Universität Tübingen
Rechnernetze und Internet
Prof. Dr. Georg Carle

Animal observation using embedded technologies – Rat movement tracking –

diploma thesis

Chair of Computer Networks and Internet
Wilhelm-Schickard-Institute for Computer Science
University of Tübingen

by

cand. inform.

Corinna Schmitt

Advisor:
Prof. Dr.-Ing. Georg Carle
Prof. Dr. Hanspeter A. Mallot, Chair of cognitive Neurobiology
Prof. Dr.-Ing. Klaus Wehrle, RWTH Aachen
MS CS Olaf Landsiedel, RWTH Aachen

Begin: April 15th, 2006
Accepted: October 15th, 2006

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Tübingen, den 15. Oktober 2006

Abstract

Today biologists are interested in understanding the general behaviour of animals. They want to analyse the social behaviour of animals and want to observe the moving behaviour. For a humane treatment the studies should be done in the natural environment of the animals without disturbing it. The requirements for field studies are different than in laboratories: The hardware of the used sensor motes must be robust, dimensioned corresponding to the animal, consist of enough memory storage to store as much data as possible, must provide communication possibilities, and get along with limited power resources. All those requirements are provided by embedded systems.

In this diploma thesis the technology of the MICA-Family of the Berkeley-Motes are used. They are small and light weighted motes, a radio communication possible, and can also collect sensor readings. The aim of the project is to observe the behaviour of rats corresponding to their meeting behaviour by a static/mobile network. Therefore two algorithms are programmed – "Basestation" and "RatMote". Further more the application side will be programmed to make an analysis of the collected meeting information possible. Finally the functionality is tested in a laboratory.

Heutzutage sind die Biologen am allgemeinen Verhalten von Tieren und dessen Verständnis interessiert. Sie wollen das Sozialverhalten der Tiere analysieren und die natürlichen Lebensbedingungen nicht zu verändern. Die Anforderungen für die Feldstudie sind vielseitig. Die Hardware der eingesetzten Sensorknoten muss robust, entsprechend der Tiere dimensioniert sein, genügend Speicher für maximale Daten besitzen, eine Kommunikation ermöglichen und mit limitierten Energieressourcen auskommen. Alle diese Anforderungen werden durch eingebettete Systeme erfüllt.

In dieser Diplomarbeit wird die Technologie der MICA-Familie, die zu den Berkeley-Motes zählt, eingesetzt. Diese Motes sind klein und leicht, eine Funkkommunikation ist möglich und sie können auch Sensordaten sammeln. Das Ziel des Projects ist es, das Verhalten von Ratten entsprechend ihres „Treff"-Verhaltens mit Hilfe eines stationären/mobilen Netzwerkes zu beobachten. Hierzu werden zwei Algorithmen programmiert – „Basestation" und „RatMote". Weiterhin wird die Applikationsseite implementiert um eine Analyse der gesammeltem „Treff"-Informationen zu ermöglichen. Abschließend wird die Funktionalität mittels Laborversuchen getestet.

Contents

1. Introduction

This chapter introduces the goals of this diploma thesis and characterizes the structure of this document.

1.1. Goal of this diploma thesis

For research projects in the field of the social behaviour of wild animals and their habitats it is essential to make extensive observations in nature. These observations are often not possible without deep impacts on the environments of animals that again distort the results. One should therefore strongly avoid affecting the environment of animals.

New technologies in the field of highly embedded systems allow observing animals in their natural environment and in their burrows with help of radio networks' technology. Thus, one can analyse the social behaviour of animals without disturbing the natural environment. The animals are wearing the mote in a kind of jacket or neckband, which does not handicap them.

The aim of this diploma thesis is to develop a sensor network and implement the corresponding software by using the sensor nodes of the Mica-family developed by the University of Berkeley. The motes are small sensors that can be used in several enviroments to observe the current conditions. In cooperation with the department of cognitive neurobiology such a network is tested in a laboratory by observing rats in an arena. First, experiences in the field of developing such a system, collecting information and analysing the accuracy of the sensors are described in this work.

1.2. Structure of this document

The next chapter discusses the background information in the field of informatics and biology. The Mica-Family, especially the sensor nodes Mica2dots, is introduced as well as the used software – TinyOS – and the behaviour of rats in their natural environment. In chapter 3 the steps of the developed algorithm is introduced, including design steps and implementation examples. Chapter 4 deals with the application of the algorithm – the information of the laboratory environment, the experiments and results. Finally a summary is given.

2. Background Information

This chapter is divided in three parts. In the first part the informatic background is introduced followed by the biological background that gives basic information about rats. The last part characterizes related works that also use motes of the Mica2-Family and compares them with our approach.

2.1. Part 1: Informatic background

In this part of chapter 2 basic terms are introduced as well as the technology of the Berkeley-Motes, the TinyOS components and the programming language nesC. For more detailed information take a look at the project "Communication Architecture - Analysis and comparison focusing on communication subsystems -" [Schm05]

2.1.1. Basic terms

The basic terms are: embedded system, sensors, sensor nodes and sensor network.

The technology of **highly embedded systems** deals with very small systems that are embedded in larger systems. Besides a new operating system they need a sophisticated hardware. Although there is not much space available, plenty sensors should be placed. The data is collected and has to be processed and transferred to a base station. Although the systems contain a highly developed communication, the problems of numerous limiting factors (e.g. memory storage, power resources) still have to be coped with. The Berkeley-Motes – Mica2 and Mica2dot – used in this research project concern this group in the same way as the technology of the LEGO Mindstorm Project, Smart-Its, and Scatterweb.

The **sensor nodes** are the basis for the sensor networks. They consist of a great number of single components. Wireless transmission (e.g. infrared, radio frequency) is used to enable communication between the sensors nodes. A node consists of a data recording unit that is responsible for the data collection from the sensors, a computation unit that processes information collected by the sensors, and a radio unit that handles the transmission between different nodes. The energy supply of the sensor nodes is provided by batteries or accumulators. **Sensors** are devices that measure data of the environment (temperature, pressure, light etc.) and can pass them on. The cooperation of the components is shown in figure 1.

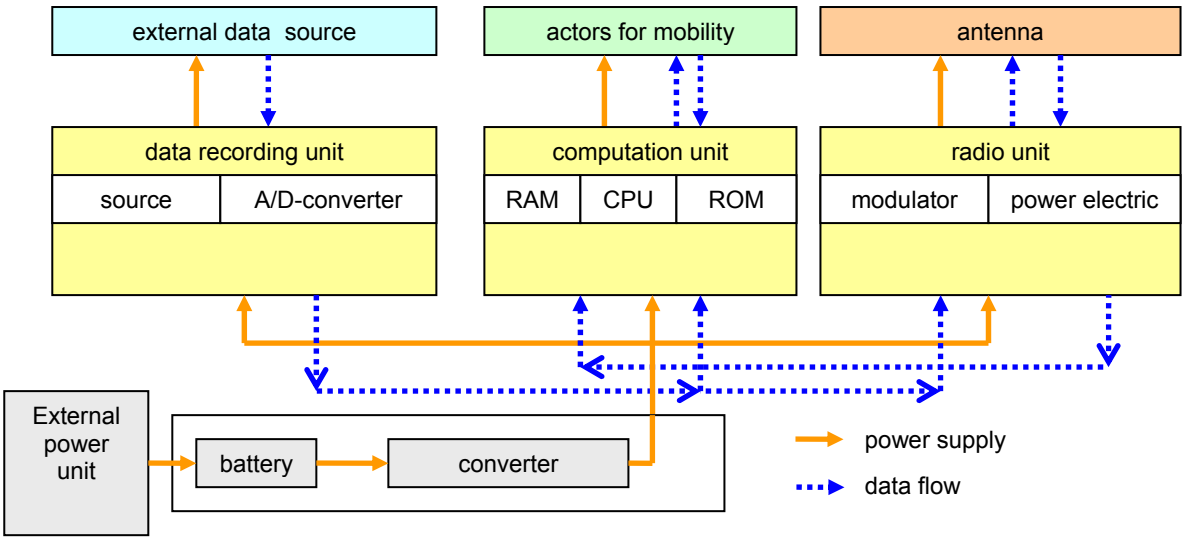


Figure 1: Components of a sensor node

Figure 1 also shows how the energy supply is guaranteed in the sensor nodes. Energy consumption is a limited factor in those systems. The life time of a battery is determined by different factors. Factors, that need to be taken into account, are the dimension of the battery that is limited by the device size, the power rate (discharge), the power tops and the surrounding temperature. Accumulators in combination with solar cells are used to increase the life time of the battery; the system is able to charge itself and thus has enough energy for all components. A converter is used in the sensor nodes which takes energy from the battery and takes care of constant voltage for the connected components.

The development of the **sensor-network-research** started in the 1980s. At that time they used to be called Distributed Sensor Networks, which included DARPA and NSF projects for techniques of signal processing. Those techniques were mostly used in the military field till the late 1990s (e.g. radar systems, audio sensors to track submarines). Due to the process of miniaturization of electronic devices this area of science became more and more interesting, and thus even worthier for research. Nowadays, sensor networks can be found in all kinds of areas, like the observation of the environment, the medical field – Cold Blue Architecture [WeMM04] – , and also in daily situations (e.g. supermarket, the supervision of material). Essential for the components' development of sensor networks are fault tolerance, product and production costs, the area of usage, hardware, communication, location, and the energy consumption are. The criteria mentioned above also set a limit to the possibilities.

A **sensor network** (Fig. 2) is a network which consists of different components. The integrated sensor nodes bild a sensor patch where every one can communicate with each other. Due to the fact that sensor networks often operate without cable, messages are transmitted via broadcast. In the communication field two types are divided: If the transmission is only for one special receiver than it is addressed. If the transmission should be for every one in the neighborhood than it is send via broadcast without addressing. All the information of the sensor nodes are transmitted to a sensor gateway which is responsible for the transmission to a base station. The base staion handles the received data on to the internet where a remote user is able to get the needed data and analyses them.

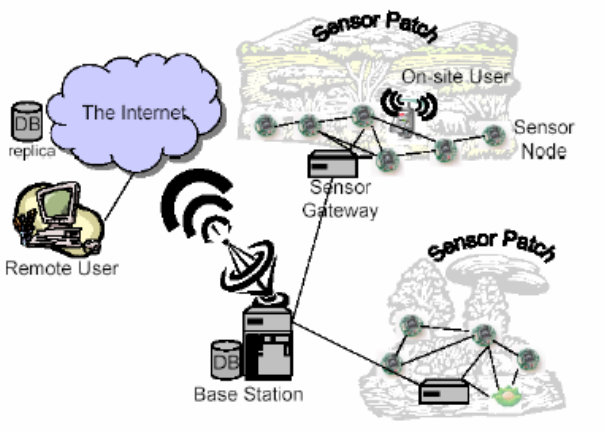


Figure 2: A sensor network consists of different systems [1]

Most of the time those networks are so called **ad hoc networks** (spontaneous network connections). Those "ad hoc networks" distinguish themselves through their spontaneous connection to local networks. Moreover, they are easily connectable to local services. Due to the wireless communication, nodes have a limited connectivity, which means the transmission could be disturbed, or there is a piece of paper in front of the optical channels of nodes, so that no signal can be broadcasted or received. Another problem of those networks lies within the security and anonymity of data sent by the nodes.

Sensor networks need synchronization to timely organize the results taking place at different sensors. Commonly logical time is used, which is generated through a beat giver.

2.1.2. The technology of the Berkeley-Motes

In 2000 Jason Hill, and David Culler from the University of California at Berkeley/USA developed an operating system – the **TinyOS**, which has been financed by DARPA, and is one of the most prevalent ones in the field of sensor motes nowadays. TinyOS is a noval program module based on components. **Berkeley Motes** are the corresponding hardware platform.

Berkeley Motes (Fig. 3) contain a microprocessor including flash-memory with "Static Random Access Memory" (SRAM), and "Electrically Erasable and Programmable Read-Only Memory" (EEPROM) as its Kernel. Actor- and sensor-components are attached to this Kernel. At this it is more precisely a co-processor, which is essential for programming the flash-memory, radio-interfaces, and serial-interfaces.

Figure 3: Berkeley Mote [HSW02]

[1] http://medien.informatik.uni-ulm.de

The **radio-sender/-receiver** is a simple transceiver with an antenna, where the sending- and receiving-unit can be altered. Three control modes are possible – sending, receiving, and shut down. This chip contains no integrated backup memory, so that every transferred Bit in the main memory has to be processed at once. Another disadvantage is that transferred data cannot be saved for a long time because the storage on a mote is limited. It is also important to keep in mind that one data stream can be send at once.

Throughout the years newer versions have been developed, which contain a faster master control unit, a bigger flash-memory, and RAM. Further improvements are the sensors that have been moved to a so called Sensor Board, which is linked to the main board via a bus. The development was fast and the main characteristics are shown in table 1.

Mote	mica	Mica2	Mica2dot
Mikrocontroller			
Type	Atmega128		
Program memory (KB)	128		
Ram (KB)	4		
Active Power (mW)	8	33	8
Sleep Power (μW)	75	75	75
Wakeup time (μs)	180	180	180
Nonvolatile Storage			
Chip	AT45DB041B		
Connection Type	SPI		
Size (KB)	512		
Communication			
Radio	TR1000	CC1000	CC1000
Data Rate (kbps)	40	38.4	38.4
Modulation Type	ASK	FSK	FSK
Receive Power (mW)	12	29	29
Transmit Power at 0 dBm (mW)	36	42	42
Power Consumption			
Minimum Operation (V)		2.7	
Total Active Power (mW)	27	89	44
Programming and Sensor Intervace			
Expansion	51-pin	51-pin	19-pin
Communication	*	*	*
Integrated Sensors	no	no	no

* IEEE 1284 (programming) and RS232 (requires additional hardware)

Table 1: Development of the hardware platforms [Körb06]

2.1.2.1. The Mica2

The Mica2 Mote family is produced by Crossbow Technology Inc. [4] in the USA and is the third generation mote module that was developed for sensor networks. The main characterizations are low-power, wireless communication and compatibility with Mica2Dot. For the experiments the Mica2 sensor (Fig. 4) is used with the radio frequence 916MHz. Figure 6 shows the architecture of the mote in a block diagram and the table 2 shows the corresponding features of the processor/raio board.

Figure 4: Mica2 sensor [4]

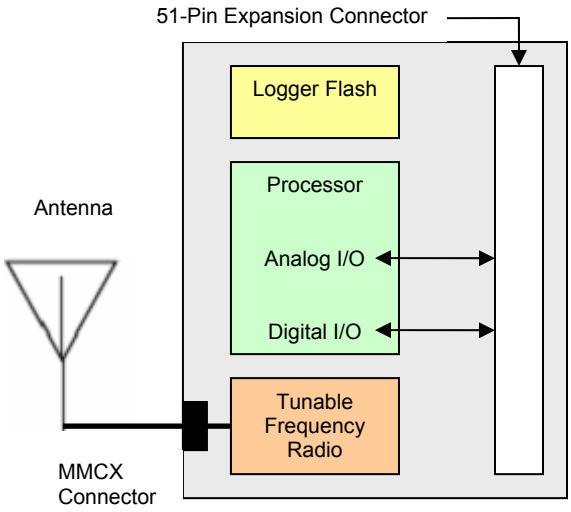


Figure 5: MPR400CB Block Diagram [4]

"The MPR400CB is based on the Atmel Atmega128L. The Atmega128L is a low power microcontroller which runs TOS from its internal flash memory. Using TOS, a single processor board (MPR400CB) can be configured to run your sensor application/processing ans the network/radio communications stack simultaneously. The MICA2 51-pin expansion connector supports Analog Inputs, Digital I/O, I2C, SPI and UART interfaces. These interfaces make it easy to connect to a wide variety of external peripherals."

Crossbow Technology Inc. [4]

Processor Reformance	
Program Flash Memory	128K bytes
Measurment (Serial) Flash	512K bytes
Configuration EEPROM	4K bytes
Serial Communications	UART
Analog to Digital Converter	10 bit ADC
Other Interfaces	DIO,I2C,SPI
Current Draw	8mA
	<15μA

Multi-Channel Radio	
Center Frequence	868/916 MHz
Number of Channels	4/ 50
Data Rate	38.4 Kbaud
RF Power	-20 to +5 dBm
Receive Sensitivity	-98 dBm
Outdoor Range	500 ft
Current Draw	27 mA
	10 mA
	<1 μA

Electromechanical	
Battery	2x AA batteries
External Power	2.7 - 3.3 V
User Interface	3 LEDs
Size (in)	2.25 x 1.25 x 0.25
(mm)	58 x 32 x 7
Weight (oz)	0.7
(grams)	18
Expansion Connector	51-pin

Table 2: Processor/Radio Board from MPR400CB [4]

2.1.2.2. The Mica2dot coin

The Mica2Dot coin (Fig. 6) is similar to the Mica2 but smaller. Its size is only 25mm, like an American Quarter. Because of this feature it is better suited for commercial development. For the experiments the processor and radio platform MPR500CA are used. The table 3 shows the corresponding features of the processor/radio board.

Figure 6: Mica2dot [4]

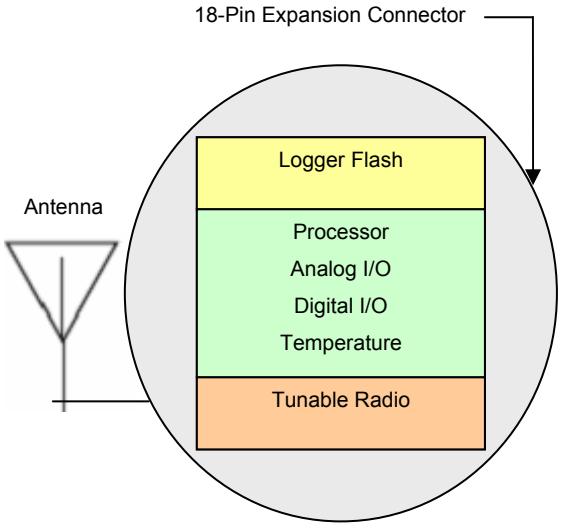


Figure 7: MPR500CA Block Diagram [4]

"The MPR500CA is based on the Atmel Atmega128L. The Atmega128L is a low power microcontroller which runs TinyOS (TOS) from its internal flash memory. Using TOS, a single processor board (MPR500CA) can be configured to run your sensor application/processing ans the network/radio communications stack simultaneously. The MICA2DOT features 18 solderless expansion pins for connecting 6 Analog Inputs, Digital I/O, and a serial communication or UART interface. These interfaces make it easy to connect to a wide variety of external peripherals."

Crossbow Technology Inc. [4]

Processor Reformance	
Program Flash Memory	128K bytes
Measurment (Serial) Flash	512K bytes
Configuration EEPROM	4K bytes
Serial Communications	UART
Analog to Digital Converter	10 bit ADC
Other Interfaces	DIO
Current Draw	8mA
	<15μA

Multi-Channel Radio	
Center Frequence	868/916 MHz
Number of Channels	4/ 50
Data Rate	38.4 Kbaud
RF Power	-20 to +5 dBm
Receive Sensitivity	-98 dBm
Outdoor Range	500 ft
Current Draw	27 mA
	10 mA
	<1 μA

Electromechanical	
Battery	3V Coin Cell
External Power	2.7 - 3.3 V
User Interface	1 LED
Size (in)	1.0 x 0.25
(mm)	25 x 6
Weight (oz)	0.11
(grams)	3
Expansion Connector	18 pins

Table 3: Processor/Radio Board from MPR500CA [4]

In our approach a sensor board is used that consists of a microphone, photo sensor and a accelorometer. The accelorometer is not activated in our case. If the photo reading is zero, the sensor was in a dark area, otherwise its value can rise up to 65535 = “high brightness”. The value of the microphone reading stands for the brightness (= zero) or darkness (= 65535) of the vibration and its value is Hz.

2.1.2.3. The Mote Interface Board

The mote interface board (Fig 8) is needed to get the users program on the mote. In our case the MIB510CA Mote Interface Board, shown in figure 9, is used. It has an interface for Mica2dots as well as an interface for Mica2 motes. It has a serial interface for both programming and data communication that is shown in the corresponding block diagram in figure 9.

Figure 8: MIB510CA Mote Interface Board [4]

“The MIB510CA allows for the aggregation of sensor network data on a PC as well as other standard computer platforms. Any MICA2 node (MPR4xx) can function as a base station when mated to the MIB510CA serial interface board. In addition to data transfer, the MIB510CA also provides an RS-232 serial programming interface.”

Crossbow Technology Inc. [4]

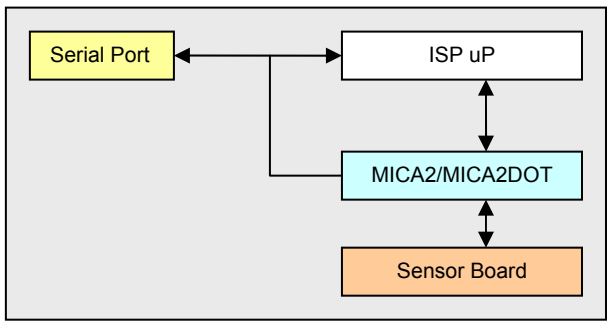


Figure 9: MIB510CA Block Diagram [4]

2.1.2.4 Power Management

The power resources of the motes are limited. The Mica2 motes work with two AA batteries. If the mote is awake the processor needs 12mA; if it sleeps it needs only 1 microA. For a communication over 100m the needed power is 1.8mA. If it is in the receive mode, it needs 1.8mA. Similar power consumption occurs by Mica2dot motes which work with coin batteries. If it is awake the processor needs 8mA; if it sleeps it needs only 15 microA. The radio needs 25mA for a transmission over 300m and for the receive mode 8 mA. The motes are need for colleting sensor readings which require power as well. Table 4 shows the energie consumption for sensor readings.

