OFFICE MONITORING WITH SENSOR NETWORKS

Projektarbeit
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Samuel Bissig
April 2006

Leiter der Arbeit:
Professor Dr. Torsten Braun
Institut für Informatik und angewandte Mathematik

Abstract

In this report of a computer science project we present an implementation of an office monitoring application using a Wireless Sensor Network. Thereby, the application is not only able to detect the presence of persons in the monitored room, it is also able to determine if the person legally working in the room or is unauthorized to be there.

Each sensor node in the sensor network is equipped with three different kinds of sensors: movement sensor, microphone and a vibration sensor. The nodes are sensing the intensities of an event and sending these values over a radio connection to the master node. There the present threat gets computed using Fuzzy Logic. If necessary an alert is thrown. The application was implemented on real hardware and tested with different realistic scenarios.

Additionally we describe a case-study about an application of an intensity-based object localization scheme in a real wireless sensor network.

Contents

Chapter 1

Introduction

1.1 Project description

Goal of this computer science project was to observe an office with a set of Embedded Sensor Boards (ESB) [1]. Thereby, one requirement of the monitoring application was to be able to detect a person in an office. Furthermore the application should be able to decide if the person is there legally, e.g. a student working at his place, or illegally, e.g. an unauthorized person entering the office which wants to steal something. Multiple ESB nodes are spread in a room and collect data with their sensors. The sensor nodes are stationary and placed in a predefined arrangement. On the ESBs simple filter and aggregation techniques are implemented. The collected and filtered data are sent to a sink node which evaluates them and decides if an alert should be thrown or not. For the evaluation Fuzzy Logic concepts are used. The sensor network uses static routing with fixed paths.

A motivation for this project was also to figure out the abilities of the ESB nodes in real applications.

1.2 Wireless Sensor Networks

In this section we give a short introduction to Wireless Sensor Networks. At the beginning, Wireless Sensor Networks (WSN) were mainly developed for military applications. A number of prominent research projects like Smart Dust [2] or Nest [3] are commonly regarded as the origin of sensor network research. This let to a de facto definition of WSN which described large scale (thousand of nodes, spread over wide geographical areas), wireless, ad hoc, multi-hop and unpartitioned networks. The nodes of these networks were homogeneous and tiny. Later on when also applications for civilian domains began to evolve the network attributes changed what was necessary to deal with specific constraints introduced by the new applications like environmental monitoring, production delivery and health-care. New features included the support of heterogeneity, mobility, simpler topologies and made use of existing communication infrastructures.

Roemer and Friedmann proposed dimensions of WSN in [4] with different classes:

- **Deployment**: random vs. manual; one-time vs. iterative

- **Mobility**: immobile vs. partly vs. all; occasional vs. continuous; active vs. passive
- **Cost, Size, Resources and Energy**: brick vs. matchbox; grain vs. dust
- **Heterogeneity**: homogeneous vs. heterogeneous
- **Communication Modality**: radio vs. light vs. inductive vs. capacitive vs. sound.
- **Infrastructure**: infrastructure vs. ad hoc
- **Network Topology**: single-hop vs. star vs. networked star vs. tree vs. graph
- **Coverage**: sparse vs. dense vs. redundant
- **Connectivity**: connected vs. intermitted vs. sporadic
- **Network Size**
- **Lifetime**

Applying these dimensions to our applications it looks as the following: The nodes for the intrusion detection are deployed manually. They are arranged one-time and are immobile. The network is heterogeneous as we use different types of nodes. The ESB nodes communicate over a radio connection, they are communicating direct with the base station. The Coverage with sensors is depending on the sensing range of different sensor types: The coverage of microphone sensors is redundant, the coverage of vibration and movement sensors is dense. As there is always a connection between the nodes and the base station the network is obviously ”connected”.

1.3 Used Hardware and Software

In this section we give an overview over the used hardware and software in our application.

1.3.1 Embedded Sensor Board

The hardware components we have used are ESB sensor nodes from ScatterWeb, which is a spin-off enterprise and collaboration partner of the Freie Universität Berlin. These are small nodes equipped with different sensors. A node is similar in its size to a fist, where the three AA batteries makes about half of the size. These ESB nodes are using a MSP430 CPU, 2kb of RAM and 60kb flash memory [5]. To communicate the ESB is equipped with a low power consuming radio transceiver. Alternatively the infrared interface can also be used for communication. To interact with its closer environment the ESB node has three colored LEDs, a beeper, a reset button, a free programmable button and a power switch.

The ESB node is equipped with the following sensors:

- **Passive Infrared Sensor**: The passive infrared sensor (PIR) allows to detect movements of objects within a radius of $100°$. The maximum measuring distance is, depending on the angle, between 1.5m and 5m [6] [7].

- **Temperature Sensor**: Measures temperatures from -55 °C to +125 °C. The thermometer provides 9-bit temperature readings. Within the temperature sensor comes a Real Time Clock which provides seconds, minutes, hours, day, date of the month, day of the week, month and year. There is also an alarm functionality [8].
- **Microphone**: The ESB boards are equipped with a microphone (low impedance, 1.5 - 10 V, 0.5 mA, S/N ratio 40 dB, 120 dB max). In our tests the microphone was more sensitive for deep noises like hitting on the table or closing the door. Normaly loud speaking people has not been recognized with the actual firmware and configuration. Standard applications of the microphone is noise detection with adjustable thresholds [9].
- **Vibration Sensor**: To monitor tilt or vibrations the ESB contains a vibration sensor [10]. As we have seen in our tests the vibration sensor is not very sensitive. It needs quite a heavy vibration to recognize it.
- **Infrared Sender and Receiver**: There is an infrared light to frequency converter for receiving infrared (IR) signals on the board which responds over an infrared range from 800nm to 1100nm [11].

For wired communication the ESB is equipped with an RS232 serial interface. The JTAG interface is used for real time debugging or to flash the nodes with new firmware [12] [13].

