio interface defines the functions are needed by the higher layers and have to be implemented by the radio hardware driver. This makes it easy to change the radio module, only a new driver has to be implemented.

The fail safety features are distributed among the different layers. The automatic retransmit is provided by the MAC layer of the radio hardware module. The acknowledge message is provided by the Network Layer and is used for end-to-end transmission confirmation. If one of these mechanisms detects an error, the message will be written into the flash memory. If afterwards a new link to the base is available, the messages will be retransmitted. Therefor the flash is used as a FIFO. The flash driver scans on the startup of the node the whole flash for saved messages and sets the internal pointers accordingly to the start and end points. In fact that only whole blocks can be deleted and a sector

can only program once, it is necessary to check the flash for free but not erased blocks. The flash driver provides these functions.

4.2.1 Task Operation Description

The component functionality described in the section before, is used by different tasks to get a running network stack. The RX and TX buffers as well as external interrupts from the radio module synchronize the tasks of the network stack. Additional, the tasks have a priority assigned and high priority tasks can interrupt lower priority tasks. The priorities and scheduling sequence is shown in Section 3.3.4.

4.2.1.1 RX Task

The assignment of the receive task is to read the data from the radio module as fast as possible. This is important to get the radio receiver back online, because as long as the old message is not read, new messages can't receive. In Figure 4.6 the flow of the task is shown. If a radio interrupt occurs, the data will read from the radio module over the SPI interface and write the message to the RX buffer. The functions called from this task are from the radio component.

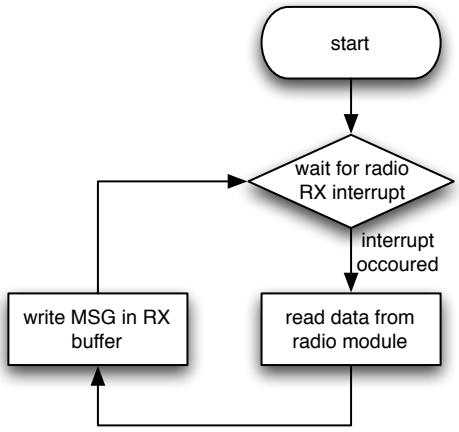


Figure 4.6: RX task flow chart.

4.2.1.2 Process Message Task

This task uses a function of the radio controller to manage the incoming MSGs and is shown in Figure 4.7. It is synchronized by the receive buffer, that means as long as no MSG is in the buffer the task is sleeping. After a message is read from the buffer, the header is checked. If it has been noticed that the message is split, the message is merged to a temporally mrgMSG and if the message is not completely received, the task checks the buffer for new messages and go sleep or starts from new. Otherwise, if the message is

not split or already complete, the destination address will be checked. If the destination is not the current node, the MSG is forwarded to the right destination. If the message has arrived at the destination, the message type is read, and the MSG will be processed accordingly. If an unknown MSG type was read, the MSG will be deleted and the buffer will be checked for new messages.

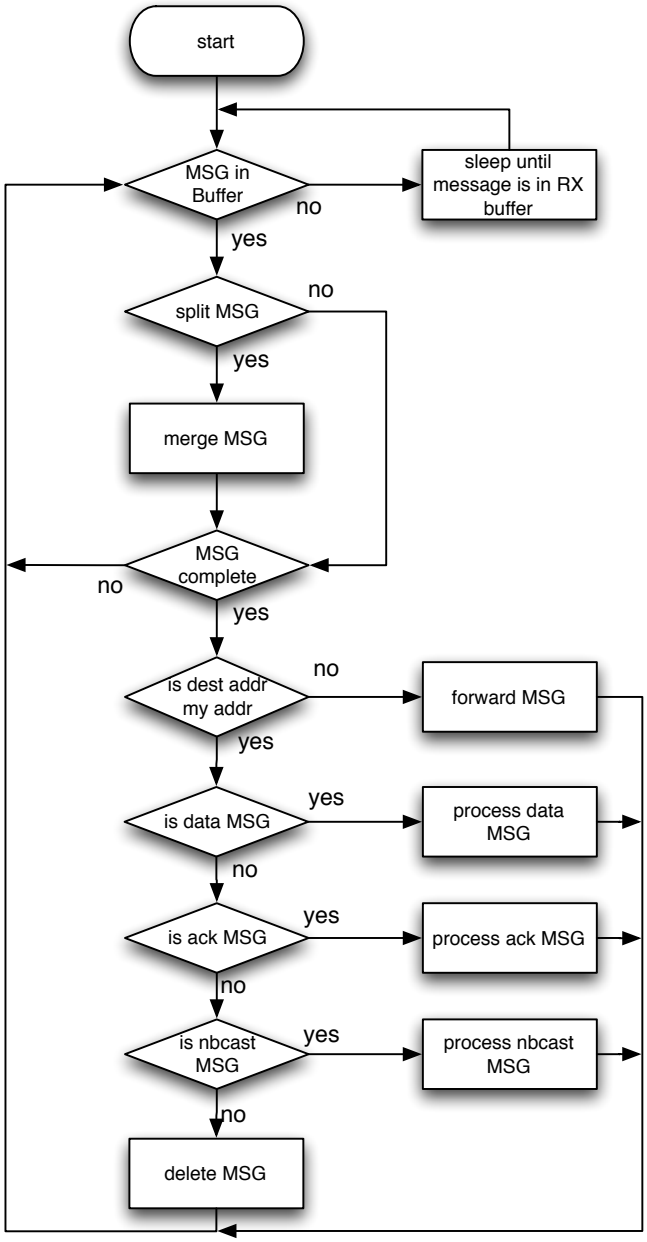


Figure 4.7: Process message task flow chart.

4.2.1.3 TX Task

The transmission task is also located in the radio component and manages the transmission. It is shown in Figure 4.8. Therefor the TX buffer will be checked for messages. If a message is read from the transmission buffer, the next hop address will determine by the routing algorithm and if the address is valid the message is written to the radio module. Afterwards the task waits for an auto acknowledge of the send message from the hop node. If the auto-ack is received, and the send MSG was a data MSG, the last two steps

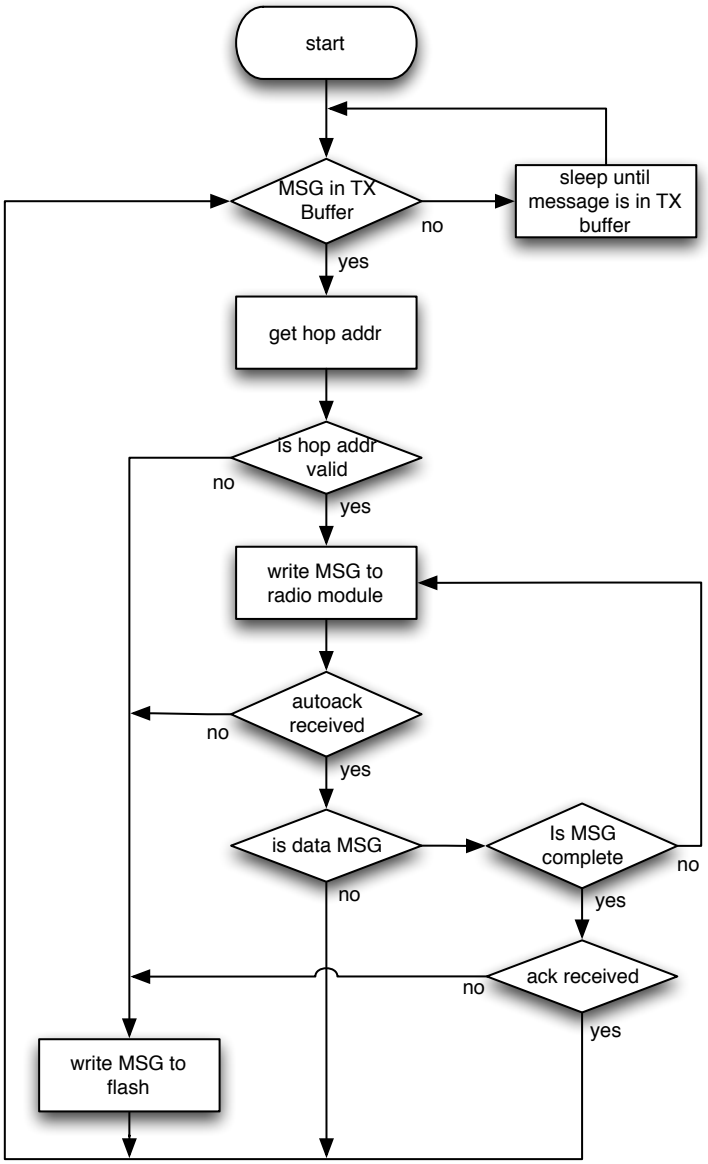


Figure 4.8: TX task flow chart.

are repeated as long as the message is completely transmitted. Then, the task waits for the acknowledge message from the base station which confirms the receiving of the data message. If the message is no data MSG the transmission is finished after the auto-ack is received and the task starts all over again. If anytime a failure, like no valid hop address, occurs the message will be written to the flash memory for a later retransmission.

4.2.2 Communication Flow

The communication flow in Figure 4.9 shows the communication between one base station and two nodes. The software and the communication protocol, and so the super frame slots, are scaleable and works with different network sizes.

