Datalogisk Institut
KØBENHAVNS UNIVERSITET

Evaluation of Accelerometers Mounted on Wireless Sensor Motes

Marcus Chang

(Joint technical report with Cambridge University)

Technical Report no. 06/02
ISSN: 0107-8283

Dept. of Computer Science
University of Copenhagen • Universitetsparken 1
DK-2100 Copenhagen • Denmark

Evaluation of Accelerometers Mounted on Wireless Sensor Motes

Marcus Chang
Department of Computer Science
University of Copenhagen
Denmark
marcus@diku.dk

Abstract

In this project we investigate the properties of accelerometers mounted on wireless sensor devices with the purpose of mounting wireless sensors on athletes to facilitate performance measurement. To this end a heterogenious data collection architecture with a system wide sampling rate of 200 Hz was developed to facilitate the evaluation. A variant of the Reference Broadcast Synchronization protocol was implemented and without MAC-layer timestamping we achieved an accuracy of 4 ms for a WSN with a single gateway and 27 ms for a WSN with two gateways connected to the same time source. The subsequent evaluation showed that, in order to use acceleration measurements to calculate the velocity and position, a gyroscope is crucial in order to compensate for rotation and a calibration within 0.1% accuracy is necessary as well. With a non-rotating accelerometer we were able to establish the position of an elevator with an accuracy within 10%.

keywords: wireless sensor networks, WSN, accelerometers, dead reckoning, Analog Devices ADXL202E, Freescale MMA1260D, Freescale MMA6261Q, TinyOS, time synchronization, Reference Broadcast Synchronization

Figure 1: Four Crossbow MicaZ motes (back) and two Freescale MC13192SARD (front).

Contents

Contents **4**

1 Introduction **5**
1.1 Motivation . . . 5
1.2 Purpose . . . 5

2 Freescale MC13192SARD **6**
2.1 TinyOS . . . 7
2.1.1 EVB13192 vs. MC13192SARD . . . 7
2.1.2 Loading Code . . . 8
2.1.3 Communicating over RS-232 . . . 8
2.1.4 Reading from Accelerometers . . . 9
2.1.5 Communicating with MicaZ . . . 9
2.2 Future Work . . . 9
2.3 Summary . . . 10

3 Acceleration **11**
3.1 Related Work . . . 11
3.2 Reference Frames . . . 11
3.3 Hardware . . . 12
3.4 Application . . . 12
3.4.1 Sampling rate . . . 12
3.4.2 Transmit . . . 13
3.4.3 Gateway . . . 13
3.4.4 Xlisten . . . 14
3.5 Evaluation . . . 14
3.5.1 Calibration . . . 14
3.5.2 Zero-motion Drift . . . 16
3.5.3 Power . . . 20
3.5.4 Walking . . . 22
3.5.5 Elevator . . . 26
3.6 Future Work . . . 29
3.7 Summary . . . 29

4 Time Synchronization **30**
4.1 Related Work . . . 30
4.2 Protocols . . . 30
4.2.1 Time Beacon . . . 31
4.2.2 Reference Broadcast . . . 31
4.3 Implementation Issues . . . 32
4.4 Evaluation . . . 33
4.4.1 Round-Trip-Time . . . 33
4.4.2 Clock Drift . . . 33
4.4.3 Agreement . . . 36
4.5 Future Work . . . 38
4.6 Summary . . . 40

5 Conclusion **41**

References **42**

A Application Source Code **44**

B Acceleration Data Files **44**

C Synchronization Data Files **44**

1 Introduction

This report is the result of a six week visiting student project carried out at the Computer Laboratory at the University of Cambridge, England. The project contributes to the Sentient Computing and Sports research at the Digital Technology Group and is part of the Embedded WiSeNts[1] project in which the Department of Computer Science at the University of Copenhagen, Denmark, is a participant.

I would like to thank the people at the University of Cambridge, especially Professor George Coulouris and Dr. Marcelo Pias for hosting and supervising the project, Dr. Alastair Beresford for organizing residence, and Dr. Robert Harle, Brian Jones and Alastair Tse for helping with the local computer system. I would also like to thank the people at the Univerity of Copenhagen, especially Associate Professor Philippe Bonnet for supervising, and Jan Flora and Esben Zeuthen for technical support regarding the Freescale devices.

The project was funded by: The EU Framework 6 Embedded WiSeNts Project, Cambridge University Computer Laboratory, and DIKU University of Copenhagen.

1.1 Motivation

In the field of sports technological advances have been used for a long time to improve the performance of athletes. Until recently this has been done either inside a lab with wired sensors, or outdoor with external measuring equipment (e.g. a stop watch). The goal of the Sentient Sports project is to combine these two approaches and enable athletes to wear wireless sensors, without restrictions to the environment, in order to help improve performances.

One of the physical properties we intend to measure is acceleration. With this information it is possible to calculate physical parameters such as velocity, position, step length, and frequency. Also, with multiple sensors, it will even be possible to measure these properties for the body as a whole, as well as for each individual body part.

1.2 Purpose

The key issue we want to understand is the precision of the physical properties obtained from acceleration measurements. This includes the drift of the sensors over time, and, of course, the growth of error over time. The main purpose of this project is thus to study and evaluate accelerometers mounted on a wireless sensor device. In order to perform this evaluation we had to develop a data collection infrastructure.

This report consists of three parts which can be read individually. Section 2 describes the process of making the Freescale MC13192SARD motes ready for TinyOS and how to use the accelerometers with TinyOS. Section 3 consist of the actual development of the data collection infrastructure and the subsequent evaluation of the accelerometers. Finally, Section 4 consist of the implementation of a time synchronization protocol which is essential when several motes are measuring the same events.

[1] Cooperating Embedded Systems for Exploration and Control using Wireless Sensor/Actuator Networks

2 Freescale MC13192SARD

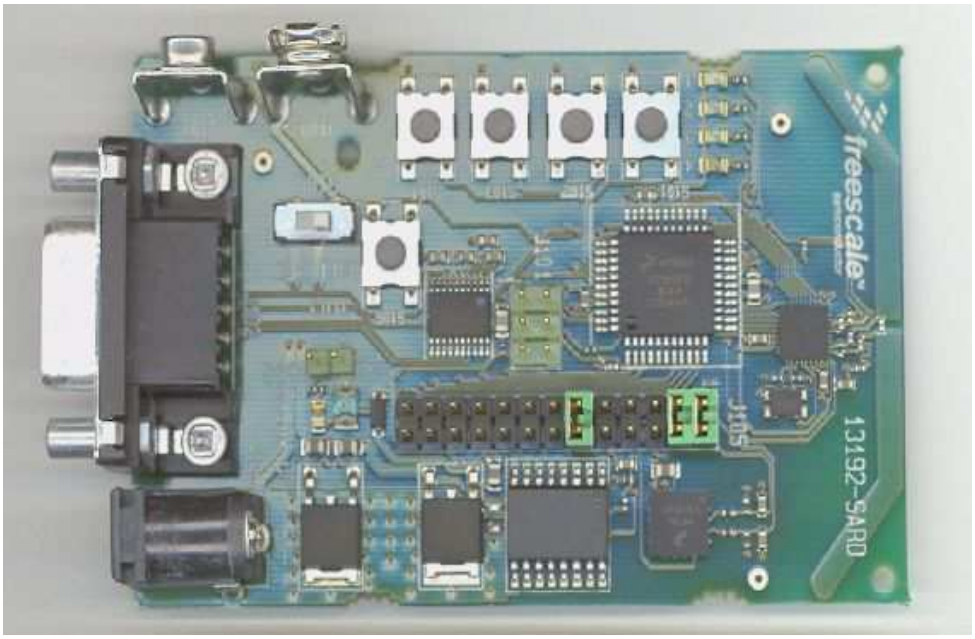

Figure 2: Freescale MC13192SARD wireless sensor mote [1].

One of the devices used in this project is the Freescale MC13192SARD[2] [1] mote (shown in Figure 2), which is based upon the Freescale EVB13192[3] [2]. The core of the device is the Freescale MC9S08GT60 [3] Micro Controller Unit which has the following key features:

- 0-40 MHz HCS08 8-bit CPU
- 4K RAM
- 60K FLASH
- 2 two-channel 16-bit timers
- 8-channel, 10-bit ADC
- 2 Serial Communications Interface modules (SCI)
- 1 Serial Peripheral Interface module (SPI)

Furthermore, the MCU has several power saving modes which make it possible to reduce power consumption by shutting down internal subsystems and changing operating frequency, e.g. power consumption, when running at 16 MHz, can vary from 6.5 mA to 25-675 nA. Also, the 8-channel ADC is in fact an 8-channel multiplexer equipped with cross-talk preventing circuitry[4] with a maximal sampling rate of 14.0 μsec.

Connected to the SPI module is the Freescale MC13192 transceiver [4], which is a low power, short range 2.4 GHz ISM band radio with the following key features:

- IEEE 802.15.4 compliant (ZigBee)
- 16 channels (5 MHz intervals)

[2]Sensor Application Reference Design
[3]Evaluation Board
[4]Dedicated circuitry to avoid signals from one channel to interfere with another.

- 250 kbps transfer rate

Power consumptions range between 34-37 mA when transmitting/receiving and 500 μA in idle mode. Two on-board antennas are used for sending and receiving respectively.

Connected to three of the ADC ports are the Freescale MMA6261Q dual-axis accelerometer [5] and the Freescale MMA1260D single-axis accelerometer [6]. Both accelerometers have a $\pm$1.5 g range of operation, but the MMA6261Q has a noise[5] and sensitivity rating of 3.5 mV and 800 mV/g respectively, where the MMA1260D is rated as 5.0 mV and 1200 mV/g respectively. This means that the single-axis accelerometer has a higher sensitivity, but with a lower precision.

Besides these three key parts, the MC13192SARD also has 4 LEDs, 4 push buttons, and 1 reset button. The device can be powered either by a 9 V battery or a 3-9 V DC power supply. Programming can be done either via the BDM interface (Background Debug Module), or via the 9 pin RS-232 port which is connected to the SCI module `SCI1` on the MCU. This port can also be used as a regular communications port.

2.1 TinyOS

Since the key constraints, when developing a wireless sensor networks (WSN) application, are the limited resources on each device such as processing power, storage capability, and electrical power, it is important that the operating system in use honours these constraints. TinyOS is an open-source operating system built specifically for WSN [8]. It uses an event driven execution model and utilises a component-based architecture, which makes it easy to move the boundary between dedicated hardware and software emulation and save power by shutting down unused components. Also, this component-based architecture minimizes the overall program size since only essential components are included. The TinyOS core itself only requires 400 bytes of code and data memory combined. All these features make TinyOS a much preferred operating system for wireless embedded sensor networks and are therefore available on a wide range of platforms, in particular the MicaZ motes from Crossbow [7].

However, the support for TinyOS on the Freescale EVB13192 is a work in progress, currently being carried out at the Department of Computer Science at the University of Copenhagen (DIKU). The current build is available from CVS at SourceForge[6] under `cvs:tinyos/tinyos-1.x/contrib/diku/`. Since the MC13192SARD is a platform in itself, it is only appropriate to have the same support in TinyOS as for the EVB13192. Therefore, the work in the following sections has been integrated with the EVB13192 code tree at SourceForge as of October 12th 2005, so the MC13192SARD will benefit from any improvements made to the EVB13192.

To compile to the MC13192SARD platform the compile command

```
> make dig536
```

can be used once inside the `contrib/diku/evb13192/apps/<application>` directory. The current build requires direct access to the internet and `curl`[7] must be installed, since the compiling and linking are done remotely on a server located at DIKU. This is because currently no open-source compiler compatible with the HCS08 instruction set exists. The TinyOS code is thus precompiled to C-code locally and the result is transmitted to the compile host at DIKU. This host has a propriertary compiler installed which finally compiles the executable. Work in progress though on porting the current propriertary compiler to an open-source compiler. This would make it possible to perform the entire build process locally without the need for internet access.

2.1.1 EVB13192 vs. MC13192SARD

Because of several differences between the EVB13192 and the MC13192, several software modifications and additions had to be made in order to run TinyOS on the MC13192SARD.

[5] Where noise is measured as the RMS error when operating at 0.1 Hz - 1.0 kHz. $RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$

[6] http://www.sourceforge.net

[7] cURL, http://curl.haxx.se/

The two most obvious differences between the EVB13192 and the MC13192SARD are the extra serial-over-USB port (connected to the second SCI module, `SCI2`) on the EVB13192 and the two accelerometers on the MC13192SARD. Not so obvious is the bootloader program pre-installed on the MC13192SARD which is not compatible with code generated through DIKU. In order to develop TinyOS applications to the MC13192SARD these three problems must be solved.

Additionally, as mentioned above, the MC13192SARD is equipped with two on-board antennas, each dedicated to transmit and receive respectively. Since the EVB13192 only has one antenna, it uses a special pin to set if either the input or output port of the transceiver is connected to the antenna at a given time. Because this pin is left unused on the MC13192SARD, programs written for the EVB13192 are thus also compatible with the MC13192SARD, eventhough the radio is connected differently. Had the process been reversed and had the original development first been performed on the MC13192SARD, instead of the EVB13192, this would not had been the case.

2.1.2 Loading Code

The pre-shipped bootloader was replaced by a new one written by Jan Flora and Esben Zeuthen at DIKU (which are the people responsible for a major part of the DIKU contribution to the TinyOS repository). The new bootloader is a low-speed version of the one used at DIKU and uses the RS-232 port instead of the USB port. With a Background Debug Module this new bootloader was installed on all of the MC13192SARD devices.

Programming through the RS-232 port is done with the `Developer's Serial Bootloader` program [10]. Since the EVB13192 has not been fully integrated with TinyOS yet, the usual

```
> make <platform> install,<id> <programmer>,<location>
```

command is not supported. In order to change the value of the system variable `TOS_LOCAL_ADDRESS` after compilation, the script

```
> set-mote-id <id>
```

located at `contrib/diku/evb13192/tools/scripts` must be used.[8] The new executable is called `app.s19.<id>` and resides in the `<application>/build/evb13192` directory. The executable file can then be uploaded with the command:[9]

```
> hc08s.exe <com-port-number> <com-speed> <file>
```

2.1.3 Communicating over RS-232

As mentioned above the EVB13192 is equipped with both a RS-232 port and a USB port, but because the USB connection can serve as a power supply as well, all work at DIKU has been done under the assumption that this port would be the preferred communications port.

The former `HPLUART0` module, located in the `contrib/diku/evb13192/tos/platform/hcs08` directory, was the primary UART communication module and resided at the lowest level, such that all other applications, in need of UART communication, were build on top of this. This module was connected directly to the USB port via the `SCI2` module. Since the RS-232 port is connected via the `SCI1` module, the most consistent way to implement RS-232 connectivity is to create two new modules, `HPLUART1` and `HPLUART2`, where the `HPLUART2` module is simply the old `HPLUART0` module renamed and the `HPLUART1` module has had all communication diverted to the `SCI1` module instead of the `SCI2` module.

Thus it becomes possible to communicate with the RS-232 port on the MC13192SARD by using the `HPLUART1` module and to communicate with both the USB and RS-232 port simultaneously on the EVB13192.

[8] The default value for `TOS_LOCAL_ADDRESS` is 1.

[9] The execution of this command must be combined with a reset of the MC13192SARD. The sequential order of this process can vary from time to time, and usual several retries are necessary.

2.1.4 Reading from Accelerometers

According to the MC13192SARD schematics [1] the dual-axis accelerometer's x and y-axis are connected to the two ADC ports `AD0` and `AD1` respectively, and the single-axis accelerometer (which becomes the z-axis) is connected to port `AD7` on the ADC. Since all values are with respect to the common ground, all measurements become absolute values and sampling from the two accelerometers hence can be done simply by reading the values of the three ADC ports. The three modules XaxisM, YaxisM and ZaxisM do exactly this.

