FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Location Detection in Smart Home Environments with the Oracle® Sun SPOT Technology

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur/in

im Rahmen des Studiums

Medieninformatik

eingereicht von

Eduard Rudolf Margulies
Matrikelnummer 0226250

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer/in: Univ. Prof. Mag. Dr. Schahram Dustdar
Mitwirkung: Univ. Lektor DI Dr. Manfred Siegl

Wien, 28.04.2013

(Unterschrift Verfasser/in) (Unterschrift Betreuer/in)

Ing. Eduard Margulies, BSc.
Fischagasse 7
2483 Ebreichsdorf

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien im April 2013
Ing. Eduard R. Margulies, BSc.

Contents

List of Figures

List of Tables

Abstract

The term smart-home has become very popular over the last years, gaining a great deal of importance in technological fields of research, especially in the field of Ambient Assisted Living (AAL). AAL represents an important branch of the Human Computer Interaction (HCI) research and focuses on supporting older and disabled persons in their everyday life. For disabled persons, for instance people in wheelchairs, the smart-home can offer increased support and an overall better experience in everyday life. Further, non-disabled persons could also benefit from the smart-home experience with many supportive functions for day to day living. The smart-home field of research covers a wide area of technological topics, one of these is location inference. Location plays an integral role in smart-home environments, thus many functions and actions depend on the user's current location in the home. For outdoor environments robust solutions like the American GPS for navigation systems, aGPS for smart-phones, or the European Galileo already exist. For indoor-environments such as smart-homes, only a few useable technologies are available. For high-resolution location inference i.e. high precision of the estimated current location, only expensive and complicated solutions are available. On the other hand, inexpensive approaches lead to only a rough location determination (low precision), thus not practicable for the smart-home purpose, where a high precision is required.
With this master thesis, we are interested in research on an easy-to-setup and fairly inexpensive approach for location determination systems. For that reason we use an embedded device from ORACLE®: the Sun SPOT. This device offers a variety of mechanisms which can be used for the location determination as, for example, the accelerometer as well as offering the ability for wireless communication based on the IEEE standard 802.15.4. First, we will show that our experiments with the accelerometer device yields imprecise localization results related to erroneous measurement-readings from the device's accelerometer-chip itself, thus not usable in indoor environments.
The second part of this thesis covers the location inference based on the wireless network communication of the Sun SPOT, which in turn is based on key performance indicators (KPIs) and the Weighted Centroid Localization (WCL) algorithm. We will show that this inexpensive and effective approach is suitable for roughly determining location, because of its imprecise measurement values and dealings with common issues in a wireless network, like interferences and reflections.

Überblick

Der Begriff des 'Smart-Homes' ist über die letzten Jahre hinweg sehr populär geworden und erlangte eine große Wichtigkeit bei diversen wissenschaftlichen Tätigkeitsfeldern der Technik. Insbesondere der Bereich des Ambient Assisted Living (AAL), ein wichtiger Forschungszweig der Human Computer Interaction (HCI), befasst sich mit der Unterstützung von älteren und eingeschränkten Personen im alltäglichen Leben. Das europäische Forschungsprojekt SM4ALL - smart homes for all - setzt neue Meilensteine in diesem Bereich. In dieser Diplomarbeit untersuchen wir einen wichtigen Teilbereich der 'Smart-Home' - Funktionalität, nämlich die der Bestimmung des Ortes. Für den Outdoor-Bereich gibt es bereits viele robuste und gute Lösungen, wie zB. das amerikanische GPS für Navigationssysteme, aGPS für Smartphones, oder das europäische Galileo. Für den Indoor-Bereich, wie in unserem Forschungsgebiet der 'Smart-Homes', gibt es jedoch nur eine geringe Anzahl an Möglichkeiten mit akzeptabler Genauigkeit und Fehleranfälligkeit. Für hochgradig-präzise Positionsbestimmung muss meistens mit sehr hohen Kosten, aufwendiger Wartung und aufwendigem Setup gerechnet werden. Auf der anderen Seite bieten billigere Lösungen keine akzeptable Genauigkeit sowie keine notwendige Robustheit an.
Wir untersuchen die Möglichkeit einer kostengünstigen und effizienten Positionsbestimmungslösung für den Indoor-Bereich der 'Smart-Homes'. Hierfür verwenden wir die Technologie von ORACLE®, nämlich den Sun SPOT, welcher unteranderem interessante Sensoren und Funktionen für diese Herausforderung anbietet. In dieser Arbeit wollen wir einen Überblick über bereits vorhandene Indoor-Varianten geben und danach zwei spezielle Positionsbestimmungsansätze, dem Accelerometer (Beschleunigungssensor) und dem Drahtloskommunikationsmodul (IEEE 802.15.4), mit dem Sun SPOT untersuchen. Im ersten Teil der Arbeit zeigen wir, dass die Experimente mit dem Beschleunigungssensor ungenaue Positionsbestimmungen liefert, dies auf fehlerhafte Messwerte vom Sensor selbst zurückzuführen ist und somit nicht für die Positionsbestimmung im Indoor-Bereich geeignet ist. Im zweiten Teil befassen wir uns mit der Positionsbestimmung basierend auf der Drahtloskommunikation des Sun SPOTs. Mit Hilfe von Key Performance Indicators (KPIs) und dem Weighted Centroid Localization (WCL) Algorithmus, werden wir zeigen, dass sich dieser Ansatz nur für eine grobe Bestimmung der Position eignet, was auf ungenaue Messwerte des Sensors und die Schwierigkeiten in einem Drahtlosnetzwerk, wie Interferenzen und Reflexionen, zurückzuführen ist.

Acknowledgements

Für meine Liebsten ...

Ich möchte diesen Absatz bewusst nicht kurz und bündig halten sondern ein wenig genauer eingehen.
An dieser Stelle möchte ich meinen tiefen Dank und Respekt all jenen zusprechen die mich während, vor und nach dieser Diplomarbeit unterstützt und motiviert haben. Bis zum Vorliegen dieses Werkes hatte ich so einige, unteranderem persönliche, Herausforderungen bewältigen müssen.
Einen besonderen Dank möchte ich natürlich meinen Eltern aussprechen, die mich während meines gesamten Studiums voll und ganz üunterstützt haben!
Auch ein besonderer Dank gilt Dr. Manfred Siegl, der mich nicht nur während der gesamten Diplomarbeit sowie Bakkalaureatsarbeit üunterstützt hat, sondern mir auch immer mit gutem Rat im alltäglichen Leben zur Seite stand und steht.
Meinem sehr guten, geschätzten und langjährigen Freund Markus Toman, der mich inspiriert, motiviert und auf diversen Ebenen zum Nachdenken angeregt hat, möchte ich auch hier einen großen Dank aussprechen!
Abschließend nochmals Danke, ihr alle habt einen Beitrag zu diesem Werk geleistet!

To my family and friends ...

This part of a paper is usually kept very short and is often full of standard phrases, but I would like to explain myself in more detail. My thanks go to all, who have supported and inspired me in my Master's Thesis. By the time I wrote the final version, I had had several challenges, including personal ones, to conquer, which life often puts before us. Many thanks to my parents who have always supported me during my studies, thank you for your patience and unconditional help and love.
But I also want to thank Dr. Manfred Siegl, who gave me great support throughout my university career, starting from my Bachelor's Thesis to this Master's Thesis. He not only gave me vital hints for my work, but also for life. Another special thank goes to my honorable friend Markus Toman, who always inspired, motivated and pushed me to think further.
Many thanks to everyone! You all played an important role in this thesis!

Conventions

Throughout this thesis we use the following conventions:

Text conventions

We use a listing-style for functions, methods, classes and other programming elements:

Listing 1: doTask - method inside of the TOCommunicationServerService

```
1 public void doTask() {
2         leds[SENDING_INDICATOR_LED_INDEX].setRGB(0, 250, 250);
3         leds[SENDING_INDICATOR_LED_INDEX].setOn();
4         try {
5             // packet structure:
6             // headerinfo;starttime;samplesperpacket
7             startTime = System.currentTimeMillis();
8             currentPkt = xmitTORFSampler.newDataPacket(packetHdr[index]);
9                 ...
```

Source code and implementation symbols are written in typewriter-style text:

`myFunction(int parameter1, int parameter2)`

The whole thesis is written in American English.

Chapter 1

Introduction

We have all probably seen movies where a house can talk to its occupants or simply can open doors, turn on lights or even brew a coffee or do the laundry. For decades this vision has encouraged many scientists to develop smart technologies and devices to create smart-home environments. The SM4ALL European Union Project has now encountered this challenge for the first time. A symphony of numerous sensors, actuators and system components have to work together in order to support an occupant in everyday life.

This may range from simply opening a front door to recognizing an occupant's retina, face, or even voice while saying "Hello" to detecting if someone is experiencing a medical emergency. Not only can this create conveniences and higher security, it can additionally support disabled people, such as those in wheelchairs, by automatically opening the doors, preparing food, and letting in authorized people who come in the case of an emergency. A wide field of applications can be applied in smart-homes. The following list emphasizes just a few but integral abilities and

Figure 1.1: SM4ALL - Logo. Source: [SM4ALL, 2012].

/ or uses of a smart-home environment [Baldoni et al., 2009]:

- The user wants to enter the home. After verifying the user's identity, the door unlocks and the user can go inside.
- A disabled person falls from the wheelchair and is immobilized. The smart-home detects the incident and further comprehends the person's situation. It automatically places an emergency call to the hospital or a caretaker so the person in need of help receives immediate attention.
- No occupants are at home. The smart-home shuts down the electricity in certain areas to save energy consumption and to respectively improve the lifetime of electronic devices.
- An occupant usually gets up at 6 am and drinks a cup of coffee. The smart home becomes aware of this routine and automatically turns the coffee machine on to brew fresh coffee.
- The smart-home turns heating off between 8 am and 4 pm, because occupants are not in the house.

As mentioned above, one can see that the functions of a smart-home environment range from supportive activities to life-saving actions. The smart home evolution has a significant impact on everyday life experience, which can be especially beneficial for disabled persons.

A vital part of the smart home environment is location inference. Without knowing the location of the user in the home environment, many or even most of the functions and activities cannot be performed by our smart-home. For example, with actions such as 'turn on the light in the room where the user is located', the location plays an integral role. Further, opening doors for handicapped persons in a wheelchair and even the control of the electricity needs the location-inference component.

The location detection problem for outdoor environments has been examined in great detail and offers nowadays very precise solutions and systems like the GPS and aGPS. These systems are widely used and offer one major benefit: only one device such as a navigation system or a smartphone is needed. The other infrastructures such as GPS positioning satellites are provided by a GPS provider and there is no need for an initial user setup of the navigation system. Further, there is no need to perform complicated calibration tasks.

There has been much research performed in the last decade for supporting location detection indoors, but most of the solutions include a complicated and

expensive setup for the system. A vital requirement of the ultra-frequency system [Eckert et al., 2009] is to place beacons in a predefined grid (for example every 2 meters). Further, these components have to be registered and the position of every node has to be stored. Other solutions like the IR-Systems, which also use beacons for the positioning estimation, provide high precision, lack the complicated system setup, but are by highly affected by external light sources i.e. sunlight or room lights [Want et al., 1992]. We can define the following requirements for our indoor-location solution:

- Precision (in smart-homes we demand a precision of less or equal to 1m)
- Affordable setup
- Location detection performance of many detectable nodes
- Resistance against sources of interferences

1.1 Motivation and Goal

The SM4ALL Project targets the location-detection challenge with a so called hybrid-solution. Based on our evaluation period, we have derived a certain degree of precision we assume or require for the location-inference component. Our research has shown that a precision of equal or less than 1 m is vital for our inference. Due to the fact that the precision cannot be as precise as the state of the art technology [Eckert et al., 2009], we have to solve this problem with different location detection solutions, building a hybrid from different approaches to reach a certain degree of precision.

In our project we initially used two well-known, tested approaches for the location-inference. First of all, we used the radio-frequency-identification (RFID) approach and secondly a Video Tracking System (VTS) to satisfy our requirements. In the initial testing phase, this combination satisfied our requirements quite well at any time we knew the correct user-location based on room-id and X and Y coordinates. But, due to fact that the VTS could not detect orientation very well or simply didn't work properly if the occupant is obscured by an object (like a box), an additional system had to be developed.

In our specific case, we examined a technology from ORACLE®: the 'Small Programmable Object Technology' (Sun SPOT), which is an experimental testbed developed for scientists. It is a very small and handy device with many built-in sensors and with a radio-communication interface based on IEEE 802.15.4. To

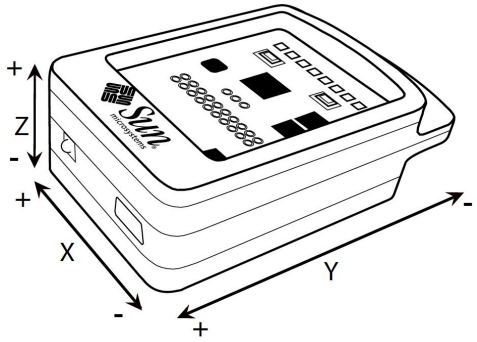

Figure 1.2: Sun SPOT built-in accelerometer. Illustration of acceleration-axes. Source: [Sun-Labs, 2009c]).

enhance our location-inference component we first examined and evaluated available indoor-location-detection systems. We have summarized the evaluated systems in the list bellow:

- Accelerometer approach with built-in Sun SPOT accelerometer sensors
- Radio-Frequency approach with built-in Sun SPOT IEEE 802.15.4 Transceiver

1.1.1 Accelerometer

An accelerometer is, as the name connotes, a device which can measure the current acceleration values based on the earth's gravity [Goldman, 2007b]. Nowadays these are found in modern smart-phones. It is often used to measure the current orientation of the mobile phone or to simulate gestures and moves.

With the laws of physics we are summing twice the acceleration-values over time and so estimate a certain user position while the user carries the Sun SPOT. The accelerometer approach is a very comfortable solution to set up in a smart-home environment. We need one mobile device carried by the user (with a built-in accelerometer sensor) and on the other hand just a few beacons, which receive and finally transmit the calculated positions to the core location-inference component.

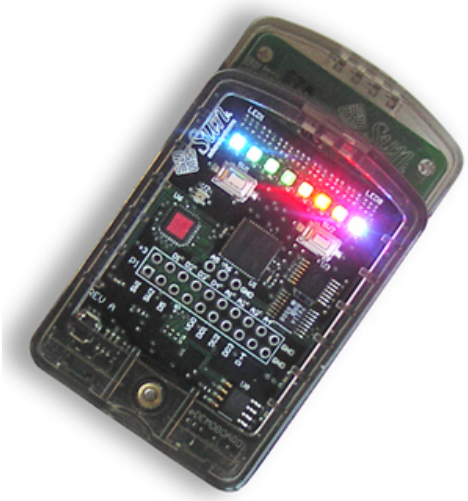

Figure 1.3: Sun SPOT from ORACLE. Source: [ORACLE, 2012b].

1.1.2 Radio-Frequency Approach

Based on WiFi-research from Microsoft RADAR [Bahl and Padmanabhan, 2000a, Bahl and Padmanabhan, 2000b] and Weighted Centroid Localization (WCL) [Behnke and Timmermann, 2008, Blumenthal et al., 2007] we evaluated a radio-frequency based location inference model. The Sun SPOT has a built-in IEEE 802.15.4 transceiver with an approximate range of 10 m in diameter. We evaluated the same principles pointed out in scientific papers [Behnke and Timmermann, 2008, Blumenthal et al., 2005, Blumenthal et al., 2007] to adapt these basic approaches and create a new system for location detection with small devices. This approach uses the received-signal-strength indicator (RSSi) and the link quality identifier (LQI) to estimate the distance between a sending and receiving device. The theory behind this is that a signal-propagation model can be created and used for distance estimation. In our research we used a theoretical signal-propagation model to measure incoming signal-strengths and estimate the distance from a given base station. So we say if a signal strength $RSSi$ is measured, the device has to be for instance $5m$ from the receiver. Due to the fact that we use a certain amount of base stations for a given location or section, we create a weighted sum to estimate the X and Y position of the mobile device. This is done by the aforementioned WCL algorithm. Further orientation information from the Sun SPOT built-in accelerometer can also be transmitted.

1.2 Thesis Organization

Chapter 1 covers the introduction, a brief problem description and the motivation for this thesis. The **second chapter** covers the technology employed and the environmental definitions, which mostly originated in the SM4ALL project [Baldoni et al., 2009]. **Chapter 3** covers the location detection approach with the built-in accelerometer in the Sun SPOT device. The physical background, the implementation and finally the feasibility are described in this chapter. The **fourth chapter** covers the radiofrequency approach with detailed evaluations on the experiments and the description of the software and localization model. The **fifth and final chapter** summarizes the thesis, points out some difficulties which should be avoided in the future, and provides topics for future research.

Chapter 2

Used Technology and Environmental Definitions

Due to the fact this project [Baldoni et al., 2009] is organized across different nations it was vital to pinpoint all requirements and environmental conditions. Further, accordance among all used technologies was made to ensure full compatibility with the core system and the communication and interaction among the devices such as sensors, actuators and others.

2.1 Requirements and considerations

The whole SM4ALL Project is going to be run in a domotic domain [Baldoni et al., 2009]. Basically we assume that the input for the system will be provided by the user, either directly or indirectly (triggered). The expected result of such an input is an automatically execution of actions/events which will be performed by the smart home. So we can summarize that after a user input a concrete house change is expected.
To fullfil this goal the system engages all devices spread around in the house. A key requirement we consider, is that this communication and interaction has to be as most transparent as possible to the user. The whole system is build in a service-based fashion, so that automatic engines may coordinate the single services to accomplish complex plans.

As mentioned in the introduction chapter the smart-home targets a wide range of people with an emphasis on the disabled, whose usability and accessibility are key

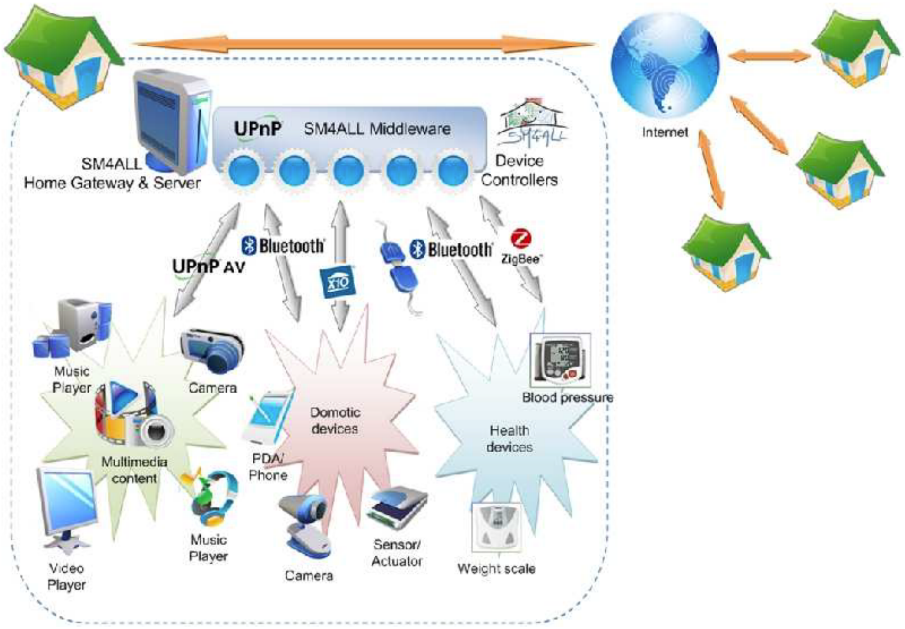

Figure 2.1: SM4ALL Network Architecture. Source: [Baldoni et al., 2009].

requirements. Robustness is another key word often used in service-oriented architectures to ensure a stable and all-time-up experience for the user. For the disabled, a smart-home environment means a new life experience regarding domestic interaction on a independent basis. Imagine a person in a wheelchair unable to open doors or even turn on lights. This new environment assists the disabled person perfectly, enabling a new level of independence.

To ensure robustness, the system has to ensure that failing services recover automatically, or if this is not possible that they offer alternative tasks to the user or even find other strategies to accomplish certain tasks.

2.2 Small Programmable Object Technology - SPOT

For our research based on mobile device technology, we used a fairly new and innovative device, called SunSPOT. The abbreviation 'SPOT' stands for 'small programmable object technology' and describes exactly what this device is. The Sun SPOT was created by Sun Microsystems (now ORACLE) in 2008 and has evolved to a device used by scientists and hobbyists to implement their technical experiments, such as creating a robot controlling device, completing sensor-network measuring

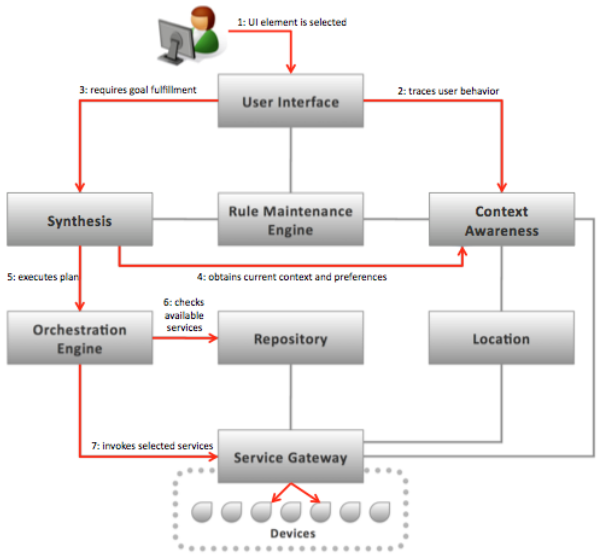

Figure 2.2: SM4ALL Component Architecture. Source: [Baldoni et al., 2009].

data, creating remotes, to name just a few examples.

Sun Microsystems introduced the term "The Internet of Things" which states that every single device can be interconnected with other devices. All devices can exchange data and information among themselves.

The key advantages of this tiny Sun SPOT device are that it has a fairly good computation performance: it is easy to create applications due to the fact that it uses a *Java virtual machine (JVM)*, it has a good basic set of sensors and finally has a radio communication interface installed.

In the past, to program mobile or small devices one had to have very specific knowledge about the internal hardware of such a device. What's more, special programming skills (e.g. C, objective C) were required to carry out a project. In the past, these were great barriers for ordinary *application programmers* and discouraged them from examining embedded devices. With the Sun SPOT, there was a breakthrough for application programmers and, of course, embedded device programmers when object-oriented programming language like JAVA was realized. JAVA has many advantages over C or objective C in terms of code-reuse, code-management and accessibility to the device internals like sensors and actuators.

Programmatic access to internal hardware, like accessing a LED-light or simply reading the current temperature from the temperature-sensor has been

Figure 2.3: Front View of a New Sun SPOT with eDEMO Daughterboard. Source: [ORACLE, 2012b].

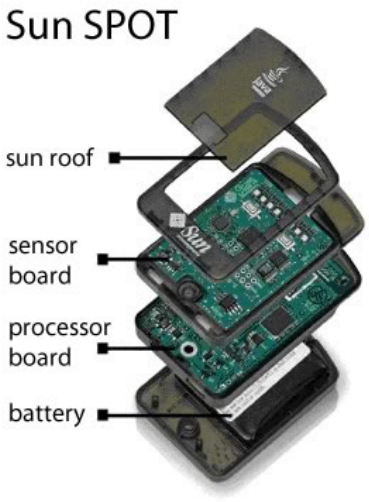

Figure 2.4: Exploded View of the Sun SPOT Device. Source: [ORACLE, 2012b].

Figure 2.5: A Wide Field of Sun SPOT Applications: Robots, Remote Controls and Cybernetic Devices such as an Electronic Glove. Source: [ORACLE, 2012b].

encapsulated in very easy-to-use API classes.

To summarize, applications and scenarios can be developed very easily, quickly and comfortably in an object-oriented way without detailed knowledge of hardware used in this embedded device, which also is indeed not necessary to know from an application programmers point of view [Sun-Labs, 2008].

We also want to point out that even though the Sun SPOT has great advantages and may be very interesting to use in future projects, it is still at the experimental stage. So there is no special commercial support for these devices and this can restrict some projects. The Sun SPOT is often used for 'rapid prototyping', and can, for example, easily create a prototype application to prove a concept.

2.2.1 Vision

The vision of the Sun SPOT project is simple and yet it is amazing to create an "Internet of Things" as mentioned earlier. Due to the fact that the Sun SPOTs are equipped with a wireless communication adapter, they can constantly exchange information among themselves and thus imitate an intelligent swarm of electronical devices or 'swarm-intelligence'. ORACLE (former Sun Microsystems) has provided a simple and convenient development platform to create applications for these embedded devices.

2.2.2 Fields of Application

These new possibilities provide new fields of application not only for ordinary embedded device programmers, but also for application programmers who are not used to or familiar with embedded hardware programming. The following list contains some very important and interesting fields of application for the Sun SPOT:

- Swarm intelligence projects
- Wireless sensor networks
- Rapid prototyping
- Educational and academic projects
- Industrial research
- Hobbyist projects and competitions

More fields of application and a detailed description of new projects are provided on `http://sunspotworld.com/media.html` (read on 20th of June 2012).

2.2.3 Hardware Overview

The Sun SPOT is a *embedded* device with a wide range of applications. It is usually used in wireless sensor nodes, for example, for collecting data or fulfilling certain tasks. The Sun SPOT Developer Kit ships with 2 ordinary Sun SPOTs and 1 Sun SPOT basestation (the Sun SPOT basestation will be referenced in future as basestation or bs). The difference between these two devices is that the bases station only contains the mainboard of the Sun SPOT device called eSPOT and has no battery, thus has to be connected to a PC or computer. The Sun SPOT device as is (not the basestation) contains, in addition to the eSPOT-board, a demonstration board, called the eDEMO board with additional sensors and functionalities we will discuss later. Further, the Sun SPOT has a built-in lithium-ion prismatic battery for use without power. So the following list shows the two types of Sun SPOTs avail-

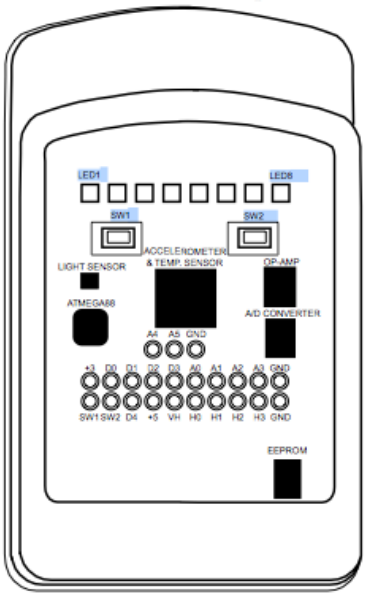

Figure 2.6: Basic Sun SPOT Blue-Print (front view of eDEMO board, LEDs, switches and other sensors). Source: [Sun-Labs, 2009c].

able:

- Sun SPOT Basestation, device with eSPOT board and has no battery (for connection between other Sun SPOTs and the PC or computer)
- Sun SPOT, device with main eSPOT and eDEMO board and battery

For the communication between the different Sun SPOTs located in a wireless sensor network, the SunSPOT uses a IEEE 802.15.4 radio controller based on a Texas Instruments Chipcon CC2420 RF transceiver for the radio-communication [Texas-Instruments, 2007, Sun-Labs, 2009c].

The eDEMO board offers a very easy and convenient way for rapid prototyping and builds applications, which interact with the real world (like a robot controlled by the Sun SPOT etc.). The eDEMO board comes with 8 tricolor LEDs, two push buttons, a temperature-sensor, an ambient light sensor, a 3-axis accelerometer, six analog input pads, four high-current high-voltage output pads, and five general I/O pads [Sun-Labs, 2009c].

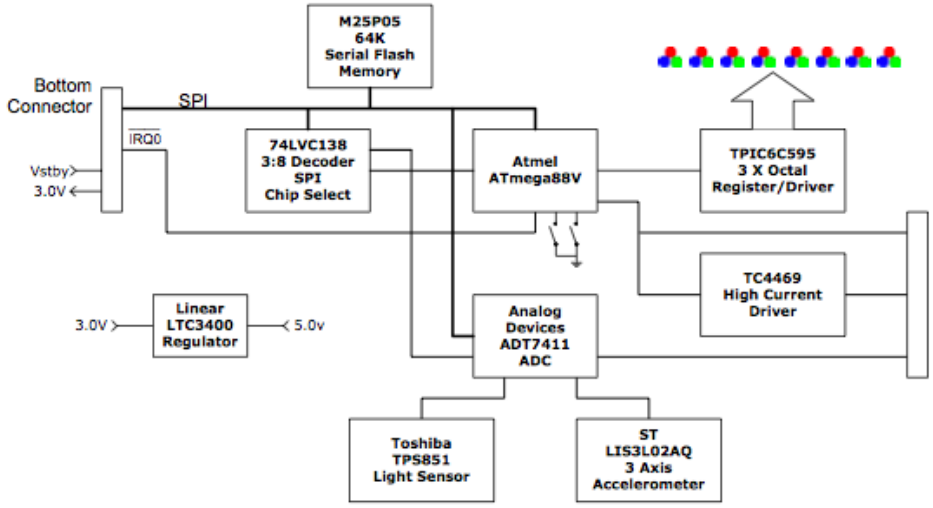

Figure 2.7: Detailed View of the Sun SPOT eDEMO Daughterboard. Source: [Sun-Labs, 2009c].

2.2.4 eDEMO Board

The eDEMO board is an extension or daughter-board for experimentation. It allows rapid-prototyping for many fields of applications. It already contains vital sensors, LEDs and push-buttons to realize common scenarios at ease. Note that the eDEMO board is only available on the ordinary Sun SPOT device and not on the basestation. Figure 2.7 depicts the detailed architecture of the eDEMO board.

