Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

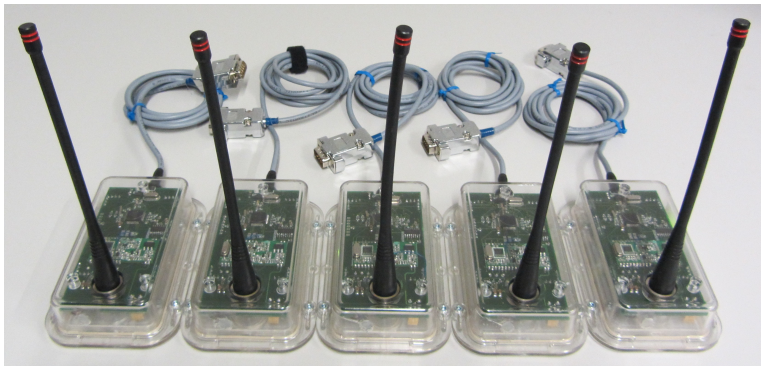
A Wireless Sensor Network for Orienteering Competitions

Master's Thesis

Martin Zoller

mzoller@ee.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich



The five woodAnt radio nodes that we developed and built.

woodAnt

Supervisors:
Philipp Sommer
Johannes Schneider
Prof. Dr. Roger Wattenhofer

November 16, 2011

Acknowledgements

I would like to thank my advisor, Philipp Sommer, for the great support he gave me during this work. His deep knowledge of TinyOS was crucial to understanding and working with this real-time operating system. Even after finishing his PhD studies, he took the time to support me with the last parts of this thesis. I am also very thankful to Prof. Roger Wattenhofer for giving me the opportunity to realize this personal project within his research group. My thanks also belong to Hans-Rudolf Benedickter from the Electromagnetic Fields and Microwave Electronics Laboratory for supporting me with RF measurements, and Thomas Kleier from the Integrated Systems Laboratory for giving me access to, and support with, the steam soldering oven at his laboratory. Special thanks go to Silvan Nellen and Elena Dürst for supporting me with radio range measurements. Further I would like to thank SPORTident GmbH, Germany, for providing me with control stations and waterproof cases free of charge.

Abstract

This thesis deals with the development of a wireless sensor node for use in orienteering competitions. The nodes are used to transmit runners' split times from control points, usually located in a forest, to a base station in the finish area. The developed sensor node is optimized for low price and uses a 500 mW radio module to achieve the required range. TinyOS is used as a software platform, and the Collection Tree Protocol (CTP) currently handles multi-hop routing. Measurements have shown that the radio module used does not comply to its datasheet, so that the desired range cannot be achieved. Possibilities to improve the node's performance in future work are shown.

Contents

Chapter 1

Introduction

Wireless sensor networks are used in many areas. They can be used to measure parameters like humidity or temperature over a large area, or to detect dangers such as fire or rockfalls. This work presents a rather unusual application of such a sensor network: The goal is to observe runners in orienteering competitions.

1.1 About Orienteering

The International Orienteering Federation defines orienteering as follows [1]:

> Orienteering is a sport that combines both a physical and a mental element. The basic idea in orienteering is to proceed from course start to finish by visiting a number of control points in a predetermined order with the help of map and compass. In order to choose the best possible route, orienteers look at the characteristics of the terrain, and the winner is determined by the fastest time to complete the course. What is unique to orienteering is that an orienteer must navigate and make quick decisions while running at high speed.

There are four major disciplines in orienteering [1]: Foot, mountain bike, ski, and trail orienteering. The most common discipline is foot orienteering; it is what the word "orienteering" is usually associated with. All orienteering disciplines share a basic problem: The competitors have to prove that they have visited all control points in the required order. This can be accomplished by placing a stamp or needle punch at each control point and letting competitors stamp or punch a paper card. However, such methods do not verify the order reliably since the card can be punched at any position, i.e. the order of the punches can be changed. Furthermore, a lot of manual labour is required to check the control cards. Nowadays, almost all orienteering competitions use an electronic punching system to avoid such problems. There are two major vendors of such systems, SPORTident [2] and EMIT [3]. SPORTident is prevalent in Switzerland and

its neighboring countries, while EMIT is used mainly in Scandinavia[4] Both systems work with a chip that is carried by the competitor. At each control point, the chip is inserted into a device, and a timestamp is written to the chip. When a competitor has finished his course, the organizer reads out his chip to obtain the running time and split times. A special software is used to process the timing data and create a result list.

1.2 Observing Orienteering Races

As opposed to endurance sports like running or cross-country skiing, there are no fixed routes in orienteering. Each runner can freely choose her route as long as all the control points are passed in the given order. This makes it very hard to observe runners during a race. A possible solution is to place human observers near some control points and let them report the passing runners over a radio handset, but this would not provide very accurate data. Since each runner has an individual start time, it would be hard to determine who is leading the race. Furthermore, such observers would make it easier to find the control points, and only a small fraction of the points could be observed this way.

A better solution is to benefit from the technical infrastructure which is already available. The electronic timing boxes at the control points can output a message which can be transmitted over the air with a radio mode. Several such devices have been built by enthusiasts and used at major events [5], [6]. An example from Switzerland is shown in Fig. 1.1. These devices usually operate on an amateur radio band and establish a direct communication link to the base station. This requires a lot of power, especially in hilly terrain: The Swiss system, for instance, uses 5 W at 425 MHz. This is sometimes not enough and it becomes necessary to add a relay station halfway to the finish. To supply the radio modems with power for the entire event, large batteries are required. This makes the radio systems inconvenient to use because the control points are usually placed on foot, i.e. many of them have to be carried at once.

Several options are available to make the system more compact. If GSM coverage is available in the entire competition area, GSM modems can be used. Commercial solutions for this case exist [7]. If no coverage is available, the range can be extended by using a multi-hop network: Each radio node communicates only with its nearest neighbors. Since the controls in orienteering competitions are always within about 1 km of each other, this greatly reduces the required range. Still, common wireless sensor nodes such as TinyNode or Shimmer are not sufficient since they use IEEE 802.15.4 compliant short-range radio. Therefore a high-range radio node for orienteering controls has been developed. A multi-hop network based on this node will be presented in this work.

Figure 1.1: Radio system used for large orienteering competitions in Switzerland.

Chapter 2

Problem Specification

The radio system to be designed has to meet a range of requirements. They arise from the planned use of the system for orienteering competitions: Radio nodes may be placed outdoors in any weather conditions, and they must have a certain range even in uneven terrain. Furthermore, it must be legal to operate the units anywhere in Switzerland or even throughout Europe, without obtaining a license for each use.

2.1 Legal restrictions

One of the goals of this project is to make radio controls more widely available. It should become possible to use them even at smaller orienteering competitions, with a modest budget, and with a minimal amount of extra work. Therefore it is crucial that they can be operated without a license. The existing Swiss radio controls use a licensed band [8], but this makes it necessary to plan their use well in advance, which makes it less attractive. With the multi-hop approach used in this work, it should be feasible to build a license-free system using one of the available frequency bands. A comparison of these bands is given in Section 4.1. The radio functionality must be provided by a third-party module that complies to the relevant EU standards and norms. The use of a license-free band implies that there may be interference from other devices, but this should not be an issue in general since most orienteering competitions are not held in urban areas.

2.2 Range Requirements

Since we have a multi-hop radio system, it is not necessary that each node can establish a connection to the base station. It is sufficient for nodes to have at least one reliable connection to a neighbor. Packets are then forwarded towards the base station, which collects data from the entire network. In usual orienteering competitions, there may be up to 1 km distance between a control point and its

nearest neighbor. It is therefore desirable to achieve a range of 1 km in forests. If the conditions are particularly bad for radio (e.g. wet and dense forest), this may be hard to achieve with a transmit power of 500 mW. It is then still possible to place some extra radio nodes in the forest to ensure that each control point has a connection.

2.3 Physical Constraints

In contrast to the existing Swiss radio system for orienteering, the system that we are designing in this work is based on a multi-hop network. Radio nodes must be placed at (almost) every control point of an orienteering competition. This mandates a compact design: The controls are usually placed on foot, and several of them are carried by one person. It is therefore important that the radio nodes do not add too much weight and volume to the controls. Their dimensions should be comparable to those of SPORTident control stations which are used for time taking today: They are $115 \times 62 \times 32$ mm large and weigh 148 grams. It would be ideal if we could use the same casing for our nodes, since there are control posts with two holders available that could accomodate a SPORTident unit and a radio node in this case. Future versions of the radio system may even be integrated into the SPORTident unit's case, however this would require modifications to the unit itself.

An additional difficulty is the environment in which the nodes are operated. Some of them may be surrounded by dense forest, the terrain may be hilly, and there may be differences in altitude between the nodes. These factors attenuate the radio signals, as described in Section 4.2. If the conditions for a node are particularly difficult, e.g. if it is located in a depression, additional relay nodes may be placed in its surroundings to provide a reliable connection.

The radio nodes must work in any weather conditions that may occur in Europe. They must be waterproof, and the operating temperature range should be from -20°C to around 40°C. For the first prototypes the range may be smaller: Most electronic components are available for several temperature ranges. It makes sense to use easily available versions of the parts in the first prototypes and replace them by the most temperature-tolerant ones in a later version of the design. Waterproofness is very important even for the prototypes, however.

The mobility of the nodes must be taken into account as well. To make the handling of the radio system easy, it should be possible to turn on the nodes before placing them in the terrain, i.e. connections that were established initially may be permanently lost later, and new connections may become available in-

stead. This mandates that each node sends out periodical beacons to announce its presence and discover its current neighbors. In order to ensure reliable connections to all nodes, it is also desirable to provide a link quality display to the user. This makes it easier to decide whether additional relay nodes must be added to the network.

2.4 Power Constraints

The radio system is going to be used for different kinds of orienteering competitions. While a sprint race with few runners can be held within an hour, a long-distance race with many competitors will take much longer. Runners will start within a window of up to four hours, and even fast runners will sometimes need more than two hours to finish their race. Usually the control points are placed at least an hour before the first start. Therefore, to ensure that live results are available throughout the race, the power supply of the nodes should last at least eight hours.

On the other hand, the size of the supply is limited by the size of the radio node. Many wireless sensor networks use solar cells for autonomous operation, but given the different locations and weather conditions in which the nodes may be operated, and the short timeframe of eight hours, solar cells would provide little or no benefit. A battery is required even if a solar cell is used, and since today's cellular phones easily achieve more than eight hours of autonomy, it seems realistic to achieve this with our system. Depending on the distance to the base station, a GSM handset may send with up to 1 W, and just like in our system, periodic beacons must be exchanged to provide coverage information to the use. More details and calculations on the power supply are provided in Section 6.5.

Chapter 3

Related Work

This project was originally motivated by the radio system used in Switzerland. It is owned and operated by the "Verein für Elektronische Posten und Zeitmessung (VELPOZ)". The system is described in more detail in the semester thesis on which this work is based [8]. Its main drawbacks are the requirement of a license for every use, the large size and weight of the radio nodes, and the lack of automatic routing mechanisms. However, it uses a transmit power of 5 W and often has sufficient range for a direct connection from control points to the finish area.

SPORTident, a manufacturer of electronic punching systems for orienteering, also offers some systems for wireless data transmission between control points. The company has developed a proprietary short-range radio system with a range of up to 10 meters [9], [10]. This system is intended for local communication at control points. It can be used to transmit punch data from RFID control stations to long-range radio boxes, or directly to a laptop via a USB dongle. For long-range communication, SPORTident provides a device called WBOX GSM [11], which collects timing data from nearby RFID control stations and uploads them to a central server via a cellular network (GSM or GPRS). Since legacy live results software only supports data input over RS232, another WBOX GSM unit can be used to download the data and feed them into the software through a serial cable. Obviously this system can only be used if a cellular network is available in the terrain. An overview of the system is given in [7], and a blog entry describing practical experiences with the system is available under [12]. SPORTident has also made attempts to create a higher-power radio system, similar to the one we have developed in this work, but this system was not commercialized. We studied the resulting report [13] as a basis for our project.

To find suitable technical solutions for our radio nodes, we compared existing wireless sensor networks. Popular sensor nodes like the Tmote sky or the TinyNode [14] use radio chips with only 10 mW transmit power, which does not provide a sufficient range. We thus considered only sensor nodes for special ap-

plications with a range of more than 1 km.

One such system is the AQUAMON Soil Monitoring Network [15], which operates at 900 MHz with 500 mW transmit power. It has a reported line-of-sight range of 10 miles, i.e. around 16 km. However its radio nodes are too large for our application since they include additional sensors and control mechanisms. It is possible to use solar panels to recharge the nodes' batteries. This is an interesting approach, but it does not seem like a good option for our system: In the forest there may not be enough direct sunlight to efficiently operate a solar panel. Furthermore, the panel's yield depends strongly on the weather conditions, i.e. the battery needs to last several days without solar power. Since our system will primarily be used for single-day races, a solar panel would thus not add any benefit.

Another long-range Wireless Sensor Node is the EM50R from Decagon [16]. It also uses the 900 MHz band and achieves a range of up to 3 miles (almost 5 km) according to its manufacturer. The Decagon sensors have been used in several research projects. In [17] they were used to measure soil moisture in greenhouses and thereby optimize irrigation. The nodes can be operated for up to a year with five AA type batteries. This shows that it is possible to build very economic sensor nodes while maintaining a high radio range. Unfortunately no technical details are available about the radio unit used in the Decagon nodes.

Chapter 4

Physical Layer

4.1 Frequency Allocations

After studying the Swiss frequency allocation plan [18] and the corresponding EU document [19], we conclude that the following frequency bands have to be considered for high-power applications:

- **169 MHz:** Some channels in the frequency band from 169.4 to 169.8125 MHz, particularly the lowest three 25 kHz channels, may be used for "Tracking, Tracing and Data Acquisition". Devices may send with up to 500 mW *Effective Radiated Power (ERP)*, at a duty cycle of at most 1%. Since the purpose of our system is, in a way, to *track* persons, it may qualify to operate in this frequency band. However, it remains to be seen whether a duty cycle of 1% is sufficient. The allocation is valid at least in Switzerland [20] and Germany [21]. It should be available in most European countries since the EU Commission decided to harmonize this band in 2005 [22], with an amendment in 2008 [23].

- **173 MHz:** The frequency band from 173.0875 to 173.3625 MHz is assigned to *Non-specific Short-range Devices (SRD)* in Switzerland. It is split into four channels with up to 25 kHz bandwidth and 500 mW ERP. One of the channels even allows for up to 2.5 W ERP. However, this frequency band is not available in the rest of Europe, i.e. its use would limit the scope of the radio system to Switzerland.

- **433-434 MHz:** The 433.05 MHz to 434.79 MHz band is also a *Non-specific SRD* band. Internationally it is known as an ISM band (*Industrial, Scientific and Medical*). All of its channels can be used with less than 10 mW ERP and 10% duty cycle. The range from 433.2375 to 434.5125 MHz is also available for *higher-power SRDs* with up to 500 mW ERP, or up to 2.5 W in eight of the channels. The maximum channel spacing in this band is 25 kHz. While the ISM band is available in most of Europe, the allocation for higher-power devices is only valid for Switzerland.

- **863-870 MHz:** Most of the frequency band between 863 and 870 MHz can be used for *Non-specific SRDs* with no more than 25 mW ERP, provided that the duty cycle is very low or a collision avoidance scheme is used [24]. The frequency band from 869.400 to 869.650 MHz allows for up to 500 mW ERP on its 10 channels with a spacing of 25 kHz. These regulations are valid in Switzerland and the European Union [19]. The 869.400 to 869.650 MHz band is the only band that allows for 500 mW ERP in the EU.

In conclusion, there are several license-free frequency bands available in Switzerland which allow for 500 mW transmit power. If the system is to be used in other EU countries, the choice is limited to the 869 MHz band or possibly the newly harmonized 169 MHz band for tracking and tracing.

4.2 Signal Attenuation in Forests

Our radio system is to be deployed in various terrains where orienteering competitions are held. Competitions in urban areas usually have a large density of controls and are not critical for range requirements. The key question is thus how much attenuation has to be assumed in forests, i.e. how much the range of a radio link will be reduced with respect to a line-of-sight connection. This question has been discussed in the semester thesis preceding this work [8]. Experiments by Tamir [25] have shown that the signal attenuation is proportional to the frequency, i.e. a lower frequency allows a longer range with the same transmit power. Tamir suggests the following the following formula for the *basic path loss* L_{b0}:

$$L_{b0} \;=\; 1570 \left[|n^2 - 1|\ \mathrm{Re}(n)\right]^2 \left(\frac{\rho}{\lambda_0}\right)^4$$

where ρ is the distance between sender and receiver, λ_0 is the signal wavelength, and n is the refractive index of the forest medium. The formula is based on the assumption that nodes are located close to the tree tops. However the measurements presented in the paper do not appear to match the formula particularly well. The trend is clear, however: The higher the frequency, the more the signal will get attenuated. The highest frequency studied is 400 MHz, and the measured path loss at a distance of 1 km is 120 dB at this frequency. A more recent paper by Chen and Kuo [26] introduces an empirical formula for frequencies between 1 and 15 GHz and verifies that it matches the results. For vertical polarization, the path loss formula is given by

$$L_{v-v}\ [\mathrm{dB}] \;=\; (0.001 \cdot f + 0.2) \cdot d + 0.5 \cdot f + 3$$

where f is the radio frequency in GHz and d is the distance in m. For 400 MHz and 1 km distance, we obtain a path loss of 203.6 dB. This is a huge difference

to Tamir's result, however it is doubtful whether the model can be applied for such low frequencies.
In a recent paper by Gay-Fernández et al. [27] the problem of signal propagation in forests is applied to wireless sensor networks. Experiments have been performed with radio nodes based on the *CC2430* ZigBee-compatible radio chip, operating on the 2.4 GHz band. Some of the nodes acted as coordinators and featured directional antennas while most of the nodes had integrated antennas without gain. In experiments the nodes were attached to trees, and their range was measured with the nodes mounted towards each other, or at the back side of the tree so that no line of sight was present. The range was shown to be reduced by a factor of more than three if the transmitter is mounted on the side *not* facing the receiver, and more than eight if the receiver is behind its tree as well. A series of measurements were made with a central transmitter and many receiver nodes along the propagation path, so that the attenuation could be measured. The received power in dBm was linearly attenuated:

$$P\ [\text{dBm}] \ = \ P_0\ [\text{dBm}] - 10 \cdot n \cdot \log(d\ [\text{m}])$$

where d is the distance from the sender, and P_0 and n are obtained by doing a linear regression on the measured power/distance pairs. The experiment was performed for receivers each mounted on a tree, at four different angles w.r.t. the propagation path. In all cases, the rate of decay n was larger than in open environments.

In summary, several models exist for radio attenuation in forests, and the problem has also been investigated for wireless sensor networks. However, to the best of our knowledge, no results have been published about the performance of sensor nodes with higher-power radio modules (e.g. on the 869 MHz SRD band) in forests. Since the propagation characteristics heavily depend on frequency, we cannot reliably predict the range from existing data. What we can predict is that radio systems at lower frequencies will experience less attenuation and thus better range. An ideal radio system for our application should thus operate on 169 or 173 MHz.

4.3 Basic Antenna Theory

In order for our radio node to achieve a good range, it is important that suitable antennas are used. The antenna is a key design element since it determines what fraction of the RF signal's power is effectively emitted from the device. Furthermore, each antenna has a specific radiation pattern. For our application, this pattern should be as even as possible: The nodes should work without rotating their antennas towards their closest neighbors. Also, this orientation might not be optimal, depending on the terrain and possible wave paths. Since we cannot

test the radio system before all nodes have been placed, it is not possible to adjust each node's antenna to optimize reception. Therefore the best solution is to use omnidirectional antennas, so that no adjustments are necessary. If some nodes have a bad connection and do not work as expected, we can still add relay nodes to the network, but most of the nodes must work "out of the box".