The readings are, however, neither in volts nor in m/s^2 but some arbitary 10-bit value taken from a linear voltage-to-value conversion table inside the ADC. These values must subsequently be transformed into measuring units in order to get an absolute acceleration. Only the raw ADC values are supplied by the modules, however, and the user is required to perform a calibration of these readings before an actual acceleration can be obtained.[10] To use the accelerometers one only needs to insert the line `SENSORBOARD=accelerometer` in the `makefile` for the application. Each of the above modules provides the interface `ADC` and can be included as any regular TinyOS component in any combination. Sampling is done asynchronously with a call to the command

```
call ADC.getData();
```

and the measurement is returned by the event

```
async event ADC.dataReady(uint16_t data)
```

Notice that even though the ADC only has 10-bits of accuracy, the measurement is stored as a 16-bit value. If performance is an issue, data transmission can be reduced by only regarding the lower 10-bits of each measurement as actual data.

2.1.5 Communicating with MicaZ

Since a `TOSMsg` compatible radiostack for the EVB13192 is still in beta test, the current stable radio stack is the `SimpleMacM` module. This component is capable of receiving messages that have been transmitted with the Crossbow MicaZ radio [7], but not the other way around. Since the messages used by the `SimpleMacM` module are not following the IEEE 802.15.4 standard and only contain pure data, i.e. no headers of any kind, these messages are automatically discarded by the MicaZ radio. This also means that all transmitted messages have no destination address and all received messages must be checked since the module in effect works in promiscuous mode.

Whether or not it is possible for the `SimpleMacM` module to achieve full compatibility with the MicaZ mote, by inserting the appropriate TinyOS headers, remains to be seen. Since a native TinyOS radiostack is well under way we did not pursue this issue any further.

On the flip side it is possible to set the radio's communication channel with the

```
call SimpleMacM.setChannel(<channel>);
```

command (where the `<channel>` ranges from 0-15 and corresponds to the ZigBee channels 11-26 respectively) which is not as easily done on the MicaZ mote since this has to be done at compile time.

2.2 Future Work

In order to perform time synchronization in a wireless sensor network it is essential to have two-way communication, since sensor motes transmitting data must be able to receive synchronization messages as well. Since the `SimpleMacM` module is capable of receiving messages from the MicaZ motes, it is possible to develop a heterogenous data collection architecture, by using a SARD mote as gateway. However, it will not be possible to time synchronize the MicaZ motes with the SARD gateway. It is thus important to establish full duplex communication between the SARD motes and other ZigBee complaint motes such as the MicaZ. Either by making the appropriate modifications to the `SimpleMacM` module or by completing the new native TinyOS radiostack.

[10] Since all analog accelerometers have a characteristic signal response, a universal calibration would be useless. And, as later experiments will show, the signal response is highly dependent of the supplied voltage.

2.3 Summary

With the TinyOS contribution library from DIKU and the new bootloader it is possible to compile and upload TinyOS programs to the MC13192SARD. With the alterations made to the UART component, communication over the RS-232 port is possible and all the test programs that uses the UART for communication now works on the MC13192SARD. With the new accelerometer modules it is possible to sample meassurements from the accelerometers and use the measurements directly in TinyOS. In effect this makes the MC13192SARD device as TinyOS compatible as the EVB13192.

3 Acceleration

The ability to use simple observations of velocity to deduce a position is known as "dead reckoning" and has been used by sailors since the early Middle Age. By throwing wooden objects overboard and counting how many seconds it took them to fall behind, navigators had a simple but efficient tool to estimate the traveled distance. The connection between acceleration and velocity and position was however first quantified with the publication of "Philosophiae Naturalis Principia Mathematica" by Sir Isaac Newton in 1687, and Newton's equations of motions have since then become the basic foundation for all common[11] motion calculations ranging from designing marry-go-arounds to launching rockets to Mars.

Using Newton's equations of motion on acceleration data gathered from people in motion, with the purpose of obtaining a position, however, has not been possible until recently when such measurement devices became small enough for a person to carry. With the advent of Micro-Electro-Mechanical Systems (MEMS) accelerometers, personal navigation systems based on "dead reckoning" made it possible to measure the position of the body as a whole. The next step would then be to measure the position of individual body parts.

The knowledge of the positions of individual body parts during a course of actions would be useful for athletes in order to improve their performances. A runner would be able to know the exact length and duration of each step instead of just an averaged value. Gymnasts would have direct access to the three-dimensional movement of their body without the need to resort to guessing from watching two-dimensional video recordings.

In order to put accelerometers on athletes it is necessary to understand the behavior of these devices and especially their drift. The purpose of this part of the project is to evaluate the accelerometers on the Freescale MC13192SARD and the Crossbow MicaZ, specifically with the goal of obtaining a position from acceleration measurements.

Section 3.1 describes work related to accelerometers and Personal Navigation Systems and Section 3.2 discusses the transformations between different reference frames. Section 3.3 presents the accelerometers used in this experiment and Section 3.4 gives an overview of the TinyOS applications developed to evaluate the two different platforms. Last Section 3.5 contains an analysis of the experimental results obtained and Section 3.7 summarizes this part of the project.

3.1 Related Work

The use of "dead reckoning" for Personal Navigation Systems has been done by [11] and [12]. In [11] the use of a sophisticated measurement device including GPS, gyroscopes and compass were used in conjunction with map-matching in order to provide an indoor navigational system. By matching the measured position with known physical viable positions the accuracy was improved. The need for a 3D-map model makes the usability limited, however. [12] uses an estimate of the step length together with directional measurements (compass and gyroscopes) to deduce a position with an accuracy of 2-20 m. Their method is prone to errors on the step length estimation which is deduced by the magnitude of the vertical acceleration. Since the tilt of the measuring equipment was not taken into account the vertical acceleration was ill-defined.

The measurement of accelerometers' drift and noise has been done extensively for stationary accelerometers by [13]. The error of the position achieved by integrating the acceleration twice was calculated and measured for several factors such as noise, sampling rate and time and it was found to grow as the square of time. This work did not encompass moving accelerometers, however.

3.2 Reference Frames

In order to understand the samples measured by the accelerometers it is important to understand in respect to what this acceleration is measured. When one performs a measurement, e.g. measuring the speed of a car driving by with a laser beam, this speed is measured relative to the measuring device. Since such devices usually are stationary this reference frame is called the laboratory reference frame (or sometimes the geocentric reference frame). If, however, the measurement device is not stationary all measurements are said to be taken relative to the local reference frame. In order to convert these measurements to the laboratory reference frame it is necessary to perform a reference frame transformation.

[11] Where common refers to the non-relativistic and non-quantum mechanical regime.

According to the *equivalence principle of mechanics* [14], measuring an acceleration of a reference frame is equivalent to measuring an opposite gravitational acceleration inside the said frame. Since the acceleration measurements in these experiments are taken with respect to an accelerated reference frame, the accelerometers are actually measuring the gravitational acceleration mentioned above. Because this corresponds to an opposite acceleration of the reference frame, the actual acceleration as seen from the laboratory reference frame can be deduced by simply changing the sign of the acceleration. This is however only true for accelerated reference frames without rotation.

If the accelerated reference frame is rotating as well, the transformation is not quite as simple. One must then perform a rotational transformation of the acceleration as well, in order to measure the acceleration relative to the laboratory reference frame. This, however, requires that the rotation vector of the accelerating reference frame is known. Since this information was not available to us due to lack of gyroscopes, all measurements hence forward are all with respect to the local reference frame.

One transformation can, however, be done without the use of gyroscopes, although it will be a purely cosmetic one. By using the Earth's gravitational acceleration, which can be measured when the accelerometers are stationary, it is possible to use this vector to rotate the local reference frame so the z-axes from both reference frames are aligned.

3.3 Hardware

The three accelerometers used were the Analog Devices ADXL202E, Freescale MMA1260D and Freescale MMA6261Q with the ADXL202E being mounted on the Crossbow MTS310CA sensor board.

Analog Devices ADXL202E

This dual-axis accelerometer has a range of ± 2 g on both axes. It has a noise density and sensitivity of 200 $\mu g\sqrt{Hz}$ and 167 mV/g respectively. Also, the temperature drift is listed to be $\pm 0.5\%$ [15].

Freescale MMA1260D

This is a single-axis accelerometer with a range of ± 1.5 g. The noise density and sensitivity are listed as 500 $\mu g\sqrt{Hz}$ and 1200 mV/g respectively, which makes it a more sensitive accelerometer than the previous one but with lesser accuracy. This accelerometer has, however, built-in circuitry to compensate for temperature drift [6].

Freescale MMA6261Q

This is a dual-axis accelerometer with a range of ± 1.5 g for each axis. The noise density and sensitivity are listed as 300 $\mu g\sqrt{Hz}$ and 800 mV/g respectively, which puts it between the two previous accelerometers with regard to noise and sensitivity. This accelerometer also has built-in circuitry to compensate for temperature drift [5].

3.4 Application

In order to sample measurements from the accelerometers and have them stored on a PC, three applications were used: `Transmit`, `Gateway` and `Xlisten`. `Transmit` and `Gateway` were developed in TinyOS while `Xlisten` is a modified version of Crossbows logger program. A reference to the source code for all three programs can be found in Appendix A.

3.4.1 Sampling rate

Since this particular type of WSN contains several layers of hardware abstractions, e.g. sensor motes, gateway, and PC, it is important to look at the limiting factors in each layer in order to achieve optimal performance. This will, in effect, determine a maximum sampling rate for each layer.

Designating the sampling rate for the actual measurements to be at the sensor level, the measurement transmission rate over the radio to be at the communications level and the measurement transmission rate between the gateway and the PC to be at the network level, an estimate of the overall limiting factor would be the network level. Because the transmission speed of the UART[12] is 38.4 kbps and the radio is 250 kbps, a standard TinyOS message[13] can

[12] In its current configuration the SARD UART has a maximum transmission speed of 38.4 kbps although it might be possible to increase this to 115.2 kbps.

[13] The standard size of a TinyOS message is 29 bytes.

be transmitted in 6 ms and 0.9 ms respectively (disregarding overhead). In comparison the ADC can be sampled every 14 μs.

Depending on the header size and the number of axes measured, a typical TinyOS packet payload can contain 3 and 4 samples for the SARD and MicaZ respectively. Since the transmission time for a packet payload is at least 6 ms at the network level, the maximum theoretical number of samples transmitted within a second would be 498 and 664 respectively. Because the transmission speed at the communication level is higher than at the network level it is reasonable to assume that the communication level can sustain a similar sampling rate.

At the sensor level the maximum sampling rate would depend on the number of sensor motes present and in contest of the communication level. In a multi-mote environment the overall sampling rate should be divided among the motes, since they all share the same communication level.

Since overhead and busy channels have not been included in the above discussion it is reasonable to believe that the 498 Hz and 664 Hz sampling rate should be regarded as a theoretical upper bound and will not be obtainable in reality.

3.4.2 Transmit

This program resides on the motes carrying accelerometers and samples each measurement when a timer is fired. The timer is controlled by a predefined sampling rate and each axis is sampled one at a time. The result is stored in a buffer together with a timestamp, which is the lowest 16-bit of the internal counter. On the Freescale motes this corresponds roughly to one count each millisecond. When the buffer is full the data is copied to the radio packet which also contains the address of the mote and a timestamp of when the packet was sent. This timestamp is the full 32-bit counter though, and if the sampling rate is high enough to avoid overrun of the measurement timestamp the upper 16-bit can be used to restore a full 32-bit timestamp for each measurement. The sampling rate is set by the compiler define

```
#define SAMPLING_RATE <number of milliseconds between each sampling>
```

and is the actual time between each sampling on each mote. With a single SARD mote this can be as low as 5 ms which is quite far from the theoretical 2 ms.

The packet structures for the above mentioned program can be seen in Table 1 and Table 2 for the SARD and MicaZ mote respectively, and both have been designed specifically for this application.

Byte offset	Data
0	mote number
1-4	32-bit packet timestamp
5-6	1st x-axis measurement
7-8	1st y-axis measurement
9-10	1st z-axis measurement
11-12	1st 16-bit measurement timestamp
⋮	
21-22	3rd x-axis measurement
23-24	3rd y-axis measurement
25-26	3rd z-axis measurement
27-28	3rd 16-bit measurement timestamp

Table 1: SARD payload structure

3.4.3 Gateway

This program resides on the mote connected to the PC, and basically dumps all received radio packets to the UART, similar to the standard TinyOS gateway program, `TOSBase`. Because of some communications incompatibilities

Byte offset	Data
0	mote number
1-4	32-bit packet timestamp
5-6	1st x-axis measurement
7-8	1st y-axis measurement
9-10	1st 16-bit measurement timestamp
⋮	
23-24	4th x-axis measurement
25-26	4th y-axis measurement
27-28	4th 16-bit measurement timestamp

Table 2: MicaZ payload structure

between the MicaZ and SARD and since it is desirable to have one gateway capable of receiving packets from both, the `Gateway` program was written almost from scratch and only runs on the Freescale platform. Since `Gateway` eventually needs to do time synchronization, a high level of control is desired which makes `TOSBase` unsuitable.

The program consists of one write buffer and one queue buffer. When a packet is received and the UART is not engaged in communication the received packet buffer becomes the new write buffer and vice versa by swapping pointers, and the program begins to transmit the write buffer over the UART connection. If however the UART is in use, the packet buffer is swapped with the queue buffer and once the write buffer has been completely transmitted, the write buffer is swapped with the queue buffer and subsequently transmitted over the UART connection. If both the write buffer and the queue buffer are in use the packet is dropped. One could add more queue buffers, but for this project a high sampling rate is desired which means that packets will be arriving at a constant rate. Since the UART connection is the bottleneck of the system, once the sampling rate exceeds the UART transmission rate no amount of queues will prevent packet drops. In a burst environment, however, a higher number of queue buffers would be advantageous.

3.4.4 Xlisten

The `Xlisten` program is one of the standard TinyOS programs to log mote data sent over the UART or ethernet. The program was modified to communicate with the `Gateway` program and print the accelerometer data packets in the correct format.

The program can be started with the following command:

```
> ./xlisten.exe -j -s=com<number>
```

3.5 Evaluation

The evaluation of the accelerometers has been divided into five subsections. Subsection 3.5.1 contains the calibration data for all the accelerometers. Subsection 3.5.2 studies the drift measured by a stationary accelerometer. Subsection 3.5.3 examines the impact of the power supply's voltage on the acceleration measurements. In subsection 3.5.4 the acceleration from a walking person is measured and finally in subsection 3.5.5 the acceleration from a purely translatorical movement is measured.

3.5.1 Calibration

The first and foremost thing one must do in order to use the accelerometers to measure an absolute[14] acceleration, is to convert and scale the readings from the ADC into proper measuring units. Since the voltage response from the accelerometers is linear[15] with respect to the measured acceleration, a straight line can be fitted to known reference points and the absolute acceleration can be deduced by the linear relation:

[14] Since we want to calculate a physical length a *relative* acceleration is unsuitable.
[15] According to the specifications this holds for most of the devices measurement range.

$$(ADC\ value) = a \times (acceleration\ measured\ in\ < m/s^2 >) + b$$

As mentioned in the Crossbow documentation [9], the simplest way to determine a and b is by using the gravitational acceleration as a point of reference. This is accomplished by first measuring the two extreme values each axis senses when it is pointed either vertical up or down. Since the accelerometers have a linear response, these two extremes would be equal to g and -g respectively and the spanned interval would be of the magnitude $2g$. Second, by measuring the acceleration the axis senses when it is oriented horizontal, one also learns the value corresponding to $0g$.