The eDEMO-board is a good example of the wide possibilities which can be realized with the Sun SPOT-device. We will not examine the components in great detail, but for the interested reader please refer to the Sun SPOT-Theory of Operation guide in the actual version[1]. This guide provides a good and very detailed explanation of all Sun SPOT components. The following list contains the vital parts of the eDEMO board:

- ATMEL Atmega88 Processor
- M25P05 64K Serial Flash Memory
- 8 Tricolor (red-green-blue) LEDs
- 2 SPST normally-open, momentary tactile push-buttons

[1] http://sunspotworld.com/docs/index.html, read on 26th of June 2012

- GPIO, general purpose digital I/O lines (support for both input or output, I2C-DATA, I2C-CLOCK)
- High Current Driver
- Analog to Digital Conversion, ADT7411 ADC
- Temperature Sensor (-40 to +125 degrees Celsius, accuracy of 0.25 degrees Celsius)
- Accelerometer, ST Microsystems 3-Axis 2g/6g Inertial Sensor LIS3L02AQ
- Light Sensor, Toshiba TPS851 light to voltage sensor

With the firmware of Atmega88 processor many functions come with the eDEMO-daughter board, for example: pulse width detection, tone generation, pulse width generation, pulse width modulation, servo controller and much more.

2.2.5 eSPOT Main Board

The main board of the Sun SPOT is the eSPOT board. The eSPOT main board consists of the following important parts:

- Main Processor Atmel AT91RM9200 system on a chip (SOC)
- Spansion S71PL032J40 Multi Chip Package consisting of a 4MByte NOR flash memory and 512KByte pseudo-static RAM (pSRAM)
- Power Circuit, the eSPOT board can be powered with any combination of rechargeable batteries, external voltage or the USB host
- 3.7V 720maH Rechargeable Lithium-ion Prismatic Cell Battery
- Wireless Radio Communication, integrated radio transceiver TI CC2420 (formerly Chipcon) which is IEEE 802.15.4 compliant
- Inverted-F Antenna tuned to 2450 MHz, characteristic input impedance of 115 Ohm

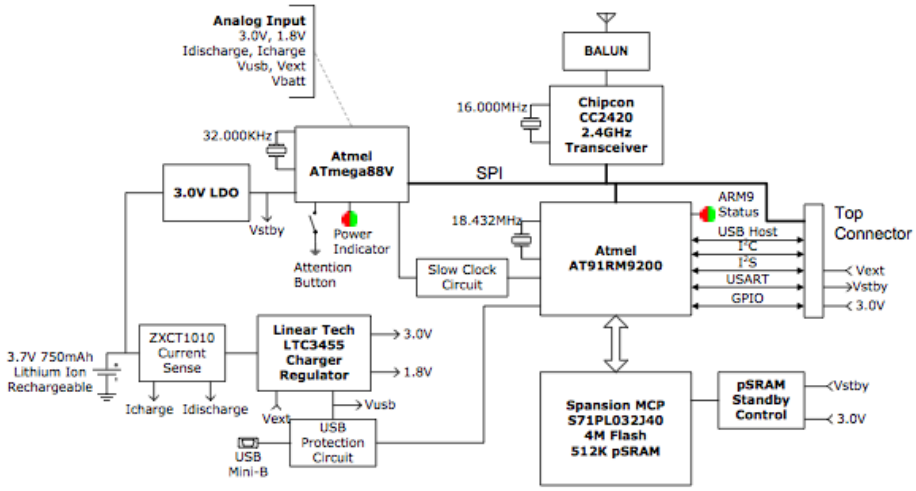

Figure 2.8: The Heart of the SunSPOT: The eSPOT Main Board. Source: [Sun-Labs, 2009c].

2.2.6 Java Virtual Machine Squawk

The Squawk Java Virtual Machine[2] is an open-source virtual machine developed mainly for small embedded devices (small, resource constrained devices). Compared to other commercial virtual machines (VMs) most of the Squawk VM code is written in Java. Other VMs are mostly written in, for example, C or assembler [Simon et al., 2006]. Due to the fact that the squawk VM is written in Java, the VM provides good portability, maintainability and ease of debugging. Further, the Squawk VM is Java compliant and is CLDC 1.1-compatible[3][Simon et al., 2006].
The architecture of the Squawk VM was inspired partly by the Squeak and Klein VM architectures and is mostly implemented in the language it executes (Java). The VM also runs on Solaris (SPARC and x86), Windows, MacOS X (PPC and x86) and Linux systems [Simon et al., 2006]. Figure 2.9 depicts the architecture of the Squawk VM compared to a standard Java VM.

2.3 Sun SPOT Service Architecture

The basic Sun SPOT architecture does not allow many applications running simultaneously on SunSPOT in a service-oriented manner. Of course the possibilities

[2]https://java.net/projects/squawk/pages/SquawkDevelopment, read on 2nd of July 2012
[3]http://www.oracle.com/technetwork/java/javame/java-me-overview-402920.html, read on 2nd of August 2012

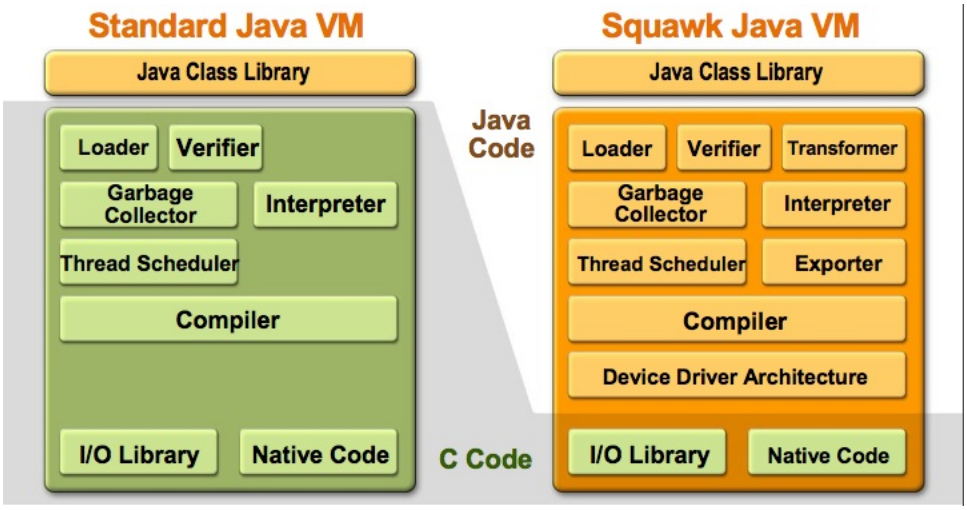

Figure 2.9: Comparison Between Squawk VM and Standard Java VM (Most code in the squawk VM is written in Java). Source: [ORACLE, 2012a].

for threading are available, but some service-oriented mechanism that allows distributed hosts or even other Sun SPOTs to use services are not possible out of the box.

On the basis of Ron Goldmann's service architecture (Spotlet) we are first able to locate services in our PAN (personal area network) and then, second of all, we can interact with these after receiving the credentials.

Commands and requests can be transferred easily via the radiogram connection. Only on that basis were we able to provide the location-inference components introduced in this thesis, which we found in the accelerometer approach (3) and in the radio frequency approach (4).

Chapter 3

The Accelerometer Approach

The accelerometer is a device often found nowadays in smart-phones and personal digital assistants (PDAs, also called handhelds). Most people use the accelerometer in their everyday life—without knowing it—when gaming, watching movies, and looking at photographs.
Basically the accelerometer measures the current acceleration on 3 axes (X, Y and Z) with respect to the earth's gravity ($g = 9.80665 m/s^2$). So if the user moves its device, the accelerometer is still aware of the current acceleration and, of course, orientation. Therefore many fields of applications can be satisfied with the presence of an accelerometer. We just want to mention a few scenarios:

- Detection of orientation, for detecting the reading direction of the device based on the position the device is being held.
- Detection of gestures, by summing acceleration and orientation over a certain period of time and performing pattern recognition.
- Detection of acceleration itself for several purposes.

Figure 3.1 shows a very common use case, where the accelerometer is used for orientation detection (and orientation changes over time). One can see that the square does not change its position with respect to the current view of the user. The application detects the change in orientation during the swing and re-arranges the layout.

As one can see, the accelerometer can enrich many applications in a everyday use. The most common example is the display of a photo or picture on a smart-phone or handheld. When the user turns the device, the accelerometer detects changing

Figure 3.1: Demonstration of the Accelerometer on a Smart-Phone.

orientation and displays the image, for example, in landscape format. Another example is the simulation of a steering-wheel with a smart-phone or handheld. One can turn the device to simulate the steering of a car and the game / application reacts to those movements. Therefore no traditional input devices like, for instance, a joystick are needed anymore.

For our approach we use the ability to simply measure the acceleration over time to estimate the position of the user in an indoor environment like a smart-home. We assume that the device is carried by the user all the time while he or she is at home. This device can simply be a handheld or smart-phone or even an especially designed device only for smart-home purpose.

In our experimental testbed we use the Sun SPOT for estimating the position information based on the accelerometer readings.

3.1 Technical Description and Capabilities

The Sun SPOT has an integrated accelerometer LIS3L02AQ built on the eDEMO board of the device [Goldman, 2007b]. The LIS3L02AQ is a low-power, three-axis linear accelerometer and is used to measure the current motion of the Sun SPOT. Further it can measure the orientation with respect to gravity. Figure 3.2 illustrates the basic axis-concept of the accelerometer.

In Figure 3.2 the plus sign (+) indicates that the acceleration vector measured at this axis increases, thus the value of the readings will be larger. Even though the Sun SPOT is not moving and laid down flat on the table (Z axis is pointing down), it experiences an acceleration of $1g$ along the positive Z axis and $0g$ along both the X and Y axis. So while gravity is pointing down (along the negative Z axis) this is equivalent to a uniform upwards acceleration of $1g$ according to the Einstein

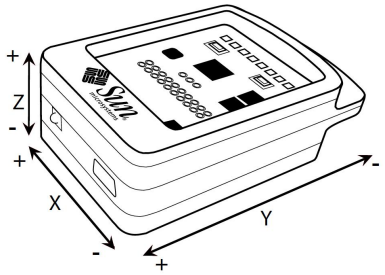

Figure 3.2: Sun SPOT LIS3L02AQ Accelerometer X,Y and Z Axes Arrangement. Source: [Goldman, 2007b].

equivalence principle (even if the Sun SPOT is NOT moving) [Goldman, 2007b].

To explain the technical background behind the accelerometer, one should know that the LIS3L02AQ consists of a Micro-Electro-Mechanical System (MEMS) sensor element. When a linear acceleration is applied on this sensor element, it will be displaced from its ordinary position and then cause an electrical imbalance that is read from eDEMO board's Analog/Digital Converter and provided to Sun SPOT API. The provided API converts the electronic readings into g-force units (which are more suitable for application programmers). The accelerometer can be scaled to a range of $\pm 2g$ and $\pm 6g$ [1].

3.2 Physical Background and Location Detection

Our main idea behind using the accelerometer as a location-detection device is that the position change over time can be calculated by the current acceleration an object experiences. So if a person is moving through space, an acceleration is applied on that person's body (or vice-versa, without accelerating the body with muscle power, the person can't move).

This acceleration can be measured by an electronic device, the accelerometer, because it is being carried by the person. The acceleration applies to the device and the person.

By the basic laws of physics, in theory, we should be able to derive the position at a certain measurement time. That means, if we measure the current acceleration of the device in equidistant positions, starting from an initial origin[2],

[1] For more detailed description see [Goldman, 2007b]

[2] An origin position p_0, like a house or room entrance.

then we are able to use this origin-position as reference and add the calculated distances to derive the current position.

The basic laws of physics state that the distance (Δx) between two times ($\Delta t = t_1 - t_0$) is the sum of all velocities ($v_i = v(t_i)$) in between. Equation 3.1 shows the mathematical form. In other words the distance-function ($\frac{\delta(x(t))}{\delta t}$) derived is the velocity ($v(t) = \frac{dx}{dt}$) [Husinsky, 2009].

$$\Delta x = \int_{t_0}^{t_1} v(t)dt = x(t_1) - x(t_0) \tag{3.1}$$

Further the acceleration is the function of the velocity derived ($\frac{\delta(v(t))}{\delta t}$) and we can define the integral of the acceleration over a given time with equation 3.2.

$$\int_{t_0}^{t_1} a(t)dt = v(t_1) - v(t_0), t_1 > t_0 \tag{3.2}$$

Based on these basic principles and laws of physics we should be able to determine the velocity $v(t)$ and of course the position $x(t)$ of a given object at any time t. For our calculation, we start from obtaining the acceleration $a_{x,y,z}$ at a given time t. To keep things easy we will explain the case for one acceleration reading a [3]. As we stated in equation 3.2, we are able to calculate the velocity for a certain time t as followed in equation 3.3.

$$\begin{aligned} &a(t) = \tfrac{\delta(v(t))}{\delta t} \rightarrow v(t_1) - v(t_0) = \int_{t_0}^{t_1} a(t)dt, t_1 > t_0 \\ &v(t_1) - v(t_0) = \int_{t_0}^{t_1} adt, a = const. \\ &v(t_1) = v(0) + a \cdot t_1, t_0 = 0 \end{aligned} \tag{3.3}$$

So we can calculate the velocity $v(t)$ based on the acceleration readings, assuming that the acceleration between the time-intervals is constant [Husinsky, 2009]. For calculating the distance-change over time, or rather the distance at a given time $x(t)$, we derive from our preceding assumptions and equation the following basis for our calculation in equation 3.4.

[3] The Sun SPOT device returns three acceleration-readings $a_{x,y,z}$ along all three axes

$$\begin{aligned} v(t) = \tfrac{\delta(x(t))}{\delta t} \rightarrow x(t_1) - x(t_0) &= \int_{t_0}^{t_1} v(0) + a \cdot t dt, t_1 > t_0 \\ x(t_1) - x(t_0) &= v(0) \cdot t + \tfrac{a \cdot t^2}{2} \Big|_{t_0}^{t_1} \\ x(t_1) &= x(0) + v(0) \cdot t_1 + \tfrac{a \cdot t_1^2}{2}, t_0 = 0 \end{aligned} \tag{3.4}$$

We have now analyzed the analytical-approach from deriving the position $x(t)$ at a given time based on a acceleration reading a. Again we assume a constant acceleration ($a = const.$) between 2 distinct times ($t_0, t_1, t_1 > t_0$) [4].

Due to the fact that integrating a function has to do with calculating the surface of the function (the surface under the function-curve), we can express the above mentioned equations as follows. We divide the curve of the velocity and of the distance into several small intervals Δt and calculate the rectangles of the function-value at a certain position $\Delta t_i \cdot V_i$. By choosing small enough time intervals Δt, we can calculate $v(t)$ and further $x(t)$ numerically. Equation 3.5 describes the calculation based on these assumptions [Husinsky, 2009].

$$\Delta x = x(t_2) - x(t_1) = \lim_{\Delta t \rightarrow 0} \left(\sum\nolimits_i V_i \Delta t_i \right) = \int_{t_1}^{t_2} v \cdot dt \tag{3.5}$$

As we noted in the above mentioned paragraphs, the Sun SPOT's accelerometer provides us the acceleration-value for all three axes at a given time t. So we are able to numerically determine the position of the Sun SPOT at any time based on an initial starting position x_0, an initial velocity v_0 and an initial time t_0. For all the three parameters, we use the value 0 (this is our so-called closure-mechanism). That means $x_0 = 0, v_0 = 0, t_0 = 0$. So we can solve above described equations easily. Again we are assuming that the acceleration between distinct two time-markers is constant. Based on these assumptions we derive our algorithm for calculating the position numerically based on "Euler's backward algorithm" (equation 3.6 shows our calculation basis for the algorithm) [Husinsky, 2009].

$$\begin{aligned} v_{n+1} &= v_n + a_n \Delta t_n \\ x_{n+1} &= x_n + v_{n+1} \Delta t_n \end{aligned} \tag{3.6}$$

Due to the fact that our main analysis and the feasibility studies are realized with

[4] Only if sufficient acceleration readings between a time-interval are available is this approach feasible. Later we will show that in our special approach this is the case, because the Sun SPOT accelerometer sampling rate is very high.

MATLAB[5], we have created a script `CrtAccSts.m` which takes an argument vector `vec` and calculates the velocities, positions and their sum. We will further examine our calculation model in a later chapter in more detail.

3.3 Conceptual Design

3.3.1 Overview

To describe the possibilities of the Sun SPOT's accelerometer device we have created three distinct experiments. First of all, we've placed the Sun SPOT on an even surface and moved it along 2 axes (y and x axis) to determine the positioning abilities along the two main axes. Due to the highly theoretical state of this experiment, we also wanted to test the accelerometer 's performance during real-use i.e. being carried by a person who moves a certain distance and then walks along a predefined track. Finally we've evaluated the performance of the accelerometer on a moving robot: the LEGO NXT 2.0. This robot moves on predefined tracks with constant velocity.

3.3.2 Experiment No. 1 - Axes-Movement

In the first experiment we created an axis-aligned grid (see figure 3.3) and moved the Sun SPOT while laying it flat on the back-side along the defined distance along the axis. One axis at one time (e.g. just $2cm$ along x-axis, $0cm$ along y-axis).

3.3.3 Experiment No. 2 - Person Carrying Sun SPOT

Our second experiment was more practical than experiment no.1 with an actual person carrying the Sun SPOT device, who had to move a predefined distance in a specified, grid-aligned, position. Of course we know that a person doesn't always move in grid-aligned way along an axis, but for reasons of experimental calculation we took only accelerations in that direction. For example, when we defined that the person moves $1m$ along the x-axis, we also obtained acceleration readings along the y-axis, but only included the x-accelerations in our calculation model.

[5]MATLAB is a computing environment from MathWorks® for solving numerical computations, http://www.mathworks.com

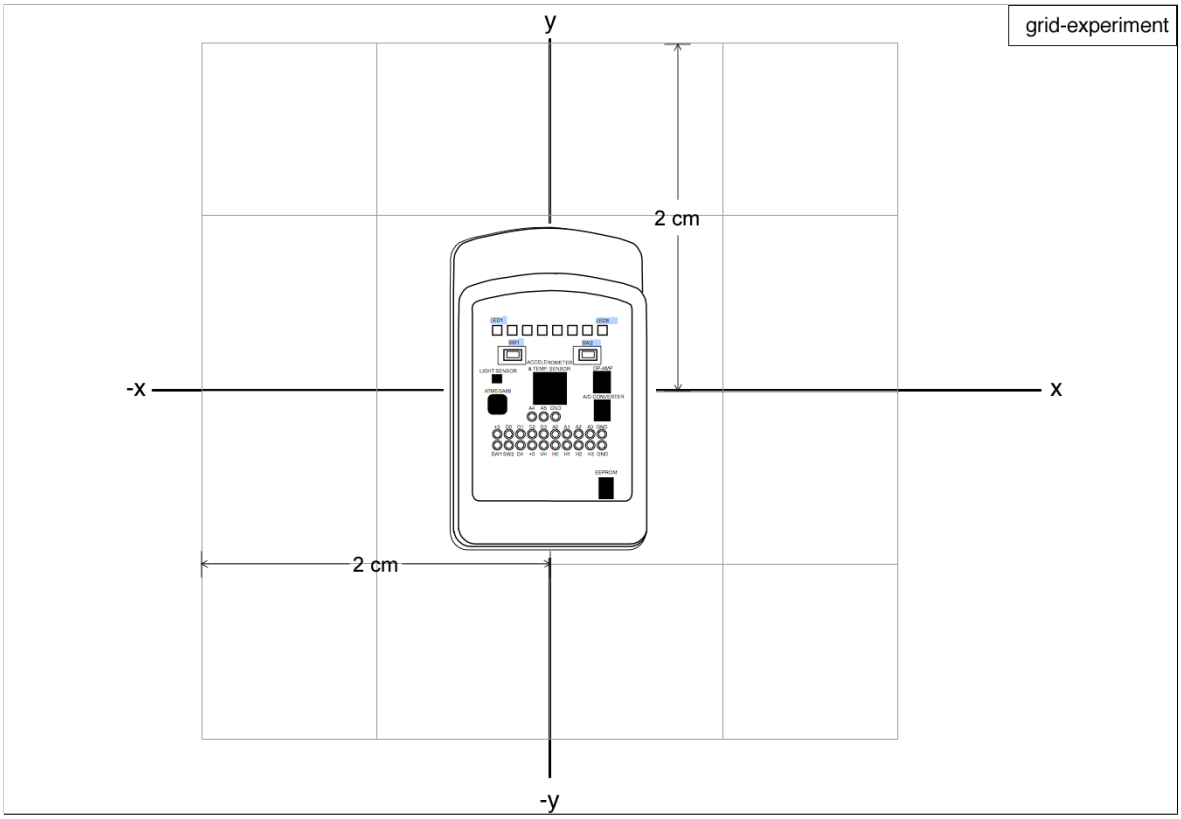

Figure 3.3: Sun SPOT Accelerometer Experiment No. 1 - Grid-Aligned Axes Movement.

3.3.4 Experiment No. 3 - Sun SPOT Mounted on a LEGO NXT 2.0 Robot

To simulate a uniform acceleration and a constant velocity we mounted the Sun SPOT device on a basic LEGO NXT 2.0[6] robot and had it move constantly to a target position along a defined route on a test-grid. In this case we measured the x and the y acceleration and included them in the calculation process.

3.4 Feasibility Study and Environmental Tests

This section covers all of the experiments we have performed and the detailed analysis of them.

[6] http://mindstorms.lego.com/en-us/default.aspx

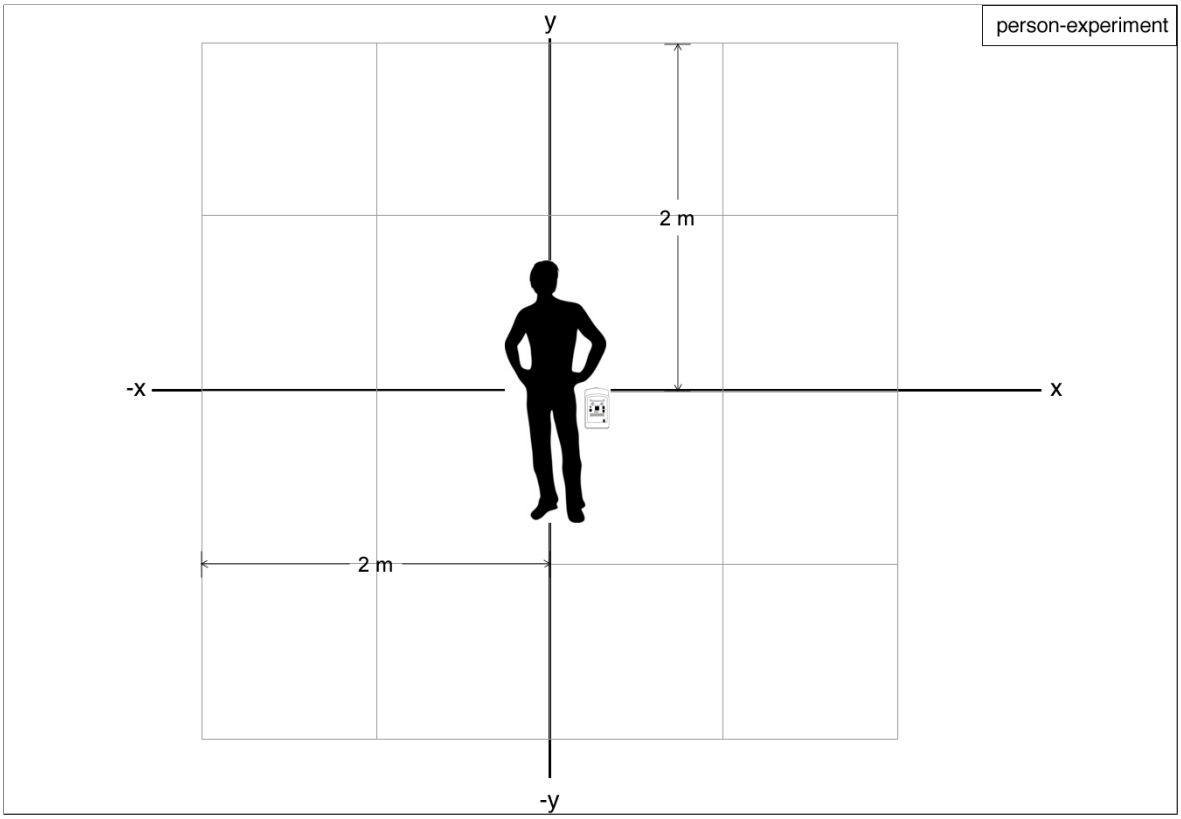

Figure 3.4: Sun SPOT Accelerometer Experiment No. 2 - Person Moving on Predefined Distance Along a Grid.

3.4.1 Consideration of Error-Readings and Noise

Every electronic device suffers from erroneous data readings and the effects of noise due to several interferences. Most of the time this noise can be handled very effectively with filters and a noise-correction algorithm.

The Sun SPOT's accelerometer also suffers from erroneous readings due to external influences/interferences. Figure 3.6 depicts a typical noise-matter which was recorded while the Sun SPOT was lying flat on it's back on an even surface. Almost $0.9s$ recorded, showing a constant noise-pattern. Only two peak-readings occured which did not fit in into the matter, occurring at $x = (0.5855, 0.5873)$. Besides those two peak-readings we filtered noise based on a 'noise-evaluating' phase to gather the noise-coefficients. Further, the Sun SPOT-accelerometer interface provided a mechanism to get zero-offsets and rest-offsets to even out noises, which worked well. To encounter noise or erroneous readings further we used an average-filter to even out peaks.

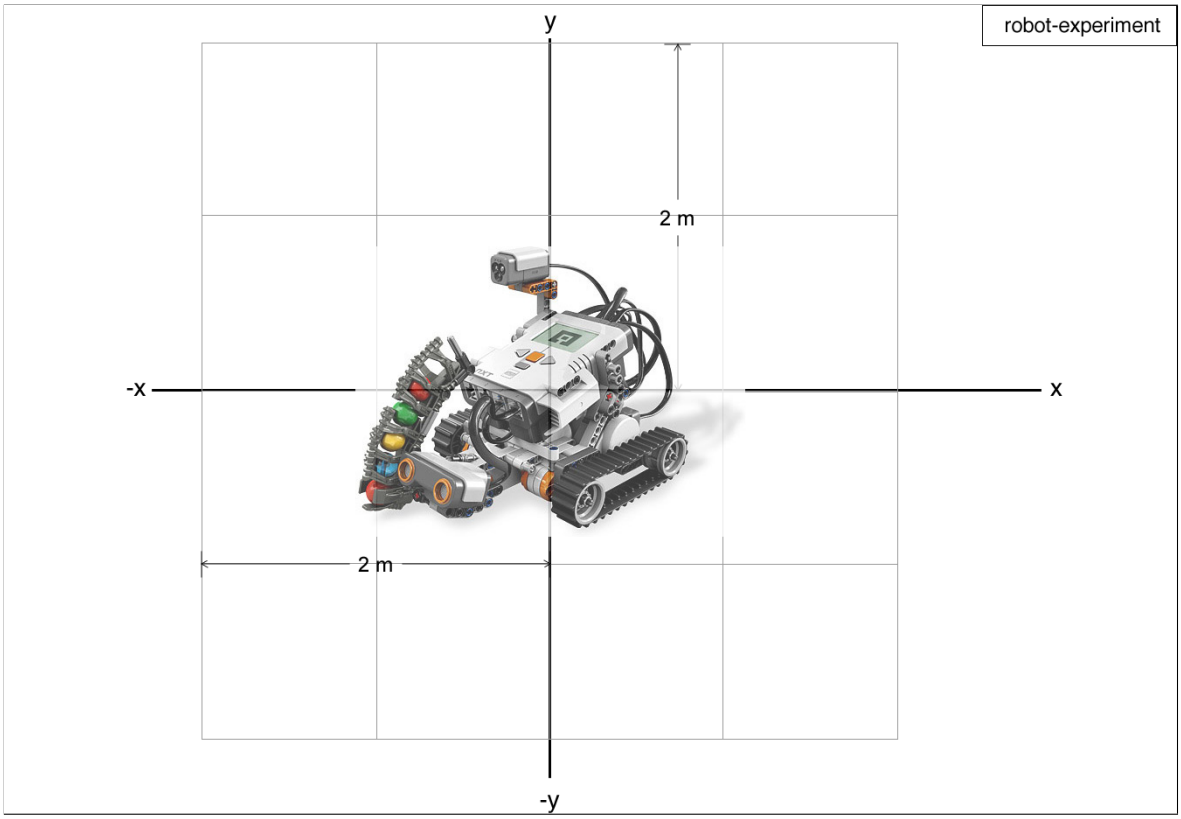

Figure 3.5: Sun SPOT Accelerometer Experiment No. 3 - Sun SPOT mounted on a LEGO NXT 2.0 Robot.

