Sensing Spaces: Light-Weight Monitoring of Industrial Facilities[1]

Yong DING [a], Friedemann LAUE [b], Hedda R. SCHMIDTKE [a], and Michael BEIGL [a]
[a] *Karlsruhe Institute of Technology (KIT) TecO, Germany,*
E-mail: {yong.ding, hedda.schmidtke, michael.beigl}@kit.edu
[b] *Technische Universität Braunschweig, Germany,*
E-mail: flaue@ibr.cs.tu-bs.de

Abstract. Defects in complex industrial facilities such as engines, power plants, or energy infrastructure are difficult to diagnose, not only because of the complexity of the machinery, but also because of its spatial structure. Large energy infrastructure, for instance, stretches over vast areas, so that human inspection and diagnosis would take several months. Wireless Sensor Networks (WSN) provide a cheap and easy way for online monitoring of environments where human inspection is impossible.

We present a light-weight distributed sensing approach for localizing defects of facilities in industrial monitoring. In contrast to current usage of WSNs in industrial monitoring, we not only collect data on the machine, but also directly process it where it is generated. We illustrate how reasoning can be used for localizing defects on a long conveyor belt. We demonstrate, with an algorithm for finding the maximum vibration measurements among a set of sensor nodes, how local knowledge exchange between neighboring nodes can establish globally valid knowledge on the WSN as a whole. Results from an experiment with an implementation on WSN hardware illustrate that our approach is practically feasible.

Keywords. Distributed Sensing, Localizing Defects, Industrial Monitoring

Introduction

Defects in complex industrial facilities such as engines, power plants, or energy infrastructure are difficult to diagnose, not only because of the complexity of the machinery, but also because of its spatial structure. Engines are constructed so as to occupy as small a volume as possible, so that many parts are usually not accessible from the outside and diagnosis itself involves disassembly of the machine. The space occupied by power plants, in contrast, has areas that are not viable for a human inspector under operation, so that, likewise, diagnosis might require shutdown. Energy infrastructure, in turn, is spread out widely, so that, again, it cannot be inspected directly. Wireless Sensor Networks (WSN) provide a way for monitoring industrial operation where human inspection is impossible

[1]The research reported in this paper has been supported by the German Research Foundation (DFG) in the project SenseCast: Context Prediction for Optimization of Network Parameters in Wireless Sensor Networks (BE4319/1).

or costly and installing standard measurement and network hardware would be infeasible because of financial, local, or technical restrictions.

Many monitoring applications use a centralized architecture, e.g. the application for monitoring the structural health of bridges in [3]. In a centralized architecture, the sensor nodes are only used to collect sensor data. The raw data are sent to a central server where they are processed. The network is only used to connect the sensor nodes with the central server. However, decentralized architectures have two main advantages over centralized architectures:

- *Lower energy consumption and communication overhead.* In the centralized architecture, the sensor nodes send unprocessed data: relevant and irrelevant information can first be separated by the central server. In a decentralized architecture, nodes collaborate and communicate mainly locally. Only information that is globally relevant is communicated globally through the network.
- *Robustness.* Failure of a node leads to local problems, which can be detected and handled locally. In a centralized architecture, any failure of the central server renders the whole system unusable.

In this paper, we illustrate a decentralized architecture for the scenario of conveyor belt monitoring in open-cast mining. We propose an approach to locally process raw data into meaningful events using only value comparison operations, simple reasoning, and local communication.

In the following sections, we introduce the overall scenario (Sect. 1) and its requirements (Sect. 2). We then discuss the underlying idea of distributed sensing and reasoning and how it can be realized on WSN (Sect. 3). Finally, we present results from two experiments using an implementation on sensor node hardware (Sect. 4).