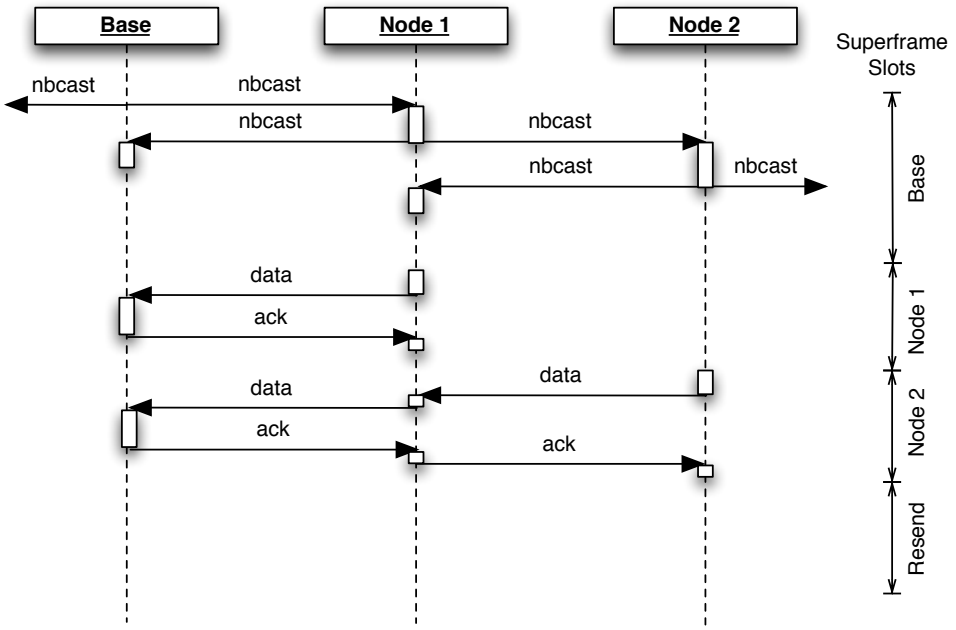


Figure 4.9: Radio communication sequence diagram.

The messages are used for the protocol as described in Section 4.5. The communication flow is regulated by a so called super frame. Thereby, on network cycle is separate in various time slots whereby during every slot only a defined task must be done. The slots are exactly synced on every node by the GPS sync signal whereby failures by a synchronous transmission will be minimized.

First the base has to send a nbcast message to its neighbors. Every node that receives a nbcast will do the same. The nbcast message includes the costs to send a MSG to the base station, and so every node builds up a routing table with the minimum cost path to the base. After this, every node has a slot to send its measured data to the base. If it's not possible to transmit the data message directly, another node has to forward it until

the base is arrived. Then the base send an ack back to the producer node. Therefore the whole path back to the node have to be written in the payload of the ack message. If the node receives the ack, the transmission is successful, otherwise the message will be saved in the flash for a later resend.

Afterwards, the next node gets the slot and will try to send its own data message. After all nodes have finished the data transmission, one node can retransmit its saved data messages. Therefor the node has to register itself by the base by setting a flag in the data message. The first node, which is registered get the permission to resend its data messages during the resend slot.

4.3 Measurement

The measurement component is the second important part of the GeoWSN node software. This component includes the different sensor drivers, the GPS receiver driver and the measurement unit.

4.3.1 Sensors

Every node has various internal and external sensors. These sensors are connected by an I2C bus to the micro controller. So all sensors use one I2C driver. Every sensor has a unique address for access, shown in Table 4.1. The sensors are read periodically, with the same rate as the GPS module. Only the acceleration sensor has to read in a shorter interval to get a correct result. This is due the low-pass behavior of the sensor. To get

Table 4.1: Sensor I2C addresses.

Sensor	Address in HEX
System temperature	0x19
Air temperature	0x18
Ground temperature	0x1B
Air Humidity	0x27
Light	0x4A
Acceleration	0x09

an easy expandability, every sensor driver has to use a defined interface. Therefore every sensor has to implement the following functions:

Sensor(uint8 addr): A Constructor with the assigned address as transfer parameter.

uint16 measure(): Returns the measured short data value.

uint8 getStatus(): Returns the status of the sensor (0 = not working, 1 = power supply ok but no signal , 2 = ready for use)

4.3.2 GPS Module

The used GPS receiver is a u-Blox 6 receiver with RS232 interface. Every node has one of these receivers to get the raw GPS information. Therefore the GPS driver consists of a serial driver with an integrated FIFO buffer and an evaluation unit. The evaluation unit also provides a configuration tool, which uses the configuration MSGs described in Section 4.5.7.3, to set up the receiver. The detailed description of the evaluating algorithm is shown in Section 4.3.3.2. The GPS receiver is not only used for the positioning, it also provides an exact synchronization signal, which is generated by the atomic clocks in the GPS satellites. This signal is transmitted additionally to the positioning and is forwarded to the micro controller every measure interval. So this signal can be used to synchronize the superframes of all nodes.

4.3.3 Task Operation Description

The two tasks are used for the data gathering are the measurement task and the GPS task, both are described in the next section.

4.3.3.1 Measure Task

The measure task is responsible for gathering the environmental information from the sensors and is shown in Figure 4.10. This happens in a periodic cycle, driven by the GPS

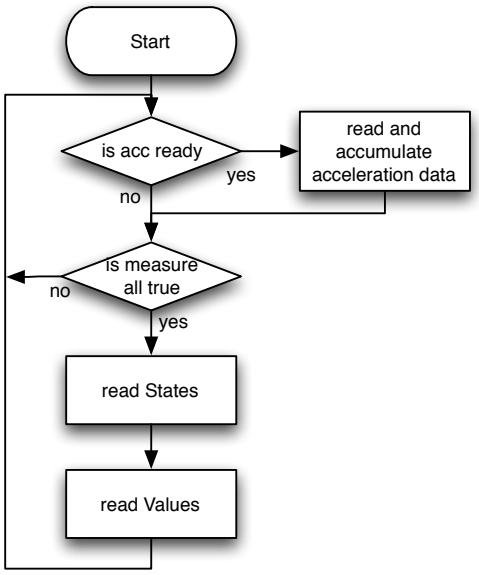


Figure 4.10: Measure task flow chart.

measure interval. Only the acceleration sensor has a shorter interval of 100 milliseconds. Thus, the acceleration sensor is read about 20 times in one GPS interval (two seconds). These measured values are accumulated to the maximum acceleration of the previous period. That is necessary, to lose no acceleration peaks through the low-pass filter of the acceleration sensor. Every 20th acceleration sensor interval, the measure all flag is true and all other sensors will be read, first the status flags and then the values. All measured data is saved in a data array until the transmission is completed.

4.3.3.2 GPS Task

The GPS receiver is linked with an RS232 interface to the micro controller, over which periodically raw data messages from the GPS receiver can be read. As usual by a serial interface, the data is transmitted byte by byte. These bytes are written into the FIFO

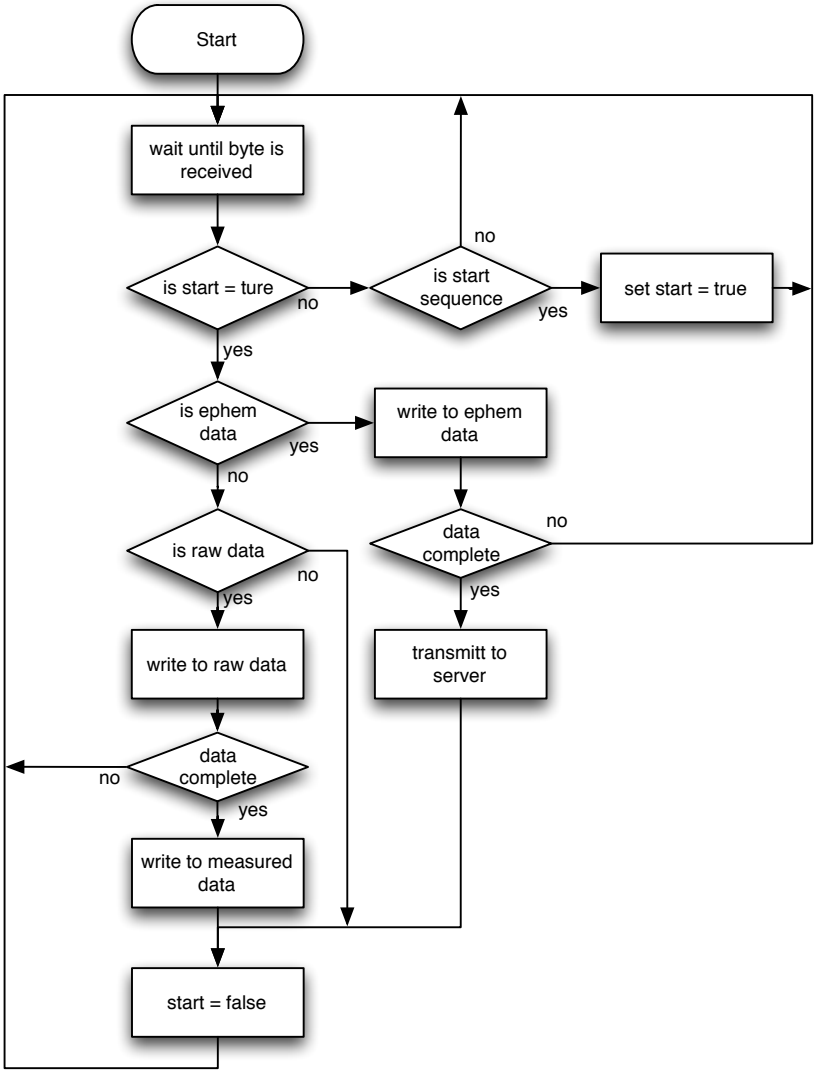


Figure 4.11: GPS task flow chart.

buffer. To recognize the start sequence of a message, it is necessary to analyze the incoming bytes in the FIFO. Therefore every byte from the FIFO is checked for the start sequence. This is shown in Figure 4.11. If the whole start sequence is received, the start flag is set true and the next received byte can be declared as a data byte. This byte is, depending on the data type (raw data or ephemerides data) written to a temporary buffer. If the raw data is completely received, the raw data buffer is written to the measured data and the start flag is set false. If the ephemerides data is completely received (will only happen in the base station), the data is immediately forwarded to the server. If the data is not completely received, the next byte from the FIFO is added to the buffer.

