FlowerNet - How to design a user friendly Sensor Network

Bjoern Gressmann and Horst Hellbrueck

Abstract Today, sensor networks are on the way from simulation and short term measurements in the laboratory to real long term applications where devices operate for several weeks without administration. In this paper we describe the design, development and evaluation of a suitable application for further measurements of sensor network protocols and applications where users can benefit from the services of this network in daily life. Energy efficiency, reliability and adaptability are key features to fulfill the requirements of the users. We decide to develop a monitoring system for potted flowers at our institute and support the users with automatically generated decisions for watering the flowers. We perform long term measurements and evaluate the whole system including the benefit that the users experience. In this paper we present first results and the long term evaluation of the system.

1 Introduction

Simulations were standard in the Mobile Ad Hoc Network (MANET) and sensor networks research community in the past for years. First experimental results show that simulation models for sensor networks are far from being realistic and protocols developed in simulations might fail in real world environments.

Reference implementation of protocols and their evaluation in MANET or sensor network testbeds help to improve the confidence in protocols. The research focus of these testbeds is evaluation of protocol and variants which does not require long term measurements in many cases. In these field experiments data and information transferred as well as user interaction were only of secondary interest.

Bjoern Gressmann
Institute of Telematics, University of Luebeck, e-mail: `gressmann@itm.uni-luebeck.de`

Horst Hellbrueck
Institute of Telematics, University of Luebeck e-mail: `hellbrueck@itm.uni-luebeck.de`

Before sensor networks can be deployed in real world application new challenges need to be solved. A very practical issue is the required interaction with individuals. Systems under test need to offer a real user benefit, so that users ideally voluntarily utilize these systems regularly.

In this paper, we introduce *FlowerNet* as a suitable application to conduct protocol evaluations as well as user acceptance surveys in a simple setup. *FlowerNet* allows long term evaluation with a multi hop radio network and is based on the ESB sensor network platform ([1]) that provides a convenient programming environment, a simple radio interface and an extension connector for additional sensors.

The user acceptance for sensor network applications poses important challenges for such a system. Acceptance depends on the reliability and correctness of the system as well as the service and the way the service is offered. Systems like *FlowerNet* should provide the user with recommendations and detect faults and recover from them silently instead of dominating the user with instructions that might be even wrong.

The rest of the paper is organized as follows. Next section introduces related work. Section 3 lists all the user requirements and some prerequisites and constraints given by the hardware. In the next sections we will discuss design decisions where we focus on the adaptability of the application. We will finish the paper with a first evaluation of the system and an outlook for the future work.

2 Related Work

Environmental monitoring and in-situ habitat monitoring are two main applications of Wireless Sensor Networks.

Possible applications besides monitoring of plants in private environment are the improvement of agricultural production, monitoring of animal populations or climate data acquisition.

A project also addressing monitoring of plants is the Botanicalls project ([2]) based on ZigBee. A main feature of this application is the creation of a new communication channel between a plant and its owner using Voice over IP. The user can call plants to get information about their condition and the plants call the user to ask for water.

On Great Duck Island a population of Ceach's Storm Petrell is monitored by a WSN consisting of Mica Nodes ([3],[4]).

A research group at Agrosphere Institute, Forschungszentrum Juelich, is working on a project called SoilNet ([5]), in which a WSN based on the ZigBee Standard acquires soil moisture data in areas of 100's of meters.

The Computational Modeling Laboratory of the National Agricultural Research Center in Japan developed a system which acquires data of different environmental parameters in paddy fields and tries to improve the crops ([6]).

Within the MarathonNet project a monitoring system for runners in a running event presents various data of the participants to the interested audience, even in the Internet ([7],[8],[9]). Besides the implementation and evaluation of the system the application puts the main focus on usability and the special requirements of the devices in a sports event.

Practical approaches to animal tracking have been developed in the ZebraNet-project ([10]) and in the Electronic Shepherd-project ([11]).