A key observation in antenna design is that the transmitter antenna is not always optimized for efficiency. In some cases, an inefficient antenna is used e.g. to reduce size and cost. The transmitter's output power can then be increased above the legal limit since only a fraction of the power is radiated into free space. The receiving antenna, on the other hand, cannot compensate losses and should thus be as efficient as possible.[28]

4.3.1 Transmission Lines

A transmission line is a means of transferring RF energy from one place to another. Its properties depend on many parameters, such as the diameter and spacing of the conductors used, or the dielectric constant of the materials that surround and separate them. A key property of transmission lines is their *characteristic impedance*: It is defined as

$$Z_0 = \sqrt{\frac{R + j\omega L}{G + j\omega C}}$$

where R is the resistance, L is the inductance, and C is the capacitance of the line per unit length, G is the conductance of the dielectric per unit length, and ω is the angular frequency. Lossless transmission lines have zero resistance and their dielectric has zero conductance. Thus, their characteristic impedance is real and frequency-independent: $Z = \sqrt{L/C}$. [29].

4.3.2 Microstrip Lines

A microstrip line is a kind of transmission line that can be built using normal traces on a PCB. It consists of a conducting strip, a layer of dielectric (usually FR-4) and a ground plane. The required width w of the trace in order to obtain a $50\,\Omega$ matching can be calculated from the dielectric constant ϵ_r of the substrate, the signal frequency f, the thickness h of the substrate, and the thickness t of the strip metallization. No closed-form expression for the width exists, but web-based tools such as [30] can be used to calculate the width of microstrip lines easily.

4.3.3 Antenna Polarization

The antenna polarization describes the orientation of an antenna with respect to ground. When an antenna is parallel to its ground, it is said to be *horizontally polarized*. If it is perpendicular to ground, it is *vertically polarized*. Usually the polarization of an antenna should match its orientation with respect to the physical ground for maximum efficiency. The transmitter and receiver antenna should always have the same polarization to avoid energy losses. In the VHF and UHF spectrums, horizontally polarized antennas provide a better noise immunity and are less prone to fading than vertically polarized antennas [28].

4.3.4 Quarter-wave Monopole Antennas

In frequency bands where a $\lambda/2$ dipole antenna would be too long, such as in the 433 MHz ISM band, quarter-wave monopole antennas are a popular alternative. They provide only one of the quarter-wave elements of a dipole, while the second one must be provided by a ground plane on the product's PCB. This plane serves as a counterpoise to the antenna. If possible, the antenna should be centered on the ground plane, as antennas mounted on the edge of the plane will not have a uniform radiation pattern. Quarter-wave monopoles may be built with the antenna perpendicular to the ground plane, or with the antenna in the same plane as the ground [31]. Mounting the antenna in parallel with the ground plane is very common e.g. in cellular phones. Thereby the length of the ground plane in the direction of the antenna is critical. For optimum performance it should be a quarter-wavelength long.

Chapter 5

MAC and Routing in Wireless Sensor Networks

5.1 Medium Access Control

Many different schemes have been proposed to handle MAC in Wireless Sensor Networks. In fact, there are so many that they have been called the "MAC Alphabet Soup" [32]. We provide an overview of the most important ones to show the challenges of MAC in multi-hop networks and possible solutions to them.

TDMA

Time Division Multiple Access (TDMA) is a channel access scheme for networks with a shared medium. It is used in many classic wireless systems such as GSM and DECT [33]. Its principle is to assign a time slot to every node, and to let the nodes take turns in transmitting their data. When a node has nothing to transmit, the channel remains unused in that node's time slot. This is a problem in large networks: The slots become very small, i.e. if some of the nodes generate bursts of packets in a short time, the network will easily get congested. Furthermore, the assignment of slots is difficult in large networks without a coordinator node. Another major issue is clock synchronization, which does not scale well to larger networks or multi-hop networks. Timing failures drastically reduce performance because no carrier sensing is used, i.e. collisions are very probable if the clocks are not synchronized. Clock drift compensation schemes can be used to minimize the difference, but a spacing must always be left between slots to accomodate timing differences.

S-MAC and T-MAC: More Flexibility

As an adaptation of TDMA for sensor networks, the *S-MAC* scheme [34] has been proposed. It divides the network into virtual clusters with common sleep schedules, without introducing active cluster heads. This improves scalability and multi-hop workability. Latency may be increased in comparison to traditional MAC protocols like CSMA/CA, but the energy consumption is greatly reduced due to shorter periods of idle listening. The principle is simple: When a node is turned on, it listens for *SYNC* messages for some time. If it receives a *SYNC* from a neighboring node, it assumes the same schedule as that node. Otherwise, it creates its own schedule and starts occasionally transmitting *SYNC* messages. The S-MAC scheme is similar to the Distributed Coordination Function (DCF) in IEEE 802.11, but it does not require *all* nodes to be synchronized and is thus better suited for multi-hop networks.
T-MAC is similar, but introduces a variable duty cycle by extending the active period of nodes when they encounter certain events. This allows for receiving several packets in one cycle. Both protocols utilize RTS/CTS packets to avoid the hidden terminal problem: Only when the channel appears free to both the sender and the receiver, the packet is sent. Neighbors of the sender and receiver may overhear the RTS and/or CTS packet and will then refrain from starting transmissions themselves. In T-MAC, this situation can cause the addressee of a waiting packet to go to sleep before the packet can be sent. This can be avoided by using a mechanism called *Future Request-to-Send (FRTS)*: The sender that has lost the contention sends an FRTS packet directly after the CTS packet it overheard. This instructs the addressee of the waiting packet to stay awake. Additionally, *full-buffer priority* can be used to avoid overflow of nodes' transmit queues: When a node with a full queue receives an RTS, it replies with an RTS of its own, i.e. it has priority over other senders.

B-MAC: Low Power Listening

The *B-MAC* scheme [35] is included in TinyOS by default. It is a lightweight access scheme that contains mainly two mechanisms: *Clear Channel Assessment (CCA)*, and *Low Power Listening (LPL)*. CCA is a form of CSMA, but it takes into account that the noise level in the radio channel depends on node location and is influenced by multipath effects. It compares the results of several consecutive RSSI measurements to decide whether the channel is free. LPL is a way to reduce the duty cycle of nodes without time synchronization: Each node periodically wakes up and enables its receiver; when it does not hear any activity, it goes back to sleep. To make sure that packets are always heard by their addressee, they are sent with a preamble at least the length of one sleep cycle. Thus the receiver will wake up at one point during the preamble, and stay awake until the end of the data packet. This access scheme is simple and

energy-efficient; it provides a good channel utilization and has a low overhead. An implementation of it is included in TinyOS by default. To address the hidden node problem, RTS/CTS may optionally be enabled in B-MAC. Unfortunately, B-MAC cannot be used with IEEE 802.15.4 compliant radio chips such as the Chipcon CC2420 since the preamble length is limited there [36].

X-MAC: Improved LPL

X-MAC [37] is an attempt to address some shortcomings of the B-MAC protocol: In B-MAC, all neighbors of a transmitting node are kept awake during the entire preamble, which wastes energy. This is addressed by replacing the long preamble by a series of short packets with pauses in between, that each contain the ID of the target receiver. Uninvolved neighbors can thus go back to sleep after receiving one such packet. Furthermore, the target receiver may send an acknowledgement directly after a preamble packet to indicate that it is awake. The sender can then start transmitting its data packet earlier, which further improves energy efficiency. To optimize latency and efficiency, the receiver duty cycle can be adjusted dynamically. The hidden node problem is not taken into consideration in this protocol.

Z-MAC: A mix of TDMA and CSMA

The *Z-MAC* scheme combines a TDMA-like channel access scheme with a contention-based scheme: Each node has a time slot in which it has priority to use the radio channel. Slots are assigned using the distributed DRAND algorithm, which allows for two nodes to use the same slot if they are more than two hops apart. The slot *owner* must always start transmitting at the beginning of the slot. If it has nothing to transmit, other nodes may use the slot shortly after. This means that the scheme is comparable to CSMA in periods of high contention. Since carrier sensing is always used, Z-MAC is robust to synchronization errors and topology changes. To avoid the hidden terminal problem, *Explicit Contention Notification (ECN)* messages are sent out by nodes when they sense a high contention. They are forwarded to selected two-hop neighbors and cause them not to initiate transfers.

SCP and Dozer: Ultra-low Duty Cycle

Special protocols have been developed for sensor nodes which are deployed for a long time and collect data at large intervals. It is essential that such nodes can maintain a duty cycle as low as possible, so that they stay in operation for several years with a compact Lithium battery. All of the MAC schemes presented above

need a minimum duty cycle on the order of 1% to remain synchronized. With techniques such as *Schedule Channel Polling (SCP)* [36] it is possible to reduce the duty cycle to about 0.1% while adapting well to variable traffic. This is achieved by combining scheduling and channel polling: Nodes synchronize their polling window with their neighbors, so that it is no longer necessary to send long preambles for packets to be detected. The synchronization is done like in S-MAC, but the wake-up tone duration and the interval at which they are sent have been optimized for lowest power consumption. To handle varying traffic well, nodes poll the channel at shorter intervals after they have received a packet.

Another approach to achieving very low duty cycles is called *Dozer* [38]. It consists of a MAC layer, topology control, and routing protocol. By combining all these task in one system, a high efficiency can be achieved. A tree topology is constructed over the network, and locally a TDMA scheme is used: Each node is assigned an upload timeslot by its parent and assigns such timeslots to its children. Periodical beacons are sent by all connected nodes. To reduce the risk of collisions between unrelated schedules, a pseudo-random delay is introduced at the end of each TDMA round. Connected nodes can predict the length of this delay since the seed of the random number generator is included in the beacons. Unconnected nodes can reply to beacons with a busy tone to enable a contention window in which they can send their connection request with a random delay. This minimizes idle listening since the long contention window is only present when needed. A weakness of Dozer is the rather high latency of packet delivery which could be further optimized. Dozer features a suspend mode that is activated when no channel activity is detected for a certain amount of time. In this mode, the node periodically wakes up to poll the channel, without ensuring that it will notice *all* activity. The queue manager of Dozer uses link acknowledgements and includes information about queue size in the acknowledgements, so that nodes know when their parent's queue is full. Also the periodic beacons are used to transfer additional information: Commands from the source to nodes may be disseminated in the network through the beacons - although with a significant delay.

5.2 Routing and Topology Management

5.2.1 Classic Routing Paradigms

We present a short summary of common routing paradigms used in computer networks. Then the special routing requirements of ad-hoc wireless networks are discussed.

Proactive Routing

In proactive routing protocols, each node maintains a database with routing information and keeps it up to date by sending and receiving update packets. The goal is to include as many destination nodes as possible in the database, irrespective of whether there are packets to be sent to these nodes. Proactive routing protocols tend to be efficient in networks with low mobility.

A well-known proactive routing paradigm is **Distance Vector Routing**. It is based on information exchange with direct neighbors. Nodes keep a table of all destinations that they know routes to, and store the next hop as well as the total number of hops to each destination. Each node sends its neighbors its "view of the world", i.e. a list of destination nodes with hop counts. Arriving updates from neighbors are processed: New destinations are added to the table, and known ones are updated when shorter routes to them are received. When the protocol has converged, each node has a complete destination list containing only shortest paths. When the cost of a path increases or a connection is lost completely, routing loops may result because nodes still advertise routes using the previous cost of that path. To avoid such problems, techniques like *split-horizon* and *poisoned-reverse* are applied. Distance vector routing is used by internet protocols such as *RIP*.

Another popular routing solution is **Link State Routing**. It uses a larger routing database that contains complete paths to all known nodes. Updates are sent using flooding. The protocol is based on the Shortest-Path-First algorithm by Dijkstra [39], which has a good convergence and always results in a spanning tree of the network. However the protocol is more complex than Distance Vector.

Reactive Routing

The concept of reactive routing is to discover only the routes that are required to transmit already waiting packets. Routes may be stored in a cache, similar to the routing database of proactive protocols, but they are not systematically discovered. This approach is useful for networks with high mobility of nodes and rather rare transmissions.

A simple example of a reactive routing protocol is **Flooding**. Packets are simply forwarded through the entire network until they reach the destination. To reduce the traffic overhead, the flooding can be slowed down and a faster "stop packet" can be flooded to prevent further forwarding after the data has arrived. Still, flooding does not scale well.

A similar approach is called **Source Routing**: A test packet is flooded, and the route is recorded in each copy of the packet. Then the destination node sends a reply that goes back along the same route. The source caches the route and starts sending data packets over it. Nodes along the route may maintain a cache of their own, so that they can answer future requests without forwarding the packets.

Routing Requirements for Wireless Sensor Networks

Usually Wireless Sensor Networks have a high mobility to data ratio: Due to varying conditions, the topology of the wireless network may change over time, and only small amounts of data are generated. However the nodes are not usually moving, although we could imagine such scenarios. In a classic network with these properties, we might opt for reactive routing. But since the most important issue in WSNs is to keep energy consumption low, unnecessary transmission must be avoided. Proactive protocols are therefore the better choice in most cases. The routing requirements in WSNs are often simple in that all data is to be collected at one node, the *sink*. This also applies to our system, where we need all information to be forwarded to the finish area. The main challenge is to find a reliable routing protocol that keeps energy consumption low and runs on modest hardware resources, e.g. a microcontroller with just a few kilobytes of RAM. Some examples of such protocols are given in the next section. A special challenge that arises from the limited resources is *topology management*: It is often impractical for a node to keep a list of all neighbors that it ever received data from. Algorithms must be found to determine the most reliable neighbors, so that information about others can be discarded. Another task of topology control is to balance traffic load between nodes.

5.2.2 Routing Protocols for Data Collection

A large number of routing protocols have been developed for use in WSNs. We list some notable examples of collection protocols here; all of them could be used to handle routing in our radio system.

MintRoute

The paper by Woo et al. [40] studies possible algorithms for link quality estimation, topology control, and routing. A common link estimation algorithm is the *Exponentially Weighted Moving Average (EWMA)*. The paper presents an extension to it, the *Window Mean with EWMA (WMEWMA)* link estimator: It

calculates an average success rate over a time period t and smoothens the average with an EWMA. This algorithm is shown to perform better than only an EWMA. For topology control, the *FREQUENCY* algorithm is proposed, which keeps a list of good neighbors based on how often successful communication occurs with them. As a metric for route choice, *Minimum Transmission (MT)* is suggested: It selects the route with the smallest expected number of transmissions, including retransmissions. An implementation of these concepts is available in TinyOS 1.x [41].

Dozer

The key concepts of Dozer have been discussed in Section 5.1. Its routing is based on a data gathering tree with exactly one root. Each node maintains a list of potential parents of which the highest-rated one is selected when the connection to the current parent is lost. Rating is based on nodes' distance to the root as well as their load, represented by the number of direct children. This information is advertised by all connected nodes in their periodical beacons. To connect to a new parent node, a busy tone is sent in reply to that node's beacon. The node then keeps listening during a contention window, and a connection request can be sent after a random delay. Even though this procedure allows for several simultaneous connection requests to the same node, only one request is processed per round. This helps balance the traffic between several nodes in a close neighborhood.

CTP: The De-facto Standard

CTP is the default routing protocol for collection in TinyOS. It was specified in *TEP 123* [42], and an implementation of it is discussed in [43]. In CTP, a network may have several roots, and nodes select a root by selecting a next hop to send their packets to. The routing gradient used is *ETX*, the number of expected transmissions. Loops are addressed by transmitting an ETX value in each packet and detecting when child nodes have lower ETX values than their parent. In this case a beacon is sent out to inform neighbors of the updated routes. Routing frames are also sent when a neighbor requests them using the *Routing Pull* flag in a packet. To avoid infinite loops in disconnected fractions of a network, routes with an ETX value above a defined threshold are ignored. Duplication of packets in case of lost ACKs is avoided by using sequence numbers, as well as an additional *Time Has Lived (THL)* field to avoid dropping packets when a routing loop occurs. A packet is thus only assume to be a duplicate if its sequence number, originator, and THL match a previously received packet.

Chapter 6

Hardware Selection and Design Decisions

6.1 Radio Module and Frequency Band

We evaluated a large number of radio modules from different manufacturers. There were three main criteria: The transmit power had to be high enough to achieve a sufficient range (i.e., as close as possible to the legal limit of the respective frequency band), the module had to be small enough to build a light-weight and compact node, and it had to be cheap enough to use it in an end product later. Some of the modules we considered are listed in Appendix D.1.

From Section 4.2 we know that the frequency should be as low as possible in order to achieve a good range in forests. The ideal frequency band would thus be the 169 MHz band for asset tracking and tracing. We found several modules from Radiometrix that operate in this band, but they seemed too expensive for the application. One of the modules in our list was much cheaper than all the others: The RFM12BP from Hope Microelectronics. It was available for the 433 MHz and 869 MHz bands. We expected to achieve a sufficient range with 500 mW transmit power at 433 MHz, and thus the module seemed worth a try. In the 433 MHz band, use of the radio nodes would be limited to Switzerland, but since an 869 MHz version of the module was available, we thought it would always be possible to build EU-compatible nodes without changing board layouts or software.

6.2 Antenna

We considered several types of antennas, such as chip antennas, on-board loop antennas, dipole and monopole antennas. The easiest antennas in terms of design are monopole and dipole antennas. For a center frequency of 433 MHz, a

half-wave antenna would be 34.5 cm long, i.e. it would make our nodes too large. Therefore a quarter-wave monopole antenna was chosen.

To keep the design waterproof, the antenna had to be mounted to the case with a seal ring. We therefore discarded the option of using a cheap antenna that can be directly attached to the PCB. Further criteria were the ruggedness of the antenna and its characteristics: Helical antennas are compact, but under suboptimal conditions in a forest they may be less efficient than a full-length antenna. We therefore selected simple, straight quarter-wave antennas from *Antenna Factor* for both hardware revisions.

6.3 Software Platform

In order to write a powerful software for our radio nodes, we needed a software platform to base it on. There are many real-time operating systems available for microcontrollers, e.g. *pico]OS* [44], *FreeRTOS* [45], or *Contiki* [46]. All of them run on microcontroller platforms and can be used for wireless sensor nodes. However, we decided to use *TinyOS* [47] since it has been used in several sensor node projects at the TIK laboratory. Such projects include the *Permasense* nodes used for permafrost monitoring [48], the *SpiderBat* project [49], and the *BTnode* platform [50], where TinyOS can be used as an alternative to the BTnut OS.

TinyOS is used in research projects around the world. It is still under active development, although releases do not occur at regular intervals: At the time of writing, the most recent release (2.1.1) is more than 18 months old. Therefore we decided to work with the latest development version from the project's source code repository.

6.4 Microcontroller Family

TinyOS can be used on two major families of microcontrollers: The Atmel ATmega series, and the Texas Instruments MSP430. The question was which of these platforms would be better suited for our prototype hardware. It would be feasible to change the platform in the future, but not all of TinyOS's features are available through a hardware abstraction layer, i.e. some changes in the source code may be necessary.

6.4.1 Feature Comparison

We compared two popular models for which TinyOS support is available: The *Atmel ATmega1281* [51], and the *TI MSP430F1611* [52]. Both MCUs have 64 pins and are available in a TQFP package, i.e. the space requirements and the number of available I/O pins are similar. The MSP430F1611 is more expensive than the ATmega1281, but when the exact requirements for flash memory and pins are known, a cheaper version can be used: There is an unofficial version of TinyOS that supports many other MSP430 chips [53]. Cheaper Atmel AVR controllers are not supported at the time of writing. A major difference between the two architectures is the supply voltage: The MSP430 has a clock frequency of 8 MHz and a voltage range of 1.8 to 3.6 V, with an absolute maximum rating of 4.1 V. The ATmega is available in two versions, one with 8 MHz clock frequency and 1.8 to 5.5 V power supply, and one with 16 MHz clock frequency that requires at least 2.7 V supply. So the ATmega is more flexible with respect to the supply voltage. On the other hand, the MSP430 has a more flexible clock management: It supports up to two external crystals and an external clock, so that a separate clock source can be used e.g. for timers. Furthermore, the clock settings can be changed at runtime, while they have to be adjusted by setting *fuses* in ATmega controllers. Both MCUs have a very low power consumption: The MSP430 uses less power in Active Mode (330 μA vs. 500 μA), while the ATmega is more economic in power-down mode (0.1 μA vs. 0.2 μA). Also, both MCU architectures provide several sleep modes between idle and full power-down. An analog-digital converter (ADC) is present in both controllers, but the one in the MSP430 controller provides a better resolution (12 bits vs. 10 bits in ATmega). Since we use the ADC only for extra features such as supply voltage surveillance, this is not an important advantage. A special feature of the ATmega is the EEPROM, which allows for storing settings in a non-volatile memory at runtime. On the MSP430, such settings may be stored in the flash memory, but since the lifetime of flash cells is limited to around 10'000 write cycles, they should not be updated too often.