4.4 Node Controller

The node controller is the main class of the GeoWSN node software. It has a different functionality for the base node and the normal node. But the basic functionality - to control the super frame time slots - is included in both. The tasks, which are responsible for this are shown in the next sections.

4.4.1 Base Task

As mentioned before, the base task controls the superframe of the base node and is shown in Figure 4.12. After the startup, the task is waiting for the sync signal from the GPS receiver. This ensures, that all super frames from all nodes are synchronous. If the sync is received, the measure task is started and the task waits until the GPS slot is over. During this time, the GPS task receives the raw data message from the GPS receiver. Afterwards, the received GPS data is sent to the server and the neighbor broadcast is transmitted during the routing slot of the superframe. Then, the task is waiting for received data messages from the nodes. If a message is received, it is forwarded to the server over the RS232. This is repeated until all messages are received or all node slots and the resend slot of the super frame are over. Afterwards the base task checks if the ephemerides have to be sent to the server. An update of these is required every 10 minutes. If an ephemerides data message from the GPS receiver have been received, it is forwarded to the server.

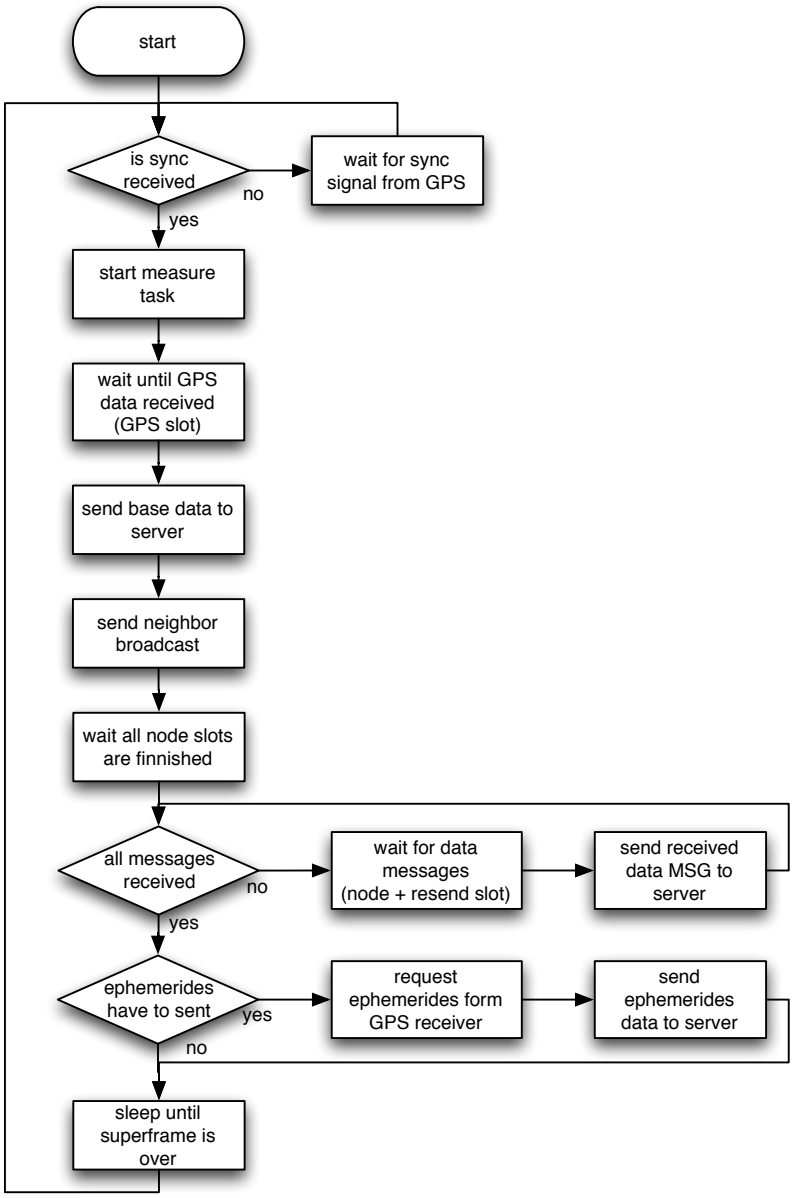


Figure 4.12: Base task flow chart.

4.4.2 Node Task

As shown in Figure 4.13, the startup is the same as for the base task. After the sync signal is received, the measure task is started and the task waits until the GPS raw data is received. Then, the task waits until the own node slot is active and the data message has to be sent. Afterwards, the node waits until all node slots are finished. If the resend slot is active and the node has got the permission from the base, the task retransmits the saved messages.

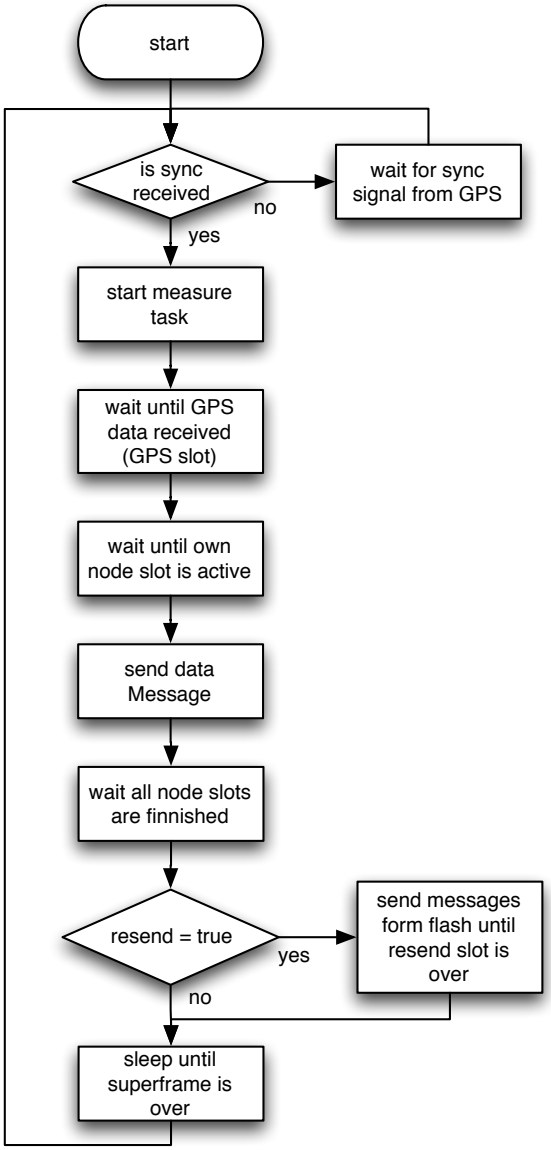


Figure 4.13: Node task flow chart.

4.5 Communication Protocols

In this Section, the communication protocols are described which are used by the GeoWSN. There are different protocols for wireless communication, the base-server communication or the GPS receiver. The wireless protocol is an extension of the standard IEEE 802.15.4 packet format and is described in the following sections. The number formats are used in the protocol descriptions are explained in Table 4.2.

Table 4.2: Number formats.

Short	Type	Size (Bytes)
U1	Unsigned Char	1
I1	Signed Char	1
X1	Bitfiled	1
U2	Unsigned Short	2
I2	Signed Short	2
X2	Bitfiled	2
U4	Unsigned Long	4
I4	Signed Long	4
X4	Bitfiled	4
R4	Float	4
R8	Double	8
Bn	Bitfield with length n	n bit

4.5.1 IEEE 802.15.4 Packet Format

The IEEE 804.15.4 describes the Wireless Personal Area Networks (WPAN). This standard is developed for low power battery driven devices with cheap hardware. Therefor the two lowest OSI layers, MAC- and physical layer, are specified by the standard. The upper layers have to be defined by other protocols like Zigbee. It uses along with Wlan and Bluetooth the license-free ISM bands. For collision avoidance, the CSMA/CA algorithm is defined in the MAC layer. Therefore the incoming signal level is measured before a transmission is started. The specified frame format from the 802.15.4 standard is shown in Table 4.3.

Table 4.3: Frame format of IEEE 802.15.4 packet.

Description:	Frame control	Sequence number	Dest. PAN	Dest. addr.	Src. PAN	Src. addr.	Payload	Check sum
Bytes:	2	1	2	2	2	2	n	2

The header length is eleven bytes and includes the frame control field, a sequence number and the addressing fields. With the PAN identifier, sub networks can be defined with the same address range. This is useful, if different personal networks are in range of each other. After the address field, the payload with a variable length and the footer with a CRC frame check follows. The frame control field is detailed described in Table 4.4 and is used for MAC control sequences.

Table 4.4: Frame control field.