Sensor	mA
Photo sensor	1.235
I²C temperature sensor	0.150
Sensor for barometric pressure	0.010
Humidity sensor	0.775
Thermophile sensor	0.170

Table 4: power consumption of different sensors [Pola03]

Thus, it is important to work with power saving methods. The idea is to wake up the components only, if they are needed. The mote can be running for several weeks or month, like in the projects described in chapter 2.3. Another possibility to increase the running time is to use accumulators which can be recharged by solar cells. This approach is only useful in sensor netzwork which are static.

2.1.3. Software

The bottom layer in the “Berkeley” Stack (Fig. 10) is the previously introduced hardware. The second layer - the Operating System (OS) - is characterized in this chapter. It consists of TinyOS and nesC which together compose the programming environment for the sensor motes. The application layer is introduced in chapter 3.

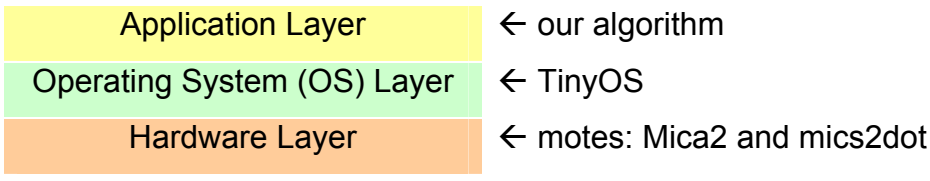


Figure 10: Structure of the Berkeley Stack [Körb06]

2.1.3.1. TinyOS

In general TinyOS (Fig. 11) is the operating system which is used for hardware with limited resources (e.g. Berkeley Motes). TinyOS has an efficient multithreading engine, which is composed of a two-level-schedule, which realizes the computer-time-spreading for threads.

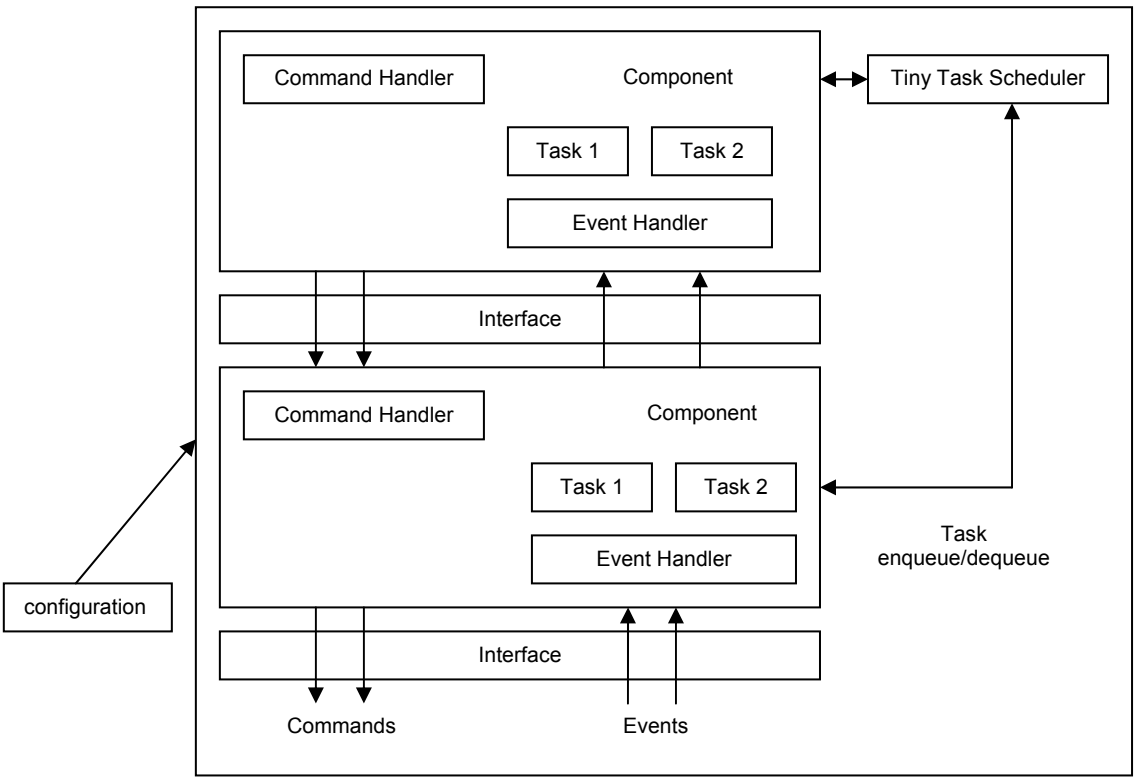


Figure 11: TinyOS Schema [Körb06]

The **two-level-schedule** consists of:

- **Tasks**, which contain current network routing and data preparation. They thus have a longer lifetime, which is shortly paused by hardware events.

- **Events** on the other hand must be handled immediatly, so that a long-time blockade caused by current applications, and data loses are prevented.

It is important to take the consumption of power of the main board into account to guarantee an efficient way of processing parallel running data streams.

The so called components merge the sensors, the processing-, and communication-units. They contain a Command Handler, and an Event Handler, that make an immediate reaction to a change in state possible.

Components of a higher level communicate with the ones on lower levels via the **Command Handler**, which send a return value after execution or dispatch about success or failure. Thus, a Command cannot cause Events. The **Event Handler** deals with Events, which

can be caused directly or indirectly by hardware-interrupts. Events can even cause Events of higher levels. Frames contain the data memory of a component. Within a component Tasks are running, which have been started by Commands, Events, or other Tasks from the same component. Typical features of Tasks are the atomic structure, and the "run-completion" semantic, i.e. the Task cannot be stopped and stops itself when finished. The advantage of this feature is that TinyOS can get along with one Memory-Stack.

This knowledge allows a classification of the components in three types:

- **Hardware abstraction components**: Mapping of physical hardware onto a components model. One of the components is the Radio-Frequence- Module (RFM), which operates the pins of the RFM sender/receiver, and informs other components of successful transmission via Events.
- **Synthetic hardware components**: Mapping of progressive hardware, e.g. the radio-byte-hardware, which sends data to the RFM, and gives a signal when a whole byte has been sent.
- **High level software components**: They contain the application-logic. They control components, and calculate the routing, and other data information.

This kind of operating system cannot execute several applications simultaneously. But due to the Event-based programming, an operational capacity during applications with a high degree of parallelism can be reached. The used programs are realized by a special programming language – nesC.

2.1.3.2. nesC – Introduction to the programming language

As mentioned above the operation system is TinyOS which is programmed in nesC. NesC is a derivative of the programming language C, designed for TinyOS. The basic concepts are

1. Separation of construction and composition
2. Specification of component behaviour in terms of set of interfaces
3. Interfaces are bidirectional
4. Components are statically linked to each other via their interfaces.
5. NesC is designed under the expectation that code will be generates by whole-program compilers.
6. The concurrency model of nesC is based on run-to-completion tasks and interrupt handlers which may interrupt tasks and each other.

In the following part the application example Blink is used to show how the software is characterized. The programs in TinyOS consist of several **components** which are linked to build the whole program. The next figure shows the file Blink.nc (Fig. 12) that deals with the configuration of the components.

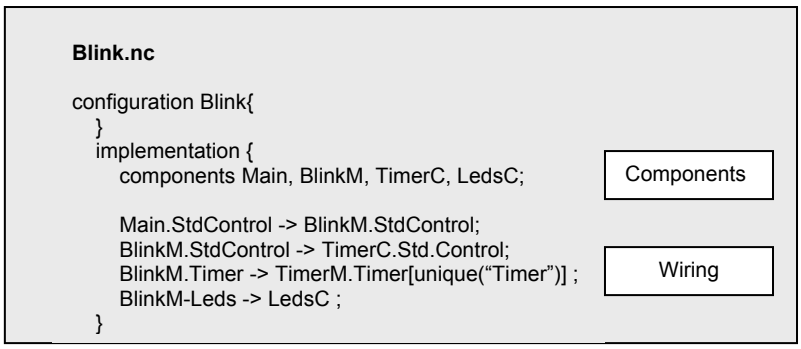


Figure 12: Code for Blink.nc [Körb06]

The components consist of two parts: One part is needed for the specification of the component and the second is needed for the implementation. Tasks are the internal correspondence to the components. Threads deal with control and are rooted either in a task ore hardware interrupt.

The **interfaces** define a pool of functions that consist of commands and events. Therefore an interface can represent a complex interaction between several components. The interfaces can be divided in two groups:

1. **Provided interfaces**: They represent the functionality of the components which it provides to the user.

2. **Used interfaces**: They represent the functionality of the components which is needed by the components to perform its job.

Figure 13 shows the interaction of those two interface groups with help of the program Blink (Fig.14).

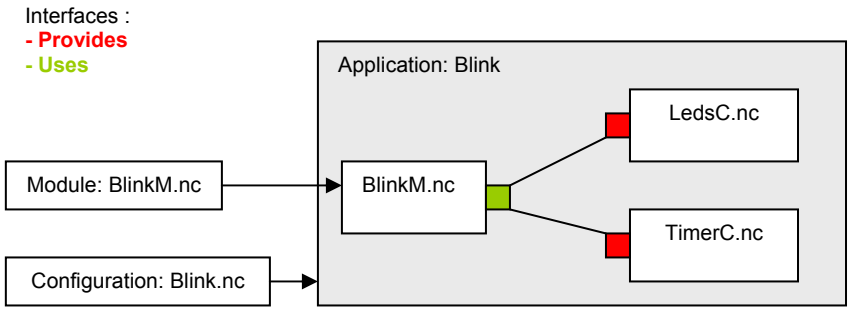


Figure 13: Interactions between interfaces [Körb06]

BlinkM.nc

```
module{
   provides {
            interface StdControl;
   }
   uses {
            interface Timer;
            interface Leds;
  }
}
implementation {
   command result_t Std.Control.init(){
            call Leds.init();
            return SUCCESS;
   }
   command result_t Std.Control.start(){
            return call Timer.start(TIMER_REPEAT,1000);
   }
   command result_t Std.Control.stop (){
            return cal Timer.stop();
   }
   event return_t Timer.fired() {
            call Leds.redToggle();
            return SUCCESS;
   }
}
```

Interface

Commands

Events

Figure 14: Code for the module BlinkM.nc [Körb06]

The nesC-compiler makes an application file app.c out of the nesC-files. Afterwards the C-compiler builds an executable file main.exe. Now the program can been run and the functionality can be tested. To visualize the cooperation of the different components of the program a graphic program can be used that is placed by the software TinyOS. It produces a corresponding component graph that shows the wiring up of the components. The circles stand for the components and the arrows show the wiring of the components. On the arrows the interfaces for the wiring can be found. The component graph for Blink.nc is shown in the following figure 15.

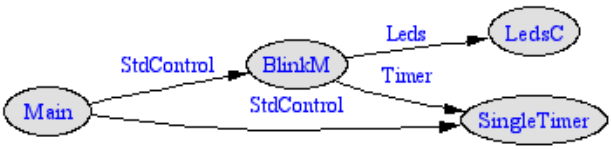


Figure 15: Component graph – Blink.nc

2.1.3.3. AVRORA – The AVR Simulation and Analysis Framework

In this section the simulator AVRORA is shortly introduced. **AVRORA** is *"a research project of the UCLA Compilers Group, is a set of simulation and analysis tools for programs written for the AVR microcontroller produced by Atmel and the Mica2 sensor nodes. Avrora contains a flexible framework for simulating and analyzing assembly programs, providing a clean Java API and infrastructure for experimentation, profiling, and analysis"* [TiLP05].

The concept of AVRORA is to make an emulation possible before using the real motes. It allows the user to observe single nodes and complex sensor netzworks. Additionally several monitorings (i.e. packet) for a better analysis of the tested algorithm are possible. The simulator also allows a detailed observation of the algorithm during the running simulation. Another advantage is that the activity of the LEDs is displayed as well as the structure of the send packets.

```
Avrora -simulation=sensor-network -monitors=packet -seconds=5.0 -
nodecount=1,1 CntToRfm.od RfmToLeds.od
Avrora  Beta 1.6.0] - (c) 2003-2005 UCLA Compilers Group

Loading CntToRfm.od    OK: 2.032 seconds]
Loading RfmToLeds.od    OK: 0.673 seconds]
=={ Simulation events}=====================================================
Node           Time   Event
------------------------------------------------------------------------
   1           2028  Red: on
   0           2376  Red: on
   1           2030  Yellow: on
   0           2378  Yellow: on
   1           2032  Green: on
                                    ...
   0       18839581  Packet sent: 33:CC:FF:FF:04:7D:03:0A:00:00:7E:57:57:57:
   1       20777676  Red: on
   0       20787229  Packet sent: 33:CC:FF:FF:04:7D:03:0B:00:00:4E:60:60:60:
   1       22458061  Red: off
   1       22458077  Green: off
   1       22458090  Yellow: on
                                    ...
========================================================================
Simulated time  36864000 cycles
Time for simulation  3.198 seconds
Total throughput  23.054409 mhz
Throughput per node  11.5272045 mhz
=={ Monitors for node 0
}=========================================================
Bytes sent  904
Packets sent  20
=={ Monitors for node 1
}=========================================================
Bytes sent  67
Packets sent  1
% _
```

Figure 16: Exemplary result of the simulator Avrora for a sensor [TiLP05]

2.2. Part 2: Biology background

In biological testing the most common animal used in laboratories are rats. They originaly live in India, south east Asia and Australia, but nowadays they are spread all over the world, because of their relation to human beings and their travelling behaviour.

Kingdom:	*Animalia*
Phylum:	*Chordata*
Class:	*Mammalia*
Order:	*Rodentia*
Suborder:	*Myomorpha*
Family:	*Muridae*
Subfamily:	*Murinae*
Genus:	*Rattus*
Species:	- *Rattus rattus* - *Rattus norvegicus*

Table 5: Scientific classification of the rat [WhKo05]

2.2.1. General physiology of the rat

The black *Rattus (R.) rattus* is 16-23cm long from head to trunk. It is smaller than the grey-brown *R. norvegicus* with 18-26cm. An important distinguishtion between the species is the relation between the length of head-trunk to the length of the tail. The tail of the *R. rattus* is longer as the body and the tail of the *R. norvegicus* is normally shorter than the body. Another typical characteristic for the species is the ears' length.

Rats have a pointed snout and a splited upper lip. A rat just has one incisor and three grind cogs per jaw half, all in all 16 cogs. The incisors have no root and grow life long, thus they need to be shortend by gnawing. A big gap exists between the incisors and the grind cogs.

The rat's stomach consists of two parts – the gastro-oesophageal vestibule and the corpus ventriculi. The gastro-oesophageal vestibule has a gland less mucous membrane and the corpus ventriculi a common stomach mucous membrane. Both parts are divided by a mucous membrane wrinkle where the gullet ends. Therfore the mucous membrane wrinkle is the connecting part of the gullet and the stomach, which causes the rat's inability to vomit. The natural food consists of heavy digest nourishments and is digesting in the big appendix.

The rat's feet and the long tail are less hairy. The rat has five toes at its hind feet and four at its fore feet, because of the thumb's rudimental development.

Rats do not have perspiration glands; it gives up the heat at the less hairy parts like the snout and the ears. The harderian gland, that produces a secretion consisting of porphyrin, is next to the corner of the eye. It is distributed during the cleaning.

The rats have a big visual field, due to their eyes being placed at the sides of the skull. Although there are overlapping visual fields it is hardly believed that there is a sterosopic vision. The olfactory sense is very marked. It is used to recognize the members of the pack as well as for food-searching. With help of the smell sense the susceptibility of females can be determined and the exertion of the animals. The sense of hearing is very good, and ranges from 1 to 80kHz. Their social communication mainly takes place in the ultrasonic area. The balance organ is physically fit and very complex. The rat's taste sense is physically fit as well. The touch hairs at the head around the snout and the eyes are needed for orientation.

2.2.2. Characterization of "*Rattus norvegicus* Long-Evans"

The experiments are done with the rat "*Rattus norvegicus* Long-Evans" (Fig. 17). The strain was introduced by Drs. Long and Evans in 1915 after cross-breeding Wistar albino females with a wild grey male.

Figure 17: *Rattus norvegicus* Long-Evans [6]

The rat's characteristic appearance is a small variety of white with black (or occasionally brown) hood over head and back of neck, with a line down the back. The dimension (Fig. 18) of a rat from the nose to the end of the tail is 38cm.

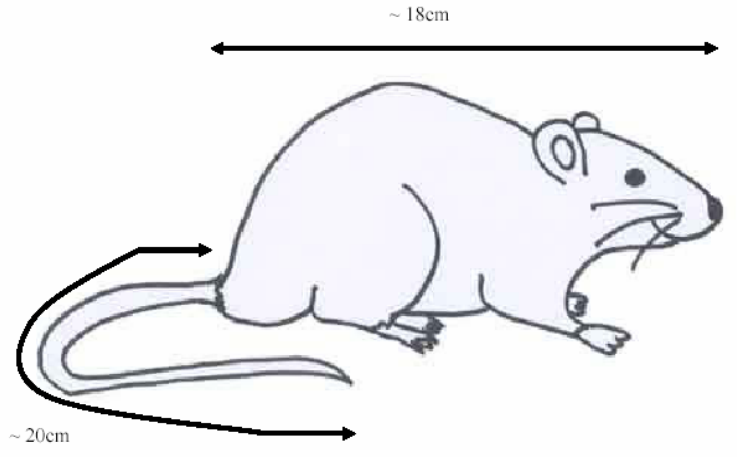


Figure 18: Dimension of a rat

Rats show special behaviours that make them useful for scientific applications. They are omnivorous, very intelligent, can be taught complicated manipulations and are nocturnal animals. A further advantage is that they are very tolerant to single caging. Those rats are quiet and easy to handle. Their life cycle can rise up to three years. Other steps of the rat's life cycle are presented in table 6 below.

Age (days)	Weight (g)	Development
Birth	5	hairless, toothless, closed eyes and ears
4	10	hair begins to appear
10	.	covered with hair
13	.	eyes and ears open
21	30-50	weaning
40-50	150-200	sexual maturity
72	200 - 400	young adult
250-300	270 - 800	full grown; males larger than females

Table 6: Life cycle of a rat

The living conditions for the experimantal rat consits of the following demands [5]:

- cage size 40cm x 25cm x 15 cm
- overcrowding must be avoided
- cages need to be changed at least weekly
- much bedding material like woodchips, paper shavings and tissues
- temperature 21-22°C
- humidity 50-65%
- good ventilation
- controlled lightning (12 hours day, 12 hours dark)
- backround sounds to avoid startel reactions
- pelleted diets and enough water as food

2.2.3. Natural behaviour of the rat

One important but often neglected behavior of the rat is its vocal communication. It lies in the ultrasonic area and cannot be heard by humans. During a fight the rats whistle, clatter with their tooth and screech. If the rat is underlaing it makes a loud shout. They show aggression and fear with the help of tooth clatter. Young rats also whistle for their mother. During a play they whistle and squeak.

It is also frequently observable that rats like to scuffle. Normally it is a harmless little fight as a game. Sometimes they fight for the position in the group. During the play they jump on each other (Fig.19 A), knock over and climb above each other (Fig.19 B). If the play becomes more serious they squeak a little bit. The underlying rat turns on its back and shows the winner its neck (Fig.19 C). The winner bites into the neck playfully and after all they clean each other. They jump on each other to strengthen their position in the group.

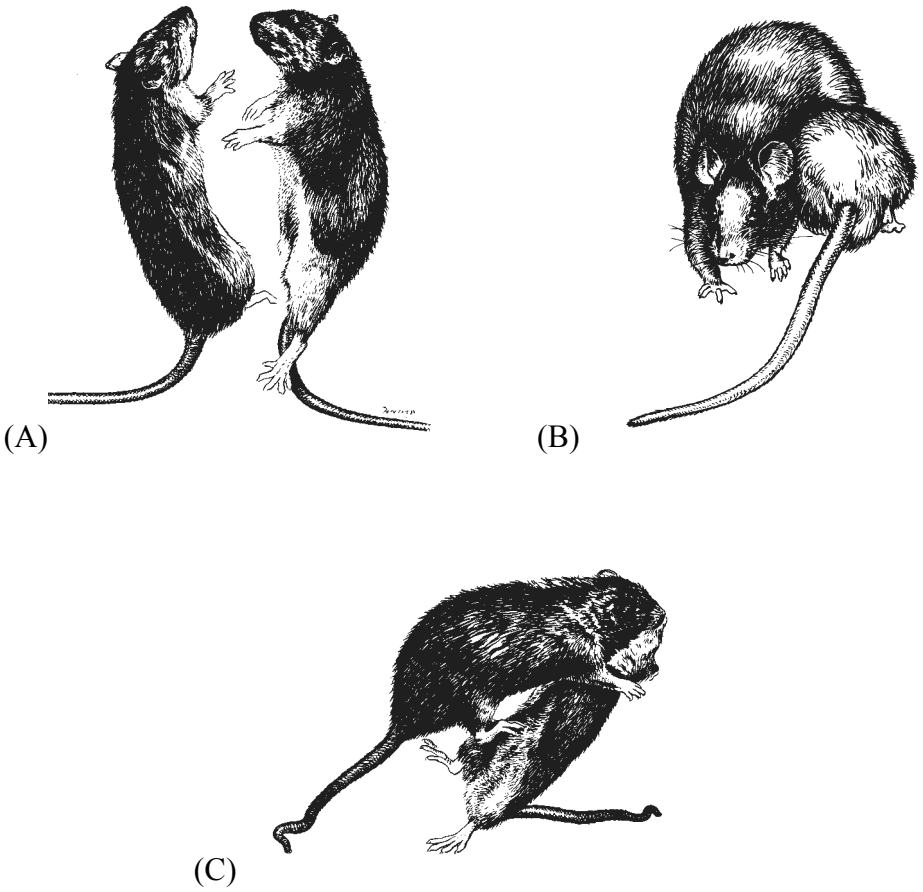


Figure 19: Playing rats [TiLP05]

In the neighborhood of rat population a social organization can be found. Females with young animals, larger males and near-term females are dominated. These groups have access to good burrow sites and food. The dominate males mark the perimeter with urine and work like a "patrol". Animals are not able to live in their population if they are subordinated and wounded; they need to find another living room.