1. Scenario: Conveyor Belt Systems in Open-Cast Mining

In open-cast mining the extraction of coal, gravel and sand without tunneling into the earth, is performed by combination of different production units with downstream transport [6]. The extraction form can be basically divided into continuous and discontinuous. German open-cast mining has accomplished two device combinations for continuous extraction, such as combination of bucket-wheel excavator and spreader. Continuous extraction requires continuous evacuation following as well. Therefore, continuous transport in open-cast mining is only possible via belt conveyor systems that are adapted to the production capacity of the mining device. These conveyor belt systems span large areas, which are expensive to monitor: Figure 1 shows an overview of the German opecast mining Garzweiler[2].

Due to the large dimensions in open-cast mining, the conveyor belt system consists of a large number of conveyor belts of different lengths and conveyor power stations at the interfaces. Figure 2 shows a schematic plan of an example conveyor belt system: coal is transported from the bucket-wheel excavator, distributed by shuttle heads and finally collected through spreaders to a coal bunker. Arrows in the plan stand on the one hand for the positions of conveyor power stations, and on the other hand show the transport direction. Each power station is equipped with three-phase asynchronous motors [5] and

[2] http://de.wikipedia.org/wiki/Tagebau_Garzweiler

Figure 1. Panorama from the open pit Garzweiler.

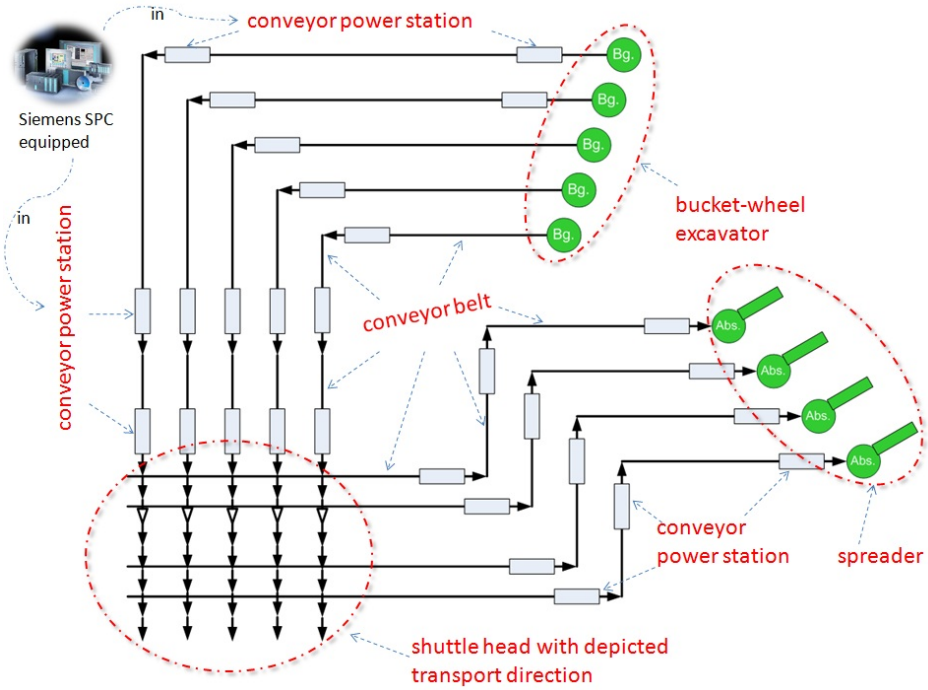


Figure 2. Schematic plan of conveyor belt system.

brakes (electronic or hydraulic). Through the Siemens SPC (stored program control) that is installed in the power station, the conveyor belt is controlled automatically.