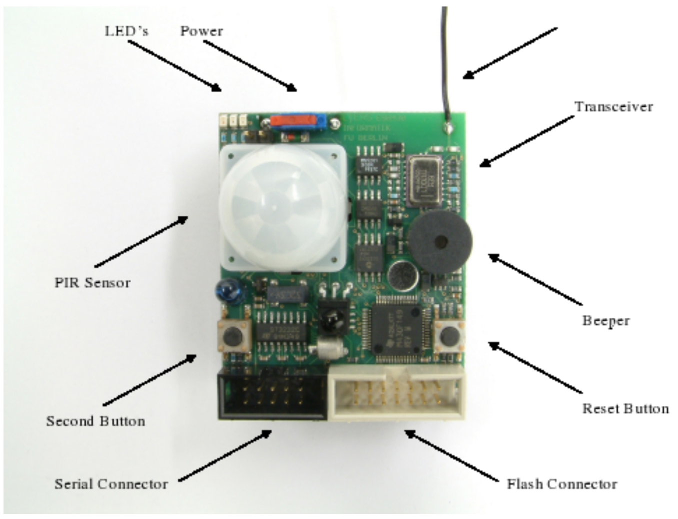


Figure 1.1: The ESB node

1.3.2 Embedded Gate

The eGate sticks (see figure 1.2) are smaller than the ESB nodes but they are not equipped with any kind of sensors. They are mainly used as an interface between an ESB network and personal computers. It exists two kinds of eGate sticks. One with a serial connector (eGate/USB) and the other with an Ethernet Interface (egate/WEB). The USB stick gets its energy over the connector, the egate/WEB stick over a powered Ethernet connection. They are both equipped with a MSP430 CPU [5] and 60kb flash memory. As the ESB nodes they have three different colored LEDs and an interface to flash new firmware onto the stick [12] [13].

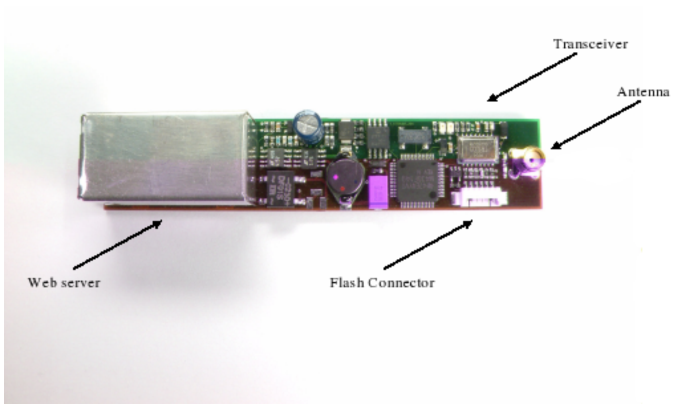


Figure 1.2: The eGate/WEB stick

1.3.3 ScatterWeb 2.2

The software for the nodes, which is written in C, is divided into two parts. One part contains the operating system. It handles interrupts, packet sending and receiving, access to the hardware and offers timer functions. In this part no modifications had to be done in this project. To adapt the firmware to our needs modifications were only done in the applications part. The application part has access to the functions of the operating system. The applications part is freely programmable. The only thing which has to be considered is to register three functions which are called by the operating system to handle external events:

- A radio handler which handles incoming data traffic
- A function which handles periodic tasks
- An event handler, which handles incoming events like a pressed button or a detected event from a sensor

The operating system part of the software needs not to be adapted for the different node types. It gets compiled for the eGates and for the ESB nodes by using macro functions. So it has to be written only once. The applications has to be adapted to the hardware to which it will be flashed on.

With the origin ScatterWeb 2.2 firmware most sensors only return binary values. For example the vibration sensor just gives the information: ”Vibration here” or ”No vibration here”. Also the PIR sensor returns only binary values. The temperature sensor, microphone and the infrared receiver are able to return not only binary but discrete values. Unfortunately the light sensor software, which would have made use of the infrared sensor to measure the light intensity, has been removed due to complex handling in the ScatterWeb 2.2 firmware [13].

ScatterWeb 2.2 comes with a lot of example applications. We adopted the Scan-Net application which is used to communicate with the ScatterViewer. It will be described in the next section.

1.3.4 ScatterViewer

ScatterViewer is a Java Software which can be used to log events and manage the ScatterWeb network [14]. You can use it either in combination with the eGate/USB or the eGate/WEB. We integrated our part of the evaluation and logging software in ScatterViewer.

1.4 Fuzzy Logic

To evaluate the measured data we have used Fuzzy Logic algorithms. We give a short introduction into the topic here.

The Fuzzy-Set-Theory was initiated in 1965. It gives you the possibility to deal not only with binary expression like yes or no, true or false or 0 or 1. You can also compute expressions with fuzzy values like a bit, pretty or heavy. This gives you more flexibility to handle the data which the sensor boards have measured.

Fuzzy Sets

In classical sets, an element can either be part of a set or not. Fuzzy Sets allows elements to be ”a little bit part of a set”. To describe the grade of the membership of an element to a set, Fuzzy Functions are used.

Fuzzy Functions

If we want do define a Fuzzy Set we need Fuzzy Functions. The Fuzzy Functions gives a gradual membership y of a value x to a Fuzzy Set. Here a short example: We want to express pleasant working temperatures in an office. We define a Fuzzy Function μ_t which you can see in figure 1.3. If it is colder then 15°C or hotter then 30°C the temperatures are not pleasant. From 15°to 20°, the temperatures getting more pleasant, from 25°to 30°less pleasant. In between 20°C and 25°C the temperatures are optimal.

On the other side in classical sets we could only define a characteristic function $f(x)$ which assigns a number y either 1 if it is an element of a set or 0 if is not. $f(x)$ defines a classical set which can be seen in figure 1.4.

Like on normal sets there exists operations on Fuzzy Sets like intersection, unify and negation. We have used the minimum operator for the intersection and the maximum operator for the union. Negations are expressed like $\neg\mu_t = 1 - \mu_t$.

Fuzzy expressions like warm, fast or old are identified as Linguistic Variables, which are described by Fuzzy Sets. They can be used in linguistic rules which have two parts: an an-

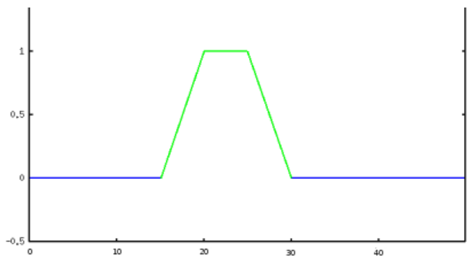


Figure 1.3: Fuzzy membership function

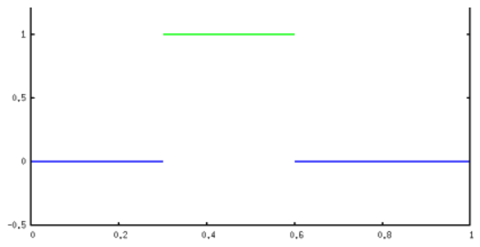


Figure 1.4: Characteristic membership function of a boolean set

tecedent block (between IF and THEN) and a consequent block (after THEN). Such a linguistic rule looks like: *IF temperature is warm AND heaven is blue THEN weather is nice* [15].