The rats live in underground burrows (Fig.20). Normally the burrow consists of many chambers that are connected with tunnels. The entries may change because of the growing population or when new segments are digged. The structor of the complex tunnel system is related to the social relationships in the population. The burrows consist of

- ~ 6.8 entrance
- ~ 16 tunnels
- ~ 4.5 chambers: cavities, nests

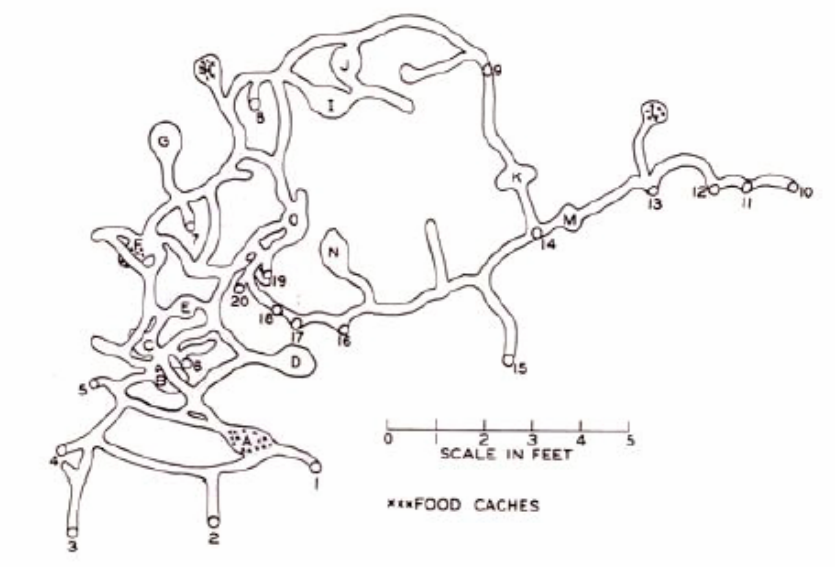


Figure 20: Burrow system [Calh63]

The entrance normally is in a covered place and may be hidden by cut grass or dirt. The tunnels are straight segments with a median width of 8.3cm, so that only one rat can pass the tunnel at a time. The median length is 29.8cm. After this it is spread in chambers or in dead-end. The cavities are small chambers that can accomondate up to 11 rats. The chambers are devided by their uses into nest cavity or food-cache. The nest consists of shredded material and serves as a sleeping place. It is devided in three types:

1. "pad" – simplest kind of nest build of shredded material
2. "cup-shaped nest" – made with finer texture grass or spredded paper
3. "hooded nest" – organized structure with hollow sphere and just one opening (e.g. for mothers and babies

The rat can breed throughout the year if the surrounding conditions are good and because of the short gestation period of only 21-23 days. The average litter of a female rat is five and every one can consist of up to fourteen, normally seven. A rat can become three years old, if it survived the first year because the mortality rate is 95%. If enough food is available, they often stay within 20 meters of their nest, otherwise the rats tend to wander extensively.

2.3. Related works

Today sensor motes, comparable to the once used in this thesis, are used to gain information about the environment. One field of expertice is "**structural health monitoring**" (SHM). It observes the status of the structural health of a building. It is used to analyze the changes in the structure of the building, which can affect the performance, and to observe the behavior of animals in their natural environment.

The following sections characterize three different projects of "structural health monitoring" – the "Golden Gate Bridge Project", the "Great Duck Island Project" and the "ZebraNet Project"; finally a comparison to our approach is made.

2.3.1. The "Golden Gate Bridge Project"

A example is the **"Golden Gate Bridge Project"** in San Francisco, USA [Kim05]. This project deals with the SMH-problem as discussed above. The aim is to observe the ambient vibration of the structure. The scientists work with Berkeley-Motes which are placed at different points at the Golden Gate Bridge as shown in the following figure 21.

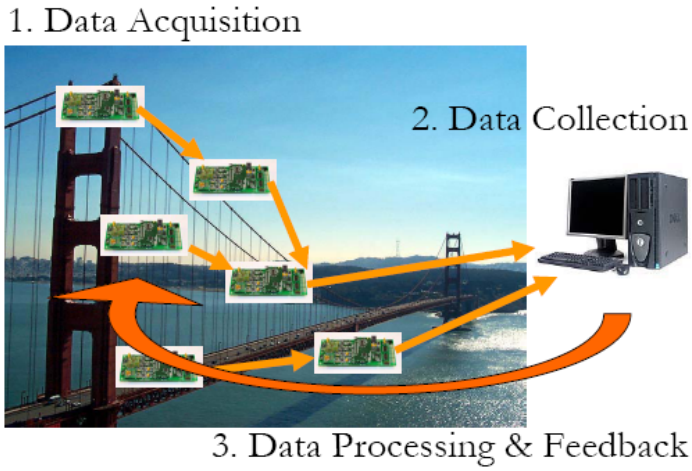


Figure 21: Experimental arrangement of the Golden Gate Bridge-Project [Kim05]

In this project the first idea was to use a Mica2 mote. The used sensor board is an accelerometer board (Fig.22) that consists of four accelerometers (2 ADXL 202E and 3 Silicon Designs 1221L), one thermometer and four analog to digital converters (ADC).

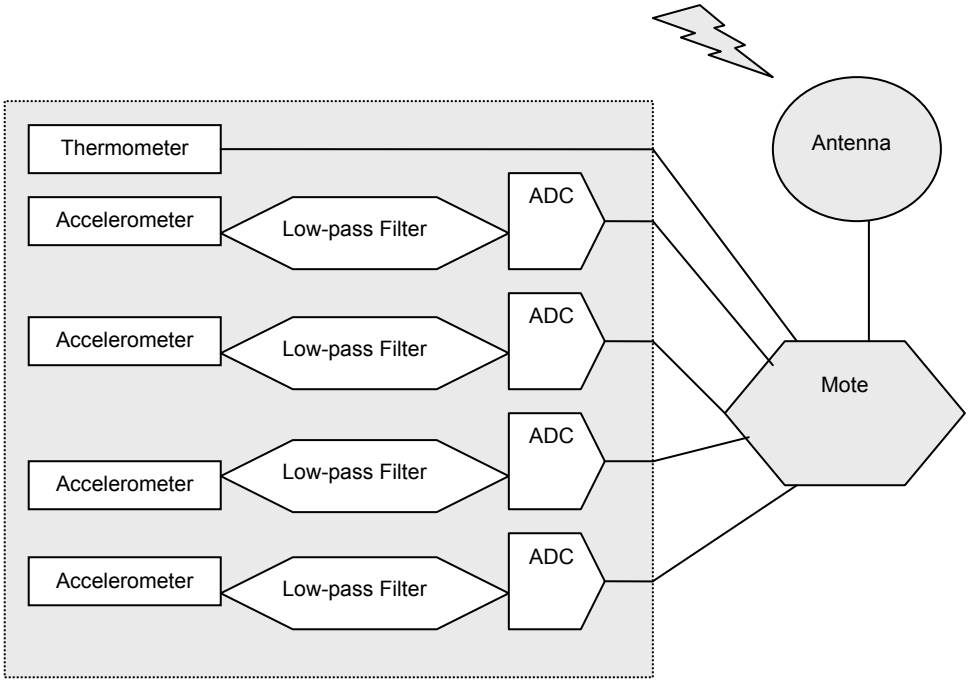


Figure 22: Block diagram of the hardware used in the "Golden Gate Bridge Project" [Kim05]

The Golden Gate Bridge is located in an area where often earthquakes and sharp sea winds occure, thus the construction needs to be stabil and must hold out strong vibrations. Due to the construction of the bridge it is essential to measure two directions perpendicular to the bridge span – one up-down and one across the span. For the towers two directions must also be measured – along the span and across the span. Two types of accelerometer are needed, because the ADXL 202E can measure in directions perpendicular two each other and the other type consist of the axis. The ADXL 202E has a range from -2G to 2G for big movements, i.e. earthquakes, and the other type has a narrow range of -0.1G to 0.1G to sample ambient vibrations.

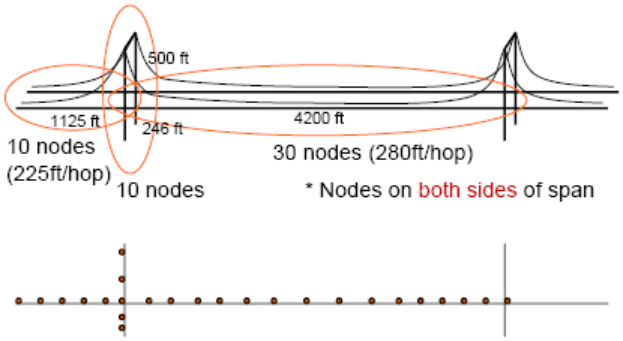


Figure 23: Deployment plan for the Golden Gate Bridge [Kim05]

The arrangement (Fig. 23) of the sensors and the antenna dimension is based on the Golden Gate Bridge construction. The collected information is transmitted via hops. The longest hop distance is 280 ft, because 30 nodes are spread over the whole distance. This distance can be overcome by using bidirectional antennas. This type is not compatible with the Mica2 mote, but with the micaZ mote that uses standard IEEE 802.15.4 radio running at 2.4GHz. A compromise must be made between the needed power for the transmission and the number of needed hops, because of the less power resources. The mote transfers data one

third of its time; otherwise is in an idle state. With this requirements and a usage of three Tadrian 5930 batteries supplying 3.6V a working period of three weeks can be realised.

As mentioned above the requirements of the used wireless sensor network is different to a deployment in a protected area like in a laboratory. Important for reliable information is a high accuracy of sample, high-frequency sampling, time synchronisation, large-scale multi-hop network, reliable command dissemination and reliable data collection. All those requirements can be realized when the operation of data collection is divided into three phases:

1. **Data Acquisition**: samples the vibration data of the structure, and logs it into EEPROM
2. **Data Collection**: transfers data reliable to an external computing resource
3. **Data Processing & Feedback**: runs analysis algorithm, determines health status, and sends feedback to nodes if needed.

The first three challenges are related to the first phase – the data acquisition. It is realized with regular sampling by uniform intervals. The sampling frequency lies by 200Hz. The component FTSP is responsible for the time synchronisation that provides 67μs error over 59- node 11-hop network.

The next three challenges relate to data collection. The component TinyOS MintRout is used for information replay and allows large-scale multi-hop network. For a correct structural analysis a complete data collection which is achived by retransmission is important. The data collection is also responsible for the needed sampling cycles. It is important to keep in mind: the less computational power, the less memory storage and the less energy. The service STRAW – Scalable Thin and Rapid Amassment Without loss – works with multi-hop on good scalability and optimizes the interval between the packets [Kim05]. It collects data reliably from a mote to PC over multi-hop network.

In the last phase the PC collects the transmitted data, analyses them, and runs a diagnosis of the structural health. If, for instance, a new calibration is essential, a feedback is send to the nodes.

This sensor network and the strategy are already successfully tested on a small footbridge from the City of Berkeley to Berkeley Marina. It is right now running at the Golden Gate Bridge, but results are not published yet.

2.3.2. The “Great Duck Island Project”

Another approach is the “**Great Duck Island Project**” in Maine, USA [17, 18], where biologists observe the microclimates in and around nesting burrows of Storm Petrels – a kind of sea bird. The researchers observe the occupancy of the burrows and the conditions of the burrows surrounding. The aim of this project is a development of a habitat monitoring kit. They put sensor devices from the Mica-Family in the underground nest (Fig.24.1) as well as on 4-inch stilts (Fig.24.2) outside their burrows.

The devices record data about the birds and their behaviour. These are send to a gateway node (Fig.24.3), that transmits the data to a computer or laptop in the research station

(Fig.24.4). After that it is transferred to a satellite dish (5) to transmit it to the base laboratory at Berkeley, USA.

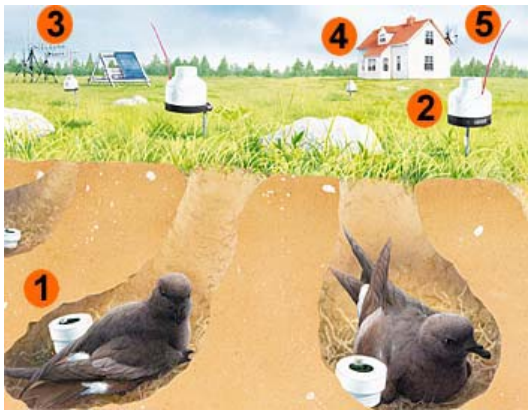


Figure 24: Experimental arrangement [Bits06]

This project uses the Mica2dots and the Mica2 motes. They are equipped in this project with the following sensors:

- Non-contact temperature module – Melexis MLX90601
- Temperature and humidity sensor – Sensirion SHT11
- Light sensor – TAOS TSL2550
- Sensor for barometric pressure – Intersema MS5534A

Figure 25: Weather mote (left) and burrow mote (right) [SMPC04]

The sensor network consists of 30 sensor motes, which are divided into two groups (Fig.25) – the burrow motes and the weather motes. The burrow motes observe the occupancy of the burrow using a non-infrared thermophiles and temperature/humidity sensor. The weather mote is used for monitoring the surface microclimate.

The small size of the motes and their autonomous running allows the biologists to observe the habitat without disturbing the current ecosystem. The motes should observe the habitat nine month long. As shown in figure 24 the Mica2dots are placed in the burrows and the Mica2 motes are placed outside. The power resources are limited by the capacity of the batteries, thus the battery provides only 8.1 mAh per day for measurments. The nodes are activated every 15 minutes for less milliseconds. The collected information is periodically broadcasted over a single radio hop or a multi-hop corresponding to their distance to the base station.

2.3.3. The "ZebraNet Project"

In general, the aim of animal observation is to collect information about the animal's behaviour. The University of Princeton, USA [JOWM+02] has developed the **"ZebraNet Project"** where zebras wear a tracking collar (Fig.26) that weighs 1090g and consists of the following parts:

- Global Positioning System (GPS) chip – GPS-MS1E
- CPU
- 640KB flash memory – uBlox chip
- Short-range radio (100m, 19.2Kbps) – Linx Technologies SC-PA series
- Long-range radio and Packet Modem for transfers over 8km with 2.4Kbps
- Lithium-Ion batteries with capacity of five days
- Solar cell array

Figure 26: Plains Zebra wearing a ZebraNet collar (left) [SaSa04], corresponding hardware of a tracking collar [JOWM+02]

The main characteristic of this sensor system is the long term animal tracking even over long distances. All the nodes are mobile and build a wireless sensor network. This arrangement allows the scientist to find and follow the zebras throughout their environment in order to collect information. The project is located at the Mpala Research Centre in central Kenya. The biologists observe Plains Zebra (E. burchelli), which live in build tight-knit uni-male, multi-female breeding groups. The movement of so-called "harems" is determined by the females, but the direction is cased by the males. Thus, only the male need to wear the collars to collect enough information.

The designers' goal is to collect as much information as possible for about one year by using low energy, which is achieved by the following conditions:

- every three minutes the position in located by GPS
- every hour detailed activity logs are taken for three minutes

The communication between the nodes is realized by an ad hoc, peer-to-peer routing. The most important challenge is that the observers are also mobile, because they represent the base station. This can cause the "base station" (the observers) not to be available for the motes at all times. When the researches collect the information, they drive through the area

and stop near a water hole because the zebras will go there if they are thirsty due to the limited water resources. Because of this situation, the collected data must be transmitted to other motes to secure the transmission to a base station. This aim is achived by a hierarchical system, which causes the nodes number to be greater than the number of the other nodes corresponding to the adjacency to the base station recently. Every node transmits its data to the node with the highest number in its surrounding; finally the data will arrive at the base station. A disadvantage of this system is the exorbitant data flow over the whole system.

2.3.4. Comparison to our approach

The approaches of the described projects are different to our approach. The different projects can be divided into two groups – static networks and mobile networks.

The "Golden Gate Bridge Project" and the "Great Duck Island Project" are part of the first group that work with stationary motes at a located position. Due to this network structure, the corresponding algorithm is not very complex. The collected data does not need to be stored in the motes, because it can be handled directly on to the other motes and finally to the base station. Thus, the motes do not need a big storage for collecting data. The received information must not be stored temporary until a base station is in the neighbourhood, which is why the information flow inside the network is minimal. Another advantage of those projects is that the capacity of the used batteries can be recharged by solar cells.

In contrast to those projects the "ZebraNet Project" is part of the second network group, thus it works with mobile nodes and with a mobile base station which is not always available. The main difference to the other projects is that in this case the researcher must localise the animals via GPS and that it can work in a wide living space, for example in a wild reserve. Thus, the information flow is enormous and every node needs a big storage to store its data and the received data of other nodes. Important for this project is also the size of the observed animal that can wear bigger motes as mentioned above and the hardware side therefore can be complexer and heavier than in our approach.

Our approach counts to the second group, because it works with mobile motes on the rats. In our case the used hardware needs to be smaller and the power resource is smaller, because the motes are not allowed to be heavy due to the rats' size and cannot be recharged by solar cells as mentioned in the "ZebraNet Project". The only static part of the network is the base station that should be localized at an often frequenced place, e.g. burrow entries; thus it might not be availability at all times. Rats normally live in burrow systems and the surrounding earth prevents the communication with the outside; thus, a GPS sensor makes no sense. The project has to be realized in a well known small area to get obligatory information. A data transfer between the nodes must also be guaranteed to insure a transmission of the collected data of every node to the base station which is why a big storage is needed. The storage must be big enough to store the motes' data and the transmitted data of other rats until the storage is emptied by a meeting with the base station.

All those ideas and requirements are realized in our project's algorithm which is described in the upcoming chapters and analysed afterwards.

3. Algorithm: Design and Implementation

The programming project consists of two software parts (Fig. 27). The communication between the motes via RF and the communication via serial link are realized by the programming language nesC as introduced in chapter 2.1.3.2. The PC component – the right part of the figure – is realized with java and the application and analysing part with the Software MATLAB and the language C.

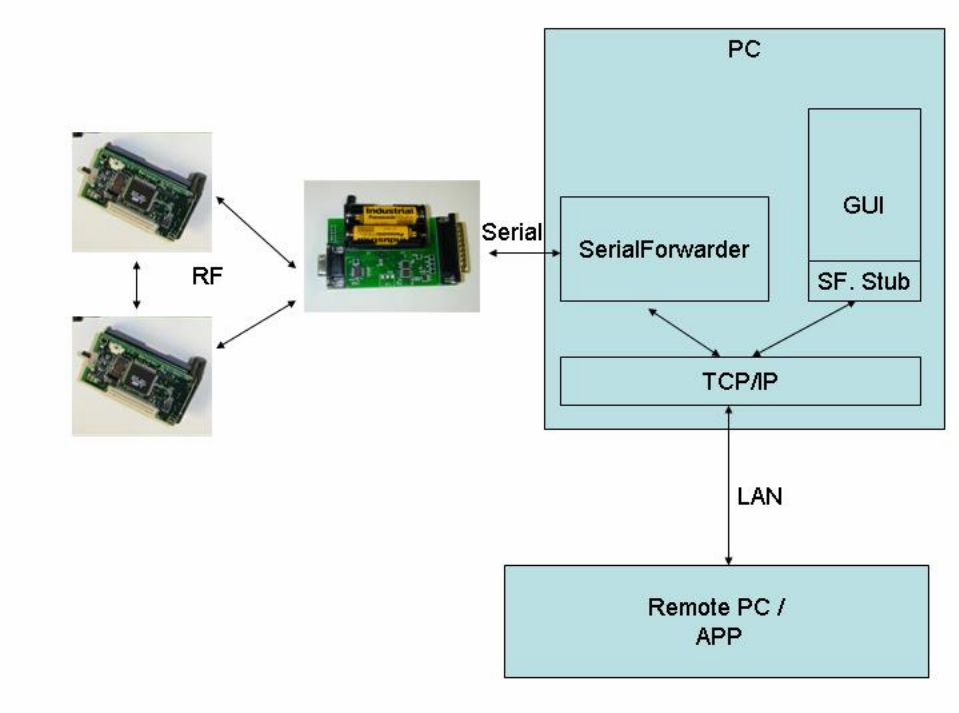


Figure 27: Schema of the communication between all software parts [1]

3.1. The idea of the algorithm

The basic idea of the algorithm is to gather information about the behaviour of two rats in a maze. Three events are distinguished:

1. Rat X meets Rat Y:
    a) Rat X will get the information that it meets Rat Y and the other way round as well as a timestamp when this event occurs.
    b) The rats will exchange some information they had collected.