The rubber belt moves forward through a series of supporting stands that are inherent parts of scaffolding, see Figure 3. Some types of collisions cannot be avoided in unmonitored, automatic operation, such as collisions caused by lateral off-set of scaffolding or by belt wear. If they remain undetected for a longer duration, such failures can lead to system collapse. When an operator detects such problems, he can repair or, in the emergency that a system collapse is imminent, manually stop the conveyor system with a safety device, the pull cord switch. A monitoring device that could determine, by status

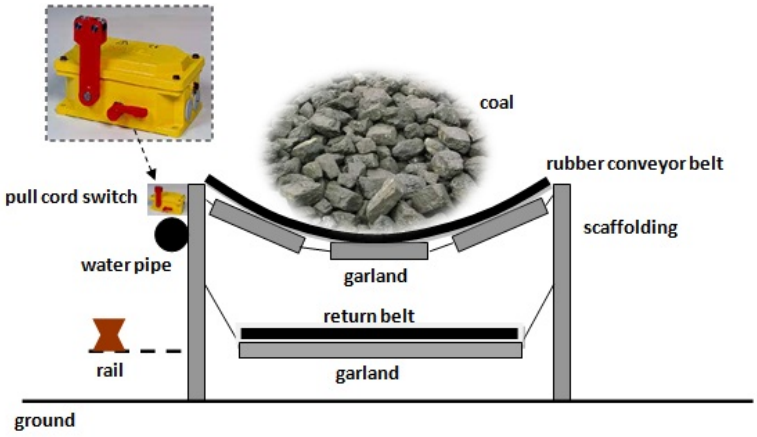


Figure 3. Cross section of the conveyor belt facility.

observation of conveyor belts, when a maintenance action should be taken, could prevent collisions that are a result of belt wear in time, so that high availability of the conveyor system could be guaranteed.

2. Requirements for Online Monitoring of Conveyor Belt Systems

During usual transport operation the conveyor belt is always in vibration, the scaffolding and garlands vibrate as well. This vibration exhibits characteristic patterns of frequency range and magnitudes under specified marginal conditions. Different materials and weather conditions (rain, strong sunshine) have significant influence on wear or damage of the rubber conveyor belt. The consequence of this can be a change in the vibration pattern. Because the worn out or damaged belt part moves forward, the vibration of the scaffolding and garlands is also affected as the damaged part passes by. Consequently, sudden changes in vibration patterns can be used to detect, localize, and track defects.

Vibration sensors attached to the scaffolding, e.g. every five meters, could be used to monitor vibration patterns (see Figure 4). By monitoring the spatial movement of the anomalous vibration we can then locate, in which 5 meters the belt's defect is to be be found. From the increase of magnitude of vibration we can then deduce the degree of wear of the rubber belt. Consequently, actuators can be activated to start one of the following actions:

- For *little wear*, the maintenance schedule should be adjusted.
- For *excessive wear*, an alert should be signaled acoustically and optically.
- For *complete damage*, the pull cord switch should be operated immediately.

In this scenario, monitoring thus is not only used to detect and localize defects, but also to trigger appropriate actions, in order to avoid collisions and ensure operational availability.

As each conveyor belt is a few kilometers long, a wireless sensor network is cheaper than a cable-based solution for sensor data transmission. The sensor nodes deployed

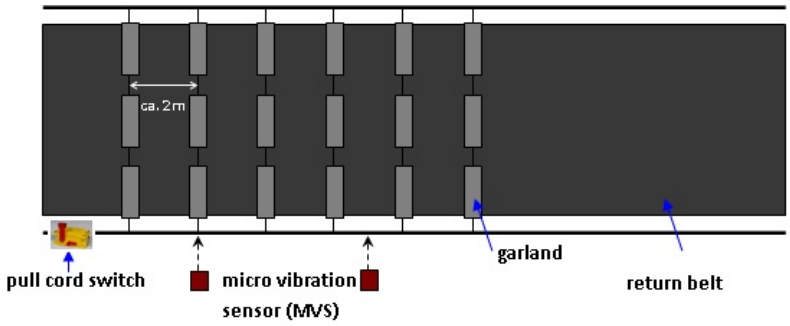


Figure 4. Top down view of the conveyor belt.

in the scenario should be as simple as possible, since a large area has to be covered and powerful sensor nodes would be financially infeasible. Typically devices used in these settings are small sensor nodes with only about 1-128k of RAM (used for volatile data), about 5-256k of ROM (used for program and constant data) and 1-100 MIPS of computing power. For monitoring vibration, acceleration sensors or even special low-cost vibration sensors can be used. This is a very ambitious setting for monitoring systems and requires us to minimize resource consumption especially regarding RAM and ROM.