Frame type	Security enabled	Frame pending	Ack. req.	Intra PAN	Res.	Dest. addressing mode	Res.	Source addressing mode
Bit: 0-2	3	4	5	6	7-9	10-11	12-13	14-15

Frame type: The three bits frame type separates the different kinds of MAC messages shown in Table 4.5

Security enabled: It specifies that the integrated security is active and the payload have to encrypt by the receiver.

Frame pending: If this bit is set, another message from the same node is outstanding and the other nodes will not occupy the channel.

Acknowledge required: The auto acknowledge mechanism is activated and a acknowledge is necessary.

Intra PAN: If the bit is set to 1 the frame can only send in the own PAN and only the destination PAN field have to exists.

Destination addressing mode: This subfield controls the destination address field in the header and should set to one of the values from Table 4.6.

Source addressing mode: The same as below, but for the source address field.

Res.: Reserved

Table 4.5: Values of frame type field.

Frame type value $b_2b_1b_0$	Description
000	Beacon
001	Data
010	Acknowledgment
011	MAC command
100 - 111	Reserved

Table 4.6: Values of addressing mode.

Addressing mode value $b_1 b_0$	Description
00	PAN identifier and address field are not present
01	Reserved
10	Address field contains a 16 bit short address
11	Address field contains a 64 bit extended address

4.5.2 Extended GeoWSN Packet Format

For the GeoWSN, the standard 802.15.4 header have to be extended with some additional fields. These fields are essential for the multihop routing algorithm and are shown in Table 4.7. Thereby, the two defined layers remain unchanged; the changes take place in the network layer. Therefore the standard header is extended with network address and packet manage fields for multihop communication. These packet fields are required for message splitting into several packets. This is caused by a limit of 125 bytes per transmission. Minus the header length of 21 bytes, one packet is only able to load 104

Table 4.7: Extended GeoWSN frame [14].

FC	SN	Hop PAN	Hop addr.	My PAN	My addr.	MSG Type	Dest. addr.	Src. addr.	PID	PN	Pay-load size	Pay-load	CRC
Standard Header						Extended Header							

Short form	Bytes	Description
FC	2	Frame control field form standard header
SN	1	Sequence number
Hop PAN	2	Equal to destination PAN from standard header
Hop addr.	2	Equal to destination address from standard header
My PAN	2	Equals to source PAN from standard header
My addr.	2	Equals to source address from standard header
MSG Type	1	Defines the message type of the extended frame
Dest. addr.	2	Destination address in the multihop WSN
Src. addr.	2	Source address in the multihop WSN
PID	1	Packet ID if more then one packets are required for a data message
PN	1	Number of packets in this message
Payload size	1	Payload size of this packet
Payload	n	From zero to 104 bytes
CRC	2	CRC check sum

payload bytes. With the sequence number and the packet ID a unique assignment of every packet is possible. Thus, split messages are able to choose different paths through the WSN and the packets can be joined at the destination to the original message. Also a message ID field is added to the header, because the extended protocol defines some more message types then the IEEE 802.15.4 standard. These are described in Table 4.8. The data field is explained in Table 4.10

Table 4.8: GeoWSN message types.

Message type value	Short form	Description	Payload
2	data	Data Message	Data, Section 4.5.4
3	ack	Acknowledgment Message	Backroute, Section 4.5.5
12	nbcast	Neighbor Broadcast Message	Cost to base, Section 4.5.6

4.5.3 Base Frame Format

The frame shown in Table 4.9 is used by the serial communication from the base node to the server. For a better readability the bytes are transmitted in hexadecimal ASCII code. So one data byte needs two bytes ASCII code for transmission. The start identifier can be a “P” (normal data message) or a “Q” (ephemerides data) and need one byte. This is the sync sequences for the GeoWSN server. After this, a sequence number and the source address follow. The source address is needed to match the received data message to a node.

Table 4.9: Base frame format [14].

ID	SN	Src. addr.	MSG Type	Data size	Data

Short form	Bytes	Description
ID	1	Start identifier
SN	2	Sequence number
Src. addr.	4	Source address in the multihop WSN
MSG Type	2	Defines the message type of the frame
Payload size	2	Payload size of this packet
Payload	2*n	n payload Bytes

4.5.4 GeoWSN Data Payload Format

The payload field of the GeoWSN message frame is explained in this section. It is divided into network (net) information, sensor data, status information and GPS data. This is shown in Table 4.10

Table 4.10: GeoWSN payload format [14].

	Byte Offset	Number Format	Name	Unit	Description
Net info	0	U1 [6]	hopPath	-	Seven bytes with the address of every hop
	6	U1 [7]	hopRSSI	-	RSSI of every hop transmission
	13	U2	netload	b/s	Node network load in Byte per second
	15	X1	netNeighbor	-	Bit mask of which nodes are in range
Sensor data	16	U2	tempSys	-	System temperature
	18	U2	tempAir	-	Air temperature
	20	U2	tempGnd	-	Soil temperature
	22	U3	light	-	Meassured illumination value
	25	I1	accX	mG	Acceleration x-axis in milli G
	26	I1	accY	mG	Acceleration y-axis in milli G
	27	I1	accZ	mG	Acceleration z-axis in milli G
	28	U2	humidity	-	Meassured humidity
States	30	X1	senseState	-	Sensor status mask described in Table 4.11
	31	X1	gpsState	-	GPS sensor status
	32	U2	eInput	-	Energy input level
	34	X1	eState	-	Energy source state
	35	U2	eLevel	-	Energy store level
	37	X1	eLevelState	-	Energy store state
GPS	38	I4	iTOW	ms	GPS time in millisecond of GPS week
	42	I2	week	weeks	GPS week at measurement
	44	U1	numSV	-	Number of satellites (N)
	45	U1	reserved	-	Reserved
	Repeated block from Table 4.12 (numSV times)				

The net information fields are used for statistic analysis of the network topology, routing information and the network-load. Every node, which is forwarding a message, fills in the own address and the signal strength of the received message. The first hop uses field hopPath[0] and hopRSSI[0], the second hopPath[1] and hopRSSI[1] and so on. Thus, the complete message movement is comprehensible. The transmitted sensor data are most raw data, and the have to be converted by the base to double values. The senseState field gives an overview of the working sensors, the sensor is operating correctly if the corresponding bit is set. This is shown in Table 4.11. The transmitted GPS information starts with a header, which includes the satellite time and the number of measured satellites. After this a repeated block shown in Table 4.12 follows a number of satellites times.

Table 4.11: Sensor state mask [14].

sense state	Description
b_0	Humidity sensor
b_1	Iluminiation senor
b_2	Air temperatur sensor
b_3	Soil temperatur sensor
b_4	System temperatur sensor
b_5	Acceleration sensor

4.5.5 GeoWSN Acknowledge Payload Format

Due the routing algorithm, the base station does not know the path to any other node. To transmit an acknowledge message correct, the complete back-path have to be sent with the message. Thus, the acknowledge message has a 7 byte payload with the route back to the node. The last byte in the payload corresponds to the first hop node, the second last corresponds to the second hop, and so on and so forth. Furthermore, the acknowledge message has the same sequence ID as the data message.

4.5.6 GeoWSN Neighbor Broadcast Payload Format

The neighbor broadcast has the estimated costs to the base station as payload. Every node who gets the nbcast from another node, estimates the new costs with the function described in Section 3.3.3 and sends the new value with a own nbcast to its neighbors. Thus, every node can choose the cheapest path to the base.

4.5.7 GPS Frame Format

The communication with the U-Blox GPS receiver take place over an RS232 interface. U-Blox has defined a lot of messages to configure the receiver and also to get the positioning data in different formats. For the used positioning algorithms, only the RXM-RAW data and the RXM-EPH data are interesting, these are described in the following Sections 4.5.7.1 and 4.5.7.2. For the configuration of the receiver various messages are defined. The most important messages are described in Section 4.5.7.3.

4.5.7.1 GPS Raw Data Message

A GPS raw data message (RXM-RAW) consists of a header and a repeated block. The header includes a start sequence, the length of the payload, the sattelit time in weeks and milliseconds, as well as the number of connected satellites (N). The raw data block

is repeated N times, so every received satellite data gets one block. This is shown in Table 4.12.

Table 4.12: GPS RXM-RAW frame [14].

Header	ID	Length (bytes)	Payload	Checksum
0xB5 0x62	0x02 0x10	$8 + 24 * numSV$	see below	CK_A CK_B

Payload Content				
Byte Offset	Number Format	Name	Unit	Description
0	I4	iTOW	ms	GPS time at measurement in millisecond of GPS week
4	I2	week	weeks	GPS week at measurement
6	U1	numSV	-	Number of satellites (N)
7	U1	reserved	-	Reserved
Start of repeated block (numSV times)				
8+24*N	R8	cpMes	cycles	Carrier phase measurement
16+24*N	R8	prMes	m	Pseudorange measurement
24+24*N	R4	doMes	Hz	Doppler measurement
28+24*N	U1	sv	-	Space vehicle number
29+24*N	I1	mesQI	-	Nav measurements quality indicator
30+24*N	I1	cno	dbHz	Signal strength C/No.
31+24*N	U1	lli	-	Loss of lock indicator
End of repeated block				

4.5.7.2 GPS Ephemerides Data Message

The GPS ephemerides data are base parameters for building up a ionosphere model and is described in Table 4.13. The information is divided in three subframes shown in Table A.1,

Table 4.13: Ephemeris data for a SV [14].