2. Rat X separates from Rat Y: Rat X will get the information that it separates from Rat Y and the other way round as well as a timestamp when it happened.

3. Rat meets an exit: The collected information must be transmitted to the base station.

Those events are correlated with the storage of information. Current times and received data as well as sensor readings must be stored in special structures which are introduced in detail in chapter. 3.1.1. The mentioned communication between the motes and via the serial link is realized with packets (Fig. 28), which have a special structure:

- **Header** (5 bytes):
    - Destination address (2 bytes)
    - Active Message handler ID (1 byte)
    - Groupe ID (1 byte)
    - Message length (1 byte)

- **Payload** (up to 29 bytes):
    - Individual readings
    - cyclic redundancy check (crc) of 4 bits if the payload has not a specified length

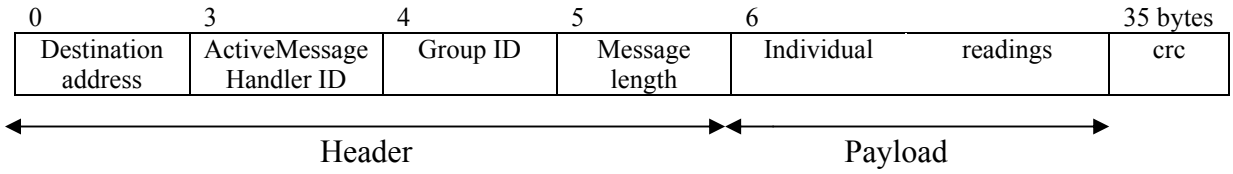


Figure 28: Packet structure

To test the functionality of the algorithm, either the motes are used directly or the simulator AVRORA (cp. chapter 2.3.1.3.) is used.

The following chapters describe the challenges of the software corresponding to the algorithmen. The next chapter deals with the values, followed by the program for the motes and for the base station.

3.1.1. The used values

To handle the different events special values must be set that are realized in the file values.h. The mentioned meetings are stored in a structure called meetingdata. It consists of seven entries:

- WhatKindOfSend
- MyRatName
- MeetingStartTime
- MeetingDuration
- MetRatName
- MetRatMeetingStartTime
- hallo = current position in the storage

The category whatKindOfSend is needed to distinguish between an entry of meeting or of sensor reading. For a meeting entry, it is set to 0. In the second category the name of the current mote is stored to assign the event to the proper mote later on. In the next entry the time of the current mote is stored when the meeting has started, followed by the entry for the met mote and the current time of the met mote when the meeting started. The last category stores the current position in the global storage meetingTable.

6	7	8	12	16	17	21 bytes
whatKind OfSend	myRat Name	meeting StartTime	meeting Duration	metRat Name	metRat Meeting StartTime	current position in the storage

Figure 29: Payload of a packet of a meeting

In addition to the collection of the meetings the algorithm should collect sensor readings from the microphone and from the photo sensor. The interesting information for the scientist is:

- MyRatName
- Time of a sensor reading
- Sensor reading of the microphone
- Sensor reading counter
- Sensor reading of photo sensor

- Current position in the storage

This information must be stored in a packet as well (Fig. 30). The serial link can only handle packets of the same type that is why the sensor readings must be stored in the same structure meetingTable. For this reason the category "whatKindOfSend" is set to 5 to show that this entry is a sensor reading.

6	7	8	12	16	17	21 bytes
whatKind OfSend	myRat Name	time of a sensor reading	sensor reading of microphone	sensor reading counter	sensor reading of photo sensor	current position in the storage

Figure 30: Payload of a packet of a sensor reading

To show other nodes in the surrounding that they are availible the nodes must first send a beacon - "HELLO"-packet - that consist of the mote-ID and the current time of the mote. This information is packed in a packet of the structure meetingdata. The category "whatKindOfSend" is set 1 to show that it is a beacon. The other categories are not interesting and filled with zero (Fig. 31).

6	7	8	12	16	17	21 bytes
whatKind OfSend	myRat Name	meeting StartTime	0	0	0	0

Figure 31: Payload of a beacon - "HELLO"-packet

An additional storage – seeingTable of the structure seerat – must be created to realize that a meeting has happened. This structure is filled with the information of the incoming beacons. The first category handles the rat's name followed by the time of the met node. The next category is a counter that counts the incoming packets of the met rat. The fourth category is set by the current mote. It is the current time when it received the first beacon. The last category is needed to determine the meeting duration. It is filled with the current time value when it receives a beacon of the mote.

The variable values are also stored in the file value.h to make the algorithmen adaptable to the scientists and their requirements:

- MAXRATOBSERVATION – determines the number of rats that wear motes inclusive the mote for the base station – node-ID0
- MAXSTORE – determines how many entries are allowed in the meetingTable
- SENSORFIRING – determines when the mote should collect sensor readings

In the following chapters the algorithm of the motes and the base station are described in detail.

3.1.2. Programming the motes – RatMote.nc and RatMoteM.nc

The motes work with the components Main, RatMoteM, SingleTimer, LedsC, GenericComm, SimpleTime, CC1000RadioC, MicC and Photo. The components **MicC** and **Photo** are needed for the collection of the sensor readings. The compont **CC1000RadioC** is needed to set the radio range. The **SimpleTime** – component is needed for the timestamps. It gives the time in 64 bits that is stored in two 32 bits entrys and the time is given im binary milliseconds: 1 sec = 1024 binary seconds. For our algorithm the low 32 bits are enough. An experiment duration of 48.5 days. The component **GenericComm** is needed for sending and receiving events. The component **LedsC** is just a control instance to check the functionality of different parts of the algorithm. The componet **SingleTimer** is needed for firing of the timer that sends the "Hello"-pakets and sometimes collects the sensor reading. Figure 32 shows the wiring of several components.

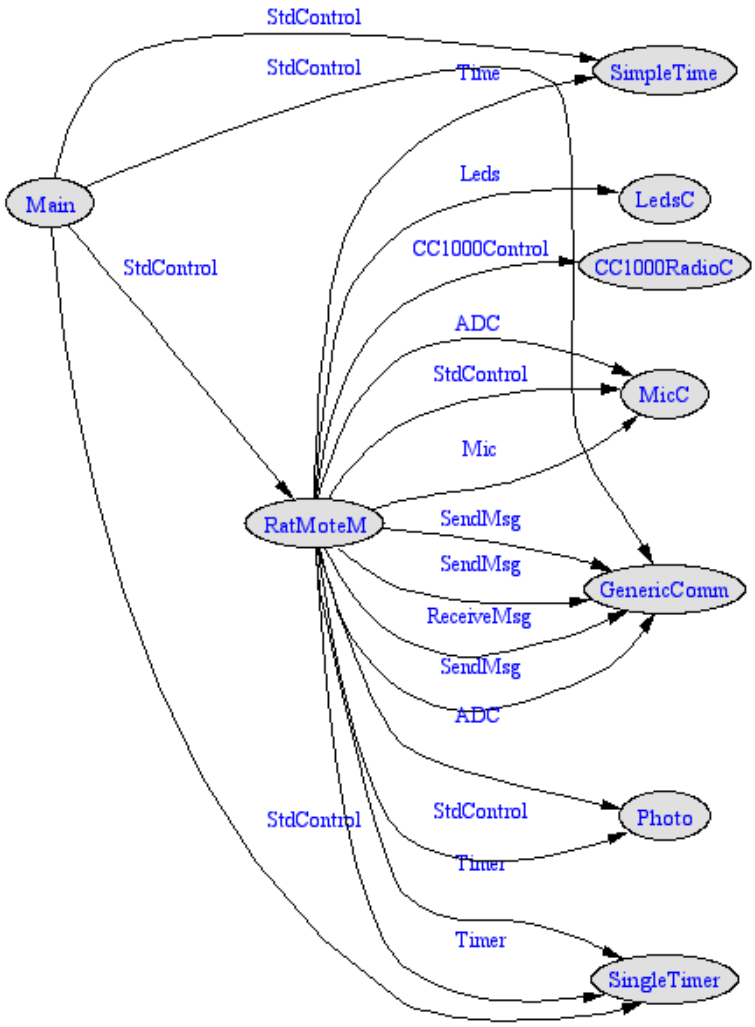


Figure 32: Wiring of the components of RatMote.nc

The module RatMoteM.nc handles the working between the several components and the filling of the meetingTable. It includes the file values.h mentioned above. For the

functionality of the algorithm more variables are needed that cannot be changed by the user. Those variables are declared global in the program.

In the **command StdControl.init()** the different interfaces like Timer, MicControl, PhotoControl, Leds and CC1000Control are started. The first and third category of the seeingTable is set as well as the variable msg-pendingsend. The seeingTable is filled in this programming part, because it is not realized with a rotated storage. It would use too much storage. The scientist can observe 20 motes inclusive the basestaion (node-ID0).

In the **command StdControl.start()** the same interfaces as mentioned above are started. Here the fire rate of the timer is set. The timer fires every 1000ms.

In the **command StdControl.stop()** all the started interfaces are stopped and the LEDs are turned off.

The **event Timer.fired()** causes the firing of the timer every 1000ms. Together with it the beacon is packed with the current time of the mote, the mote-ID and with the mark whatKindOfSend=1. If nothing is waiting to be send which is checked by the variable msg_pendingsend the packet is send out via the broadcast to all motes in the neighbourhood. Simultaneously the counter counterTimerfires is increased. When it is equal to 30 the senor readings are made. In this case the microphone is activated via the command **MicADC. getData()** that requires the **event MicADC.dataReady()** that stores the sensor reading in the variable micData. The photo reading is started with the command call **PhotoADC.getData()** and requieres the **event PhotoADC.dataReady()** that stores the data in the variable photoData. This information is stored together with the current time in an entry of the meetingTable (cp. Fig 28). After this entry the counter is reset.

The event **SendCmdMsg.sendDone()** occurs to clear the variable msg_pendingsend. It shows that the broadcast is free for upcoming sendings.

The main work of the algorithm is done in the **event ReceiveCmdMsg.receive()**. Here the mote handles the incoming packets and reacts in the following ways:

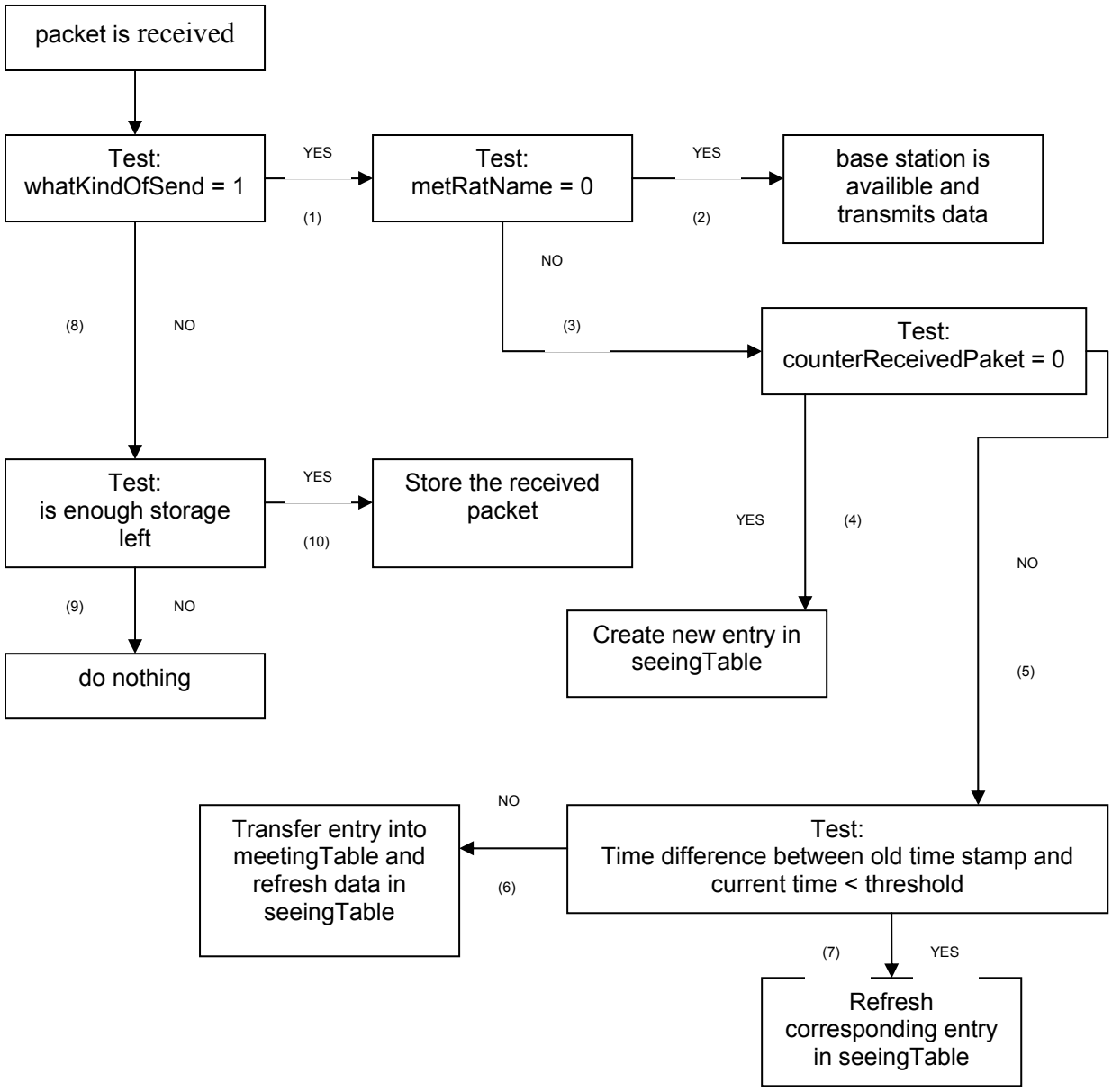


Figure 33: Handling of incoming packets

When the case (Fig.33 (1)) occurs the received packet is a beacon. Before the programm can work with the whole information of the packet, it must be stored in a temporary storage that has the structure of meetingdata (cp. chapter 3.2.). The most important information is the name of the met rat and the timestamp of the packet. First the seeingTable is searched for the corresponding entry of the rat's name and than the counter counterReceivedPacket is checked (Fig.33 (3)). If the rat was not met before (Fig.33 (4)), the remaining categories of the seeingTable are filled with the following information:

- metRatMeetingStartTime = timestamp of the received packet
- counterReceivedPack is increased by one
- myTimeByFirstReceive is set to the current timestamp of the mote
- myTimeAtMoment is set to the current timestamp of the mote

0	1	5	6	10 bytes
metRat Name	metRatMeeting StartTime	Counter Received Paket	myTimeBy FirstReceive	myTimeAt Moment

Figure 34: Entry in seeingTable by first meeting

If the rat was met before (Fig.33 (5)) the difference between the motes' current time and the entry of myTimeAtMoment in seeingTable is checked. If the difference between them is smaller 2000ms, the counterReceivedPaket is increased, and the variable myTimeAtMoment in the seeingTable is set to the current time of the mote. The threshold is set to a specified time because it might occur that the traffic on the broadcast is high so that a beacon might not be received. If the difference between them is bigger than the bound a meeting ends and if enough storage is left, an entry in the meetingTable (c.p. chapter 3.2) is filled with the corresponding information of the seeingTable:

- whatKindOfSend = 0
- myRatName = TOS_LOCAL-ADDRESS ← my mote-ID
- meetingStartTime = entry of myTimeByFirstReceive in seeingTable
- meetingDuration = myTimeAtMoment – myTimeByFirstReceive (both are entries in seeingTable)
- metRatName = entry of metRatName in seeingTable
- metRatMeetingStartTime = entry of metRatMeetingStartTime inseeingTable

After the information has been transfered into the meetingTable the entry in the seeingTable must be renewed. The consequence is that the variable counterReceivedPaket is set to 1, the variable metRatMeetingStartTime is filled with the information of the received beacon and the last two categories are filled with the current timestamp of the mote. The counter nextfreeMeetingTable is also increased. This variable is needed to realize a rotated storage. At the beginning both variables firstfullMeetingTable and nextfreeMeetingTable are set to zero.

In case (Fig.33 (2)) - the received paket comes from the base station: metRatName = 0. In this case, additionally to the first case, the mote checks if it has stored more than one meeting; it then needs to transmit stored meetings to the base station to get free storage. If the mote has stored more than one meeting, it fills a packet with the information of the meetingTable which is stored in the position firstfullMeetingTable (c.p chapter 3.2). After it has been send, the variable firstfullMeetingTable is increased and one storage place becames free.

If the received packet comes from a mote that is not the base station (Fig.33 (4)), the mote checks if the counterReceivedPack of the met rat is equal to three. In this case it tries to transmit the last entry of the meetingTable. It is important to keep in mind that the send paket can be a meeting or an entry of sensor readings.

The first three cases deal with incoming pakets with whatKindOfSend = 1. As mentioned in the last part it might occure that the received packet is not a beacon, which means whatKindOfSend can be five or zero. In this case (4) the received packet should be added to the motes meetingTable. First the mote must check if there is enough storage left. The received information is then stored in a temporary storage and is tranmitted to the meetingTable. The Listen-tool that is used during the transmittion via serial link the last category hallo is set to 255 to realize this event. The variable nextfreeMeetingTable is increased as well.

It is important to keep in mind to check the variables nextfreeMeetingTable and firstfullMeetingTable when they are increased if they are equal to MAXSTORE. In this case the corresponding variable must be set to zero, which has to be done due to the rotated storage.

After this big event the two events **SendToBase.sendDone()** and **SendToNext.sendDone()** are finally called to reset the variable msg_pendingsend which is one every time when the mote wants to send anything via broadcast.

3.1.3. Programming the mote-ID0 – Basestation.nc and Basestation.nc

As mentioned above the mote with the ID zero has a special role in our algorithm. The mote represents the base station and is responsible for the transmission of the meetingdata and the sensor reading to the PC via the serial link. It is basically the same program as for the other motes.

The main differences occur in the file BasestationM.nc. The mote does not collect sensor readings and has no meetingTable. The consequence is that the number of the global variables is lower. The following part describes the differences between BasestationM.nc and RatMoteM.nc.

During the **event Timer.fired()** only the beacon is send. The packet is filled with the introduced information corresponding to chapter 3.2.. The base station does not collect sensor readings which is why the corresponding commands and events are not programmed in this case.

The **event ReceiveCmdMsg.receive()** handles the incoming pakets. The figur 35 shows the needed handling.

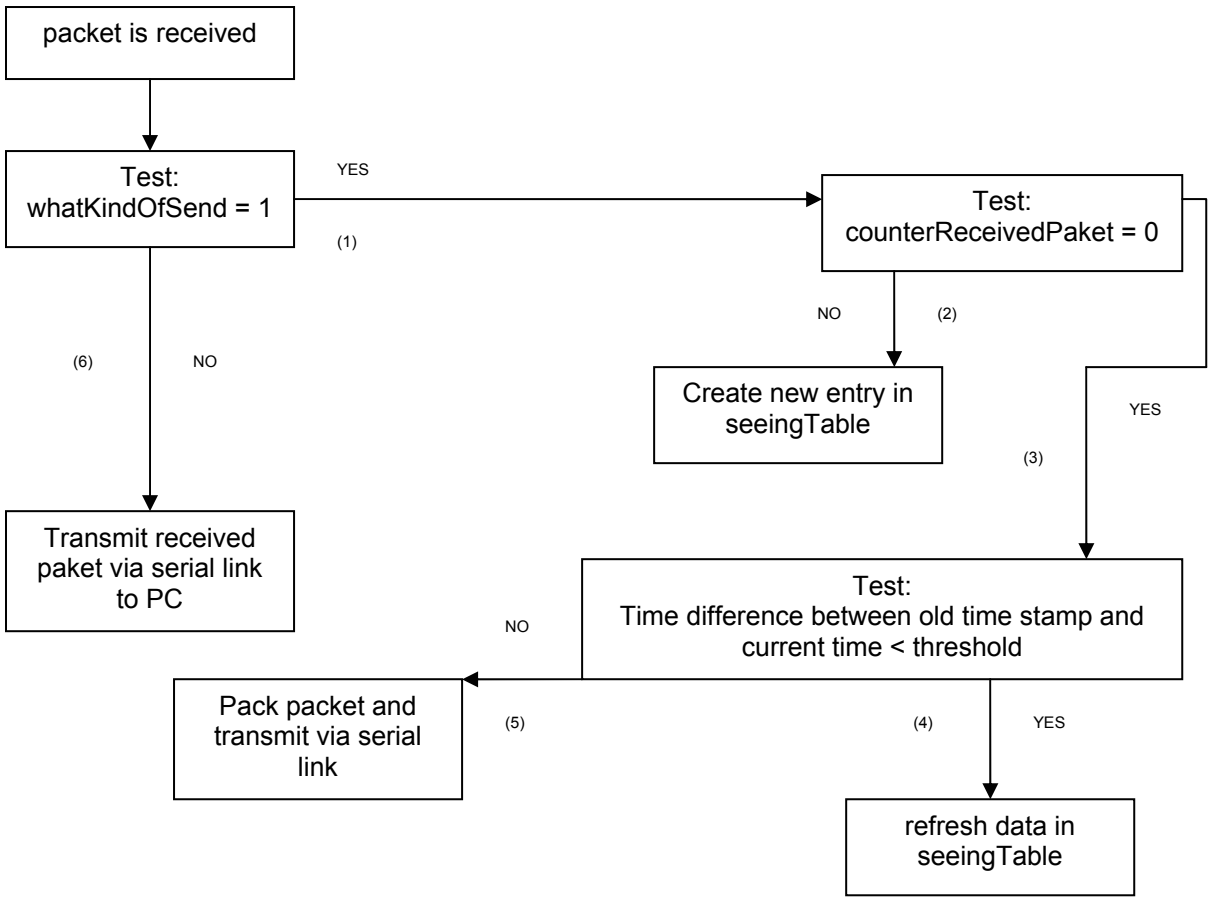


Figure 35: Handling received paket by the base station

In the first case (Fig. 35 (1)) the incoming packet is a beacon. It is handled in the same way as in RatMoteM.nc. The program differences occure in the if-case where the timestamp are compared (Fig. 35 (3)). If the difference is bigger than the threshold (Fif.35 (5)) the mote packs a packet of the type meetingdata directly, instead of building an entry in meetingTable (Fig. 35 (4)). The packet is directly sent via serial link to the PC.

