Building an intelligent sensing system: a case study

Waltenegus Dargie* and
Alexander Schill

Chair of Computer Networks,
Department of Computer Science,
Technical University of Dresden,
Dresden, 01062,
Germany
E-mail: waltenegus.dargie@tu-dresden.de
*Corresponding author

Abstract: The conception and development of pervasive systems involve interdisciplinary team work. The team consists of people with different research background. While such a composition is desirable to solve real-world problems, it brings with it also challenges. To begin with, team members should establish a shared understanding of what should be done. Secondly, separately developed subparts should be brought together to make up a unified, consistent and side-effect free system. In this paper, we discuss the development of the senceive system as a graduate project course work. The project involves computer science, computer engineering and electrical engineering students. Technically, the Senceive system offers a stepwise abstraction of low-level concerns (sensing) from higher-level use of meaningful features.

Keywords: activity recognition; project work; complex system design; wireless sensor networks; network configuration.

Reference to this paper should be made as follows: Dargie, W. and Schill, A. (2012) 'Building an intelligent sensing system: a case study', *Int. J. Autonomous and Adaptive Communications Systems*, Vol. 5, No. 1, pp.3–17.

Biographical notes: Waltenegus Dargie is a Researcher at the Technical University of Dresden. He obtained a PhD in Computer Engineering from the same university in 2006. He holds a MSc from the Technical University of Kaiserslautern, Germany (2002) and a BSc from the Nazareth Technical College, Ethiopia (1997), both in Electrical Engineering. Prior to his current position, he has been a Researcher at the University of Kassel, Department of Electrical Engineering and Computer Science (2002–2005) and at the Fraunhofer Institute of Experimental Software Engineering (2002–2003). His research interests include autonomous computing, context-aware computing, wireless sensor networks and digital signal processing. He is a member of the IEEE and IEEE Communication Society.

Alexander Schill is a Professor for Computer Networks at Dresden University of Technology. His major research interests are distributed systems and service-oriented architectures, multimedia communication and teleconferencing and mobile and ubiquitous computing. He holds a MSc and a PhD in Computer Science from the University of Karlsruhe, Germany, and received a Drhc from Universidad Nacional de Asuncion, Paraguay. He also worked with the IBM

Thomas J. Watson Research Center, Yorktown Heights, New York, and he has been cooperating intensively with industry. He is the author or co-author of a large number of publications on Computer Networking and Distributed Systems, including several books.

1 Introduction

Wireless sensor networks are networks of a large number of nodes which integrate an array of MEMS sensors, a small processing subsystem and a wireless transceiver. These nodes self-organise to carry out distributed sensing and to enable energy-efficient, multi-hop communications. Because of the combined features of sensing, processing and wireless communication, the networks have inspired a large number of applications.

A close scrutiny into the proposed applications reveals that similar to most applications developed within the scope of pervasive computing, the development of wireless sensor networks involves several people from different research fields. For example, in habitat monitoring, expert knowledge of wild birds and animals is required (Mainwaring et al., 2002); in structural health monitoring, knowledge of buildings and bridges as well as the static and dynamic properties of structures is essential (Koh and Dyke, 2007); for health-care applications, one is required to know about Parkinson's disease (PD) or body segments and activity of daily life (ADL) (Benbasat et al., 2007), and (Lorincz et al., 2008); for pipeline monitoring, knowledge of fluids and pipelines as well as their thermodynamic property is useful (Kim et al., 2008) and toxic gas detection in oil refineries requires knowledge about toxic gases (Dargie et al., 2009; Chao et al., 2008).

Prototype of post graduate projects require a significant conception and development time. Some of the members that make up the project team may not have system engineering or programming experience. Sometimes this results in frustrations and a significant portion of the project period is spent as members learn how to organise themselves; how to share tasks and how to integrate individual contributions.