Header	ID	Length (bytes)	Payload	Checksum
0xB5 0x62	0x02 0x31	$(8) or (104)$	see below	CK_A CK_B

Payload Content				
Byte Offset	Number Format	Name	Unit	Description
0	U4	svid	-	SV id
4	U4	how	-	Hand-Over Word of first subframe 0 indicates no ephemeris
Start of optional block				
8	U4[8]	sf1d	-	Subframe 1 Words 3..10 (SF1D0..SF1D7)
40	U4[8]	sf2d	-	Subframe 2 Words 3..10 (SF2D0..SF2D7)
72	U4[8]	sf3d	-	Subframe 3 Words 3..10 (SF3D0..SF3D7)
End of optional block				

A.2 and A.3. It is important to know that the number format of these tables is in bits, so B10 means 10 bits.

4.5.7.3 GPS configuration messages.

The messages, which are described in the following, are used for the the GPS receiver configuration. This is necessary to get the needed data from the receiver.

- With the config-save message shown in Table 4.14, it is possible to clear, save and load configurations. With the deviceMask the memory (FLASH, SPI FLASH, battery buffered RAM or EEPROM) source can be selected.

Table 4.14: GPS CFG-CFG frame [14].

Header	ID	Length (bytes)	Payload	Checksum
0xB5 0x62	0x06 0x09	12 or 13	see below	CK_A CK_B

Payload Content				
Byte Offset	Number Format	Name	Unit	Description
0	X4	clearMask	-	Mask with configuration sub-sections to clear
4	X4	saveMask	-	Mask with configuration sub-section to save
8	X4	loadMask	-	Mask with configuration sub-sections to load
12	X1	deviceMask	-	Mask which selects the devices for this command

- In Table 4.15, the rate config message is shown. Hence, it is possible to configure the GPS measure interval.

Table 4.15: GPS CFG-RATE frame [14].

Header	ID	Length (bytes)	Payload	Checksum
0xB5 0x62	0x06 0x08	6	see below	CK_A CK_B

Payload Content				
Byte Offset	Number Format	Name	Unit	Description
0	U2	measRate	ms	Measurement rate in millisecond
2	U2	navRate	cycles	Navigation rate, in u-bloc 6 it cannot be changed
4	U2	timeRef	-	Alignment to reference time: 0 = UTC, 1 = GPS time

- With the CFG-MSG frame, the rate of the transmitted messages can be configured. Thereby, shown in Table 4.17, the msgClass and msgID defines the message and the rate defines the message send rate on the different IOs. Therefore the navigation rate, which is defined before, is the base of the message rate. If the message rate is set to 10, every tenth positioning cycle the selected message is transmitted to the node.

Table 4.16: GPS CFG-MSG frame [14].

Header	ID	Length (bytes)	Payload	Checksum
0xB5 0x62	0x06 0x01	8	see below	CK_A CK_B

Payload Content				
Byte Offset	Number Format	Name	Unit	Description
0	U1	msgClass	-	Message Class
1	U1	msgID	-	Message Identifier
2	U1[6]	rate	-	Send rate on I/O Target (6 Targets)

- The SBAS configuration is used to disable the SBAS reception on the GPS receiver. This additional information from other satellites is not used for these positioning algorithms. Thus, the SBAS message must be disabled.

Table 4.17: GPS CFG-SBAS frame [14].

Header	ID	Length (bytes)	Payload	Checksum
0xB5 0x62	0x06 0x16	8	see below	CK_A CK_B

Payload Content				
Byte Offset	Number Format	Name	Unit	Description
0	X1	mode	-	SBAS mode, (1: enabled, 0: desabled)
1	X1	usage	-	SBAS usage
2	U1	maxSBAS	-	Maximum number of SBAS tracking channels
3	X1	scanmode2	-	Continuation of scanmode bitmask
4	X4	scanmode1	-	Which SBAS PRN numbers to search for Bitmask

4.6 Class Discription

The developed node software is written in C++, because of its data encapsulation. Figure 4.14 shows the class diagram of the developed node software. It is split into two main parts, the radio classes and the measurement classes. Furthermore, the Node Controller connects both parts.

Node Controller: Is the main class of the software. It implements the measured data and the radio controller and controls the superframe communication. Also the synchronization between the measurement and the radio module is handled by this class.

Radio Controller: This class implements the top layer of the network stack, with the process message task and the high level transmit and receive functions. It is connected by the RX and TX buffers with the lower radio layers.

TXRX Buffer: Is implemented as a queue, first in first out, whole message objects can store in the buffers.

Radio: Implements the basic transmit and receive functions and resolve the MSG object to a byte stream.

Radio Hardware IF: Defines all functions which have to be implemented by every radio driver to guarantee an easy adapting to another radio hardware.

Radio Hardware: Implements all hardware specific radio functions which are defined by the Radio Hardware IF. These are functions like write to TX buffer or read from RX buffer and sleep or reset the radio module.

Radio SPI Driver: This class implements all SPI functions used in the Radio Hardware to communicate with the radio module.

Flash Driver: This class implements all functions to handle the flash memory. The flash is use as a FIFO memory. So the start and stop pointer, the write and read functions, just as erase functions are implemented.

Measured Data: All sensors are held, read and controlled by this class. Also the measured data array is stored in this class until the transmission was successful.

Sensors: Every physical sensor has a class which includes the I2C driver and controls the functionality of the sensor.

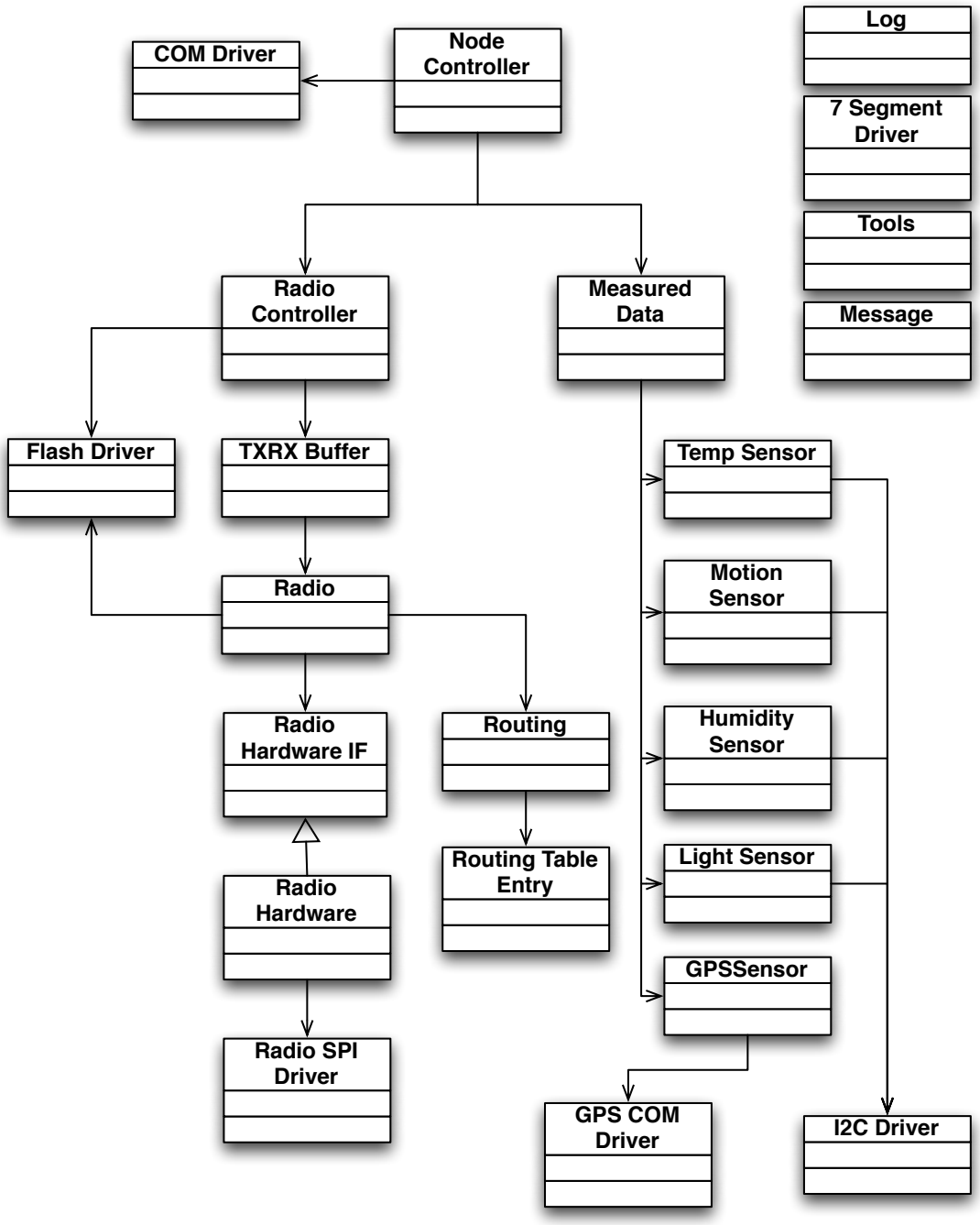


Figure 4.14: Class dependency diagram.

GPS Sensor: Implements the GPS functionality, waits for data form the GPS receiver and parse the information from the data stream. Also the configuration functions are included in this class. For communication, it implements the GPS COM driver.

GPS COM Driver: This class implements the COM driver for the GPS communication.

I2C Driver: Is used by the sensor drivers to communicate with the hardware. It also manages the access on the bus.

COM Driver: This class implements the COM driver for the server communication.

Log: Implements a serial log and debug interface.