3.4.2 Sampling Rates

It is important to point out at which rate an application needs to read the current acceleration readings. The rate varies greatly and also depends on the type of accelerometer data being measured (for example whether it is possible to read raw acceleration values or the current orientation of the Sun SPOT). As pointed out in [Goldman, 2007b], experiments have shown that for measuring the orientation or for gesture-detection a sampling rate of 10-20 readings per second is more than sufficient. For measuring motion, for example if the Sun SPOT is mounted on a moving device, a sampling rate of 100 readings per second was more than adequate [Goldman, 2007b]. Based on the Nyquist-Shannon sampling theorem we can state that the sampling frequency needs to be greater than twice the signal bandwidth. For that we had to take our signal-bandwidth into consideration. We assume that people in a smart-home environment mainly move slowly and smoothly, unless, of course, higher accelerations occur, for example if a person falls down or has an emergency. But based on the analysis of [Goldman, 2007b] we

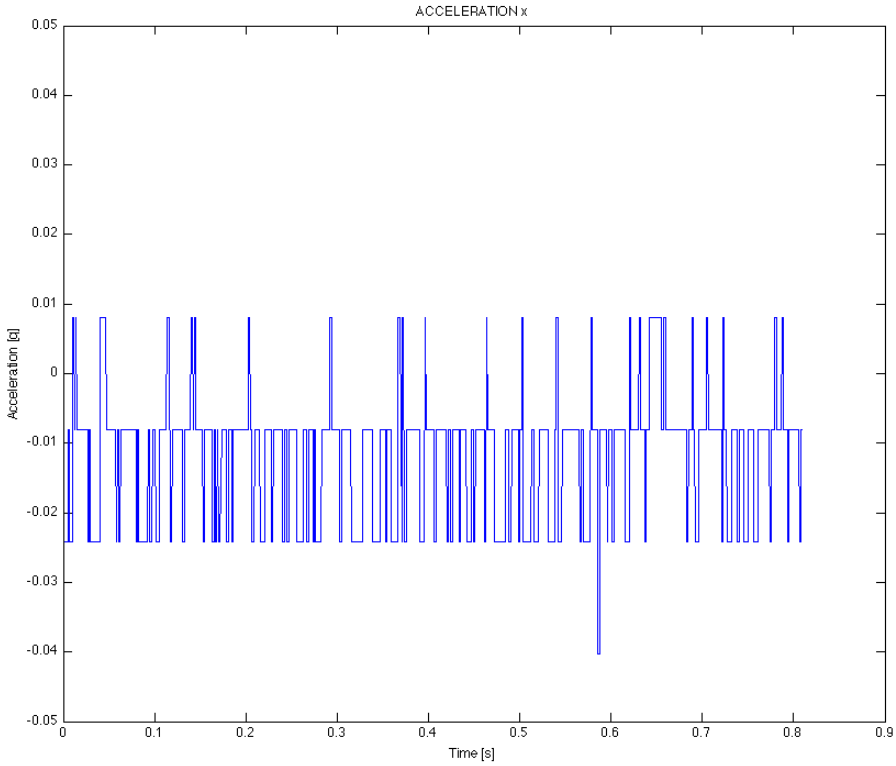

Figure 3.6: Sun SPOT Accelerometer Noise-Readings in the first $0.8s$.

defined a sampling rate of 100 readings per second to be sufficient for a person's smart-home environment movement pattern.

The exact description on sampling rates can be found in [Goldman, 2007b]. We only want to point out the important rates and values for our field of application. As stated in [Goldman, 2007b], a theoretical sampling rate of $320Hz$, or slightly higher, is possible to match up with a cutoff frequency of $160Hz$ which is sufficient for our case.

Figure 3.7 depicts the different time-deltas between successive readings. For faster evaluation we used the faster *AT91 Timer/Counters* to measure these minuscule time intervals. The mean time interval is $0.4509ms$ resulting in a sampling rate of maximum $2.2KHz$. But note that for these calculations almost 100% CPU performance was available for the calculation (no radio-transmission or calculation was performed).

To illustrate how fast the acceleration-reading from a register takes place we plotted the time-deltas over each reading.

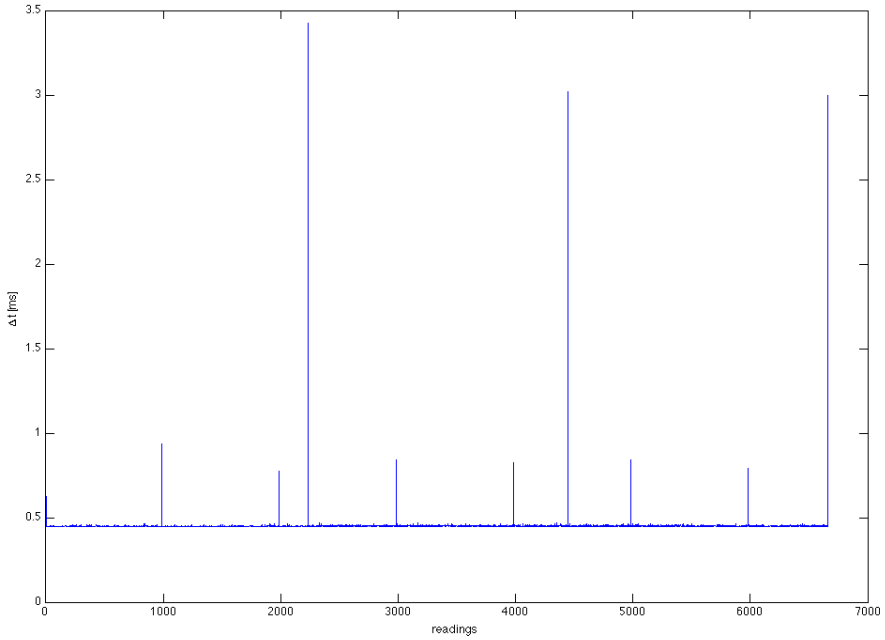

Figure 3.7: Sun SPOT Accelerometer Time-Deltas between Successive Readings.

Note the higher peaks, resulting from the garbage collector. There is a maximum time-delta of $3.425ms$.

3.4.3 Experiment No. 1 - Analysis

As described in the subsection 3.3.1 we have performed a movement on a grid with a specified distance. In our experiment we repeated the movement 10 times and took the mean value from that series. Before examining the overall-experiment results, we first want to further describe one experiment series and the problems we encountered.

The acceleration plot 3.8 shows a characteristic curve regarding our movement. Note that the noise is already removed from the curve. After a short still-standing-phase we performed a constant acceleration on the Sun SPOT device, pushing it forward to the desired distance mark, placed at $y = -6cm$ (along negative y-axis).

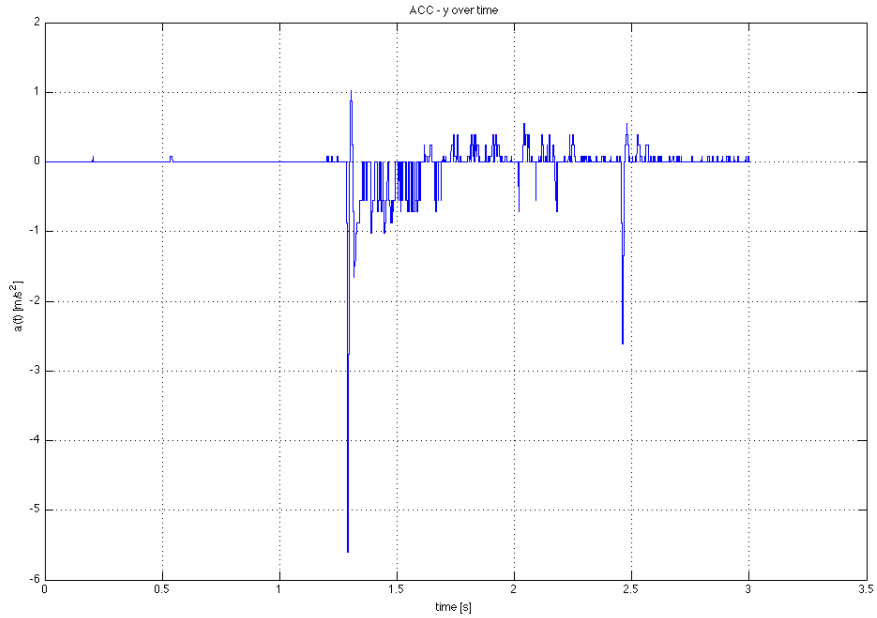

Figure 3.8: Experiment No. 1 Acceleration-Plot $a(t)$. Unit is $[m/s^2]$.

The highest peak indicates the initial acceleration we have performed, peaking at $a = -5.609m/s^2$ at position $t = 1.28s$. After a certain period of time we found gains in the negative direction until getting positive gains (except a few negative peaks, one high peak at $t = 2.45s$ with a value of $a = -2.6m/s^2$).

After evaluating the plot further we see that the acceleration is obviously *unbalanced*. We gave the device a positive impulse as we moved it forward to our target mark and then suddenly stopped as we reached that mark. The strongest initial impact is recorded by the highest negative peak as stated before, but there is no positive counter-part acceleration indicating the stop of the Sun SPOT device, though we found an unbalanced acceleration behavior.

This unbalanced behavior can be also found in the velocity - plot 3.9. After the initial impact the velocity drops very quickly to a peak value of $v(t) = -0.1498m/s$ at $t = 1.689s$ and slowly decreases but remaining altogether negative and still indicating a forward movement in the impulse direction. In theory, the velocity should have returned to zero, but that is not the case due to the fact that the acceleration is unbalanced. Further, the velocity remains constant as if the acceleration were con-

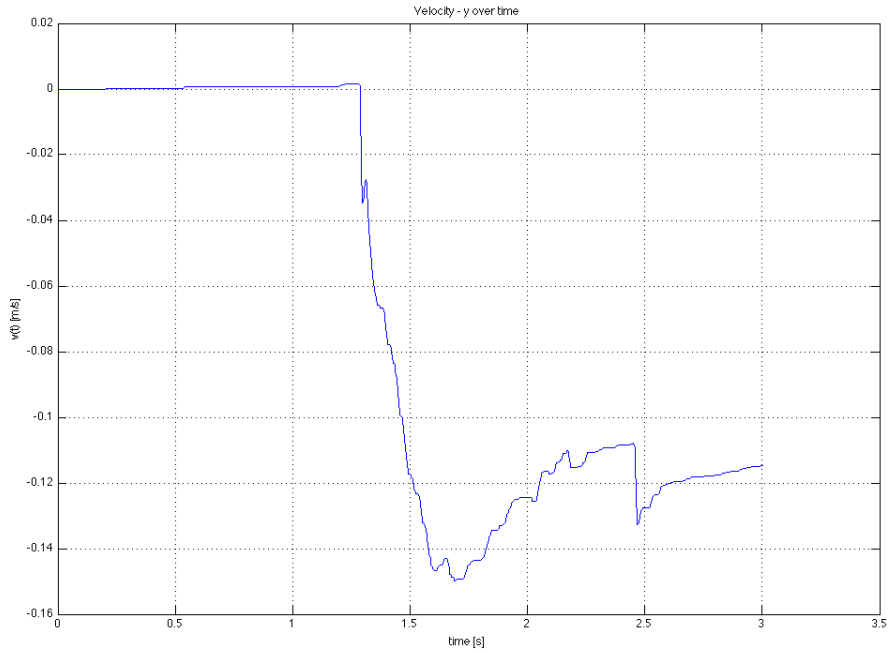

Figure 3.9: Experiment No. 1 - Velocity-Plot $v(t)$. Unit is $[m/s]$.

stant, resulting in a further developing error-rate in the distance calculation (the distance still increasing).

The distance-plot 3.10 also shows a linear increasing value after $t = 1.3s$ until it reaches $x(t) = -0.2m$ at $t = 3s$. Remember, we only moved the Sun SPOT $6cm$. That results in an error-rate of $6cm/20cm = 0.3$ or 300% error during $3s$ of the time passed. Taking this into account, further experiments are meaningless due to the rapidly increasing error after such a short period of time.

Table 3.1 shows the detailed experimental outcome of the above mentioned series. We performed every movement recorded in that table at least 10 times and took the average value. Figure 3.11 shows a plot of the acceleration-values from the experiment. Only 5 out of 10 data-curves are displayed in this plot. Note that the same experiment shows different acceleration-plots, making it very hard to find a common pattern (different accelerations found in the experimental test series are given different colors).

Further, we've performed this experiment with varying distances from $d = [0cm, 50cm]$ and encountered similar error-rates. Besides, note that in a real-time

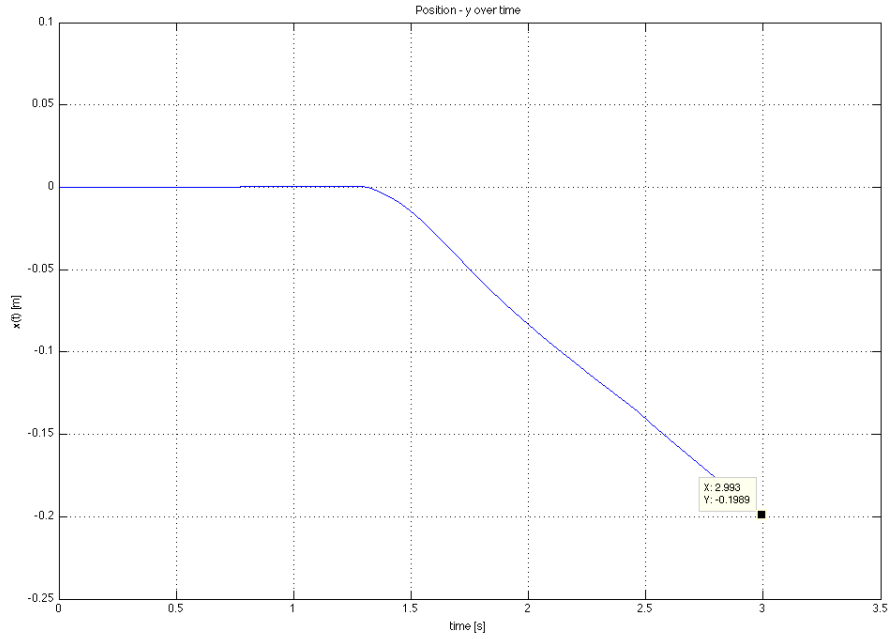

Figure 3.10: Experiment No. 1 - Position-Plot $x(t)$. Unit is $[m]$.

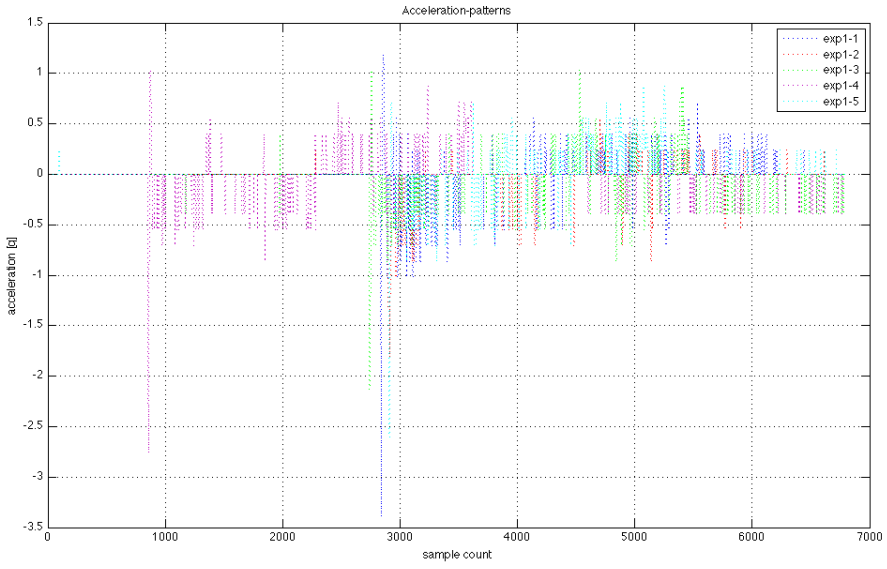

Figure 3.11: Experiment No. 1 - Same Experiment with Different Acceleration Characteristics.

	distance	mean	max	min
Movement y	-6cm	68.5%	267%	9%
Movement x	-6cm	71.67%	300%	15%

Table 3.1: Error-Rates for the First Accelerometer-Experiment.

calculation environment the error once gained will not remain constant, or be reset. Instead it will be propagated further in succeeding calculations. Imagine we perform the above-mentioned experiment but don't stop the calculation right after we pushed the device forward. In other words, first we start the calculation, then we apply the movement (for example $10cm$), stop moving the device any further, but let the calculation continue for some time. Due to the fact that the acceleration is and will stay unbalanced, the sum of the preceding accelerations will tend to a major positive or major negative acceleration. This effects the velocity which remains at a certain calculated value v_{last} yielding permanent position-changes in a certain direction, even if the device is not moving at all! Note that the velocity cannot return to $0m/s$, as theoretically should be the case. Hence, the velocity still has some value. We cannot say that if we do not receive any acceleration we will reset the velocity to $0m/s$, because an acceleration of $0m/s^2$ can also be a constant movement at a certain velocity (a body not receiving any acceleration can be standing or moving at a constant velocity). So at this time there is no solution for this error-propagation problem. We assume that this imbalance between the accelerations occurs because of the fact that some important values are read erroneously or were simply skipped by the electronic device. In both cases the correct acceleration will be missing in the acceleration-plot. We thus learned that only more accurate data can solve this problem.

Another issue we encountered is the fact that if the user is carrying an accelerometer, the device-axes are not always the same. Imagine the user is carrying this device like mobile phone. The axes of the devices compared to the absolute axes of the user can change due to various positions according to how the device is being held (perhaps carried in the pocket, or in the hand, etc.). Of course, we can calculate the tilt, based on the accelerations of the device and arranging the axes of the device to the absolute axes of the user, but note that this is a very time-consuming process and will lead to more error-rates.

3.4.4 Experiment No. 2 - Analysis

After the results from experiment no. 1, we did not expect a better outcome from the second experiment. So in this case we only point out, which experiments we performed and the error-rates we calculated.

In our experiments we let the user walk a distance of, for example $2m$ along a defined path on a grid. The Sun SPOT was held in the user's hand with the back of the Sun SPOT facing down and the antenna facing away from the user. The user was instructed to hold the device as uniform as possible, not moving it in any direction. We performed each test series 10 times. Table 3.2 shows the results of the second experiment.

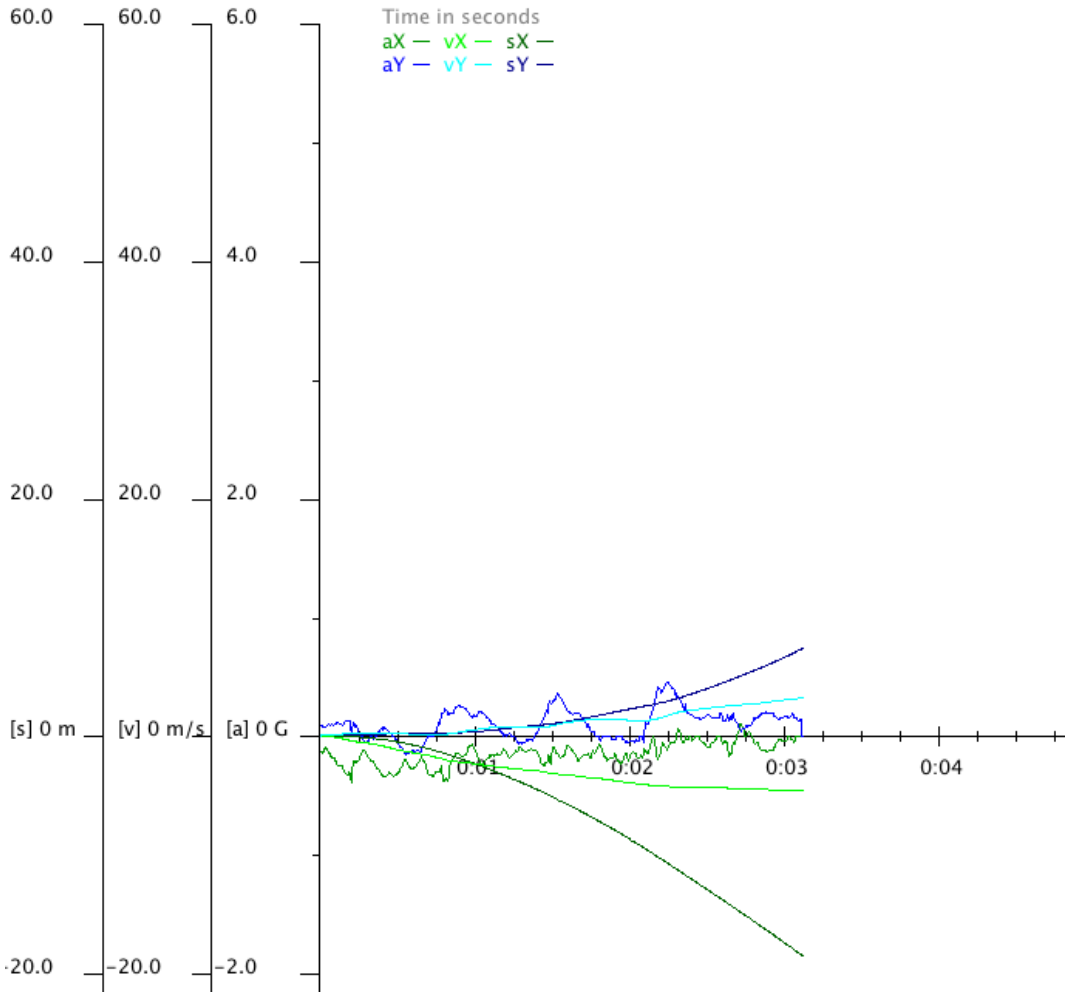

Figure 3.12: Experiment No. 2: A Walk along a Predefined Path, $2m$ in Negative y Direction

In figure 3.12, a plot of our software-prototype depicts a special case of the second experiment: a user had to walk forward on a predefined route along the negative y-axis $-2m$. The sinusoidal wave depicted in blue (aY) shows a person's typical

walking-pattern [7]. Further, we can see that the distance error (depicted in dark-blue) rises very quickly due to the error-propagation. In that case, we reached a total distance of $s_y(t_{end}) = x_y(t_{end}) = x_y(3s) = 7.3085m$ instead of the walked $-2m$, resulting in an error of $\Delta x = 9.3085m$. Also note that we didn't walk any distance along the x-direction, but also got a very high error-distance along x-axis (depicted in dark-green). This is due to the user experiencing an acceleration in the x-direction, while walking. In our special case, we got a calculated distance of $s_x(t_{end}) = x_x(t_{end}) = x_x(3s) = -18.5286m$. This is, as mentioned in the first experiment, an unacceptable result for a location inference component.

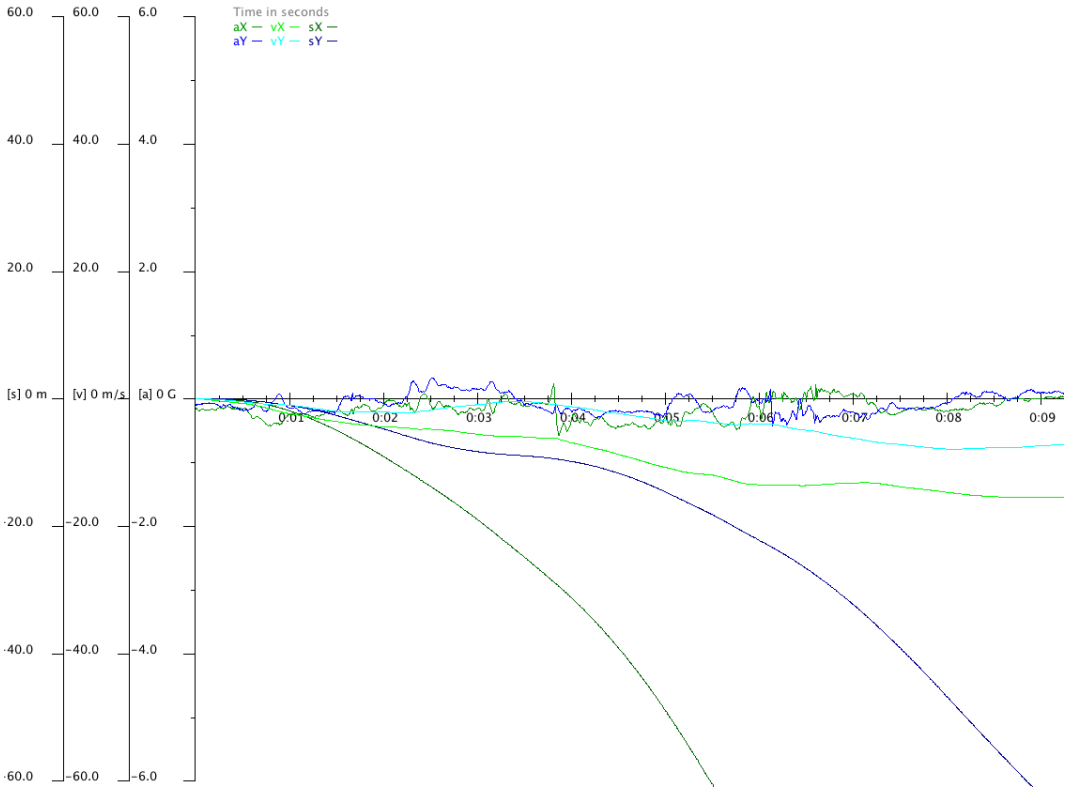

Figure 3.13: Experiment No. 2: A walk from a starting position to a defined mark and back to the starting position. The error's rise is exponentially fast. In this experiment, only $6m$ were walked in total.

Figure 3.13 depicts another case of the experiment. Here the user had to walk to a defined mark at $3m$ along the negative y-axis and had to walk back to the starting position.

Again we encountered a very great error-distance of $-73.0563m$. Theoreti-

[7] We will later have a short discussion on pattern-recognition with the accelerometer.

cally, the distance or rather the position should be zero ($-3m$ along negative y-axis and back $3m$ along positive y-axis result in $0m$).

	x_x	x_y	x_xcalc	x_ycalc	err_y
Testseries no.1	0m	-2m	-9.55m	4.96m	348%
Testseries no.3	0m	-2m	-9.32m	-1.21m	39.5%
Testseries no.8	0m	-2m	-13.55m	22.61m	1230.5%
Testseries no.9	0m	-2m	-9.64m	9.76m	588%
Testseries no.10	0m	-2m	-18.52m	7.30m	465%

Table 3.2: List of Some Results of the Test-Series from Experiment No.2: Walking $2m$ along Negative y-Axis.

As we can see in table 3.2, the results for the calculated distances vary greatly and have a mean error of $7.46m$ or 373.43%, again showing us that this approach does not represent a satisfying alternative for a location inference component.

3.4.5 Experiment No. 3 - Analysis

In the third experiment we wanted to examine the calculation-behavior of the Sun SPOT accelerometer if mounted on a moving device like a LEGO NXT 2.0 robot. The main idea was to derive a constant behavior and get more insight into the error produced by the Sun SPOT accelerometer.

Figure 3.14 illustrates an experiment recorded with our software-prototype where the NXT 2.0 robot moved $-60cm$ along the negative x-axis. The calculation yields a result of $-8m$, that is a total error of 1233.3%. As the experiments no.1 and no.2 sufficiently proved that the location inferences with the accelerometer are not satisfactory, so too does experiment no.3. We will not examine the error-rates in great detail, but error-rates around 500% to maximum errors of 1343.3% were very common. We found the pattern behind a robot movement interesting. Compared to the sinusoidal movement we found while walking, the constant movement of the robot shows a dizzy pattern with positive and negative accelerations varying very quickly over time. Figure 3.15 depicts that behavior. In theory, after an initial acceleration with a constant velocity, no further acceleration should occur till the final stop of the robot. But here, again and again, the robot experiences accelerations yielding an erroneous calculation result.

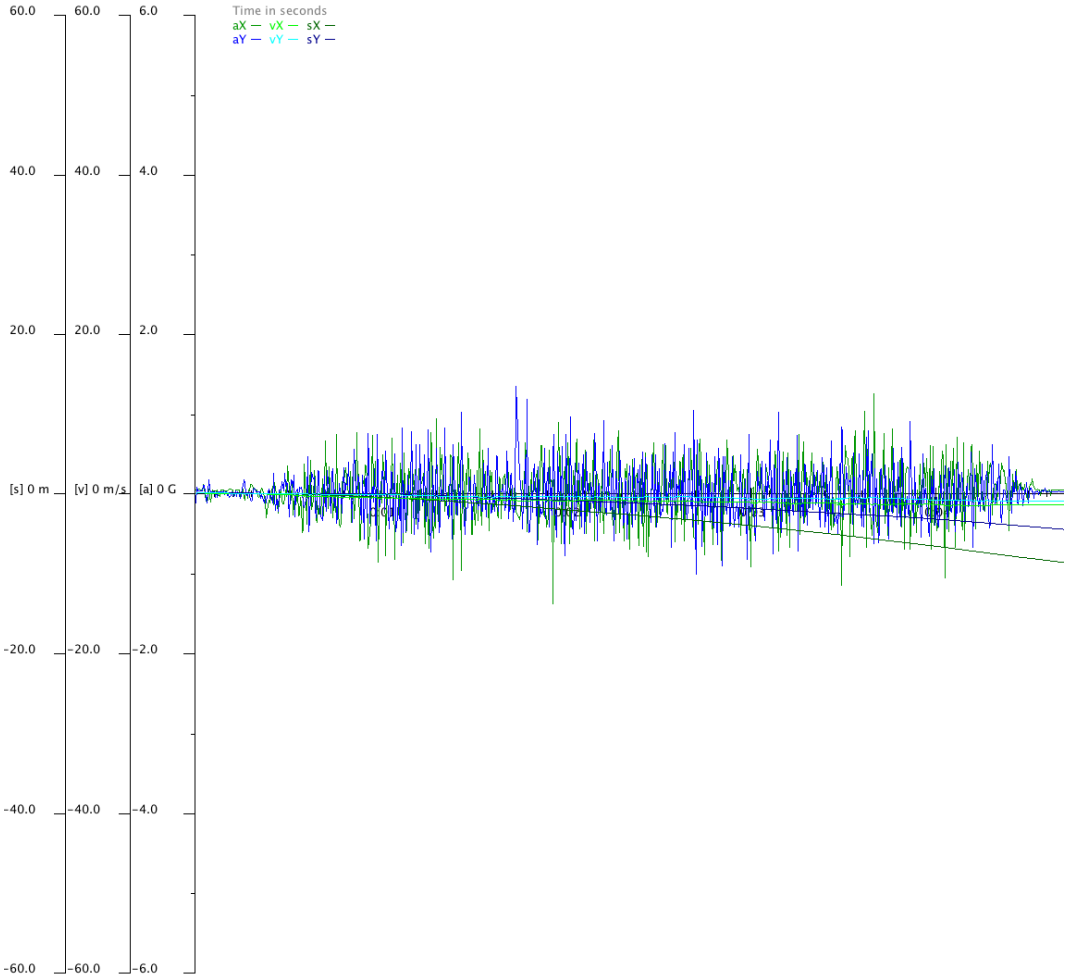

Figure 3.14: Experiment No. 3 – NXT 2.0 robot moving $-60cm$ along negative x-axis with the same velocity. This plot includes starting and stopping of the robot. As in the other preceding experiments, the error increases exponentially. In this experiment the robot moved $-60cm$ in total, resulting in a calculated distance of approx. $-8m$.