This paper aims to demonstrate how students of different background and expertise can come together to build an intelligent sensing system. The graduate level study of computer science and engineering at the Technical University of Dresden includes a compulsory practical project work before students begin writing theses. The aim of these projects is to prepare students to develop complex systems by working in teams and by applying both theoretical and technical backgrounds. The project requires interdisciplinary interactions and a variety of hardware and software technologies.

The team size as a rule does not exceed five people. Students are expected to invest four hours a week to come together and work as a group. On average, a total of 80 hours are required to successfully complete the project. Additional 8 hours are required for seminar sessions.

1.1 Challenges

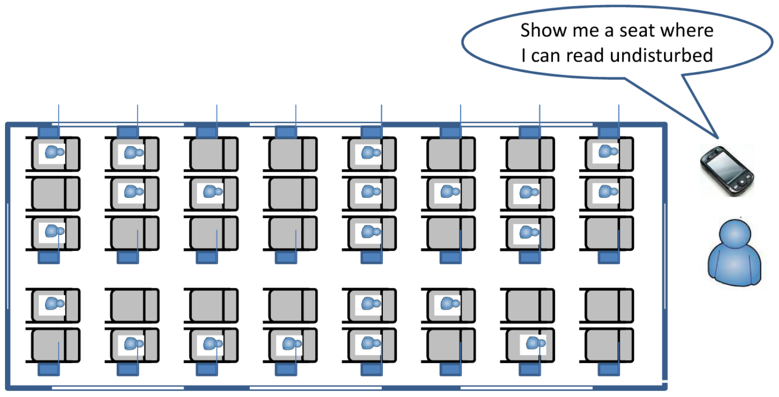
One of the typical challenges of these project courses is ensuring that all members of a team contribute equitably – neither dominant nor passive members make up an ideal team. Another challenge is the need for self-discipline and keeping deadlines – there is an associated penalty with each deadline violated. There are altogether six deadlines, five of them for reporting progress and one of them for the final defense and demonstration of results. An additional challenge is related to the division of task: on the one hand, tasks are divided according to the students' experience and interest; on the other hand, some students may not get a task for which they are well-qualified and should, therefore, acquire new. Their learning pace determines the pace of the entire team.

1.2 A model topic

As a model project, this paper introduces the Senceive system.[1] It consists of a wireless sensor network and a probabilistic reasoner to recognise various everyday human activities in different settings, such as in lecture and seminar rooms at the university and in different carriages of a passenger train. Figure 1 illustrates how the senceive system can be used in a train carriage.

The remaining part of this paper is organised as follows: in Section 2, a higher-level description of the system's features is given. In Section 3, the detailed execution plan of the project will be presented. In Sections 4 and 5, a higher-level conceptual architecture and its refined version are discussed. In Section 6, the implementation and evaluation of the Senceive system is given in detail. Finally, in Section 7, the lessons learned during the implementation of the project will be reported.

Figure 1 Activity recognition in a train carriage (see online version for colours)



2 Requirements

At the beginning of the project, students were given a general description of the system along with a few basic requirements.

2.1 Platform

Senceive will be used in smart environments. Users should interact with it to determine the activities taking place in remote physical places. Some of these activities are lectures, parties, meetings, casual discussions, etc. in lecture and seminar rooms; fighting, casual conversations, shouts, etc. in train carriages. The users are mostly mobile and own mobile devices – for example, students and professors would like to locate lecture sessions or meetings with the help of Senceive. Therefore, the system should be able to support mobility and mobile devices.

2.2 System integration

During the project work, team work is very much expected, but individual members should also be able to work independently. The balance can be made by requiring students to develop loosely-coupled components that can be brought together by well-defined interfaces. To ensure the smooth integration, students are required to develop the conceptual architecture of the systems and to specify the necessary interfaces together.