7 Segment Driver: Controls the seven segment display.

Tools: Provides some general functions like oscillator on and off.

Message: Represents a radio message and implements different functions for generate, merge and split messages.

4.7 Simulation Tool WSNsim

WSNsim is a little WSN simulator written in java. The implementation of the simulator was necessary to get relevant results, because it would be very complex to make real test. Also the needed hardware was not available at this time. The simulator consists of different classes which represents different components of the developed node software. These classes are described in the following:

Main: This class is the controller of the simulation. First it configures the nodes and sets the network topology. Then the simulation is started by startup the base node every simulation interval.

Node: Every node object implements several functions for routing and transmission issues. Especially the routing functionality is exactly the same as in the real node software. The nodes are connected by virtual wireless connections, represented by the radio class. Thus, a complete network communication with broadcasts and data messages can be simulated.

To estimate the consumed energy, every node calculates the radio-on time of every interval. With the measured power of the different components of the real hardware and the calculated time, the estimated energy consumption can be assumed. The remaining energy levels of the sensor nodes are tracked during the simulation.

Radio: The radio class implements the radio channel. Therefor different parameters like RSSI, distance and packet loss property are assigned to each connection. This parameters are determined by a measurement with real hardware. Thus, it is possible to simulate a simple wireless radio channel.

To estimate the network lifetime, the energy consumption is deduced every interval form the energy levels of the nodes. It is also possible to test different routing algorithms in one interval to get a representative result.

Another test case is to simulate the packet loss probability. Therefor the RSSI radio channel parameters has been decreased every interval to get a higher packet loss probability. Hence, the implemented safety mechanisms have been tested. This tests also have also been made with different topologies and different number of hops . The gained results are presented in Chapter 5.2.

Chapter 5

Results

This chapter deals with the results made in this master thesis. Especially the improvements which are achieved by the implemented network stack and the routing protocol are presented. These results are determined by a self-implemented simulation tool called WSNsim. This is caused by the non availability of a simulator for SYS/BIOS. Furthermore, the use of an existing WSN simulation tool like Ns2 or OMNet++ would have a much higher adaption-effort, caused by the higher complexity and the resulting training period.

5.1 Energy-aware

One requirement on the GeoWSN was a power-aware routing protocol, which have to increase the network-lifetime. Therefor, as described before, a network stack was imple-

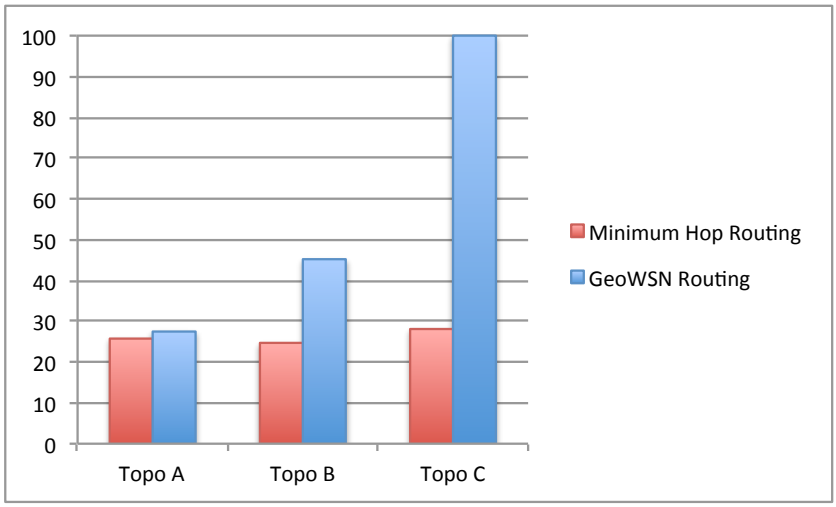
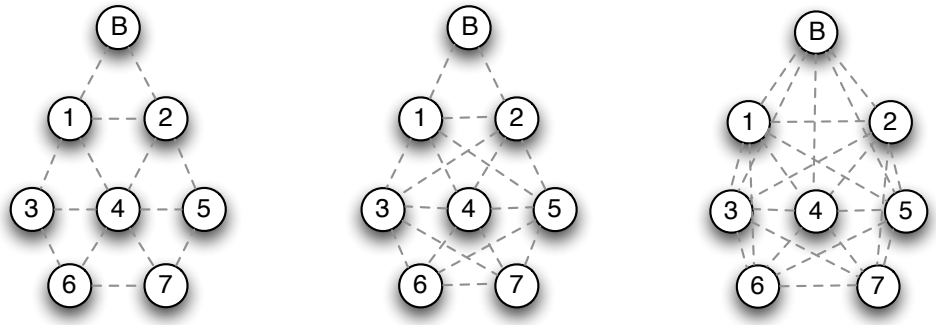


Figure 5.1: Network lifetime (without GPS).

mented. In Figure 5.1, the network lifetime increase by using the implemented routing protocol is shown. The simulation was done with different network topologies, which are described in Figure 5.2. The results shown in this figure are made without the GPS receiver. So it is possible to get a three times higher network lifetime, if only the node power consumption is considered and the topology of the nodes is overlapping as in Figure 5.2c. This is possible, because due the overlapping topology the multihop messages are distributed on several notes, depending on the energy state of the notes. It is not so efficient with simple star topologies (Figure 5.2a), caused by the fact, that the whole communication have to be managed by node 1 and 2.



(a) Topo A (b) Topo B (c) Topo C

Figure 5.2: Topologys of simulated WSN.

Figure 5.3 shows the network lifetime increase with the activated GPS receiver. Because the receiver is the largest energy consumer, the improvement is reduced to only 20 percent by using the third network topology.

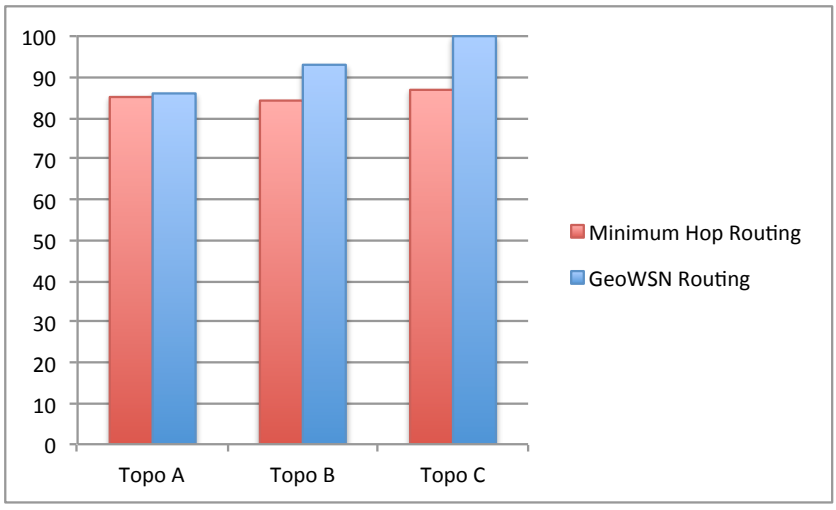


Figure 5.3: Network lifetime.

5.2 Fault-tolerant

The second big requirement was to get a fault-tolerant system, which guarantee, a data transmission from the nodes to the base. As shown in Figure 5.4, the packet-loss probability without safety mechanisms is increasing with the transmission distance. One data set, which was determined by a real measurement, has been included into the WSNsim transmission layer model. Thus, the simulated results based on real radio channel parameters. This parameters are determined by a distance measurement that was done with the used radio modules and conforms the packet-loss probability without safety mechanisms curve. The simulation has been done with enabled auto-acknowledge and with enabled flash memory. Looking at these results, it can be assumed that the usable transmission range can be doubled by using the implemented mechanisms. This results are only valid for no-hop topologies.

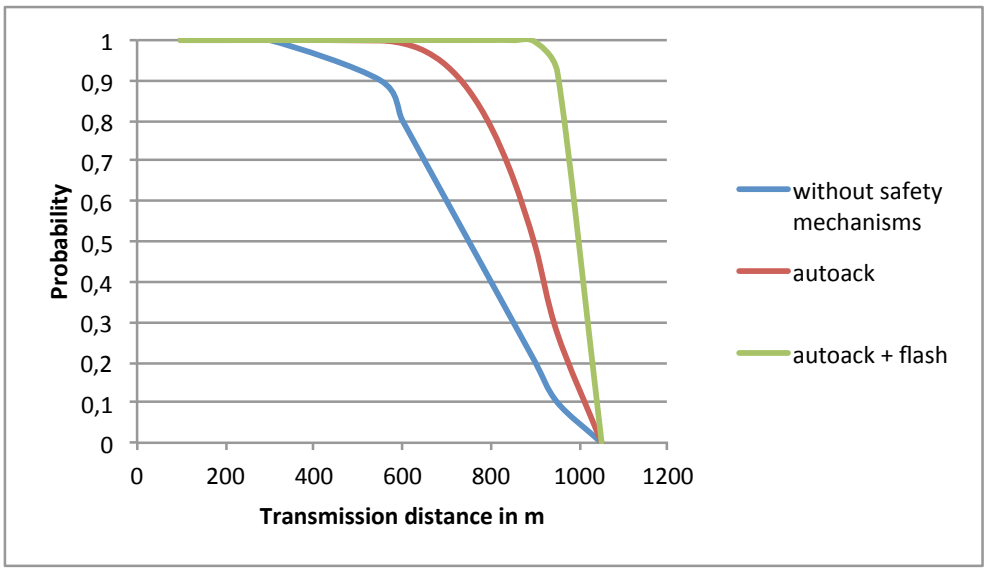


Figure 5.4: Packet receive probability with different safety mechanisms.