In the second case (Fig.35 (6)), the mote handles the received packets of the other motes that are filled with their meetingdatas or sensor readings. The received packet is first stored in a temporary storage of the type meetingdata; than it is packed in a corresponding packet and transmitted via serial link to the PC.

All following event are the same as in RatMoteM.nc.

3.2. Testing the algorithm with AVRORA

The next step is to test the functionality of the programmed algorithm. It is important to keep in mind that the node-ID of the base station must be zero otherwise the whole algorithm would not work. The first possibility is to work directly with the motes and to make outputs via the activation off the LEDs. This possibility means a lot of work and cannot check the correctness of the send packets, which is why it is better to test the algorithm with the simulator AVRORA as described in the this chapter.

To test the programs the user needs to compile them and set the mote-ID for each mote, because concrete mote-IDs are needed for several if-cases of the program. The next step is the creation of an od-file for a Mica2 mote which is needed for the simulator. The first step is to set the mote-ID. The user must verify the input and the output as well as the ID which is done by the command set-mote-id <input> <output> <ID>. Afterwards the corresponding od-file thats needed for the simulator must be created with the command avr-objdump –zhD <input> > <output>. The same commands can be used to program more nodes with RatMote.nc just replace 1 with another number > 1. The mote-ID 0 is reserved for the base station that can be converted correspondingly.

The simulator can than be started with. The user needs to specify the kind of simulation, the monitors, the duration of the simulation in seconds, the number of simulated nodes and the corresponding od-files. The result can directly be saved in a txt-file that can be find in the folder Avrora.

If the result of the simulator corresponds to the requirements, the development of the algorithm is finished and can be tested directly on the motes. In the following chapters the "programming of the PC-part" and the analysing part with MATLAB will be introduced.

3.3. Programming the PC-part

To transmit the collected information to the PC, the user needs to start the SerialForwarder. The COM-Port of the serial link and the baud-rate of the mote (i.e. 19200 – Mica2 mote) must be specified. This program is used to read the packet data from a serial port and forward it over to the PC. Now other programs like Listen.java and Dump.java are able to work with the information.

The software TinyOS provides different java programs to work with the received information. The needed java programs are:

1. Listen.java
2. Dump.java

Listen.java opens the serial port and dumps the incoming packets to the screen. It calls the program Dump.java which processes the incoming packet and displays it on the desktop like the AVRORA-simulator without the LEDs activity. Figure 36 shows a result example for Listen.java.

```
7E 00 04 7D 0D 00 03 C0 00 00 00 20 C5 01 00 01 00 21 01 00 09
7E 00 04 7D 0D 05 01 40 34 03 00 00 00 00 00 06 05 01 00 00 16
7E 00 04 7D 0D 00 00 40 45 03 00 A0 36 00 00 01 80 FD 02 00 0F
7E 00 04 7D 0D 00 03 C0 CD 01 00 40 46 00 00 01 E0 ED 02 00 0B
```

Figure 36: Example for the original version of Listen.java

The first five entries build the header followed by the information of the received packet of the structure meetingdata that consists of:

1. whatKindOfSend
2. myRatName
3. meetingStartTime
4. meetingDuration
5. metRatName
6. metRatMeetingStartTime

For further analysis it is important to keep in mind that the information is given in hexadecimal and is needed to be converted in decimal.

3.3.1. Modification on Listen.java

In the program Listen.java the user needs to make changes in the for-loop of the case try because the header information is not interesting for the analysis. In the case try the program tries to read the incoming packets. In the original version the incoming packet is directly handled on to the program Dump.java which is responsible for decoding and printout of the packet information. In our case only the payload of the packet without the header is interesting. The idea is to cut the header off and just save the information of our individual readings of the structure meetingdata, which is realized in the second for-loop in the case try. Here the new version of Listen.java cuts off the first 5 information and the rest of the incoming packet is saved in a second byte structure called pack2. In the next step the new pack2 should be handled on to the program Dump.java. Here it is better to use the original version. It prints the pakets in hexadecimal format and dumps them to the screen. In our case the result is stored in the txt-file result-Listen.txt.

3.3.2. Processing data with DataProcessing1.java and DataProcessing2.java

The program DataProcessing1.java works with the result of Listen.java. It imports the information of the file result-Listen.txt. It translates the hexadecimal information to decimal which is needed by MATLAB. The mote sends the data in little-endian format, so that the least-significant-byte stands in front of the most-significant-byte. The first step should be the changing of the position of special parts of the incoming information – meetingStartTime, meetingDuration, metRatMeetingStartTime. The other components are easy to convert. This conversion and storing in the newly created text file ListenData.txt is done by DataProcessing1.java. The text fill now consists of meetingdata and sensor readings.

To get a better overall view, the programm DataProcessing2.java should run. Its job is to separate the information and store the data in the corresponding files (Fig.37). The data is

divided into meetingdata, sensor reading of the microphone and sensor reading of the photo sensor.

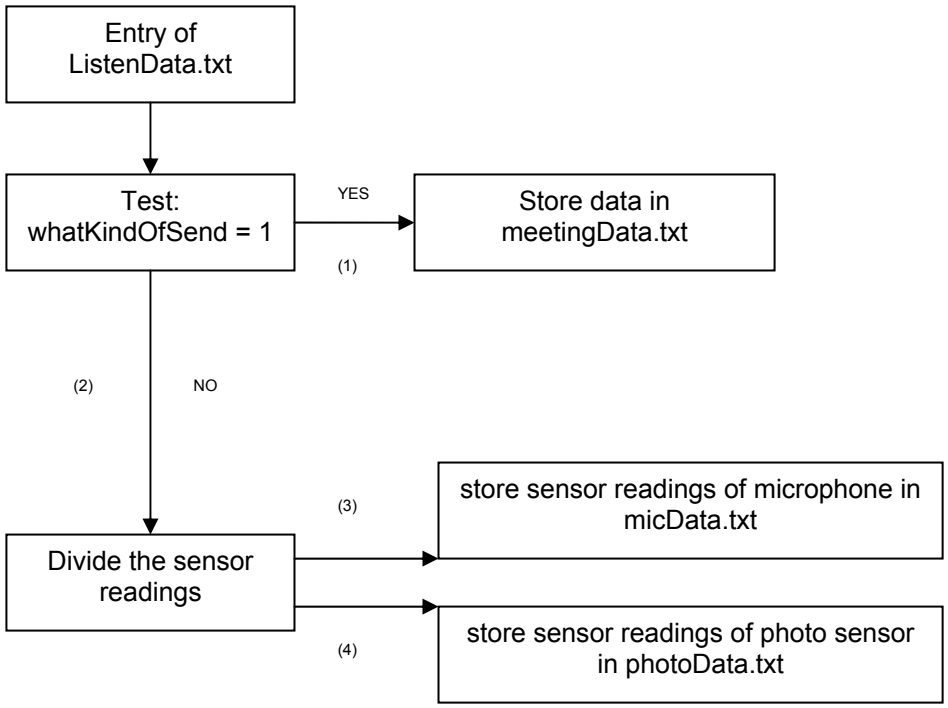


Figure 37: Dividing information of ListenData.txt in three files

The first type (Fig. 37(1)) is easy to determine, it is characterized by the first entry in the row that should be zero. In this case the whole information of the row is stored in the file meetingData.txt. If the first entry is five, the row consist of sensor readings (Fig. 37 (2)) and must be divided. The first three entries are the same in both kinds of sensor readings. The fourth entry is the entry of the microphe followed by the entry of the reading counter. Those five entries are saved in this order to the file micData.txt (Fig.37 (3)). The first three entries are also stored for the other senor reading followed by the sixth entry – the photo reading (Fig.37 (4)). The last entry in the file photoData.txt (case 3) is the original fifth entry – the reading counter.

Those txt-files are used by MATLAB which is responsible for the application of the collected information. It is important is to keep in mind that the entry which stands for times are in binary milliseconds: 1sec = 1024 binary milliseconds.

3.3.4. Application with MATLAB

The idea of the MATLAB function txt_einlesen_fct.m is to include the file meetingData.txt into the workspace of MATLAB. Then the scientist can choose the name of the rat whose behaviour should be observed. In the for-loop the corresponding data is selected. Just the name of the observed rat, the time when a meeting has started and the duration of the meeting are interesting. The time is converted from binary milliseconds to seconds. The corresponding values are stored in the workspace of MATLAB.

For a two dimensional plot more information is needed. To make a correct time analysis the observed times must be adapted corresponding to the different starting points (Fig. 38).

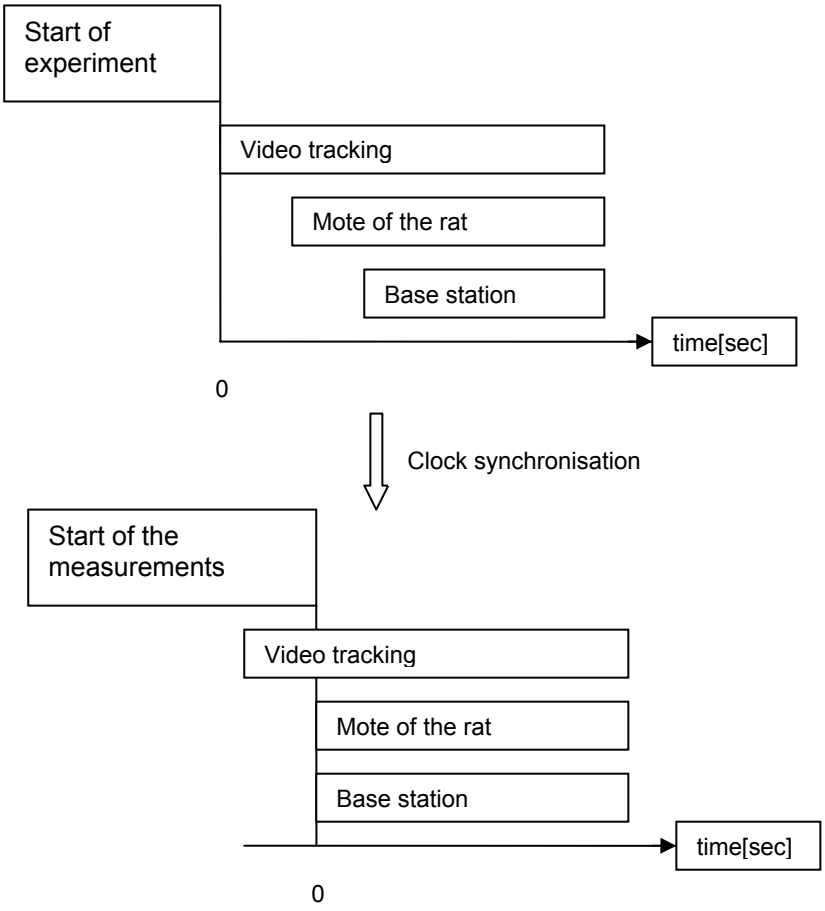


Figure 38: clock synchronisation

Later in the plot all graphs should be started at the same time. Thus, the clocks of the observed rat and the base station must be synchronizied. The called function txt_einlesen.txt calculates the avarage time difference between and adds it to the time of the base station. The plotted application shows the time in seconds on the x-axis and if a meeting was recognized on the y-axis.

After this application is plotted the information of the tracking programm must be added. In this case the information of the file *projectname.txt* is stored in the workspace by proceeding the function videodata.m. This programm divides the information into three categories: time in seconds, x- and y-coordinates. The time value is also converted to seconds and the time difference between the starting of the tracking program and the mote of the rat are substracted. Afterwards the programm trackermodification.m is called to divide the data of the tracker into two groups: inside and outside. The user must specify the radio range of the base station before. Correspondingly to the determination if the current measurement is inside or outside the radius in the array inside[], an entry of one or zero is made. In the procedure the plotting command is included. As a result an application with the information of the base station, the mote of the rat, and the tracker is plotted.

For an analysis of the sensor readings, an equivalent procedure to videodata.m, was programmed – sensordata.m. This procedure stores the sensor readings in the current workspace if it is activated with the following command:

```
[type, node, time, dta, position]=sensordata('micData.txt');
```

The variables in the brackets show the structure of the data in the file of the sensor readings. On the right side the path of the source file is specified. In this case the microphone data should be stored. To plot the information, the command plot(time,data,'g*') is needed which verifies that the x-axis stands for the time in seconds and that the y-axis gives the current data of the microphone's measurement.

The next part is to test the algorithm in the laboratory with rats. This is described in the upcoming chapter 4.

4. Application of the algorithm

In this chapter the algorithm is tested with rats in a laboratory. The next chapter describes the environment of the laboratory. The other chapters deal with the pre-work and the real test with the rats.

4.1. Laboratory environment

The laboratory is a 2x2 m arena placed on the ground which has two tunnel sections. The tunnel sections are needed to show the functionality of the photo sensor. The dimension corresponds to a rat's burrow. The arena is surrounded with 30cm high wooded walls that prevent the rat from climbing out.

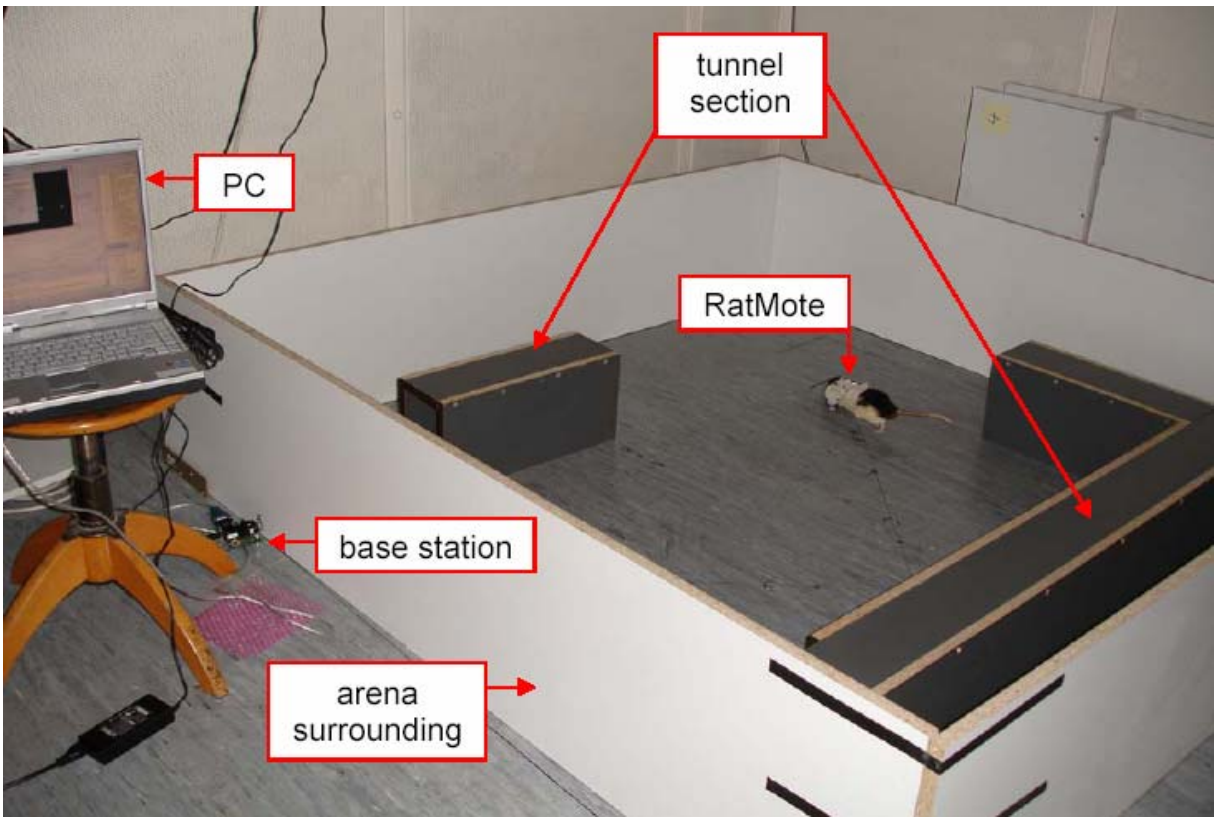


Figure 39: Laboratory environment

At the ceiling a lamp and a video camera are installed. The lamp can be dimmed. The camera is needed for other experiments in order to make video tracking possible. In our case the camera is used to make a video of the experiments and for the analysis of the algorithm

with one mobile mote by video tracking. The biggest disadvantage of this program is that it can only track one reflecting point in a session. In case our algorithm should be tested with more than one rat, the tracking does not work. Before the experiment can start, it is essential to measure the static position of the base station. Afterwards the results are checked into the tracking program. The rat must also wear a reflecting point during the experiments for tracking purposes. The tracking program recognizes the position of the reflecting point as often as possible and stores the corresponding information to the file *projectname.txt*.

4.2. Setting the radio frequency power

Before starting to collect data the user needs to make several adjustments. The user needs to test the radio frequency power (RFPower) of the nodes that act as transmitter and receiver. First, the user work with one transmitter and one receiver corresponding to the biological experiment. The best way to test the radio frequence is to test it with one Mica2dot mote and one Mica2 mote, because it has three LEDs that can be used for a visual output. One node is programmed with Basistation.nc and the other node with RatMote.nc. The scientist activates the LEDs in the programming code at the following positions to obtain a visual output:

- send beacon → call Leds.redToggle();
- packet is received:
  - when an entry in seeingTable is made → call Leds.yellowOn();
  - when a paket is send to the base station → call Leds.greenOn();

  → Both Leds are turned off again at the beginning of the event

The idea is shown in the following figure 40:

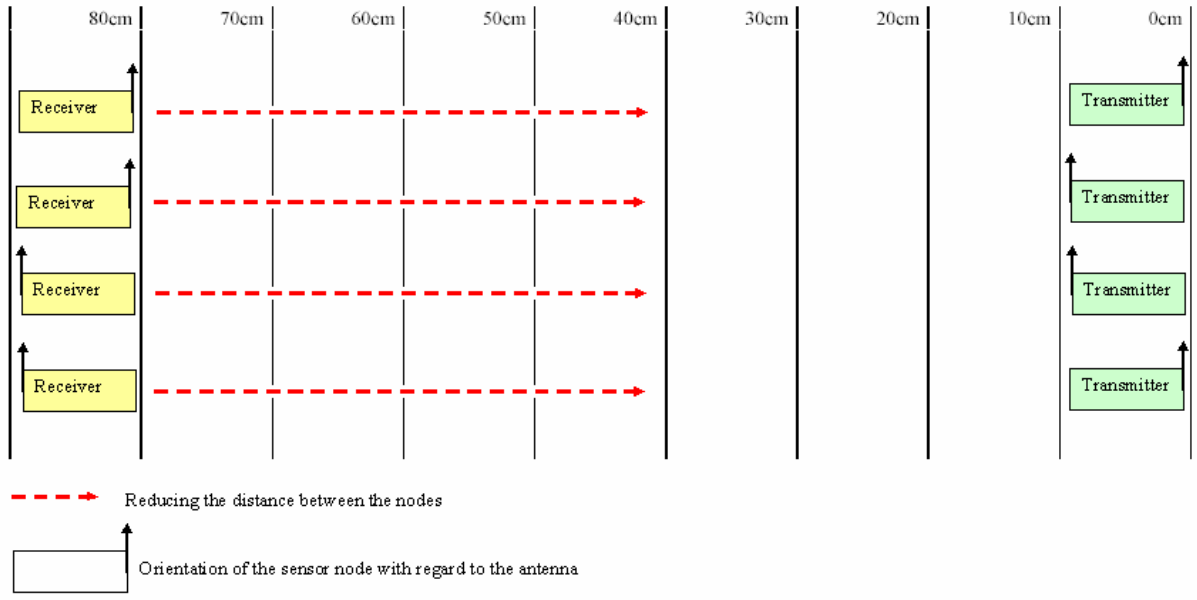
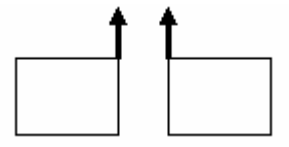
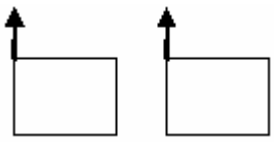


Figure 40: Experimental structure for optimizing the distance between two nodes with regard to working RF communication

To collect information within a special radius of a rat and under consideration of the characterisation of the wooden walls of the maze, you need to use the interface CC1000Control. In the **event StdControl.init()** the RFPower can be set by the call CC1000Control.SetRFPower(0x00). The user needs to change the entry in the brackets corresponding to appendix A – RFPower Settings – to be able to vary the RFPower. The following two cases have to be tested in order to analyze the possible RFPower settings in the laboratory:



case 1: "heads up"



case 2: "no heads up" = train

The test is made with one Mica2 and one Mica2dot mote. Important to keep in mind is that the radio range dependent on the electric interference in the laboratory. The results are shown in table 7.