In the Heathland experiment network characteristics have been evaluated in practical application ([12]) to collect experimental results about connectivity and data loss rates.

These examples of related work mainly focus on the network properties and the data collection itself. The collected data is provided to research people so that user friendliness in the first place is not that important. Researchers for soil measurements are interested in fine granular, long term raw data for a certain region. However, *FlowerNet* focuses on getting experience and results in the acceptance of the use of WSN technology in everyday life and not on the new network protocols as many other related work. Users of *FlowerNet* expect valuable information instead of raw data, which calls for a different focus of the project. The user expects the sensors to be self-adaptive and self-learning, so that they support the watering of cactuses in sandy soil as well as roses in humus. We will discuss the requirements of the users for a support system for flower care in the next sections.

3 Requirements

FlowerNet as a typical wireless application requires flexibility, fault tolerance, usability, reliability and a long life time.

As the user will not be interested in the technical aspects of the application, the presentation of data and the notification of the events has to be user friendly and appealing. Although routing is not the focus of this work the routing algorithm needs to support moving of flowers without user interaction to the system. When events of interest occur, the system sends an email to the user. The user can monitor his flowers by visiting a website which presents the acquired data graphically. Configuration and deployment of the network must be easy and intuitive. Thus, another aspect to achieve user friendliness is the adaptability of the event detection mechanism. Thereby the system adapts to the watering behavior of the user and special knowledge of the characteristics of the plants is not required for using *FlowerNet* .

Besides the technical requirements, the enclosure of the nodes should not affect the decorative function of the flowers and should be nearly invisible. The risk of damaging the roots of the flowers by inserting the measurement contacts into the soil should be minimal. There is a need for mechanisms to filter double events or wrong notification by incorrectly detected events. The nodes have to be protected by a housing from water and physical damage but still allow for easy battery change. An e-mail sent to the user in order to replace batteries is also a must in the system.

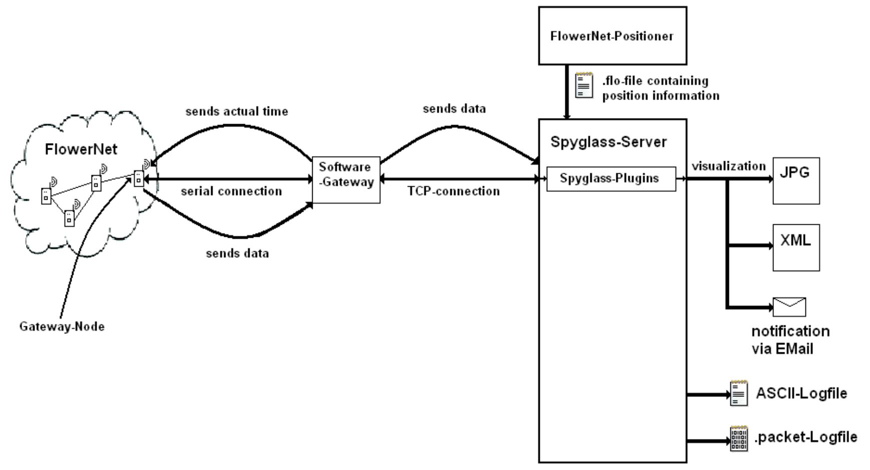


Fig. 1 *FlowerNet* Components

4 Network Design

This section describes the most important network design decisions that were made to fulfill requirements of *FlowerNet* .

The *FlowerNet* -application consists of several components (Figure 1). Data is collected in *FlowerNet* and routed to the gateway node, which propagates data to a software gateway. This software gateway also acts as a time server for the synchronization. The SpyglassServer ([13]) processes the data and creates the graphical output, an XML/XSLT file with the actual state of the plants. Additionally, log files are written, containing the acquired data since system start. Also, it sends the notification emails to the users.