If one ore more hops are needed by a message to reach the base station, the results shown in Figure 5.5 are appropriate. There can be seen that every hop reduces the usable transmission distance. This is caused by the different operations of the safety mechanisms. The auto-acknowledge operates between two nodes and the flash-retransmit mechanism operates on the end-to-end connection. This mechanisms only works until the network load is not too high for retransmitting the messages. In addition, more hops mean higher failure probability, the network overload is reached faster. But it can also be seen that the useable distance decreases very slow by increasing the hops counts. Thus, the useable

transmission range is never below 600 meters. For the GeoWSN project the maximum usable range even increase to 800 meters, because of the routing is limited to three hops.

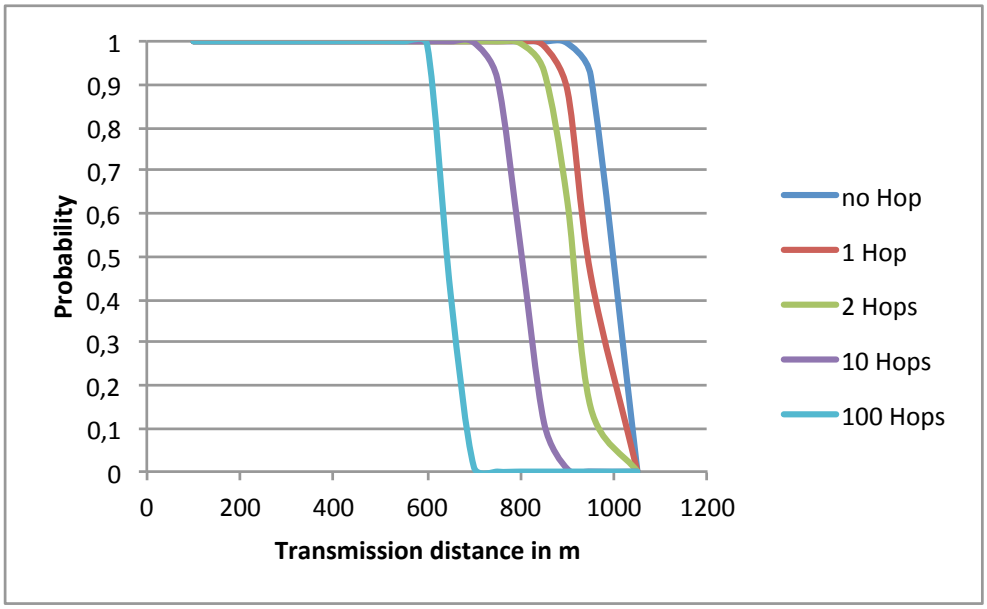


Figure 5.5: Packet end-to-end receive probability.

Chapter 6

Conclusion and Future Work

The GeoWSN project has the purpose to show, that it is possible to build a GPS equipped WSN for environmental monitoring. This master thesis deals with the software used in the nodes of this WSN. Therefor special requirements, which come with the monitoring and alarming system, have to be met by the WSN. Thus, it was necessary to design and develop a new software for the nodes of the WSN.

The software is implemented on a real-time OS that is called SYS/BIOS, which is provided by Texas Instruments for use with MSP430 micro controller. This have been necessary to met the near real-time requirements of the WSN. Furthermore, the OS supports multithreading with preemptive priority scheduling, which allows several tasks to perform quasi-simultaneously. Thus, it was possible to divide the software into three parts, the Node Controller, the Network Stack and the Measurement Unit.

Node Controller: It is the main component of the software and controls and synchronizes the other components. It is also responsible for the superframe management.

Network Stack: This component provides all functionality needed for a stable wireless network communication. It implements the safety mechanisms as well as the routing protocol and the radio interface.

Measurement Unit: It is appropriate for the continuous data acquirement from the connected sensors and the GPS receiver communication. It also provides the GPS sync signal.

6.1 GeoWSN Features

The features, which have been implemented to achieve the given requirements are summarized in this sections.

Superframe: The superframe is used for network synchronization to minimize transmission failures. The superframe of every node is triggered at the same time. Further, the superframe is divided into several time slots, whereby every node has its own slot for sending. Thus, the possibility of hidden or exposed station problem is minimized.

Auto Ack & Retransmission: This mechanism is provided by the MAC layer of the radio module and it requests an acknowledge from the receiver. If the receiver doesn't answer, the transmission is repeated until the maximum tries are reached. This feature makes the connection between two nodes saver.

Acknowledgment: The Acknowledgment is the end-to-end transmission control of the multihop network. If a data message reach the base, an acknowledge message will be send back. Only, if this message is received by the node, the transmission is successful.

Flash Memory: If any safety mechanisms fails and the transmission wasn't successful, the current data message will be saved in the flash memory. Afterwards, if the communication is possible again, the saved messages will be retransmitted to the base.

Minimum Cost Routing: This routing protocol tries to find the cheapest path to the base. Therefor a cost function estimates the costs for a transmission. This function also includes a power indicator, thus a maximization of the network lifetime is possible. The routing algorithm is also able to make a route discovery in only one iteration and minimizes the control messages.

These features and mechanisms make it possible to get a stable, fault tolerant and power-aware WSN, which is able to handle the requirements for the slope monitoring system.

6.2 Results

These results are determined by a self-implemented simulation tool called WSNsim. The simulation was used to show the power-awareness and the fault tolerance of the implemented system.

Through the implemented features a magnification of the lifetime of about 20 percent is possible. If the main energy consumer, the GPS receiver, is off, even a lifetime enhancement of about 300 percent is possible.

The fault tolerance, by using the implemented network stack, is about three times higher. That means a stable end-to-end connection without any message loss is possible with a three times greater transmission distance between the hops. That enlarges the monitoring area and guarantees that no event can be missed.

6.3 Future Work

As mentioned before, all requirements of the WSN are met by the implemented software. But there are some possible improvements, which are listed in the following:

- The security of the wireless communication between the nodes can be improved. Therefor, a AES 128 algorithm should be implemented to ensure a secure communication.
- Also a data compression should be implemented. To reduce the energy consumption the payload of the data messages have to be decreased. This is possible with the reduction of redundant data or by changing the number representations.
- Another improvement would be reached by the integration of a new power-on-radio transceiver. This transceivers include a sleep mode for multihop protocols. That means, that the transceiver is able to power on by a radio request and forward the message without waking-up the main processor.

Appendix A

Ephemerides Subframes

This following tables describes the subframes of the ephemerides data. It is important to know that the number format of these tables is in bits, so B10 means 10 bits.

Table A.1: Subframe 1.

	Bit Offset	Number Format	Scale Factor (LSB)	Name	Unit	Description
Word 3	0	B10	1	WN	weeks	Week number
	10	B2	1	C/A or P on L2	-	Codes on L2 channel
	12	B4	-	URA index	-	SV accuracy
	16	B6	1	SV health	-	SV health
	22	B2	-	IODC (MSBs)		Issue of Data, Clock
	24	B6	-	P	-	Parity bits
Word 4	30	B1	1	L2 P data flag	-	Data flag for L2 P-code
	31	B23	-	R	-	Reserved
	54	B6	-	P	-	Parity bits
W 5	60	B24	-	R	-	Reserved
	84	B6	-	P	-	Parity bits
W 6	90	B24	-	R	-	Reserved

	114	B6	-	P	-	Parity bits
Word 7	120	B16	-	R	-	Reserved
	136	I1	2^{-31}	T_{GD}	s	Estimated group delay differential
	144	B6	-	P	-	Parity bits
Word 8	150	U1	-	IODC (LSBs)		Issue of Data, Clock
	158	U2	2^{4}	t_{oc}	s	SV clock correction parameter
	174	B6	-	P	-	Parity bits
Word 9	180	I1	2^{-55}	a_{f2}	s/s^2	SV clock correction parameter
	188	I2	2^{-43}	a_{f2}	s/s	SV clock correction parameter
	204	B6	-	P	-	Parity bits
Word 10	210	B'22	2^{-31}	a_{f0}	s	SV clock correction parameter
	232	B2	-	t	-	Noninformation bearing bits
	234	B6	-	P	-	Parity bits

Table A.2: Subframe 2.

	Bit Offset	Number Format	Scale Factor (LSB)	Name	Unit	Description
Word 3	0	U1	-	IODE	-	Issue of Data (Ephemeris)
	8	I2	2^{-5}	C_{rs}	m	Amplitude of the Sine Harmonic Correction Term to the Orbit Radius
	24	B6	-	P	-	Parity bits
Word 4	30	I2	2^{-43}	Δn	semi-circles/s	Mean motion difference from computed value
	46	I1	2^{-31}	M_0 (MSBs)	semi-circles	Mean anomaly at reference time
	54	B6	-	P	-	Parity bits
W 5	60	I3	2^{-31}	M_0 (LSBs)	semi-circles	Mean anomaly at reference time
	84	B6	-	P	-	Parity bits
Word 6	90	I2	2^{-29}	C_{UC}	radians	Amplitude of the Cosine Harmonic Correction Term to the Argument of Latitude

	106	U1	2^{-33}	e	-	Eccentricity
	114	B6	-	P	-	Parity bits
W 7	120	U3	2^{-33}	e	-	Eccentricity
	144	B6	-	P	-	Parity bits
Word 8	150	I2	2^{-29}	C_{US}	radians	Amplitude of the Sine Harmonic Correction Term to the Argument of Latitude
	166	U1	2^{-19}	$\sqrt{A}$ (MSBs)	$\sqrt{m}$	Square root of the semi-major axis
	174	B6	-	P	-	Parity bits
W 9	180	U1	2^{-19}	$\sqrt{A}$ (LSBs)	$\sqrt{m}$	Square root of the semi-major axis
	204	B6	-	P	-	Parity bits
Word 10	210	U2	2^{4}	t_{oe}	s	Reference time ephemeris
	226	B1	-	FIF	-	Fit interval flag
	227	B5	-	AODO	-	Age of Data Offset
	232	B2	-	t	-	Noninformation bearing bits
	234	B6	-	P	-	Parity bits

Table A.3: Subframe 3.