2.3 Sensing

Students were required to use existing wireless sensor nodes (Micaz and Mica2 nodes (Yick et al., 2008)) to establish the sensing subsystem. A wireless sensor network is preferred because of the ease to deploy the nodes (it is not desired to reconstruct buildings or carriages or disrupt their normal functions). Moreover, wireless sensor networks are a part of the research focus of the Chair, thus, the expected results should be exploited by some of the actively running research projects.

Furthermore, the sensing subsystem should support multiple applications. In the research community, a wireless sensor network is developed for a specific purpose so that its performance and power consumption can be optimised (Intanagonwiwat et al., 2003). This premise is the basis for developing energy efficient communication protocols and data aggregation algorithms. Because wireless sensor nodes operate with exhaustible batteries, charging or replacing these batteries is not a simple task, given the large number of nodes that should be deployed. At the same time, however, wireless sensor networks are emerging technologies. While designing energy efficient networks is important, identifying suitable applications for them and rapid prototyping and testing is vital as well. Subsequently, for this project, the energy consumption issue was not a priority concern.

The sensing subsystem offers three interfaces each of which is associated with a particular role. The first interface enables applications to access sensed data through a well-defined query language. The second interface enables an administration to have a complete knowledge of the deployed nodes and to configure the network. The administrator may not have detailed knowledge of network programming and his task is limited to adjusting some known parameters, manage access rights and monitor if all existing nodes are functioning properly. The third interface enables an experienced programmer to perform fundamental reconfigurations, not only by changing parameters, but also by entirely changing a part of the modules that are deployed on individual nodes. This type of reconfiguration is essential because:

1 during deployment, complete knowledge of the deployment setting may not be known

2 both the application requirements and the properties of the environment can change overtime

3 once a network is deployed, it is necessary to detect and fix bugs while the network is still performing the sensing task. This is, particularly, important since manual reconfiguration of individual nodes is not desirable, as it means collecting a large number of nodes that are deployed in an extensive field.

3 Milestones and procedures

One way of ensuring the timely completion of the project work is to define a limited number of milestones. Students are reminded to keep deadlines. Given the complexity of the system and the relatively short duration of the project, the challenge is not so easy. To help students better organise themselves, these milestones are predefined and the same for all teams, regardless of the topic they choose. This way, it is possible to ensure that all groups progress uniformly. Table 1 shows a detailed time plan and the milestones for the Senceive project.

To further assist students to keep deadlines, each supervisor broke down the system development task into subtasks. The breaking down of tasks into subtasks would enable the students define short and long-term goals and to better organise themselves. For the Senceive system, Table 2 displays the procedure that was prepared and made available to the students by their supervisor.

Table 1 The Senceive project schedule and milestones

Week	*Meeting*	*Assignment*	*Milestone*	*Result*
1	All	Introduction		Group formation
2	Team meeting	First team meeting; division of task		Division of task
3	Team meeting	Refinement of requirement analysis		
4	Presentation	Design of the conceptual architecture	M1	Initial specification
6	Presentation	Design of the architecture	M2	Specification
9	Implementation	Prototype	M3	In part
12	Presentation	Prototype	M4	Complete
14	Presentation	Testing and evaluation	M5	Refinement
15	Team meeting	Finalising the documentation	M6	Submission

Table 2 A guideline for a step-by-step implementation of the Senceive system

Step	*Task*	*Remark*
1	Refining the architecture	Detailed understanding of the components of the architecture
2	Platform specification	Identification of the devices and technologies required
3	Protocols and algorithms	Definition of the network's topology and data-processing algorithms
4	First phase prototyping	Implementation of the sensing subsystem
5	Modelling	Extracting higher-level features and feature modelling
6	Reasoning	Implementing and training the recognition scheme
7	Testing	Testing the system as a whole
8	Final presentation	Submission of the final document

4 Conceptual architecture

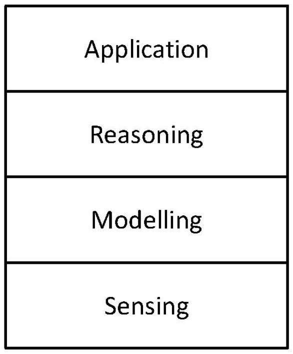
Three weeks after the project has formally started and two weeks after the teams met with their supervisor, the first milestone would be reached. By this time, students have closely studied the typical features of the system they would develop; identified their area of interest; made requirements analysis and drafted a higher-level conceptual architecture which would serve as a basis for division of labour.