For an easy configuration of the application, the tool *FlowerNetPositioner* has been developed. This tool enables easy configuration of positions of the flowers and create the graphical representation of the building map to achieve a nice graphical presentation of the collected data to the user. Furthermore, it allows the administrative users to enter email addresses of users in charge of each flower for the system notifications that are sent. The user receives e-mails in case of low battery to change the batteries and if the gateway node detects that nodes do not send packets for a long time period. This allows the user to react to error situations.

To assure a long life time, the system operates very energy efficient and nodes avoid collisions on the medium. The used routing protocol assures flexibility of positioning the nodes and achieve reliability of data transmission (Subsection 4.2). Figure 1 depicts the important components as described so far.

4.1 Schedule-based Time Division Multiplexing

This section presents the MAC layer of *FlowerNet* , which is determined as a schedule based time division multiplexing (STDM) with fixed intervals in which the radio transceivers are set into sleep mode to reduce energy consumption. As the number of nodes is known in advance and there are no real-time requirements, this simple solution fits well to the application.

For STDM, suitable time synchronization between the nodes is mandatory. *FlowerNet* synchronizes periodically. Each synchronization event is initiated by the gateway node which receives the actual time from the Software-Gateway. Each node has an interval of ONPERIOD seconds in which the transceiver is active as shown in Figure 2. This interval is called sending period. Afterwards the radio interface is switched off for OFFPERIOD seconds. During the sending period, all nodes have to be active too, as data forwarding to the gateway node is performed in a single sending interval. The duty cycle can be reduced by adapting the parameters ONPERIOD and OFFPERIOD. During the sending period, the node has a power consumption of 7.15mA. When the receiver is turned off, the power consumption reduces to 1.5 mA. So, the theoretical life time of a node using batteries with an energy of $E_{battery}$ in mAh is calculated by formula (1).

$$\frac{E_{battery}}{\frac{ONPERIOD}{OFFPERIOD} \times 7.15mAh + (1 - \frac{ONPERIOD}{OFFPERIOD}) \times 1.5mAh} \tag{1}$$

In our tests, we used 2000mAh batteries and set ONPERIOD to 6 and OFFPERIOD to 66, which results in a theoretical lifetime of 41.4 days. Although we know that this approach does not scale for large sensor networks, in *FlowerNet* with about tens of nodes it is appropriate.

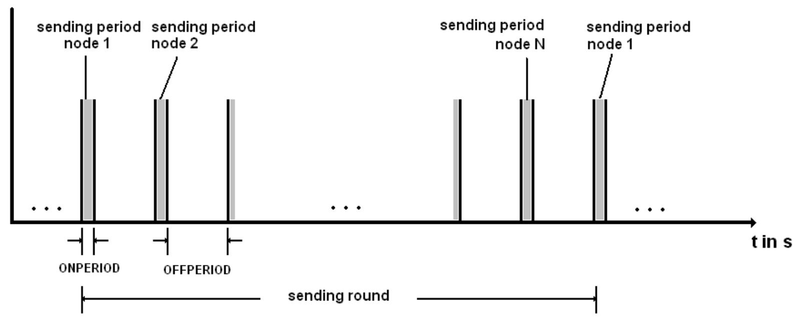


Fig. 2 Time Division Multiplexing scheme

There are several situations in which nodes need to connect to the network. In this case, nodes are not synchronized with the network yet. After the system start, a node stays active and periodically sends SIGN-Requests which indicate that this node is not synchronized yet. When the node receives network synchronization response messages it sets its own time and multiplexing information accordingly. This approach offers flexibility in deployment of the nodes, as there is no need to deploy any node at any special point of time.

4.2 Routing

As robustness of the system is a key requirement, we decided to use a routing protocol based on link qualities in the local neighborhood of the nodes as routing metric.

A suitable link metric has been presented in [14] and is implemented in *FlowerNet* . When a node receives a packet, it calculates a link quality for the link from the neighboring node and itself. This quality metric is calculated from sequence numbers in the data packets. After each interval of 32 packets sent by a neighbor the packet delivery rate is calculated and defines the downstream link quality. For further details we refer to [14].