	Bit Offset	Number Format	Scale Factor (LSB)	Name	Unit	Description
Word 3	0	I2	2^{-29}	C_{ic}	radians	Amplitude of the cosine harmonic correction term to the angle of inclination
	16	I1	2^{-31}	Ω_0 (MSBs)	semi-circles	Longitude of ascending node of orbit plane at weekly epoch
	24	B6	-	P	-	Parity bits
W 4	30	I3	2^{-31}	Ω_0 (LSBs)	semi-circles	Longitude of ascending node of orbit plane at weekly epoch
	54	B6	-	P	-	Parity bits
Word 5	60	I2	2^{-29}	C_{is}	radians	Amplitude of the sine harmonic correction term to the angle of inclination

	76	I1	2^{-31}	i_0	semi-circles	Inclination angle at reference time
	84	B6	-	P	-	Parity bits
W 6	90	I3	2^{-31}	i_0	semi-circles	Inclination angle at reference time
	114	B6	-	P	-	Parity bits
Word 7	120	I2	2^{-5}	C_{rc}	m	Amplitude of the cosine harmonic correction term to the orbit radius
	136	I1	2^{-31}	ω (MSBs)	semi-circles	Argument of perigee
	144	B6	-	P	-	Parity bits
W 8	150	I3	2^{-31}	ω (LSBs)	semi-circles	Argument of perigee
	174	B6	-	P	-	Parity bits
W 9	180	I3	2^{-43}	$\dot{\Omega}$ (LSBs)	$\sqrt{m}$	Rate of right ascension
	204	B6	-	P	-	Parity bits
Word 10	210	U1	-	IODE	-	Issue of data (ephemeris)
	218	B'14	2^{-43}	IDOT	-	Rate of inclination angle
	232	B2	-	t	-	Noninformation bearing bits
	234	B6	-	P	-	Parity bits

Bibliography

[1] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad hoc networks*, 3(3):325–349, 2005.

[2] N.A. Ali, M. Drieberg, and P. Sebastian. Development of Wireless Sensor Network for slope monitoring. In *Intelligent and Advanced Systems (ICIAS), 2012 4th International Conference on*, volume 1, pages 285–290. IEEE, 2012.

[3] B. Buchli, F. Sutton, and J. Beutel. GPS-equipped wireless sensor network node for high-accuracy positioning applications. *Wireless Sensor Networks*, pages 179–195, 2012.

[4] J.H. Chang and L. Tassiulas. Maximum lifetime routing in wireless sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 12(4):609–619, 2004.

[5] Y. Cui and H. Qin. A Novel Rumor Routing for Wireless Sensor Network. In *Genetic and Evolutionary Computing (ICGEC), 2010 Fourth International Conference on*, pages 795–797. IEEE, 2010.

[6] Marcus Dietl. Entwurf und Implementierung der Hardware-Architektur eines drahtlosen Sensorknotens zur Überwachung gefährdeter Hänge mittels GPS. to be published, Institute for Technical Informatics, Graz University of Technology, 2013.

[7] XL Ding, WJ Dai, WT Yang, XW Zhou, J. Lam, Q. Zhang, and L. Wang. Application of multi-antenna GPS technology in monitoring stability of slopes. *Proceedings of Strategic Integration of Surveying Services, FIG Working Week*, 2007.

[8] A. Dunkels, B. Gronvall, and T. Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462. IEEE, 2004.

[9] Shijie Fan, Yanxiong Liu, Caijun Xu, Jiming Guo, and Zhenjie Wang. GPS kinematic Precise Point Positioning based on sequential least squares estimation. In *Geoinformatics, 2010 18th International Conference on*, pages 1 –5, june 2010.

[10] T. He, J.A. Stankovic, C. Lu, and T. Abdelzaher. SPEED: A stateless protocol for real-time communication in sensor networks. In *Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on*, pages 46–55. IEEE, 2003.

[11] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2000.

[12] W.R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185. ACM, 1999.

[13] J. Henaut, A. Lecointre, D. Dragomirescu, and R. Plana. Radio interface for high data rate wireless sensor networks. *arXiv preprint arXiv:1004.0204*, 2010.

[14] L.B. Hörmann, S Gether, H. Mock, M. Dietl, and C. Steger. GeoWSN - Implementierung und Anwendungsrichtlinien. Technical report, Institut für Technische Informatik, Technische Universität Graz, 2013.

[15] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *Networking, IEEE/ACM Transactions on*, 11(1):2–16, 2003.

[16] K.T. Kim and J.G. Han. Design and Implementation of a Real-Time Slope Monitoring System Based on Ubiquitous Sensor Network. In *Proc. 25th International Symposium on Automation and Robotics in Construction (ISARC 2008), Vgtu, Lithuania*, pages 330–336, 2008.

[17] M. Kohvakka, T. Arpinen, M. Hännikäinen, and T.D. Hämäläinen. High-performance multi-radio WSN platform. In *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, pages 95–97. ACM, 2006.

[18] Bofeng Li and P.J.G. Teunissen. Real-Time Kinematic positioning using fused data from multiple GNSS antennas. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 933 –938, july 2012.

[19] Jian Li. Quality of Service (QoS) Provisioning in Multihop Ad Hoc Networks. Master's thesis, University of California, 2006.

[20] L. Li and J.Y. Halpern. Minimum-energy mobile wireless networks revisited. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 1, pages 278–283. IEEE, 2001.

[21] S. Lindsey and C.S. Raghavendra. PEGASIS: Power-efficient gathering in sensor information systems. In *Aerospace conference proceedings, 2002. IEEE*, volume 3, pages 3–1125. IEEE, 2002.

[22] Mykhailo Lytvyn, Christoph Pollabauer, Markus Troger, Katrin Landfahrer, Leander Hormann, and Christian Steger. Real-Time landslide monitoring using single-frequency PPP: Proof of concept. In *Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing,(NAVITEC), 2012 6th ESA Workshop on*, pages 1–6. IEEE, 2012.

[23] A. Manjeshwar and D.P. Agrawal. TEEN: a routing protocol for enhanced efficiency in wireless sensor networks. In *1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, volume 22, 2001.

[24] A. Manjeshwar and D.P. Agrawal. APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks. In *Proceedings of the 16th International Parallel and Distributed Processing Symposium*, page 48, 2002.

[25] ZHU Nanhao, DU Wan, F.M. David NAVARRO, and O.C. Ian. High Data Rate Wireless Sensor Networks Research.

[26] C. Park and P.H. Chou. eCAM: ultra compact, high data-rate wireless sensor node with a miniature camera. In *Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 359–360. ACM, 2006.

[27] N. Sadagopan, B. Krishnamachari, and A. Helmy. The ACQUIRE mechanism for efficient querying in sensor networks. In *Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on*, pages 149–155. IEEE, 2003.

[28] K. Smarsly, K. Georgieva, M. König, and KH Law. MONITORING OF SLOPE MOVEMENTS COUPLING AUTONOMOUS WIRELESS SENSOR NETWORKS AND WEB SERVICES. 2012.

[29] K. Sohrabi, J. Gao, V. Ailawadhi, and G.J. Pottie. Protocols for self-organization of a wireless sensor network. *Personal Communications, IEEE*, 7(5):16–27, 2000.

[30] J.Y. Teo, Y. Ha, and C.K. Tham. Interference-minimized multipath routing with congestion control in wireless sensor network for high-rate streaming. *Mobile Computing, IEEE Transactions on*, 7(9):1124–1137, 2008.

[31] K. Thuro, T. Wunderlich, and O. Heunecke. Development and testing of an integrative 3D early warning system for alpine instable slopes (alpEWAS). *Geotechnologien Science Report, S*, pages 101–112, 2007.

[32] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 70–84. ACM, 2001.

[33] F. Ye, A. Chen, S. Lu, and L. Zhang. A scalable solution to minimum cost forwarding in large sensor networks. In *Computer Communications and Networks, 2001. Proceedings. Tenth International Conference on*, pages 304–309. IEEE, 2001.

[34] J. You, D. Lieckfeldt, F. Reichenbach, and D. Timmermann. Context-aware geographic routing for sensor networks with routing holes. In *Wireless Communications and Networking Conference, 2009. WCNC 2009. IEEE*, pages 1–6. IEEE, 2009.

[35] J. You, D. Lieckfeldt, J. Salzmann, and D. Timmermann. GAF&Co: Connectivity aware topology management for sensor networks. In *Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on*, pages 2260–2264. IEEE, 2009.

[36] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical report, Citeseer, 2001.

[37] S. Zhao, F. Yu, and B. Zhao. An energy efficient directed diffusion routing protocol. In *Computational Intelligence and Security, 2007 International Conference on*, pages 1067–1072. IEEE, 2007.