Chapter 2

Measurement and Evaluation

2.1 Measurement on Sensorboards

2.1.1 ESBs

As mentioned in section 1.3.3 movement and vibration are measured binary by default. This causes a problem: It does not make sense to use Fuzzy Logic algorithms with binary input values. Further more we needed more precise information to decide if a person is legally or illegally in a room. To get not only binary values we developed a mechanism which is able to return the intensity of events. We defined the intensity of an event by the number of changes of the state during a certain time span. The sensors are sampled a certain number of times during a predefined interval. The final intensity is the sum of the sensed events during the interval. A concrete example: The movement sensor senses eigtht times during a second. Each time it senses movement it increases a counter. After one second the counter states the movement intensity. Using the PIR and vibration sensor, this solution let to reasonable results.

The microphone writes a voltage value into the memory which represents the actual sensed noise. We would have been able to read each time this value from the memory. We tested this method but the values we get were not explainable because the microphone voltage value was noisy, it is impossible to filter out the microphone intensity. The microphone was not designed to measure sound intensities by reading the different microphone voltages. So we decided to use the mentioned method, which we have used for sensing movement and vibration for measuring noise intensities.

We explain now the implementation of the measuring algorithm on the ESB nodes. First the sensors on the ESB nodes have to be initialized. Implemented in C this looks as the following:

```
if(AppConfig.sensorMask & SENSOR_VIBRATION)
                Data_vibrationOn(1000/modVal);
if(AppConfig.sensorMask & SENSOR_MOVEMENT)
                Data_movementOn(1000/modVal);
if(AppConfig.sensorMask & SENSOR_MICROPHONE)
                Data_microphoneOn(1000/modVal, 470);
```

Listing 2.1: Initialization of ESB nodes

The *modVal* variable defines how many slots there are to remeasure during each interval. An interval is during one second. Tests have shown, that the *modVal* variable should not be set higher than eight, which means maximal one sample each 125 milliseconds. By using higher sampling rates the sensed values are getting less reliable because of the rather long sensor trigger times. The function which switchs the microphone sensor on (see listing 2.1 line 6) uses a second parameter: As mentioned the microphone value is read as an absolute voltage value. If the difference between the predefined voltage level and the voltage level which was written in the memory is higher than the second parameter, the event is recognized as noise. In other words the second parameter is the threshold which has to be reached to recognize noise. The microphone voltage level dithers. That is why we have set the second parameter to 470. If it was set smaller some nodes sensed noises in a completely silent environment.

If an activity is sensed by a sensor the *handleEvent* function gets called by the system. The event gets forwarded to the *updateFuzzyValues* function listed in 2.2. Here some flags are set: The *movOn* flag if movement was sensed, *micOn* if noise was sensed or *vibOn* if vibrations were sensed during the last slot.

```
void updateFuzzyValues(sdata_t* data) {
        if(data->sensor == SENSOR_MOVEMENT) {
                if (data->value) {
                        movOn = 1;
                } else {
                        movOn = 0;
                }
        }
        if (data->sensor == SENSOR_MICROPHONE) {
                if (data->value) {
                        micOn = 1;
                } else {
                        micOn = 0;
                }
        }
        if (data->sensor == SENSOR_VIBRATION) {
                if (data->value) {
                        vibOn = 1;
                } else {
                        vibOn = 0;
                }
        }
}
```

Listing 2.2: Updating sensor flags

The *updateLoopState* function listed in 2.3 checks these flags after each slot. If a sensor flag was set the sensor intensity counter gets increased by one. If it is the last slot of an interval the nodes sends its intensities with a time stamp to the master node if there has been any sensed activities.

Finally the intensities are reset. The highest possible intensity is equal to *modval*, the lowest is 0. The function *updateLoopState* gets called by a timer, which is set at the end of each call.

```
void updateLoopState() {
        if (loopState < modVal) {
                loopState++;
                if (vibOn) {
                        vibIntensity++;
                }
                if (movOn) {
                        movIntensity++;
                }
                if (micOn) {
                        micIntensity++;
                }
        } else {
                if (vibIntensity > 0
                    || movIntensity > 0
                    || micIntensity > 0) sendFuzzyValues();
                loopState = 0;
                vibIntensity = 0;
                movIntensity = 0;
                micIntensity = 0;
        }
        Timers_add(1000/modVal + 1, updateLoopState, 0xFFFF);
}
```

Listing 2.3: Updating sensor flags

The sensor nodes use a flag which describes its node type. We have defined the following node types:

- **Door node**: These nodes are fixed at the door.
- **Window node**: These nodes are fixed at the window.
- **Normal nodes**: These nodes are spread anywhere else in the room.

All node types works the same but their sensed values are handled different by the evaluation software. The type flag can be switched by pressing the programmable button on the ESB nodes. By default the nodes are ”Normal nodes”, pushing once makes them to ”Door nodes”, twice ”Window nodes”.

2.1.2 eGates

For the eGate sticks only a few modifications were necessarily because the nodes only have to format and print out the received data. The evaluation of the data could also be done here. We decided to do this on a computer for a first implementation, which will be described in the next section.

2.2 Evaluation of Measured Data

The evaluation of the sensed data is implemented in the ScatterViewer software which is written in Java. We decided to use the ScatterViewer software because it offers already features we needed: The communication between the eGate stick and the software is already implemented. Furthermore we had not to care about the network initialization, which is also implemented. By starting the ScatterViewer all reachable nodes are searched by broadcasting scan messages. Sensors which want to join later the network are recognized dynamically. The Scan-Net application on the nodes sends scan messages when switching on the nodes. In the initialization phase we did a modification: The nodes not only answer the Scan packages, they also send their node type which is then saved in the data structure of ScatterViewer. We also added a functionality to prevent time drift: The ScatterViewer synchronize the time on the nodes every 60 seconds by sending its own system time to the nodes.

The ScatterViewer filters the sensed sensor intensities from other information which is sent by the ESB nodes and writes it into a database. Each time new values come in, the actual threat gets checked.