6.4.2 Practical Differences

We were also interested in practical questions, such as whether suitable developer tools are available. We knew that there is a complete Free Software toolchain to program ATmega controllers: The AVR version of the GNU C library (*avr-libc*), C Compiler (*avr-gcc*), linker, assembler, and other tools (*avr-binutils*) [54]. Other available software includes a programming tool (*avrdude*, [55]), a simulator (*Simulavr*, [56]) and a JTAG debugger (AVaRICE, [57]). For the MSP430 there is a port of GCC and the GNU C Library as well (*mspgcc*, [58]), there is a programming and debugging tool called *MSPDebug* [59], and a simulator (*msp430simu*) is apparently under development. In summary, there are useful

Free Software development tools available for both MCU families.

In addition to software tools, a *programmer* is needed to transfer a compiled program onto the MCU. Traditionally, programmers and debuggers often used the Parallel Port, which is no longer available on recent computers. However, it turns out that USB programmers are available for both platforms, so this is no longer an issue.

6.4.3 Decision

We decided to use the MSP430F1611, mainly because it provides the option to use a cheaper MCU for the final product, but also because the SPORTident control stations use the very similar MSP430F149. We later realized that this decision is not optimal: Since a 3.7 V Lithium-ion battery is used as a power supply, the ATmega1281 with its wide supply voltage range would be better suited for the application. The effective voltage of a charged battery is around 4.1 V, i.e. the MSP430F1611 is being operated at its absolute maximum rating, which may reduce its lifetime. The SPORTident control stations do not have this problem: They use a 3.6 V Lithium battery whose effective voltage is very close to the nominal voltage. A possible solution is to use a DC/DC converter such as the *TPS61221* from Texas Instruments [60], which converts the battery voltage to a stable supply voltage for the MCU. However, it seems more reasonable to switch to the ATmega1281 in the next hardware revision.

6.5 Power Supply

The requirements for size and reliability of the radio nodes leave no other option than using a battery. The basic question is whether to use a non-rechargeable battery that will last for at least several weeks, or whether to work with a rechargeable battery that must be charged every time the equipment is used.

The SPORTident control stations are powered by 3.6 V Lithium batteries. They use 1000 mAh cells in the small stations and 2000 mAh in the large ones. At a transmit power of 500 mW, the RFM12BP radio module dissipates about 3 Watts. If we assume a duty cycle of 10%, a 2000 mAh battery would last about 25 hours, which corresponds to only 3-4 events. We therefore conclude that the use of non-rechargeable batteries is not an option: They would have to be changed too frequently.

In terms of energy density, the best choices available are Lithium-Ion and Lithium-Polymer batteries. The cheapest such batteries are those used in com-

mercial handheld devices, such as cellular phones. We compared some common battery types and searched compatible models from Chinese manufacurers. The resulting table is given in Appendix A. Finally, we chose the *BP-4L* battery from Nokia: It has a capacity of 1500 mAh and several manufacturers provide affordable replacements for it, including *Pisen* which is one of the more reputable Chinese battery brands.

One problem that remains to be addressed is how to provide the 12 V supply for the RFM12BP module's power amplifier. This can be achieved e.g. by using a DC/DC converter. Details on the implementation of this circuit are given in Section 7.3.2.

Chapter 7

Hardware Implementation

7.1 Toolchain

An important tool during the practical work was *DokuWiki*: It was used as a collection of notes, web links, and documents. For instance, the data sheets of all the components used in the hardware design were stored in the wiki. References to websites or articles were added directly in BibTeX format. The wiki proved to be a good replacement for paper notes, text files, and even simple spreadsheets. For the PCB design, the free software suite *KiCAD* [61] was used. It provides all the functionality needed to draw circuit schematics and implement them as printed circuit boards. The Windows-based tool *GC-PREVUE* was used to verify the design before submitting it for manufacturing.

7.2 Hardware Revision 0

The first prototype is based on the MSP430-P1611 evaluation board [62] from Olimex. We call this the "revision 0" of our hardware since it has not been developed specifically for our application. The board contains an MSP430F1611 microcontroller from Texas Instruments and provides basic functionality such as a 3.3 V Linear Regulator, RS232 connector and JTAG interface. Its prototyping area with a 2.54 mm pitch leaves ample space for custom components. Pin headers are provided to access the controller's I/O pins.

7.2.1 Assembly

Evaluation Boards

Two evaluation boards were used: An MSP430F1611 and an MSP430F169, with a slightly different microcontroller variant. The first enhancement to the boards was to add **status LEDs**: A green, a yellow and a red LED were soldered

onto the board, including resistors. Pin headers were added to allow for flexible wiring of the LEDs to the microcontroller pins. Then, the radio module was added. Since the contacts of the RFM12BP module have a 2 mm pitch, but the board's prototyping area has a 2.54 mm grid of holes, the mounting pins had to be long enough to bend them accordingly. We soldered only pin sockets to the boards and plugged the long pins into them, so that the radio module was removable. Then a row of pin headers was added next to the module's pins to wire them to the microcontroller pins. Additionally, an 8 MHz crystal was connected to the controller's XT2 pins using the extra socket on the evaluation board. A watch crystal is present on the board by default, and a faster clock may also be generated using the controller's internal oscillator, but the extra crystal ensures that the fast clock is accurate as well.

Antennas

For the first prototype, an antenna with an attached cable was used. We used the ANT-433-PW-QW from Antenna Factor. This is a quarter-wave monopole antenna, i.e. it needs a proper ground plane to achieve full performance. Since the evaluation boards do not provide an ideal ground plane due to the grid of holes on more than half of its area, we decided to mount the antenna on a separate board. A copper-covered FR-4 board was used for this purpose. The ground plane was cut to 10×10 cm, which is the size the antenna is optimized for according to the datasheet [63]. Then the antenna was mounted in the center of the ground plane. The evaluation board's ground plane is connected to the antenna as well, but this connection is very indirect: The antenna's coax cable is soldered to the radio module, which is connected to ground only through jump wires attached to pins at its socket. We therefore ignored this ground plane.

Power Supply

The evaluation boards contain an *LM1117* Linear Regulator to create a stable 3.3 V supply voltage. This regulator accepts input voltages up to 15 V [64]. We therefore decided to use a large 12 V battery to power the nodes, so that we can directly provide the supply voltage for the radio module's power amplifier. A laptop battery was used for one prototype and two 6-Volt lead-acid batteries for the other. Originally we intended to use more compact batteries, particularly 3.7 V Lithium-Ion batteries, in order to test the DC/DC converter necessary to generate a stable 12 V supply from such batteries. However we did not achieve the required stability when assembling a DC/DC converter on the prototyping board. Therefore we decided to implement the converter directly in the second prototype.

Setup

To make the radio nodes easier to carry, each of them was mounted on a plastic tray. A board, antenna board, programmer, and USB to RS232 adapter was attached to each tray using screws and cable ties. The batteries were attached to the trays as well. A complete radio node is shown in Fig. 7.1.

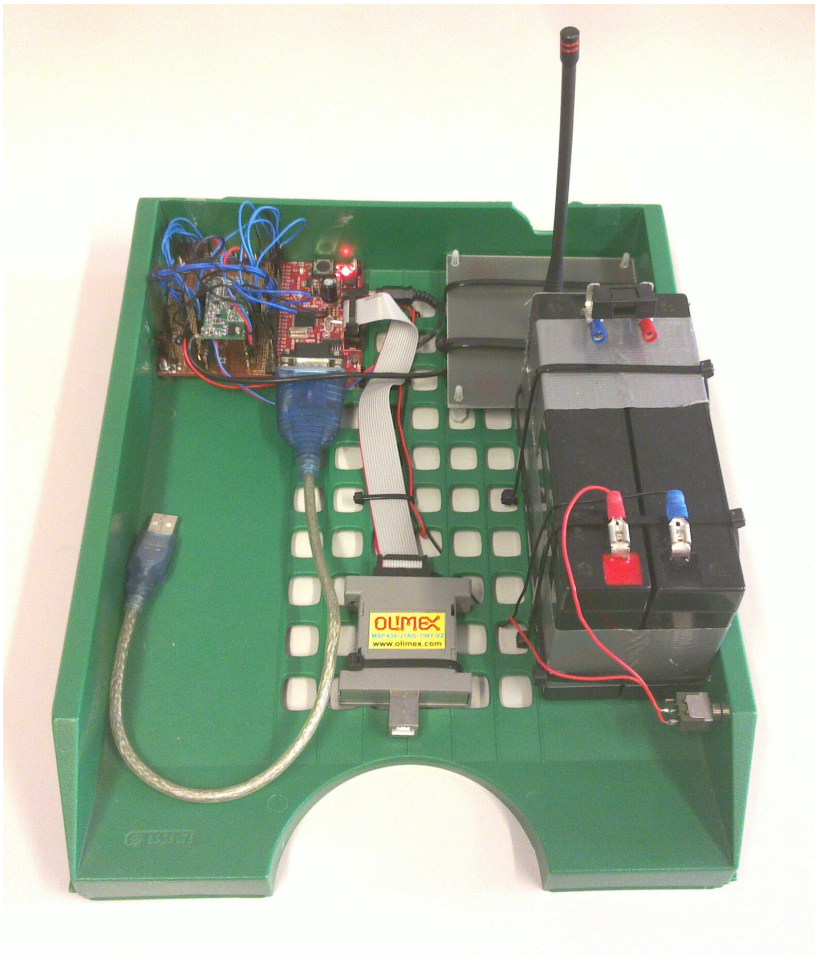


Figure 7.1: One of the two nodes of hardware revision 0.

7.2.2 Testing

The evaluation board was first tested using simple TinyOS applications. After creating a TinyOS platform called *msp430p1611* and assigning functions to the I/O pins, we were able to test basic functionality such as the serial port and the *Leds* interface. For testing the radio module we used a normal C program first,

mainly because such a program was already available and only had to be adapted to our pin configuration. The serial port was often used to output debugging information at runtime, or to log data during the range tests, which are discussed in Section 9.1.1.

7.2.3 Conclusions

From the experience with this first testing platform, we concluded that the selected MSP430 controller was suitable for our application and that a similar circuit could be used in the next hardware revision. We were not happy with the range of the RFM12BP radio module, but hoped that it might perform slightly better on a "cleaner" hardware without jump wires. Up to that point, we had not found any other radio module with a price below 100 USD or so, therefore we decided to use the same module in hardware rev. 1.

7.3 Hardware Revision 1

After basic radio functionality had been established on the revision 0 nodes, we designed revision 1 of our hardware: A more customized and compact radio node. The goal was for this design to be compact enough to be used in orienteering competitions, to be waterproof, and to allow for testing routing protocols. For such tests to make sense, more than two nodes are necessary; it was therefore decided to build five pieces of this design.

7.3.1 Case

The first design decision was which case would be used for the nodes. The case had to be waterproof, ideally complying with the IP67 standard [65], but it also had to be affordable since future versions of the design might be manufactured in larger quantities. After doing a comprehensive search of component suppliers, the choice was narrowed down to three cases, as shown in 7.1. Finally, the SPORTident case was chosen, since it has been thoroughly tested in orienteering and is very rugged. It is shown in Fig. 7.2. Furthermore, it is transparent, i.e. LEDs may be placed anywhere on the device to be visible from the outside. Last but not least, the case is affordable, and SPORTident GmbH agreed to provide five cases for the prototypes free of charge.

7.3.2 Circuit Design

The circuit was improved and enhanced with respect to the first prototype. The same microcontroller, MSP430F1611, was used. It may be possible to use a

Manufacturer	Dimensions	Ingress Protection [65]
Hammond	26×55×80 mm	IP66
Bulgin	53×72×80 mm	IP67
SPORTident	125×80×31 mm	IP67

Table 7.1: Three affordable cases that could be used for sensor nodes.

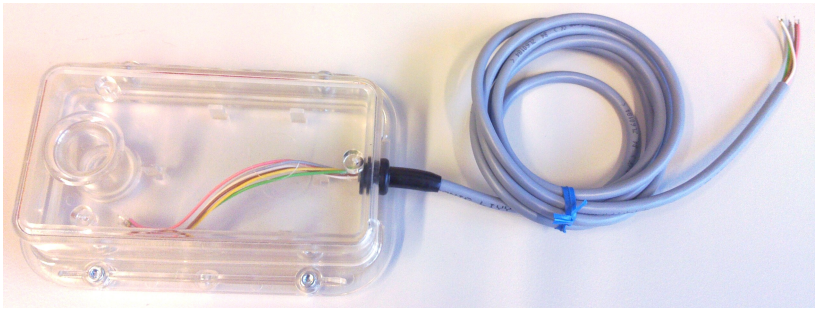


Figure 7.2: SPORTident BSF7 case with built-in RS232 cable.

cheaper controller in a later version of the design, but it was important to have enough flash memory and RAM while the firmware was still under development. The basic circuitry around the controller is shown in Fig. 7.3. It was left unchanged with respect to the first prototype.

Crystals

As in the first prototype, two crystals were added to the design: A watch crystal at 32.768 kHz, and an 8 MHz crystal for the fast clock. We were not sure yet whether they would both be required for reliable operation, so it was important to at least have the option of using them. To provide a stable clock signal, the load capacitors were calculated according to an application note by Texas Instruments. [66]

Battery Holder and Contacts

Since our design uses a non-standard battery, there are no holders for it available on the market. We therefore decided to construct a holder from pin sockets. The holder consists of three corners, with seven pins each. It was planned to better attach the battery using wires between the sockets, therefore one pin was left unconnected in each corner. However it turns out that the pin sockets create enough tension on the battery to hold it safely. The battery contacts were realized using special metal pins with flat ends from the lab's stock. We cut them into the appropriate shape with a pair of pliers.

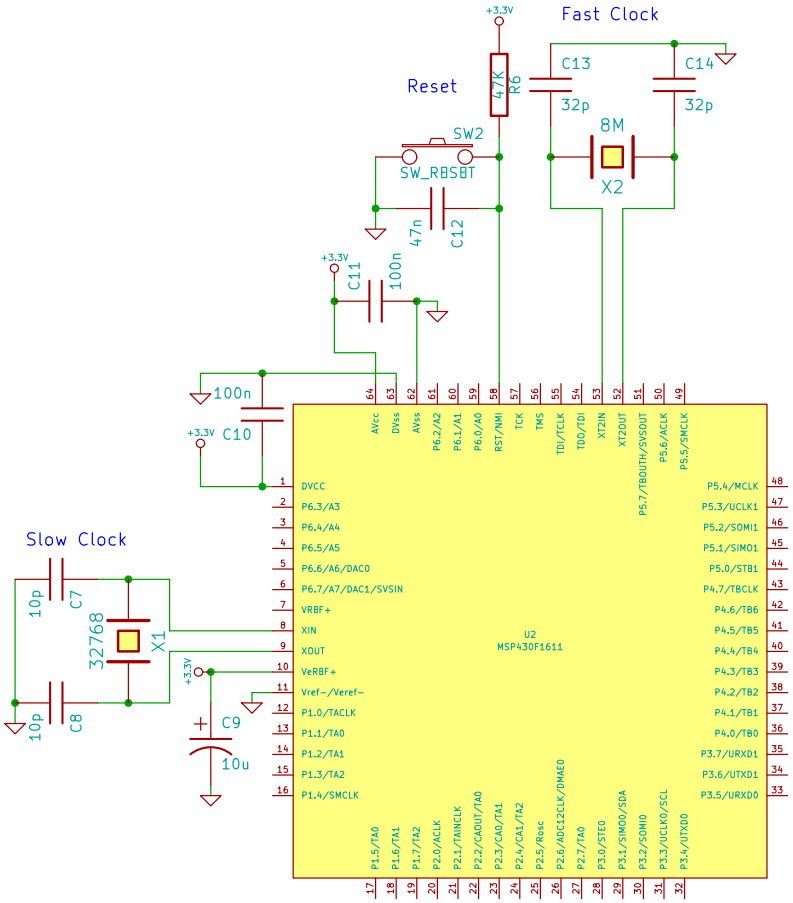


Figure 7.3: MSP430F1611 microcontroller with power supply, reset and two crystals.

Step-up Converter

The RFM12BP radio module requires a second supply voltage of 12 V for its power amplifier. To provide this voltage, a *TPS61085* step-up converter from Texas Instruments is used. The circuitry around it is according to its datasheet [67], with a *PMEG2010AEH* Schottky diode, several capacitors and resistors, and a 6.8 μH inductor. According to the datasheet, this circuit can supply up to 600 mA at 12 V, which is more than enough for the radio module: In transmit mode, it draws 200 mA according to the datasheet [68]. To check the stability of the second supply voltage during operation, the voltage can be measured by the microcontroller on one of the ADC pins. A 1:10 voltage divider is used. To check whether the step-up converter actually worked, we tried building it on a breadboard. Unfortunately, we could not find a suitable inductor for it. Several experiments with different inductors produced the desired voltage, but it did not remain stable when a current was drawn. While designing our PCB, we finally found the right type of inductor at *Reichelt Elektronik*. However, we did not have time to test the circuit with it before ordering the PCB. Luckily the circuit was correct and worked flawlessly.

Radio Module

The radio module itself is connected to the microcontroller in the same way as determined in the first prototype. All I/O pins are connected, although not all of them are necessary to operate the module. This allows for testing special features of the module, such as using its clock output as a clock source for the microcontroller. Unfortunately, the radio module does not provide its analog RSSI signal through a pin. Therefore it is contacted directly on the module using a wire, and connected to one of the ADC inputs of the microcontroller. This is the only way of getting more than one bit of signal strength information from this module.

UART interfaces and GPS module

The microcontroller has two UART interfaces. One of them is connected to a *MAX3232* level shifter, which converts the logic levels of the signal to RS232-compatible levels. This interface is intended to communicate with the SPORTident control station, which sends out a data record whenever a runner punches the control with an RFID chip. However RS232 is also very useful for debugging purposes. The second UART interface is connected to an optional GPS module. We used the GPS-ST22 module [69], which is distributed by Perthold Engineering LLC. This module has an integrated antenna, is easy to use and is cheaper than modules from other vendors. It transmits location and time data over UART. The GPS also has an output called “1PPS” (One Pulse Per Second) which is

connected to an interrupt pin of the microcontroller and may be used as an exact time source if the GPS has a good signal. Since the main purpose of the GPS is to provide an accurate time reference, it is sufficient to have one node with a GPS in the network (e.g. the base station). The time can then be synchronized between the nodes before they are placed in the terrain. Since the relevant clock for orienteering races is that of the SPORTident control stations, there should also be a time synchronization between the radio node and the SPORTident station. This can be implemented using a special command of the SPORTident station, see Section 8.5.

Accelerometer

As an experimental feature, an accelerometer has been added to the circuit. It has three analog output signals, which depend on the acceleration in X, Y and Z direction, respectively. These are connected to 12-bit analog/digital converter inputs of the microcontroller. The remaining I/O pins of the accelerometer are digital and are connected to other pins of the controller.