The types of algorithms that can be executed on such a platform are highly restricted. Current solutions for monitoring industrial facilities therefore usually have a centralized architecture: sensor nodes send measurement raw data to a central server, which then performs all computations necessary for monitoring. As mentioned above, problems of the centralized architecture are: communication overhead for continuous sending of raw data and lack of robustness, as failures of the central server render the whole system unusable. Therefore decentralized architectures have been proposed. With a decentralized architecture, monitoring is realized locally on the sensor nodes. An example is the light-weight Prolog implementation proposed in [7]. However, energy restrictions in our scenario make a more light-weight approach to distributed monitoring necessary, which can only be realized by combining the computational power of several sensor nodes.

3. Distributed Sensing

In our scenario, energy consumption of sensor nodes, and robustness are critical,. Communication overhead must be compensated through energy harvesting from the continuous motion of the belt. We present first results on parts of a qualitative reasoning task, that can be performed using only local value comparison and local communication between nodes. We discuss how a global property, namely that a node has a maximal value for some measurement dimension, can be computed. For the rest of the paper we discuss the following simplified scenario. We consider a row of sensor nodes along a conveyor belt. A typical problem described above is conveyor belt wear. In our simplified example, nodes measure vibration on the scaffolding and go for several minutes into an alert state when they detect that a certain threshold of vibration has been exceeded. The nodes then track the maximal vibration, by successively computing it, every 10 seconds. A node that believes it has the maximum at a given time lights a red LED, so that a maintenance

	$P(y,z)$	y = z	$iP(y,z)$
$P(x,y)$	$P(x,z)$	$P(x,z)$	any
$x=y$	$P(x,z)$	x = z	$iP(x,z)$
$iP(x,y)$	any	$iP(x,z)$	$iP(x,z)$

	converse relation
$P(x,y)$	$iP(y,x)$
$x=y$	$y=x$
$iP(x,y)$	$P(y,x)$

Table 1. Composition table and converse relations for a partial order P, $=$, and its inverse iP. The entry *any* denotes the set of all three relations, that is, a situation, in which no information can be inferred.

worker can find the position in the conveyor belt that has the defect before it becomes critical and maybe even repair it without stopping the machine.

Although the computation of the maximum in a node itself is not a challenging computational task, it serves as an example to illustrate how local reasoning steps with an interesting class of relations can establish global knowledge.

3.1. Partial Orderings of Sensor Values

Partial order reasoning can be used for reasoning about sensor values. It provides mainly transitivity inference, a type of inference that can be realized efficiently and is also particularly useful: examples of partial orders are the *part-of* relation underlying mereotopological reasoning [4], the sub-concept relation of description logics, and the logical consequence relation of itself. Also, any linear ordering relation, including sensor value comparison, is a special type of partial order.

A partial order P is a relation that has the properties of antisymmetry (1) and transitivity (2). It is called *strict*, if it is asymmetric (3). The converse relation of P is called iP (4).

$$\forall x, y : P(x,y) \wedge P(y,x) \rightarrow x = y \tag{1}$$

$$\forall x, y, z : P(x,y) \wedge P(y,z) \rightarrow P(x,z) \tag{2}$$

$$\forall x, y : P^s(x,y) \rightarrow \neg P^s(y,x) \tag{3}$$

$$\forall x, y : iP(x,y) \leftrightarrow P(y,x) \tag{4}$$

Table **??** illustrates the simplicity of transitivity reasoning with a partial order: inference yields either a unique, that is certain, result or no information (entries *any*). This makes it an ideal candidate for implementation on a sensor network.