The calibrations made were done with $g = (9.81258 \pm 0.00005)\ m/s^2$ which were obtained from the National Physics Laboratory.[16] Hence all values forward will be in SI units.[17]

The raw accelerometer measurements were obtained by attaching each mote (one at a time) to a spirit level which was then used to align each axis parallel with the gravitational acceleration. Each extreme was measured between one and two thousand times and the median were used for the calibration since the median is more robust to sudden spikes and variations than a simple mean. Table 3 and Table 4 shows the calibration data and the RMS error for the Freescale motes and Crossbow sensor boards respectively. Figure 3 show an example of an actual calibration curve for Freescale mote 6. A reference to the raw data files can be found in Appendix B.

mote 5			mote 6			mote 7			mote 8		
	median	RMS		median	RMS		median	RMS		median	RMS
x up	255	1.4	x up	239	1.9	x up	220	1.4	x up	233	1.7
x hor.	501	3.2	x hor.	492	2.7	x hor.	460	7.1	x hor.	490	4.6
x down	747	1.3	x down	746	1.7	x down	713	1.4	x down	745	1.4
y up	248	1.9	y up	271	2.1	y up	255	2.0	y up	263	1.7
y hor.	494	2.9	y hor.	513	2.5	y hor.	499	2.9	y hor.	499	4.1
y down	741	1.8	y down	755	2.1	y down	747	2.0	y down	739	1.7
z up	239	1.0	z up	226	1.2	z up	240	1.2	z up	241	1.2
z hor.	483	4.7	z hor.	474	1.5	z hor.	487	5.7	Z hor.	485	3.0
z down	728	1.1	z down	723	1.1	z down	732	1.2	z down	732	1.2

Table 3: Calibration data for Freescale motes

board 1			board 2			board 3			board 4		
	median	RMS		median	RMS		median	RMS		median	RMS
x up	413	0.4	x up	405	0.4	x up	464	0.5	x up	447	0.7
x hor.	467	0.5	x hor.	455	1.0	x hor.	513	4.1	x hor.	502	1.4
x down	524	0.1	x down	502	0.5	x down	564	0.5	x down	558	0.6
y up	436	0.6	y up	481	0.6	y up	466	0.9	y up	476	0.7
y hor.	492	1.4	y hor.	532	1.5	y hor.	519	1.7	y hor.	532	0.7
y down	549	0.5	y down	583	0.5	y down	570	0.5	y down	588	0.3

Table 4: Calibration data for Crossbow sensor boards

It can be seen from Table 3 and Table 4 that the measuring range of $[-g;g]$ is of the order of 500 and 100 ADC units respectively for the Freescale and Crossbow motes. This results in a resolution of 0.0393 $\frac{m}{s^2 <ADC\ units>}$ and 0.196 $\frac{m}{s^2 <ADC\ units>}$ respectively.

There are however several problems with this calibration method. First of all, it is based on the assumption that all three accelerometer axes have been perfectly aligned with the gravitational acceleration. If this is not the case the measured value will be smaller than the actual $|g|$ since only a projection of the gravitational acceleration is measured, and, subsequently, all absolute accelerations will thus be larger than their real values. Second, the calibration is only based on three reference points spanning an interval of $[-g;g]$, hence one should not expect the calibration to be valid far from this interval.

[16] By using the formula at http://www.npl.co.uk/mass/faqs/gravity.html with the latitude 52.211048° and height 24 m above sea level. These values were taken for the William Gates building with the help of Google Earth, http://earth.google.com.

[17] Système International d'Unités (International System of Units), http://www.bipm.fr/en/si/

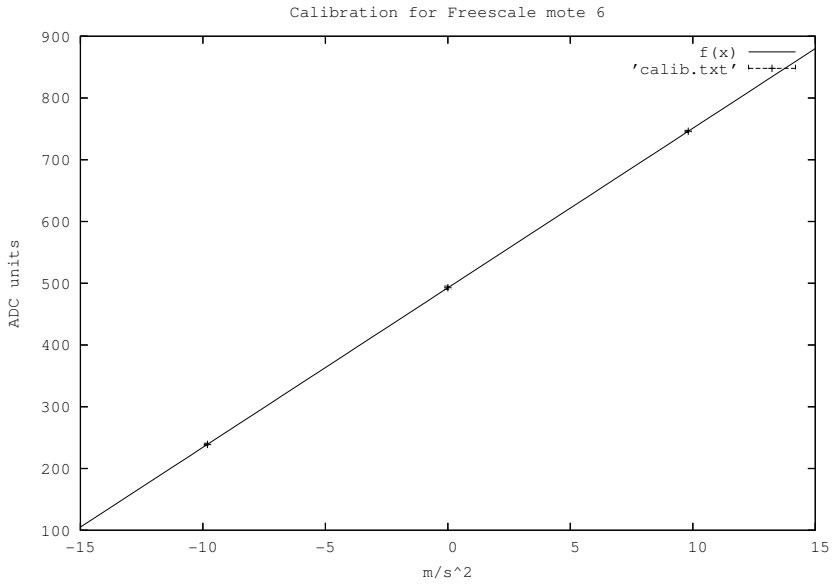

Figure 3: Calibration curve

A more precise calibration could be achieved by inducing a constant acceleration to the sensors of different magnitude so more reference points can be collected. This could be done by placing the sensors on the rim of a rotating disk, and by varying the rotational speed different magnitudes of acceleration can be measured.

3.5.2 Zero-motion Drift

The zero-motion drift is the change of acceleration measured from a stationary mote. This kind of drift usually manifests itself over a long period of time and can primarily be attributed to temperature changes. Since the accelerometer's temperature rises after prolonged use it is expected that the measured acceleration would change for a stationary mote, and if one knew the exact temperature, it would be possible to compensate for this drift.[18]

A measurement of this zero-motion drift was made for three of the Freescale motes and one of the Crossbow sensor boards. First, one Freescale mote was set to run for five hours uninterrupted. For each axis the median of the first 1000 measurements were used as reference value in calculating the ratio for all the measurements.

Ratio: $\frac{a_{measured}}{a_{reference}}$

The CDF plot of the ratio for each axis can be seen in Figure 4. The experiment was then repeated with three Freescale motes (this time only for 100 minutes though). The plots can be seen in Figure 5, Figure 6, and Figure 7.

The figures show that over 90% of all the measurements lie between $\pm 1\%$ of the reference value. For the five hour long measurement data, a straight line was fitted to each axis and the RMS error was calculated to be 1.29, 1.89 and 1.11 for the x,y and z-axis respectively (all in ADC units).

As it can be seen in Figure 8 the range of acceleration values spanned by the three lines are less than that of the RMS error for each axis respectively. Since the magnitude of this apparent drift is smaller than the precision of the measurements it is not possible to measure any drift in the Freescale accelerometers as expected, since the Freescale accelerometers have temperature correction circuitry.

Similar experiments were conducted with one Crossbow sensor boards and the corresponding ratio CDF plots for the five hour long experiment can be seen in Figure 9. As seen with the Freescale accelerometers, over 90% of the measurements are within $\pm 1\%$ of the reference value. In fact, the noise is so low on the MicaZ accelerometer that one can say over 90% of the measurements are within $\pm 0.4\%$ of the reference value.

[18] This only applies for the accelerometers on the MicaZ motes however, since the accelerometers on the SARD motes have built-in circuitry to compensate for temperature drift.

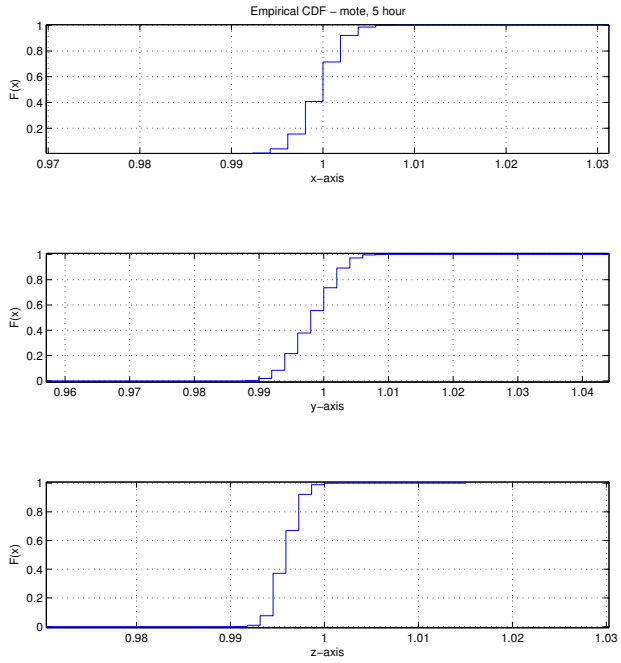

Figure 4: Ratio CDF plot of 5 hours measurements. Freescale accelerometer.

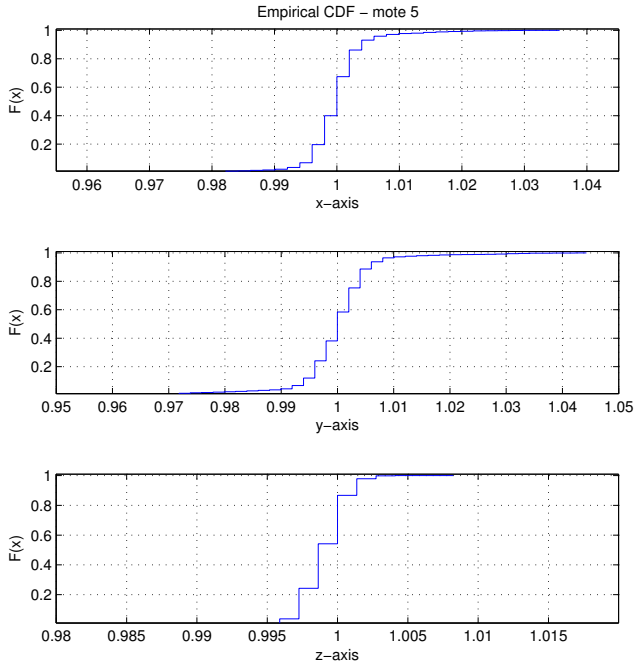

Figure 5: Ratio CDF plot of mote 5, 100 minutes. Freescale accelerometer.

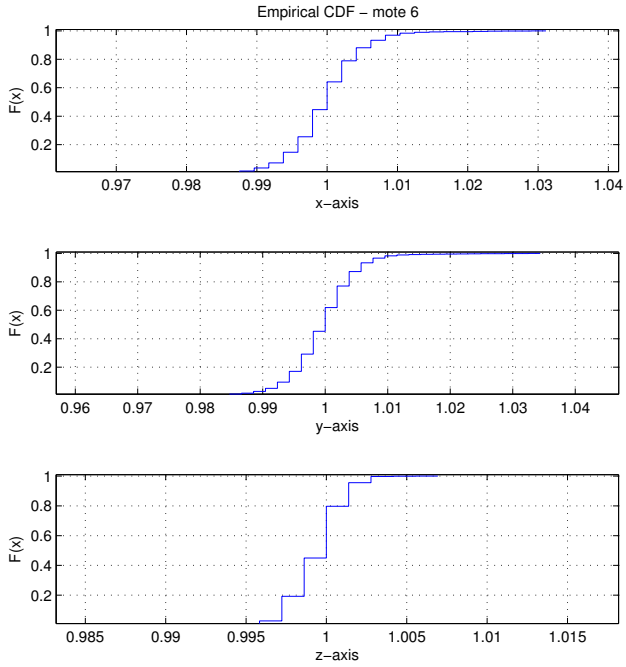

Figure 6: Ratio CDF plot of mote 6, 100 minutes. Freescale accelerometer.

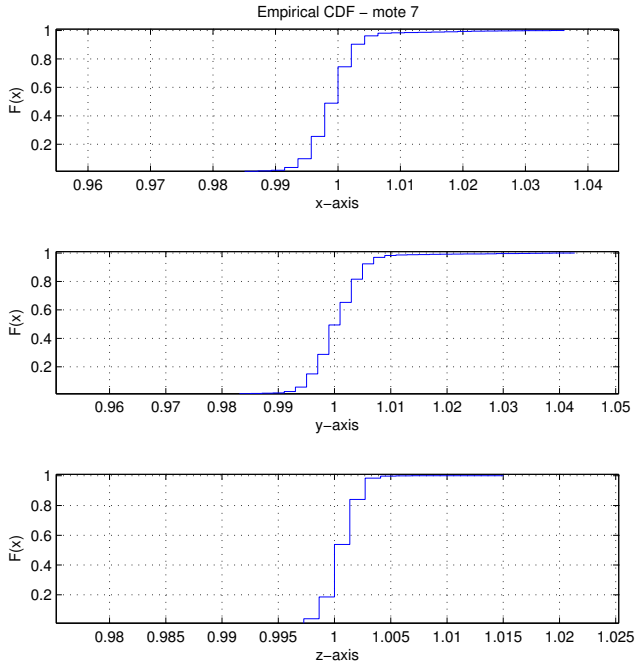

Figure 7: Ratio CDF plot of mote 7, 100 minutes. Freescale accelerometer.

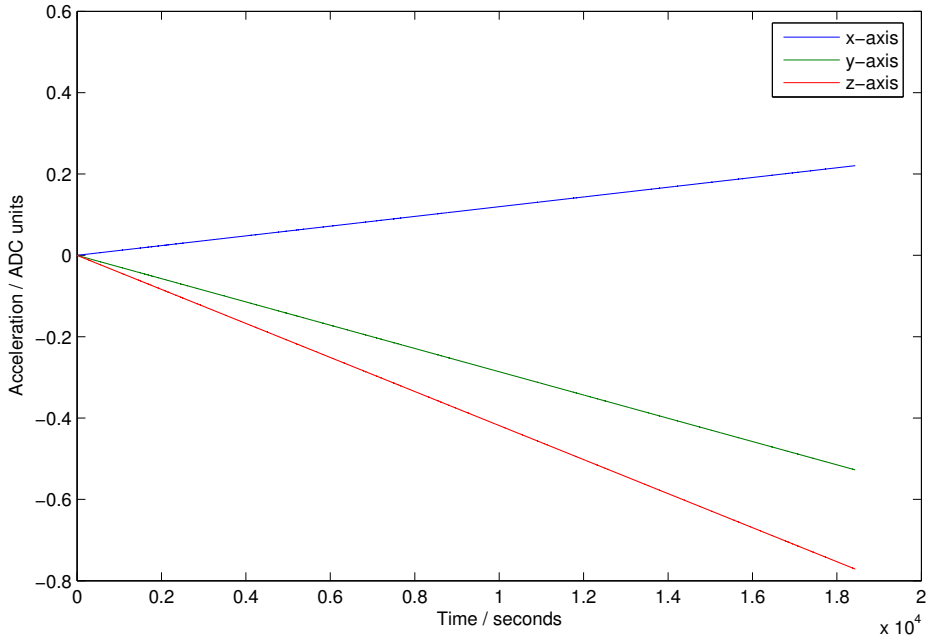

Figure 8: Three straight lines fitted to the 5 hour experiment data. All lines have been moved to the common origin to emphasize the drift. The RMS errors are 1.29, 1.89 and 1.11 for the x,y and z-axis respectively.

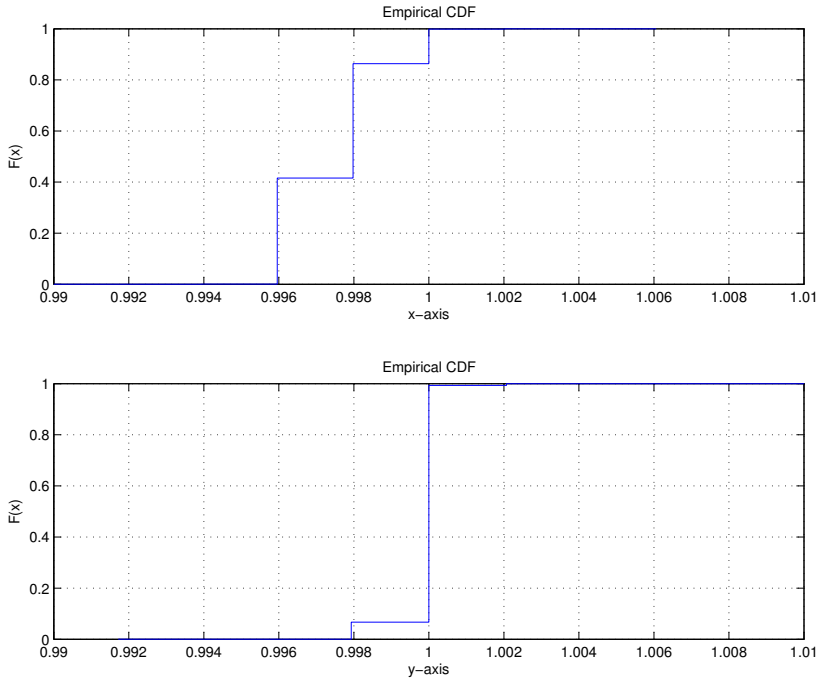

Figure 9: Ratio CDF plot of 5 hours measuremements. MicaZ sensorboard.