To evaluate the sensed data with Fuzzy Logic we have used the "Fuzzy engine for Java 0.1a" [16]. This engine allowed creating Linguistic Variables and Fuzzy Functions which can be seen in listing 2.4.

```
this.stdVIB = new LinguisticVariable("vibration");
this.stdVIB.add("no", -1, 0, 0, 1);
this.stdVIB.add("few", 0, 1, 1, 2);
this.stdVIB.add("heavy", 1.5, 2.5, 8, 8);
```

Listing 2.4: Initialization of a Linguistic Variable

First a Linguistic Variable is initialized with its name as parameter. Then membership functions are added to the Linguistic Variable. The first parameter of *add()* is the name of the membership function. The second gives the minimum (0) of the membership function, the third and fourth the beginning and end of the maximum (1), the fifth gives again the minimum (0). This construct allows to build membership functions represented by trapezoids, triangles or rectangles. To operate with the Linguistic Variables Fuzzy Functions are created (see listing 2.5).

```
rulesArray[0] =
        "if␣" + this.stdMIC.getLVName() + "␣is␣no␣and␣" +
        this.stdMOV.getLVName() + "␣is␣no␣and␣" +
        this.stdVIB.getLVName() + "␣is␣no␣then␣" +
        this.lingVar.getLVName() + "␣is␣empty";
```

Listing 2.5: Example for a Fuzzy Function

The Linguistic Variables and the Fuzzy Functions are registered in the Fuzzy engine. The sensed data is set as input. The Fuzzy Functions are evaluated as described in section 1.4: The *OR* operates as max operator, the *AND* as min operator. The result is assigned to the part after *THEN*.

We have defined four Linguistic Variables where the first three are based on the sensors movement, microphone and vibration. The fourth Linguistic Variable is defined by the system time. The four resulting variables with their membership functions are: **time** (*officeHours, night, earlyMorning, lateMorning*), **microphone** (*no, few, heavy*), **vibration** (*no, few, heavy*) and **movement** (*no, few, heavy*). Based on these four we have defined more specific Linguistic Variables. This architecture allows more flexibility: If for example the *modVal* variable on the sensor nodes has changed, the sensed intensities are in a other range. We only have to modify the membership functions of the four basic Linguistic Variables. The additional Linguistic Variables are:

- **kindOfMovement**: Describes if a person is *walking, sitting* or *silent*. **kindOfMovement** is based on **movement**.
- **distribution**: Checks if a person is walking through the whole room or only in parts of the room. Possible states are *locally, partly* and *everywhere*. This linguistic variable is based on **kindOfMovement**.
- **Room**: Checks if the room has been empty during a predefined time span. Only two states are possible: *empty* or *notEmpty*. **Room** is based on **vibration, movement** and **microphone**.
- **SuspiciousEvents**: Checks if at a certain moment have been suspicious events, for example a lot of noise and vibrations. Three different levels are possible: *notsuspicious, suspicious* and *verysuspicious*. It is based on **vibration** and **microphone**
- **TimeRisk**: We split the time of a day into *norisk, mediumrisk* and *highrisk*. This express the risk of a burglary depending on time. **TimeRisk** is only based on **time**.
- **enteringWindow**: Tells us if someone tries to enter through a window when the office is empty. Either we have vibration at a window or we do not have vibrations at a window (*dangerous* or *noDanger*). For this decision we made use of the Linguistic Variable **vibration**.

Each time an activity gets sensed and sent to the eGate stick, the risk of an intrusion gets checked by the *AlertChecker* class which makes use of the above listed Linguistic Variables. The precondition to detect an intrusion is that the room has been empty longer than a predefined time span, Room Empty Time (RET). If this precondition fails we suppose a person to work in the room, so no intrusion detection is necessarily.

The first rule which is checked is if somebody enters through the window while the room is empty (see listing 2.6). If somebody opens the window commonly, there must have been events in the room before, which means that no alert will be thrown.

```
if  (this.roomEmpty.getStatus() == RoomEmpty.EMTPY
        && this.windowVibrations.getStatus() ==
        WindowVibrations.WINDOWMOVING) {
                this.alertEvents.add(
                new AlertEvent(
                        new Date().getTime() / 1000,
                        ”Alert:␣Window␣opened”,
                        AlertEvent.ENTERINGWINDOW)
                );
}
```

Listing 2.6: Window alert

To watch how a person behaves after having entered the room, the system waits a predefined time, the Intrusion Watching Time (IWT) and analyzes the sensed data after the expiraton of that time. To recognize a burglar during the day two conditions have to be fulfilled: First most of the nodes have to sense a maximal movement intensity during the IWT. Maximal intensity means that no other node has sensed a higher value at the same time. Furthermore the entered person has to behave suspicious. If those conditions are fulfilled an alert is thrown (see listing 2.7).

```
if      (this.distributedWalking.getStatus() ==
        DistributedWalking.EVERYWHERE &&
        supiciousStatus == SuspicousEvents.VERYSUSPICOUS) {
                this.alertEvents.add(
                        new AlertEvent(
                        new Date().getTime()/1000,
                        ”Alert:␣burglar␣in␣the␣room.␣”,
                        AlertEvent.ROOMALERT)
                );
}
```

Listing 2.7: Burglar in the room during the day

If it is night, the first condition is not necessarily. So the person has only to behave suspicious. There needs to be some noise and vibrations during the IWT to throw an alert.

For testing purpose we added to the ScatterViewer a functionality, which allows logging the data during a certain time span in an external file. The thrown alerts are logged there, too.

Chapter 3

Evaluation of Intrusion Detection

To test how the application performs in different cases we designed different test cases. It was not possible to cover all different patterns how burglars behaves. On the other side we had to be sure that no alert is thrown by legally working people. In four different test cases we covered some of the possible scenarios. In one test case no alert should be thrown, there is a student working in the room. In the three others a burglar was simulated. The four test cases will be described in detail below. Each test case was carried out ten times. The subject tried to behave the same in each run. All tests were done by the same person.

All four test cases have been carried out in the same room. The room had a size of 5.5m x 10m. The tests have been done by daylight. Figure 3.1 shows the experiment set-up. We have used 5 ESB sensor nodes (green quarters with identifier numbers). The node with the identifier number 30 only senses vibrations. On that node the window vibration flag was set (see section 2.1.1), the node was fixed on the window frame (W). The red triangles shows the directions in witch the PIR sensors are pointing to. The sensors 20 and 60 are placed on the borders of the tables (T), node 40 is placed on an advance on the table next to the working place (P) and sensor 10 is on a cabinet (C) which is higher then the tables. On the bottom of the diagram you can see the door (D).

To save time for testing the RET was set to 20 seconds. So we had not to wait too long until the room was identified to be empty by the application. The RET should be set higher in real applications. The IWT was set to 25 seconds. This value has to be adjusted to the size of the room to warrant the entering person has the possibility to walk through the whole room before the IWT past.