The routing is organized in rounds triggered by the gateway node as a central component. It initiates a new routing round periodically, where nodes build a spanning tree on top of the existing meshed network graph. For this purpose, the gateway node broadcasts a routing update packet that is forwarded within the network These routing update packets contain the actual parent node and the sum of the link quality on the optimal path to the gateway node. When a node receives a routing update packet, it compares its actual quality of the routing path with the sum of the received path quality plus the quality of the link to the sending neighbor. If the new offered path has better quality the receiving node updates its routing information and broadcasts this new path to the other nodes in a new routing update packet.

5 Adaptive Dryness Measurement

One of the main requirements of the application is the adaptability of *FlowerNet* to specific characteristics of a plant. To achive this goal, nodes monitor data and learn about dryness thresholds in a learning phase at the start of the application and adapt these thresholds during operation.

The implementation of the event detection was based on certain assumptions. The dryness of ground increase with time and a watering will have a sudden measurable effect on the dryness of the soil.

There are several uncertainties that affect the measurement. For instance, the electric conductivity of different soil types can vary and different plants need different humidity levels in the soil which requires adaptability of the system. The

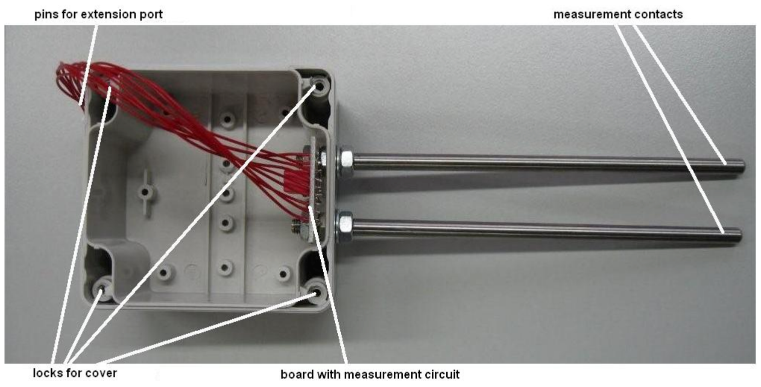


Fig. 3 Enclosure of the nodes

measured value depends on the insertion position of the contacts. So, the system will experience sudden changes in the measured values if the user repositions the plant. After the node has collected steady data again it gets back into a stable mode.

To measure the dryness we charge a capacitor through the soil with a current of 1mA. The soil acts a resistor in that case. The raw value is the time in milliseconds until the capacitor reaches a threshold voltage. The lower the resistance of the soil, the lower is this measured time, so we measure the dryness of the soil.

Measurements are repeated ten times and averaged to reduce the influence of variations due to measurement errors. This averaged value is called a dryness measurement (*dryness*) in the following.

Nodes detect on their own several events:

- The flower has been watered.
- The flower has to be watered because the soil is dry.
- The flower has to be watered because it's time.
- The flower has been watered, but the system did not detect a necessity for it.
- An error has been detected, no valid measurements possible at the moment.
- Node left error state, node collects valid dryness data again.

After a measurement has been executed, the node forecasts the next dryness value by using the delta between two measurements (*diff*). Diff is adapted by using exponential smoothing first order:

$$diff_{n+1} = (1 - \alpha_{diff}) \times diff_n + \alpha_{diff} \times new\_diff \qquad (2)$$

The parameter $\alpha_{diff} \in [0..1]$ controls the filtering. The forecast dryness value is then $dryness_{forecast} = dryness_n + diff_n$.

If new dryness value differs from this forecast value more than a threshold, a watering event or an error is detected, depending on the new value. If the actually measured value is smaller, the dryness level of the soil has decreased, thus, if (3) evaluates to *true*, the system detects a watering event:

$$dryness_{n+1} \leqslant DRY\_FRAC \times dryness_{forecast}, \;\; DRY\_FRAC \in [0..1] \tag{3}$$

On the other side, if actual measurement value is larger, the dryness level of the soil has suddenly increased, possibly by changing the position of the two contacts, thus, if (4) evaluates to *true*, the node enters an error state:

$$dryness_{n+1} \geqslant ERROR\_FRAC \times dryness_{forecast}, \;\; ERROR\_FRAC > 1 \tag{4}$$

When the situation stabilizes again, node returns to normal mode again.