3.5 Software Prototype

For testing purposes and for simulation we created a software-prototype. As described in chapter 2 we created a software component based on the Sun SPOT service architecture, using the 'Telemetry framework' from Ron Goldman as our initial framework for carrying out communication with a single remote Sun SPOT and the host-system.

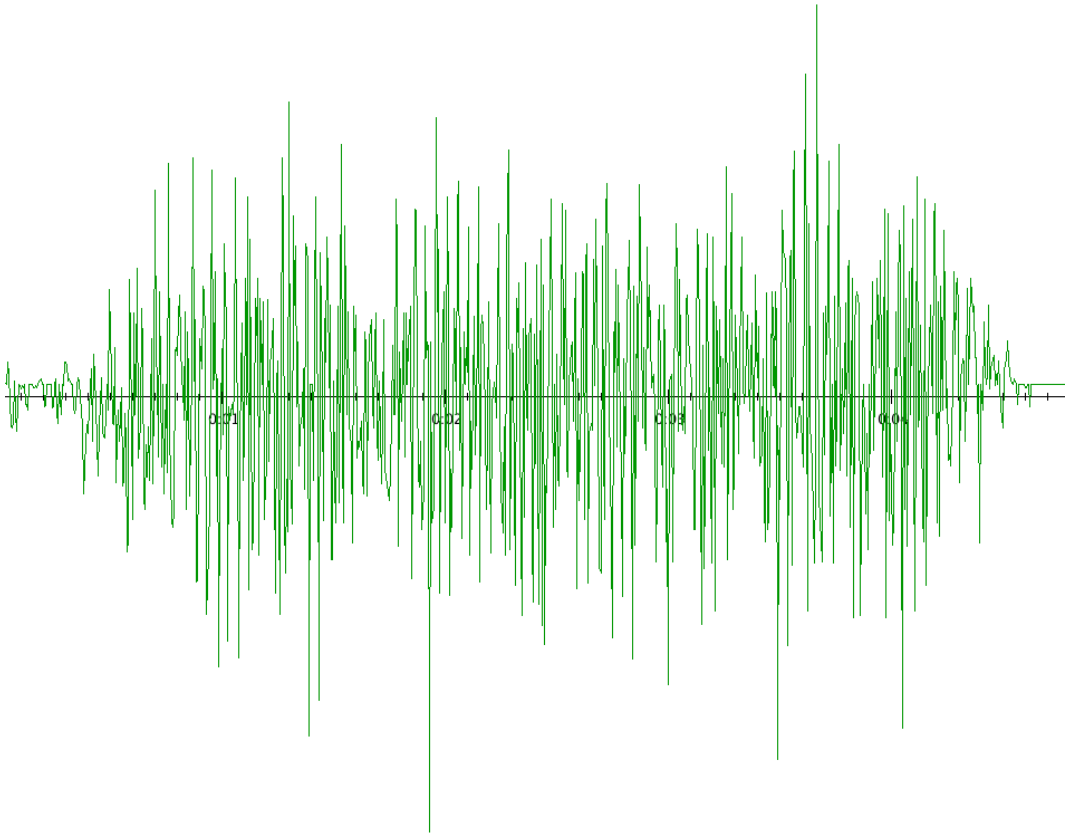

Figure 3.15: Experiment No. 3 - NXT 2.0 robot move-sequence zoomed-in. We find quickly varying components, even though the device moves at a constant velocity.

The main idea behind the software is that we have a core-system running on a host, which gathers incoming accelerometer readings and performs the distance calculation based on this reading. One reading represents a 4-dimensional feature vector $V_f = (a_x, a_y, a_z, \Delta t)$ containing the acceleration-readings along the three axes and the time-delta to the previous reading.

The Sun SPOT accelerometer service on the Sun SPOT device gathers a specified amount of readings together into one datagram packet until it sends the packet to the core-system. This increases the performance of the accelerometer-reading mechanism, because sending a radiogram packet takes at least $4ms$. During this sending period the reading-loop of the accelerometer is on hold, which may perhaps result in erroneous or skipped readings.

A few words about the process-chain: after the Sun SPOT has registered itself at the core system, the core-system client can initiate the gathering of accelerometer-readings from the Sun SPOT. After initializing and receiving the start-signal from the core-system, the Sun SPOT continuously sends the reading-packets to the core-system without performing any additional calculation. The core-system receives and records the readings and performs the location-inference algorithm. These results will be graphically displayed and the view can be manipulated by the GUI-elements (e.g. x-axis zoom or applying filters, etc.).

The user-interface provides a 2-dimensional graph for displaying the total acceleration a_{total}, the acceleration along three axes a_x, a_y, a_z, the velocity v_{total}, v_x, v_y, v_z, the position s_{total}, s_x, s_y, s_z and a control-panel for data-gathering, data-deletion/graph-clearing, saving and Sun SPOT control (LED-blinking, calibration of the accelerometer, etc.). The figure 3.16 depicts the user-interface of the Sun SPOT Accelerometer application in action.

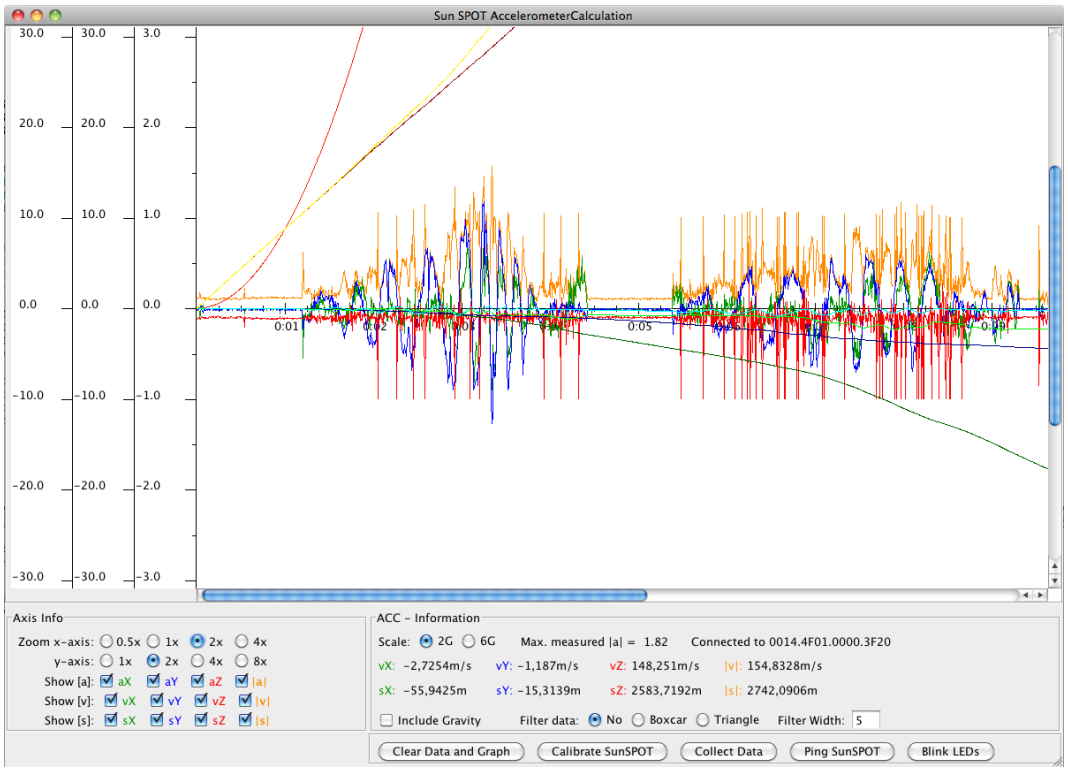

Figure 3.16: Sun SPOT Accelerometer Sample Application GUI.

One last note: this application was not designed for inclusion in a location inference component but only for prototyping purposes (analysis and performance evalua-

tion).

3.6 Conclusion

After developing the tools and software-prototype to perform the experiments and evaluation on the accelerometer approach, we found out that this approach is not usable for location-inference mechanisms. It is far too imprecise (error-rates around 300% or slightly less are very common).

Our experiments showed that some accelerations were not recognized by the accelerometer resulting in an unbalanced acceleration relationship (e.g. more negative accelerations than positive). An unbalanced acceleration relationship yields an incorrect velocity calculation and further false positions (error propagation). The error drastically increases to unacceptable error rates. We have tried to apply various filters and calculation methods to improve the calculation and have also increased the sampling-rate to $2.2KHz$, which should be more than sufficient for the smart-home purpose, but we could still not achieve a satisfying result.

Besides the error-rates, we have encountered other problems. For example, while the device is being carried by the user and the user suddenly changes the orientation of the device, the axes have to be realigned. That means further calculations regarding the actual tilt or orientation of the device have to take place at the same moment while evaluating the current acceleration readings. That would lead to a computational overhead which cannot be handled in real-time.

Nevertheless the accelerometer can still be used for some sort of gesture detection or for evaluating the current orientation of the carried device and many applications of smart-phones make use of this orientation nowadays. An additional use case could be a 'fall detection' system, which reacts if the person falls down.

There are many fields of application for the accelerometer, but unfortunately not for a location-inference component. Future work could examine the possibilities of a gesture or fall detection system and ways of including such components in a core-system.

Chapter 4

The Radio Frequency - Approach

The radio frequency approach tries to estimate the current user's position based on the 'signal strength' of the device being carried. The main idea behind this approach is to place strategic beacons around the environment, which periodically receive incoming messages from moving sensors. They forward the received messages to the core system, where the exact positions of these devices are then estimated.

As mentioned in the introduction (chapter 1), several methods have been developed in the past with WiFi-network technology such as the RADAR System from Microsoft [Bahl and Padmanabhan, 2000b]. These systems do not provide very accurate results for the position estimation and show an error distance too great for smart-home use. Further, it is not easy to implement a WiFi module on tiny devices without much battery consumption.

Our approach makes use of the IEEE 802.15.4 standard [Ergen, 2004], to be exact

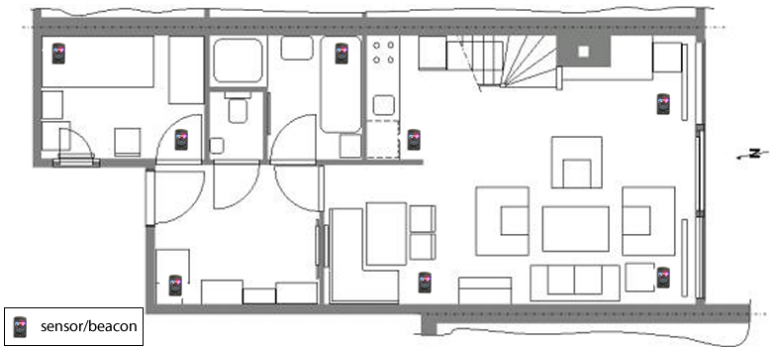

Figure 4.1: Illustration of the Strategic Placement of Receiving Beacons.

the received signal strength (*RSSi*) and the link quality identifier (*LQI*). Some papers and commercial solutions show that these values perform very well in simple environments [Blumenthal et al., 2007, Behnke and Timmermann, 2008]. In this chapter we show how this method performs in our experimental environments.

4.1 Technical Description and Capabilities

4.1.1 Sun SPOT Wireless Network Communications Technical Specification

The Sun SPOT is equipped with an integrated radio transceiver, the TI CC2420 chip and an inverted-F antenna. The TI CC2420 is IEEE 802.15.4 compliant and operates in the 2.4GHz to 2.4835GHz ISM unlicensed bands [Texas-Instruments, 2007]. The antenna is printed on the top layer of the printed circuit board and is tuned to $2450MHz$ and has a characteristic input impedance of 115Ω [Sun-Labs, 2009c].

4.2 Physical Background and Location Detection

Based on these technical capabilities, we built up a sensor network to measure the received signal strength (*RSSi*) and the link quality identifier (*LQI*). First of all we want to explain the characteristics and meanings of these two key performance indicators (KPIs) so that one can understand why these values are so important for our approach.

4.2.1 Received Signal Strength Indicator (RSSi)

To estimate the position of the device carried by the user, several distances between the strategic positioned beacons must be estimated. After determining the single distances from the sending node (carried by the user) and the strategic positioned beacons, an algorithm can perform the calculations to estimate the final position. A good indicator for estimating the distance between the sending and the receiving device is the received signal strength indicator (RSSi). The RSSi measures the strength (power) of the signal for the packet. In embedded devices, the received signal strength will be converted to a received strength indicator (RSSi), which is defined as the ratio of received power to reference power (P_{Ref}). The reference

power usually represents an absolute value of $P_{Ref} = 1mW$. So the following equation is for obtaining the RSSi [Blumenthal et al., 2007]:

$$RSSi = 10 \cdot \log \frac{P_{RX}}{P_{Ref}} \qquad [RSSi] = dBm \tag{4.1}$$

In our special case the TexasInstrument CC2420 chip [Texas-Instruments, 2007] used a similar formula for the RSSi which had more parameters to adjust [Aamodt, 2006].

$$RSSi = -(m \cdot n \cdot \log_{10} d + A) \qquad [RSSi] = dBm \tag{4.2}$$

- n, is the signal propagation constant, also called propagation exponent
- d, the distance from the sending device
- A, received signal strength at a distance of one meter
- m, a multiplier-value

The Figure 4.2 shows a typical curve-plot of a RSSi versus the sender to receiver distance.

One can see that the RSSi decreases with increasing distance. So the RSSi correlates well with the distance and therefore can be modeled as a function of the distance itself. In theory this works very well, but we wanted to take a closer look at the Sun SPOT behavior regarding RSSi and distance in a practical scenario. For this, we created a simple experiment: One Sun SPOT is placed on a *stationary* point and a second Sun SPOT was moved linearly from the closest position to the farthest position. In our special case we used a length 10m away from the initial starting point. Table 4.1 summarizes the experiment.

SunSPOTs	2
Grid	33cm
Total length	10m

Table 4.1: Experiment No. 1 - Empirical RSSi from Sun SPOTs

Our research has shown that the RSSi measurements do not exactly correlate with the theoretical model. Figure 4.3 shows the measured RSSi from experiment no. 1.

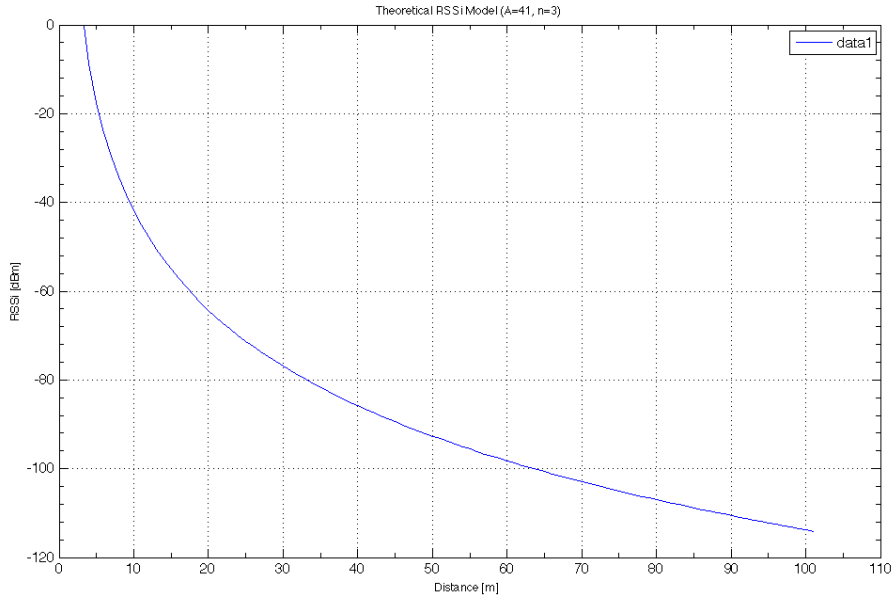

Figure 4.2: Theoretical RSSi versus Distance Plot with $A = 41$, $n = 3$ and $m = 10$ Parameter Setting, based on Equation 4.2.

We learned that these measurements differ from the theoretical signal propagation from equation 4.2. This uncertainty comes from a different source, such as noise, or influences from multipath components (just to mention a few). Most of this noise will be handled by chip hardware itself. Later we will introduce our software enhancements to even out these varying components as much as possible. Some final words on the theoretical RSSi model, based on equation 4.2: the parameters n and A significantly influence the resulting values and the calculation [Aamodt, 2006]. We will later discuss the best parameters found in our research experiment.

4.2.2 Link Quality Identifier (LQI)

In addition to the RSSi we also want to analyze the performance of the Link Quality Identifier (LQI) in the location determination calculation with the WCL-Algorithm. Referring to the IEEE 802.15.4 Standard, the LQI measurement is a characterization of the strength and/or quality of a received packet [Ergen, 2004]. As pointed out in [Blumenthal et al., 2007], the LQI shows an intense correlation with the distance and therefore can be used for the calculation of the current position. In their research [Blumenthal et al., 2007] showed that the LQI measured at four different reference-nodes in their experiment-setup (they refer to them as beacons) show

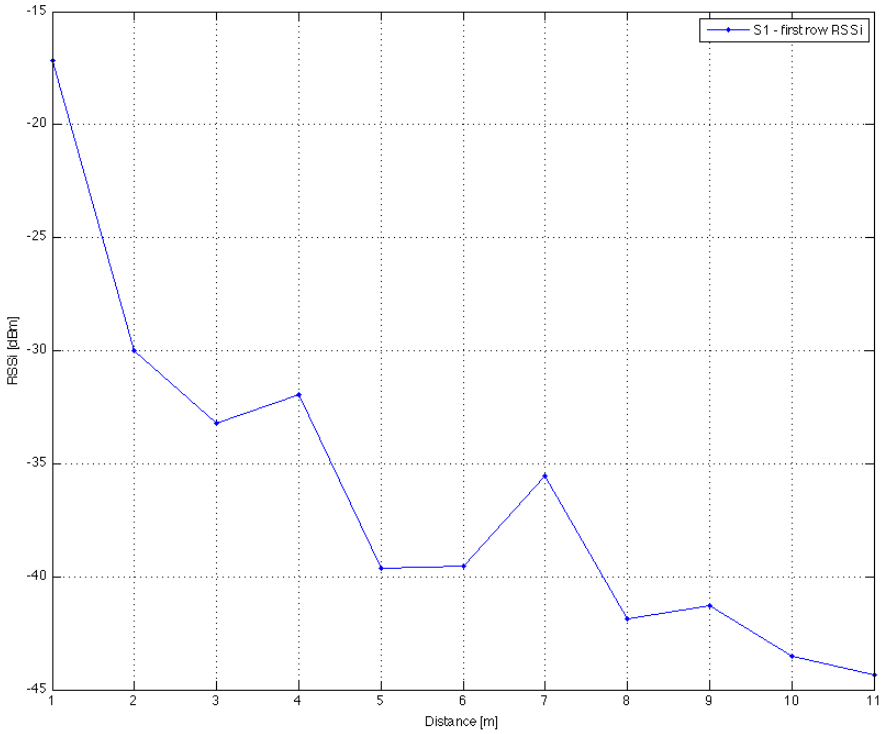

Figure 4.3: First RSSi empirical experiment, shows a downwards trend but not a exact correlation with the theoretical RSSi-model.

characteristic curves and satisfactorily shows the reproducibility of the distance determination.

For the experiment they varied the distance from the reference-nodes to the unknown-node between $0m$ and $40m$ and measured the LQI at each measuring-point 20 times. Similar to the RSSi the LQI decreases with the increasing distance and has systematic outliners which can be found at $d = \{4, 8\}$ (figure 4.4).

Figure 4.4 depicts the result from the LQI-Measurement experiment [Blumenthal et al., 2007]. Four different reference-nodes have measured the LQI from an unknown-node while varying the distance. Further, the figure also shows a good and reproducible correlation of the distance and the LQI [Blumenthal et al., 2007].

For the calculation we did not use our empirical model, instead we used a theo-

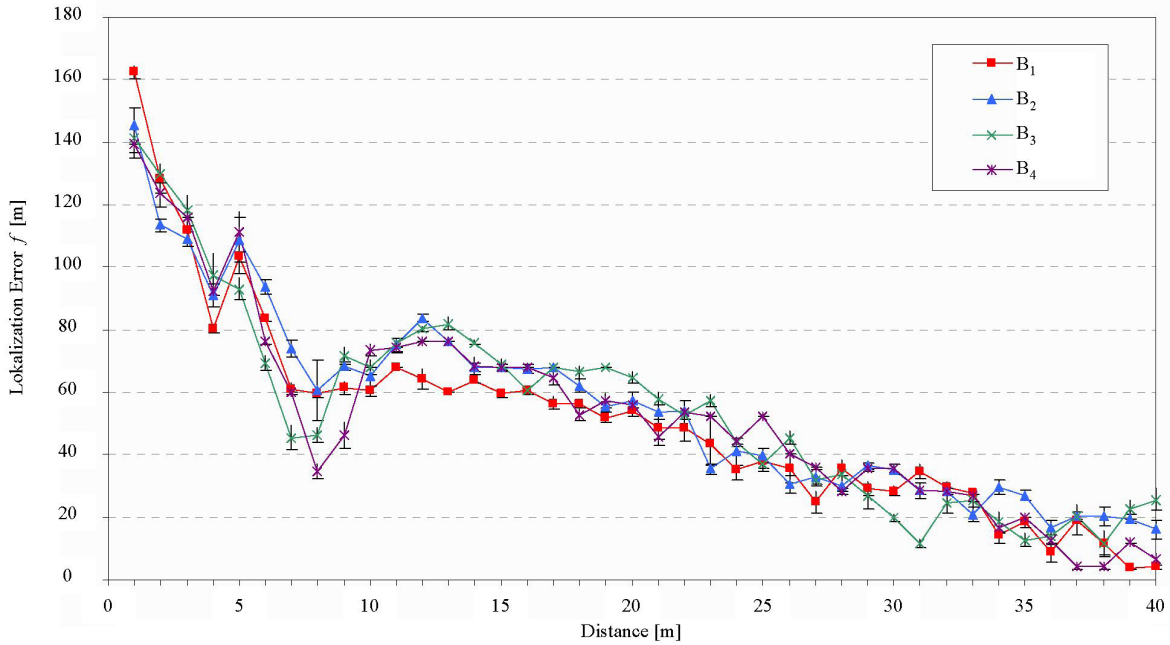

Figure 4.4: LQI measured at 4 different reference-node in a range varying from 0 to $40m$. Source: [Blumenthal et al., 2007].

retical model from [Behnke and Timmermann, 2008]. Equation 4.3 shows the used equation and its parameters.

$$LQI = \begin{cases} 255 & d = 0 \\ 25 \cdot \left(\ln \left(\left(\frac{R_C}{100 \cdot d} \right)^2 \right) + 10 \right) & 0 < d \leq R_C \\ 0 & d > R_C \end{cases} \tag{4.3}$$

Note that the R_C is the transmission range of the reference-nodes in $[m]$. For our four reference-nodes (Sun SPOTs) we have used a transmission range of $10m$ which seems to be realistic and adequate for this approach.

4.2.3 The Weighted Centroid Localization Algorithm

Introduction

The Weighted Centroid Localization Algorithm (WCL) [Blumenthal et al., 2007] is a good algorithm for locating so-called unknown sensor nodes in wireless sensor

networks. This algorithm is derived from the original Centroid Localization (CL) [Bulusu et al., 2000], which infers the location by averaging positions between known sensor nodes or beacons B_i (reference node is also a very common term and we will use this term in our research) and unknown nodes S_i.

These simple reference nodes are able to determine their exact position (for example with a built-in GPS module), or - if they are stationary - their position will be configured in an initial setup-phase. Usually it is satisfactory to store the X and the Y coordinate.

Based on an indicator value (in our special case we used the LQI and RSSi), these reference nodes are weighted and therefore have just a certain influence on the location determination process for the unknown node[1]. Imagine the case just between two reference nodes $Ref_1(x, y)$ and $Ref_2(x, y)$. The closer the unknown node $N_{unknown}(x, y)$ is moving to one reference node $Ref_1(x, y)$, based on the theoretical signal-propagation model, the better the value for the indicator value (for example, the RSSi). Therefore, this reference node $Ref_1(x, y)$) has to get more weight than the other reference node $Ref_2(x, y)$. The WCL algorithm is based on this theory. Figure 4.5 shows the above described basic concept of varying weights depending on indicator values. In this special case the RSSi is used.

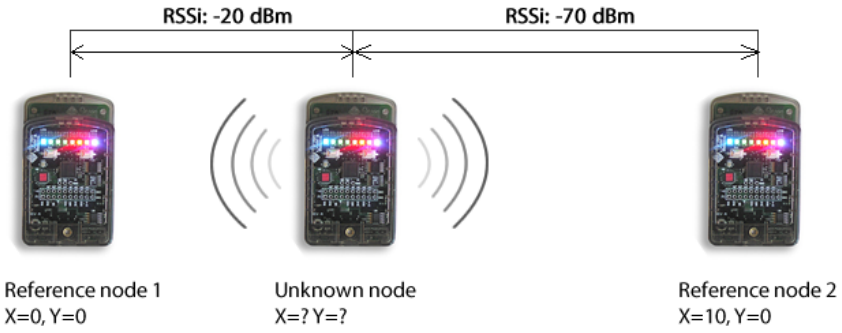

Figure 4.5: WCL - Basic concept, the better the indicator-value, RSSi, the more weight the reference node gets. This correlates at a close distance to the reference node.

[1]Nodes which contain their position we will call *reference nodes* and the node whose position should be estimated is the *unknown node*.

The WCL Algorithm in Detail

The WCL algorithm is an adaption of the CL algorithm. We define a sensor network consisting of N_R reference nodes and N_S sensor nodes.The reference nodes can determine their own position or the position is assumed to be fixed. The sensor nodes do not know their position initially and will estimate their position based on the reference nodes in their transmission-range (not all nodes in the sensor-network will be used for determination process) and the WCL algorithm. As stated in [Blumenthal et al., 2007], the WCL-formula can be derived from the CL initial assumptions.

We start from equation (4.4) to derive the formula for the WCL algorithm:

$$P_i'(x,y) = \frac{1}{n}\sum_{j=1}^{n} B_j(x,y) \tag{4.4}$$

And the error between the exact position (x, y) and the calculated position (x', y') is defined in equation 4.5.

$$f_i(x,y) = \sqrt{(x'-x)^2 + (y'-y)^2} \tag{4.5}$$

Due to the fact that the WCL algorithm uses weights to improve the location determination, equation (4.4) can be altered by representing n as the sum of ones and the multiplication of B_j with one. After the above described steps, equation (4.6) can be derived [Blumenthal et al., 2007].

$$P_i'(x,y) = \frac{1}{\sum_{i=1}^{n} 1}\sum_{j=1}^{n} 1 \cdot B_j(x,y) \tag{4.6}$$

The final WCL formula is derived by replacing the ones by the weight-function w_{ij}, as presented in equation (4.7) [Blumenthal et al., 2007].

$$P_i''(x,y) = \frac{\sum\limits_{j=1}^{n} (w_{ij} \cdot B_j(x,y))}{\sum_{j=1}^{n} w_{ij}} \tag{4.7}$$

To improve the position estimation calculation a weight function w_{ij} has been introduced, which takes the distances from the reference-nodes into account. Higher distances have therefore less influence on a reference-node's position than lower distances. In other words, the higher the distance from the reference node to the unknown node, the less the influence of the reference-node's coordinates (position) in the calculation. Equation (4.8) shows the weight-function [Blumenthal et al., 2007] we used in our experiments.

$$w_{ij} = \frac{1}{(d_{ij})^g} \tag{4.8}$$

One can see that the weight function and the distance are inversely proportional. The parameter g is evaluated during the experiment test series. We want to point out that the higher the value of g, the less the influence of the very distant reference nodes. This happens because the position of the unknown node moves closer to the closest reference node. Therefore the error $f_i(x, y)$ increases with higher g but can reach a minimum at a certain value of g. This value varies depending on the environment or other influences and therefore has to be evaluated empirically (as mentioned above we will show an optimum value for g during our experiment series in our case).

Figure 4.6 illustrates the WCL-algorithm scheme with four reference nodes $R_N = \{R_1, R_2, R_3, R_4\}$ and one unknown node U. One can see that the reference node R_2 is the closest to the unknown-node and therefore had the highest weight.

4.3 Conceptual Design

4.3.1 Introduction

This section covers the conceptual design of the *Radio Frequency Approach Experiment*. To prove the feasibility of the location determination capabilities of a wireless sensor network based on IEEE 802.15.4, we created an indoor experimental testbed. We will introduce the main idea behind this experiment and give a detailed description of the used hardware, the environmental conditions and the software-architecture we developed to carry out this experiment. Detailed measurements were taken and evaluated in *MatLab-Software* for detailed analysis and explanations of the result sets. For real-time presentation, a software-framework based on *Sun SPOT service architecture* by Ron Goldman was developed to show graphically how location determination works in our framework. In the last part of this section

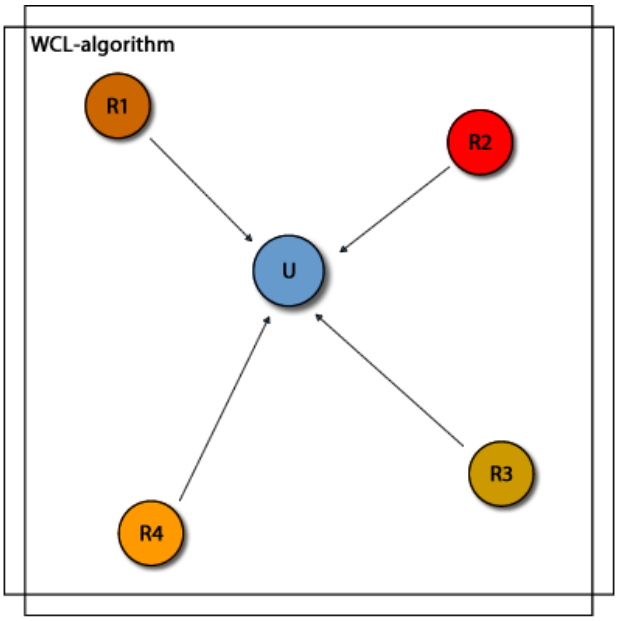

Figure 4.6: WCL Algorithm with Four Reference Nodes. Reference node R_2 is the closest node to the unknown node U and therefore has more weight than the other nodes.

we will examine the experimental results and show the current performance of this approach and how it could be improved and adapted in the near future.