To visualize the logging data we made three types of figures:

- A figure which shows the data over the whole network and all sensor types. If events were sensed at different nodes by the same kind of sensors at the same time the maximum of these values is shown (e.g. figure 3.2 (a)).

- Figures which show the movement intensity by single nodes. We show this figure because we can thus determine where a person is located in the room. With the measured microphone values it is not possible to localize an event in such a small room (e.g. figure 3.3 (b)).

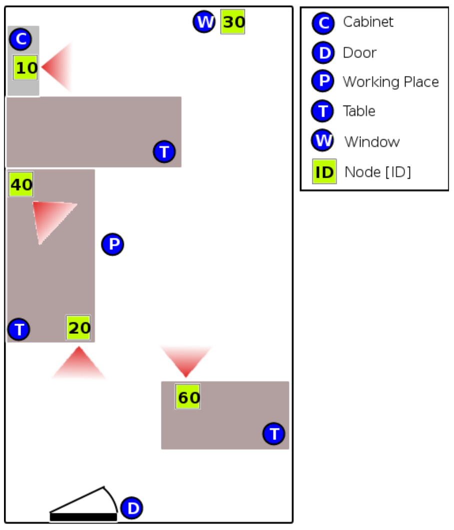


Figure 3.1: Experiment set-up

- Sometimes it is important to see which node has sensed vibration activities. In some test cases we show the figure which shows vibration intensities measured by each node (e.g. figure 3.6 (b)).

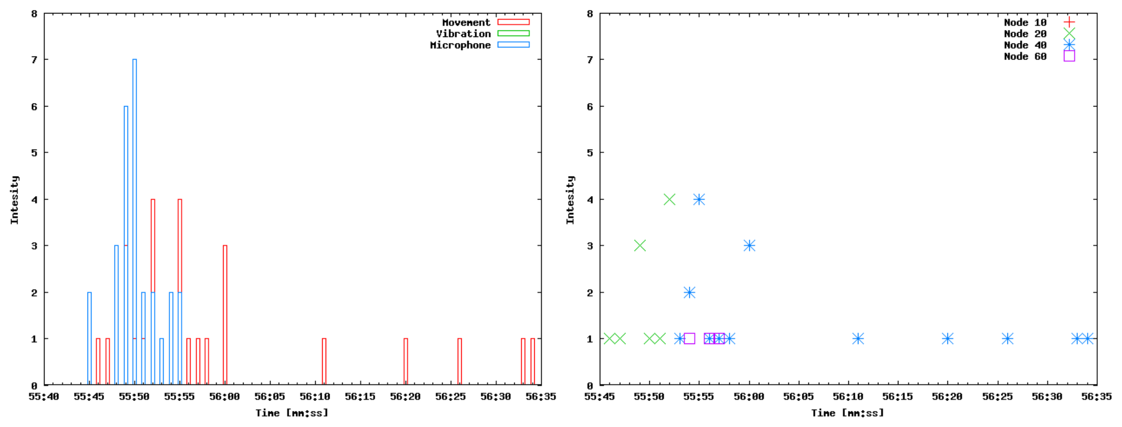
The red lines in the figures represent an alarm. The vertical axis shows the sensed intensities, the horizontal lines the time.

3.1 Case 1 - Student is working

The room has been longer empty than the RET, which means that the system is ready for sensing intrusions. The door (D) is locked. A student opens the door with a key and enters the room. He walks without generating much noise to the working place (P) and starts working on his computer. Sometimes there is a noise, the subject is speaking or listening to music. The test is successful if no alert was thrown.

3.1.1 Results and Discussion

All ten runs were successful. A characteristic run is shown in figure 3.2 (a). It shows the intensities of the measured values over all nodes and sensors. At the beginning of the run microphone activities are sensed. This is caused by unlocking the door (at 55:45). After that, the door gets opened (at 55:46 and 55:47). The student closes the door and walks through the room to his working place. Between 55:52 and 55:55 he sits down which causes movement and microphone activities. After that he works on his computer (P) which causes some movement peaks by node 40. That can be seen in figure 3.2 (b). We recognized that no vibration has been sensed during the whole run, which is one of the two reasons that no alert has been thrown. Furthermore only the two sensors 20 and 40 had movement maximums which is not enough to throw an alert (see figure 3.2 (b)).



(a) All sensors (b) Movement intensity

Figure 3.2: Test case 1

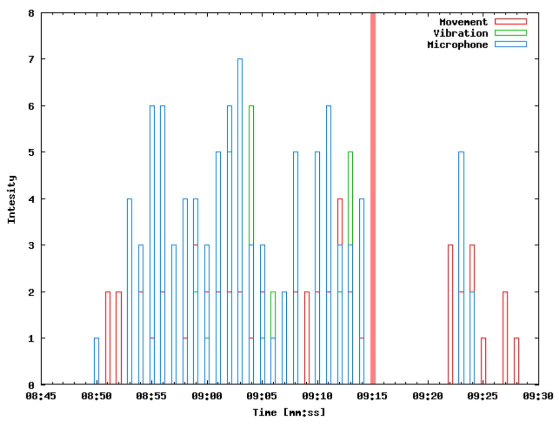
3.2 Case 2 - Burglar in the room

In this test case the room was again longer empty than the RET. The door (D) was not locked. A person opens the door and enters during the office hours. Then the subject starts searching something everywhere in the room. There are some noisy activities. The person tries to open a drawer at the cabinet (C) next to sensor 10 or is searching something on the tables (T). The test is successful if an alarm has been thrown.

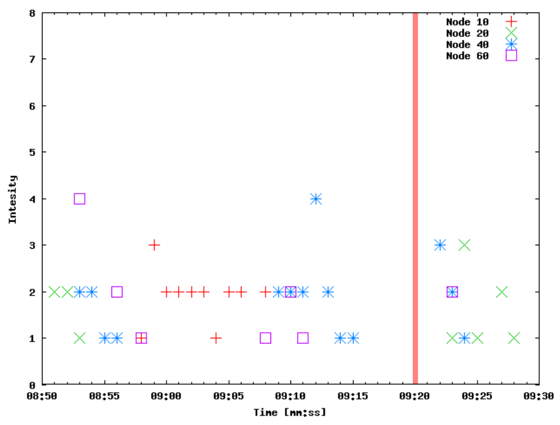
3.2.1 Results and Discussion

Nine of ten runs were successful. The successful run number five is shown in the figures 3.3 (a) and 3.3 (b). There are a lot of microphone activities. The person walks through the whole room, which can be seen by the maximums of each node during the IWT (see figure 3.3 (b)). The vibration between 08:59 and 09:06 are sensed by node 10, which is standing on the cabinet (C). Node 40 has sensed vibrations a few seconds later. The alarm was thrown at 09:15, 25 seconds after the first event was sensed which is equivalent to the IWT.