RFPower settings	Working radio distance "heads up"	Working radio distance "no heads up"
0x00	300cm	300cm
0x02	> 300cm	> 300cm
0x03	> 400cm	> 400cm
0x04	> 500cm	> 500cm
0x05	> 700cm	> 700cm

Table 7: RFPower Results for Mica2dot and Mica2 mote

For the experiments with the rats the Mica2dot will be used; the Mica2 mote is used as a base station. When comparing the working radio distance to the dimension of arena, the user can realize that the best calibration is 0x00. This is the minimum distance for the case "no heads up". The radio range can be decreased furthermore by twisting the antenna of the motes.

4.3. Calibration for the tracking program

For later analysis it is essential to make measurements for calibration means – the coordinates of the base station and the radio range. In our case the position of the base station is in the left bottom side of the arena and it's funk range is around 160cm or 300cm. With help of this data the collected information of the tracking algorithm can later on be divided into mesuarments in- and outside the radio range. As a result the tracking programm gives one data file back. The file consists of the measurements, time in milliseconds, as well as the coordinates of the reflecting point in the x- and y-axis.

4.4. Experiments and analysis

In our experiments a Mica2 mote of type MPR400CB with 900MHz and Mica2dots of type MPR500CA with 900MHz are used. The Mica2 mote is programmed with BasestationM.nc and has the node-ID0. The other motes weared by rats are Mica2dot motes and have the sensorboard MTS510CA. This sensorboard consists of three sensors – an accelerometer, photo sensor and a microphone. In our algorithm only the photo sensor and the microphone are activ.

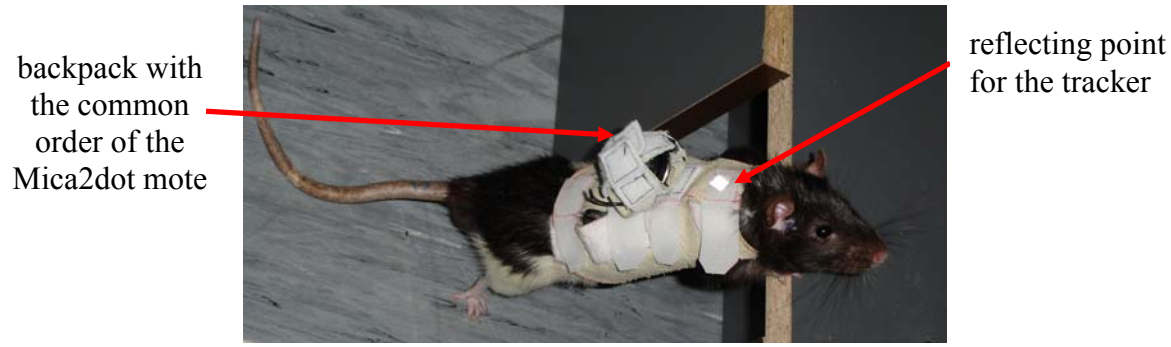


Figure 41: Construction of the west and the backpack with a complete Mica2dot mote

The test phase of the algorithm is divided into two parts: First, the functionality of the algorithmen is tested by working with one rat. In this case the tracking is possible and is used for the analysis (Fig. 41). Second, the functionality of the algorithm is tested with two mobile motes, which is done by scientists that walk around in a 1,5m² room. A tracking analysis is also impossible. The analysis can be done only with a comparision of the data received by the SerialForwarder.

4.4.1. First experiment type: static base station and one mobile mote

As mentioned above this experiment is done with a rat. The base station is a Mica2 mote linked to the programming board connected to the PC, and is located at the coordinates x = 286 and y = 72. The rat wears a west with the Mica2dot mote. It runs around in the arena motivated by chocolate flakes. The collected data of the SerialForwarder and the file of the tracking program are handled on to MATLAB for analysing purposes. Important for later analysis is to start the whole equipment in the following sequence:

1. starting the SerialForwarder and the Listen-tool on the PC
2. starting the base station
3. starting the tracking program
4. starting the Mica2dot mote
5. setting the rat into the arena

The following two chapters describe the result of the experiment corresponding to different configurations – firing rate of the timer, length of the antenna and threshold for filling an entry in the storage meetingTable. All experiments last 40 minutes but the plots only show 20 minutes.

4.4.1.1. Firing rate = 1000ms, threshold = 2000ms

In this experiment the timer fires every 1000ms and if the mote receives a beacon that's timestamp is more than 2000ms older than the received before the old meeting is finished, an entry in the meetingTable is made and the information in seeingTable is refreshed. The antenna's length is 1 cm in the first approach that means the radio range is 160cm. In the second approach the length is maximal with 7cm and the corresponding radio range is 300cm.

As mentioned above it is important to know the time difference between the tracker starting and the mobile node starting which is in the first approach with the short antenna 30.87sec. The upcoming figure 42 shows the plot for the recognized meetings of the base station, of the mobile node and the corresponding tracker information. The tracker information shows the distance between the base station and the mobile node corresponding to the decision if the mobile node was in the radio range of 160cm around the base station. If it is the case the plot increases otherwise it decreases.

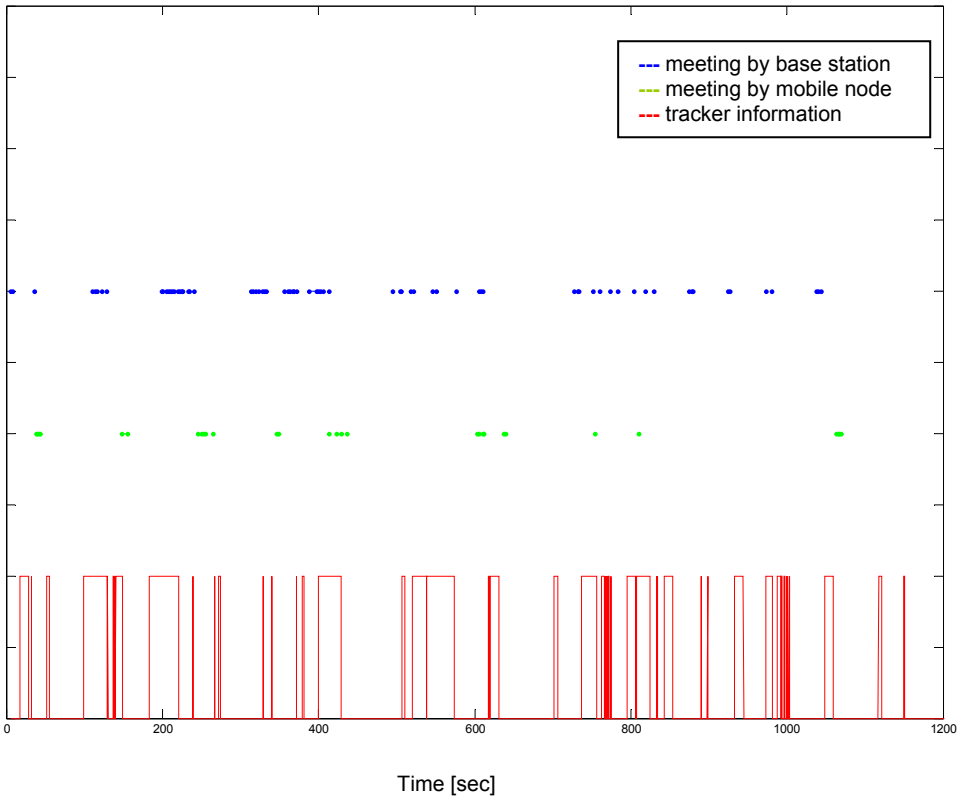


Figure 42: Experiment 1: "meeting plot" – recognized meetings by base station, mobile node and the corresponding tracking information

Corresponding to the figure 42 above it can be recognized that the information by the base station and by the mobile node differs. Thus, a high packet loss is indeed. The tracker information confirms this speculation. Every time if the mobile node is in the radio range of

the line of the tracker raises, this indeed more meetings than recognized by the motes. This speculation can be confirmed if the following plot (Fif. 43) is analysed. It shows the correlation of the distance between the base station and the mobile node corresponding to the relative frequency if it appears.

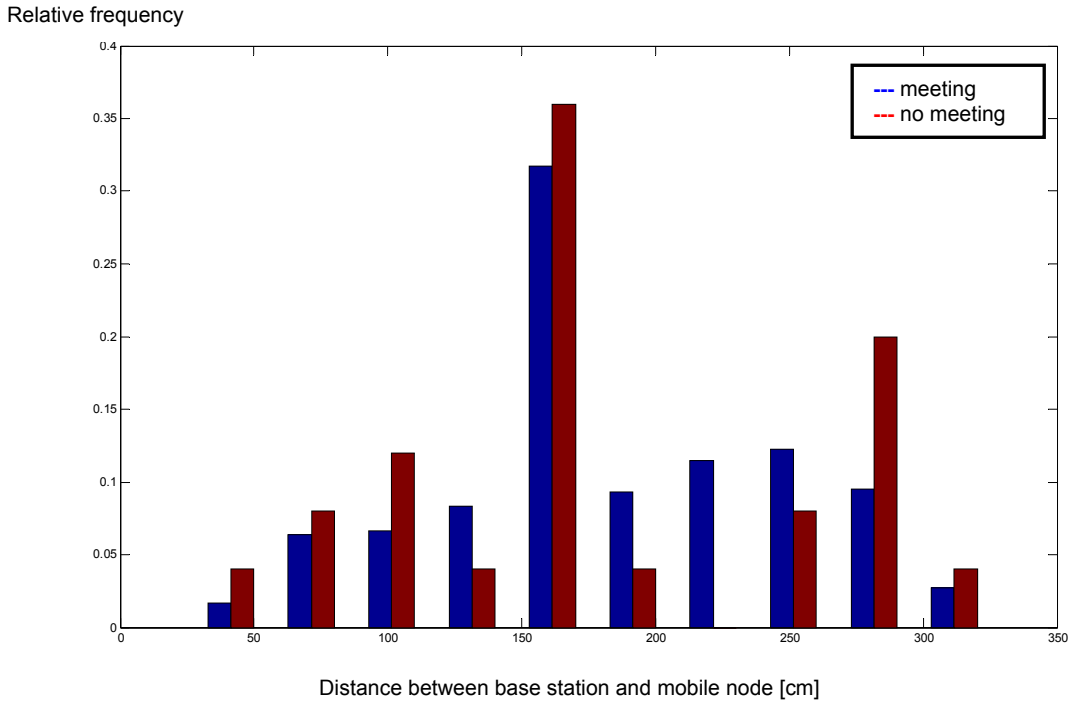


Figure 43: Experiment 1: "correlation plot" – Correlation between meetings and no meetings corresponding to the distance between the base station and the mobile node

Mainly the mobile node if it a meeting is recognized is in the radio range of the base station. The meeting appearance in a distance of more than 170cm away from the base station may occur because of the electrostatic smog which is caused by the turned on devices like PC or tracker. The rate of packet loss in this case is 86% which is very high. It can impressive shown by the plot of the microphone readings (Fig. 44) where only three measurements were received by the base station.

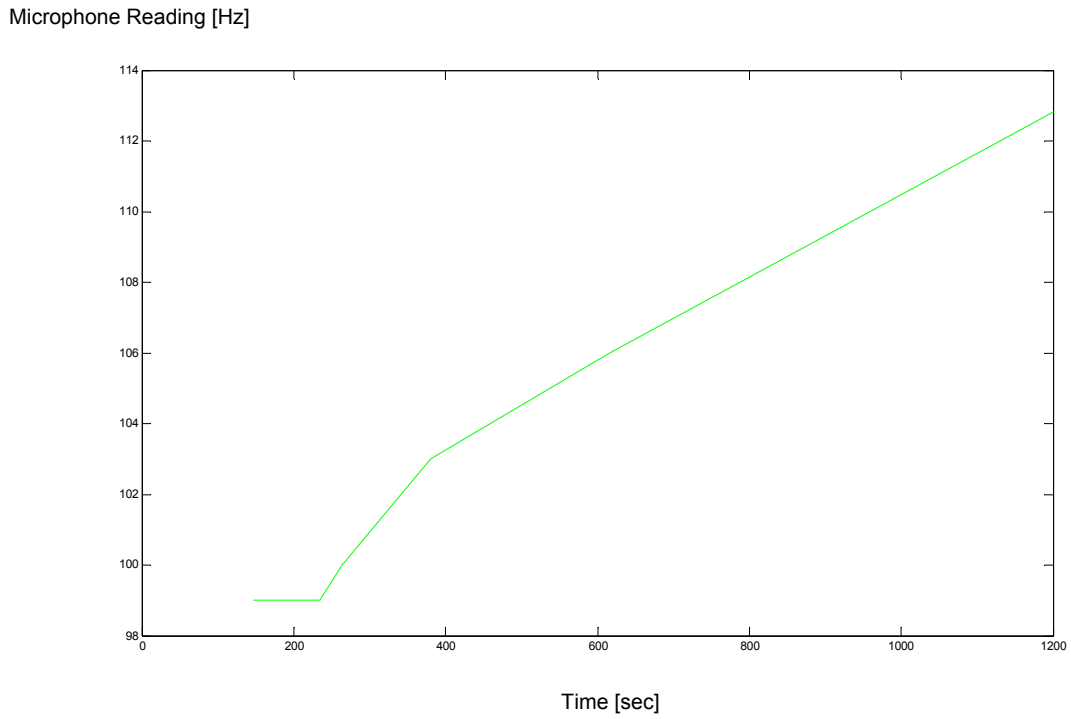


Figure 44: Experiment 1: "microphone reading plot"

As mentioned at the beginning of this chapter the experiment is done also with an antenna's length of 7cm. The speculation is that a longer antenna reduces the packet loss and optimizes the data collection and the analysing. The upcoming figure (Fig. 45) shows the corresponding plot. It confirms the speculation. The base station hears the beacons of the mobile node most of the time. The mobile node shows gaps in the plot that indeed lost packets by transmission. That is the reason for the gaps in the beginning of the measurement and in between. The tracker information corresponds to the other information. If the packet loss rate is calculated it lays by 42%. It is still high but better than before. This fact can also be seen in the correlation plot (Fig. 46) and in the plot (Fig. 47) for the microphone readings.

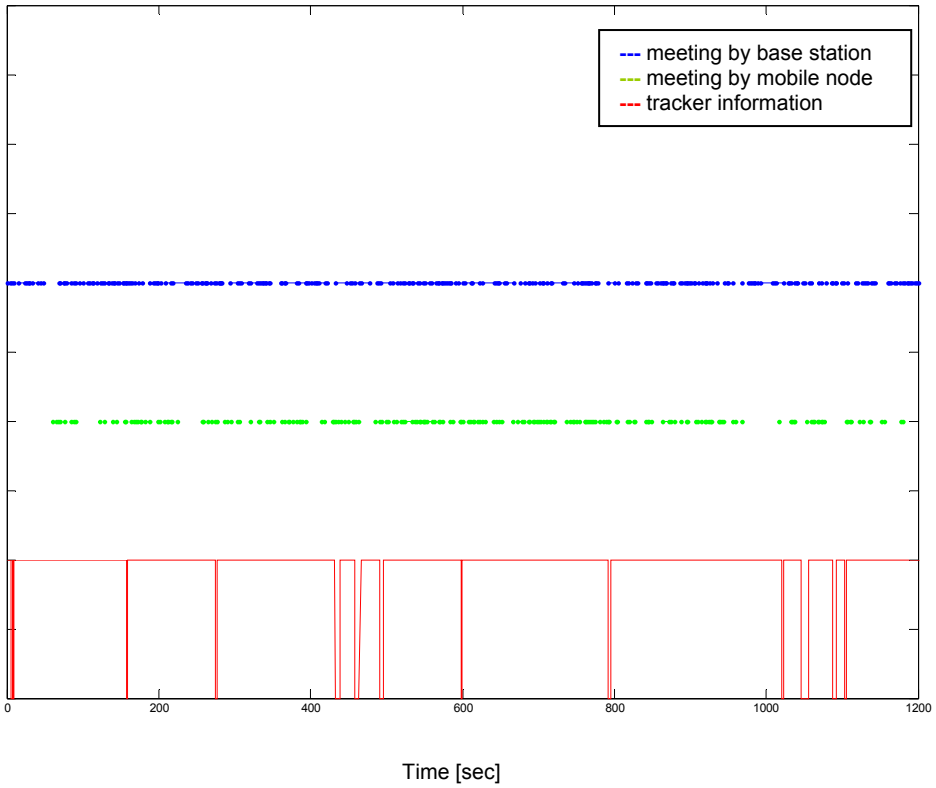


Figure 45: Experiment 2: "meeting plot" – recognized meetings by base station, mobile node and the corresponding tracking information

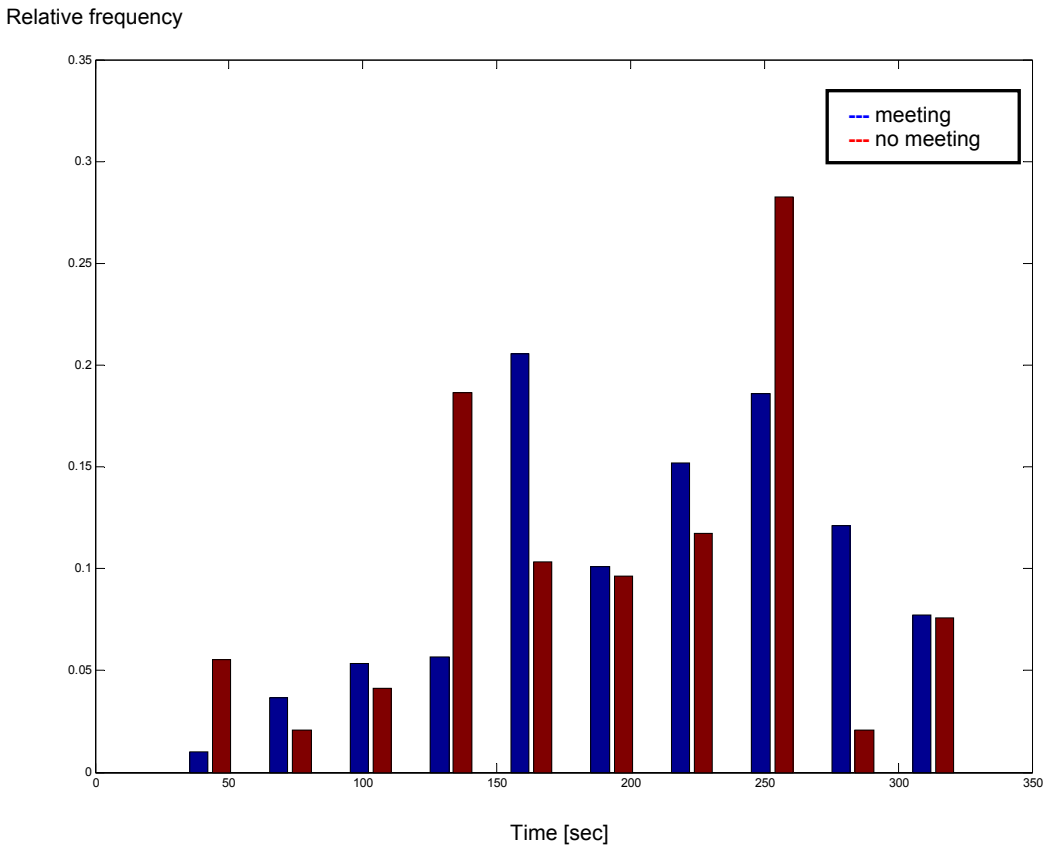


Figure 46: Experiment 2: "correlation plot" – Correlation between meetings and no meetings corresponding to the distance between the base station and the mobile node

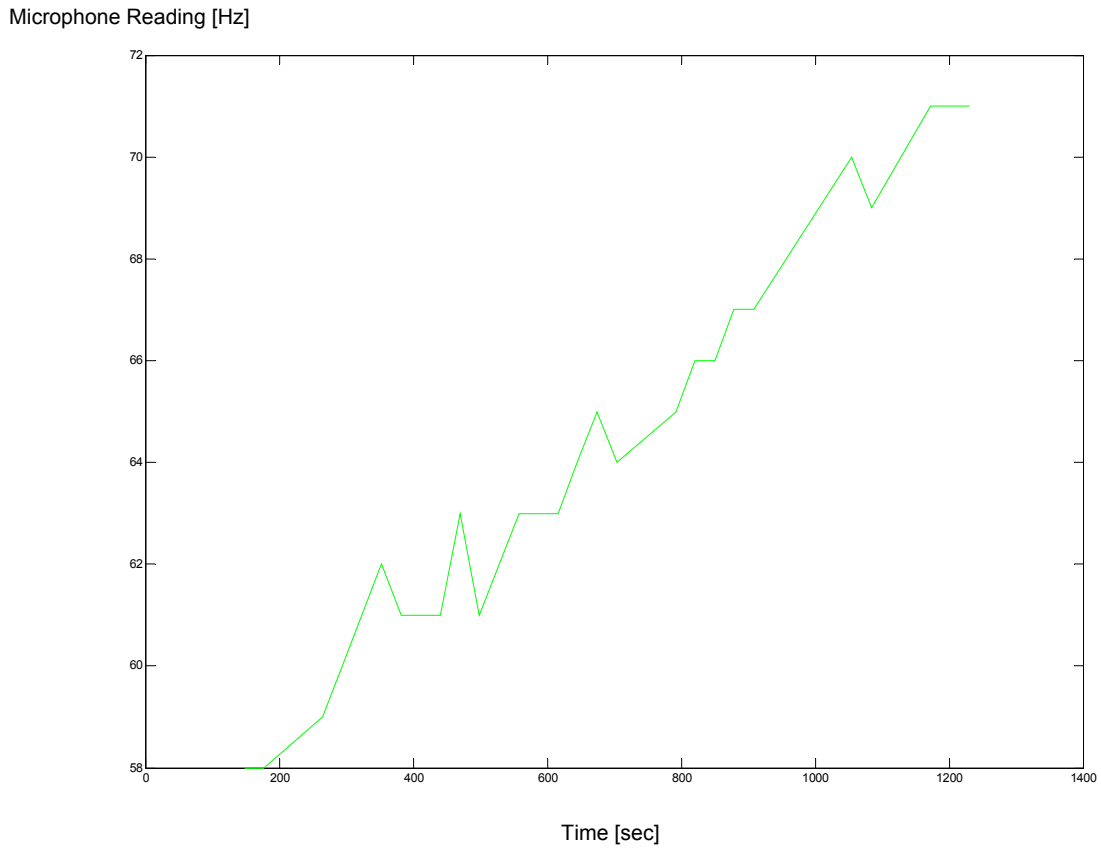


Figure 47: Experiment 2: "microphone reading plot"

The next step is to test if the performance becomes better if the other values are changed. The corresponding result is shown in the upcoming chapter.