4.3.2 Experimental Testbed Overview

For our experimental testbed we created a wireless sensor network SN_6 with 4 reference nodes $R_N = \{R_1, R_2, R_3, R_4\}$, one unknown node U and one base station BS. This sensor network works on a specified communication channel and is interconnected over the BS with a host (MacBook Pro 2.4GHz Intel Core 2 Duo with 4GB of RAM). Our developed evaluation framework records several measurement series on this host. Further it captures a certain number of samples and create a statistical file for detailed evaluation in a mathematical / statistical software environment (in our special case we used MATLAB). This evaluation framework takes measurements from all four reference nodes $R_i, i = 1, 2, 3, 4$ and records the following performance indicators:

- RSSi, received signal strength indicator

- LQI, link quality identifier
- CORR, correlation value
- Timestamp in [ms]

Figure 4.7 illustrates a rough overview of the experimental testbed and its configuration. We chose a sports hall to evaluate the indoor performance of the sensor network.

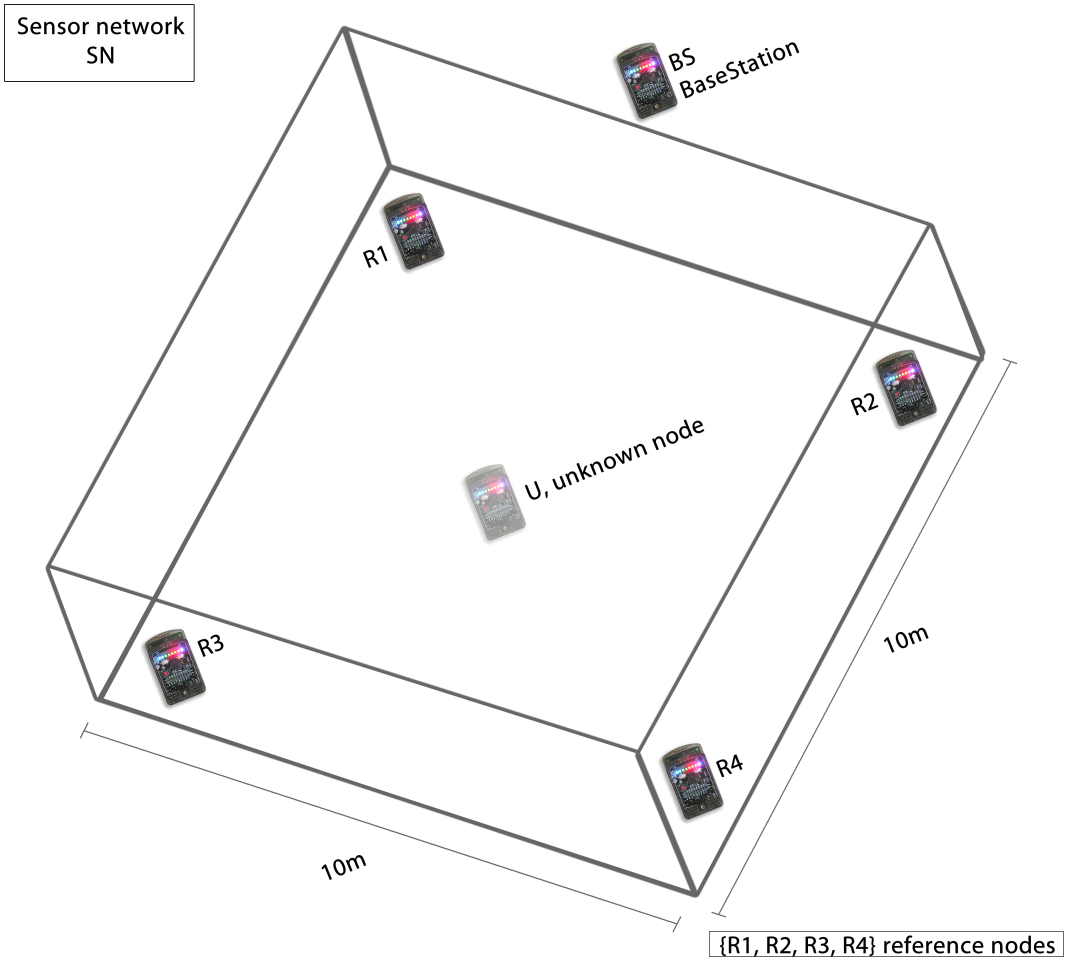

Figure 4.7: RF-Experiment, Rough Overview.

The four reference nodes $R_N = \{R_1, R_2, R_3, R_4\}$ were aligned in a rectangular order in each corner of the hall. Further, we used plastic stands to position the Sun SPOTs at a height of $1.80m$. The Sun SPOTs (reference nodes) always faced inwards into the room. Figure 4.8 illustrates the stands and the position of the devices. Please

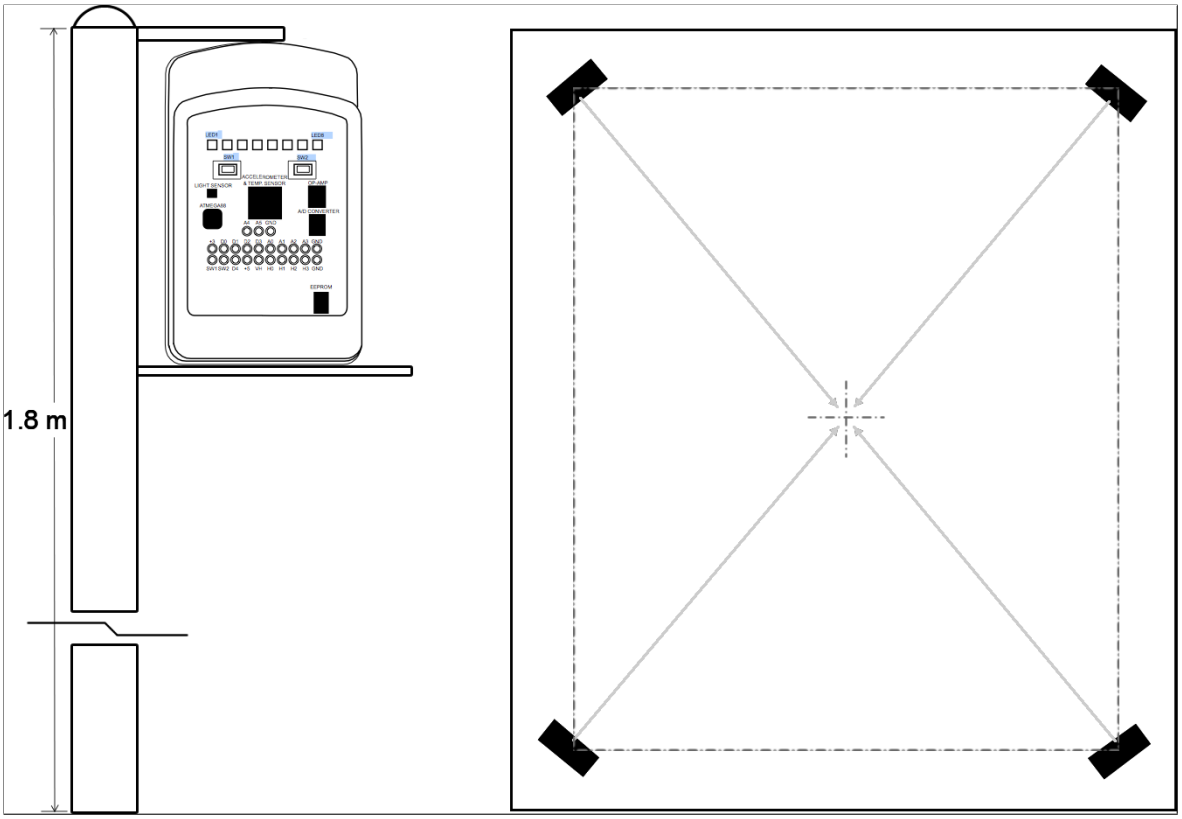

Figure 4.8: Sun SPOT Configuration Setup. The stand is depicted on the left and the right side shows a depiction of the Sun SPOTs facing into the center of the hall.

note that the front face of the Sun SPOT with the LEDs looks inwards (as does the arrow depicted in the figure).

For statistical measurements we defined a $1m \times 1m$ grid rectangle, which we moved from reference node R_4 in the X-axis linear to R_2 and in the Y-axis successively from R_4 to R_3. Figure 4.9 shows the measurement process in detail.

The orange dots represent the measurement points $M_{i,j}, i = 0(1)10; j = 0(1)10$. In each row eleven measurements were taken, starting from $0m$ to $10m$. Each measurement point includes the four indicator values (RSSi, LQI, CORR and the timestamp in milliseconds) for all four reference nodes and was measured till gathering a total number of 120 samples for each reference node. So we have extracted a four dimensional feature vector (Equation 4.9), 120 times per measurement point. That are in total 14520 measurements in the experiment ($11 \times 11 \times 120$).

$$V_{n,i,j} = \{time, RSSi, LQI, CORR\}, i = 1(1)11, j = 1(1)11, n = 1(1)120 \quad (4.9)$$

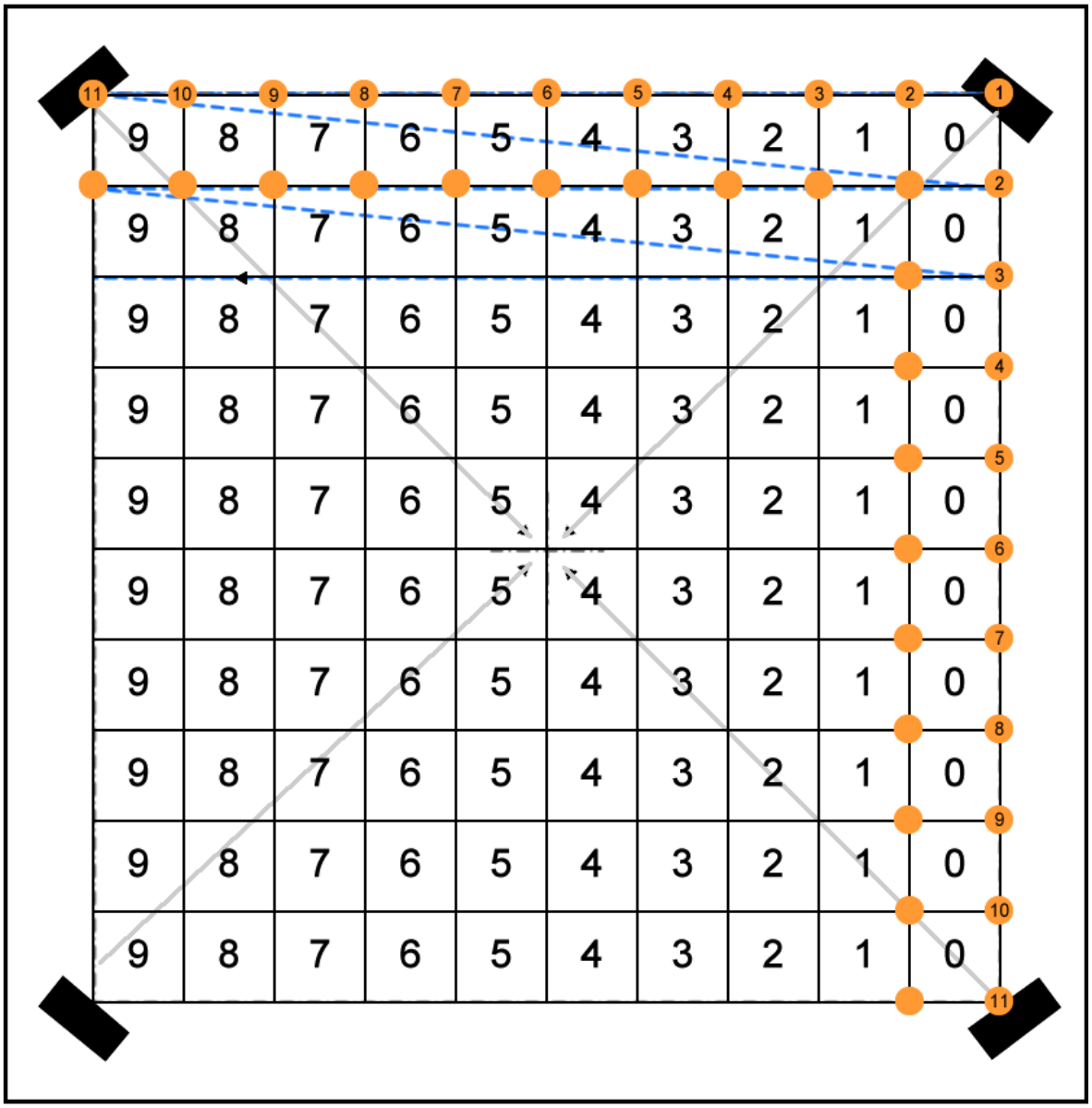

Figure 4.9: RF Approach Grid Alignment and Measurement Direction. The orange dots are the measurement points (11x11 in total). For each measurement point $M_{i,j}$ 120 feature vectors were extracted.

To eliminate or reduce the noise on measurement point $M_{i,j}$ we averaged, overall, 120 feature vectors for a given measurement point. We finally derived one feature vector $F_{i,j}$ for each measurement point $M_{i,j}$ Though in total we had 120 feature vectors, for our research we analyzed only the averaged feature vectors.

4.3.3 Software Architecture

Introduction

Before we get into the implementation details, we want to describe the process-chain behind the RF-based approach. As mentioned before, we have a sensor-network SN, reference-nodes R_i and the unknown(s) U_i . The core-system and the reference-nodes are assumed to be stationary, but the unknowns can move about the indoor environment. We created a software-framework, which represents the requirements mentioned earlier, but it is not immediately usable in a smart-home environment. The architecture needs certain adaptations to achieve other vital requirements for the smart-home environment (especially for integrating this approach as a location inference component, additional top-level-interfaces have to be provided). But we provide a basic system for communication and calculation for both the core system and the participating devices. For our experiment we've created a software based on the *Sun SPOT Telemetry Demo framework* provided in the software developer kit from Sun SPOT. To make things clearer we want to map the theoretical-model terms to our software-model terms:

- SN sensor network remains as the *sensor network*
- R_i reference-nodes are the *RadioFrequencyEvaluators*
- U_i unknown node(s) are the *TestObejctServer*
- The core-system is represented partly by the *RFDisplayServer*

The main theory behind our software-model is that we have stationary evaluators (RadioFrequencyEvaluators) which first participate in a specific test object server (TestObjectServer) and a display server (DisplayServer). The test object server broadcasts dummy packets while moving through the environment allowing the evaluators to receive these packets and measure the performance indicators necessary for the calculation at the display server. The measured packets from all evaluators are forwarded continuously to the display server and gathered together there for performing a WCL algorithm in order to calculate the current position of the test object.

Components

After this brief introduction we want to go into more detail by explaining the components of our architecture. We created a *TestObjectServer* component, which

represents the unknown device U. This unknown device starts a service and waits for other *RadioFrequencyEvaluators* to participate in this *TestObjectServer*. After a predefined period of time, normally 60 seconds, the *TestObjectServer* starts to broadcast dummy packets. Before we go further into detail regarding what happens with these dummy packets, we first want to explain what the *RadioFrequencyEvaluators* do.

There are four *RadioFrequencyEvaluators*, representing the four reference nodes $R_i, i = 1(1)4$ in our sensor network SN_6. After each begins the task of gathering dummy packets from a *TestObjectServer*, they try to connect to a host called *RFDisplayServer*. This server represents the main application, which takes all samples /

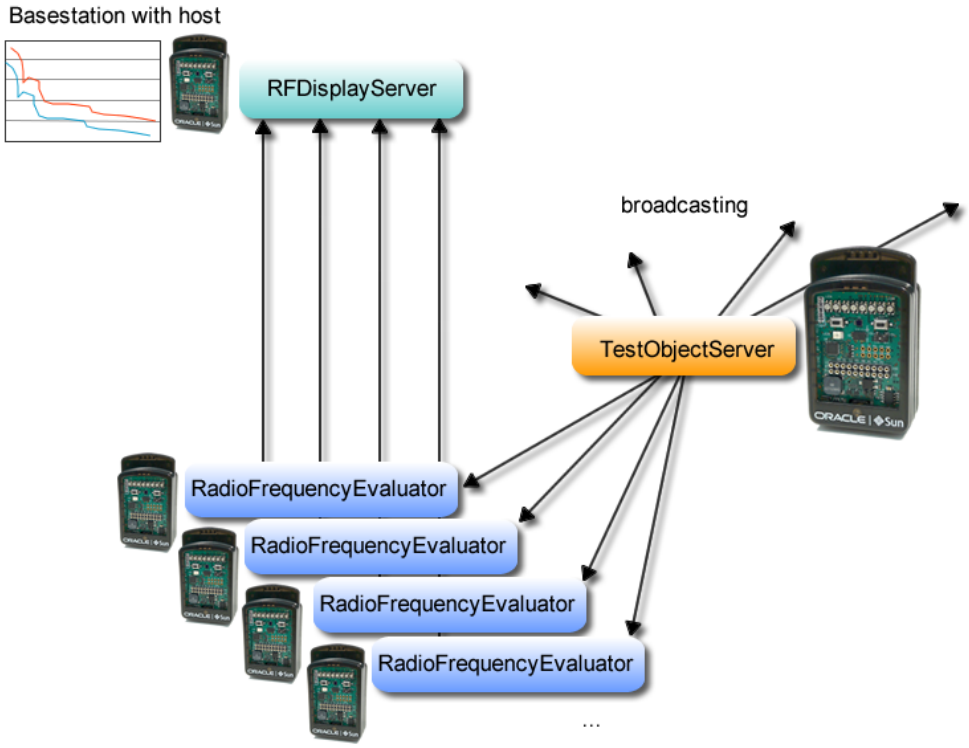

Figure 4.10: Software Architectural Overview of the RF-Based Approach.

feature vectors from all of the connected reference nodes. Of course this application is designed to be attached to more than four reference nodes. The *RFDisplayServer* can be configured to take a experiment series. This series is defined by a basic file name including the number of samples being gathered for an *experimental series*. It automatically creates multiple files with the basic filename and the number of

the experimental series. These files contain all gathered feature vectors for each connected reference node and can be simply used for further analyses in a mathematical software environment. Further, for graphic presentation, a graph-view was implemented to show how the important performance indicators develop over time (time versus value grid-view). For a real-time presentation, we also created a frame displaying the current position of the unknown-device (*TestObjectServer*, U), based on pre-configured coordinate sets for each of the four reference-nodes. With this software we have created our statistical-data environment and also are able to present how the location inference works in almost real-time. Figure 4.11 depicts the detailed components and their interfaces to get a detailed understanding of the RF approach software architecture. The following subsections cover a very detailed explanation of the components and how they interact.

TestObjectServer -TOS

First we want to describe the purpose of the *TestObjectServer*. This component represents the unknown-device in our location inference task and is running on a Sun SPOT device. The Sun SPOT was mounted on a stand and moved on a predefined grid to take measurements (see figure 4.9) for our experiments. Like the *RadioFrequencyEvaluators*, the *TestObjectServer* is built upon a service-like architecture (as described in chapter 2).
In our special case, only one service, the *TOCommunicationServerService* is installed and running. The following paragraphs describe the procedure of setting up the *TestObjectServer* and how it acts throughout its uptime.

The *TOServerMain* class initiates the services in its main startup-procedure `initialize(...)`. After initializing the service it sets the radio-related configurations and waits for a predefined amount of time at a predefined broadcast-port so that other *RadioFrequencyEvaluators* can join to the server (`waitForSpot()`). After receiving a special packet-type from an *RadioFrequencyEvaluator* (`LOCATE_TEST_OBJECT_SERVER_REQ`), the server simply replies with its own `IEEEAddress` - address [2] and the information that a server is available in the context (`TEST_OBJECT_SERVER_AVAIL_REPLY` packet type). As stated above, more than one *RadioFrequencyEvaluators* can participate at a server, but only during the waiting period. Note that in a smart-home environment there will be no restriction, because if the unknown node moves through a bigger sensor network it has to make its presence known all the time. This is due to the fact that *RadioFrequencyEvaluators* are bound only to a specific area (technical restriction due to the radio-communication). After the waiting-interval, the waiting-loop terminates and the server starts the *TOCommunicationServerService* by simply

[2] The `IEEEAddress` is an address in the format 0000.0000.0000.0000 hexadecimal

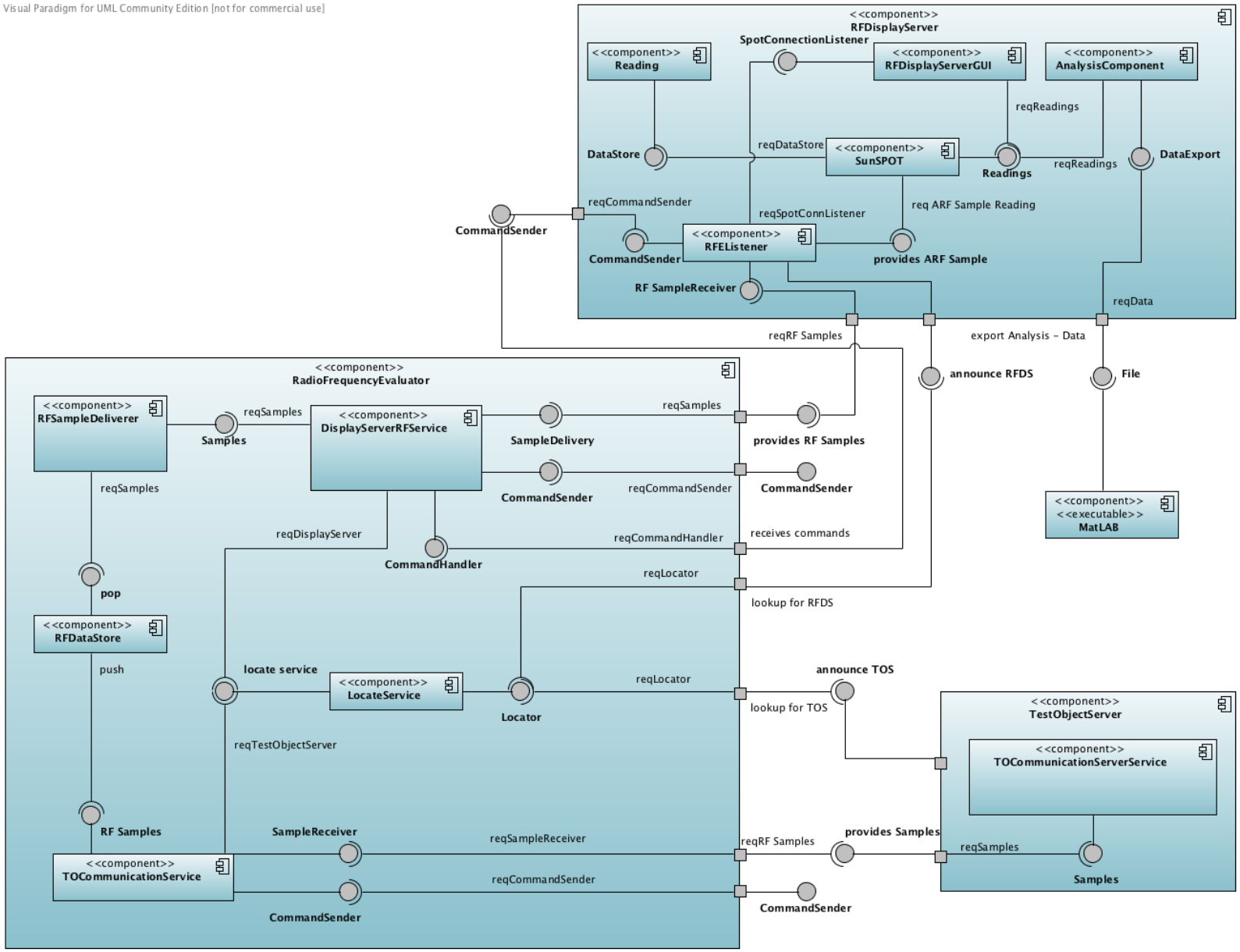

Figure 4.11: Detailed UML Component Diagram of the RF-Based Approach.

calling `tocommSrvService.start();`.

The *TOCommunicationServerService* broadcasts in the periodic-task loop datapackets p_n with a timestamp (no more information is needed $p_n = \{timestamp : Long\}$). Note that this task can be performed very quickly and does not consume too much processing-time (a detailed view of the task is listed in 4.1).

Listing 4.1: doTask - method inside of the TOCommunicationServerService

```
1  public void doTask() {
2          leds[SENDING_INDICATOR_LED_INDEX].setRGB(0, 250, 250);
3          leds[SENDING_INDICATOR_LED_INDEX].setOn();
4          try {
5              // packet structure:
6              // headerinfo;starttime;samplesperpacket
7              startTime = System.currentTimeMillis();
8              currentPkt = xmitTORFSampler.newDataPacket(packetHdr[index]);
9              currentPkt.writeLong(startTime);
10             xmitTORFSampler.send(currentPkt);
11             System.out.println("Sending␣packet␣No.␣" + rfsamples);
12             currentPkt = null;
13             ++rfsamples;
14             leds[SENDING_INDICATOR_LED_INDEX].setOff();
15         } catch (IOException ex) {
16             ex.printStackTrace();
17         }
18     }
```

This service works in an endless loop until the device is turned off. Again, for the use in a real smart-home environment, this architecture has to be adapted so that the service should be able to restart or stop at any time (initiated through a user or through the core system).

RadioFrequencyEvaluator - RFE

The *RadioFrequencyEvaluator* plays a very important role in the RF-based approach and utilizes the incoming data-packets periodically spawned by the *TestObject-Server*. Then it forwards it to an available *RFDisplayServer*, which then estimates the position of the unknown nodes or simply records the values for statistical purposes. From the *RadioFrequencyEvaluator's* point of view, it has to interact with two other independent components, first actively with the *RFDisplayServer* and then, second, passively with the *TestObjectServer*.

First of all, a description of the start-up procedure of the *RadioFrequencyEvaluator*:

similar to the *TestObjectServer*, it first creates all required services to communicate with other components. Two services, the *DisplayServerRFService* and the *TOCommunicationService* are instantiated. Both of them make use of a *Locator* to locate the other service-components in the sensor-network. The *DisplayServerRFService* is responsible for an active communication with the *RFDisplayServer*. On one hand it receives commands from the *RFDisplayServer* and, on the other hand, it provides *RFSamples* for the *RFDisplayServer*. Further the *DisplayServerRFService* gets its *RFSamples* from a global (singleton-class) data store (*RFSamplesDataStore*). This brings us to the responsibilities of the *TOCommunicationService*. This service accepts and measures incoming samples from the *TestObjectServer* (it measures RSSi, LQI, CORR, and attaches a timestamp at which time the packet was received) and pushes it into the global data store as an *RFSample* class. Both services access the data store (one pushes *RFSamples* into the data store and the other pops these *RFSamples* from the data store). The *DisplayServerRFService* periodically creates new packets with the stored *RFSamples* in the global data store and sends it to the connected *RFDisplayServer* for further processing. This process chain can be seen in the component diagram (see figure 4.11).

Listing 4.2: handlePacket - method inside of the TOCommunicationService

```
1  public void handlePacket(byte type, Radiogram pkt) {
2      try {
3          switch (type) {
4              ...
5              case TEST_OBJECT_SERVER_RF_SAMPLE:
6                  if (_demoBoardAvailable) {
7                      leds[RECEIVER_INDICATOR_LED_INDEX].setRGB(0, 250,
                           250);
8                      leds[RECEIVER_INDICATOR_LED_INDEX].setOn();
9                  }
10                 // if an radio-frequency test sample arrives push it
11                 // to the data-store; Delivery takes place
12                 // elsewhere
13                 // read information from incoming packet
14                 long senderTime = pkt.readLong();
15                 long ourTime = System.currentTimeMillis();
16                 int RSSi = pkt.getRssi();
17                 int LQI = pkt.getLinkQuality();
18                 int CORR = pkt.getCorr();
19
20                 // create new sample and push it into the data-store
21                 RFSample newSample = new RFSample(RSSi, LQI, CORR,
                       senderTime, ourTime);
22                 dataStore.push(newSample);
23
24                 System.out.println("Received␣sample␣...");
25                 if (_demoBoardAvailable) {
26                     leds[RECEIVER_INDICATOR_LED_INDEX].setOff();
```

```
27                      }
28                      break;
29              }
30          } catch (IOException ex) {
31              ...
32          }
33      }
```

Listing 4.2 shows the important parts of the `handlePacket` method and how incoming 'dummy' packets from the *TestObjectServer* are being processed and stored (line 22 and 23).

For the sample delivery, the *DisplayServerRFService* uses the *RFSampleDeliverer* for taking stored samples out of the datastore and delivering them to the *RFDisplayServer*. Listing 4.3 depicts the important part of the `doTask` method of *RFSampleDeliverer* where a delivery-packet will first be created (line 7) and then will be filled up with *RFSample* vectors (line 17 -25). Every delivery-packet to the *RFDisplayServer* contains more than one feature vector from the *TestObjectServer*. So we define a delivery-packet for the *RFDisplayServer* as follows:

$$DP_i = (RSSi_1, LQI_1, CORR_1, tSndr_1, tRcvr_1, \\ ..., RSSi_n, LQI_n, CORR_n, tSndr_n, tRcvr_n)$$

In this case n represents `RF_SAMPLES_PER_PACKET`.