When we directly compare two sensor measurement values m_a and m_b, e.g. frequencies $15Hz$ and $60Hz$, we can exactly determine whether $m_a > m_b$, $m_a = m_b$, or $m_a < m_b$. This property is characteristic for linear orders:

$$\forall x, y : P_{\text{lin}}(x,y) \vee P_{\text{lin}}(y,x) \tag{5}$$

In sensor networks however, we usually reason about a large number of values of which only local, relative comparison information is known: nodes can easily exchange information locally with their neighbors, but sharing information globally is costly. Thus, the linearity constraint, being a disjunctive constraint, is difficult to employ for practical reasoning. We should therefore only employ the partial order properties and limit transmission of global information as far as possible.

3.2. Distributed Sensing through Local Communication

Algorithm 1 illustrates, for the example of the computation of the maximum, how inference on a sensor network can be realized through communication between neighboring nodes. The algorithm serves two purposes: most specifically, it shows that a network of very simple sensor nodes can be used to monitor alarm conditions; more generally, the algorithm exemplifies how partial order reasoning can be realized on WSNs.

Sensor nodes exchange messages of a fixed format. The message reflects that the sender y is 'larger' with respect to some partial order R than some other (possibly remote) node z. The receiving node x uses the message together with its own sensory input to evaluate whether it is larger than the sender: if yes, x sends this information to all neighboring nodes; if it is not larger, it knows it cannot be the maximum and sends no message. Due to the different behaviors in the two cases, information flows 'upward' along the gradient of the measurement signal. The recipient x only sends a message if its sensor value is larger than that of the sender y.

Algorithm 1 At a node with id x given sensor value v_x and a message msg sent from another node y, determine whether x is 'larger' (with respect to a relation R) than y, and update other nodes. Reasoning is performed with numerical comparison using a default interpretation for the relation R ($\vdash$). The transitivity inference of Table 1 is achieved through the sending operation. That is, the network as a whole performs the inference, whereas the individual nodes only perform comparison operations.

Require: The node x is the ID of the node performing the computation. The variable y denotes a (possibly remote) node. The threshold v_t determines whether a measured value indicates that an event occurred.

```
while true do
    v_x ← measurement()                          ▷ retrieving measurement
    if KB ⊢ R(v_x, v_t) and KB = ∅ then          ▷ if v_x is 'larger' w.r.t. R than v_t
        LED on                                   ▷ going into alarm state
        send({v = x, R(x, x)})                   ▷ updating the neighborhood about event
        timer.start()                            ▷ going into alarm state
    end if
    if message received then
        msg ← {v_y = y, R(y, z)} ← received()    ▷ using fixed format messages
        if msg ⊢ R(v_x, v_y) then                ▷ evaluating
            send({v_x = x, R(x, y)})             ▷ updating the neighborhood
        else if msg ⊢ R(v_y, v_x) then           ▷ evaluating
            LED off                              ▷ the source of the alarm is not here
        end if
    end if
    if timer.done() then
        KB ← ∅                                   ▷ ending alarm state
    end if
end while
```

As a distributed algorithm on the network, the algorithm computes the maximum value measured and lights the LED on the node that measured it. From a local perspective

of a single node, the algorithm informs the node about its relation to possibly distant other nodes and allows it to detect whether it is closest to some measurable event. The goal for each node is to determine whether it is the node closest to the defect, that is, the node with the most vibration. As long as it believes it is the node closest to the defect, it keeps the warning LED on. If it is updated that another node measured a higher value it turns its LED off. If another node measured a lower value, but sent it out, nodes in the immediate environment have to be updated that there is a larger value.

The transmitted information can become invalid as sensory input changes. As long as sensory information remains constant however, the information transmitted through the algorithm establishes a global hierarchy of nodes based on their sensor values. In our experimental evaluation below, we achieved this by keeping the measurement function constant for a certain time after an event occurred.

4. Experimental Implementation

We implemented the algorithm on a JN5139-based sensor node platform[3] with the low-cost mirco-vibration ballswitch sensor described in [2]. To illustrate our scenario we performed two experiments. The first experiment was conducted at a realistic scale but with simulated sensor data for showing the general setting of sending and reception of messages in the scenario. The second experiment was set up at a smaller scale but with actually measured sensor data, illustrating how the algorithm works on actual sensor data. We separated the two experiments, as it is difficult to generate vibrations in a larger scale. However, both experiments were performed using the same distributed sensing algorithm.