Bill of Materials

We used several suppliers of electronic components to obtain the parts for the second prototype. *Mouser Electronics* was selected to be the main supplier, i.e. all components which were available were bought there. The reason for this choice is that Mouser has a very large selection of components and generally offers lower prices than suppliers shipping from Switzerland, such as Distrelec or Farnell. The bill of materials for the PCB is given in Appendix B.2.

7.3.3 Printed Circuit Board

Design

The Printed Circuit Board (PCB) of the radio node was designed to fit into the SPORTident case. The case has three screw holes, but SPORTident only uses two of them in recent products: The third screw hole is very close to the hole through which the RS232 cable enters the case. The rubber protector that makes this connection waterproof partly covers the hole. A gap is left in the PCB to allow room for this protector and the cable end. Another reason to use only two screws is that tensions in the PCB are avoided when large forces are applied to the case. Since we also use an RS232 cable, we decided to use the same PCB shape as the SPORTident control stations. The cases also have a large hole where the RFID chips are usually inserted, so we had to leave this clear in our PCB as well, even though the hole is not used in our design. We marked all the

holes and clear areas on the silkscreen layer of the PCB, on both sides. A sketch with the board's dimensions is shown in Fig. 7.4.

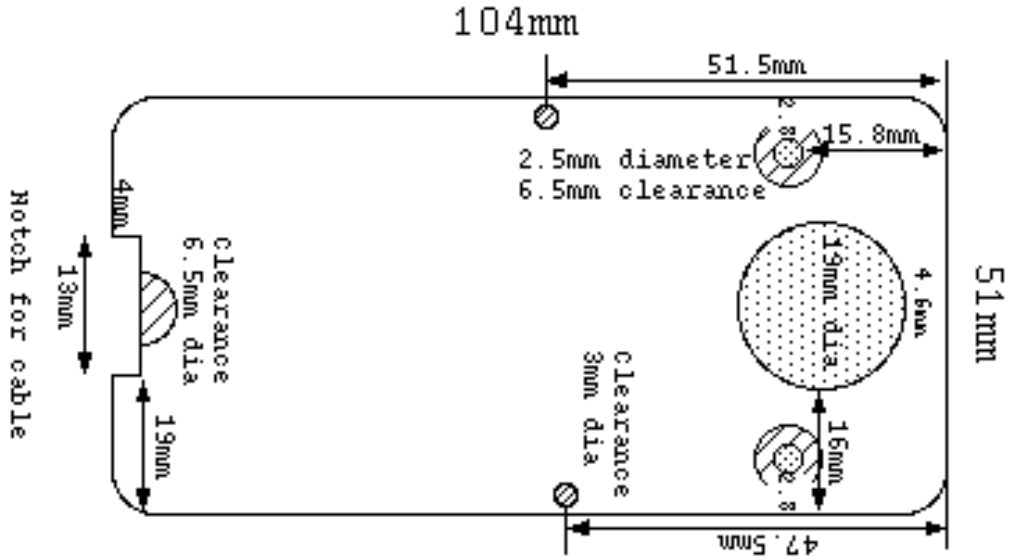


Figure 7.4: Sketch of a PCB layout that fits into the SPORTident BSF7 case. Not drawn to scale.

Placement and Routing

The parts were placed on both sides of the board. This was the only option, due to restrictions of the design: First of all, the battery had to be on the bottom side to be removable. The SMA connector and push buttons were placed on the bottom as well to make them accessible to the user. Since the battery took up most of the PCB's area, there was no space left for the microcontroller and most of the remaining components. We decided to put the DC/DC converter on the bottom layer as well, in order to separate this potentially noise-generating circuit a bit from the rest, and to make sure that there is enough vertical space for the inductor. Also the LEDs were placed at the bottom since it makes sense to have the entire user interface on one side (i.e., to place the LEDs close to the buttons). The remaining components were all placed on the top layer. We started by placing the radio module and the microcontroller near the center of the board, and added the level shifter, crystals, and accelerometer on different sides of the controller.

When we had placed all the parts, it became obvious that it would be hard or impossible to route the PCB on two layers. Even though we tried to put each component near the microcontroller pins it would be connected to, the space for

tracks was scarce near the controller. We therefore decided to make a four-layer design: One of the inner layers served as a ground plane, and the other one was used as an additional routing layer. In this way we were able to avoid dividing the ground plane with tracks, except for the vias, which are present on all layers. It was rather challenging to place all the resistors for the eight LEDs, as well as the capacitors for the accelerometer. In the end, however, we accomodated all the little parts and connections. Special attention was paid to the RF track: There was only one such track between the radio module and the SMA connector. We calculated its width so as to create a textitmicrostrip line (see Section 4.3.2).

Manufacturing

The PCB design was sent to *Beta Layout GmbH* for manufacturing. An advantage of this supplier was that a free stencil for the application of solder paste was shipped with the manufactured boards. Before submitting the design, we exported it from KiCAD in Gerber format and imported the resulting files in a tool called *GC-PREVUE*. This Windows-based software was supplied by the PCB manufacturer to preview the design files and save them in one common file. Since the Gerber format has separate files for tool definitions and the actual layout data, this step is useful to ensure that the correct tool definitions are used. GC-PREVUE further allowed us to adapt the layer numbers in the layout data, so that they matched the manufacturer's layer stack definition [70]. After submitting the file, the PCBs were manufactured without further delays or requests from the manufacturer. We received the boards after a processing time of eight working days. One of them is shown in Fig. 7.5 and 7.6.

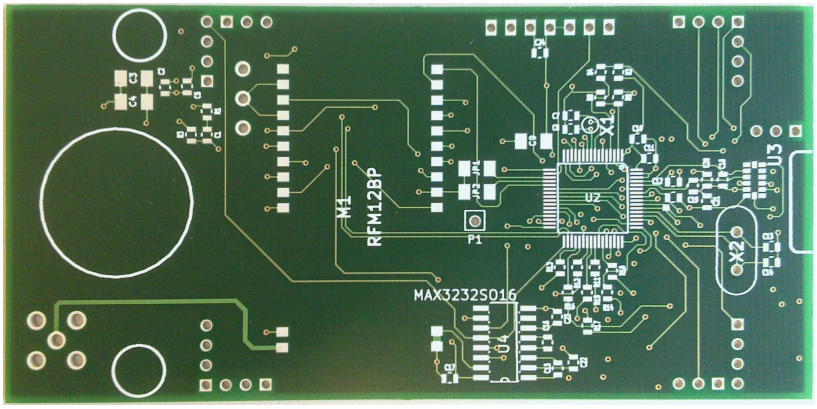


Figure 7.5: Top side of the PCB for our second prototype.

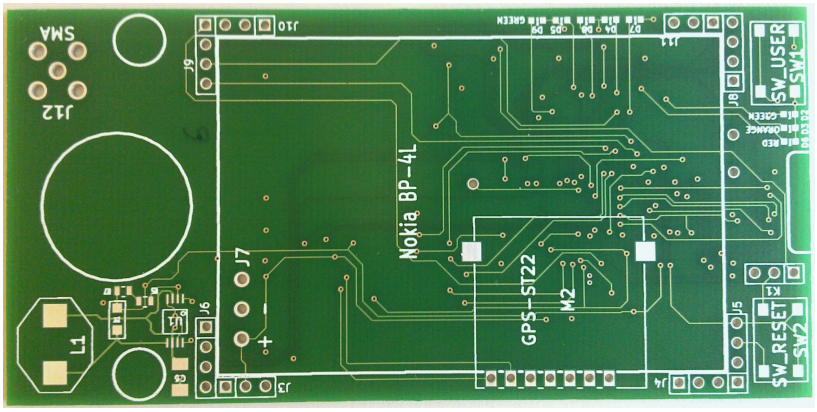


Figure 7.6: Bottom side of the PCB.

7.3.4 Assembly

After we had received the PCBs and all required components, we assembled the five radio nodes. First we had to prepare the PCBs: The larger holes and the gap at one side were drawn on the silkscreen, but not drilled by the factory. We made these drills ourselves since the tolerance was large enough to do them by hand, and the factory would have charged extra for doing them directly. Then, to solder the top layer of the PCBs, we decided to use a soldering oven. The main reason for this decision was the Freescale accelerometer, whose pads are underneath the part and very hard to solder by hand. The Integrated Systems Laboratory of ETH Zurich allowed us to use its vapour phase soldering oven, an *IBL SV260*. Such ovens have a more homogeneous temperature distribution than simple reflow soldering ovens since they use a special medium that evaporates during the soldering process. The first step was to place all components on the PCB's top side using solder paste. We omitted the RFM12BP radio module since it might have been damaged during the soldering process. The solder paste for the remaining components was spread on the board using the metal stencil that came with the PCBs. The reflow soldering worked quite well, except that the stencil had slightly oversized openings for the MSP430 microcontroller. Too much solder paste was placed on the controller's pads, and some of the pads were soldered together. This had to be corrected by hand using solder wick and a microscope. Then the remaining parts, including all parts on the bottom side of the board, were soldered by hand. The biggest difficulty in this was to solder the SMD inductors, whose pads we designed slightly too small. Creating useful battery contacts was also challenging: The holes for the contacts turned out to be almost a millimeter too far away from the battery's border. We made specially shaped contacts from small metal parts and managed to achieve a reliable electric contact. A completely assembled PCB is shown in Fig. 7.7.

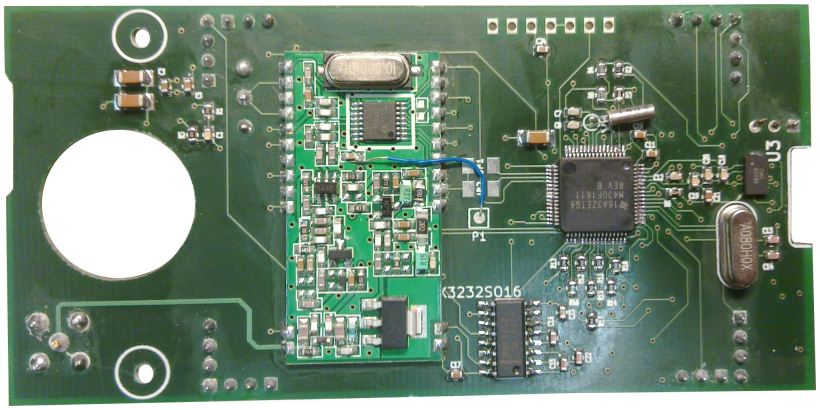


Figure 7.7: Top side of an assembled PCB.

Finally, the PCBs had to be mounted in the cases. We started by drilling a hole for the SMA connector in each case, so that the antenna could be mounted on the outside. This hole had to be made slightly larger than the plug, so that the plug could be inserted diagonally, with the PCB already inserted below it. The RP-SMA antenna plug was crimped to a piece of RG-174 coaxial cable with an SMA connector on its other end. This connector was then attached to the one on the PCB. The PCB was mounted to the case using two screws. One corner of the third screw holder in the case had to be milled off because the accelerometer would have touched it. We tried to avoid this by keeping the screw holes clear of parts, as shown in Fig. 7.4, but we found no other way of accommodating the accelerometer. To complete the assembly, the three wires of the RS232 cable were soldered to the PCB. The cases came with female RS232 connectors, but we replaced these with male ones and exchange the RxD and TxD wires. This way, the radio node acts as the host, and a SPORTident control station can be connected to it directly. On the other hand, a null modem adapter and a gender changer is now necessary to connect the node directly to a PC.

7.3.5 Testing

Basic Functionality

The basic functionality was tested using simple TinyOS applications. To test the on-board step-up converter, we measured the voltage at the 12 V supply pins in the lower right corner of the battery holder. We added a load to test the voltage stability. On all nodes, the DC/DC converter was able to supply 300 mA at 12 V. The UART interfaces were tested with the *UartForwarder* application

that allowed to output received data from UART0 on UART1, and vice versa. With this application and a data source, we were able to test the UART interfaces of several nodes in a chain. Radio functionality was tested with both the TinyOS radio driver and the simple test program written in C.

Radio Performance

We measured the performance of several parts of our radio system. First of all, the output power of the radio module was measured after the coaxial cable. We connected the cable to a power meter via a 20 dB attenuator since the power meter had a limited range. The result was rather surprising: We measured an output power of nearly 30 dBm, i.e. 1 W. This is problematic since the output power of 500 mW specified in the module's datasheet is the maximum permitted by law. In the 433 MHz SRD band, even if the output power is below 500 mW, channels may only have a bandwidth of 25 kHz. The regulations are explained in detail in Section 4.1. We then also tried to reduce the output power: The radio chip has a configuration command called *TX Configuration Control* that supports eight different settings. However, it turns out that the influence of the setting on the measured output power is small and differs greatly between modules. Our test results with two nodes are shown in Fig. 7.8. Probably the power amplifier compensates for the reduced output power of the chip and amplifies the output signal more, which makes the setting useless. The module thus doesn't have a "Programmable TX power" as claimed in the datasheet [68].

For further measurements we used a spectrum analyzer, again with a 20 dB attenuator to handle the high output power of the device. We averaged the measurements of 10 seconds to determine the RF spectrum at the radio module's output. In Fig. 7.9, the spectrum is shown for a frequency deviation of 90 kHz. This is the setting we used for all other experiments. The distance between the two peaks is exactly 180 kHz, i.e. the deviation is correct. We performed the same measurements for a deviation of 15 kHz (the smallest value that the module supports) with results shown in Fig. 7.10. Here the peaks are exactly 30 kHz apart, i.e. this setting command of the radio chip works flawlessly. The spectrum as such is acceptable as well, except that the peak power is again 30 dBm rather than 27.

Power Analysis

We performed a power analysis using an *Agilent N6705A* analyzer with all five nodes. Before the analysis we had already measured the supply current of each node with a multimeter and discovered differences in some cases. Some of these were due to shorted microcontroller pins that caused undesired currents and were

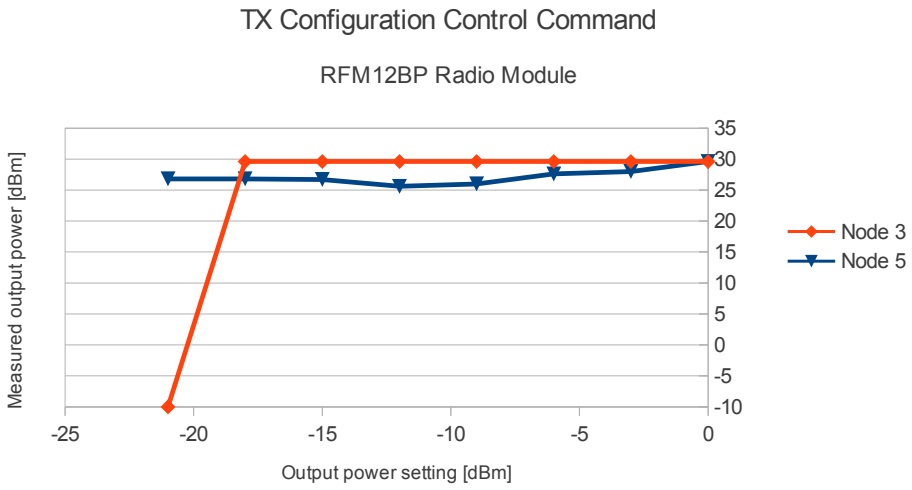


Figure 7.8: Effect of the "TX Configuration Control" command on the radio module's output power.

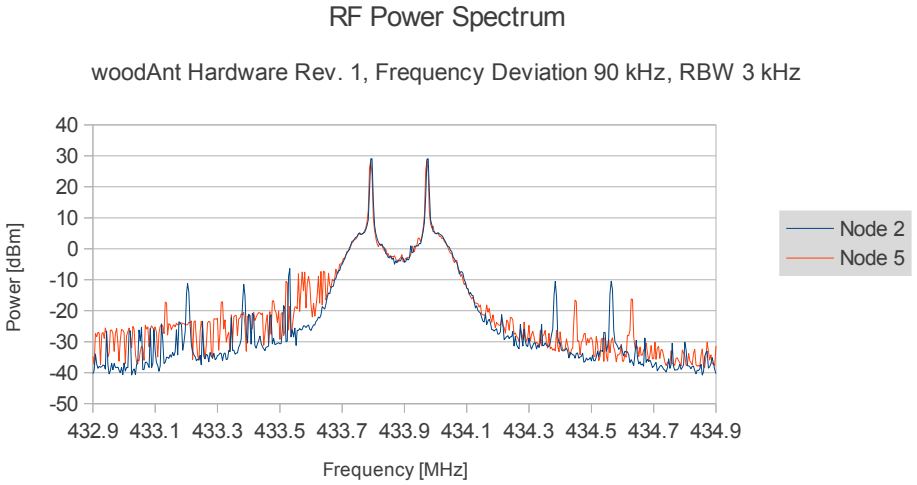


Figure 7.9: The power spectrum, as measured at the SMA plugs of two nodes.

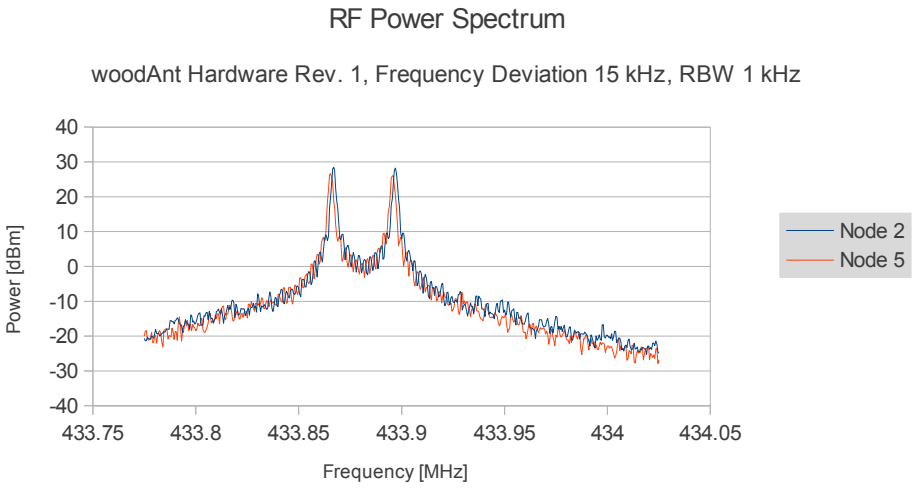


Figure 7.10: Main lobe of the power spectrum with the RFM12BP set to its minimum frequency deviation.

removed. In the analysis, only one node (No. 3) still shows a strange behavior.

First we determined the power consumption in idle state. For this purpose we loaded the *Null* application of TinyOS on each node. This application does nothing, but it still includes some features of the platform it is compiled for. In our case, this means that the fast clock initialization is performed since it is wired to the platform initialization in *PlatformC*. However, optional features such as the radio are not activated. We configured the power analyzer to use a supply voltage of 4 V, which is outside the specifications of the microcontroller, but is realistic since our batteries have 4.1 V when fully charged. The voltage during the idle test is shown in Fig. 7.11. Since it always remained constant after it had reached 4 V, we will only consider the current for the remaining experiments. It turns out that there is a current peak whenever the node is connected to the supply, as shown in 7.12. This is due to the DC/DC converter circuit which uses an inductor. Our measurements show a peak of around 800 mA, but the value depends on the sampling rate used, i.e. it can be assumed that the peak is actually even higher and shorter. A plot of the current after this turn-on peak is given in Fig. 7.13. There we can see a current of 11.5 mA for about 200 ms in most nodes. This is due to two LEDs being flashed during clock initialization. Afterwards the current remains at 3.3 mA in all nodes. This is a high idle current, but considering our intended battery life of eight hours, it is accept-

able. We were unable to determine why the current is so high, but probably the DC/DC converter causes some losses since it cannot be turned off, and the radio module may also draw some current when idle. The microcontroller has an idle consumption of less than 1 μA and uses less than 1 mA even in active mode, i.e. a software error cannot be the cause of the high current.