Listing 4.3: doTask method inside the RFSampleDeliverer

```
1  public void doTask() {
2          if (_demoBoardAvailable) {
3              leds[SENDING_INDICATOR_LED_INDEX].setOff();
4          }
5          if (!dataStore.isEmpty()) {
6              try {
7                  if (currentPkt == null) {
8                      // packet structure:
9                      // headerinfo;starttime;samplesperpacket
10                     startTime = System.currentTimeMillis();
11                     currentPkt = xmit.newDataPacket(packetHdr[index]);
12                     currentPkt.writeLong(startTime);
13                     currentPkt.writeByte(RF_SAMPLES_PER_PACKET);
14                     currentSample = 0;
15                 }
16                 // get next available sample
17                 RFSample sample = dataStore.pop();
18
19                 if (sample != null) {
```

```
20                      //format: RSSi from packet, LQI form packet, CORR
                            from packet, timestamp from test-object, timestamp
                             from receiving packet at receiver
21                      currentPkt.writeInt(sample.RSSi());
22                      currentPkt.writeInt(sample.LQI());
23                      currentPkt.writeInt(sample.CORR());
24                      currentPkt.writeLong(sample.SenderTime());
25                      currentPkt.writeLong(sample.ReceiverTime());
26
27                      if (++currentSample >= RF_SAMPLES_PER_PACKET) {
28                          xmit.send(currentPkt);
29                          if (_demoBoardAvailable) {
30                              leds[SENDING_INDICATOR_LED_INDEX].setRGB(0,
                                    250, 250);
31                              leds[SENDING_INDICATOR_LED_INDEX].setOn();
32                          }
33
34                          System.out.println("Sent␣packet␣to␣display␣server
                                ␣via␣Deliverer...");
35                          currentPkt = null;
36                      }
37                  }
38              } catch (IOException ie) {
39                  main.queueMessage("IO␣exception:␣" + ie.toString());
40              }
41          }
42      }
```

RFDisplayServer - RFDS

The last part of our software-architecture is the *RFDisplayServer* component, responsible for receiving radio frequency sample data from all registered *RadioFrequencyEvaluators*, for performing the storage of this data and the location inference. It fulfills the following tasks:

- Registering available *RadioFrequencyEvaluators*
- Recording incoming live data from *RadioFrequencyEvaluators*
- Creating experimental test series based on predefined configuration and defined post-processing
- Performing location-inference calculation on a predefined environmental setting

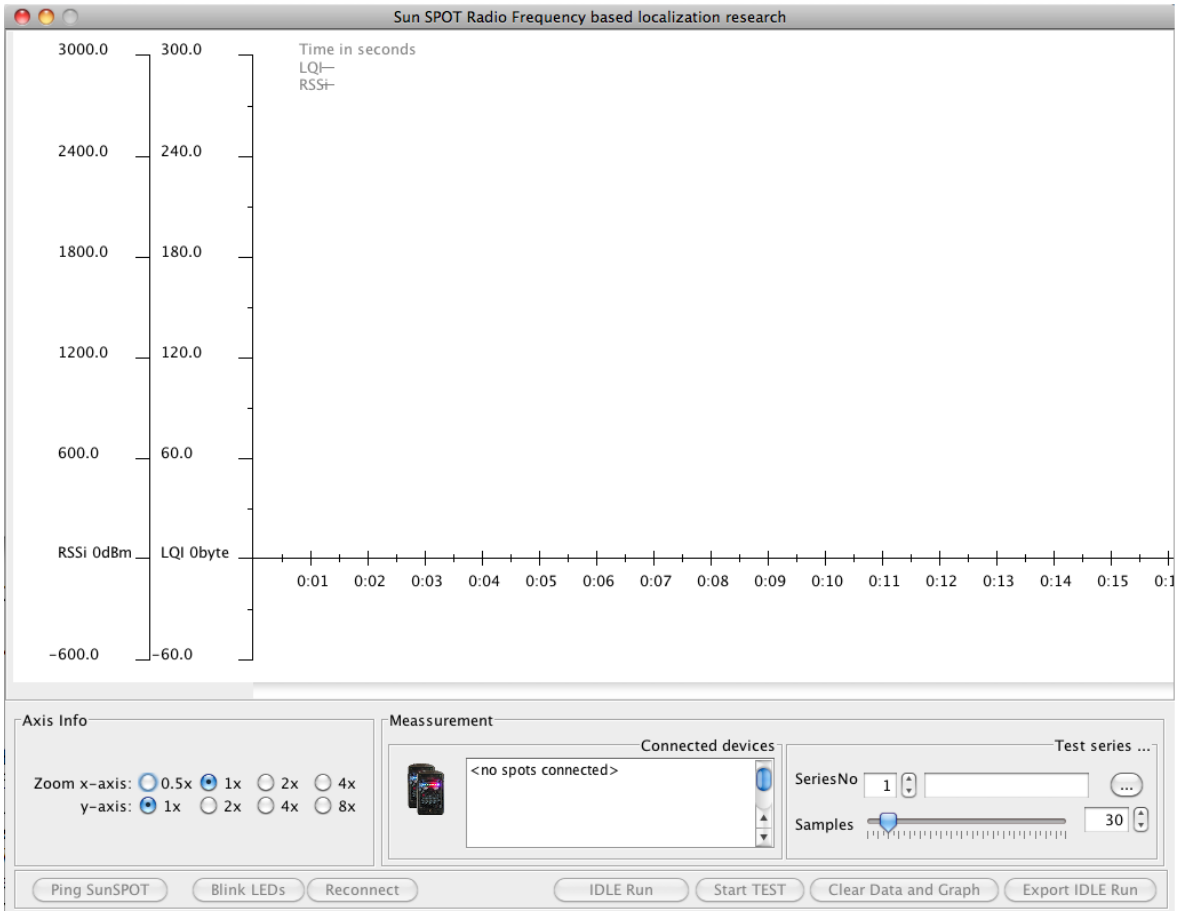

Figure 4.12: The main window of the *RFDisplayServer* component shows a 2-dimensional grid for displaying the collected data from different *RadioFrequencyEvaluators* and offers control panels for the connected devices and for experimental test-series.

- Graphically displaying the received statistical data from diverse *RadioFrequencyEvaluators* in a 2-dimensional graph
- Graphically displaying the current position of the unknown device in a 2-dimensional graph

All these items are described in the following paragraphs in detail. Regarding the first item, the *RFDisplayServer* waits, after setting itself up, a certain amount of time and gives other *RadioFrequencyEvaluators* the possibility to connect to this server-component. The *RFDisplayServer* keeps track internally of all registered *RadioFrequencyEvaluators*, so that the incoming data can be mapped correctly. After the initial setup and registration phase, it starts to collect incoming live data from the registered evaluators and displays them in the 2-dimensional grid. The main

component for the collecting purpose is the *RFEListener*. This component receives incoming data packets and assigns, based on the sender-address, the received data packet to Sun SPOT-Reading-store (Listing 4.4 shows this procedure). So internally all data packets will be assigned to specific reading tables of every registered Sun SPOT-device (class `SunSPOT` and class `Reading`). As one can see, based on figure 4.11, these readings will be used for the graphical representation and a data-export for further analysis and/or post-processing.
If the user provides coordinates for at least four *RadioFrequencyEvaluators*, the localization inference for one unknown object can be performed live. The current position is calculated based on the WCL algorithm and displayed on a 2- dimensional grid (see figure 4.13).

Listing 4.4: receive - method inside of RFDisplayServer

```
1     synchronized private void receive(Datagram dg) {
2
3         // only proceed if we are collecting, otherwise ignore incoming
              responds
4         // we will only look at the first spot regarding sample - size,
              we assume that all spots collect the same amount
5         if (_collecting && (!CollectedSamplesReached() ||
              _samplesToCollect == INFINITE)) {
6             try {
7
8                 String address = dg.getAddress();
9                 long timeStamp = dg.readLong();         // NOTE that
                      timestamp offset - handling will be done in SunSPOT
                      class
10                int sampleSize = dg.readByte();         // Number of
                      SensorData contained in the datagram
11
12                IEEEAddress ieeeaddress = new IEEEAddress(dg.getAddress()
                      );
13                SunSPOT addressedSpot = ConnectedSunSPOTs.Instance().
                      GetSPOTByIEEEAddress(ieeeaddress);
14                AddressedRFSample receivedSample;
15                System.out.println("RECEIVED_PACKET");
16                for (int i = 0; i < sampleSize; i++) {
17                    int RSSi = dg.readInt();
18                    int LQI = dg.readInt();
19                    int CORR = dg.readInt();
20                    long senderTime = dg.readLong();
21                    long receiverTime = dg.readLong();
22
23                    receivedSample =
24                            new AddressedRFSample(ieeeaddress, RSSi, LQI,
                                  CORR, senderTime, receiverTime);
25
26                    _collectedSamples.add(receivedSample);
```

```
27
28                      if (addressedSpot != null) {
29                          addressedSpot.AddLQIReading(receivedSample.
                                SenderTime(), receivedSample.LQI());
30                          addressedSpot.AddRSSiReading(receivedSample.
                                SenderTime(), receivedSample.RSSi());
31                          addressedSpot.AddCORRReading(receivedSample.
                                SenderTime(), receivedSample.CORR());
32                      }
33                  }
34
35                  graphView.takeData();
36              } catch (IOException e) {
37                  e.printStackTrace();
38              }
39          } else if (_collecting) { // enough samples gathered stop
                collecting and notify listener
40              StopCollecting();
41              boolean enough = true;
42              for (ICollectingFinishedListener listener :
                    _collectingListeners) {
43                  listener.finishedCollecting(_collectedSamples.size(),
                        enough);
44              }
45          }
46      }
```

4.4 Feasibility Study and Environmental Tests

4.4.1 Introduction

This section covers one of the main foci of this thesis, which is to analyze the possibilities of a radio-frequency based approach with the Sun SPOT. We will further explain all statistical methods and analysis we performed to examine the feasibility of this approach. We used MATLAB to perform this analysis and also developed a great set of scripts to first aggregate the available data and then perform the detailed analysis.

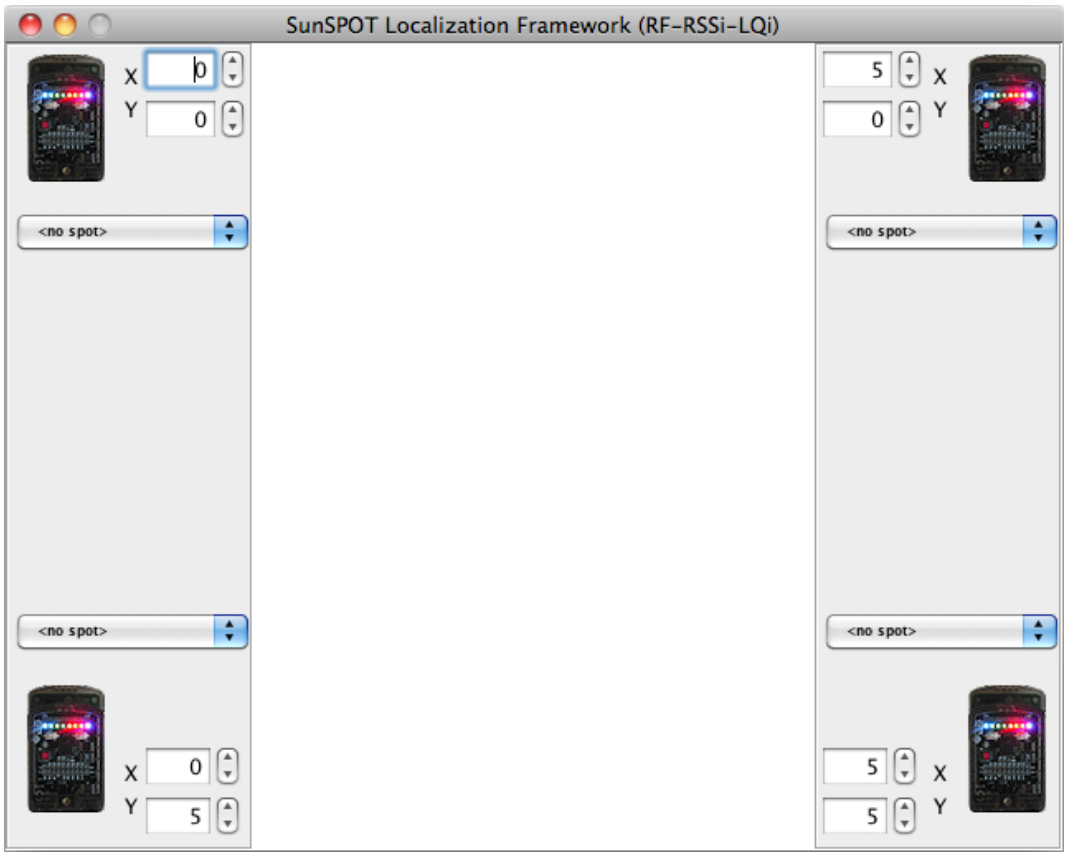

Figure 4.13: The live location-inference window of the *RFDisplayServer* component shows a 2-dimensional grid for displaying the current position of the unknown node.

4.4.2 Collected Data

As mentioned in the previous chapters we collected 11×11 statistical files from our *RFDisplayServer* component. These files contain for every measuring-point $M_{i,j}$ all performance indicators we needed for further statistical analysis. First of all we had to create 2 matrices $m_{LQI} = 11 \times 11, m_{RSSi} = 11 \times 11$ for each reference-node R_i from these files to start our analysis (in total there were $4 \cdot 2 = 8$ matrices). After this procedure we obtained the measuring data - once for the LQI and the RSSi— for each and every sensor (in MatLab we called them $S1, S2, S3, S4$ which refer to $R1, R2, R3, R4$). To make clear the relationship between the position and naming of the sensors/reference-nodes, figure 4.14 illustrates the exact addresses (#NNNN), reference-node number (RN) and sensor-number (SN).

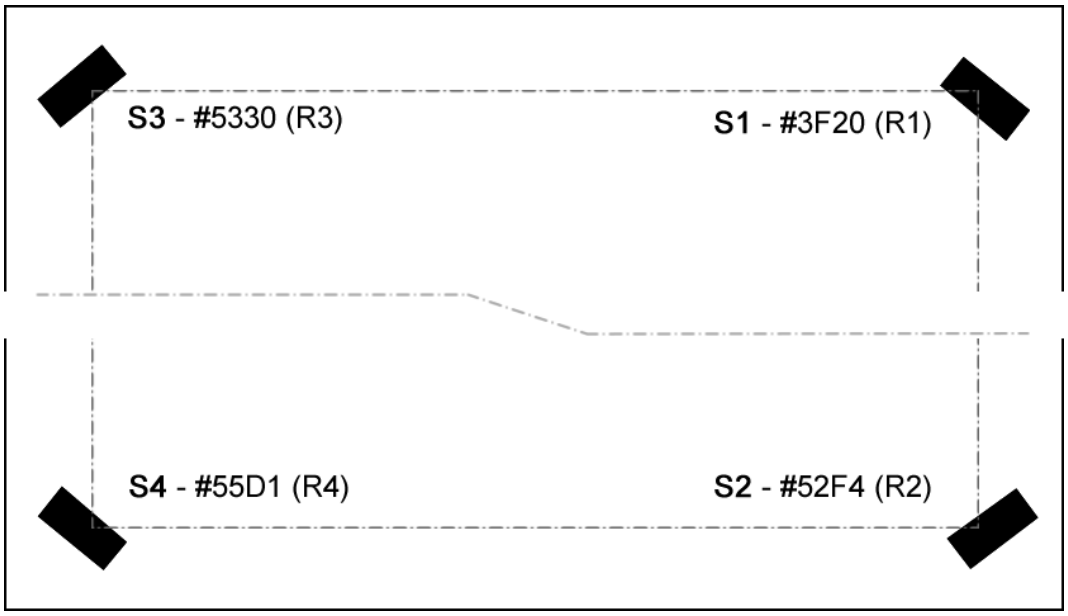

Figure 4.14: Naming and Address Relationship of the Sensor Network.

LQI Measurements

In this subsection we will present our results of the measurements-series regarding the link quality identifier (LQI). In total we've extracted four 11×11 matrices $m_{LQI_i}, i \in \{1, 2, 3, 4\}$ (i represents the sensor number of S_i) containing the average LQI values of every measurement point $M_{i,j}$.

All four plots represent a common behavior of the LQI because it decreases with increasing distance to the sensor's positions ($p_1(10, 10)$,$p_2(10, 0)$,$p_3(0, 10)$,$p_4(0, 0)$). But compared to the theoretical signal propagation model (equation 4.2), we often encounter systematic outliners which do not fit in the theoretical model. This phenomena is common in LQI measurements (noise, other sources of interferences), but it will negatively effect our calculation later [Blumenthal et al., 2007].
We will present the different functions and their impact on the location determination in the later sections. Note that the highest LQI value of every sensor is not equal. We had therefore calculate a mean curve, so that we could create a 'best-fit' theoretical model for our purpose (we will also do this in the RSSi-approach).

Table 4.2 summarizes the key indicators of each LQI-plot: the mean, the maximum and the minimum values of all four sensor-reading matrices.

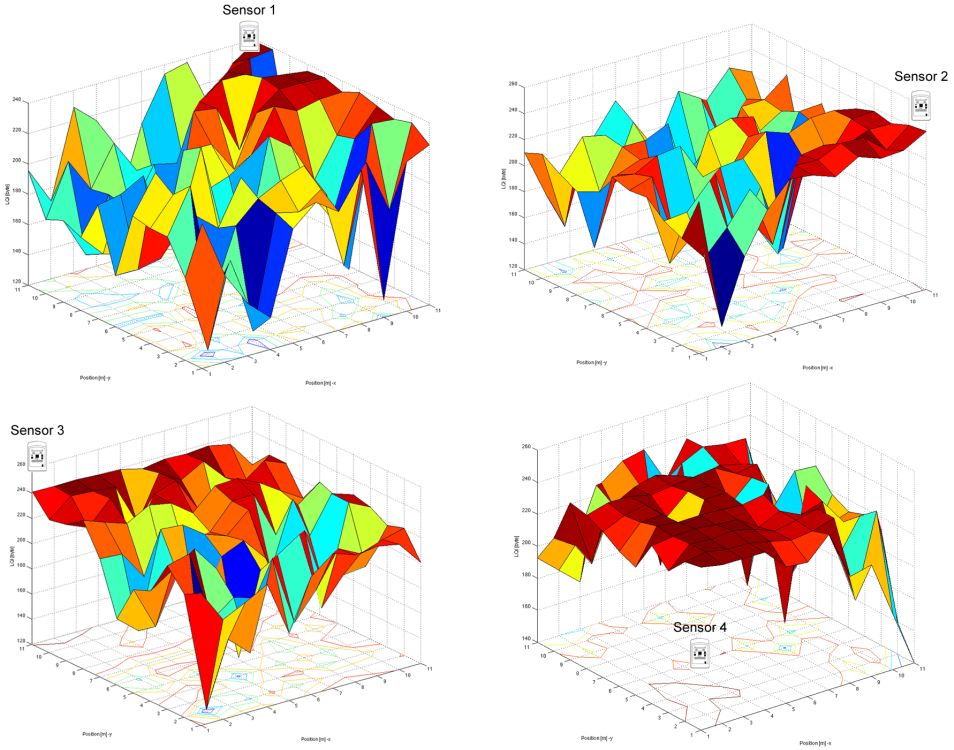

Figure 4.15: Plot of all 4 LQI matrices ($m_{LQI_i}, i \in \{1, 2, 3, 4\}$) extracted from our data set. Each cell indicates an average value taken from 120 measurements at the measurement-point $M_{i,j}$.

As we can see the values in table 4.2, the minimum values vary most. So it is difficult to map the various LQI values correctly to a distance which satisfies all sensor-readings. For this reason we calculated the mean LQI curve to get the average trend of all LQI-readings.

Sensor one, physically located in the upper right corner, clearly shows a common behavior of the LQI development over distance. Note that the LQI readings in the right half of the sensor ($S4 : S2$) contain higher LQI values than the left half of the sensor. We also collected very high readings at the diagonal of the area (direct line between $S1$ and $S4$).

Regarding the second sensor $S2$, located in the bottom right corner - we learned that the diversity of the readings is very low, resulting in high readings even if the distance increases. That makes the position calculation based on these sensor readings very difficult. Distances usually farther away from the sensor node will be mapped closer to the node, because of the high LQI readings. For example

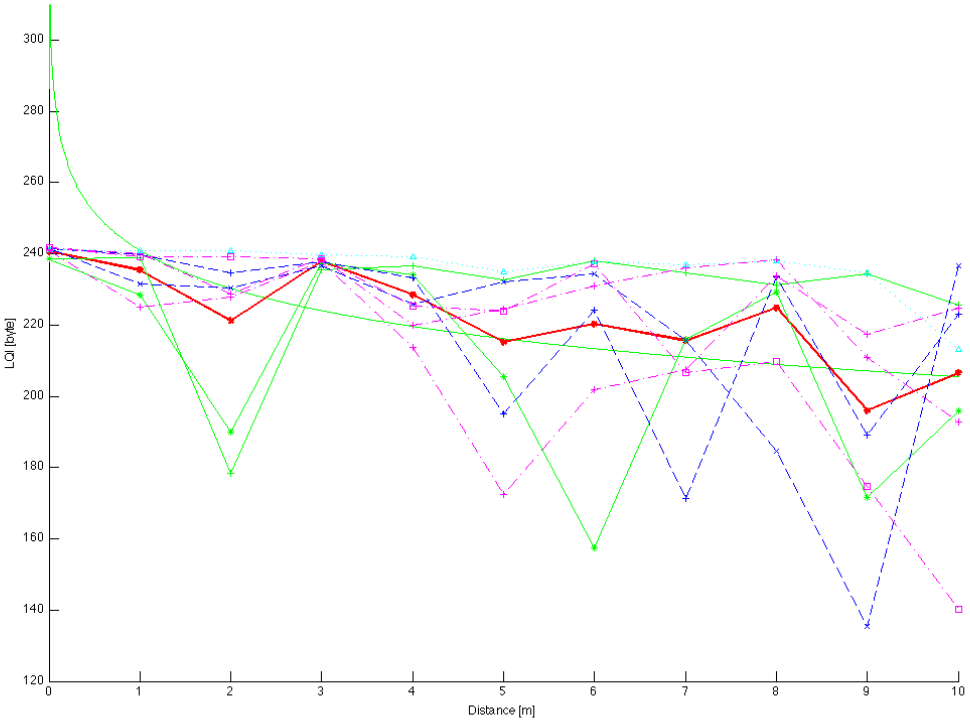

Figure 4.16: Sensor LQI-readings along main-axes (2 axes per sensor in x and y-direction), mean-curve (red) and best-fit logarithmic curve derived from the theoretical model [Blumenthal et al., 2007].

at $m_{LQI_2}(4,7)$ we get an LQI value of 231.8 (note that this measurement data is $8.06m$ away from the sensor) and at the origin where the LQI is at its peak we get $m_{LQI_2}(10,0) = 241.4$. In our theoretical curve at $d = 8.0$ we have set a value to $lqi = 209.8$. So this high LQI - reading will be misinterpreted and mapped at a closer distance to the sensor.

If we take a closer look at sensor 3's $S3$ readings (this sensor is located at the upper left corner), we find a similar picture as in sensor 2's $S2$. Most of the LQI readings are very high and not well distributed over the area, also resulting in an incorrect distance and further inaccurate calculation.

Sensor 4's plot shows us the most inaccurate results among the 4 total sensor plots regarding the LQI measurements. Here we find many high LQI readings, even though we are far away from the sensor's location. This sensor will have a major impact on the weight-functions gathered during the location-calculation with the

	$S1$	$S2$	$S3$	$S4$
$max(LQI)$	239.85	241.43	241.06	241.8
$min(LQI)$	122.43	135.52	121	140.32
$mean(LQI)$	198.1473	209.3329	212.3865	223.2686

Table 4.2: Key-indicators of each LQI-plot: the mean, the maximum and the minimum values of all four sensor-reading matrices.

result that positions tend to be closest to this sensor in the bottom left area (we will depict this phenomena later in this section).

The four plots of each sensor show that LQI values will not be well distributed over distance, resulting in a very inaccurate transfer to distance and further an imprecise calculation of the exact position.

WCL Algorithm with LQI Measurement Data

In this section we perform the WCL algorithm implemented by a MATLAB script (`CalculateDistancePos.m`). This script takes four parameters, a reference-model (M_{Ref}), the value for g (signal propagation parameter defined in equation 4.8) and four measurement matrices (those for $S_1, ..., S_4$). As mentioned in the introduction chapter of the RF-based approach (see 4), we used a theoretical formula (presented in [Behnke and Timmermann, 2008]) for the LQI values (equation 4.10) with a transmission range R_C of $10m$. We adapted this formula for our needs, so that the logarithmic curve best fits our sensor readings from the Sun SPOT device.

$$LQI = \begin{cases} 255 & d = 0 \\ 7.7 \cdot \left(\ln \left(\left(\frac{R_C}{100 \cdot d} \right)^2 \right) + 35.9 \right) & 0 < d \leq 10 \\ 0 & d > 10 \end{cases} \tag{4.10}$$

This theoretical model represents our empirical measurements quite well and produces the best results in our calculation process (we repeated the localization calculation based on LQI certain times to empirically estimate these values). First of all we want to examine the overall Euclidian error distance which is calculated by the formula $\sqrt{(x - x')^2 + (y - y')^2}$.

The figure 4.17 depicts the overall error-distance. As pointed out at the interpretation of the sensor reading plots, we find out that the localization error is at its minimum in the bottom left area near sensor $S4$. This sensor provides the highest

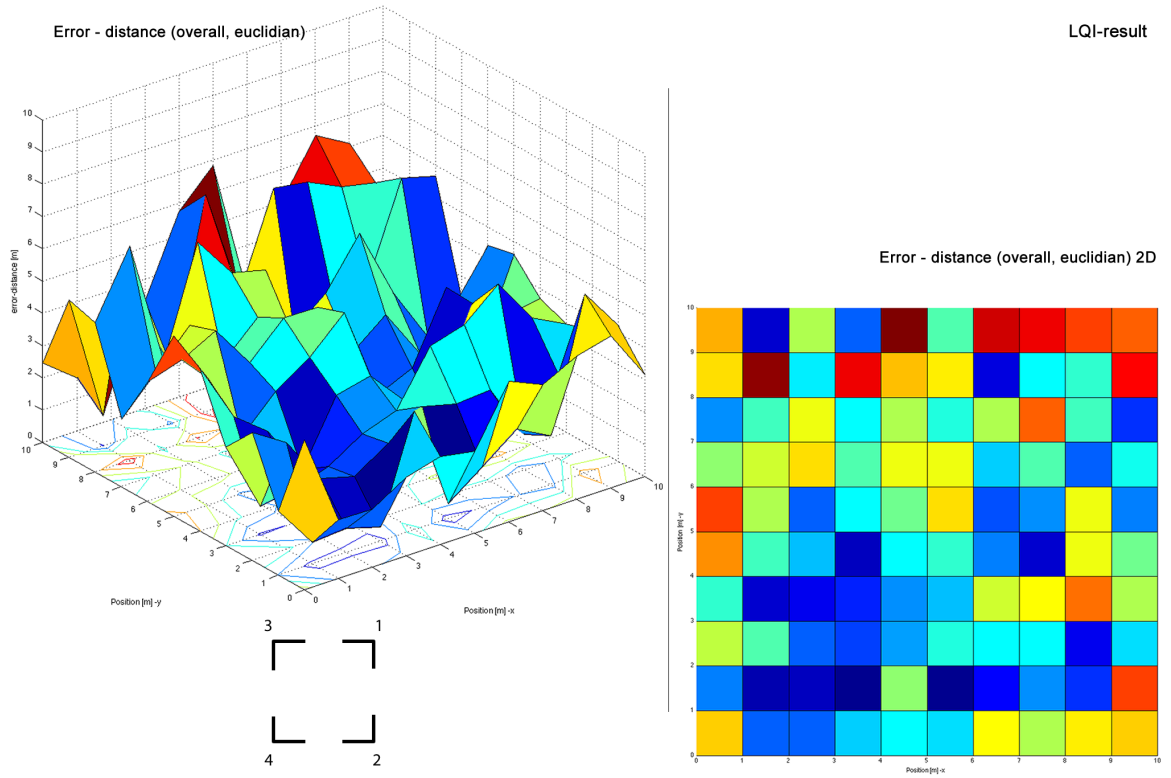

Figure 4.17: Euclidean Error Distance in m ($\sqrt{(x-x')^2+(y-y')^2}$).

LQI readings among a very large area thus resulting in large weights. The error drastically increases while moving away from the sensor's position as the error between calculated and real position rises constantly. We also see that the position calculation in the right bottom corner at sensor 2's $S2$ position performs quite well, because of the high LQI readings around the sensors-position.

In contrast, one can see that the error at sensor 1 $S1$ and sensor 3 $S3$ compared to the other sensors is very high (especially at $S1$). Figure 4.18 depicts the calculated versus the real positions as directional vectors to show the calculation outcome.

If we take a closer look at the error-distances in the x direction (figure 4.19) we see that the error-rates become high when we get closer to sensors $S2, S1$. It reaches its peak at sensor 1. Also note that the error at the border between $S3, S4$ become very high. We have a maximal error of $err_{max_x} = 6.11[m]$, a minimum error of $err_{min_x} = 0.03[m]$ and a mean-error of $err_{mean_x} = 2.21[m]$.