However, if a line is fitted to the measured data, as seen in Figure 10, the drift is higher than that observed with the Freescale accelerometer. Calculating the RMS error for the two fits one get 0.4 and 0.3 for the x and y-axis respectively. Since the error of the fit for both axes is lower than that of the observed drift (measured acceleration interval) the lines do indeed represent measured drift from the accelerometers. This is consistent with the expected drift from temperature and in the absence of correction circuitry this must be taken into account when actual measurements are performed.

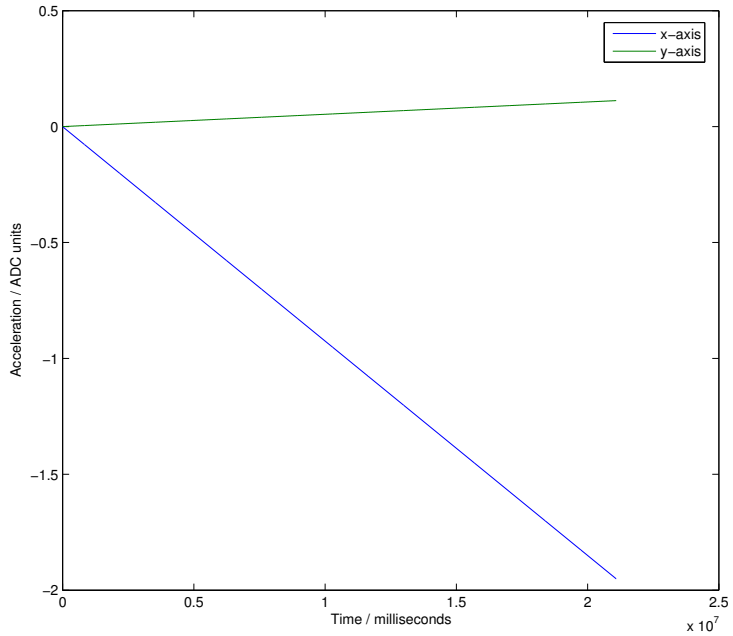

Figure 10: Two straight lines fitted to the 5 hour experiment data. All lines have been moved to the common origin to emphasize the drift. The RMS errors are 0.40 and 0.26 for the x and y-axis respectively. MicaZ sensor board.

A reference to the raw data can be found in Appendix B.

3.5.3 Power

Another interesting source of error was found in the power supply. Since the accelerometers on the Freescale motes require a larger amount of voltage than the rest of the mote in order to function properly, it is possible to continue sampling even though the output from the accelerometers cannot be trusted. This can be seen in Figure 11 where the left part of the plot is taken from a mote with a battery close to depletion, and the right part is taken from a mote with a new battery. When calculating the RMS deviation from the mean value one gets 12.6 m/s^2 and 2.0 m/s^2 for the left and right part respectively. Although it appears that only the noise increases and the mean remains constant when the battery is nearing depletion, Figure 12 shows this not to be the case and that it is vital to use batteries with sufficient voltage. It should be noted that neither motes where moving during the collection of data. A reference to the data files can be found in Appendix B.

According to the Freescale accelerometer documentation [5] and [6] the devices stop functioning properly when the supplied voltage drops below 2.7V and 4.75V for the dual-axis and single-axis accelerometer respectively. Hence if one knew the power supply voltage (and thus the voltage supplied to the accelerometers) it would be possible to determine when the measurements were invalid. However, it is undetermined if the noise observed in Figure 11 increases as the voltage approaches the cut-off voltage or if it only occurs after this point.

It remains to be seen if the accelerometers on the MicaZ motes have the same kind of behavior.

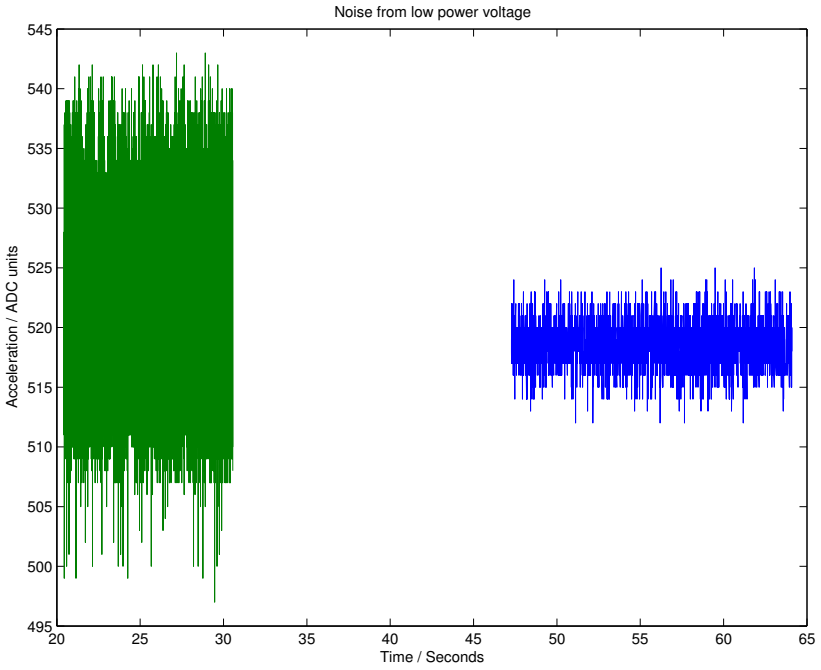

Figure 11: Comparison between a Freescale mote with low power (left) and full power (right).

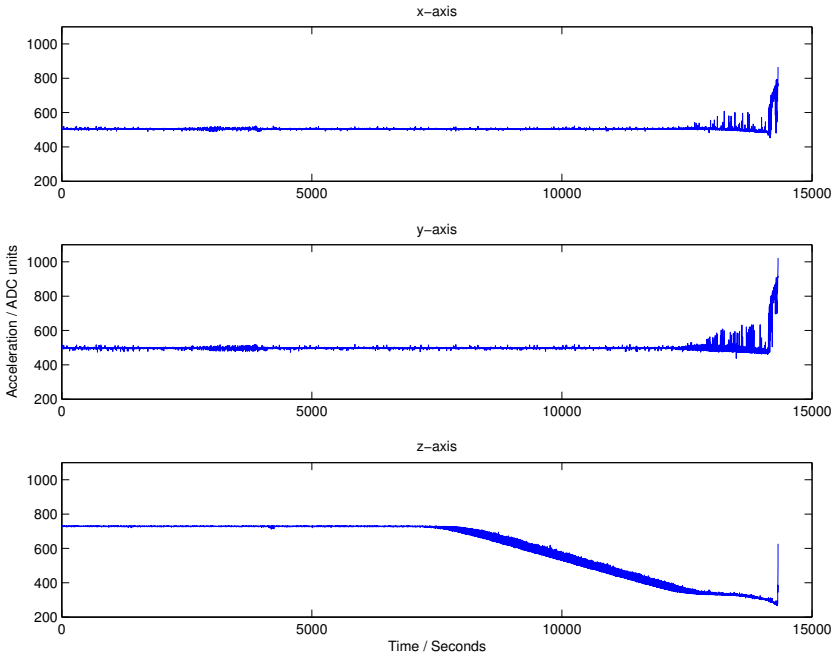

Figure 12: Freescale mote running out of power.

3.5.4 Walking

After these preliminary experiments we measure the actual acceleration of a person walking. A Freescale mote was mounted inside a plastic case and the case was attached to the chest of the test subject. The most simple experiment would be to walk a straight line. Two tracks were set up by sticking adhesive tape on the floor perpendicular to the walking direction as seen in Figure 13 and Figure 14. The first track was 2.5 m long and the adhesive tape were placed with 50 cm intervals. The second track were 15 m long and had adhesive tape placed with 60 cm intervals. The purpose of the adhesive tape was to ensure a fixed step length and hence the two tracks were 5 and 25 steps long respectively.

Figure 13: Short track. 2.5 m long and with stickers every 50 cm.

Several tests were conducted on the two tracks and Figure 15 and Figure 16 show some typical measurements. Before and after each walk several seconds of stationary measurements were sampled in order to establish a clear point of reference for the orientation of the mote. This data was subsequently used to rotate the local reference frame as mentioned in Section 3.2. The measurements were also filtered for any corrupted data, by enforcing that time is moving forward and by removing unrealistic acceleration e.g. several g's.[19]

Figure 15 is a plot of measurement data taken from the short track and by looking at the z-axis it is possible to count the number of steps by counting the number of large peaks. The increased acceleration is caused by the fictitious force (measured by the mote) being directed towards the Earth's gravitational field (as explained in Section 3.2). It can be seen from the figure that exactly five steps were taken, which corresponds to the number of strips of adhesive tape on the track. By calculating the time difference between each peak it is also possible to deduce the duration of each step. By defining the middle of each step to be the acceleration turning point when the force applied to the mote is exactly equal to and opposite the gravitational acceleration. Table 5 shows a summary of each step duration.

No.	Middle <s>	Duration <s>
1	3.780	-
2	4.440	0.660
3	5.069	0.629
4	5.688	0.619
5	6.277	0.589

Table 5: Step duration

[19] These defects were probably caused by transmissions errors, either between the radios or the serial connection, since no error control were present.

Figure 14: Long track. 15 m long and with stickers every 60 cm.

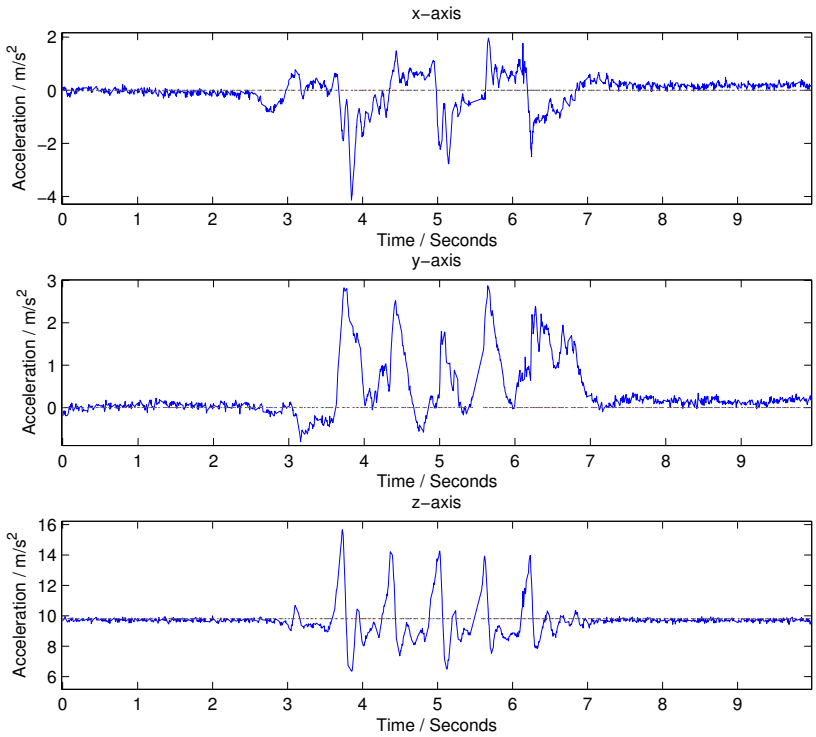

Figure 15: Walking experiment on the short track.

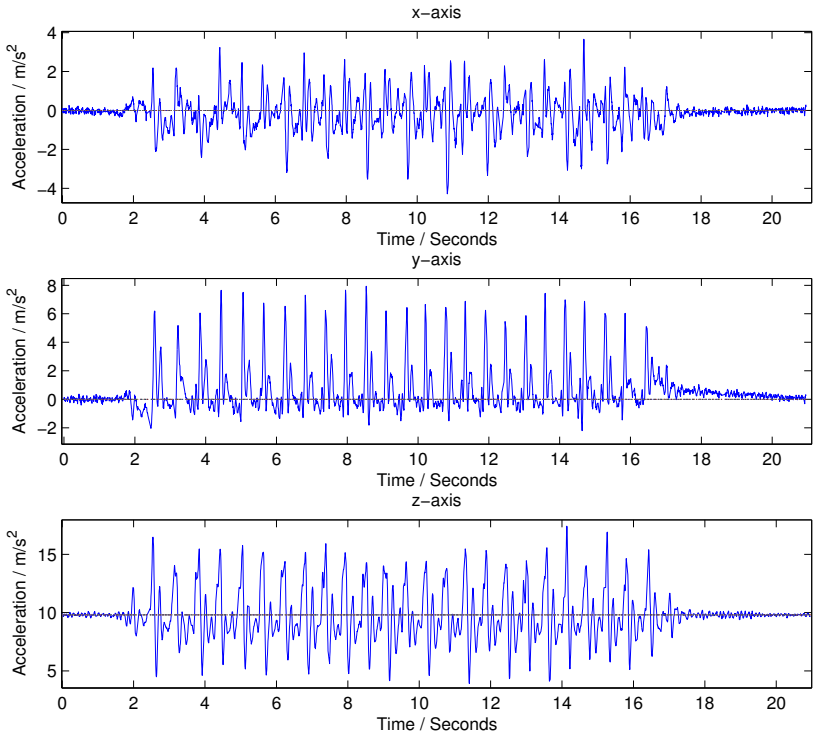

Figure 16: Walking experiment on the long track.

Unfortunately, besides the step duration it is not possible to deduce any quantitative information from the three plots. Ideally, the x and y-axis should be integrated twice with regard to time, in order to get the velocity and position as a function of time, which would be a plot of the walked distance and combined with the step duration each step length could be deduced. However, disregarding drift[20] the measurements for each axis should return to their original values at the end of the experiment since the test subject was standing still before and after the experiment. Since neither of the three axes have this behavior a simple integration would lead to a non-zero velocity at the end of the experiment and thus resulting in an unbounded position. This fact is most obvious by looking at the acceleration in the y-direction since it is easy to see that the majority of the acceleration is positive and hence leading to a positive velocity at the end.

The experiment leading to the measurements in Figure 16 was conducted on the large track and was carried out in the same manner as the one on the short. The figure shows great resemblance with the former, especially the z-axis, which again enables one to count the actual number of steps taken and calculating the duration of each step which has been summarized in table 6. Although the acceleration for all three axes return to their origin after the experiment is completed, the acceleration in the y-direction has the same problem as in the former experiment, since the overall acceleration is clearly positive which makes it futile to deduce a position from the acceleration.

No.	Middle <s>	Duration <s>
1	2.572	-
2	3.236	0.664
3	3.916	0.680
4	4.485	0.569
5	5.109	0.624
6	5.673	0.564
7	6.272	0.599
8	6.841	0.569
9	7.445	0.604
10	7.898	0.453
11	8.587	0.689
12	9.161	0.574
13	9.710	0.549
14	10.26	0.55
15	10.84	0.58
16	11.37	0.53
17	11.96	0.59
18	12.49	0.53
19	13.08	0.59
20	13.62	0.54
21	14.17	0.55
22	14.73	0.56
23	15.31	0.58
24	15.88	0.57
25	16.49	0.61

Table 6: Step duration

This trend of not being able to recover the actual events from the acceleration measurements (besides the step duration and count) is present in all the gathered measurements collected from walking with the accelerometers. Several different positions and motes were tried but all yielded the same result. A probable cause for this could be the rotation of the mote itself, hence leading to the equations of motion for the mote, not only including acceleration, but also rotation, since the coordinate system, which the accelerations are measured with respect to, itself is rotating. This sounds plausible since other Personal Navigation Systems such as the one mentioned in article [12] are equipped with gyroscopes to compensate for rotation of the reference coordinate system during movement. This would explain why the acceleration measured for each axis did not return to the original value

[20] Since the duration of the experiment is of the order of seconds, it is reasonable to disregard drift.

once the experiment was over since the reference point for each axis had been changed. This mixing of axes could also explain the cumulative velocity measured along the y-direction, since the oscillations on the mote could cause the overall buildup of acceleration along this axis (e.g. positive acceleration was mostly measured by the y-axis while negative acceleration was mostly measured by the x-axis).