We also performed an analysis with the radio module in receive mode, with results shown in Fig. 7.14. It turns out that the circuit uses around 16 mA in this mode, except for node No. 3 which appears to have a hardware problem. If we assume that we can use 80% of the capacity of our 1500 mAh battery, our nodes will thus work for 75 hours in receive mode. Surprisingly, there are large differences in the time it takes to enable the radio module over SPI: Node 2 enabled its radio within 100 ms after turning on the node, whereas node 5 took almost a second. The measurements were triggered by turning on the power supply, i.e. they should be synchronous.

Finally we were interested in the power consumption of the radio module in transmit mode. This was tested with an application called *PowerTestTX* that sends a packet every 100 ms. In Fig. 7.15 we can clearly see the send phases. The current in transmit mode varies greatly between nodes, with 550 mA drawn by node No. 4 and 770 mA by node No. 2. Again node 3 should be disregarded since it probably has a hardware problem, but the large differences between the other four modules are not acceptable. We think that the most probable reason for them is a high tolerance in the production of the radio modules. A varying efficiency of the step-up converter could also be the reason, but given its simple layout and stable behaviour, we consider this highly improbable. To continue our estimation of battery life, we assume a duty cycle of 10% for the transmitter, and a current of 770 mA in transmit mode. The battery would then still last 13.1 hours, which is sufficient for our application.

Field Tests

The field tests are discussed in Chapter 9.

7.3.6 Issues

Assembly Problems

A serious issue during the assembly was the SMA connector on the PCB. We had ordered board-mounted connectors where the cable would be attached perpendicular to the board. Unfortunately we had not checked the length of the

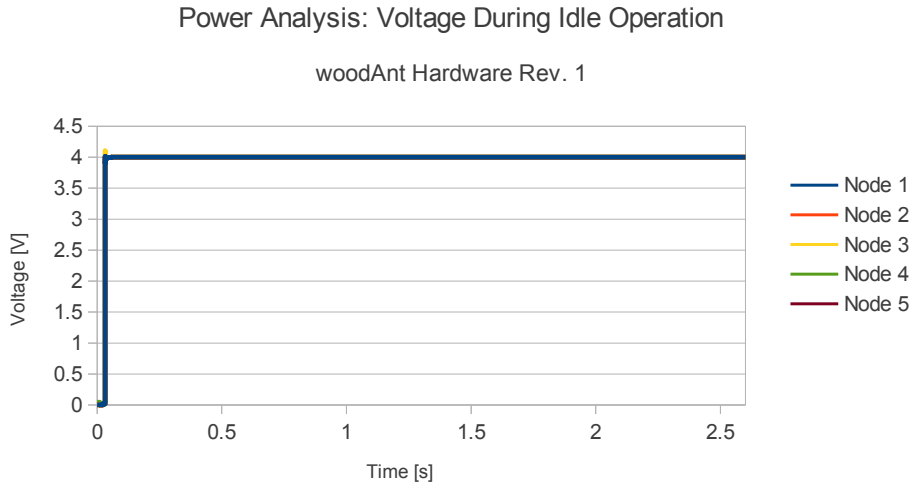


Figure 7.11: Supply voltage while testing our nodes with the Null application.

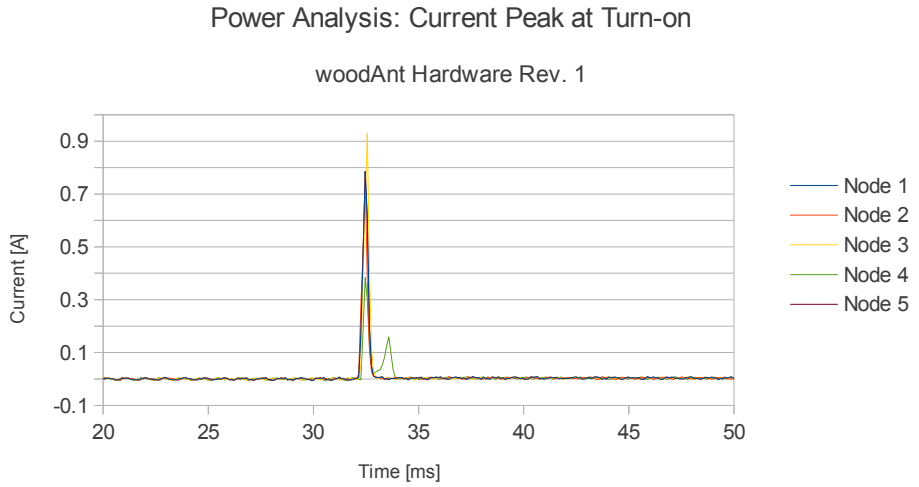


Figure 7.12: A current peak occurs when the radio nodes are turned on.

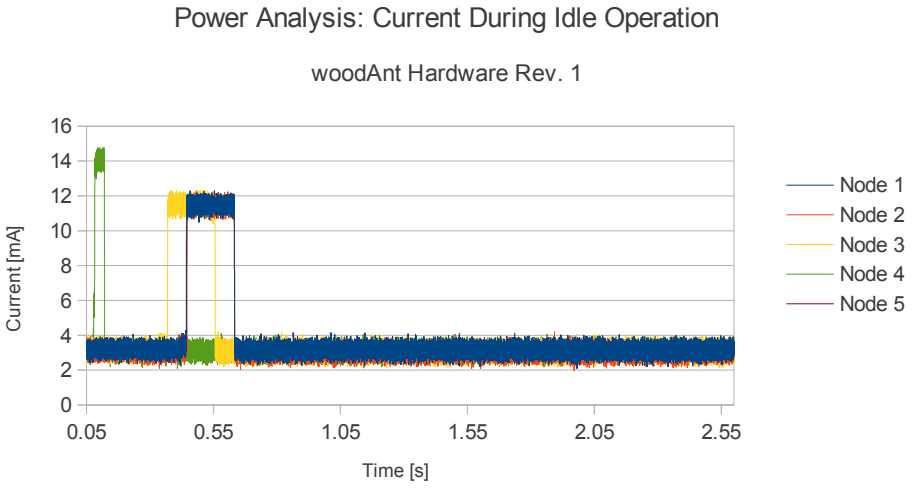


Figure 7.13: Current drawn by the five nodes when running the Null application.

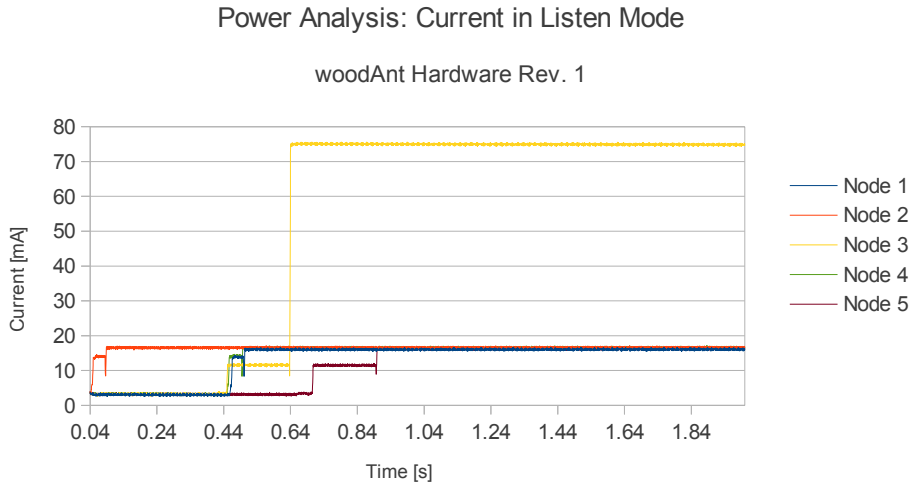


Figure 7.14: Current drawn by the five nodes with the radio in receive mode.

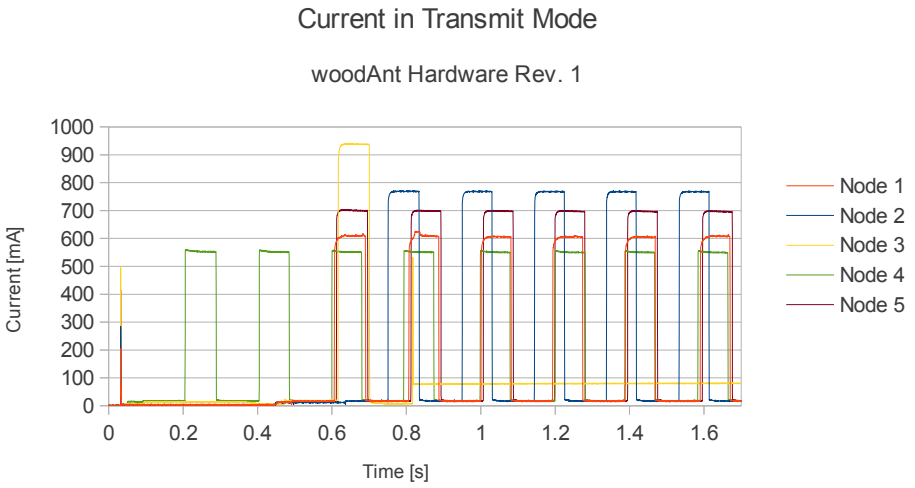


Figure 7.15: Current drawn by the five nodes with the radio in send mode.

crimped plug, and during assembly we noticed that it did not fit into the case. Even if an angular plug was used, it would still be too long. The solution was to order an angular SMA connector with the same footprint. This connector almost covered one of the screw holes in the PCB, but it was the best available workaround to the problem.

The other end of the coaxial cable also caused problems. The crimped cable end with the RP-SMA plug was long, and a position had to be found where it could be screwed into the plastic case. In vertical direction, there was not enough space for the plug above the PCB, so the only feasible solution was to put it *beside* the PCB. This further complicated the assembly and made it necessary to drill a diagonal hole for the RP-SMA plug, but at least it worked. Later in the testing phase, the construction was changed: Since it turned out to be better to mount the antenna vertically, it was plugged into the hole in the SPORTident case. A notch for the coaxial cable was cut into the border of the enclosed hole. To keep the system waterproof, the hole was closed from below. Silicone could be applied around the antenna to complete the water protection.

While assembling the boards, we also noticed that we had ordered the wrong level shifter ICs: The footprint of the MAX3232CPWR was too small for the pads on the PCB. Luckily, we found exactly the IC we needed in the lab's component stock. We had checked the stock before ordering, of course, but only

found these parts when looking for the second time.

Minor Design Flaws

While assembling the radio nodes, we noticed several small design flaws that could be corrected in future versions:

- **Battery Contacts**: The holes for the contacts are almost a millimeter too far away from the battery's border. It would be easier to make reliable contacts if the holes were moved to the correct position.
- **Silkscreen Font Size**: Some of the component numbers on the silkscreen are hardly readable, especially those of the smaller SMT components. These should be enlarged to make the boards easier to assemble.
- **Accelerometer Position**: The accelerometer is currently located directly below one of the three board rests of the SPORTident case. It should be moved to another location to avoid having to mill out a part of this board rest.
- **SMA Connector on PCB**: With the current layout, the SMA connector on the PCB almost covers one of the screw holes. The screw is still accessible, but it would be better to move the connector about a millimeter towards the center of the board, away from the hole.
- **Drill sizes**: Some of the drill sizes are not optimal. The drills for the 32 kHz crystal should be slightly larger, while the battery holder pins and the SMA connector have rather too large holes.
- **RP-SMA Connector**: The connector that attaches to the case is not in an ideal position. It is currently necessary to drill a diagonal hole and insert the connector diagonally, while tilting the PCB. However the options are limited. One would be to replace the large hole in the PCB by a notch, so that the board could be inserted sideways below the already mounted SMA connector. This would mandate a plug or longer wires for the RS232 interface, though. The antenna could also be mounted on top of the case - changing its polarization to vertical - but in this case, a hole in the PCB would be necessary to accommodate the coax cable.

Chapter 8

Software Design

8.1 Toolchain

A GNU/Linux based operating system was used for the entire software design and deployment. The software for the radio nodes was compiled with *mspgcc* [58], a port of the GNU C compiler to the TI MSP430 platform. We worked with version 4.4.5 of the compiler. Unfortunately the mspgcc4 project was discontinued by the developers during the thesis, and future work will have to use the mspgcc3 branch which is still under active development. To program the microcontrollers, the tool *mspdebug* [59] was used. It supports the MSP-FET430UIF programmer from Texas Instruments, as well as the MSP430-JTAG-TINY-V2 from Olimex. Both JTAG debugging and Spy-by-wire are supported. A logic analyzer called "Saleae Logic" was used to make debugging even easier.

8.2 Basic Structure

The software of our radio nodes is based on TinyOS. Programs in TinyOS do not have a classic "Main" function that manages all actions. Instead, it uses a scheduler that calls procedures when certain events occur, or when the microcontroller is idle. Each application consists of several *components*, which are linked together using *interfaces*. Each component specifies the interfaces that it *uses*, and the ones that it *provides*. The components can be divided into two categories: *Configurations*, which provide a clearly defined functionality to their parent components using one or more child components, and *modules*, which are wired to a component and are part of the implementation of its functionality. While *configurations* only wire several interfaces together, *modules* may actually implement them. If they *use* an interface, they may run its *commands*, and they must handle its *events*. On the other hand, if they *provide* an interface, they have to implement all its *commands* and can *signal events*.
Each wireless sensor node has its own *platform* in TinyOS. A platform consists of a set of definition files, platform-specific adaptations of TinyOS source files,

and a *chips* folder with platform-specific parts of hardware drivers. The most important file of a platform is *hardware.h*, where the pin assignment of the microcontroller is defined and each pin's mode and status is initialized. Outside the platform folder, each platform has a make target file that defines commands and parameters to compile and flash applications to a node. This file also contains the model number of the microcontroller that is used by the platform. In revision 0 of our hardware, we therefore had to create two make targets since we had two slightly different evaluation board: One with an MSP430F169, and one with an MSP430F1611. Rather than creating two platform directories, we created a symbolic link to share one directory between two TinyOS platforms.

8.3 Radio Driver

Radio Module Specifications There are several sources of information available about the RFM12BP radio module. The manufacturer *Hope RF* provides a short datasheet [68] and a programming guide [71]. The programming guide lacks important information, such as a description of the sixteen interrupt bits that may be set by the module. The specifications that are available contain some errors, and the example source code it contains reportedly does not work. However, the datasheet of the *RF12B* radio chip used by the module [72] is virtually error-free - which may be due to the fact that it is a plagiate of the *Silicon Labs Si4421* datasheet [73]. There is also a German website [74] that has collected most of the information about the module. With the help of these sources, we were able to understand the initialization commands in source code examples and to adapt them to our needs.

Test Application in C Before writing a radio driver for TinyOS, we tested the functionality of the radio module with a test application in plain C. We found a library for using the RFM12B on an MSP430 [75], so this seemed like the fastest way to get the module working. The RFM12B is a low-power radio module with the same radio chip as the RFM12BP; all we had to change was that the *RXEN* and *TXEN* pins needed to be controlled as well. Some formal changes were necessary to make the code compile on *msp430-gcc*.

After some debugging, we got the radio module to work. We played around with its configuration parameters and improved some settings. Then we also tested event-based communication over SPI, i.e. using the interrupt output rather than periodically polling pins. This also worked quite well with the test application. Unfortunately the test application did not support hardware SPI; instead, it used so-called *bit-banging* where normal I/O pin functionality is used

to transfer bits over SPI.

For a reliable radio link, we need to be able to determine whether the channel is free. The RFM12BP provides a commmand that reads the *Data Quality Detection* bits in order to

Driver for the TinyOS Radio Stack The largest software development effort involved was to write a driver for the RFM12BP radio module. We used the driver for a different radio chip, the RF212, as a basis for this driver. It implements communication between the MCU and the radio module using hardware SPI. The communication scheme is event based: After the radio module has been initialized and configured, the receiver can be enabled. The module will then pull the interrupt pin (nIRQ) low whenever it detects the synchronization pattern of an arriving packet. The radio driver captures these interrupts and stores the data in a buffer. For each additional byte of data that arrives, the interrupt is enabled again and the byte is transferred over SPI. A similar procedure is used to send data: After the transmitter has been activated, the radio module sets an interrupt when it is ready to receive the first data byte. The byte can then be sent using an SPI command. When the packet is complete, an extra byte ("post-amble") has to be sent to ensure correct transmission. Then the transmitter can simply be turned off.

Internally, the radio driver uses nine states. It starts in *STATE_P_ON* and enters *STATE_SLEEP* after initializing the radio module. When it is woken up, it enters the transitional state *STATE_SLEEP_2_TRX_OFF* and sets a timer to wait until the module is ready. Then it enters *STATE_TRX_OFF*, from which the receive mode *STATE_RX_ON* can be synchronously activated. There is also a transitional state called *STATE_SLEEP_2_RX_ON* to enter receive mode directly from sleep mode. When the first byte of a packet arrives, the driver enters *STATE_DOWNLOAD* and returns to *STATE_RX_ON* after receiving the packet. Before transmitting a packet, a Clear Channel Assessment (CCA) is performed using the radio's analog RSSI output. During this check the driver is in *STATE_CCA*. Then the transmitter is turned on and *STATE_TX_ON* is entered while the packet's bytes are transferred to the radio module. When the transfer is complete, the driver returns to *STATE_RX_ON*. A state diagram is shown in Fig. 8.1.

On top of the radio driver, TinyOS is running an entire stack of network layers. It supports software-based acknowledgements, CSMA, packet management, time stamping, and more.

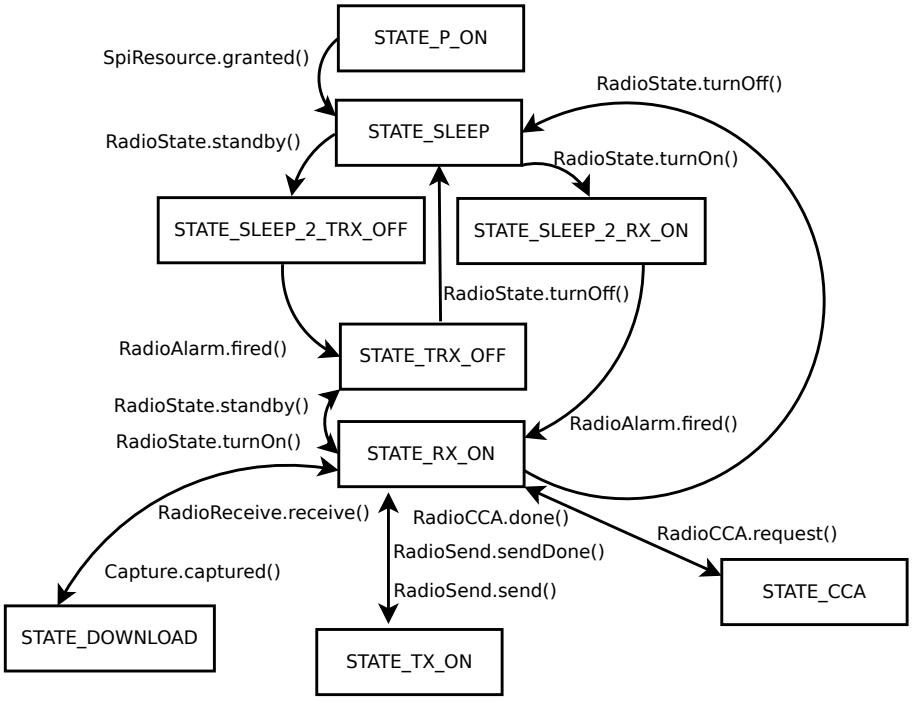


Figure 8.1: State diagram of our TinyOS radio driver.