In the first experiment, we placed sensor nodes along a hallway at a distance of about 5 meters as in the open-cast mining scenario: nodes were initialized so as to always measure a fixed value, as shown in Figure 5. They were laid out along the gradient of this simulated measurement value. To start the experiment, we sent a start-signal to every node and after 2-3 seconds the LEDs on every node were switched off except for the node with the maximum.

The second experiment (see Figure 6) focused on the sensory aspect. We placed the sensor nodes on a table and generated a vibration signal by knocking once on the table, so that the communication between neighboring nodes would be triggered. Due to the short distances between the nodes, the computation of the maximum was completed faster than in the first experiment. After about one second, the node number four with its still switched on LED indicated that it detected the most vibration after the knock on the table. We could also notice the process of exchanging messages between the nodes and the area of high vibration values getting smaller and smaller. At first, node number two, six and seven switched off their LEDs because they received information about a vibration value which was higher than their own measurement. The transmitter of these messages could, for instance, have been node number two, four or five, since they had less distance to the knock. The same happened with the other nodes except for number 4 which never received a higher value than its own. So the maximum and the origin of the vibration was found. The results illustrate that our algorithm for distributed sensing

[3] http://www.jennic.com/products/wireless_microcontrollers/jn5139

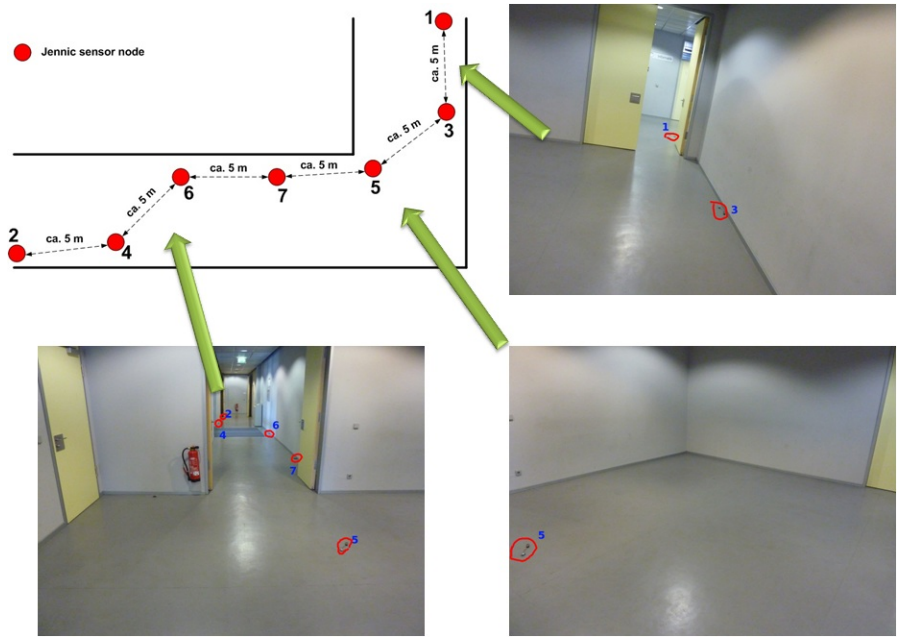
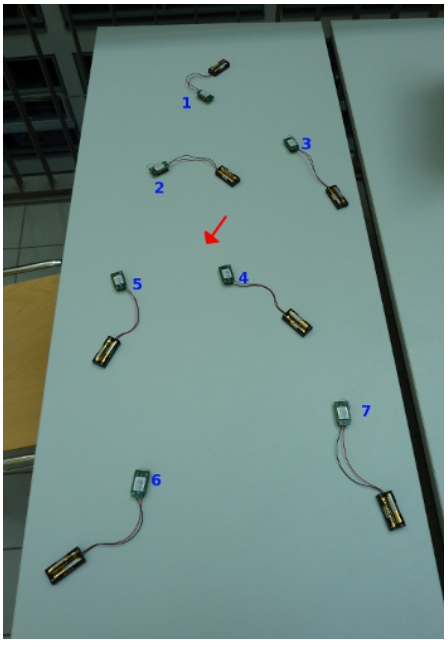


Figure 5. Experiment 1: sensor nodes are laid out along a hallway at approximately 5m distance.