4.4.1.2. Firing rate = 500ms, threshold = 2000ms

In this experiments the firing rate is decreased to 500ms, the threshold is the same and the antenna's length is 1cm or 7cm. First, the case for the antenna's length 1 cm is analysed. Figure 48 shows the plot that shows the meeting registration of the base station, the mobile node and the corresponding information of the tracker system.

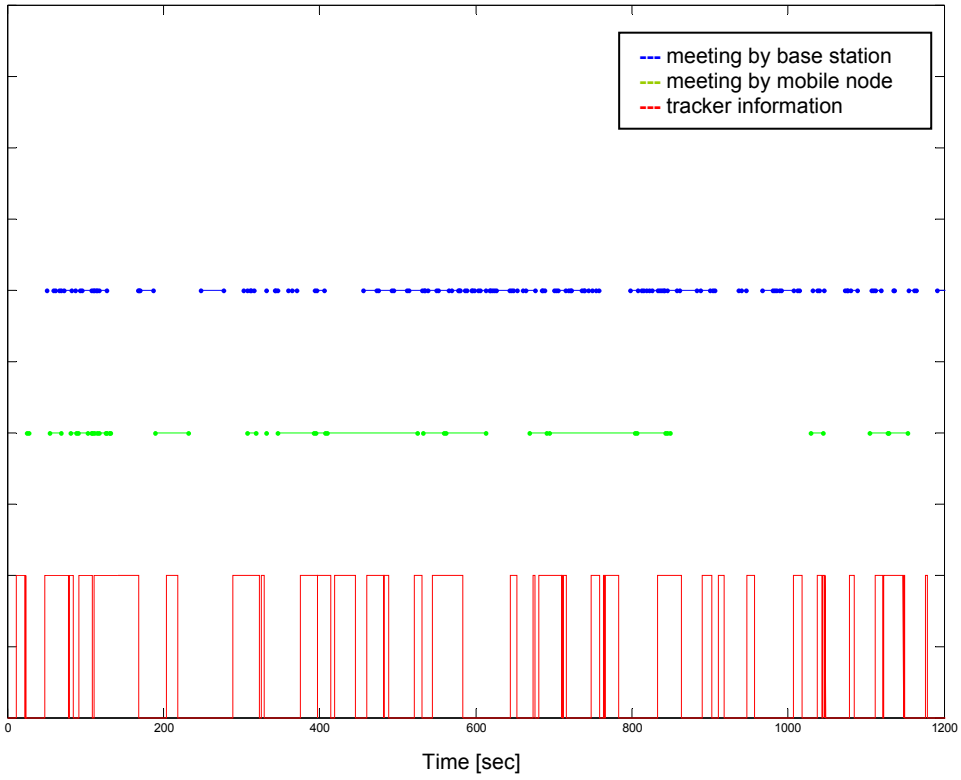


Figure 48: Experiment 3: "meeting plot" – recognized meetings by base station, mobile node and the corresponding tracking information

The plot shows also gaps in the case of the base station and the mobile node but it seems to be better than the figure 42 Also the tracker information seems to correspond. The plot might indeed that the synchronisation of the clocks was not successful but it was done. The tracker was started 60.597sec before the mobile node. The gaps in the beginning can than only base on packet loss. If the file ListenData.txt is analysed it is confirmed. The packet loss rate in this case is 48%. This fact can also be shown by the correlation plot (Fig. 49). This plot indeed that the communication between the base station and the mobile node was also successful if the distance was bigger than 160cm. This might be possible because of the electrostatic smog around or because of errors by the tracking program.

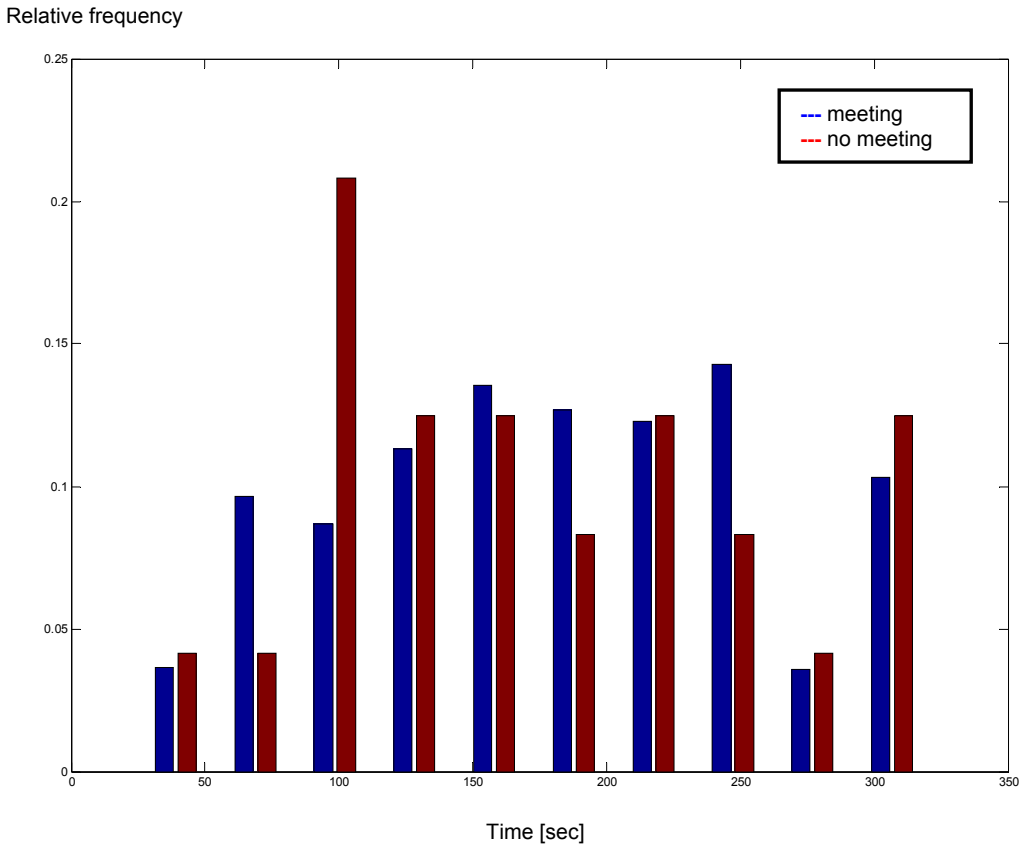


Figure 49: Experiment 3: "correlation plot" – Correlation between meetings and no meetings corresponding to the distance between the base station and the mobile node

In this experiment even the transmission of the sensor readings was very successful that is shown in the figure 50. The packet lost may mainly concern the meeting information. The microphone readings in this case show that the noise in the surrounding differs. Either the rat ate something, sniffles or gnawed at the tunnel section.

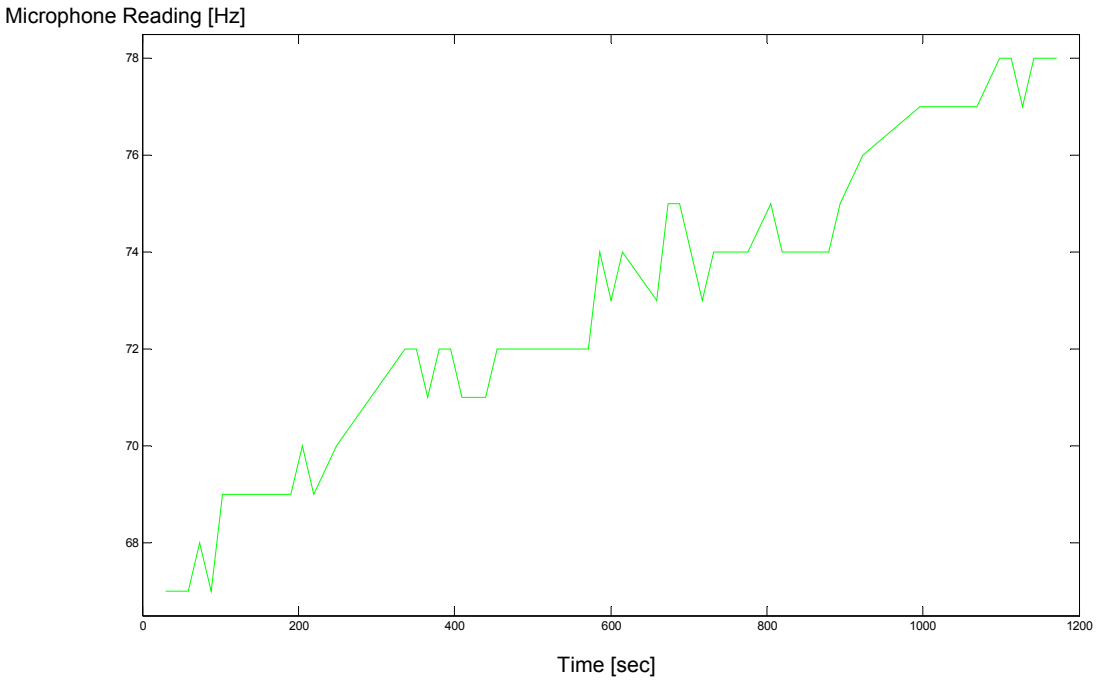


Figure 50: Experiment 3: "microphone reading plot"

In the next case the values are the same but the antenna's length is 7cm. As the results (Fig. 51) show the experiment was not as successful as the others before. The tracker information indeed the mobile node was mostly always in the radio range of the base station. But the plots of the base station and the mobile nodes do not correspond; only in the time space 400-700 sec it does.

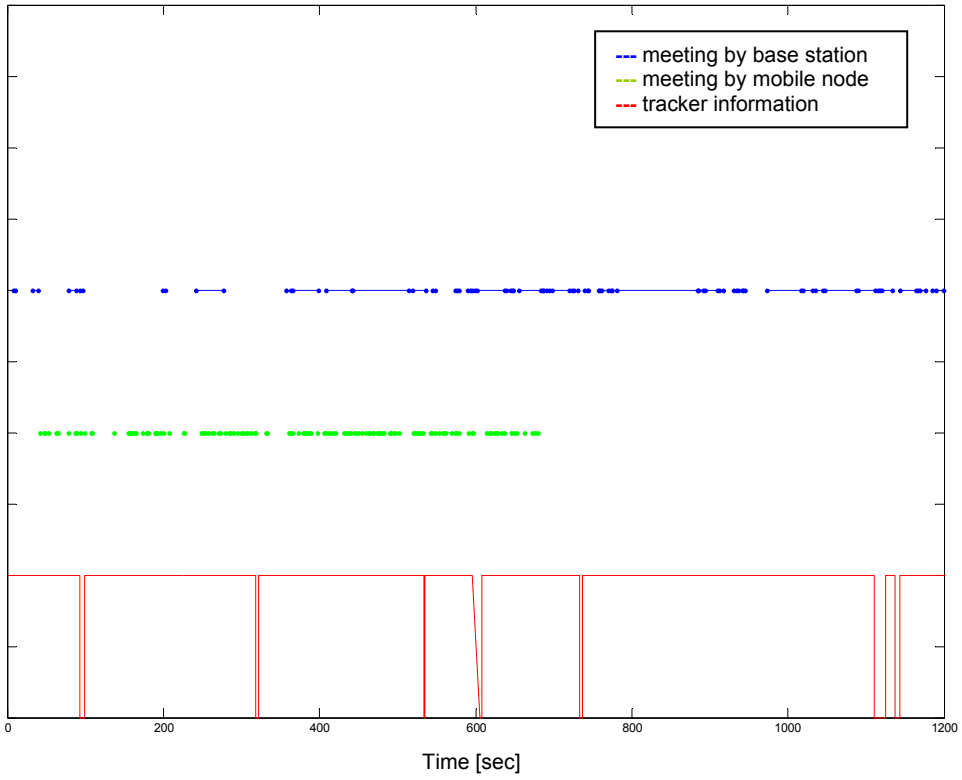


Figure 51: Experiment 4: "meeting plot" – recognized meetings by base station, mobile node and the corresponding tracking information

The packet loss rate in this case is 30,67% for the first 700sec. Afterwards it is nearly 100% because the base station still recognizes the beacons of the mobile node. But the mobile node seems not to transmit the saved meetings. A possible reason might be the less power resources as mentioned in chapter 2.1.2.4. The correlation plot (Fig. 52) confirms that around half of the meetings were not recognized by the mobile node but by the base station. The plot for the sensor reading is not analysed in this case because it is not a strong statement because of the packet loss rate.

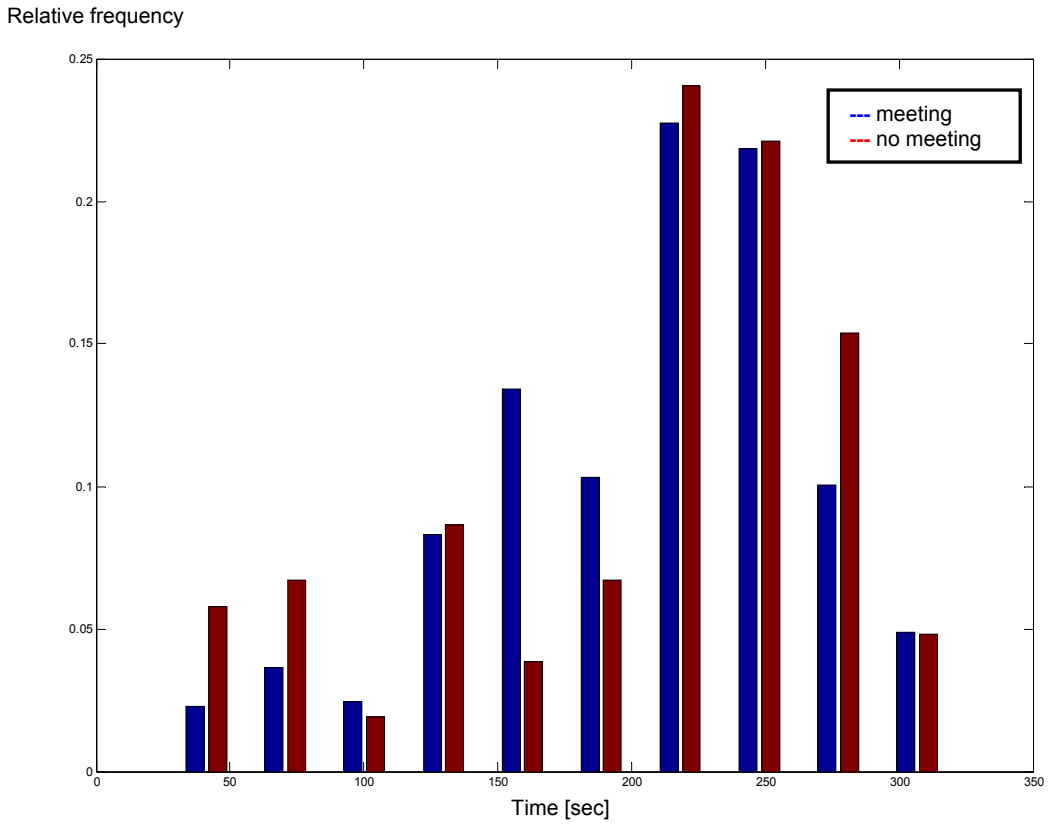


Figure 52: Experiment 4: "correlation plot" – Correlation between meetings and no meetings corresponding to the distance between the base station and the mobile node

4.4.2. Second experiment type: static base station and two mobile motes

The second experiment is done with help of scientists that walk around for 25 minutes. The antennas are full extended. Additionally, with this setting it is possible to show a data transmittion of full packets between the mobile nodes. The firing-rate of the timer is 1000ms, every 30 seconds the sensor readings are activated and the threshold is 2000ms. In this experiment the packet transmission between the motes and the functionality of the sensors should be tested. The meeting behaviour is shown in the figure 54 and shows a less packet loss rate between 11-23% corresponding to the observed mote.

Every time when a paket transmission between the mobile motes occure the entry in the last column is FF that means 0xFF = 255 in decimal. This received packet is stored at the current next free position of the storage meetingTable of the received node. With this redundancy in the network the completeness of the data increases and a better analysis as well as application of the meeting behaviour is possible.

```
00 02 E0 29 02 00 00 00 00 00 00 0F 04 00 39
05 03 E0 49 02 00 8F 00 00 00 04 DB 03 00 00 FF
05 02 E0 49 02 00 91 00 00 00 04 CA 03 00 00 3B
00 00 80 17 06 00 00 27 00 00 02 20 32 04 00 0B
```

Figure 53: part of the result of the SerialForwarder for the second experiment

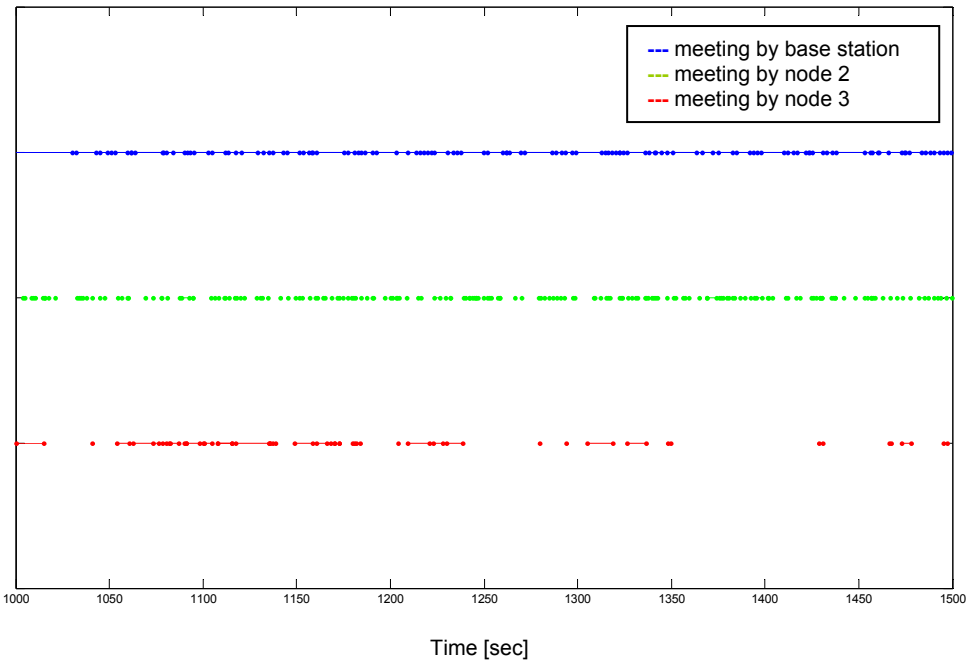


Figure 54: Experiment 5: "meeting plot" – recognized meetings by the base station and the two mobile nodes in the time space 1000-1500sec

The sensor readings were tested in this experiment also and the result of mobile mote 3 is analysed. In the next two figures the sensor reading of the microphone are shown. The figure 55 shows the readings during a time space 0-250sec. The graph starts by zero because when the first entry in meetingTable is made the requested sensor readings are not available yet (cp. 4.4.1). The figure 56 shows the time space 1900-2450sec. It shows that the sensor readings are different in every cycle.