8.4 User interface

We wrote a configuration that provides access to the user interface of our hardware: The three status LEDs, the five signal strength LEDs, and the user button. TinyOS already provides the *Leds* interface, which allows access to up to eight LEDs if its platform-specific configuration, *PlatformLedsC*, is set accordingly. However, by default only three LEDs can be directly accessed using commands like *led0On()* or *led3Toggle()*. We therefore created a platform-specific version of the *Leds* interface that is compatible with the basic version, but provides extra functions. Furthermore we wrote a *Button* interface, and a component that provides it, with *push()* and *release()* events. On top of this hardware abstraction, we added the *woodAntGuiC* component that provides the *woodAntGui* interface. It uses a timer to provide events for three different push durations of the button, and it can assign six different states to LEDs, such as flashing at different speeds. This maximizes the amount of information that can be displayed on the eight LEDs. As a usage example of the interface, the *woodAntMode* component has been created. It uses a "very long" button press (currently defined as four seconds) to switch between several display modes. A battery state display and a clock is integrated in the platform, and the "normal mode" can be used by the application. The *woodAntMode* interface that provides access to this mode has been very useful for debugging, since it makes it easy to display e.g. the routing table on the status LEDs.

8.5 SPORTident Control Station

The SPORTident Control Stations have their own communication protocol. It is based on a simple RS232 link, with no hardware flow control or other special features. Two baud rates are supported by current models: 4800 baud and 38400 baud. The baud rate can be changed through the communication protocol, i.e. the radio node must first detect the current baud rate when a control station is connected to it. The communication protocol uses control characters to frame the packets: Each packet starts with *STX* (0x02) and ends with *ETX* (0x03). Some commands are acknowledged by *ACK* (0x06), and *NAK* (0x15) is sent in case of transmission errors. To escape control characters, a character called *DLE* (0x10) is sent before each such character that occurs within the data portion of a packet. The protocol has two basic operation modes: *Autosend Mode* and *Normal Mode*. Additionally, the behaviour of SPORTident stations depends on the function for which they are programmed: Read-out stations are intended for reading the entire chip, whereas controls and start/finish stations only write one record to the chip and optionally transmit that record over the serial interface. The case we are interested in is a control in Autosend Mode.

The communication protocol is documented in detail in the *SPORTident Programmer's Guide*, which is made available to developers only. Two commands are relevant to our radio nodes:

Time synchronization The SPORTident control stations have an accurate real time clock that is used to generate a split time whenever a runner punches the station. It is desirable to be able to synchronize the real time clocks of all control stations in the network over the radio system. In order to achieve this, a time synchronization between the radio nodes themselves must be available. As a reference time, either the RTC of a SPORTident station can be used (e.g. the one at the base station), or the time signal from the optional GPS module on our hardware. The only missing link is then a time synchronization from a SPORTident station to a radio node, and vice versa. The communication protocol provides two commands for this, as shown in Table 8.1. These work only in BSF7/BSF8 control stations or newer; for older control stations, the legacy commands 0x76 and 0x77 have to be used.

Punching Record Transfer When a runner punches the SPORTident station, the number of his RFID chip and the punching time must be transferred to the radio node. This is done by enabling the *autosend mode* of the SPORTident station. Either this is done by the radio node, or the user has to do it beforehand using SPORTident's *SI-Config* software. In this mode, whenever the station is punched, a record is sent over the serial port. Its data structure is shown in Table 8.2. The table also shows a command to read a record from the station's backup memory: This can be used if a transmission error occurs (e.g., the CRC sum is wrong), and the record must be sent again. It could also be useful to allow the base-station to request older data packets that it did not receive - after all, our hardware currently does not have enough memory to save all packets if a radio connection is interrupted for several minutes.

8.6 Routing Protocol

Since we did not have enough time to develop a custom routing protocol, we performed some tests using the *Collection Tree Protocol (CTP)*. CTP is a best effort protocol, i.e. it is not guaranteed to deliver 100% of transmitted packets. This is not ideal for our application: A protocol that detects all transmission errors would be better. Such a detection could also be added above CTP, though: The base station could check the order of sequence numbers and request the re-delivery of packets through a dissemination protocol if some are missing. The routing gradient of CTP is *ETX*, the expected number of transmissions to deliver a packet to the base station over the given route. ETX values are additive: The ETX of a node is the ETX of its parent plus the ETX of the link to the parent. In

Command	Syntax	Response Syntax
Set Time 0xF6	STX, 0xF6, LEN, CD2, CD1, CD0, TD, TH, TL, TSS, CRC1, CRC0, ETX	STX, 0xF6, LEN, CN1, CN0, CD2, CD1, CD0, TD, TH, TL, TSS, CRC1, CRC0, ETX **or** NAK
Get Time 0xF7	STX, 0xF7, LEN, CRC1, CRC0, ETX	STX, 0xF7, LEN, CN1, CN0, CD2, CD1, CD0, TD, TH, TL, TSS, CRC1, CRC0, ETX **or** NAK

Legend:

LEN	1 Byte	Length of command payload (0, 7 or 9 in these commands)
CN1,CN0	2 Bytes	Station code number (1..999)
CRC1,CRC0	2 Bytes	16-bit CRC value. Command byte and LEN are included in the computation.
CD2,CD1,CD0	3 Bytes	Calendar: Year, Month, Day
TD	1 Byte	Day of the week / Week counter / AM-PM
		Bit 7, Bit 6: Not used
		Bit 5, Bit 4: Relative week counter
		Bit 3..1: Day of week: 0 = Sunday, 1 = Monday, ..., 6 = Saturday
		Bit 0: Half of day: 0 = AM, 1 = PM
TH,TL	2 Bytes	12-hour time in seconds
TSS	1 Byte	Sub-second value, 1/256 sec

Table 8.1: Commands for time synchronization in the SPORTident communication protocol.

Command	Syntax	Response Syntax
Transmit record 0xD3	None (Autosend mode)	STX, 0xD3, LEN, CN1, CN0, SN3, SN2, SN1, SN0, TD, TH, TL, TSS, MEM2, MEM1, MEM0, CRC1, CRC0, ETX **or** NAK
Get backup 0x81	STX, 0x81, LEN, MEM2, MEM1, MEM0, NUM, CRC1, CRC0, ETX	STX, 0x83, LEN, CN1, CN0, MEM2, MEM1, MEM0, DATA, CRC1, CRC0, ETX **or** NAK

Legend:

SN4...SN0	4 Bytes	SI-Card Number
MEM2...MEM0	3 Bytes	Backup memory start address of the data record
NUM	1 Byte	Number of bytes to read. Must be 8 for a single punch record.
DATA	NUM Bytes	Backup data: CN3, CN2, CN1, CN0, TD, TH, TL, TSS

The remaining entities are explained in Table 8.1.

Table 8.2: Structure of the SPORTident punch record in Autosend mode.

practice, the ETX value of a link is estimated by checking how many out of five transmitted packets arrive at the receiver. If all packets are dropped, the ETX is 6. An exponentially weighed moving average is used to update the ETX value over time. CTP does not make a difference between forwarded and transmitted packets. This means that the same transmit queue is used for packets generated by the node itself as for packets received from others.

We wrote a test application that generates a packet every two seconds and tries to send it towards the base station. Each packet contains a 16-bit counter value. To see whether the nodes were able to discover each other, we added a special feature to the CTP routing engine: It displays its routing table on the five green LEDs. Each of our nodes was assigned a unique ID at compile time. The test worked well, all nodes were able to connect to the network and send data to the base station. Most of them used a direct connection, but occasionally one or two used a route via another node, i.e. relaying worked.

8.7 Issues

The hardest part of the software design was writing a radio driver for the RFM12BP module. It took several attempts to find a structure for the driver

that was compatible with the higher-level network layers of TinyOS and allowed for event-based communication with the module. We first tried to adapt the CC2420 driver for our module, but since that radio chip has much more features, it turned out to be unsuitable as a basis. Finally we based our driver on the RF212 driver. Most states and state transitions still had to be redefined, though.

One issue that we were unable to resolve is related to the interrupts that the radio module generates. Sometimes the module generates an interrupt without providing a recognizable interrupt bit, i.e. it is unknown why the interrupt is generated. There appears to be a regularity in the appearance of such interrupts, but we were unable to track down their source. However, they do not impact the operation of our driver since we can simply ignore them.

Chapter 9

Field Tests

9.1 Range Measurements

9.1.1 Hardware Revision 0

With our two first prototype nodes we conducted range measurements on Lake Zurich. The sender and receiver were moved along the lake borders to vary the distance between them. A GPS receiver at each of the nodes was used to determine their exact position during the experiments. The experiments were based on a simple transmitter program that sent 500 packets, each with the same data, and a receiver program that counted the correctly arrived packets. Each packet consisted of 26 bytes. The largest distance at which a reliable connection was still possible was 1.2 km. Fig. 9.1 shows the results in more detail.

The measured range was below our expectations. In a previous version of the RFM12BP datasheet, the range was specified as "more than 3000 m" [76], however this information has been removed in the current version. Other radio systems with similar specifications often claim to achieve 10 or even 20 km range. In section 7.3.5 we have shown that the transmit power of the module is even more than the specified 500 mW. If we use the free-space path loss model [77] to calculate the received signal strength at a distance of 1.2 km, we obtain:

$$P_{RX,dB} = P_{TX,dB} - FSPL = P_{TX,dB} - 20 \cdot \log\left(\frac{4\pi \cdot d \cdot f}{c}\right)$$

$$= 30\,\text{dBm} - 20 \cdot \log\left(\frac{4\pi \cdot 1200\ \text{m} \cdot 433.9\,\text{MHz}}{2.998 \cdot 10^8\ \text{m/s}}\right) = -56.78\,\text{dBm}$$

The receiver sensitivity of the RFM12BP, however, is specified as **-117 dBm**. In favor of the module, we could assume that the transmit power is 500 mW as specified, that only half of this power is radiated over the antenna, and that the range is 2.27 km. At this distance we once managed to transmit some packets, but did not get a reliable connection. Under these conditions, we get a sensitivity of *-68.32 dBm*, which is still way below the specifications.

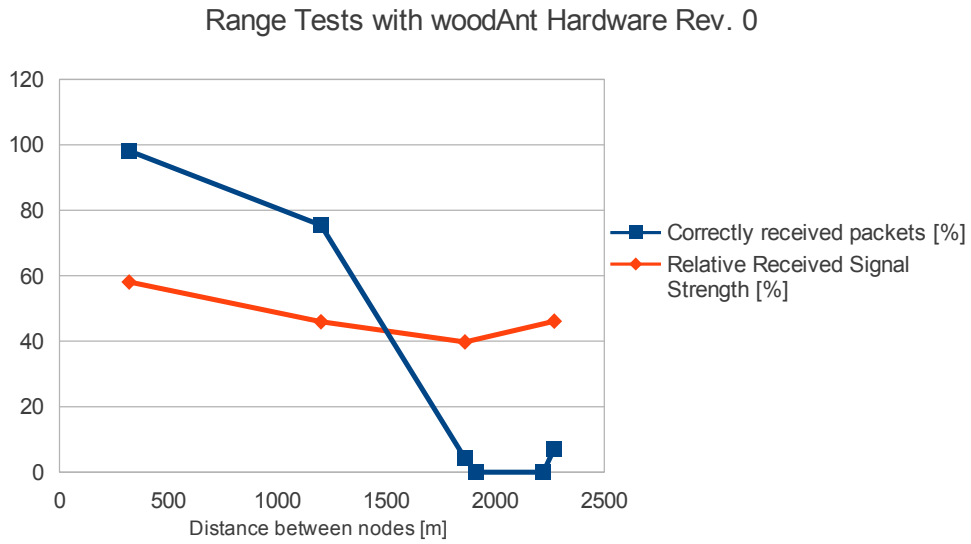


Figure 9.1: Results of the first range tests.

9.1.2 Hardware Revision 1

We also tested the range of the second prototype at Lake Zurich. There it became apparent that the range is drastically reduced when the antennas are pointing towards each other. This is due to the radiation pattern of a monopole whip antenna, which is not completely omnidirectional, but has the shape of a torus around the antenna. For the remaining tests, the nodes were held with their antennas perpendicular to the ground. Unfortunately the tests still yielded very bad results: At a distance of 320 meters, the connection was unreliable already, and only 46% of packets arrived. At a distance of 700 meters, no connection was possible at all. After the range tests of the first prototype, these results were clearly below the expectations.

We concluded that the antenna setup was the most probable reason for the bad results: The transmit power had been measured at the devices' SMA plug, so the problem had to be an insufficient radiation of energy from the antenna. The antenna was still the same as in the first prototype, but it had been placed at the side of the radio nodes' case, parallel to the PCB. This decision was justified by recommendations of the antenna manufacturer [31]. However, if the antenna is mounted in parallel to the ground plane, the ground plane should be at least a quarter wavelength (17.3 cm) long; maybe this is the reason for the bad performance.

In any case, we performed measurements using the hardware rev. 1 nodes with two different setups: One where the antenna was put into the vertical hole of the casing, perpendicular to the PCB, and one where the external 10×10 cm ground planes from the first prototypes were used. The nodes were mounted on poles for all experiments rather than holding them in hand to obtain more constant antenna characteristics. To allow for quick changes at both nodes, the measurements were performed on streets rather than on the lake. Some pictures of the measurement setup are given in Fig. 9.3, 9.5, and 9.4. On Birmensdorferstrasse, a particularly straight road in Zurich, we were able to do measurements with up to 890 meters distance between the nodes. Since the streets in question sometimes feature heavy traffic, and e.g. a truck in the radio channel will strongly affect performance, the reliability numbers may be slightly lower than with an ideal line of sight. The transmitter was programmed to send 600 packets with unique IDs, then wait around one minute and start sending again, etc. Results for all three setups are shown in Fig. 9.2. Clearly, the best range is achieved by using an additional ground plane to which the antenna is attached directly, but mounting the antenna in the hole of the node's case is already an improvement. The drawback of this setup is that the ground plane has a hole directly below the antenna, and the antenna is located near the border of the plane, which is far from optimal.

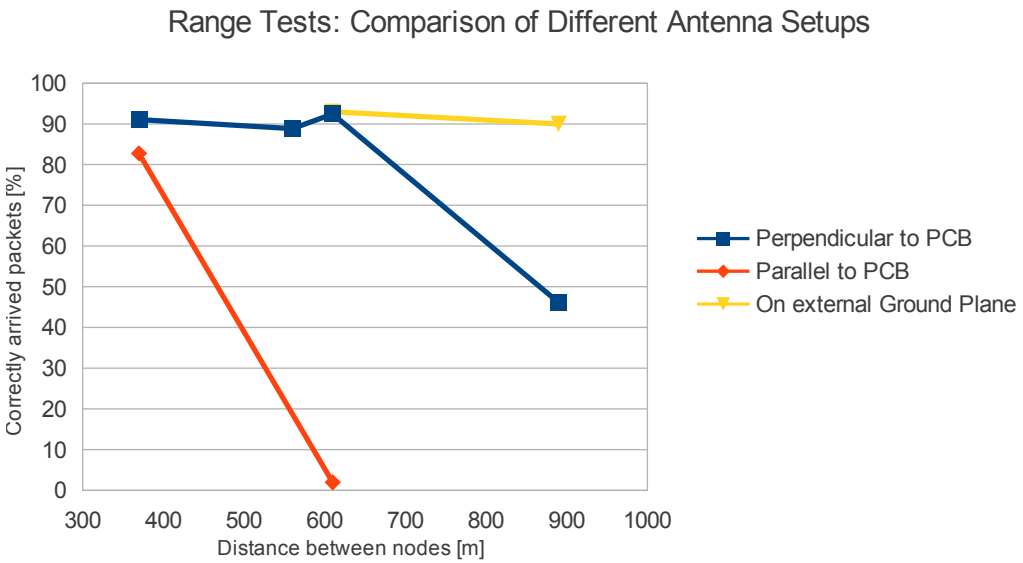


Figure 9.2: Results of the range tests with different antenna setups.

Figure 9.3: Example of a radio testing site in Zurich, with a woodAnt node mounted on the pole.

Figure 9.4: Test setup with an external ground plane.

Figure 9.5: woodAnt node with antenna mounted in the case hole.

Chapter 10

Conclusions

The main research question of this work was whether a multi-hop wireless sensor network can be built that works reliably in conditions commonly found at orienteering events. This was to be proven practically, i.e. by developing and building a system that performs well in such a setting. The hardware we have developed does not fully achieve the desired performance, but we have gained important insights on how future versions of it could be improved. Concrete ideas for improvements are listed in Section 6.4.3.

One important conclusion is that the selected radio module was not the best choice. We selected the RFM12BP module mainly due to its cheap price, the availability of two versions for different frequency bands, the high transmit power of 500 mW, and the generally convincing specifications in the module's datasheet. Unfortunately, some of these specifications have turned out to be completely wrong. To some extent, imprecisions of specifications had to be expected given the low price and the lack of reputation of the manufacturer *Hope Microelectronics*. But from the numerous reports we read about projects with the RFM12BP, we concluded that the module worked and was worth a try. After all, our system was to be optimized for low price, so that it could be built and sold in larger quantities if it was successful. The module does work, but its range is much lower than that of comparable devices from other manufacturers. We will therefore use a different radio module to continue our development.

Another insight we gained is that power supply is not the main issue for our application. Our radio nodes can be operated in receive mode for 75 hours with a cellular phone battery, and if the transmitter is turned on with a 10% duty cycle, battery life drops to 13.1 hours. Other radio modules that we considered may draw higher currents in receive mode, but all of them draw less in transmit mode. If we used the *Digi XBee-PRO 868* radio module, for instance, battery life would be reduced to around 11 hours, which is still within our desired range.

Whether or not a sufficient range can be achieved in a forest has to remain unanswered. We did not perform any range tests in a forest since the line-of-sight range seemed insufficient to get conclusive results from such tests.

With respect to software, we can conclude that TinyOS is a useful basis for developing embedded system software. The scheduler and the innovative architecture of nesC make it easy to write event-based code that can be run concurrently with other tasks on a microcontroller. We have also profited from the rich code base of TinyOS, which contains e.g. a complete radio layer stack and predefined routing protocols. CTP, the default data collection protocol of TinyOS, has been successfully tested on top of our radio driver. However, we think that the routing could still be optimized: A protocol like *Dozer* may drastically reduce the radio duty cycle of our nodes compared to CTP without affecting performance. Unfortunately many of the routing protocols developed for TinyOS are research protocols, i.e. they are not being actively maintained as free software projects. For instance, there is no stable version of Dozer available that runs on TinyOS 2.x, so a considerable development effort would be required to use it in our project.

Lessons learned We have gained valuable experience in the field of embedded systems design during this project. An important point is that, unlike in integrated circuit design, it is not necessary (and seldom possible) to aim for "first-time-right" design in this field. Prototypes can be built pretty quickly, and it is often easier to develop a custom PCB early on than to work with clumsy and unreliable circuits on evaluation boards or "breadboards". What is often underestimated about developing a PCB is the effort of searching all the parts of it. Often several component suppliers have to be used, and long lists of parts have to be evaluated to find the best option. This process may easily take more time than drawing the schematic of the PCB.

In this project, we designed a PCB using the free software tool *KiCAD* for the first time. The tool is easy to use and provides all the functionality necessary for a design project. It was also the first board that we submitted to a professional manufacturer for production. Thanks to the clearly defined submission process and careful design rule checks, we did not encounter any problems while submitting the PCB.

One of the hardest things about developing a device, particularly a compact one, is to plan the complete assembly without errors. This is often more of a challenge than designing an error-free PCB. In our design, we had problems mainly with the cable connection from the board to the antenna, as discussed in

Section 7.3.6.

A special task in the development of our hardware was to build a step-up converter from 3.7 V to 12 V. Initially we were skeptical as to whether such a converter would fit on the board, given that it had to supply more than 200 mA to the radio module. However, with the TPS61085 IC, the converter turned out very compact and was surprisingly easy to design. The fact that the circuit worked flawlessly on the PCB, even though we were unable to test it beforehand, was even more surprising.

Chapter 11

Outlook

11.1 Next Hardware Generation

There is much room for improvements to the hardware of our woodAnt sensor nodes. As discussed in Section 9.1.1, the radio module's performance is poor compared to its transmit power. We will therefore use a different module for the next hardware revision, either the Digi XBee-PRO 868 [78], or one of the modules in the 169 MHz band. The XBee-PRO has a line-of-sight range of up to 40 km according to its manufacturer, so we hope it will achieve at least 1 km in forests, despite the high frequency. The module is affordable and compact; it seems well suited for our application.