There are two thresholds which can be reached to indicate that a watering of the flower is necessary. The robustness and adaptability of the measurement is achieved by learning the initial thresholds during operation in a monitoring mode. The dryness threshold is set to the measured value immediately at the point of time of watering. The time threshold is initialized to the time interval between the first two watering events.

After a detected watering, the dryness threshold and the time threshold are adapted according to the following formula.

$$threshold_{n+1} = (1 - \alpha_{threshold}) \times threshold_n + (\alpha_{threshold}) \times dryness_n \tag{5}$$

$$timeThreshold_{n+1} = (1 - \alpha_{time}) \times timeThreshold_n + \alpha_{time} \times time_n \tag{6}$$

The parameters $\alpha_{threshold} \in [0..1]$ and $\alpha_{time} \in [0..1]$ describe how strong the new threshold is adapted.

The dryness threshold is the marker at which the plant should be watered when dryness level of the soil rises. If (7) evaluates to *true*, the user is notified that the flower has to be watered because the soil is dry.

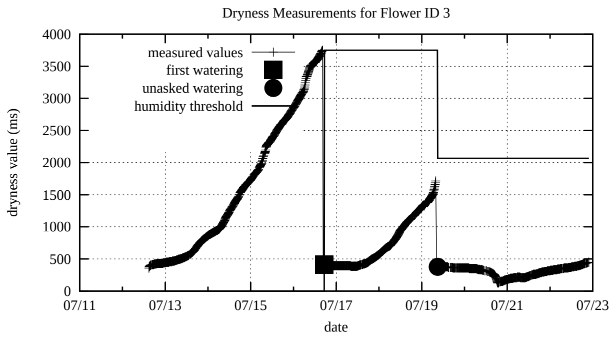
$$dryness_{n+1} \geqslant THRESHOLD\_FRAC \times threshold_n, \;\; THRESHOLD_FRAC \in [0..1] \tag{7}$$

THRESHOLD_FRAC is the fraction of the dryness threshold that the measured dryness value has to exceed for notifying the user to water the plant.

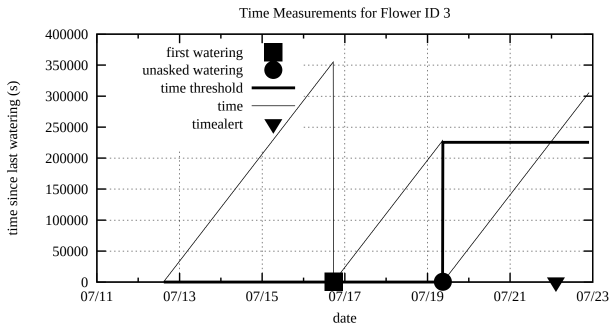
Because of errors that lead to an irregular decrease in the dryness level of the soil we add another condition (8) to notify the user. This condition is true when a period of time has elapsed since the last watering. Then the user is notified that it is time to water the plant again.

$$time_{actual} \geqslant (timeThreshold_n + (TIME\_INTERVAL \times 3600)) \tag{8}$$

A delay for the notification can be set by TIME_INTERVAL. There are several parameters to configure the event detection and the adaptation of the thresholds that have to be set in a proper way to achieve a coorect behavior of the system.