Accordingly, the Senceive group met twice and individuals shared their experience and expressed their interest. It is worth to note that the students who made up the team had quite a diversity of cultural and educational background. This was a challenge at the outset, since establishing a shared understanding of the main task was difficult. The result of the two meetings was a hierarchical conceptual architecture, which is illustrated in Figure 2.

The group identified four main layers in the architecture. The bottom layer, the sensing layer, delivers raw sensor data which is the basis for reasoning the activities that take place in various places. The modelling layer extracts higher-level features from the sensed data and establishes relationships between these features. This layer requires signal processing and stochastic analysis. The features are stored in a knowledge base. The reasoning layer receives various features and feature models in order to recognise the higher-level activities that are represented by the features. Finally, applications put query and subscription requests to the recognition layer to intelligently make decisions.

Once the group agreed on the conceptual architecture, the next step was division of labour.[2] The team members could easily identify where they could best fit. The task that needed more work was at the sensing layer, since besides establishing the network and efficiently collecting data, there was also a management task. Two people volunteered to work together (and the others agreed), one being responsible for establishing the network and defining various commands that would enable dynamic network management, and the other being responsible for implementing the commands and designing the management interface that would enable both an experienced programmer and a lay administrator to configure the network. The remaining three students decided to work in the remaining layers.

Figure 2 The conceptual architecture of the Senceive system



5 Refining the architecture

In the following weeks, students focused mainly on researching related work; acquainting themselves with existing technologies and refining the layer of the conceptual architecture for which they were responsible.

5.1 The sensing layer

The sensing layer supports two specific tasks: establishing the wireless sensor network for collecting raw data from various places and managing the network to accommodate multiple applications and dynamic network (re)configuration. The student who was responsible for establishing the network with Micaz and Mica2 sensor nodes (Section 2), identified three essential tasks: interpretation, collection and configuration. For each of these tasks, he defined a corresponding component and a set of interfaces. Moreover, he defined ten basic control commands that would enable users and administrators to interact with these components and thereby, with the sensor network. Table 3 displays the commands and their specific purpose.

The student who was responsible for designing the network management subsystem identified three main components that would permit flexible operation and management. These are the query/subscription service, the lower-level configuration service and the higher-level configuration service. The detailed description of the management subsystem is displayed in Figure 3.

5.1.1 The query/subscription processing service

The query/subscription processing service enables applications (according to the conceptual architecture, multiple modelling services) to access sensor data declaratively. The premise for the existence of this service is that whether the network is setup with a single application in mind or not, most existing applications extract sensor data from the network and perform data processing elsewhere. For example, structural health monitoring (Chintalapudi et al., 2006) and active volcano monitoring (Werner-Allen et al., 2006) applications collect raw data from the sensor network, but feature extraction takes place with the support of resource-rich computers outside of the network. Likewise, existing sensor nodes do not support higher-level digital signal processing such as extraction of Mel Frequency Cepstral Coefficients (Dargie and Tersch, 2008) from an audio signal at the local level. Therefore, applications can declaratively express interest by specifying the duration and sampling frequency of data that should be collected by the network.