We might want to use a different microcontroller as well. As mentioned in section 6.4.3, the MSP430 series of controllers would need an additional DC/DC converter circuit in order to work within its specifications, whereas Atmel controllers have a larger supply voltage range. Since there is no prohibitive difference between the two controller series, switching to Atmel may be the easiest and cheapest solution. Since other parts are being changed as well, the PCB will have to be completely redesigned anyway. The software will need only minor changes.

With the insights gained about antenna performance, we will also consider using a different case. SPORTident offers a version of its BSF7 case without a hole. This increases the available space on the PCB. The possibilities for antenna placement will still be limited due to the battery, but we may be able to move the antenna closer to the center of the board. In any case, the suboptimal ground plane will not be an issue anymore: At 869 MHz, a half-wave dipole antenna can be used.

A meaningful detail that could be optimized is the clock generation. We used two crystals in hardware revisions 0 and 1, mainly because we wanted to avoid

any issues with inaccurate clock frequencies. The slow 32.768 kHz crystal is used mainly for timers and is necessary for exact time measurements. However, we noticed that with the timer abstraction of TinyOS, the accuracy of timers is much worse than when they are used directly. One way around this would be to exclude one hardware timer from the abstraction and write a specific routine to use it as a real time clock. In this case the 32.768 kHz crystal would still be needed. The other option is to add an external RTC chip to the PCB, which could provide exact time signals via the MCU's interrupt pins. In both cases, the controller's internal oscillator could be used as the main clock. Its accuracy is sufficient to generate UART signals, at least for the baud rates that are required by SPORTident control stations (up to 38.4 kbps). An 8 MHz crystal is thus not necessary.

During development, it became clear that feedback to the user is very important. With the eight LEDs that are currently on the board, a lot of information can be displayed, but it will never be self-explanatory. Since our system is supposed to be easy to use even for beginners, it must provide a human-readable feedback. Nowadays graphical Liquid Cristal Displays (LCD) are available for 5 USD [79], or sometimes even cheaper [80]. Even though the amount of information to be displayed is small (mainly the radio connection status), such a display would make the system more user-friendly and should therefore be added.

A complete user interface also comprises buttons. With only one user button, no user-friendly interface can be designed. And since the button is soldered to the PCB in our current hardware, it can only be accessed when the case is open. Any buttons that need to be accessible to the end user should thus be replaced by capacitive buttons that can be pushed from outside the case. Atmel offers such solutions with their *QTouch* technology, consisting of a software library for its microcontrollers and sensor ICs such as the AT42QT1040 [81]. Despite the elegance and affordability of such a solution, it should not be exaggerated: A single on/off switch is probably sufficient. Special functions like timed wake-up could always be programmed from the base station via radio.

Finally, the cable connection needs to be optimized. A connector for a battery charger could be incorporated into it, so that the case does not have to be opened to recharge the battery. The connection should also be made more waterproof: The SPORTident control stations with serial cables are usually intended for indoor use, and the protector at the entry point of the cable is not as waterproof as the rest of the case. The RS232 connector itself could be replaced by a waterproof connector to avoid the risk of short circuits when it gets too wet. An alternative would be to integrate the radio system into the case of the SPORTident control station directly - but since this would require modifications

to the design of the control stations themselves, it should be done in cooperation with SPORTident. It also wouldn't solve the cable problem entirely, unless the nodes could be charged using an induction-based system.

11.2 Other Application Scenarios

The woodAnt radio nodes constitute a wireless sensor network optimized for short-term, long-range applications. They could be used for any measurement tasks with these requirements if appropriate sensors are connected to them. However, typical applications of sensor networks like monitoring of temperature, humidity or seismic activity need nodes that allow long-term deployment. To enable the woodAnt nodes for such applications, an external power source such as a solar panel would be necessary.

An interesting scenario that is closely related to the one discussed in this work is the GPS-based tracking of runners. If the size of the nodes could be further reduced, possibly at the cost of reducing battery life to one or two hours, they could be used to track the position of runners in world-class orienteering events. GPS trackers that communicate over GSM or GPRS networks are widely available and have become very compact, but they cannot be used when GSM coverage is not available. A radio based tracking system would work just like our system for control stations, with the added difficulty of permanent mobility. If both systems were used simultaneously, the fixed radio nodes could act as an infrastructure network and collect data from the mobile nodes. Routing of packets to the base station would then be much more straightforward: Whenever a mobile node is within range of an infrastructure node, it can transmit its position data and have it forwarded to the base station over a stable route.

Bibliography

[1] "About Orienteering," 2011. [Online]. Available: http://orienteering.org/about-orienteering/. Archived at: http://www.webcitation.org/62sDMcGpc

[2] "SPORTident," 2011.

[3] "EMIT," 2011. [Online]. Available: http://emit.no/en

[4] "Wikipedia: Postenkontrollsystem," 2011. [Online]. Available: http://de.wikipedia.org/wiki/Postenkontrollsystem?stableid=82994683. Archived at: http://www.webcitation.org/62sDRrRKF

[5] "OTS: Orienteering Telemetry System," 2010. [Online]. Available: http://www.gpprojects.com/si.htm. Archived at: http://www.webcitation.org/6330rKP19

[6] "O-Lynx: Radio Link Technology for Orienteering," 2011. [Online]. Available: http://o-lynx.com. Archived at: http://www.webcitation.org/6330fEBe5

[7] "SPORTident GSM Radio System," 2009. [Online]. Available: http://www.sportident.com/media/public/pdf/SPORTident-GSM-Set%20_radio_en.pdf. Archived at: http://www.webcitation.org/62rppPvrn

[8] M. Zoller, "Design Considerations for Radio Controls in Orienteering," 2010. [Online]. Available: http://martin.zoller.tv/publications/2010/ikt_semester_thesis_radio_controls.pdf. Archived at: http://www.webcitation.org/62sFDNA1V

[9] "SPORTident goes wireless," 2009. [Online]. Available: http://www.sportident.com/index.php?site=site.html&dir=&sisess=es7lq8eace4l06c93crju658q0&siteid=979&nav=190&entryid=83. Archived at: http://www.webcitation.org/62rpqV8vm

[10] "SPORTident SRR-Kit," 2009. [Online]. Available: http://blog.plohni.com/wp-content/uploads/2009/09/SPORTident-SRR-kit_en.pdf. Archived at: http://www.webcitation.org/62rqj0KYy

[11] "SPORTident WBOX GSM," 2009. [Online]. Available: http://blog.plohni.com/wp-content/uploads/2009/09/SPORTident-WBOX_de.pdf. Archived at: http://www.webcitation.org/62rqiPHp9

[12] M. Plohn, "SPORTident WBOX GSM & BSF8-SRR, ein erster Funkpostenversuch," 2009. [Online]. Available: http://blog.plohni.com/tag/wbox-gsm/. Archived at: http://www.webcitation.org/62rqt2BBJ

[13] "Vernetztes Identifikations- und Timingsystem mit mobilen Erfassungs- und Informationsgeräten für den Schul-, Breiten- und Leistungssport im Outdoorbereich, publisher = AES GmbH," 2008.

[14] "Tinynode 184," 2011. [Online]. Available: http://www.tinynode.com/?q=product/tinynode184/tn-184-868

[15] "AQUAMON Soil Monitoring Network," 2011. [Online]. Available: http://www.cermetek.com/Catalog/SensorMon/DataSheet/AquaMon/. Archived at: http://www.webcitation.org/62rtEAkgl

[16] "EM50R Wireless Radio Data Logger," 2011.

[17] J. D. Lea-Cox, A. G. Ristvey, D. S. Ross, and G. F. Kantor, "Using Wireless Sensor Technology to Schedule Irrigations and Minimize Water Use in Nursery and Greenhouse Production Systems," in *Combined Proceedings*, vol. 58. Seattle, WA, USA: International Plant Propagators' Society, 2008. [Online]. Available: http://www.psla.umd.edu/faculty/lea-cox/Publications/Acta%20Hort/Lea-Cox%20et%20al,%202008.%20IPPS%2058_512-517.pdf. Archived at: http://www.webcitation.org/62tYTzduE

[18] "Frequency Allocation Plan," 2011. [Online]. Available: http://www.ofcomnet.ch. Archived at: http://www.webcitation.org/62MJYKfzA

[19] "Commission decision of 30 June 2010 amending Decision 2006/771/EC on harmonisation of the radio spectrum for use by short-range devices," 2010. [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2010:166:0033:0041:EN:PDF. Archived at: http://www.webcitation.org/62MJmTeZ5

[20] "Technical Interfaces Regulations - 169 MHz Meter Reading Systems," 2009. [Online]. Available: http://www.ofcomnet.ch/cgi-bin/rir.pl?id=1003;nb=04. Archived at: http://www.webcitation.org/63D3BPuzb

[21] "Erkennung, Aufspürung und Ortung von Personen und Objekten," 2009. [Online]. Available: http://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/BNetzA/Sachgebiete/Telekommunikation/Regulierung/Frequenzordnung/

Allgemeinzuteilung/FundstelleId17781pdf.pdf;jsessionid=
AC7F7FFDD4A94C36BC50CD297DB90F43?__blob=publicationFile.
Archived at: http://www.webcitation.org/63D2qqFaw

[22] "Commission Decision of 20 December 2005 on the harmonisation of the 169,4-169,8125 MHz frequency band in the Community," 2005. [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2005:344:0047:0051:EN:PDF. Archived at: http://www.webcitation.org/63D4ilD4U

[23] "Commission Decision of 13 August 2008 amending Decision 2005/928/EC on the harmonisation of the 169,4-169,8125 MHz frequency band in the Community," 2008. [Online]. Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2008:220:0029:0029:EN:PDF. Archived at: http://www.webcitation.org/63D4qAGC4

[24] "Harmonised frequency ranges," 2011. [Online]. Available: http://www.bakom.admin.ch/themen/frequenzen/00652/00753/index.html?lang=en. Archived at: http://www.webcitation.org/62MJkVcOs

[25] T. Tamir, "On radio-wave propagation in forest environments," *Antennas and Propagation, IEEE Transactions on*.

[26] H.-Y. Chen and Y.-Y. Kuo, "Calculation of radio loss in forest environments by an empirical formula," *Microwave and Optical Technology Letters*, vol. 31, no. 6, pp. 474–480, 2001. [Online]. Available: http://dx.doi.org/10.1002/mop.10066

[27] J. Gay-Fernandez, M. Garcia Sanchez, I. Cuinas, A. Alejos, J. G. Sanchez, and J. Miranda-Sierra, "Propagation analysis and deployment of a wireless sensor network in a forest," *Progress In Electromagnetics Research*, vol. 106, pp. 121–145, 2010.

[28] "Antennas: Design, Application, and Performance," 2011. [Online]. Available: http://www.linxtechnologies.com/resources/documents/an-00500.pdf. Archived at: http://www.webcitation.org/5yxx368h4

[29] "Wikipedia: Characteristic Impedance," 2011. [Online]. Available: http://en.wikipedia.org/wiki/Characteristic_impedance. Archived at: http://www.webcitation.org/62Os0RBGf

[30] "Microstrip Line Calculator," 2011. [Online]. Available: http://www1.sphere.ne.jp/i-lab/ilab/tool/ms_line_e.htm. Archived at: http://www.webcitation.org/63EO8sAVV

[31] "Understanding Antenna Specifications and Operation," 2011. [Online]. Available: http://www.linxtechnologies.com/resources/documents/an-00501.pdf. Archived at: http://www.webcitation.org/5yxxHtz6u

[32] R. Wattenhofer, "Lecture Notes: Ad-hoc and Sensor Networks," 2010. [Online]. Available: http://www.disco.ethz.ch/lectures/hs10/asn/lecture/6/chapter06mediaaccess.pdf. Archived at: http://www.webcitation.org/639J5QWWu

[33] "Wikipedia: Time Division Multiple Access," 2011. [Online]. Available: http://en.wikipedia.org/wiki/Time_division_multiple_access. Archived at: http://www.webcitation.org/639MLEY6z

[34] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, 2002, pp. 1567 – 1576 vol.3. [Online]. Available: http://www.isi.edu/~johnh/PAPERS/Ye02a.pdf

[35] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, ser. SenSys '04. New York, NY, USA: ACM, 2004, pp. 95–107. [Online]. Available: http://doi.acm.org/10.1145/1031495.1031508

[36] W. Ye, F. Silva, and J. Heidemann, "Ultra-low duty cycle MAC with scheduled channel polling," in *Proceedings of the 4th international conference on Embedded networked sensor systems*, ser. SenSys '06. New York, NY, USA: ACM, 2006, pp. 321–334. [Online]. Available: http://doi.acm.org/10.1145/1182807.1182839

[37] M. Buettner, G. V. Yee, E. Anderson, and R. Han, "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks," in *Proceedings of the 4th international conference on Embedded networked sensor systems*, ser. SenSys '06. New York, NY, USA: ACM, 2006, pp. 307–320. [Online]. Available: http://doi.acm.org/10.1145/1182807.1182838

[38] N. Burri, P. von Rickenbach, and R. Wattenhofer, "Dozer: ultra-low power data gathering in sensor networks," in *Proceedings of the 6th international conference on Information processing in sensor networks*, ser. IPSN '07. New York, NY, USA: ACM, 2007, pp. 450–459. [Online]. Available: http://doi.acm.org/10.1145/1236360.1236417

[39] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959, 10.1007/BF01386390. [Online]. Available: http://dx.doi.org/10.1007/BF01386390

[40] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, ser. SenSys '03. New York, NY, USA: ACM, 2003, pp. 14–27. [Online]. Available: http://webs.cs.berkeley.edu/papers/sensys_awoo03.pdf

[41] "TinyOS 1.x Source Tree: MintRoute," 2005. [Online]. Available: http://www.tinyos.net/tinyos-1.x/tos/lib/MintRoute/. Archived at: http://www.webcitation.org/63Bb4CUEX

[42] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo, "The Collection Tree Protocol (CTP)," 2006. [Online]. Available: http://www.tinyos.net/tinyos-2.x/doc/html/tep123.html. Archived at: http://www.webcitation.org/63Bb6w3Nm

[43] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '09. New York, NY, USA: ACM, 2009, pp. 1–14. [Online]. Available: http://sing.stanford.edu/gnawali/ctp/sensys09-ctp.pdf

[44] "pico]OS RT Operating System," 2010. [Online]. Available: http://picoos.sourceforge.net/index.html. Archived at: http://www.webcitation.org/62zTo3OOI

[45] "FreeRTOS: Features Overview," 2011. [Online]. Available: http://www.freertos.org/FreeRTOS_Features.html. Archived at: http://www.webcitation.org/62zTYokuj

[46] "The Contiki OS: About Contiki," 2010. [Online]. Available: http://www.contiki-os.org/p/about-contiki.html. Archived at: http://www.webcitation.org/62zTEN1P6

[47] "TinyOS," 2011. [Online]. Available: http://tinyos.net. Archived at: http://www.webcitation.org/62zTjd1tt

[48] I. Talzi, A. Hasler, S. Gruber, and C. Tschudin, "PermaSense: investigating permafrost with a WSN in the Swiss Alps," in *Proceedings of the 4th workshop on Embedded networked sensors*, ser. EmNets '07. New York, NY, USA: ACM, 2007, pp. 8–12. [Online]. Available: http://doi.acm.org/10.1145/1278972.1278974

[49] G. Oberholzer, P. Sommer, and R. Wattenhofer, "SpiderBat: Augmenting wireless sensor networks with distance and angle information," in *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, April 2011, pp. 211 –222.

[50] "BTnode rev3 Product Brief," 2005. [Online]. Available: http://www.btnode.ethz.ch/pub/files/btnode_rev3.22_productbrief.pdf. Archived at: http://www.webcitation.org/62zWkicdb

[51] "8-bit Atmel Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash: Preliminary Summary," 2011. [Online]. Available: http://atmel.com/dyn/resources/prod_documents/2549S.pdf. Archived at: http://www.webcitation.org/62znTe7OK

[52] "MSP430F15x, MSP430F16x, MSP430F161x Mixed Signal Microcontroller," 2011. [Online]. Available: http://www.ti.com/lit/gpn/msp430f1611. Archived at: http://www.webcitation.org/5yy2vmtkc

[53] "TinyOS porting for small MSP430 chips," 2011. [Online]. Available: http://code.google.com/p/tinyos-msp430/. Archived at: http://www.webcitation.org/5yy1qC6wY

[54] "AVR Libc Home Page," 2011. [Online]. Available: http://www.nongnu.org/avr-libc/. Archived at: http://www.webcitation.org/62zhsxtQ5

[55] "AVR Downloader/UploaDEr," 2011. [Online]. Available: http://savannah.nongnu.org/projects/avrdude. Archived at: http://www.webcitation.org/62zi09zRM

[56] "Simulavr: an AVR simulator," 2010. [Online]. Available: http://savannah.nongnu.org/projects/simulavr/. Archived at: http://www.webcitation.org/62zhPMBcQ

[57] "AVaRICE," 2011. [Online]. Available: http://avarice.sourceforge.net/. Archived at: http://www.webcitation.org/62zi4lYs5

[58] "The GCC toolchain for the Texas Instruments MSP430 MCUs," 2011. [Online]. Available: http://mspgcc.sourceforge.net/. Archived at: http://www.webcitation.org/62zj7IN5t

[59] "MSPDebug," 2011. [Online]. Available: http://mspdebug.sourceforge.net/. Archived at: http://www.webcitation.org/62zjytNOi

[60] "TPS61221: Low Input Voltage Step-up Converter in 6-pin SC-70 Package," 2011. [Online]. Available: http://www.ti.com/lit/gpn/tps61221. Archived at: http://www.webcitation.org/5yxureLUc

[61] "KiCAD GPL PCB Suite," 2011. [Online]. Available: http://kicad.sourceforge.net/wiki/DE:Main_Page

[62] "MSP430-P1611 Development Board," 2011. [Online]. Available: http://olimex.com/dev/msp-p1611.html. Archived at: http://www.webcitation.org/62WV91FtO

[63] "ANT-433-PW-QW Datasheet," 2011. [Online]. Available: http://www.antennafactor.com/resources/data-guides/ant-433-pw-qw.pdf. Archived at: http://www.webcitation.org/5yxxzIydC

[64] "LM1117: 800mA Low-Dropout Linear Regulator," 2006. [Online]. Available: http://www.national.com/ds/LM/LM1117.pdf. Archived at: http://www.webcitation.org/637IcNtEq

[65] "Wikipedia: IP Code," 2011. [Online]. Available: http://en.wikipedia.org/wiki/IP_Code. Archived at: http://www.webcitation.org/63DrMYRM5

[66] "Application Report: MSP430 32-kHz Crystal Oscillators," 2009. [Online]. Available: http://www.ti.com/lit/pdf/slaa322

[67] "TPS61085: 650 kHz/1.2 MHz, 18.5 V Step-up DC-DC Converter," 2011. [Online]. Available: http://www.ti.com/lit/ds/symlink/tps61085.pdf. Archived at: http://www.webcitation.org/62zzh4Ski

[68] "Universal ISM Band FSK Transceiver Module with 500 mW Output Power: RFM12BP," 2006. [Online]. Available: http://www.hoperf.com/upload/rf/RFM12BP.PDF. Archived at: http://www.webcitation.org/63DibJV5R

[69] "SkyTraq Venus 6 GPS Module ST22," 2010. [Online]. Available: http://www.perthold.de/BINARY/gps-st22.pdf. Archived at: http://www.webcitation.org/63DvkZnhG

[70] "Format für Extended Gerber RS274X Daten im PCB-POOL®," 2011. [Online]. Available: http://www.pcb-pool.com/download/spezifikation/deu_cmso020_ext_gerber.pdf. Archived at: http://www.webcitation.org/63DwEpZyT

[71] "RF12B Programming Guide," 2006. [Online]. Available: http://www.hoperf.com/upload/rf/RF12B_code.pdf. Archived at: http://www.webcitation.org/63Dp8L7GT