Another possible cause could be interference on the ADC on the mote. If there was any cross-talk between the three channels on the ADC, any measurement in one axis would be tainted by values from the other, which could explain the cumulative velocity buildup.

A reference to the raw data files can be found in Appendix B.

3.5.5 Elevator

In order to measure whether the effects of the mote's rotational oscillations could account for the observed problems, the walking motion from the last experiment were transformed into a purely translatorical movement by replacing the human test subject with an elevator car. Since the motion of the elevator car is restricted by vertical tracks and controlled by an electrical engine, this would in effect give a repeatable environment with a close to perfect translatorical movement. The obvious downsides of this are the inability to automate the experiment process and that the primary measuring axis would be skewed by gravity which could lead to boundary effects since the measurements are being carried out at the boundary of the calibrated measuring interval.

The experiment was carried out at the public elevator at the William Gates Building which is two stories high. The distance between each floor was measured to be 3.50 ± 0.01 m. Four different kind of trips were carried out with the elevator:

1. Moving from the ground floor directly to the second floor.
2. Moving from the second floor directly to the ground floor.
3. Moving from the ground floor to the first floor and from the first floor to the second floor (with the only brake being the doors opening and closing on the first floor).
4. Moving from the second floor to the first floor and from the first floor to the ground floor (with the only brake being the doors opening and closing on the first floor).

As with the other experiments the measurements were filtered for corrupted transmissions and the reference coordinate system was rotated once to align the z-axis towards gravity. As an example the measurements of each axis sampled in experiment 4 are plotted in Figure 17 (the other experiments showed similar results).

First and foremost, it can clearly be seen from the measurements sampled from the x and y-axis that the motion indeed was translatorical since no acceleration can be measured in these two axes. Second, these two axes also show that if there is any crosstalk between the ADC channels it must be of the same order of magnitude or less than the background noise. Also, disregarding gravity, one can clearly see symmetry between the different peaks on the z-axis which one would expect since the elevator car was stationary at the beginning and at the end of each experiment. Since this is directly opposite of what was seen in Section 3.5.4 it seems reasonable to try to calculate the velocity and position of the elevator car.

After subtracting $g = 9.81258\ m/s^2$ from the z-axis and filtering the measurements with a moving average, a lowpass filter and a Savitzky-Golay[21] filter (in that order), the acceleration data was integrated twice (with respect to time) to yield the velocity and position as a function of time. These two functions can be seen in Figure 18.

It can clearly be seen that neither the velocity nor the position are bounded which seems odd as the symmetry in the acceleration would suggest that. Since the shape of the velocity and position curves, although skewed, seem to correspond to the behavior of the elevator, we suspect that the calibration of the accelerometers might be off and that the measured acceleration are offset from the real acceleration.

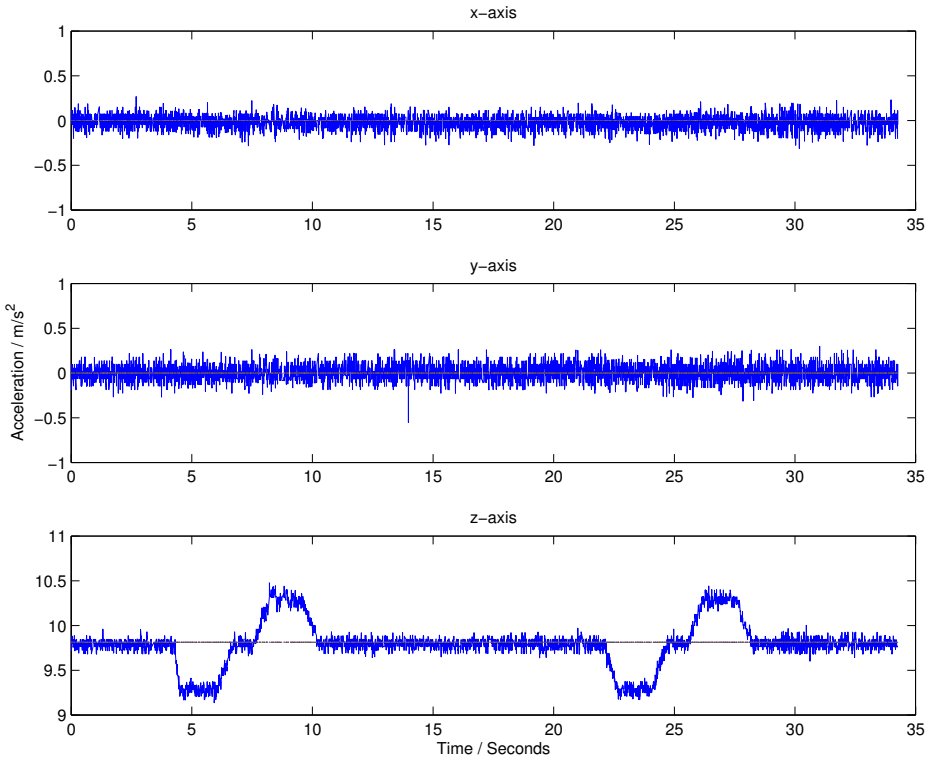

Figure 17: Measurements from an elevator going from second to first to the ground floor.

This seems to be the case since when we shift the value of the gravitational acceleration that is subtracted from the z-axis to $g = 9.7942\ m/s^2$ the plots of the velocity and position correspond to the theory as seen in Figure 19.

The two plateaus for the position correspond to -3.6 m and -7.2 m respectively and the maximum velocity is -1.0 m/s. This measured elevator car speed seems reasonable since typical speeds for elevator cars from this particular firm are 0.5, 1.0, 1.6 and 2.5 m/s [16]. Similar results were obtained from the three other elevator experiments and the values can be seen in Table 7. Values for g were chosen as to minimize the end velocity as much as possible.

Experiment	$g <m/s^2>$	max. velocity $< m/s >$	1st $< m >$	2nd $< m >$
1	9.8019	1.0	-	6.9
2	9.8026	-1.0	-	-7.1
3	9.7967	1.0	3.9	7.7
4	9.7942	-1.0	-3.6	-7.2

Table 7: Elevator experiments

With a new $g = (9.800 \pm 0.005)\ m/s^2$ the travelled distance of the elevator car was estimated to be $(7.2 \pm 0.5)\ m$. According to these experiments, performing an accurate calibration is of high importance since even when the calibration is off by as much as 0.1 % the results can change from being useless to fairly accurate. However, since the resolution of the Freescale accelerometer, when used in conjunction with the on-board ADC, is 0.0393 $\frac{m}{s^2 <ADCunits>}$ with the general noise level being of the order of 1-2 ADC units, it is not possible to achieve the high accuracy calibration that is required.

The problem described above is mainly caused by the movement being parallel to the gravitational acceleration

[21] This noise filter is better to preserve the area under the graph than a regular moving average. A mathematical definition can be found at http://www.mathworks.com/access/helpdesk/help/toolbox/curvefit/ch_data8.html

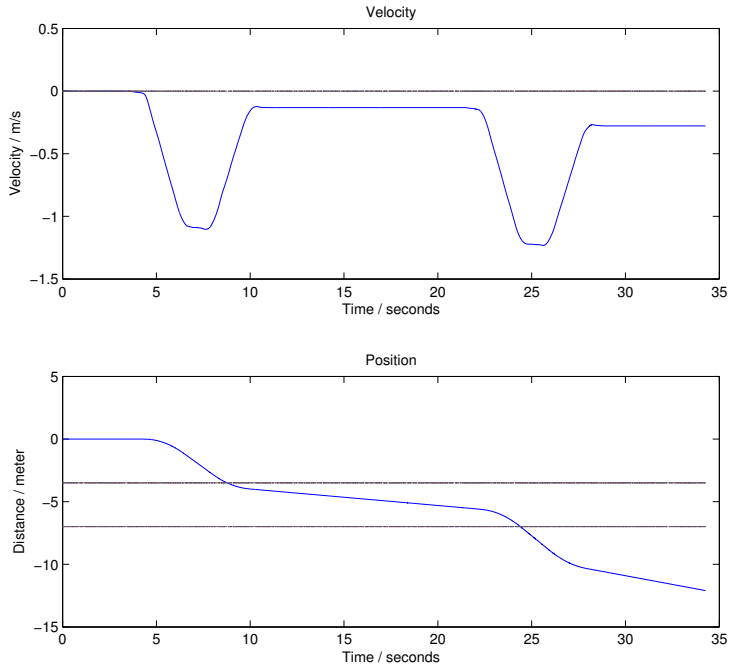

Figure 18: Velocity and position of elevator car with $g = 9.81258\ m/s^2$.

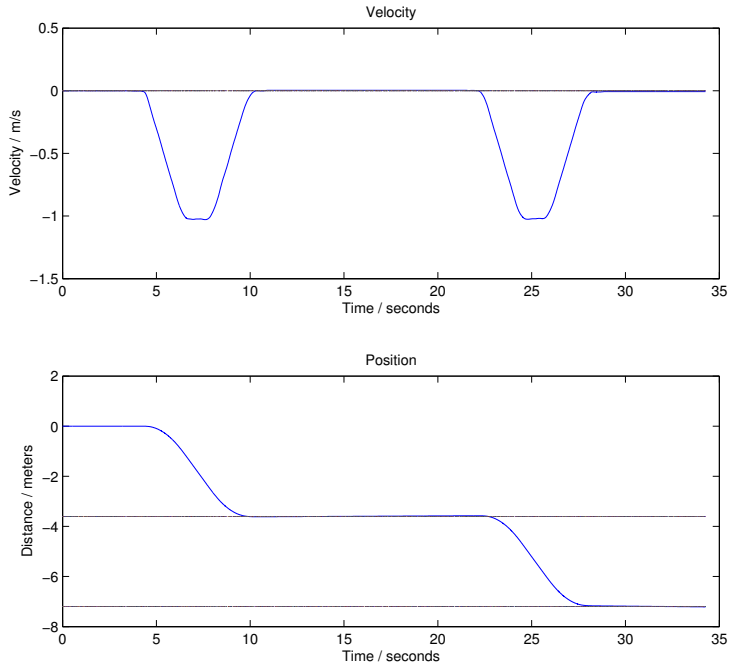

Figure 19: Velocity and position of elevator car with $g = 9.7942\ m/s^2$.

which must be removed in order to calculate the travelled distance. It remains to be seen if the axes orthogonal to the Earth's gravity field might be less sensitive to this calibration error.

In the light of the last two experiments we were discouraged to perform the same set of experiments with the MicaZ sensorboards, since rotation would have the same negative effect on these devices and the five time lower resolution would have a worse impact on the calibration and subsequently on the numerical integration.

A reference to the raw data files can be found in Appendix B.

3.6 Future Work

As seen by the huge differences in the results obtained from walking and driving the elevator, the need to compensate for rotation of the accelerometers is of paramount importance. Without the assistance of gyroscopes to adjust for these rotations of the local reference frame it is futile to use accelerometers to achieve a position from the acceleration. For future work this integration of gyroscopes must be performed.

Also, the data collection programs are far from complete. The measurement packets could be optimized by exploiting the fact that the ADC only samples measurements with 10-bit accuracy although this is represented with 16-bit in TinyOS. By reclaiming this unused space it would be possible to include an entire measurement more in each packet and hence increase the overall sampling rate. The gateway could also be made more robust for transmission bursts by increasing the number of queueable packets before one is dropped.

The power level remains an issue as well, since a depleted battery renders the accelerometers useless. A workaround for this problem could be to constantly monitor the noise level (the deviation from the current mean value) locally at each sensor mote and sound an alarm if the noise rose above a certain threshold. This would however effect the overall performance negatively. Another option would be to attach a voltmeter to each sensor mote, and regularly read the voltage on the battery. This could be done with minimal additional hardware by using the on-board ADC as a simple voltmeter although this would increase the power consumption.

The drift from a stationary accelerometer was measured, however, it remains to be seen if a moving accelerometer has the same drift. This could be measured by moving the accelerometer in a known oscillating pattern and observe deviations over time from this pattern (e.g. by mounting the mote on an electrical model train moving backworth and forworth on a fixed track and subsequently measure the differences in the acceleration pattern).

Finally, as seen in Section 3.5.5, the measured accelerations were not precise enough to be used for position estimation by numerical integration. Future experiments should be done with accelerometers with both a higher sensitivity and a lower noise density than the ones used in this experiment. Preferably, the accelerometer should be capable of measuring the gravitational acceleration with a precision of at least two orders of magnitude higher than the desired precision on the position.

3.7 Summary

A set of TinyOS applications was developed to facilitate sampling of measurements and even with room for several optimizations we achieved a system wide sampling rate of 200 Hz. Calibration measurements were collected for the different accelerometers and an upper bound on the precision was found to be 0.0393 m/s^2 and 0.196 m/s^2 for the SARD and MicaZ mote respectively. The zero-motion drift was measured on both the MicaZ mote as well as the SARD mote and we found that only the MicaZ mote had detectable drift. We also found the power supply to be a source of error if the voltage drops below a certain point.

Several experiments were performed by walking on a straight line carrying accelerometers and, although it was possible to deduce the step duration and the number of steps taken, the experiment especially showed the effects that rotations of the local reference frame can have on the acceleration measurements and the subsequent need for gyroscopes to compensate for these rotations. This was even more clear in the last experiment where the rotation factor had been removed by using an elevator car to ensure a translatorical movement. This experiment yielded, however, another source for error namely the sensitivity of the calibration and that the given accelerometers lacked the desired precision. With this taken into account it was possible to deduce the velocity and position of the elevator car with 10% accuracy.

4 Time Synchronization

One of the key aspects of a WSN is the ability to observe the environment simultaneously from multiple sites. In order to compare measurements collected from different motes a temporal ordering is often necessary. Especially if several motes are monitoring the same event the collected data must be in temporal agreement.

Since the overall project goal is to monitor the performance of athletes with several wireless sensor motes placed on different parts of the athlete's body, it is essential to have a fine grained time ordering of the collected measurements in order for the data to be compatible. This involves first and foremost a timestamping of all the collected measurements. Nevertheless, since this has to be done at each and every mote in the WSN the clocks, from which the timestamping originates, must be synchronized in order for the timestamps to be compatible.

The purpose of this part of the project is to investigate and implement a clock synchronization protocol in the data collecting architecture developed in the previous chapter. Since there are several fundamental constraint differences between regular computer networks and WSN, it should be emphasized that the protocol in question must honor these constraints and be scalable in order to be suited for a WSN.

Section 4.1 describes work related to time synchronization in WSN. Section 4.2 analysis two different protocols based on broadcasting and Section 4.3 discusses several implementation issues. Section 4.4 evaluates the accuracy of the implemented time synchronization. Finally Section 4.5 describes different aspects future work might focus on and Section 4.6 summarizes this chapter.

4.1 Related Work

The field of time synchronization for WSN have been thoroughly studied and much of the work has been summarized by [17] and [18]. Particularly one of the paradigms studied is the usage of message broadcasts as a common point of reference.

This technique was first theorized by [21] which also established an upper bound of $(1 + \frac{1}{K})\varepsilon$ on the precision attainable after K broadcasts, where ε is the time it takes to receive a broadcast message. Since the wireless medium facilitates broadcasts, this approach is particularly well suited for WSN. Actual implementations based on this work have been carried out by [20] and [19].

In [20] an internal synchronization of the network is performed by having a reference mote broadcast message beacons. The receival times of these beacons are captured by each of the other motes and put in a local timetable. By exchanging timetables with all the motes, each mote is capable of calculating an offset that minimizes the overall RMS error of the local timetables. The drawback of this approach is the increased overhead caused by the exchange of timetables among the motes, which has a quadratic increase in the number of participating motes.