(a)

short address	sent messages	received messages	LED on time (ms)
1	1	53	585
2	1	96	87
3	13	31	272
4	7	6	∞
5	2	13	804
6	5	39	220
7	6	87	123

short address	sent messages	received messages	LED on time (ms)
1	4	13	1170
2	4	82	87
3	8	48	544
4	8	7	∞
5	3	15	704
6	1	87	220
7	1	118	89

(b)

Figure 6. Experiment 2: design (a) and timings for two runs (b). The red arrow in (a) indicates the location of the knocking event and the numbers indicate the short-addresses of the sensor nodes, as listed in (b).

is able to detect the origin of an occurring event using sensor values of several nodes. It worked effectively in both the small and wider range layout.

5. Conclusions and Future Works

We presented a distributed sensing system usable for monitoring industrial facilities and described an example scenario of conveyor belt monitoring in open-cast mining. The approach features robustness of the whole system, lower communication overhead and light-weight implementation. Information is transmitted along the gradients of a measurable vibration event, and actuators can be triggered by the nodes that are closest to an event without any communication with a back-end server. This guarantees high network reliability, as failure of a single node can be compensated by its neighbors. Two experiments with an implementation were presented to illustrate actual feasibility in the scenario.

The computation of the node registering the maximal vibration is based on the particularly simple inference mechanisms underlying partial order reasoning. The presented study suggests that this type of reasoning is light-weight enough to be practically feasible as a reasoning mechanism for WSNs and versatile enough to apply to a broad range of interesting problems. In future works, we will develop a more expressive communication protocol instead of the fixed message format. The presented work is a first step towards light-weight reasoning facilities for context-awareness on sensor node hardware [1].

References

[1] Michael Beigl, Albert Krohn, Tobias Zimmer, Christian Decker, and Philip Robinson. Awarecon: Situation aware context communication. In *Ubicomp*, pages 132–139. Springer, 2003.

[2] Dawud Gordon, Georg Von Zengen, Hedda Rahel Schmidtke, and Michael Beigl. A novel micro-vibration sensor for activity recognition: Potential and limitations. In *Proceedings of the Fourteenth International Symposium on Wearable Computers (ISWC 2010)*, 2010.

[3] Sukun Kim, Shamim Pakzad, David Culler, James Demmel, Gregory Fenves, Steven Glaser, and Martin Turon. Health monitoring of civil infrastructures using wireless sensor networks. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 254–263, New York, NY, USA, 2007. ACM.

[4] D.A. Randell, Z. Cui, and A.G. Cohn. A spatial logic based on region and connection. In *Knowledge Representation and Reasoning*, pages 165–176. Morgan Kaufmann, 1992.

[5] Detlev Roseburg. *Elektrische Maschinen und Antriebe.* Carl Hanser Verlag, 1999.

[6] Rolf Dieter Stoll, Christian Niemann-Delius, Carsten Drebenstedt, and Klaus Muellensiefen. *Der Braunkohlentagebau: Bedeutung, Planung, Betrieb, Technik, Umwelt.* Springer Verlag, 2008.

[7] Martin Strohbach, Hans-Werner Gellersen, Gerd Kortuem, and Christian Kray. Cooperative artefacts: Assessing real world situations with embedded technology. In Nigel Davies, Elizabeth D. Mynatt, and Itiro Siio, editors, *Ubicomp*, pages 250–267. Springer, 2004.