Run number four failed. Between 00:05 and 00:10 microphone activities are registered (see figure 3.4 (a)). At that time the room was empty, the values must be from a very loud noise outside or from the upper floor. But this is not the reason for the failure because the person entered at 00:40, more then 20 seconds (=RET) after this event. So the application identified the room to be empty again when the subject entered. The reason for the failure was that there have not been enough suspicious movements, so no alarm was thrown.

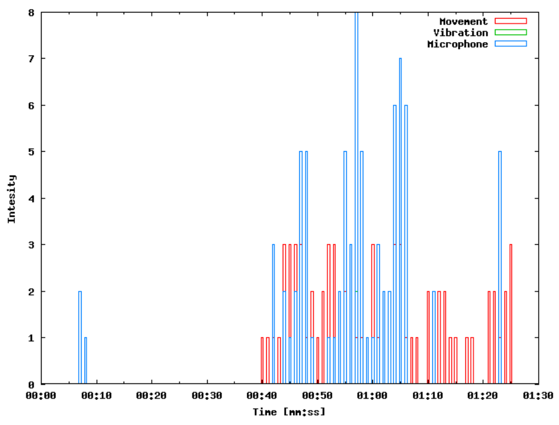


(a) All sensors

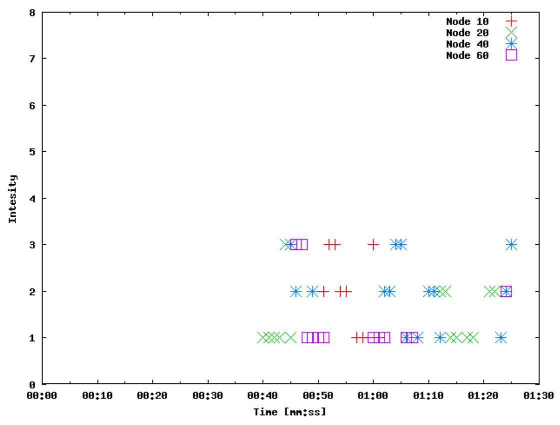


(b) Movement intensity

Figure 3.3: Test case 2



(a) All sensors



(b) Movement intensity

Figure 3.4: Failed test case 2

3.3 Case 3 - Burglar during the night

As in test case 1 and 2, the room was longer empty than the RET. The door is locked. A burglar opens the door (D) and enters the room during the night. He generates some noise and / or vibrations by moving things around or searching something on the tables (T) but he does not walk through the whole room as in test case 2. The test is successful if an alert has been thrown.

3.3.1 Results and Discussion

Seven out of ten runs were successful. A characteristic run is run number three, which is shown in figure 3.5. There are lots of microphone activities. None of the nodes has measured vibrations. The reason therefore is that none of the nodes has been close enough to a vibration source. The high intensities of the microphone sensors are an evidence that there have been lots of activities like moving a table or opening the cabinet. The alarm was thrown at 12:30.

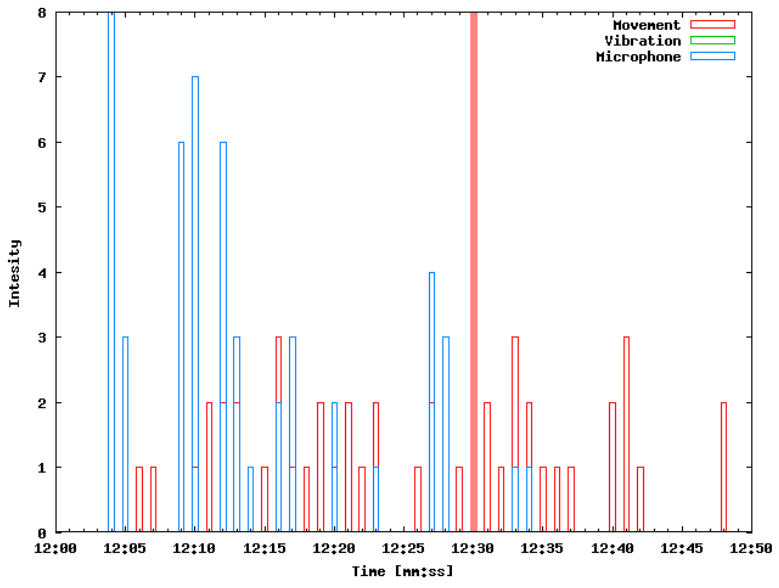


Figure 3.5: Test case 3, all sensors

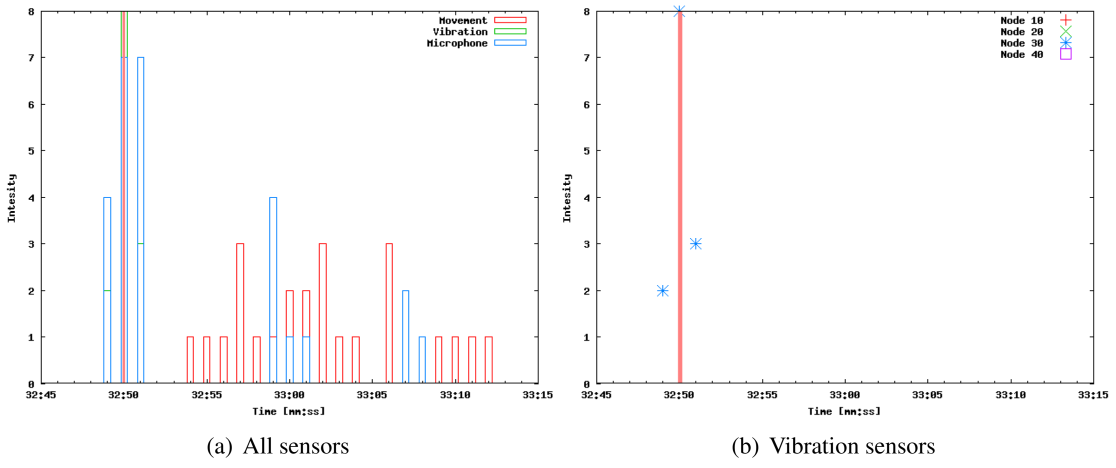
The reason for the tree runs which have failed are that the activities of the person were to silent and there were not enough vibrations sensed. To improve the system we could make the system more sensitive during the night. So an alert is already thrown with less noise or vibrations activities.