This lack of scalability was improved by [19] which assumes that the transmission delay time between the reference mote and each of the other motes is the same. By having the reference mote calculate the transmission time to one particular mote and afterwards broadcast this delay to all the other motes, all motes within communication range can be synchronized with a total of three messages transmitted. As will be shown later, this can be improved to a total of two transmitted messages.

4.2 Protocols

The main target platform for this time synchronization is the collection of acceleration measurements from sensor motes placed on athletes. Since a high sampling rate of the accelerometers is desired the time synchronization protocol must be as light as possible in order not to decrease the overall performance.

The system we want to time synchronize consists of a logger program located on a PC, a gateway program connecting the sensor motes with the PC and the actual application on the sensor motes. The data communication in this setup is highly asymmetrical with the sensor motes transmitting data to the gateway which in turn transmits data to the PC. The gateway is in one-hop communication range with all the motes. This also means that the gateway is a common reference point suitable for the other motes to synchronize with. Since the gateway also is permanently connected to the PC, this connection can be used to synchronize the gateway with the local time on the PC as well.

4.2.1 Time Beacon

If only time synchronization between the individual sensor motes is required e.g. there is no need to associate each measurement with real time, a simple time beacon protocol can be used. Since all the motes are in direct communication range with the gateway, a packet containing a timestamp broadcasted from the gateway will reach all the sensor motes simultaneously if one disregards the propagation time of electromagnetic waves.[22] Each sensor mote can thus calculate an offset from the motes local time to the timestamp and by adding this offset to the local time all the sensor motes will in effect be synchronized with each other.

This protocol does not account for drift and clock skew which means that at some point the local time on each sensor mote will be unaligned again. This can be compensated for by transmitting another time beacon before the drift and skew becomes significant. Since this protocol does not account for the transmission delay, it is not possible to extend it to a multi-hop environment. Also, since the sensor motes are not synchronized with an external source, it would not be possible to extend this protocol to encompass several gateways that are not in communication range.

4.2.2 Reference Broadcast

To overcome the short comings of the time beacon protocol mentioned above it is necessary to perform an external synchronization which in turn requires the knowledge of the transmission delay. In order for mote A to calculate the most accurate transmission delay to mote B at least two packets and four timestamps are required:

- Mote A transmits a packet with the timestamp, t_0, of the transmission time.
- Mote B timestamps the packet upon receival with the arrival time, t_1.
- Mote B returns the packet to mote A, but timestamps the packet with the transmission time, t_2, as well.
- Mote A timestamps the packet with the receival time, t_3, and is now in possession of four timestamps.
- By calculating the difference $t_3 - t_0$ mote A has the Round-Trip-Time (RTT).
- By calculating the difference $RTT - (t_2 - t_1)$ mote A has the total transmission time and by dividing with 2 mote A also has the average transmission time.
- The average transmission time combined with the timestamp t_2 enables mote A to synchronize itself with mote B.

This is known as Christian's method [22]. In order for this algorithm to be effective it is necessary to do the timestamping as close to the physical layer as possible or else the time difference $t_2 - t_1$ will be neglible.[23]

Since the radio transmission bandwidth is limited and mostly saturated with acceleration measurements it would be infeasible to do a sensor mote-to-gateway RTT calculation for each and every mote. This approach would also have the defect of not being scalable to larger networks.

Instead of calculating the exact RTT for each mote we use an approximation of the RTT as proposed in [19]. The principle is called Reference Broadcast and is based on the same fact as the time beacon protocol that broadcasts arrive close to simultaneously at all the sensor motes in communication range. By appointing a master mote to initiate a RTT-calculation with the gateway (transmitted as a broadcast), the other motes can use the RTT-request and reply to estimate the RTT. This approach is different than that proposed in [19] since both messages are transmitted as broadcasts and hence each mote in the network is capable of calculating the RTT. Also, since the message flow is highly asymmetrical, the gateway should have a lower reply latency than the sensor motes because the former only receives messages while the latter transmit them.

[22] Since this is close to the speed of light traveling in the atmosphere, the time delay caused by propagation for motes placed within meters of each other will be of the order of nanoseconds. Since this is several orders of magnitude smaller than the desired accuracy it is reasonable to disregard this time delay.

[23] It should be noted that all timestamps are taken with respect to the local clock and are thus not directly compatible, however since the time difference is only calculated with timestamps taken from the same mote e.g. both t_2 and t_1 are from mote b, the calculated time differences are compatible with one another.

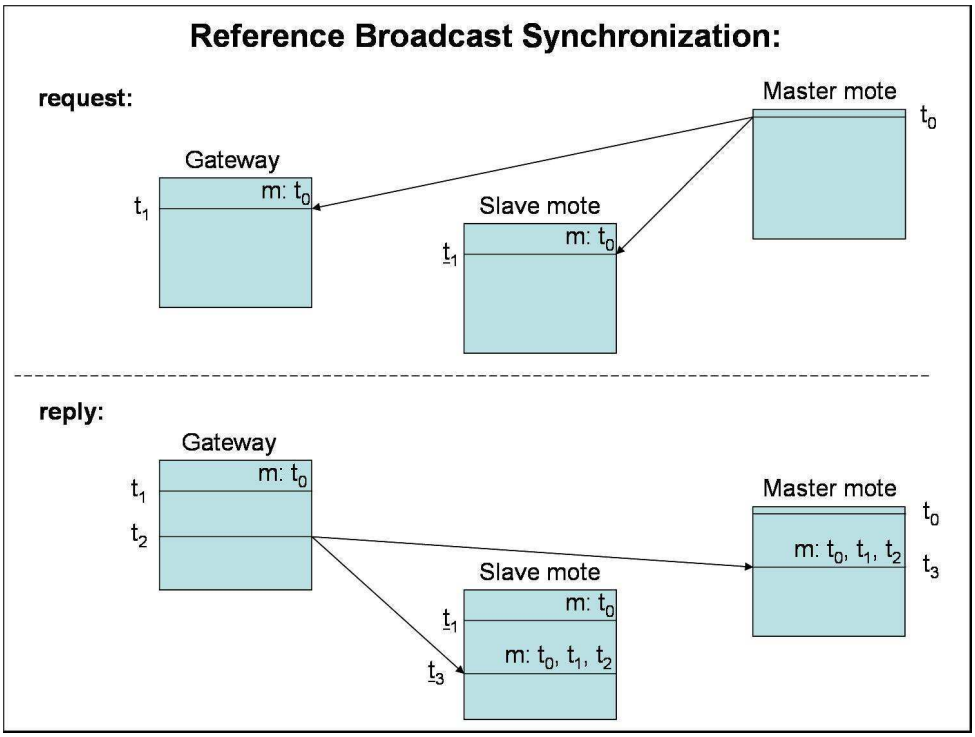

Figure 20: Overview of the Reference Broadcast Synchronization protocol.

The protocol is illustrated in Figure 20. The master mote initiates synchronization with the gateway by broadcasting a synchronization request containing the local transmission time t_0. This is symbolized by the two arrows on the upper part. At time t_2 the gateway broadcasts the reply containing t_0, t_1 and t_2 (where t_1 is the receival time of t_0). The master mote then has the complete information to perform a regular RTT calculation by using the algorithm mentioned above. The slave mote(s) however cannot do the same calculation since the timestamps t_0 and $\underline{t}_3$ are not compatible since $\underline{t}_3$ is measured with a different clock than t_0. Instead an estimate of the transmission time can be calculated by assuming that $t_1 \approx \underline{t}_1$. The partial RTT is thus $\underline{t}_3 - \underline{t}_1$ and the transmission time from the gateway to the slave mote can be calculated by:

$$T_{transmission} = (\underline{t}_3 - \underline{t}_1) - (t_2 - t_1)$$

Since only the partial RTT is measured, it is not necessary to divide by 2 to obtain the transmission time.

This protocol scales well with larger networks and is extendable to multi-hop networks as well. It does not account for clock drift and skew, but this can be compensated for, as mentioned in the previous section. One flaw with this protocol, however, is the presence of a master mote, which is a single-point-of-failure. In order to overcome this problem a rotation and/or an election scheme must be added to ensure fault-tolerance.

4.3 Implementation Issues

Since the current MAC-layer for the Freescale MC13192 has not been fully implemented in TinyOS[24] yet, it was not possible to implement MAC-layer timestamping of packages and thus the value $(t_2 - t_1)$ could not be measured since this this difference was neglible at the application layer given the current resolution of the clock. Since $t_1 \approx t_2$ only one timestamp was added to the synchronization reply instead of two.

In this simple implementation the master mote mentioned above was one of the sensor motes since these transmit a constant flow of measurement packets. Since each measurement packet contains the transmission time, each measurement packet can be used as a RTT-request packet. However, since a reply is required from the gateway it would not be feasible to turn all the measurement packets into RTT-requests. It is, however, part of the TinyOS

[24]The current TinyOS radio stack is actually a wrapper around a Freescale proprietary MAC-layer and thus not easy to modify.

packet header to include the packet type, and by defining a new packet type that symbolizes a measurement packet with a piggybacked RTT-request, time synchronization can be initiated without the need for any extra packets. The RTT-reply cannot be piggybacked though since no transmissions from the gateway to the sensor motes are necessary during data collection. This packet is unavoidable but fortunately it only needs to contain two timestamps.

The gateway was synchronized with the PC in a similar manor except that the RTT-request was a completely separate packet. Although it would be possible to use the piggy-bagged RTT-request from the master mote to synchronize the gateway with the PC as well, it would not be optimal since only after the second synchronization would the motes be synchronized with the PC.

With time synchronization embedded in the acceleration collection application, the application was modified to enforce transmission windows to minimize packet collision. By dividing the time interval between each packet transmission with the sensor motes and the gateway, and using the mote address to allocate time slots, the process was self-configuring.

4.4 Evaluation

The evaluation of the time synchronization protocol has been divided in three sections. First the Round-Trip-Time is measured in Section 4.4.1, second the drift is measured in Section 4.4.2 and finally the precision is measured in Section 4.4.3. However, since the MicaZ motes were not able to receive messages transmitted by the SARD motes, and thus unable to participate in the time synchronization, only the latter were used in the experiments.

4.4.1 Round-Trip-Time

In order to measure the calculated RTT at each mote over a period of time, a network consisting of four SARD motes and one gateway were resynchronized every 600 ms. The time difference between the RTT-request and the RTT-reply can be seen in Figure 21 and a reference to the data can be found in Appendix C.

As mentioned above since the slave motes only calculate a partial RTT (compared to the master mote's RTT), the slave motes should have a difference half of that of the master mote.

This explains the lower times for the slave motes compared to the master mote as seen in Figure 21 to a certain degree. However, since these times are generally lower than half of that of the master mote this deviation is probably caused by the RTT calculation being done at the application level and with only two timestamps instead of four. It should be noted, however, that the three slave motes have an almost identical distribution of time differences which means that the broadcast messages were indeed received almost simultaneously at the three slave motes. This shows that the assumption in which the Reference Broadcast protocol was based upon[25] is indeed sound.

By looking at the variances between half the master mote's RTT and the slave mote's RTT one should not expect the precision attained from this synchronization to be higher than 1-2 ms.

4.4.2 Clock Drift

The clock drift is the deviation observed with respect to some reference frame after a certain amount of time. By synchronizing all the motes once and subsequently calculate the offset from some reference clock, the drift can be measured. Two reference clocks were used: the PC clock and the master mote clock.

Since all packets are timestamped upon receival at the PC and the PC clock is used by the sensor motes to perform the external synchronization, the offset between the PC and the sensor motes can be used as a measure of the absolute drift of the sensor motes' clocks. This difference between the PC and the sensor motes' clocks can be seen in Figure 22.

It can clearly be seen from the figure that the change of offset are almost identical for all four motes since they all possess the same saw-tooth characteristic. Also, the order of magnitude is several times larger than that of the

[25] All broadcasted messages are received (almost) simultaneosly by neighboring motes.

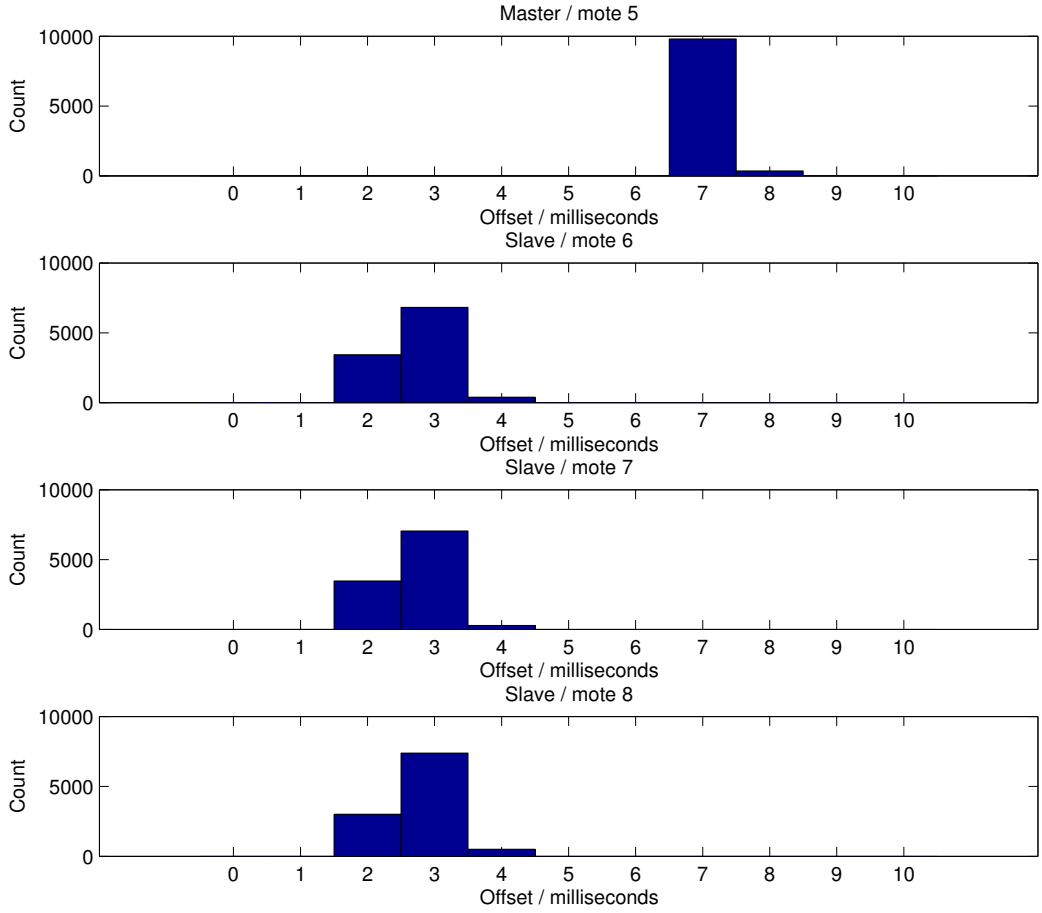

Figure 21: Histogram of 10,000 Round-Trip-Time calculations. The histogram for the master mote is the full RTT while the ones for the slave motes are the partial RTT.

measured transmission time which can thus be disregarded. The cause of this systematic drift must be the lack of precision of the PC clock, caused by the differences in process scheduling by the operating system on the PC.

By looking at the relative differences between the master sensor mote's clock and the clocks on the other motes, it is possible to remove the PC's clock as a source of error and hence measure the relative clock drift. Since the timestamps taken from each mote represents a discrete function, evaluated at different points in time, they are not directly compatible. The timestamps used for the relative offset calculations were found by interpolating the timestamps from the three slave motes. These three continously functions were subsequently evaluated at the same points in time as the master mote, in effect creating a new set of compatible discrete functions. The relative offsets can be seen in Figure 23.

Compared to the former figure the offset is now of the same order of magnitude as the variance of the transmission time which seems more reasonable since this is of the same order as the expected achievable precision. It is also clearly seen that mote 7 has a linear drift compared to the other motes which can thus be compensated for. In order to maintain a precision similar to the variance of the transmission time (1-2 ms) it is thus necessary to resynchronize the sensor motes at least every 2 min. (120000 ms).[26] Similarly, an accuracy of 4 ms can be maintained by resynchronizing at least every 4 min. A reference to both data collections can be found in Appendix C.