[72] "RF12B Universal ISM Band FSK Transceiver," 2006. [Online]. Available: http://www.hoperf.com/upload/rf/RF12B.pdf. Archived at: http://www.webcitation.org/63Dppe0xA

[73] "Si4421 Universal ISM Band FSK Transceiver," 2011. [Online]. Available: http://www.silabs.com/Support%20Documents/TechnicalDocs/Si4421.pdf. Archived at: http://www.webcitation.org/5zgVL25Rh

[74] "Mikrocontroller.net: RFM12," 2011. [Online]. Available: http://www.mikrocontroller.net/articles/RFM12. Archived at: http://www.webcitation.org/5zf4Q9HbT

[75] "RFM12 Library for MSP430," 2011. [Online]. Available: http://svn.assembla.com/svn/msp430/

[76] "RFM12BP Datasheet Rev 2.2," 2006. [Online]. Available: http://www.ottomat.hu/archivum/RF_modulok/RFM12BP.pdf. Archived at: http://www.webcitation.org/63E0r44Wj

[77] "Wikipedia: Free-space Path Loss," 2011. [Online]. Available: http://en.wikipedia.org/wiki/Free-space_path_loss. Archived at: http://www.webcitation.org/63E1BSjjo

[78] "XBee-PRO 868: Long-range embedded modules for OEMs," 2011. [Online]. Available: http://www.digi.com/pdf/ds_xbeepro868.pdf. Archived at: http://www.webcitation.org/63Ee8ONHB

[79] "LCD Character Display Modules: NMTC-S0802XRYHS-10," 2011. [Online]. Available: http://ch.mouser.com/ProductDetail/Microtips-Technology/NMTC-S0802XRYHS-10/?qs=sGAEpiMZZMt7dcPGmvnkBmsW81dYzVy9VmzJf3mmgRQ%3d. Archived at: http://www.webcitation.org/635sRPbLr

[80] "LCD-Modul C0802-04," 2011. [Online]. Available: http://www.pollin.de/shop/dt/NzczOTc4OTk-/Bauelemente_Bauteile/Aktive_Bauelemente/Displays/LCD_Modul_C0802_04.html. Archived at: http://www.webcitation.org/635sZqIJM

[81] "AT42QT1040: QTouch 4-key Sensor IC," 2011. [Online]. Available: http://www.atmel.com/dyn/resources/prod_documents/doc9524.pdf. Archived at: http://www.webcitation.org/635vbpylE

Appendix A

Comparison of Common Batteries for Consumer Electronics Devices

Battery/Device	Voltage	Capacity	Orig. price	Clone price	Charger price
Samsung i9000	3.7 V	1500 mAh	USD 13.90	USD 3.00 to 5.30	USD 3.30 to 8.39
Nokia BP-4L	3.7 V	1500 mAh	USD 19.21	USD 3.10 to 7.17	USD 3.31 to 6.00
Nokia BL-5J	3.7 V	1320 mAh	USD 15.56	USD 2.70 to 8.70	USD 4.50 to 7.70
Sony Ericsson BST-41	3.7 V	1500 mAh	CHF 39.00	USD 3.80	USD 8.50
HTC Evo 4G	3.7 V	1500 mAh	unknown	USD 5.00 to 6.00	USD 8.20
Huawei HB4F1	3.7 V	1500 mAh	unknown	USD 3.80 to 5.40	
Motorola BP6X	3.7 V	1390 mAh	USD 11.00	USD 5.30	USD 4.00
Sony PSP	3.7 V	1200 mAh	USD 15.90	USD 3.60 to 4.57	USD 5.56 to 15.68
Nintendo DS lite	3.7 V	1600 mAh	unknown	USD 3.00 to 4.37	USD 5.33 to 5.74

Table A.1: Some easily available batteries for consumer electronics devices, with prices surveyed mainly on DealExtreme.com.

Appendix B

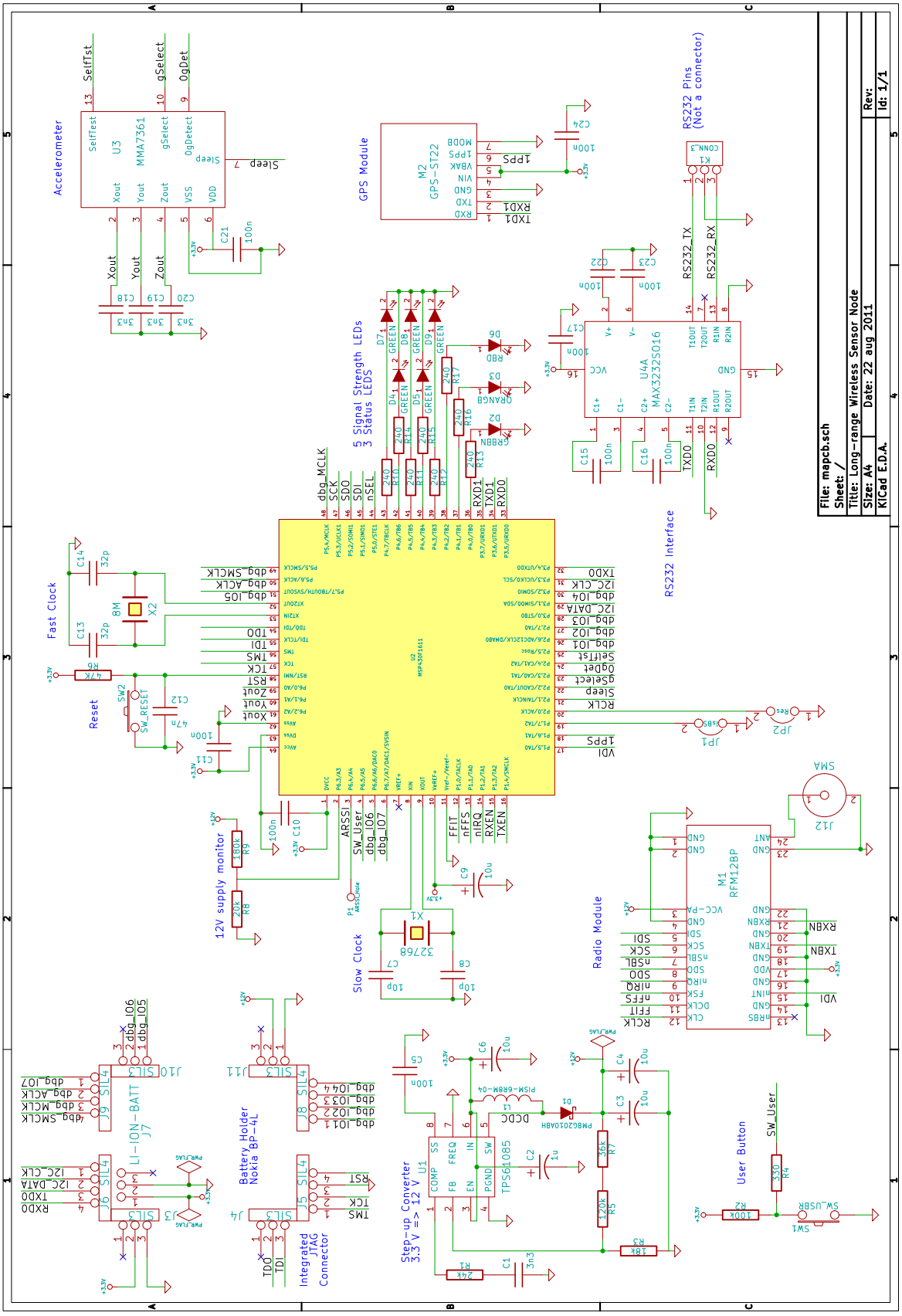
PCB Bill of Materials

Ref	Value	Footprint	Supplier	Order No.	Qty
C1	3n3	SM0603	TIK	n/a	25
C2	1u	SM0603	Mouser	810-C1608Y5V1C105Z	5
C3	10u	SM1206	Mouser	810-C3216Y5V1C106Z	25
C4	10u	SM1206	see C3	see C3	
C5	100n	SM0603	Mouser	810-C1608Y5V1C104Z	50
C6	10u	SM1206	see C3	see C3	
C7	10p	SM0603	TIK	n/a	10
C8	10p	SM0603	see C7	see C7	
C9	10u	SM1206	see C3	see C3	
C10	100n	SM0603	see C5	see C5	
C11	100n	SM0603	see C5	see C5	
C12	47n	SM0603	Mouser	810-C1608X7R1H473M	5
C13	33p	SM0603	Mouser	77-VJ0603A330JXAAC	10
C14	33p	SM0603	see C13	see C13	
C15	100n	SM0603	see C5	see C5	
C16	100n	SM0603	see C5	see C5	
C17	100n	SM0603	see C5	see C5	
C18	3n3	SM0603	see C1	see C1	
C19	3n3	SM0603	see C1	see C1	
C20	3n3	SM0603	see C1	see C1	
C21	100n	SM0603	see C5	see C5	
C22	100n	SM0603	see C5	see C5	
C23	100n	SM0603	see C5	see C5	
C24	100n	SM0603	see C5	see C5	
D1	PMEG2010AEH	SOD123F	Mouser	771-PMEG2010AEH-T/R	5
D2	GREEN	LED-0603	Mouser	630-HSMG-C190	30
D3	YELLOW	LED-0603	TIK	n/a	5
D4	GREEN	LED-0603	see D2	see D2	
D5	GREEN	LED-0603	see D2	see D2	
D6	RED	LED-0603	TIK	n/a	5
D7	GREEN	LED-0603	see D2	see D2	
D8	GREEN	LED-0603	see D2	see D2	
D9	GREEN	LED-0603	see D2	see D2	
J3	SIL3	PIN_ARRAY_3X1	Mouser	535-03-0518-10	25
J4	SIL3	PIN_ARRAY_3X1	see J3	see J3	
J5	SIL4	PIN_ARRAY_4X1	Mouser	535-04-0518-10	25
J6	SIL4	PIN_ARRAY_4X1	see J5	see J5	
J7	LI-ION-BATT	LI-ION-BATT	TIK	Self-made contacts	15
J8	SIL4	PIN_ARRAY_4X1	see J5	see J5	
J9	SIL4	PIN_ARRAY_4X1	see J5	see J5	
J10	SIL3	PIN_ARRAY_3X1	see J3	see J3	
J11	SIL3	PIN_ARRAY_3X1	see J3	see J3	
J12	SMA	SMA-STR	Mouser	571-5-1814400-1	5
JP1	isBS	SM1206	TIK	n/a	10
JP2	Res	SM1206	see JP1	see JP1	
K1	CONN_3	PIN_ARRAY_3X1	n/a	Just holes	
L1	PISM-6R8M-04	FASTRON-PISM	Reichelt	L-PISM 6,8µ	6
M1	RFM12BP	RFM12BP-433	Pollin	94-810 115	2
M2	GPS-ST22	GPS-ST22	Perthold Eng.	n/a	1
P1	ARSSI_Hole	PIN_ARRAY_1X1	n/a	Just a hole	
R1	24k	SM0603	Mouser	652-CR0603JW-243ELF	10
R2	100k	SM0603	TIK	n/a	5
R3	18k	SM0603	Mouser	652-CR0603JW-183ELF	10
R4	330	SM0603	TIK	n/a	5
R5	120k	SM0603	Mouser	667-ERJ-3EKF1203V	10
R6	47K	SM0603	Mouser	652-CR0603-JW-473ELF	10
R7	36k	SM0603	Mouser	71-CRCW0603J-36K-E3	10
R8	20k	SM0603	TIK	n/a	5
R9	180k	SM0603	Mouser	652-CR0603-JW-184ELF	10
R10	240	SM0603	Mouser	652-CR0603-JW-241ELF	100
R11	240	SM0603	see R10	see R10	
R12	240	SM0603	see R10	see R10	
R13	240	SM0603	see R10	see R10	
R14	240	SM0603	see R10	see R10	
R15	240	SM0603	see R10	see R10	
R16	240	SM0603	see R10	see R10	
R17	240	SM0603	see R10	see R10	
SW1	SW_USER	SMT_SWITCH_CKCOMP	Mouser	611-PTS645SK43SMTRLF	10
SW2	SW_RESET	SMT_SWITCH_CKCOMP	see SW1	see SW1	
U1	TPS61085	TSSOP8	Mouser	595-TPS61085PWR	5
U2	MSP430F1611	PQFP64	Mouser	595-MSP430F1611IPMR	5
U3	MMA7361	LGA16	Mouser	841-MMA7361LCT	5
U4	MAX3232CPWR	maxim-7a-SO16	TIK	n/a	5
X1	32768	CRYSTAL2	Mouser	AB26T-32.768KHZ	5
X2	8M	CRYSTAL1	Mouser	815-ABL-8-B2	5
ANT1	ANTENNA	ANTENNA_SMA	Mouser	712-ANT-433-CW-QW	5
CN1	RF_CONN	PLUG_SMA_RG174	Mouser	712-CONSMA007	5
CN2	RF_CABLE	CABLE_RG174	Mouser	566-8216-100	1 (100 ft!)
CN3	RF_CONN	BLKHD_RPSMA_RG174	Mouser	712-CONREVSMA014	5
BAT1	BP-4L	NOKIA_BP-4L	DealExtreme	39142	10
CAS1	CASE	BSF7_CASE	SPORTident	33010	5
CN4	DB9_CABLE	CABLE_SI_6LEAD	SPORTident	Included with cases	5
CN5	DB9_PLUG	DB9_PLUG_MALE	Mouser	706-301A10019X	5

Table B.2: Bill of materials for our PCB.

Appendix C

PCB Schematic


File: mapcb.sch
Sheet: /
Title: Long-range Wireless Sensor Node
Size: A4
Date: 22 aug 2011
Rev:
KiCad E.D.A.
Id: 1/1
Accelerometer
GPS Module
5 Signal Strength LEDs
3 Status LEDS
RS232 Pins
(Not a connector)
RS232 Interface
Fast Clock
Reset
12V supply monitor
Slow Clock
Radio Module
SMA
Step-up Converter
3.3 V => 12 V
LI-ION-BATT
J7
Battery Holder
Nokia BP-4L
Integrated
JTAG
Connector
User Button

Appendix D

Radio Modules

Manufacturer	Device Name	Freq. Band	Size	Max P_{TX}	RX sensitivity
Radiometrix	BiM1H	151.3 MHz	33×23×12 mm	500 mW	-120 dBm @ 12 dB SINAD
Radiometrix	SHX1	140-175 MHz	67×30×9 mm	500 mW	-118 dBm @ 12 dB SINAD
Radiometrix	ENX1	169.40-169.44 MHz	49×34×8.7 mm	100 mW	-120 dBm @ 12 dB SINAD
Radiometrix	TR1M	135-175 MHz	59×38×12 mm	100 mW	-118 dBm @ 12 dB SINAD
Radiometrix	UHX1	140-175 MHz	67×30×12 mm	500 mW	-118 dBm @ 12 dB SINAD
Amber Wireless	AMB8355	869.4-869.65 MHz	97×38×15 mm	500 mW	-110 dBm
Telit	TinyOne Pro 868	869.4-869.65 MHz	38×21×4 mm	500 mW	-105 dBm @ .001 BER
Telit	PowerOne 868	869.4-869.65 MHz	94×52×13.5 mm	500 mW	-115 dBm @ .001 BER
Atim	ARM-C8	868 MHz	64×32×4 mm	500 mW	-110 dBm
Adeunis	HP868	869.525MHz	16×32×2.8 mm	500 mW	-110 dBm
Adeunis	ARF54	869.4-869.65 MHz	42×20×4 mm	500 mW	-108 dBm
STE	BK77B5	432-436.1 MHz	95×50×7.5 mm	600 mW	-115 dBm @ .01 BER
STE	BK78B5	867-871 MHz	95×50×7.5 mm	600 mW	-115 dBm @ .01 BER
Coronis	Wavecard	433/868 MHz	37×30×7 mm	500 mW	-110 dBm
RF DataTech	LC450	406-475	78×52×20 mm	750 mW	-118 dBm @ 12 dB SINAD
RF DataTech	LC869	869-870 MHz	78×52×20 mm	750 mW	-118 dBm @ 12 dB SINAD
RF DataTech	LRT470-1	406-475 MHz	78×52×20 mm	750 mW	-118 dBm @ 12 dB SINAD
RF DataTech	LRT870-1	820-950 MHz	78×52×20 mm	750 mW	-118 dBm @ 12 dB SINAD
RF DataTech	LRM470TR-1/SPI	406-470 MHz	78×52×20 mm	1 W	unspecified
RF DataTech	LRM869TR-1/SPI	863-870 MHz	78×52×20 mm	1 W	unspecified
UDEA	UFM-A12WPA	869.4-869.6 MHz	70×33×8 mm	500 mW	-120 dBm
Hope RF	RFM12BP	433/868 MHz	20×40 mm	500 mW	-116 dBm @ .001 BER
Shenzhen HAC	HAC-LM12	433 MHz	53×38×10 mm	500 mW	-124 dBm
Radiocrafts	RC1280HP	869.4-869.65 MHz	19.5×60.5×6 mm	500 mW	-108 dBm
Digi	XBee-PRO 868	868 MHz	22×33×4 mm	315 mW	-112 dBm

Table D.1: List of radio modules that we considered using.

Device Name	**Data rate**	**Mod. scheme**	**Voltage**	$I_{TX,max}$	I_{RX}
BiM1H	max. 3 kbps	FSK	5.0 V	290 mA	8 mA
SHX1	max. 5 kbps	FSK	5.0 V	280 mA	20 mA
ENX1	max. 3 kbps	FSK	5.0 V	75 mA	12 mA
TR1M	max. 5 kbps	FSK	4.5-16 V	110 mA	27 mA
UHX1	max. 5 kbps	FSK	5.0 V	270 mA	24 mA
AMB8355	2.4-19.2 kbps	2-GFSK	7-30 V	530 mA @ 7 V	75 mA @ 7 V
TinyOne Pro 868	4.8-38.4 kbps	GFSK	3-3.6 V	600 mA @ 3.6 V	35 mA @ 3.6 V
PowerOne 868	9.6 kbps	GMSK	3.6 V	600 mA	55 mA
ARM-C8	19.2 kbps	GFSK	5 V / 3-3.3 V	350 mA	unspec.
HP868	10-57.6 kbps	GFSK	3-3.6 V	600 mA	18 mA
ARF54	0.6-57.6 kbps	FSK	3-3.6 V	700 mA	35 mA
BK77B5	1.2-9.6 kbps	GMSK	5.0 V	250 mA	50 mA
BK78B5	1.2-9.6 kbps	GMSK	5.0 V	250 mA	50 mA
Wavecard	4.8-19.6 kbps	GFSK	unspec.	450 mA	18 mA
LC450	max 4.8 kbps	any	5.0 V	350 mA	22 mA
LC869	max 4.8 kbps	any	5.0 V	350 mA	22 mA
LRT470-1	max 4.8 kbps	any	5.0 V	unspec.	22 mA
LRT870-1	max 4.8 kbps	any	5.0 V	unspec.	22 mA
LRM470TR-1/SPI	0.15-9.6 kbps	FSK/GMSK	5-9 V	unspec.	unspec.
LRM869TR-1/SPI	0.15-9.6 kbps	FSK/GMSK	5-9 V	unspec.	unspec.
UFM-A12WPA	4.8 kbps	FSK	3 V & 3.3-5 V	500 mA	50 mA
RFM12BP	0.6-115.2 kbps	FSK	2.2-3.8 V & 12 V	200/230 mA	20/23 mA
HAC-LM12	1.2 - 38.4 kbps	GFSK	4.75-5.5 V	400 mA	40 mA
RC1280HP	unspecified	FSK	3.2-4.2 V	600 mA @ 3.3 V	21 mA @ 3.3 V
XBee-PRO 868	24 kbps	unspecified	3.3 V	500 mA	65 mA

Table D.2: Additional specifications of the radio modules from Table D.1.