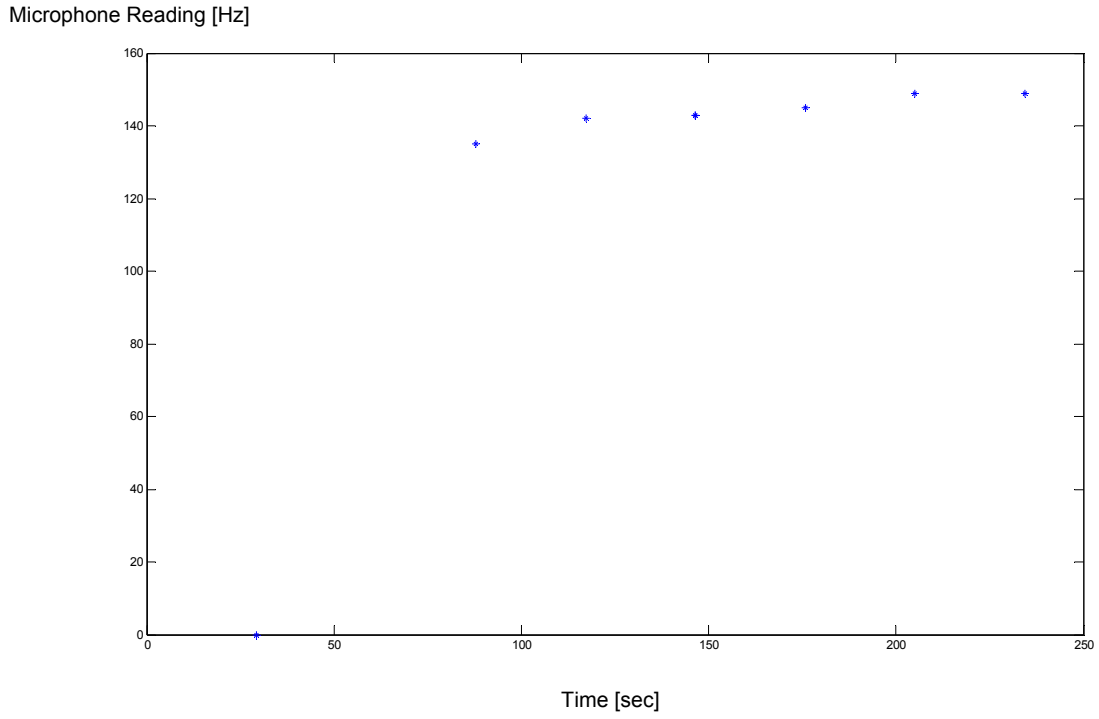


Figure 55: Experiment 5: mote 3, sensor reading of the microphone, 0-250sec

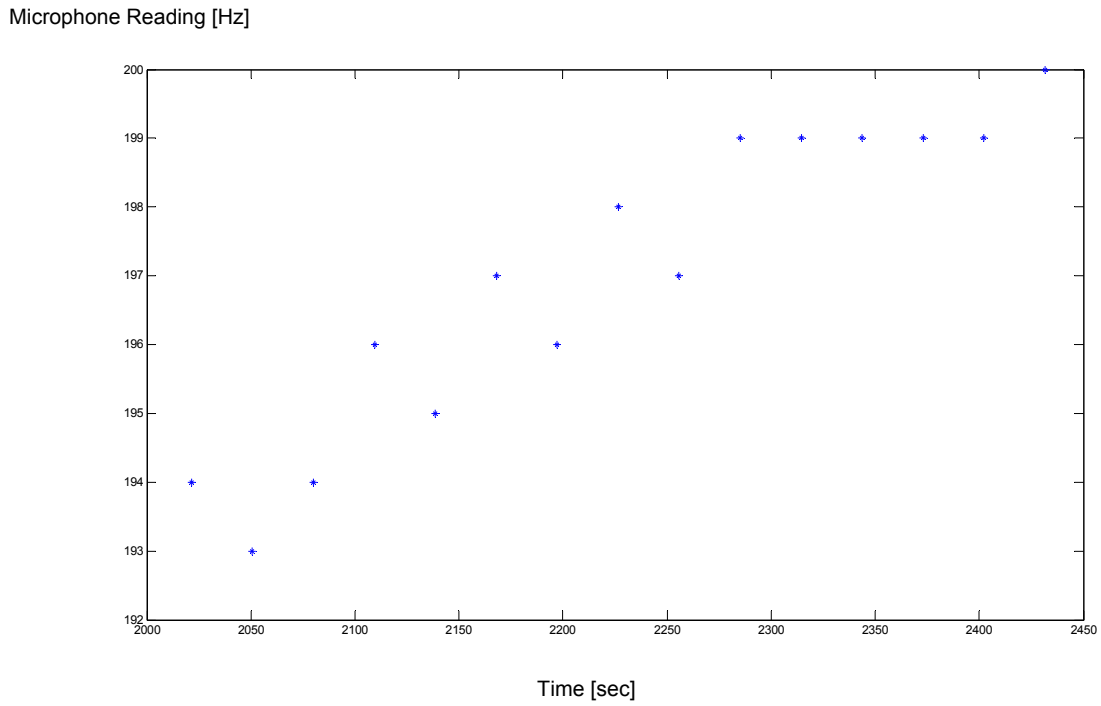


Figure 56: Experiment 5: mote 3, sensor reading of the microphone, 1900-2450sec

In this experiment it is possible to test the functionality of the photo sensor as well. The next figures 57 and 58 show the result of the sensor readings. The first reading is zero because of the same reasons mentioned above. The two figures also show that the readings differ and that the sensor does not always measure the same value. Figure 58 shows the time space 2300-2350sec that the brightness in the surrounding of the mote decreases because the scientist was in a darker area as before and later on.

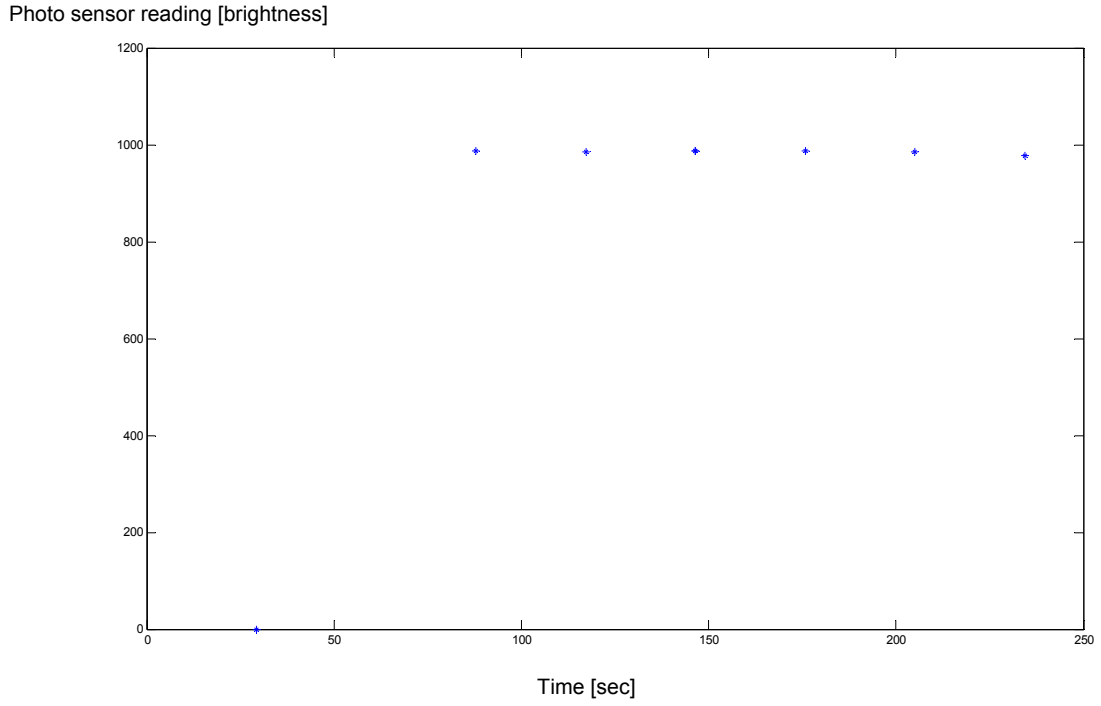


Figure 57: Experiment 5: mote 3, sensor reading of the photo sensor, 0-250sec

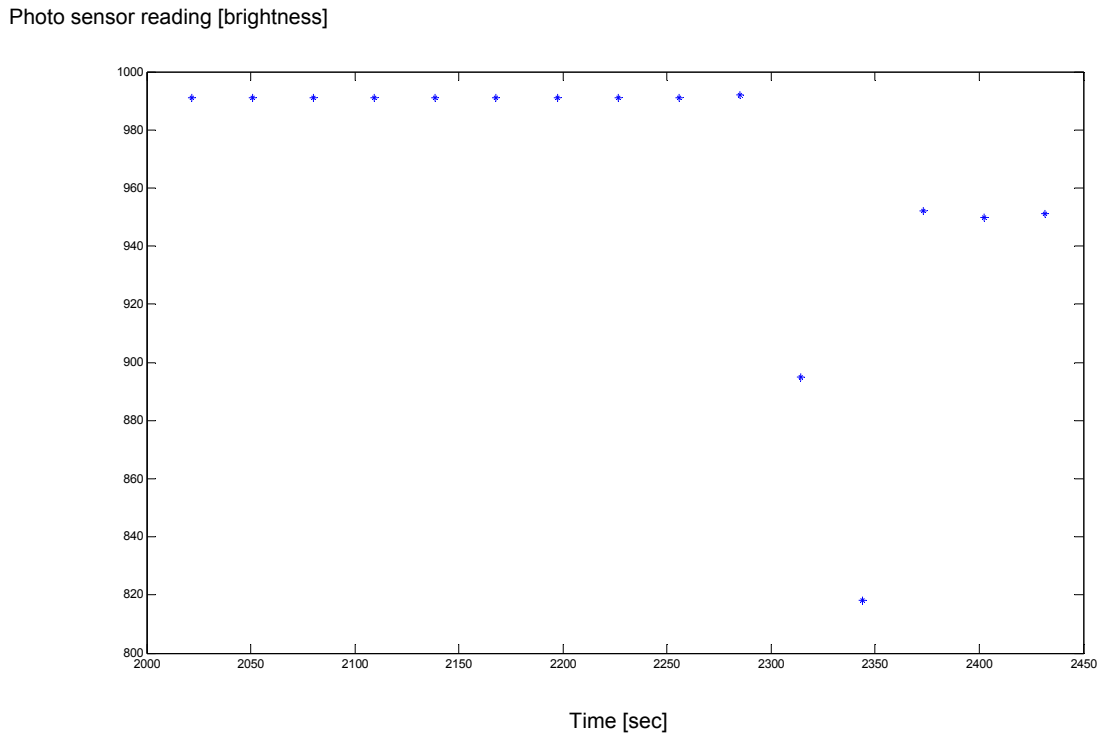


Figure 58: Experiment 5: mote 3, sensor reading of the photo sensor, 2000-2450sec

This analysis shows that the algorithm works even with more than one mobile node. In this experiment it could be shown that the microphone and the photo sensor work. Finally every user must choose the values of the variable corresponding to the experiment condition, especially in consideration of the electro smog in the surrounding.

5. Conclusion and summary

Together with this diploma thesis the behaviour of rats corresponding to their meeting behaviour can be understood better than before. Now it is possible to observe this behaviour by putting the rat a west with a Mica2dot on. This mote is able to send beacons, collecting sensor readings of a microphone and photo sensor, as well as processing received beacons and recognizing meetings. This algorithm is tested in a laboratory with one rat and with several scientists that wear the motes instead of the rats, because the testing arena is too small for observing more than one rat. Additionally to the required algorithm for the motes an analysis method is programmed as well, which is able to produce corresponding to the collecting data of the meetings an application with MATLAB.

One big challenge of this diploma thesis was to get along with small storage and limited power resources by collecting maximal information at the same time. Thus, a feedback reaction by the base station to reduce the packet-loss-rate is not implemented yet. Another challenge was the programming language nesC, and the needed software that did not run stabile at the used system – Windows XP. In the experiments the rats wear wests with a backpack including the Mica2dot. The west handicaps the rat and should be replaced with a flatter construction (Fig. 59) in the future in order to make moving more comfortable.

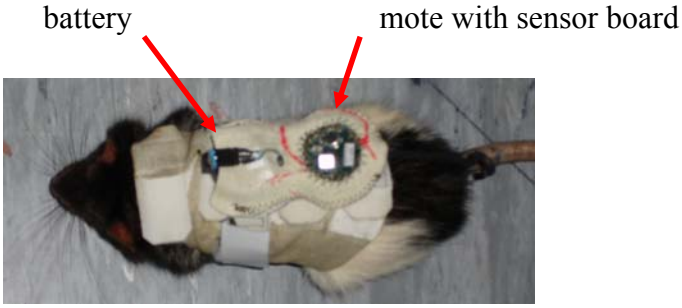


Figure 59: flatter construction of the backpack with divided battery and Mica2dot platform

The experiments have shown that the algorithm works. The quality of the measurements corresponds to the chosen values – firing rate of the timer, threshold and antenna's length. Another important fact is the observed area where the motes can communicate with each other. Is the area small, the paket loss rate is smaller than in bigger areas. The sourounding electrostatic smog is also important for the quality of the measurements.

In the future the development will go on. Another project is in progress to collect accelerometer data that than can be added to the current algorithm. The result will be a possible analysis of the movement of the rats additionally to the meeting behaviour. The aim is to make a map of the rat's burrow corresponding to the movements. Together with the sensor readings of our approach the information can be used to get a more detailed analysis of the behaviour of rats.

The next step is to test the approach in the rat's natural environment with the flatter construction of the backpack with divided battery and a Mica2dot platform used as this thesis' basis. On the one side, the future aim is to get more information of the rat's behaviour, and on the other side to test the requirements corresponding to the programmed values and the conditions of the if-cases in the algorithm. Another aim is to reduce the loss-rate of packets by programming a feedback message of the base station when a packet of meeting data or sensor reading is received. It must be tested if the usage of a feedback message corresponding to the storage space of the mote and the availability of the base station is useful. If the observed area is small, as in our case, and the radio range is set correspondingly, it is not needed. Due to the fast development of the technology, the storage capacity might grow, and the motes might become lighter, which makes an employment with smaller animals possible (e.g. mice and bats).

Figure and Table Catalogue

References

Articles

[Bits06] J.A.Bitsch: **"Implementation of standard internet protocols for sensor motes to observe rats in the wild"** (2006), diploma thesis, University of Tübingen

[Calh63] J.B.Calhoun: **"The ecology and sociology of the norway rat"**, U.S. Public Health Service Publication, 1963

[Ewal06] Thilo Ewald: **„Virtuelle Ressourcen für Kommunikationssystme"** (2006), diploma thesis, University of Tübingen

[HaBi79] D.J. Hartley, J.A.Bishop: "**Home range and movement in populations of Rattus norvegicus polymorphic for warfarin resitance**", The Linnean Society of London (1979, p. 19 - 43)

[HSWC+02] J.Hill, R. Szewczyk, A. Woo, S.Hollar, D.Culler, K.Pister: "**System Architecture Directions for Networked Sensors**" (2002), http://webs.cs.berkeley.edu/tos/papers/tos.pdf (7.2.2005)

[Jack82] W.B. Jackson: "**Norway Rat and Allies**", The John Hopkins University Press (1982, p. 1077-1088)

[JOWM+02] P. Juang, H. Oki, Y. Wong, M. Martonosi, L.-S,S. Peh, D. Rubenstein: **"Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet"**, The Proceeding of ASPLOS-X, San Jose, October 2002, http://www.princeton.edu/~mrm/asplos-x_annot.pdf (17.6.2006)

[Kim05] S. Kim: **"Wireless Sensor Networks for Structural Health Monitoring"**, (2005), http://www.eecs.berkeley.edu/~binetude/work/report.pdf (17.6.2006)

[Körb06] Dipl.-Ing. H.-J. Körber: **"Low Power Wireless Sensor Networks"**, http://www.hsu-hh.de/download-1.3.1.php?brick_id=YYgNZmeQf1vZvACt (11.5.2006)

[LeLe03] P.Levis, N.Lee: "**TOSSIM: A Simulator for TinyOS Networks**" (2003), http://www.cs.berkeley.edu/~pal/pubs/nido.pdf (7.2.2005)

[LLWC03] Philip Levis, Nelson Lee, Matt Welsh, and David Culler: **"TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications",** In Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys) 2003, (Nov. 2003), http://www.cs.berkeley.edu/~pal/pubs/tossim-sensys03.pdf (7.2.2005)

[LMGP+04] Philip Levis, Sam Madden, David Gay, Joe Polastre, Robert Szewczyk, Alec Woo, Eric Brewer and David Culler: "**The Emergence of Networking Abstractions and Techniques in TinyOS**", To appear in Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004), http://webs.cs.berkeley.edu/tos/papers/tinyos-nsdi04.pdf (7.2.2005)

[Pola03] J.R.Polastre: **"Implementation of Wireless Sensor Networks for Habitat Monitoring"** (2003), http://www.polastre.com/papers/masters.pdf (14.10.2006)

[SaSa04] P.Sang, C.M. Sandler: **"Hardware Design Experiences in ZebraNet"**, SensSys'04, November 2004, Baltimore, Maryland, USA, http://www.princeton.edu/~csadler/sensys04.pdf (3.10.2006)

[Schm05] Corinna Schmitt**: "Communication Architecture - Analysis and comparision focusing on communication subsystems -"** (2005), student research project, University of Tübingen

[SMPC04] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler: **"An analysis of a large scale habitat monitoring application"**, The Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys), November 3-5, 2004, www.eecs.harvard.edu/~mdw/ course/cs263/papers/gdi-sensys04.pdf (17.6.2006)

[Stro82] D.C.Stroud: "**Population Dynamics of Rattus Rattus and R. Norvegicus in a riparian habitat**", Journal of Mammalogy (1982, Vol.63(1), p. 151-154)

[TiLP05] B. Titzer, D. Lee, and J. Palsberg, "**Avrora: Scalable Sensor Network Simulation with Precise Timing**". In Proceedings of IPSN'05, Fourth International Conference on Information Processing in Sensor Networks, Los Angeles, 2005

[Webs04] UC Berkeley WEBS Project: **"nesC: A Programming Language for Deeply Networked Systems"** (2004), http://nescc.sourceforge.net/papers/nesc-ref.pdf (11.5.2006)

[WeMM04] M.Welsh, D. Malan, S. Moulton **"Wireless Sensor Networks for Emergency Medical Care"**, Cooperration of Havard University and Bosten University School of Medicine (2004) , http://www.eecs.harvard.edu/~mdw/talks/ge-codeblue.pdf (28.9.2006)

[WhKo05] I.Q.Whishaw, B.Kolb: "**The Behavior of the Laboratory Rat – A Handbook with Tests**", ISBN: 0-19-516285-4, Oxford University Press (2005)

Homepages

[1] UC Berkeley, **TinyOS Project**: www.tinyos.net (11.5.2006)

[2] **Habitat Monitoring on Great Duck Island**: www.greatduckisland.net (17.7.2006)

[3] **Berkeley Wireless embedded systems (WEBS)**, http://webs.cs.berkeley.edu/ (11.6.2006)

[4] **Crossbow Technology Inc.**, http://www.xbow.com/ (11.5.2006)

[5] **National Center for Biotechnology Information (NCBI)**, http://www.ncbi.nlm.nih.gov/ (12.6.2006)

[6] University of Michigan Museum of Zoology, **Animal Diversity Web**, http://animaldiversity.ummz.umich.edu/site/index.html (12.6.2006)

Appendix A – RFPower settings

Pout (dBm)	PA_POW (hex) 433/315 MHz	Current Consumption, typ. (mA)	PA_POW (hex) 433/315 MHz	Current Consumption, typ. (mA)
-20	0x01	5.3	0x02	8.6
-19	0x01	6.9	0x02	8.8
-18	0x02	7.1	0x03	9.0
-17	0x02	7.1	0x03	9.0
-16	0x02	7.1	0x04	9.1
-15	0x03	7.4	0x05	9.3
-14	0x03	7.4	0x05	9.3
-13	0x03	7.4	0x06	9.5
-12	0x04	7.9	0x07	9.7
-11	0x04	7.9	0x08	9.9
-10	0x05	7.9	0x09	10.1
-9	0x05	7.9	0x0b	10.4
-8	0x06	8.2	0x0c	10.6
-7	0x07	8.4	0x0d	10.8
-6	0x08	8.7	0x0f	11.1
-5	0x09	8.9	0x40	13.8
-4	0x0a	9.4	0x50	14.5
-3	0x0b	9.6	0x50	14.5
-2	0x0c	9.7	0x60	15.1
-1	0x0e	10.2	0x70	15.8
0	0x0f	10.4	0x80	16.8
1	0x40	11.8	0x90	17.2
2	0x50	12.8	0xb0	18.5
3	0x50	12.8	0xc0	19.2
4	0x60	13.8	0xf0	21.3
5	0x70	14.8	0xff	25.4
6	0x80	15.8		
7	0x90	16.8		
8	0xc0	20.0		
9	0xe0	22.1		
10	0xff	26.7		

Table 8: RFPower settings [4]

Appendix B – Contents of the CD

- Diploma thesis
- Experiments
  - 4.4.1. first experiment type
  - 4..4.2. second experiment type
- Figures
- Literature
- Programs
  - nesC-Programs
    - RatMote
    - Basestation
  - Java-Programs
  - MATLAB-Programs
- Video of an experiment with the rat