[26]This figure is taken as an estimate based on Figure 23 and has the noise taken into account as well.

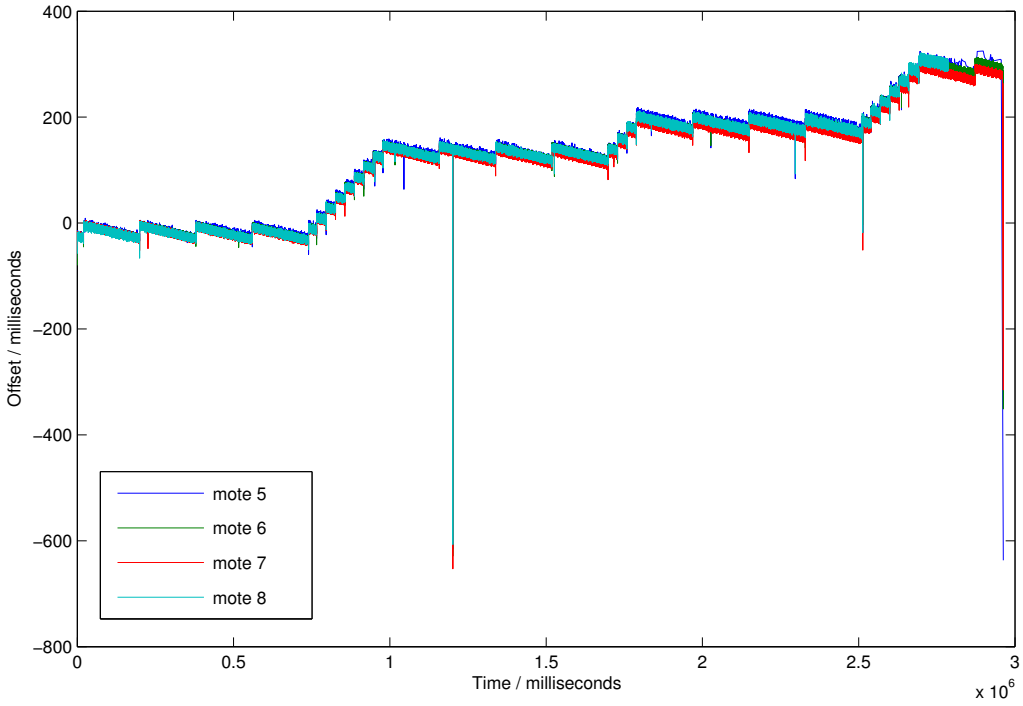

Figure 22: Evolution of time differences between the PC clock and the four sensor motes' clocks after one synchronization.

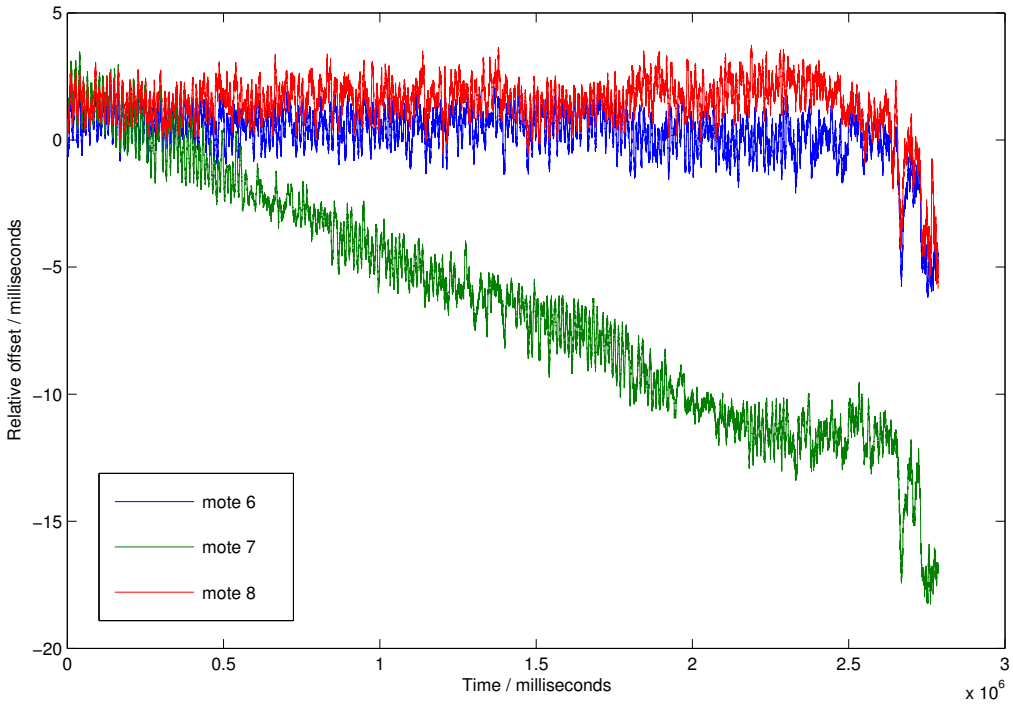

Figure 23: Evolution of time difference between the master mote and the three slave motes after one synchronization. Each graph represents the time difference between a slave mote and the master. All three graphs have been smoothed with a moving average with a window size of 100.

4.4.3 Agreement

The most important reason to perform time synchronization in a WSN is the need to be able to combine measurements taken from several different sensor motes without losing the temporal dimension. This can be evaluated by comparing measurements taken from different sensor motes monitoring the same event. In this case the four SARD motes were strapped to a metal beam and with the motes synchronizing every 10 s the beam were shaken by turn in all three directions parallel to the accelerometers' axes. In order for the time synchronization to be successful, the measurements must be in agreement with each other. However, since the magnitude of the acceleration is dependent on the calibration of each sensor, one should not expect the motes to agree on the size of the acceleration but only on when it occurred, e.g. peaks in the acceleration corresponding to the extreme points of the beam's position can be compared directly.

Figure 24 shows the acceleration measurements sampled from the four sensor motes. The three rectangular boxes mark the movement of the beam parallel to the axis in question and a close-up of each box can be seen in Figure 25. By looking at the time difference between the first and last mote crossing the zero-acceleration line one can estimate the accuracy of the time synchronization to be of the order of 4 ms. An example of such a reading can be seen in Figure 26. This is worse than the former estimate which was based upon the variance of the transmission time. A probable cause is the timestamping taken place at the application layer which is prone to errors caused by the current workload. This seems reasonable since the gateway in the former experiment only processed packets every 150 ms on average while in the latter the average time between packets has decreased one-tenth to 15 ms.

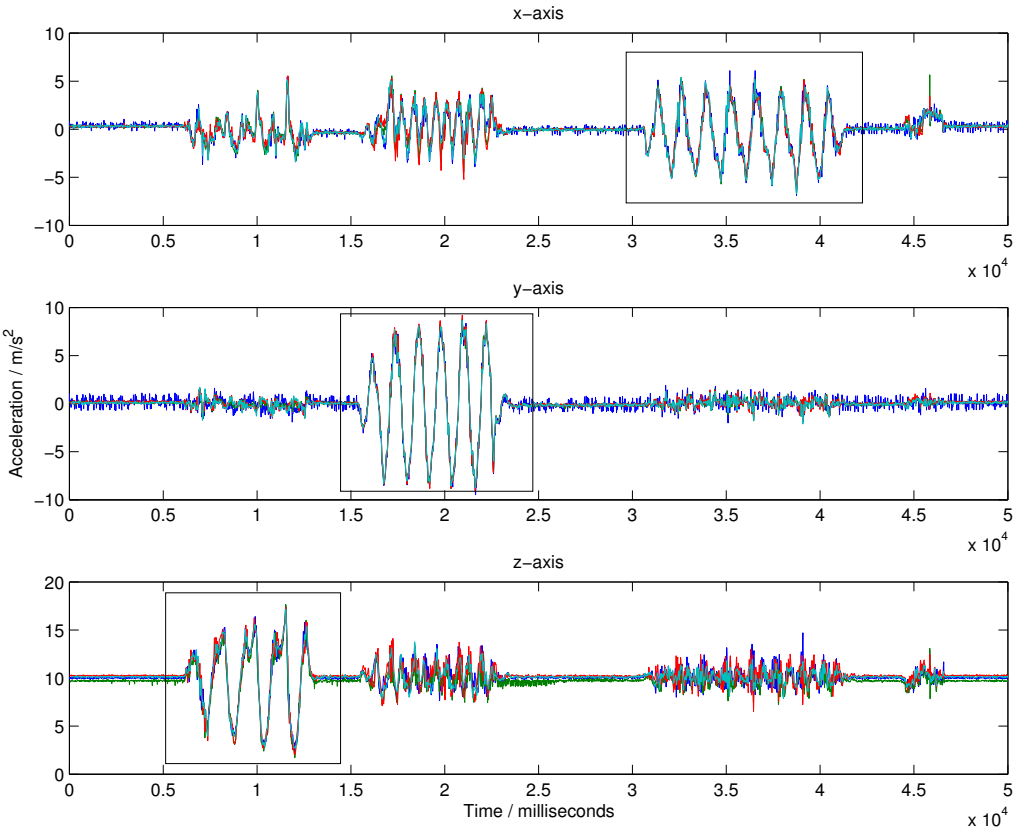

Figure 24: Acceleration measurements taken from four motes synchronized with Reference Broadcast. All motes are measuring the same event and each mote are represented by a distinct color.

Subsequently, the experiment was repeated with two gateways instead of one. The WSN were partitioned in two identical parts consisting of one gateway and two sensor motes by assigning a different radio frequency to each partition. The four sensor motes were still attached to the same beam however. The purpose of this experiment was to simulate an environment were measurements collected from different WSN had to be compared solely on

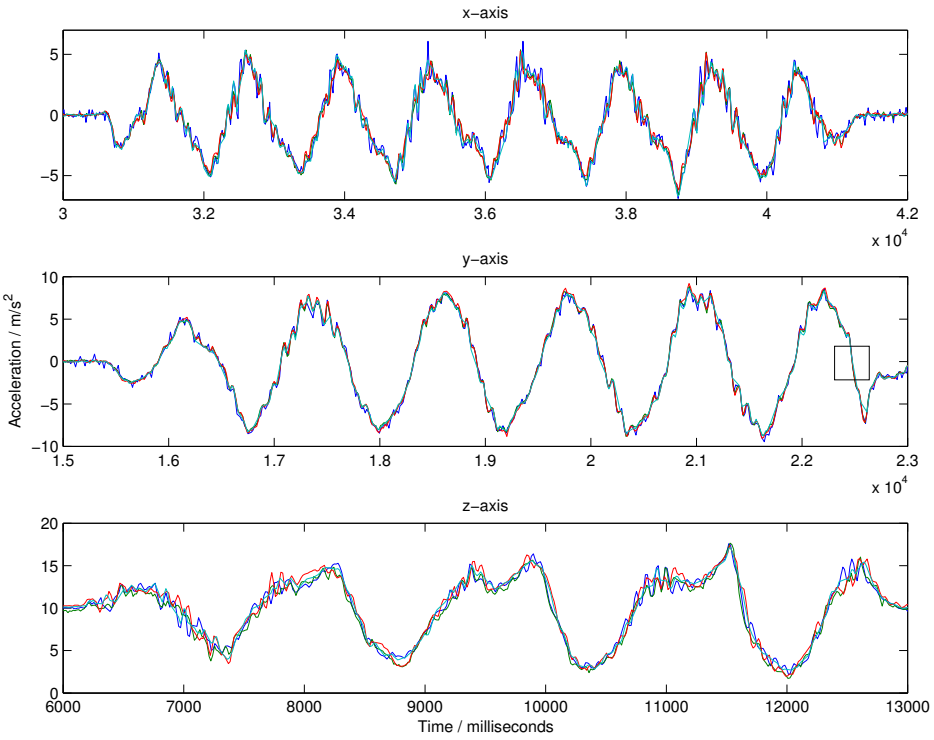

Figure 25: Close-up of each of the axes as marked with rectangular boxes in Figure 24.

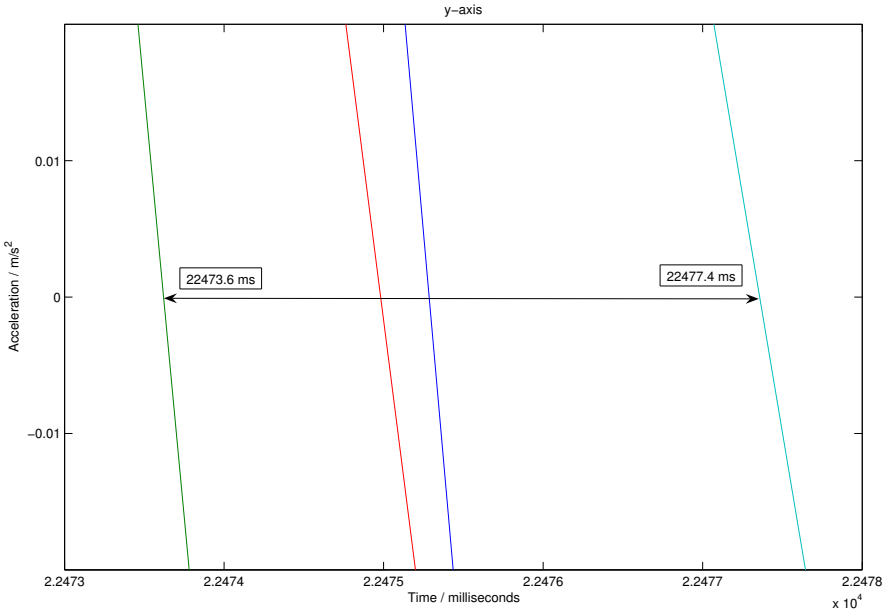

Figure 26: Close-up of the y-axis as marked by the rectangular box in Figure 25. The arrow represents the magnitude of the disagreement between the four motes when measuring the same event. As indicated by the readings the accuracy of the time synchronization is of the order of 4 ms.

the timestamp. This could also be seen as a large WSN where measurements were collected by the use of two distinct gateways instead of using one gateway and a multi-hop protocol.[27] As seen in Figure 22, the sampling of the PC clock is not very accurate. Hence the two gateways were connected to the same PC to minimize this source of error.

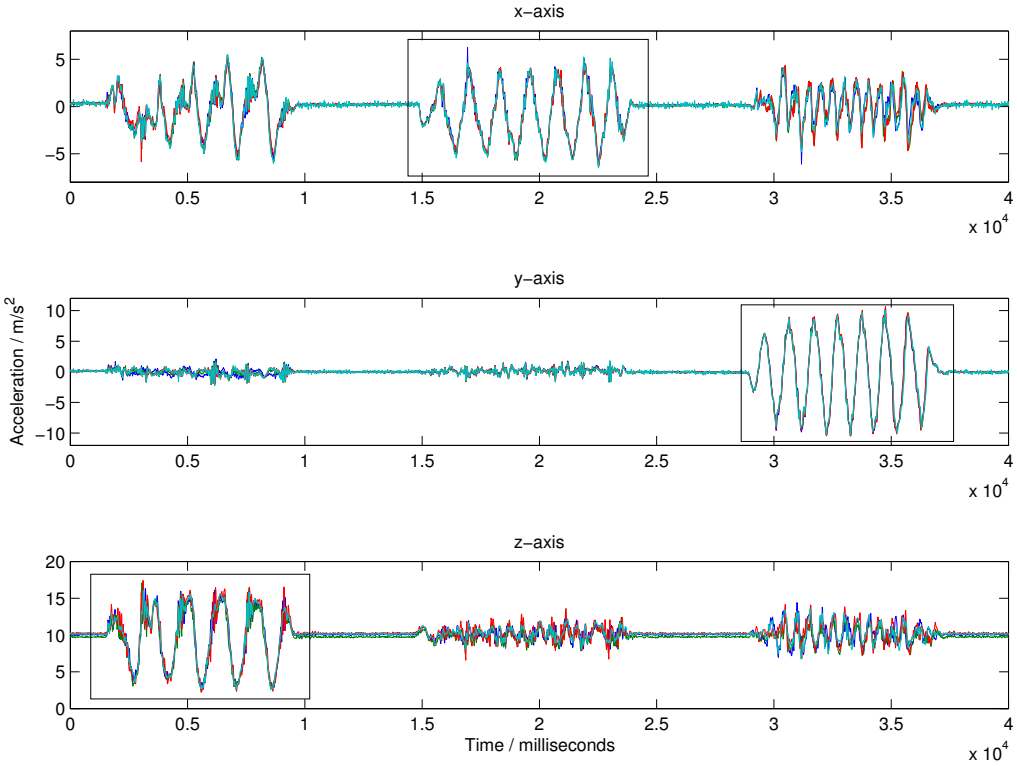

Figure 27: Acceleration measurements taken from four motes synchronized with Reference Broadcast similar to Figure 24 but with the use of 2 gateways. All motes are measuring the same event and each mote are represented by a distinct color with the motes represented by blue and red belonging to one gateway and the motes represented by cyan and green belonging to the other.