(a) dryness measurement



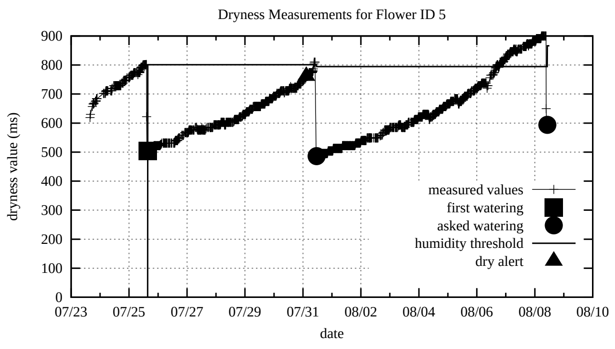
(b) time measurement

Fig. 4 Test run number 1; node with id 3

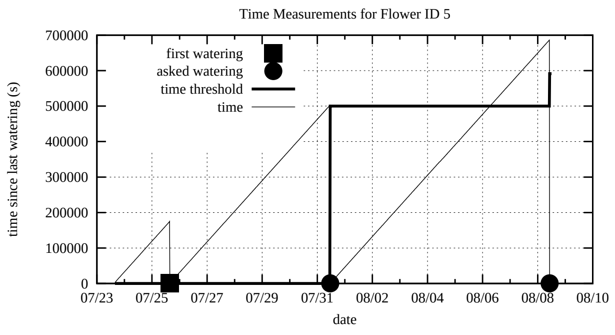
6 Evaluation

As described so far the design and implementation focus on robust and simple solutions. In this section we will present first evaluation results. Therefore, we started 2 test runs for approximately 2 weeks, where we apply the above described algorithm and analyzed the gathered data from the server across the network. The network runs stable for a period of 2 weeks time until nodes run out of battery power.

The diagrams Figure 4(a) and Figure 5(a) illustrate the measured dryness values and the dryness thresholds. The diagrams Figure 4(b) and Figure 5(b) show the time since the last watering and the time threshold after the second watering. Additionally



(a) dryness measurement



(b) time measurement

Fig. 5 Test run number 2; node with id 5

the diagrams display the events detected by the system. A watering of the flowers results in one of the three events:

- The flower is watered for the first time, then the node detects a first watering event.
- The flower is watered after it sent a notification for need of water to the user, then the system detects an asked watering event.
- The flower is watered, but sent no notification to the user, then the system detects an unasked watering event.

Events that trigger a notification are dry alerts which are detected if dryness values have reached the dryness threshold and time alerts if the time since last watering has exceeded the time threshold.

As can be seen in Figure 4 and Figure 5 in the first days, the system just monitors and collects dryness values. The measured value increases slowly due to loss of humidity until the plant is watered for the first time by the user. Then the sensor node sets a first threshold for a dry alert to inform the user when this state is reached again in the future. A dry alert can be seen in Figure 5 at day 07/31. The time threshold is learned after the second watering, as can be seen in Figure 4(b) and Figure 5(b). The user is informed to water his flower because of a time alert in Figure 4(b) at day 07/22. This is reasonable because the measured dryness values after the second watering differ a lot from the measured dryness values before the first and after the first watering (Figure 4(a)) and the dryness threshold might be reached lately in the future. Furthermore, the sensor nodes detect also extra watering and adapt the threshold to a new value as can be seen in Figure 4 at day 07/19. The different characteristics of the soils in Figure 4(a) and Figure 5(a) result in completely different ranges of the dryness values and show the necessity of a self-learning event detection mechanism.

For the proof of reliability we created the following test cases:

- Switch off sensor nodes simulates device failures
- Move plant with sensor node out of range simulates transient network failures
- Pull sensor nodes out of the pot to produce dryness measurement errors
- Run until end of battery to test battery replacement messages

FlowerNet had passed all test cases as nodes created error messages accordingly that were sent as e-mails to the user and successfully recovered from the errors again.

The test users respond in a positive way to e-mails about detected watering events as a helpful feedback for the correct operation of the system. As the user has to perform regular watering first so that the system can learn the threshold values, the system is not able to give reasonable decision support for users who just water their plants in a random fashion. The system was also confused be the effect that users water their plants before some days of vacation as *FlowerNet* is not aware of the days of a week.