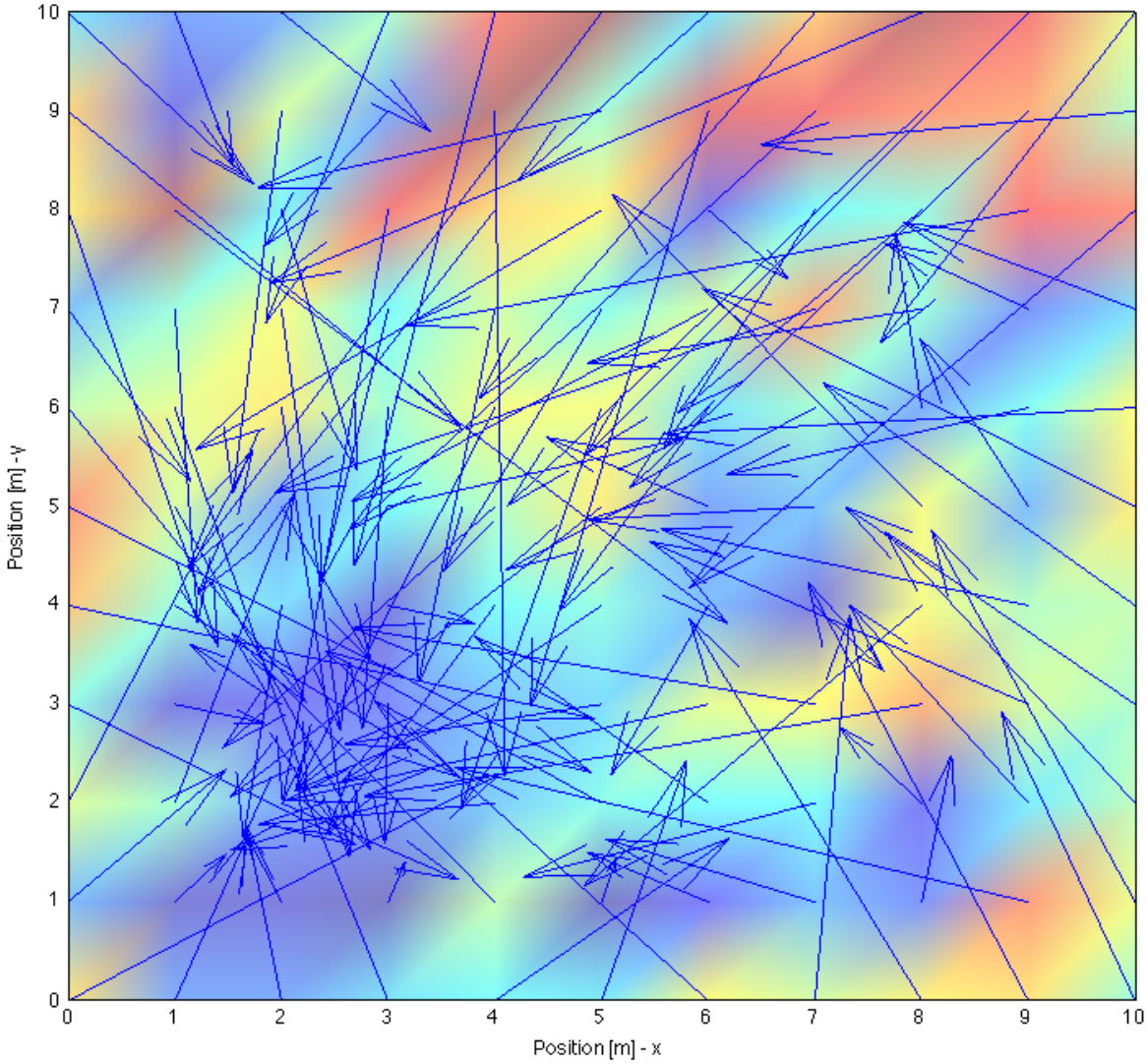

Figure 4.18: LQI: Velocity-Plot of the Euclidean Error Distance in m ($\sqrt{(x-x')^2+(y-y')^2}$).

The y-direction shows a different picture (figure 4.19), because the error rates rise between the area of sensor 1 $S1$ and sensor 3 $S3$. In the lower half between $S4, S2$ the error rates are not that high except at the bottom border of the area near sensor 2. We have a maximal error of $err_{max_y} = 6.73[m]$, a minimum error of $err_{min_y} = 0.13[m]$ and a mean-error of $err_{mean_y} = 2.22[m]$.

To improve the location-inference we've implemented a simple filter-approximation ($\overline{LQI}$) and performed the WCL algorithm again with the same parameters as mentioned before. Note that the analysis above is based on the normal WCL-algorithm without any filtering to show how this algorithm performs

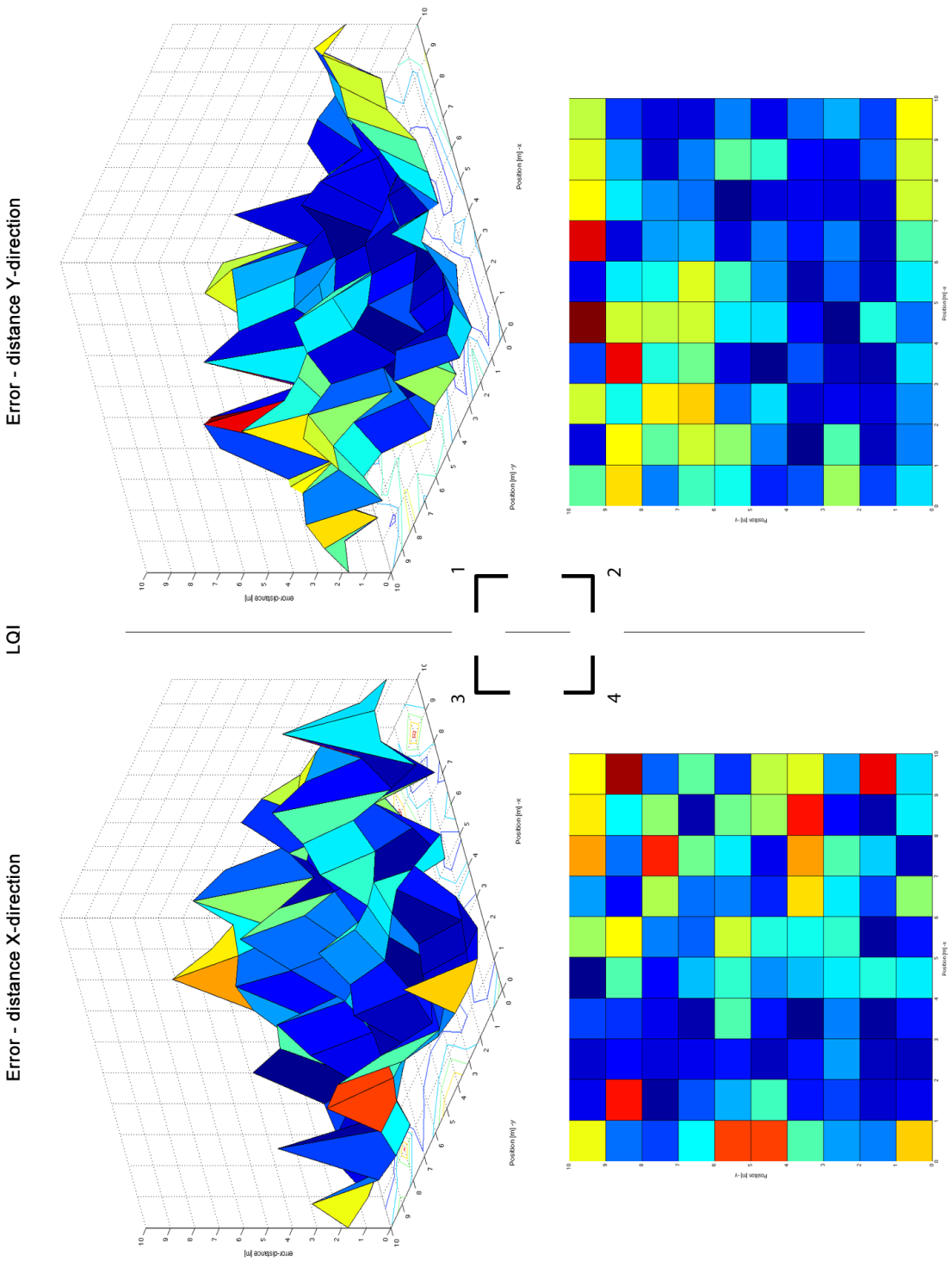

Figure 4.19: Error Distance in m (on the left $|(x - x')|$) and on the right $|(y - y')|$).

without any improvements.

Equation 4.11 shows the filter formula we used. In the algorithm we set the parameter $a = 0.75$ which is a recommended value from [Aamodt, 2006] and seems to be a good value for the approximation.

$$\overline{LQI_n} = a \cdot LQI_n + (1 - a) \cdot LQI_{n-1} \tag{4.11}$$

The table 4.3 shows all results taken from the position calculation and compares the selection of different parameters. One can see that a higher value of g results in an inaccurate location detection and higher error-rates. We varied $g = 1(1)3$ to demonstrate how the localization performance develops based on this parameter. Further we have included the calculation results by using a the simple averaging filter ($\overline{LQI}$).

	max	min	mean	xMax	xMean	yMax	yMean
LQI, $g = 1$	6.8271	0.4414	3.3526	6.1100	2.1235	6.7288	2.2245
$\overline{LQI}$, $g = 1$	6.9634	0.6876	3.2986	6.1376	2.1658	5.7746	2.1440
LQI, $g = 2$	8.6253	0.2342	3.8091	7.6037	2.3924	8.2435	2.4573
$\overline{LQI}$, $g = 2$	7.9039	0.3725	3.8202	7.5708	2.4582	7.4893	2.4524
LQI, $g = 3$	10.0494	0.0463	4.0938	8.4473	2.5703	8.7869	2.6156
$\overline{LQI}$, $g = 3$	8.8669	0.0463	4.1306	8.0342	2.6256	8.0099	2.6379

Table 4.3: Localization Computation Results Based on LQI, normal calculation and averaging filter results with a varying value for parameter g. Unit for this rates is $[m]$

After examining the LQI calculation results, we found out that the best method is the use of a simple average filter noted as $\overline{LQI}$ and the parameter set $g = 1$. With this configuration we achieved an Euclidean mean error of $3.3m$ (33% error rate) and a Euclidean maximum error of $6.96m$ (69.63% error rate), which can be found in the second row of table 4.3. Note that the maximum error-rate is still higher than with the use of no averaging. But if we take closer look at figure 4.20 we see that the error-distribution of the use with the average-filter is slightly better than without. This figure contains the Euclidian-error-rate distribution and the important percentiles (0.25, 0.5, 0.75 and 0.9) are highlighted with black lines.

RSSi Measurements

In this section, we will present our RSSi measurement series and analyze it in great detail to get some understanding of the behavior of the RSSi regarding distance/-

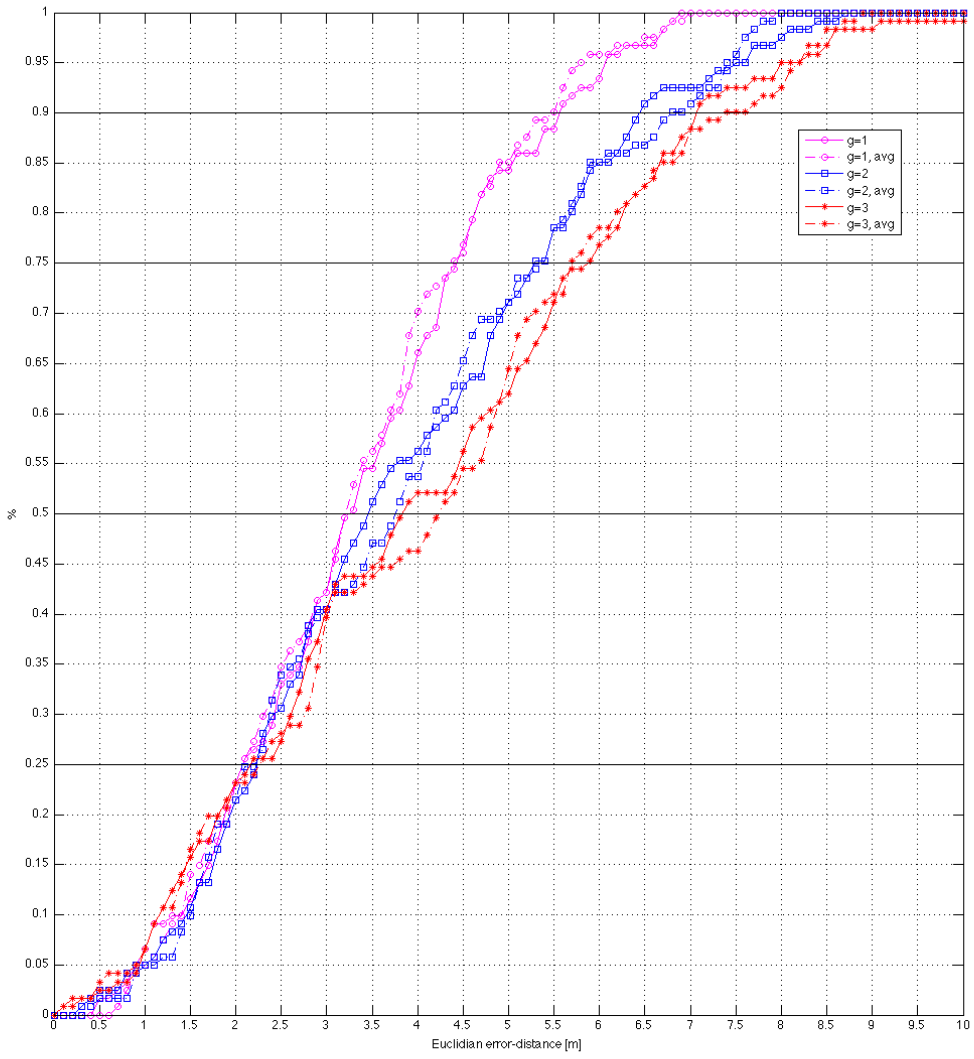

Figure 4.20: Error Distance Distribution of the LQI Method with Varying Parameter g and Filter Methods.

position. As stated in the previous LQI section we define p_x as position coordinates for every sensor S_x (e.g. p_1 is the position for S_1).

The plot (Figure 4.21) contains the RSSi plots for all four sensors $S_i, i \in \{1, 2, 3, 4\}$. A small sensor picture was placed to show the location of the sensor position on

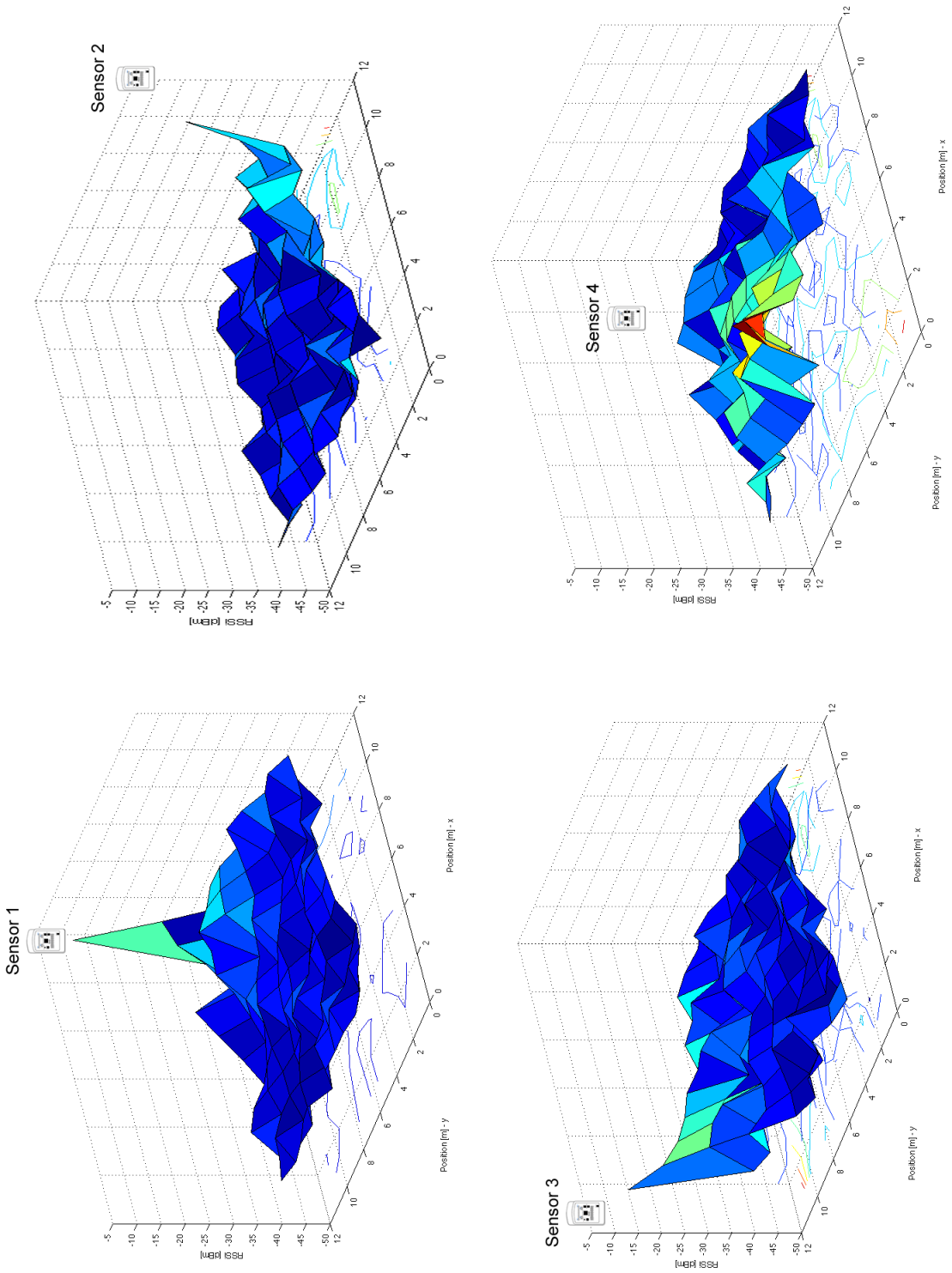

Figure 4.21: Sensor 1 to Sensor 4 RSSi Plot.

the grid.

All four plots represent the common behavior of the RSSi as it is decreasing with increasing distance to the sensor's position ($p_1(10,10)$,$p_2(10,0)$,$p_3(0,10)$,$p_4(0,0)$). Note that compared to the theoretical signal propagation model (equation 4.2), we encounter some systematic outliners which do not fit in exactly. Though this is a regular behavior of the RSSi in practice, noise and other sources of interferences, will later negatively effect our calculation. This is due to the fact that the point-fitting in our RSSi-to-distance curve (smallest distance to the theoretical curve) will not always be correct. Some points will be mapped farther away and other points closer to the sensor's position, resulting in an inaccurate distance value and further in an inaccurate weight-function result. We used several transfer functions to improve the RSSi-to-distance mapping function to get a better calculation result. We will present the different functions and their impact on the location determination in the later sections.
Note that the highest RSSi values of each sensor are not equal. We, thus, had to calculate a mean curve to create a 'best-fit' theoretical model for our purpose. Table 4.4 contains the key-indicators of each RSSi plot i.e. the mean, the maximum and the minimum values of all four sensor-reading matrices.

	$S1$	$S2$	$S3$	$S4$
$max(RSSI)$	-9.5700	-15.7900	-12.4300	-17.1400
$min(RSSI)$	-42.7800	-44.2500	-45.2600	-44.3400
$mean(RSSI)$	-37.7163	-39.1137	-39.1097	-36.3648

Table 4.4: Key-indicators of each RSSi-plot: the mean, the maximum and the minimum values of all four sensor-reading matrices.

As we can see, the values are not too far away from each other, showing us a good distribution and measuring performance during the experiment. This provides a good data set for the upcoming WCL algorithm calculation procedure.

WCL Algorithm with RSSi Measurement Data

As mentioned in 4.4.2 we also performed the WCL algorithm on our RSSi data set. For our RSSi reference model we used the following equation 4.12:

$$RSSI = -(3.6 \cdot 3.0 \cdot \log_{10} d + 30.2), [RSSI] = dBm \tag{4.12}$$

Note that we used the following parameter configuration $n = 3.0, A = 30.2, m = 3.6$ [Aamodt, 2006] for our reference model (equation 4.2) and used the

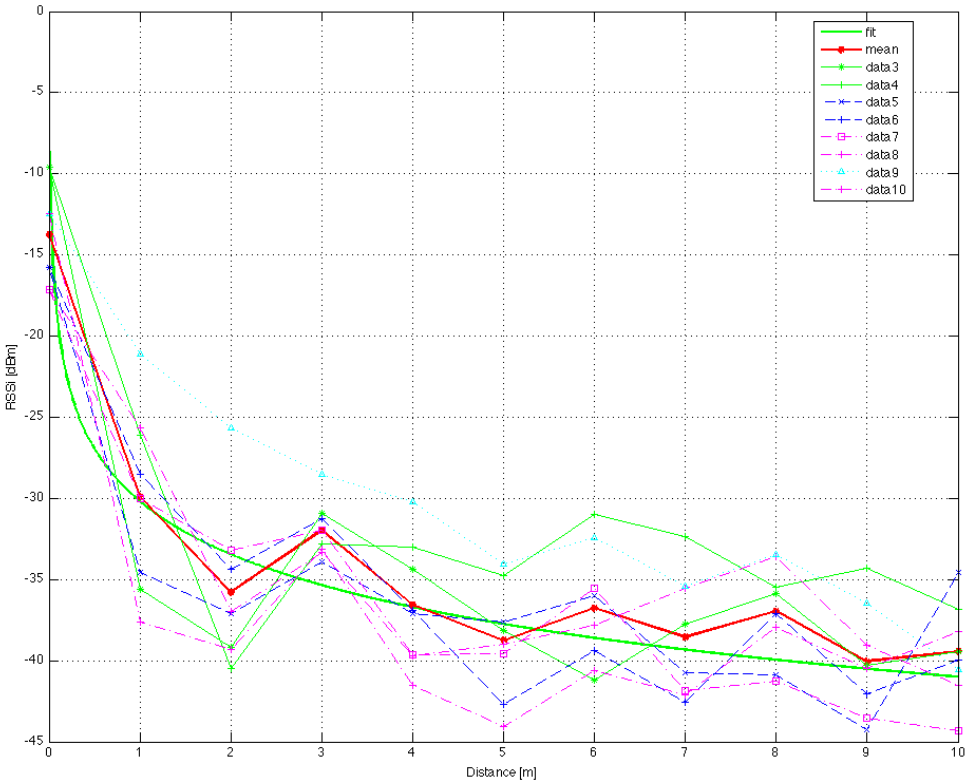

Figure 4.22: Measured RSSi Curves. Two curves per sensor were measured (in x and in y direction). Of the eight directional curves, we calculated the mean curve and their variance curve. The mean curve is depicted in red. The green curve is our best-fit theoretical curve. Equation 4.12 contains the exact parameter configuration.

`CalculateDistance` MatLab script for our empirical estimation process. The values for A, n and m have been empirically determined by repeating the calculation process as long as we have minimized the error distance (90% percentile) for the localization calculation distribution of the Euclidian error rate. Besides, this new model fits best the natural behavior of the Sun SPOT signal propagation (based on our measuring-series and the experiment itself). Figure 4.22 shows a combined plot of the main measurements series of each sensor $S1, ..., S4$, the mean-curve and the fitted theoretical curve (our empirical model, best-fit curve).

Figure 4.23 shows the result of the WCL algorithm position calculation in a 11×11 matrix P_{RSSi} . This plot shows the Euclidean error distance between the real position $p_{real}(x, y)$ and the calculated position $p_{calculated}(x', y')$. Each plot, if not

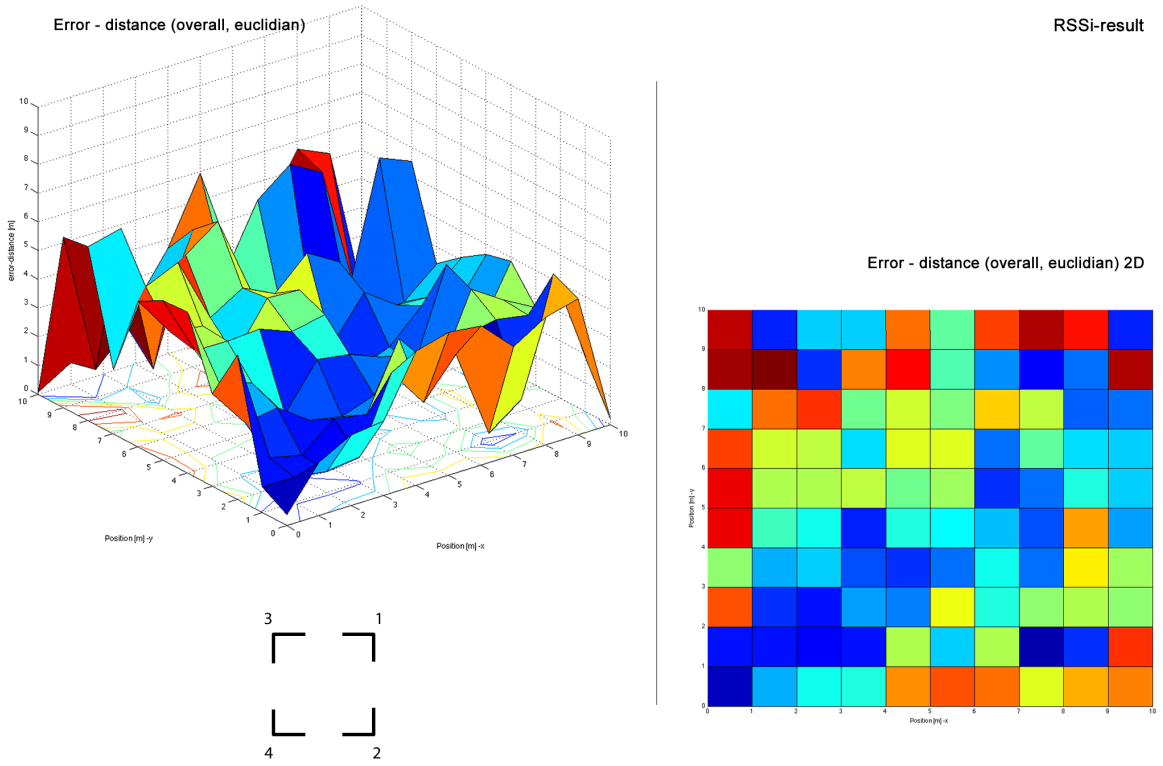

Figure 4.23: Euclidean Error Distance in m ($\sqrt{(x-x')^2+(y-y')^2}$).

denoted otherwise, is the result of a WCL algorithm with parameter $g = 1$, with no averaging. Later we learn that averaging is slightly more effective, but for now we want to show the regular results of the performance of the WCL algorithm without any filtering.

As we can see, the location calculation performs quite well in the middle of the area (blue-fill) and produces the most errors at the borders of the area (red and yellow fills). This is due to the fact that the RSSi values are decreasing with increasing distance to the sensors. But in practice the unknown node U is still close to a sensor, but is not recognized by the algorithm as 'close' (while moving at the borders) because of a weak RSSi reading. There is a so-called 'critical Range' in the empirical RSSi curve as stated in [Blumenthal et al., 2007]. When we examine the mean curve depicted in figure 4.22, we see that at distance $d = 2$ the RSSi value drops suddenly compared to the value measured at distance $d = 3$. Due to this, values between that range ($d_{critical} = [1; 3]$) have a high probability to result in high error-distances (red fills in the error-distance figure). We have a maximal error of $err_{max} = 6.34[m]$ (63.4% error), a minimum error of $err_{min} = 0.03[m]$ (0.3%) and

a mean error of $err_{mean} = 2.91[m]$ (29.14%). In the later conclusion subsection we examine the error rates in more detail.

To get an idea of why the error increases while getting closer to the border we will examine the error of x- and y-axis in separate.

For the x-axis a good localization-performance of the unknown device U could be achieved with a maximal error of $err_{max_x} = 5.86[m]$ (58.64%), a minimum error of $err_{min_x} = 0.02[m]$ (2.29%) and a mean-error of $err_{mean_x} = 1.69[m]$ (16.89%). In Figure 4.24 we can see that the error increases by reaching the border of our experimental test-area. The highest error-rates can be found on the bottom border between $S3$ and $S4$ and between the critical range. So we can summarize that the localization performance works quite well in the middle of the observed area and produces higher error-rates at the border.

The y-error rates are not as systematic as those for the x-axis. They show in part the same trend of creating high error-rates at the borders and between the critical-range but also provide higher error-rates as we get closer to the middle of the observed area. Especially at sensors S_4 and S_1 area, we encounter the highest error-rates. We've got following performance indicators: maximal error of $err_{max_y} = 5.25[m]$ (52.49%), a minimum error of $err_{min_y} = 0.02[m]$ (0.2%) and a mean-error of $err_{mean_y} = 2.07[m]$ (20.66%)

As introduced in the LQI based WCL-calculation we have also used a simple filter-approximation to test the performance of filtering in this context 4.13.

$$\overline{RSSI_n} = a \cdot RSSI_n + (1 - a) \cdot RSSI_{n-1} \tag{4.13}$$

The table 4.5 shows all results taken from the position calculation based on RSSi and compares the selection of different parameters. Like in the before mentioned LQI results one sees that a higher value of g results also in an inaccurate location detection and higher error rates compared to the standard calculation scheme without filtering, except for the choice of parameter $g = 1$ and average-filter. In that case noise and other negative interferences will be smoothed, causing a slightly better result. Results for the simple average filter are denoted as ($\overline{RSSI}$).

To illustrate the distribution of the error distances we also plotted the result in figure 4.25. Again we learned that the use of a simple filter method and the parameter set $g = 1$ proved to be the best choice.