3.4 Case 4 - Burglar entering through the window

In test case 4 the room also needs to be empty, there was no activity in the room before. In this run a burglar entering through the window should be simulated. Of course we have not broken ten windows. We tried to simulate this by banging against the window, which caused vibrations. After that the burglar enters through the window and searches something in the room.

3.4.1 Results and Discussion

Nine of the ten runs have been successful. Run eighth is shown in the figure 3.6 a and b. At 33:49 the first vibration was sensed. There was no activity in the room before. This vibration was measured by node 30, which is the special window node. After that an alert is thrown (32:50). As we can see there are two more vibration events caused by opening the window, after which the person enters through the window and walks through the room.



(a) All sensors (b) Vibration sensors

Figure 3.6: Test case 4

The problem with the failed run was that there was a failure in the logging software, there was no data in the log file.

3.5 Conclusion

Under the bounded circumstances of the experiment set-ups 90% of the runs were successful (see figure 3.7). In case 1 where a student was working all runs were successful. The worst test case was case 3 where a burglar during the night was simulated. Here only seven out of ten cases were successful. In case 2 and 4 only one run failed.

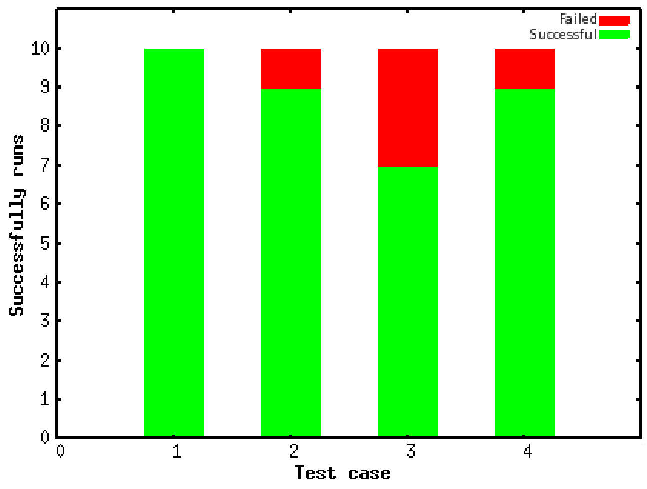


Figure 3.7: Overview over all test cases

Testing the application by a person which does not know anything about the implementation and where the sensor nodes are placed would make the tests more realistic. Also if the testing person tried to be as neutral and objective as possible, it nevertheless had some knowledge about the implantation, which could falsify the result.

Chapter 4

Evaluation of Object Localization

An application to sense the intensity of movement, vibration and noise (section 2.1.1) which emanated from this project was investigated in our previous work [17]. To give you a further insight about the potential and the boundaries of the intensity based object localiation and object tracking we give a short overview over this work in the following.

The goal in [17] was to prove the applicability of the intensity-based localization scheme, which has been published in [18], on real existing hardware. The intensity-based localization algorithm theoretically enables to localize an event solely based on the knowledge of the signal intensities measured on sensor nodes. To prove this in practice a net of ESB nodes (section 1.3) was used to do different real life experiments.

In the first experiment eighth ESB nodes were placed on the boarder of a quadratic area with a dimension of 9 square meters. The PIR sensors were aligned to the center of the quad. A single hand was moving at three different wholes in the middle of the square during 20 seconds. The three position are marked with the large symbols *0,* + and *X* in figure 4.1. The test was done eight times for each whole. The intensity was determined as the average of the intensities measured in one run. The result has shown that there is a spatial correlation of the real location of the hand and the estimated positions of the hand given. The small symbols *0,* + and *x* in figure 4.1 show the barycenters of the location estimations of each experiment.

In a second experiment a five meters wide corridor with five sensor nodes on each side was built. A subject, which was moving with a constant moving pattern at the same place, on two different points in the middle of the corridor was tried to be localized. The two points are marked with the two large symbols 0 and X in figure 4.2. There where done eight runs, 20 seconds each. The average of the intensities measured during one run was used as the overall average for that run. In this experiment a correlation between the real location of the person and the estimated position existed, too. The small symbols *0 and x* in figure 4.2 shows the location estimations of each run.

In a third experiment the person was moving along a line in a four meters wide corridor. A schematic experiment set-up and the results can be seen in figure 4.3. The person walked along the middle of the corridor, represented as a solid line in figure 4.3, with a constant speed. The experiment has been repeated eight times and the average of the intensities determined in the according intervals of the individual runs were taken. Figure 4.3 shows that the estimations along the path are concentrated to the horizontal center of the corridor. This is caused by the

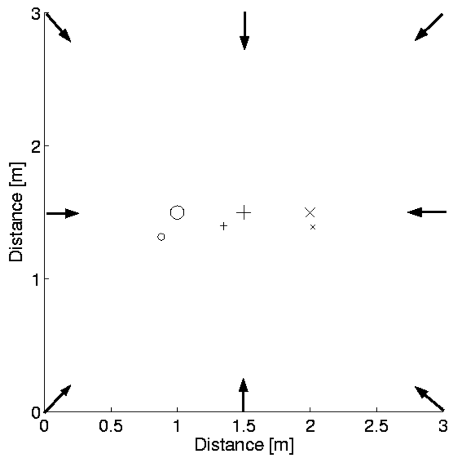


Figure 4.1: Localization of a human hand at three different places

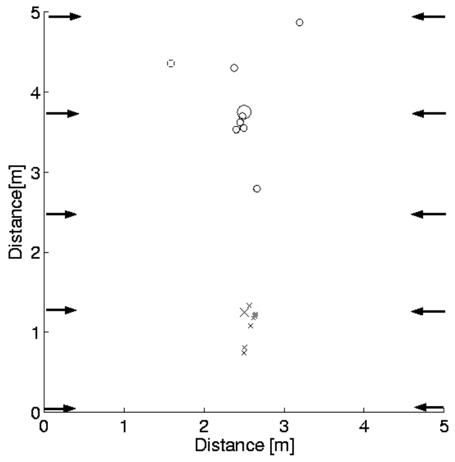


Figure 4.2: Location estimation of a person constantly moving at two places

fact that there are no more nodes placed further than the left and right boarders. So less sensors sense movements at the boarders then in the middle.