7 Conclusion

In this paper we described a robust implementation of a sensor network where users communicate interactively with the system. Sensors detect watering and anomalies, set thresholds themselves and forward this data backwards along a spanning tree to a central server where e-mails are generated to remind users to water the plants. The system adapts to new situations where for example the sensor is moved or the user decides to change the watering behavior of the plant. The measurement is very accurate, so that even the daily (temperature) cycles are visible in the dryness measurements. For the future we work on the improvement of the lifetime of the system, so that the system is not discontinued every two weeks. This is even more important as users do not water a plant in very regular fashion. The use of replacement batteries could rise the lifetime up to six weeks and makes longer studies possible. There are certain situations like weekends or vacations where users change the watering behavior unpredictable such as increase the amount of water or water the plant too early in time. This shows that sensor networks need to take into account the needs of the users to offer an optimal service. With too many false positives, users will ignore the support, and will even be annoyed by "wrong" support decisions. We address these issues at the moment.

References

1. “Scatterweb,” http://cst.mi.fu-berlin.de/projects/ScatterWeb/.
2. “The botanicalls project,” http://www.botanicalls.com.
3. A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, “Wireless sensor networks for habitat monitoring,” in *WSNA ’02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. New York, NY, USA: ACM Press, 2002, pp. 88–97.
4. R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, “An analysis of a large scale habitat monitoring application,” in *SenSys ’04: Proceedings of the 2nd international conference on Embedded networked sensor systems*. New York, NY, USA: ACM Press, 2004, pp. 214–226.
5. “Soilnet - a zigbee based soil moisture sensor network,” http://www.fz-juelich.de/icg/icg-4/index.php?index=739.
6. “Field server - a wireless sensor network for plant and field condition monitoring,” http://www.agnet.org/library/pt/2004033/.
7. H. Hellbrueck, M. Lipphardt, D. Pfisterer, S. Ransom, and S. Fischer, “Praxiserfahrungen mit marathonnet - ein mobiles sensornetz im sport,” *PIK - Praxis der Informationsverarbeitung und Kommunikation*, vol. 29, no. 4, 2006.
8. D. Pfisterer, M. Lipphardt, C. Buschmann, H. Hellbrueck, S. Fischer, and J. H. Sauselin, “Marathonnet: Adding value to large scale sport events - a connectivity analysis,” in *Proceedings of the International Conference on Integrated Internet Ad hoc and Sensor Networks (InterSense)*, May 2006.
9. M. Lipphardt, H. Hellbrueck, D. Pfisterer, S. Ransom, and S. Fischer, “Practical experiences on mobile inter-body-area-networking,” in *Proceedings of the Second International Conference on Body Area Networks (BodyNets’07)*, 2007. [Online]. Available: http://www.bodynets.org/
10. P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi, “Hardware design experiences in zebranet,” in *Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM Press, 2004, pp. 227–238.
11. B. Thorstensen, T. Syversen, T.-A. Bjornvold, and T. Walseth, “Electronic shepherd - a low-cost, low-bandwidth, wireless network system,” in *MobiSys ’04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*. New York, NY, USA: ACM Press, 2004, pp. 245–255.
12. V. Turau, C. Renner, M. Venzke, S. Waschik, C. Weyer, and M. Witt, “The heathland experiment: Results and experiences,” in *Proceedings of the REALWSN’05 Workshop on Real-World Wireless Sensor Networks*, Stockholm, Sweden, Jun. 2005.
13. C. Buschmann, D. Pfisterer, S. Fischer, S. P. Fekete, and A. Kröller, “Spyglass: A wireless sensor network visualizer,” in *SIGBED Review, Vol. 2, No. 1, 2005*, 2005.
14. M. Yarvis, W. Conner, L. Krishnamurthy, J. Chhabra, B. Elliott, and A. Mainwaring, “Real-world experiences with an interactive ad hoc sensor network,” 2002. [Online]. Available: citeseer.ist.psu.edu/yarvis02realworld.html