In figure 4.25 we see that we have a 50% chance to get an error of $2.76m$ which is 27.6% error in our $10m \times 10m$ test-area. Table 4.6 shows all details on the

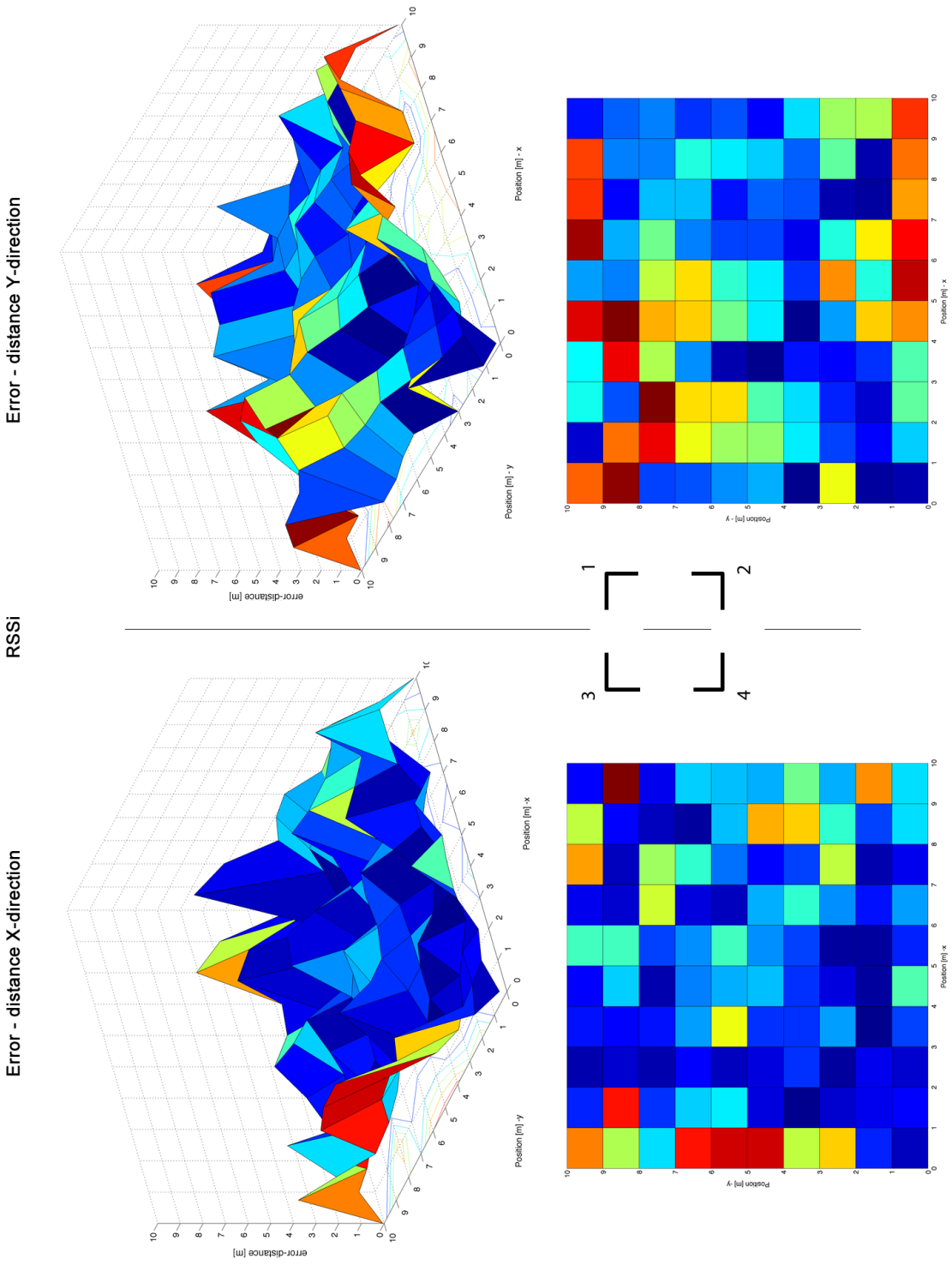

Figure 4.24: Error Distance in m (on the left $|(x - x')|$) and on the right $|(y - y')|$).

	max	min	mean	xMax	xMean	yMax	yMean
$RSSI, g = 1$	6.3394	0.0334	2.9147	5.8642	1.6891	5.2489	2.0657
$\overline{RSSI}, g = 1$	6.0495	0.0767	2.8415	5.3924	1.6832	5.1351	2.0118
$RSSI, g = 2$	8.0193	0.0000	3.4884	7.6367	1.9898	7.1088	2.3856
$\overline{RSSI}, g = 2$	7.7564	0.0002	3.3398	5.9434	1.9519	7.0293	2.2979
$RSSI, g = 3$	9.5264	0.0000	3.9700	8.5359	2.3413	7.7789	2.6599
$\overline{RSSI}, g = 3$	8.7198	0.0000	3.8237	7.0642	2.2826	7.7477	2.5561

Table 4.5: Localization Computation Results Based on RSSi: normal calculation and average filter results with a varying value for parameter g. Unit for these rates is [m].

distribution plot and the percentiles.

	$p = 0.25$	$p = 0.5$	$p = 0.75$
$RSSI, g = 1$	1.4954m	2.7573m	3.9025m
$\overline{RSSI}, g = 1$	1.5835m	2.6440m	3.7184m
$RSSI, g = 2$	1.9766m	3.0645m	4.9764m
$\overline{RSSI}, g = 2$	2.1401m	3.1806m	4.6008m
$RSSI, g = 3$	2.2874m	3.5933m	5.5590m
$\overline{RSSI}, g = 3$	2.1261m	3.7293m	5.2891m

Table 4.6: Distribution Percentiles of the Error Rates with Varying Parameter g and Average Filtering denoted as $\overline{RSSI}$. Unit is [m]

To see how the location-inference performs graphically with the WCL-algorithm, figure 4.26 provides deeper insight into the positioning of the measurement points. The dots denote the real position whereas the corresponding vectors point to the calculated position by the WCL. Note that we did not scale the vectors, so they represent the real calculated position.

Comparison between the RSSi- and the LQI Method

After evaluating both the RSSi and the LQI-based WCL algorithm we point out that the RSSi method seems to be the more accurate method of the two. Table 4.7 shows the error differences taken from both best-result methods of LQI and RSSi (RSSI and LQI with average filter and $g = 1$).

The result shows us that the RSSi method surpasses the LQI in any case. Also the x coordinate and y coordinate calculation shows that the RSSi method is the

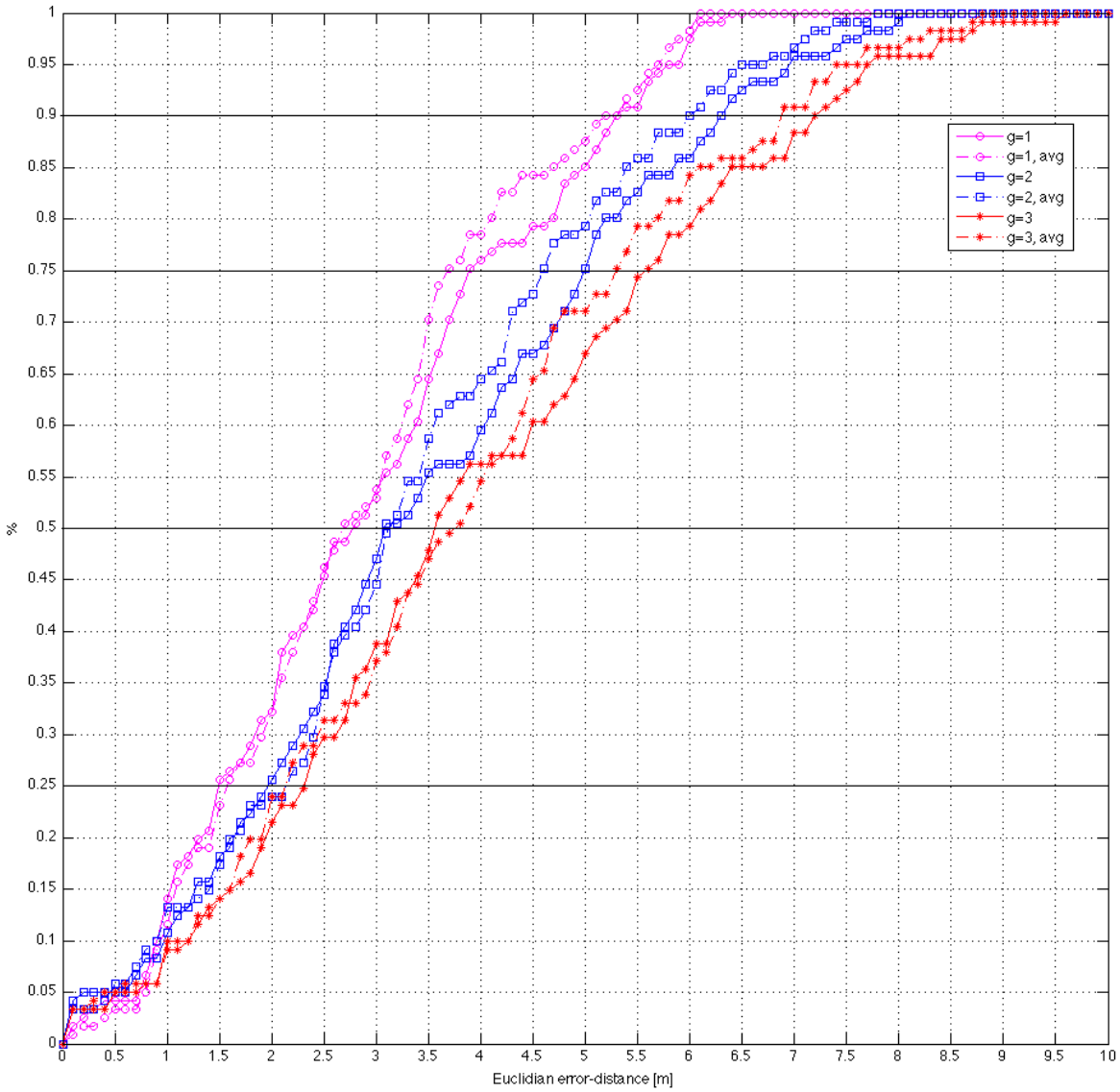

Figure 4.25: Error Distance Distribution of the RSSi Method with the Varying Parameter g and Filter Methods.

better choice by surpassing both max and mean errors by 13.82%, 28.67% and 12.45%, 6.75%. The overall localization performance from the RSSi method based on the Euclidian error distance is at maximum 15.11% and in mean 16.09% better than the LQI method. In summary, of the two methods, we would choose the RSSi-method for location inference. We also point out that we cannot confirm the statement by [Blumenthal et al., 2007] that the LQI is a better indicator for calculating the distance (with respect to the fact that we used the Sun SPOT device

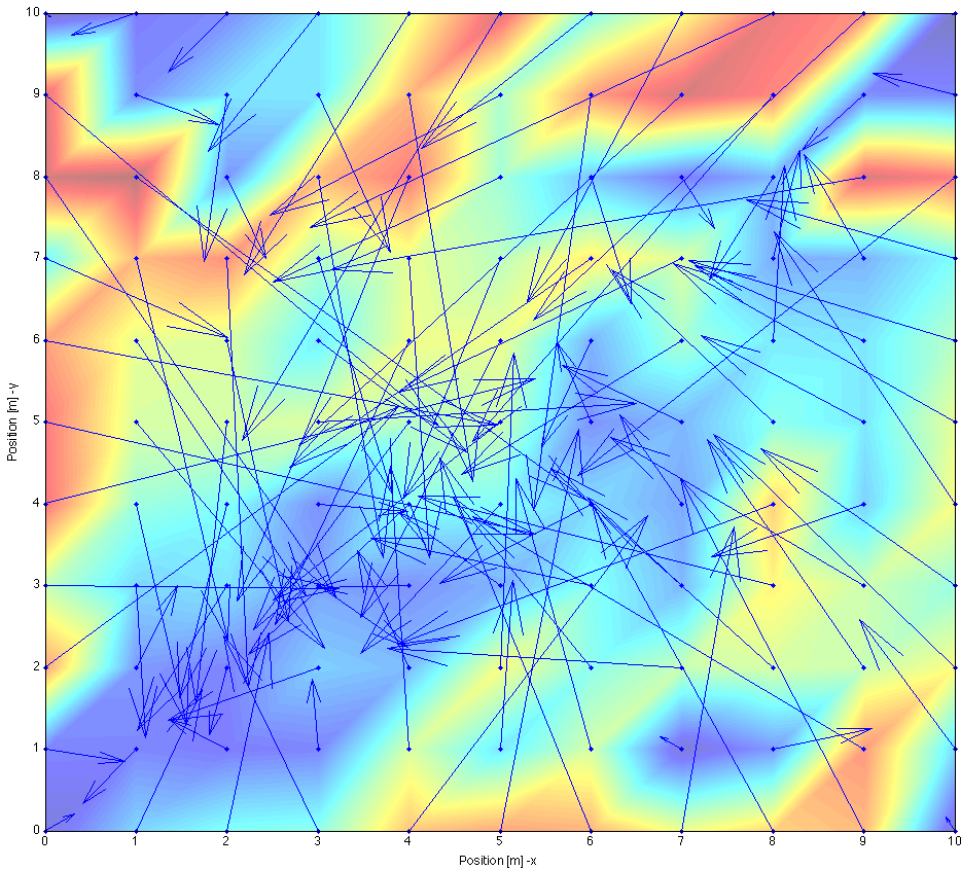

Figure 4.26: RSSi: Calculated vs. Real Position of the Unknown Device: the arrows point to the calculated position.

	max	min	mean	xMax	xMean	yMax	yMean
$\overline{RSSI}, g = 1$	6.0495	0.0767	2.8415	5.3924	1.6832	5.1351	2.0118
$\overline{LQI}, g = 1$	6.9634	0.6876	3.2986	6.1376	2.1658	5.7746	2.1440
$\triangle\overline{LQI} - \overline{RSSI}$	0.9139	0.6109	0.4571	0.7452	0.4826	0.6395	0.1322
$\triangle$ %	15.11%	796.48%	16.09%	13.82%	28.67%	12.45%	6.57%

Table 4.7: Localization Computation Comparison between the RSSi and LQI Methods: the RSSI method surpasses LQI in each situation.

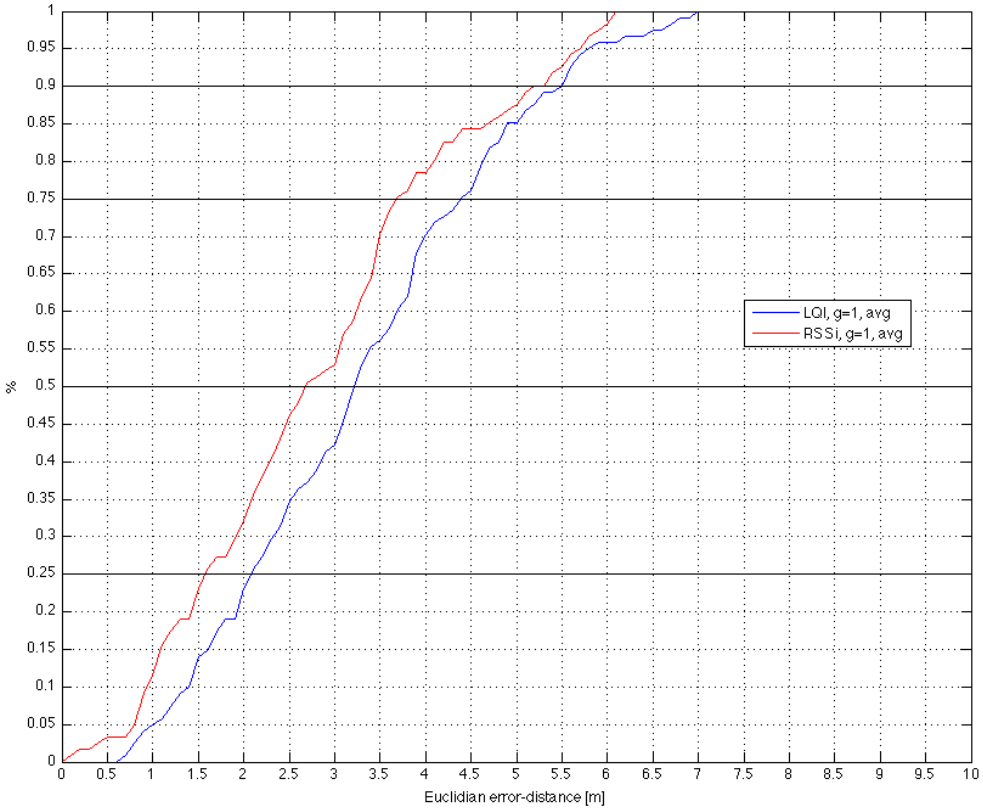

Figure 4.27: Error Distribution of Euclidian Error Rates. Comparison between the RSSi method (red) and the LQI method (blue). Both use a parameter $g = 1$ and averaging.

and we have a significantly smaller experimental test-bed area). As mentioned in [Blumenthal et al., 2007] our experimental test-area is in the critical-range, though yielding imprecise results compared to their approach.

Conclusion

After our research on the RF-based approach we found out that the best method to calculate the unknown's position is with RSSi. Figure 4.27 summarizes the result of the two approaches by depicting the error distribution of the Euclidian error rate of the RSSi and the LQI method (though only the best results are displayed by using a parameter $g = 1$ and averaging).

At this point we want to remind of our requirements for a smart-home location detection. Error rates greater than $1m$ are not acceptable in our context, though our best method, the RSSi, fails to satisfy these requirements sufficiently as we have an error distance of maximum $5.2m$ (that is a 52% error rate in our test-area) with a probability of 90%. We also tried to improve our location-inference by developing a sort of dynamic WCL algorithm, which takes preceding positions into account and varies between the variances of a certain key-performance indicator value (LQI, RSSi). But again, we found out that the data provided from the device itself (raw values) is not as accurate as needed for the calculation. For larger environments (like outdoor-environments, where larger error-rates are not that critical), or for a rough location inference, the RF-based approach may be a good, cost-effective and easy-to-develop approach. But in our context of 'smart-home location-inference' these error-rates are not acceptable. Queries such as 'is user next to object A?' or similar ones are not feasible with such error rates.

These error rates are yielded perhaps due to the Sun SPOT's poor antenna design. The research from [D'Andria, 2010] showed that the direction of the Sun SPOT or the presence of obstacles near the Sun SPOT device may yield very unstable values for both RSSi and the LQI. Taking this into consideration would make the calculation more erroneous than it is now. Note that we used a stationary Sun SPOT as an unknown device and moved it along a defined grid. So there were no obstacles between the devices nor did a human hold the unknown device. A better device design with an omnidirectional antenna could make better results without being effected by the direction or position in which it is held. But this is work for the future.

Chapter 5

Summary and Future Work

This master's thesis evaluated two different approaches to determine the user 's current location in a smart-home environment based on the Sun SPOT's technological capabilities. First of all, we tried to calculate the distance / position with an accelerometer device. This device can measure the current acceleration an objects experiences. Based on the principal laws of physics, we tried to derive the position with very basic equations. In theory, this model works well and is very easy to set up, though just one device has to be carried by the user (today's smart-phones often include such an accelerometer) and the calculation-process is not time-consuming. But our three different experiments have shown that the provided measurement values from the LIS3L02AQ accelerometer are far too imprecise to derive the position $x(t)$ at any time t.

We also tried to apply various filters such as averaging or to interpolate the measurement values, but non of these mentioned practices showed satisfactory results. An error rate reaching from at least 39.5% to maximum error rates around 1230% is more than common, rendering them unusable in any location inference component (either outdoor or indoor).

Another negative influence in the inference of the location is the fact that the error gained from preceding calculations will propagate and lead to even higher error-rates. With our prototype application we showed that if we move an object for only $1m$ and do not apply any other movements afterwards, but let the calculation process continue, the error-distance drastically (exponentially) increases in a short period of time (e.g. 200 m after 2 minutes of calculation but moving the object for only $1m$). Besides the erroneous data and the error propagation, we also learned that if we use this approach in a real smart-home environment, more considerations about the calculation have to take place. It is not a requirement to

hold the device in a predefined position. Our experiments had very restrictive requirements for the positioning of the device.
In practice, the orientation can change any time, thus making it very hard to calculate the position based on the axes of the devices' gained accelerations (a_x, a_y, a_z). This is due to the fact that the axes of the devices may not correlate with the axes of the user. Of course, there are possibilities to correct that and align those two axes-systems based on the tilt of the device. But any additional calculation is more time and cpu-performance consuming. So it will be hard to achieve an almost real-time calculation model based on this approach if the data were even more accurate. A suitable alternative may be provided by a new gyroscope device. This device is able to measure acceleration along any axes with respect to the orientation in which the device is held. Future work could evaluate the use of a gyroscope in state-of-the-art smartphones for the indoor-location-inference. Nevertheless, the accelerometer does offer some interesting fields of application in a smart-home environment like gesture-detection or downfall-recognition systems. Even weak vibrations can be recognized by the accelerometer, which may be useful for matters of security.

The second approach, the Radio Frequency Approach was the main focus of this thesis. For the first part, we did not need to create a great experimental testbed. Only a predefined grid and a robot for the last experiment were needed. But for the RF-approach, we had to set up a large test-environment and create several restrictions and requirements.

There were several papers and contributions referred to such as [Blumenthal et al., 2007], [Aamodt, 2006], [Bahl and Padmanabhan, 2000b] and [Behnke and Timmermann, 2008], which presented location determination methods for indoor environments, but none of these pointed out the performance in smaller objects (like a single room).

We evaluated the performance of two distinct key performance indicators (KPIs): first, the received signal strength indicator (RSSi) and the link quality identifier (LQI). For the calculation model, we used the weighted centroid localization algorithm (WCL) described in [Blumenthal et al., 2007], but used the RSSi and the LQI for our evaluation. The WCL is a very robust algorithm which is not resource consuming. It can be easily deployed on tiny devices (embedded devices) with low processing power and less memory. For our prototype software we developed an extensible architecture which allows the participation of several reference-nodes R_i and several unknown nodes U_i. This architecture operates distributed on a service-basis allowing new reference-nodes and unknown-nodes to participate at any time without prior knowledge of the environment. The data needed for the location-inference will be gathered together on a core-system where the calculation takes place.

We examined the signal-propagation of both the RSSi and LQI indicators in great detail and also provided theoretical models for both indicators. On that basis we performed our location inference calculation.

The LQI approach and the RSSi approach both yielded unsatisfying results, thus not useable in our smart-home-environmental context. At this point, we want to remind of our requirements for a smart-home location inference component, which is a maximum error distance of $1m$ in any case. Out of the two evaluated methods, we found that the RSSi method surpassed the LQI method in every aspect. Though yielding an error distance of $3.7814m$ (75%ile) and $5.2m$ (90%ile), which is unacceptable in our context. Further, we adapted the WCL algorithm with a dynamic component, where we tried to take preceding calculation results into account and vary between the variances of a measured value. This unfortunately led to even more erroneous calculation results.

The high error-rates are the result of imprecise values provided by the Sun SPOT radio-interface. Reflections, noise or other sorts of interferences drastically decrease the quality of the received indicators. Erroneous indicator values will be incorrectly mapped via our transfer-function to our theoretical model, resulting in false distances to the reference-nodes. This effects our overall calculation and yields the before mentioned error-distances.

We can summarize that the Radio-Frequency approach is a low-cost and easy-to-setup method for coarse-grained location determination, which can be easily adapted and extended (more reference nodes, more unknown nodes), but it is still unable to perform location determination for fine-grained localization where a precision of less than or equal to $1m$ is required.

5.1 Future Work

The use of the Sun SPOT, though the technology is fairly new, has become obsolete for many fields of application due to the fact that nowadays smart-phones have even more CPU-performance and a richer set of sensors (for instance, state-of-the-art smart-phones already have a gyroscope and a aGPS) and the ability to develop a graphical user interface based on a modern operating system [1].

Many of these platforms already provide powerful developer tools, which make integration in other systems very easy (even for programmers not used to

[1] iOS from Apple Inc., Android from Google Inc., webOS from Palm Inc. and BlackBerryOS from Research in Motion Inc.

embedded devices). Further these devices provide more wireless communication possibilities than the Sun SPOT (for example, most smart-phones have WiFi and Bluetooth).

For future work it would be very interesting to develop a location-inference component based on WiFi with a aforementioned state-of-the-art smart-phone, which can seamlessly integrate in an existing WiFi structure, without initially knowing the network configuration and start to locate the user carrying the mobile device (the WiFi network technology is very common and can be found in many facilities). So if possible, no preconfiguration or large initial setup routine have to be performed to realize the location-inference ability of the network.

If more WiFi base stations are available (as is the case in a shopping mall, or in a museum), an application which can track the current position of the user and provide context-aware information (e.g. shops next to you based on your profile, or a guided tour in a museum) would enhance the user's experience.

Appendix A

Acronyms

AAL Ambient Assisted Living

ADC Analog to Digital Converter

aGPS Assisted Global Positioning System

API Application Programming Interface

AWCL Advanced Weighted Centroid Localization

BS Base Station, Sun SPOT Basestation

CDLC Connected Limited Device Configuration

CL Centroid Localization

CORR Correlation Value

CPU Central Processing Unit

GC Garbage Collector

GPIO General Purpose Digital Input/Output

GPS Global Positioning System

GUI Graphical User Interface

HCI Human Computer Interaction

IEEE Institute of Electrical and Electronics Engineers

ISM Industrial, Scientific and Medical

JVM Java Virtual Machine

KPI Key Performance Indicator

LED Light-Emitting Diode

LQI Link Quality Identifier

MEMS Micro-Electro-Mechanical System

PDA Personal Digital Assistant

RF Radio Frequency

RFID Radio Frequency Identification

RN Reference Node

RSS Received Signal Strength

RSSi Received Signal Strength Indicator

SM4ALL Smart Homes for ALL

SN Sensor Network

SOC System-On-a-Chip

SPOT Small Programmable Object Technology

TI Texas Instruments

USB Universal Serial Bus

UPnP Universal Plug and Play

VM Virtual Machine

VTS Video Tracking System

WCL Weighted Centroid Localization

Bibliography

[Aamodt, 2006] Aamodt, K. (2006). *CC2431 Location Engine - Application Note AN042*. Texas Instruments.

[Alkire, 2007] Alkire, B. (2007). I2c on the edemo board. *Sun SPOT Application Note*.

[Bahl and Padmanabhan, 2000a] Bahl, P. and Padmanabhan, V. N. (2000a). Enhancements to the radar user location and tracking system. *Technical Report*.

[Bahl and Padmanabhan, 2000b] Bahl, P. and Padmanabhan, V. N. (2000b). Radar: An rf-based in-building user location and tracking system. *Proc. IEEE Infocom*.

[Baldoni et al., 2009] Baldoni, R., Mecella, M., and Querzoni, L. (2009). D2.1.a the sm4all architecture.

[Behnke and Timmermann, 2008] Behnke, R. and Timmermann, D. (2008). Awcl: Adaptive weighted centroid localization as an efficient improvement of coarse grained localization. *Positioning, Navigation and Communication, 2008. WPNC 2008. 5th Workshop on*.

[Blumenthal et al., 2007] Blumenthal, J., Grossmann, R., Golatowski, F., and Timmermann, D. (2007). Weighted centroid localization in zigbee-based sensor networks. *Intelligent Signal Processing, 2007. WISP 2007. IEEE International Symposium on*.

[Blumenthal et al., 2005] Blumenthal, J., Reichenbach, F., and Timmermann, D. (2005). Position estimation in ad hoc wirelsess sensor networks with low complexity. *2nd workshop on Positioning, Navigation and Communication (WPNC05) and 1st Ultra-Wideband Expert Talk (UET05)*.

[Bulusu et al., 2000] Bulusu, N., Heidemann, J., and Estrin, D. (2000). Gps-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28– 34.

[D'Andria, 2010] D'Andria, A. (2009-2010). Gather kpi su reti di sun spot. Master's thesis, Universita degli Studi di Salerno.

[Eckert et al., 2009] Eckert, J., Dressler, F., and German, R. (2009). Real-time indoor localization support for four-rotor flying robots using sensor nodes. *Robotic and Sensors Environments, 2009. ROSE 2009. IEEE International Workshop on*, pages 23–28.

[Ergen, 2004] Ergen, S. C. (2004). Zigbee/ieee 802.15.4 summary.

[Goldman, 2007a] Goldman, R. (2007a). Using the at91 timer/counter. *Sun SPOT Application Note*.

[Goldman, 2007b] Goldman, R. (2007b). Using the lis3l02aq accelerometer. *Sun SPOT Application Note*.

[Husinsky, 2009] Husinsky, P. W. (2009). Modellbildung in der physik. Technische Universität Wien, Fakultät für Physik.

[McClellan et al., 1999] McClellan, J. H., Schafer, R. W., and Yoder, M. A. (1999). *DSP First, A Multimedia Approach*. Number ISBN 0-13-243171-8. Prentice-Hall Inc.

[ORACLE, 2012a] ORACLE (2012a). The squawk project. http://labs.oracle.com/projects/squawk/.

[ORACLE, 2012b] ORACLE (2012b). Sun spot - home of sun spot. http://www.sunspotworld.com.

[Simon et al., 2006] Simon, D., Cifuentes, C., Cleal, D., Daniels, J., and White, D. (2006). Java ™ on the bare metal of wireless sensor devices, the squawk java virtual machine. *VEE'06 Proceedings of the 2nd international conference on Virtual execution environments*, ACM 1-59593-332-6/06/0006:78–88.

[SM4ALL, 2012] SM4ALL (2012). Sm4all: Smart homes for all webpage. http://www.sm4all-project.eu/.

[Sun-Labs, 2006] Sun-Labs (2006). Remote control camera. Technical report, Sun-Labs.

[Sun-Labs, 2008] Sun-Labs (2008). *SunSPOT - Owner's Manual Blue Release 4.0*. Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95045 U.S.A., blue release 4.0 edition.

[Sun-Labs, 2009a] Sun-Labs (2009a). *Solariums User's Guide, Red Release 5.0*. Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95045 U.S.A., red release 5.0 edition.

[Sun-Labs, 2009b] Sun-Labs (2009b). *Sun Small Programmable Object Technology (Sun SPOT) Developer's Guide, Red Release 5.0*. Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95045 U.S.A., red release 5.0 edition.

[Sun-Labs, 2009c] Sun-Labs (2009c). *Sun SPOT Theory of Operation, Red Release 5.0*. Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95045 U.S.A., red release 5.0 edition.

[Sun-Labs, 2009d] Sun-Labs (2009d). *SunSPOT - Owner's Manual Red Release 5.0*. Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95045 U.S.A., red release 5.0 edition.

[Tanenbaum and van Steen, 2002] Tanenbaum, A. S. and van Steen, M. (2002). *Distributed Systems, Principles and Paradigms*. Number ISBN 0-13-121786-0. Prentice-Hall Inc.

[Texas-Instruments, 2007] Texas-Instruments (2007). *CC2420 a 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*, swrs041b edition.

[Want et al., 1992] Want, R., Hopper, A., Falcao, V., and Gibbons, J. (1992). The active badge location system.

Index

Typeset May 5, 2013