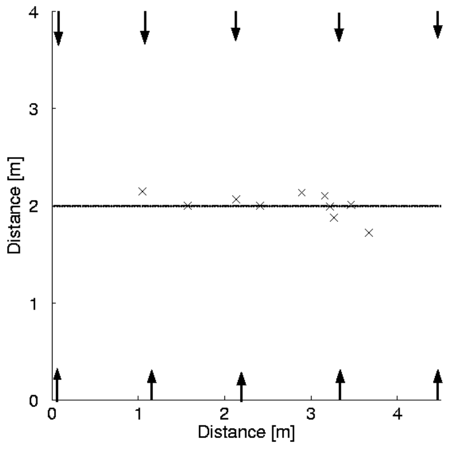


Figure 4.3: Tracking a person moving through a corridor

In [17] was shown, that a intensity-based localization on existing sensor boards is possible under certain restrictions. The sensing limitations of the current sensor hardware were identified. It was shown, that a spatial relation between an event position estimation and the real location of the event exists, an intensity-based localization with more appropriate hardware would be promising.

Chapter 5

Conclusion

The goal to observe an office with a set of Embedded Sensor Boards (ESB) was reached. Detecting if a person is in a room or not was not so complex and it was implementable without problems. To detect if a person with mean well is in a room was more challenging but under bounded circumstances also possible as you can see in the four test cases (see figure 3.7).

One of the following situations is necessarily that the applications detects an intrusion (see also section 2.2):

- The room has to be empty for a while before the person enters the room. The entering person has to move suspicious which means that there needs to be noise and / or vibrations if the subject enters during the night.
- During the day the person has to behave suspicious like before and additionally he has to be looking for something in the room, which means that he has to walk through the whole room. Also here the room has to be empty for a while before the subject enters.
- The person has to enter through the window in an empty room.

In all three situations the room has to be empty to detect an intrusion. If it is not empty we assume that an authorized person is in the room and no intrusion detection is necessary.

Of course there are scenarios we can not identify by checking these circumstances. For example a burglar which enters silent and knows exactly where the object is which he wants to steal. A light sensor would be very helpfully in such cases: If there are movements in a dark room it would be a clue that a suspicious person is in the room.

On the other side it is possible to identify an authorized person as a burglar by mistake and throwing a false alarm. We are thinking about room cleaning staff which behaves noisy and walks through the whole room. An other problem are multiple persons entering the office at the same time, which causes more sensor activity as if only one person enters. This case has never been tested. If the sensor nodes would know their own position and the direction in which the PIR sensor is pointing to, it would be possible to detect whether there is only one person or there are multiple persons in the room.

The advantage of such an intrusion detection system over a conventional one is that it works autonomous. You do not have to think about deactivating it when you enter the room or reactivate it when you leave. You can install it fast without much effort. The disadvantage is that you need a more dense network of sensors. It is for example possible to detect if somebody entered the room with a single sensor, but to decide if a person wants to steal something in the room or not, more sensors are needed.

Possible Improvements

To improve the battery life time there could be implemented more filter mechanisms on the sensor boards. If there is some body working in the room it is not necessarily that the sensors sense as often as if the room is empty, because no intrusion detection is necessarily then. Also if the room is empty not all sensors nodes have to sense as often as they do in the actual implementation. We have seen that noisy activities like opening the door are sensed by most sensor nodes in the room. One awake, noise sensing node would be enough. It could wake up the other nodes if he had sensed some activity.

Bibliography

[1] "Platform for self-configuring wireless sensor networks," Freie Universitaet, Institute of Computer System and Telematics. [Online]. Available: http://www.scatterweb.net

[2] "Nest: Network of embedded systems," University of Virginia, Department of Computer Science. [Online]. Available: http://www.cs.virginia.edu/nest

[3] Smart dust - autonomous sensing and communication in a cubic millimeter. University of California at Berkeley, Department of Electrical Engineering and Computer Sciences. [Online]. Available: http://robotics.eecs.berkeley.edu/ pister/SmartDust/

[4] K. Roemer and F. Mattern, "The design space of wireless sensor networks," Institute for Pervasive Computing, ETH Zurich, Tech. Rep., 2004.

[5] *MSP430x1xx Family, User's Guide Extract*, Texas Insturments.

[6] "Fresnel lens," Freie Universitaet, Institute of Computer System and Telematics. [Online]. Available: http://www.inf.fu-berlin.de/inst/ag-tech/scatterweb-net/datasheets/FLINSE.pdf

[7] "Polymetric infrared detector," Freie Universitaet, Institute of Computer System and Telematics. [Online]. Available: http://www.inf.fu-berlin.de/inst/ag-tech/scatterweb-net/datasheets/PIR.pdf

[8] "2-wire digital thermometer and real time clock," Freie Universitaet, Institute of Computer System and Telematics. [Online]. Available: http://www.inf.fu-berlin.de/inst/ag-tech/scatterweb-net/datasheets/DS1629.pdf

[9] "Electret condenser microphone unit," Freie Universitaet, Institute of Computer System and Telematics. [Online]. Available: http://www.inf.fu-berlin.de/inst/ag-tech/scatterweb-net/datasheets/ECM-10.pdf

[10] "Product data sheet - movement / vibration switch cm 1800-1," Freie Universitaet, Institute of Computer System and Telematics. [Online]. Available: http://www.inf.fu-berlin.de/inst/ag-tech/scatterweb-net/datasheets/CM1800-1.pdf

[11] "Tsl245 - infrared light-to-frequency converter," Freie Universitaet, Institute of Computer System and Telematics. [Online]. Available: http://www.inf.fu-berlin.de/inst/ag-tech/scatterweb-net/datasheets/TSL245.pdf

[12] A.liers, H. Ritter, and J. Shiller, “First steps with the scatterweb sensor nodes,” Freie Universitaet, Institute of Computer System and Telematics, Tech. Rep., 2005.

[13] *ScatterWeb - a platform for flexible sensor networks*, Freie Universitaet, Institute of Computer System and Telematics.

[14] *User Guide (for ScatterViewer) - A platform for teaching and prototyping wireless sensor networks*, Freie Universitaet, Institute of Computer System and Telematics.

[15] M. Hellmann, “Fuzzy logic introduction,” Laboratoire Antennes Radar Telecom, 2001.

[16] E. Sazonov, “Open source fuzzy inference engine for java.” [Online]. Available: http://people.clarkson.edu/ esazonov/FuzzyEngine.htm

[17] M. Waelchli, S. Bissig, and T. Braun, “Intensity-based object localization and tracking with wireless sensors,” in *REALWSN’06*, June 2006.

[18] M. Waelchli, M. Scheidegger, and T. Braun, “Intensity-based event localization in wireless sensor networks,” in *WONS’06*, Les menuires, France, Jan. 2006.