Figure 27 shows the acceleration measurements collected while the beam was shaken. The three rectangular boxes mark the movement of the beam parallel to the axis in question and a close-up of each box can be seen in Figure 28. An estimate of the accuracy measured from figures similar to Figure 29 shows this to be of the order of 27 ms. This is not surprising since the fluctuations from the PC clock observed over a short time period is of this order of magnitude as seen in Figure 22.

A reference to the data files can be found in Appendix C.

4.5 Future Work

As seen in the last section, the accuracy of the synchronization was of the order of milliseconds which is several orders of magnitude worse than the reported microsecond accuracy reported in [20] and [19]. Two sources of error can easily be identified as being the lack of MAC-layer timestamping and the millisecond resolution of the clock. Since without the actual transmission and receival timestamps at the lower level the calculated transmission time also includes the time spent processing at the upper levels, which is highly dependent on the current workload and thus susceptible to great variation. When a fully native TinyOS radio stack has been implemented on the

[27] This would be advantageous in a high performance environment where the overhead of packet routing would have a negative impact on performance.

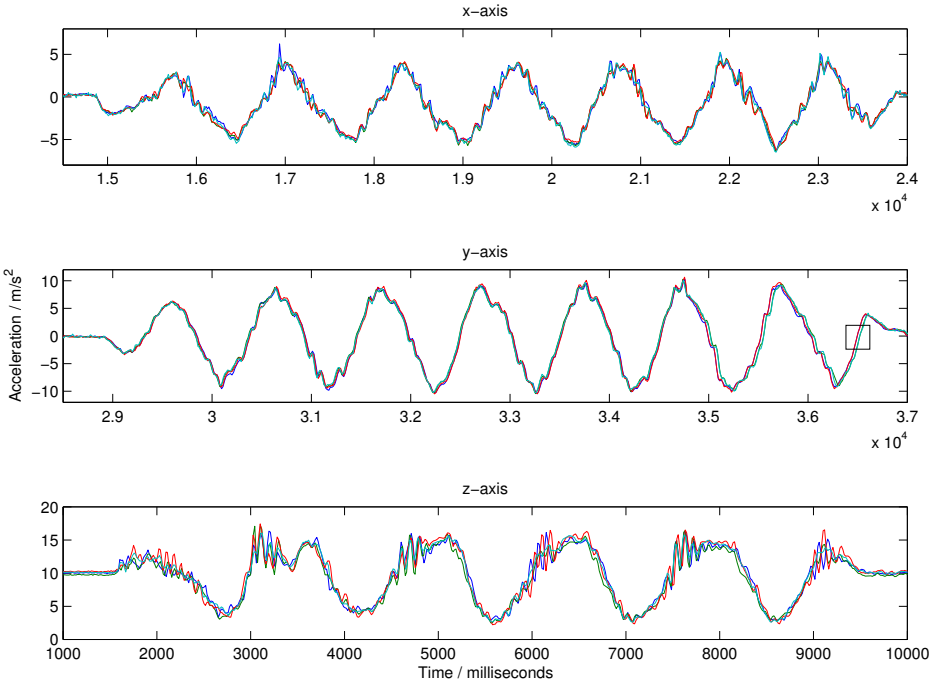

Figure 28: Close-up of each of the axes as marked with rectangular boxes in Figure 27.

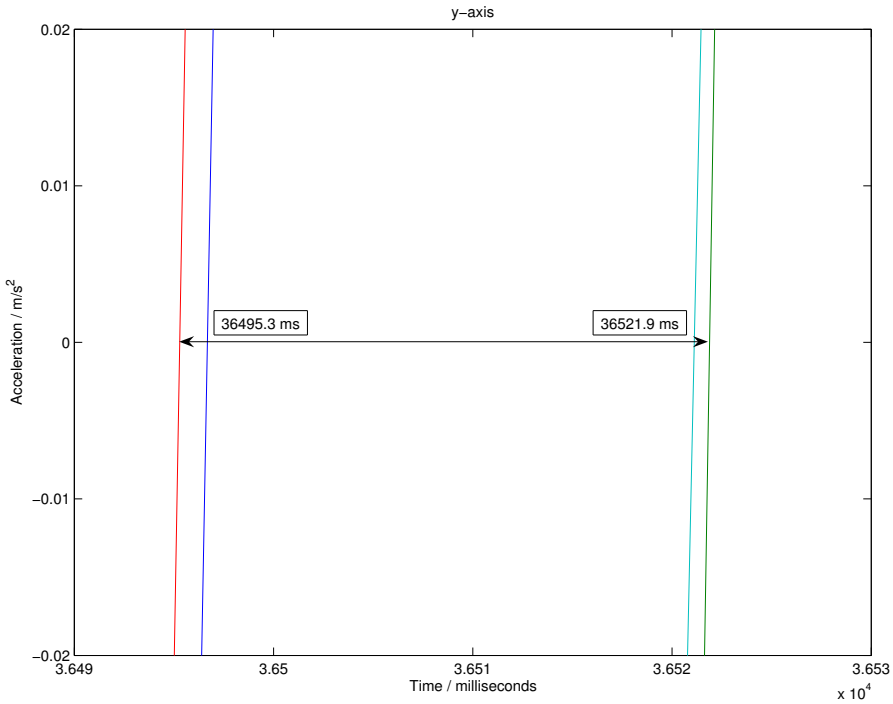

Figure 29: Close-up of the y-axis as marked by the rectangular box in Figure 28. The arrow represents the magnitude of the disagreement between the four motes when measuring the same event. As indicated by the readings the accuracy of the time synchronization is of the order of 27 ms when using two gateways.

Freescale MC13192 radio, MAC-layer timestamping combined with increased clock resolution should greatly improve accuracy.

The current implementation assumes that all the motes (master, slave(s) and gateway) are within communication range. This is however not always the case and the need for a multi-hop adaption is necessary. Similarly it would be advantageous to have multiple gateways collecting data in unison. It would then be necessary to be able to distinguish between synchronization replies transmitted from different gateways in order to calculate the exact transmission time.

As seen in Figure 22 the PC clock is the largest source of error when an external synchronization is performed. If accuracy in the microsecond range is the goal a more precise external clock must be used. This could be a GPS clock connected to either the PC or gateway since many of these devices can operate with nanosecond accuracy.

Another improvement would be a rotation scheme among the motes to remove the single-point-of-failure from the master mote. This scheme could also be responsible to elect a new master mote in the case of a failure. Algorithms for the rotation and election scheme have been studied extensively and protocols such as the Bully algorithm [23] could be used to elect at new master.

4.6 Summary

A 2-tier external time synchronization protocol was implemented on the Freescale SARD. First, the gateway were synchronized with the connected PC by Christian's method. Second, the sensor motes were synchronized with the gateway with a modified version of the Reference Broadcast Synchronization. The protocol were adapted to the highly asymmetrical and continuous data flow in the previous developed acceleration collection architecture resulting in a minimal amount of packet overhead caused by the time synchronization.

With application level timestamping and a clock resolution of 1 ms subsequent experiments showed the synchronization between the sensor motes and a single gateway to be accurate within 4 ms. Further experiments also found the accuracy between different WSN connected to the same PC to be 27 ms. This increase is primarily caused by the inaccurate reading of the PC clock since experiments showed the drift of this clock to be of this order of magnitude. The relative time difference between the sensor motes were seen to drift and the difference became comparable with the synchronization accuracy of 4 msec after a period of 4 min. The network thus has to be resynchronized at least this often in order to maintain the accuracy.

5 Conclusion

The main goal of this project was to investigate the precision of the physical properties obtained from acceleration measurements, specifically those obtained from accelerometers mounted on wireless sensor devices. Two different types of devices were used in order to accomplish this task: the Crossbow MicaZ and the Freescale MC13192SARD.

In order to develop TinyOS applications to both platforms, several modifications were made to the motes and the TinyOS contribution from DIKU. Specifically, the embedded bootloader was changed to one compatible with executables generated with TinyOS, the component responsible for UART communication was changed to use the RS-232 port instead of the USB port, and, finally, three components were developed to sample the ADC ports where the accelerometers were connected. These components were subsequently used to develop a data collection architecture capable of collecting measurements from both hardware platforms and we achieved a system wide sampling rate of 200 Hz.

Subsequently, the data collecting architecture was augmented with a 2-tier external time synchronization protocol on the Freescale SARD. Further evaluations found the accuracy of the synchronization to be within 4 ms for a single WSN with one gateway and 27 ms for two WSN connected to the same PC. With these applications acceleration measurements were collected from different experiments.

First, the accelerometers were calibrated in order to convert the readings to SI units and an upper bound of the accuracy was found to be 0.0393 m/s^2 and 0.196 m/s^2 for the SARD and MicaZ mote respectively. This is caused by the combination of voltage range on the accelerometers and the resolution of the ADC.

Second, the zero-motion drift was measured and we found that only the Crossbow motes had detectable drift. This was caused by the lack of temperature correcting circuitry which was present on the Freescale motes.

Third, it was discovered that the power voltage had a significant negative influence on both the noise and value of the acceleration measurements resulting in increased noise and drifting values.

Fourth, the necessity of knowing the orientation of the accelerometers became clear when measurements obtained during walking yielded no useful information except the step duration. This fact was emphasized when a similar experiment with fixed axes showed the velocity and position of an elevator car with 10% accuracy.

Fifth, the sensitivity of the calibration and the acceleration as a means for calculating the position were investigated and it was found that the calibration had to be accurate within 0.1% in order to measure position parallel to the Earth's gravity. As a result the accelerometers used in this project were not sensitive and precise enough to deliver the needed accuracy. It is unclear, though, if measurements obtained from axes perpendicular to Earth's gravity have the same instability.

References

[1] Freescale Semiconductor Inc.: “Sensor Applications Reference Design (SARD) User’s Guide”, MC13192SARDUG, Rev. 1.5, 07/2005. http://www.freescale.com/files/rf_if/doc/user_guide/MC13192SARDUG.pdf

[2] Freescale Semiconductor Inc.: “MC13192 Evaluation Board Reference Manual”, MC13192EVBRM/D, Rev. 0.0, 08/2004. http://www.freescale.com.

[3] Freescale Semiconductor Inc.: “MC9S08GB60/GB32/GT60/GT32GT16 Data Sheet HCS08 Microcontrollers”, http://www.freescale.com.

[4] Freescale Semiconductor Inc.: “MC13192 2.4 GHz Low Power Transceiver for 802.15.4”, MC13192/D, Rev. 2.4, 07/2004. http://www.freescale.com.

[5] Freescale Semiconductor Inc.: “+/- 1.5g Dual Axis Micromachined Accelerometer”, MMA6160Q, Rev. 2, 10/2004. http://www.freescale.com/files/sensors/doc/data_sheet/MMA6260Q.pdf

[6] Freescale Semiconductor Inc.: “ Low G Micromachined Accelerometer”, MMA1260D, Rev. 1, 10/2004. http://www.freescale.com/files/sensors/doc/data_sheet/MMA1260D.pdf

[7] Crossbow Technology Inc.: “MOTE-KIT2400 Datasheet”, 6020-0062-01, Rev. C. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Kit_Datasheet.pdf

[8] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler and Kristofer Pister: “System Architecture Directions for Networked Sensors”, ASPLOS 2000, Cambridge, November 2000. http://www.tinyos.net/papers/tos.pdf

[9] Crossbow Technology Inc.: “MTS/MDA Sensor and Data Acquisition Board User’s Manual”, 7430-0020-03, Rev. B, April 2005. http://www.xbow.com/Support/Support_pdf_files/MTS-MDA_Series_Users_Manual.pdf

[10] Freescale Semiconductor Inc.: “Developer’s Serial Bootloader for M68HC08 and HCS08 MCUs”, AN2295D, Rev. 6, 11/2004. http://www.freescale.com/files/microcontrollers/doc/app_note/AN2295.pdf

[11] Pierre-Yves Gilliéron, Daniela Büchel, Ivan Spassov and Bertrand Merminod: “Indoor Navigation Performance Analysis”, Proceedings of the 8th European Navigation Conference GNSS 2004, 17-19 May, Rotterdam, The Netherlands

[12] Cliff Randell, Chris Djiallis and Henk Muller: “Personal Position Measurement Using Dead Reckoning”, Department of Computer Science, University of Bristol.

[13] Y K Thong, M S Woolfson, J A Crowe, B R Hayes-Gill and R E Challis: “Dependence of Inertial Measurements of Distance on Accelerometer Noise”, Measurement Science and Technology. 13(2002) 1163-1172 http://stacks.iop.org/MST/13/1163

[14] J. M. Knudsen and P. G. Hjorth: “Elements of Newtonian Mechanics”, Springer, Second edition, 1996.

[15] Analog Devices: “Low-Cost ± 2 g Dual-Axis Accelerometer with Duty Cycle Output”, Rev. A, 10/2000 http://www.analog.com/UploadedFiles/Data_Sheets/53728567227477ADXL202E_a.pdf

[16] Kone http://www.kone.com/

[17] Kay Römer, Philipp Blum and Lennart Meier: “Time Synchronization and Calibration in Wireless Sensor Networks”,

[18] Bharath Sundararaman, Ugo Buy and Ajay D. Kshemkalyani: “Clock Synchronization for Wireless Sensor Networks: A Survey”, March 22, 2005

[19] Hui Dai and Richard Han: “Tsync: A Lightweight Bidirectional Time Synchronization Service for Wireless Sensor Networks”, ACM SIGMOBILE Mobile Computing and Communications Review, 8(1):125-139, January 2004.

[20] Jeremy Elson, Lewis Girod and Deborah Estrin: “Fine-grained Network Time Synchronization using Reference Broadcasts”, In *Fifth Symposium on Operating Systems Design and Implementation* (OSDI 2002), December 2002.

[21] Joseph Y. Halpern and Ichiro Suzuki: “Clock Synchronization and the power of Broadcasting”, Distributed Computing, 5(2):73-82, 1991.

[22] F. Christian: “Probabilistic clock synchronization”, Distributed Computing, Vol. 3, pp. 146-58, 1989

[23] H. Garcia-Molina: “Elections in Distributed Computer Systems”, IEEE Transactions on Computers, Vol. C-31, No. 1, pp. 48-59, 1982

A Application Source Code

http://www.distlab.dk/sensornet/sentientsports/datacollection.zip

B Acceleration Data Files

Calibration Data

http://www.distlab.dk/sensornet/sentientsports/calibrationdata.zip

Drift Data

http://www.distlab.dk/sensornet/sentientsports/driftdata.zip

Low-Power Data

http://www.distlab.dk/sensornet/sentientsports/lowpowerdata.zip

Walk Data

http://www.distlab.dk/sensornet/sentientsports/walkdata.zip

Elevator Data

http://www.distlab.dk/sensornet/sentientsports/elevator.zip

C Synchronization Data Files

RTT Data

http://www.distlab.dk/sensornet/sentientsports/rttdata.zip

Offset Data

http://www.distlab.dk/sensornet/sentientsports/offsetdata.zip

Agreement Data

http://www.distlab.dk/sensornet/sentientsports/agreementdata.zip