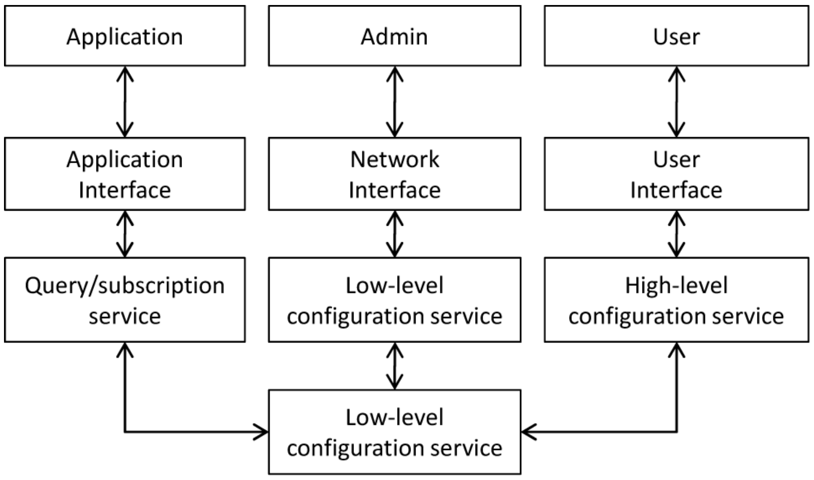
More specifically, the query/subscription processing service provides the following functionalities:

- process snapshot queries and historical queries
- start long-run queries with or without data listener
- provide information about available sensors in the network along with their present internal configurations
- register listener to process relevant events (changes in network status).

Table 3 Control commands for interacting and configuring the wireless sensor network

Command	*Purpose*
Set location	Sets the location lable for the selected node
Delete history data	Empties local storage
Auto config (AC) on	Switch on/off global automatic node configuration
AC mic gain	Sets microphone gain (0–255)
AC route update	Sets route update interval
AC storage mode	Sets storage mode value (8 or 16)
AC LPL cycle	Sets low-power listening duty cycle
Status check interval	Sets node status request interval
Node active timeout	Sets the sleeping duration of a node
Memory auto-DL	Sets downloading interval of stored

Figure 3 The conceptual architecture of the Senceive middleware


Application
Admin
User
Application Interface
Network Interface
User Interface
Query/subscription service
Low-level configuration service
High-level configuration service
Low-level configuration service

The query processing service automatically combines and optimises all active queries. If a query cannot be started due to hardware or query combination constraints, the query start method throws a *QueryException* containing a comment describing the reasons in detail. Generally, queries are processed in the same way queries are processed in Cougar (Woo et al., 2004). The middleware assigns an ID to every query. For storage queries, which do not need to specify a data listener, the user lable is part of the ID. This enables the administrator to require from or inform the query owner if it is necessary to stop the query before the specified end time.

```
Longrun Query ID: <IP>\_QueryListenerHash
Storage Query ID: <UserLabel>\_QueryListenerHash
```

The nodes themself are not able to deal with more than one long-run query at once. But the middleware combines all queries of a node to generate a union query.

5.1.2 The higher-level configuration service

The higher-level configuration service enables centralised control of the network. A centralised control ensures that the user's policy is enforced and the integrity of the network is maintained, and that only eligible applications are accessing the network. It enables also an administrator to monitor the status of individual nodes. If the network supports multiple applications, there can be some potential conflicts, for example, if three applications put an alert (alert-if-below) subscription request, in which case, the applications are interested to be notified when the temperature of a certain region falls below 12°C, 15°C and 17°C, respectively. Because an 'alert-if-below' request is a simple request, nodes can process such a subscription locally. However, a simple node may process only a few of these requests because of local memory constraints.[3] In this case, the higher-level configuration service decides which of these thresholds should be evaluated locally and which of them centrally so that the overall data traffic that results due to these requests is minimised.

More specifically, the higher-level configuration services provide the following functionalities:

- provides detailed network status information
- provides information about configuration aspects
- modifies individual node configuration
- modifies global configuration
- dynamically integrates and configures new nodes and updates corresponding routing and medium access policies.

5.1.3 The lower-level configuration service

In Section 2, we discussed some of the reasons why we need to reconfigure an already deployed network. The higher-level configuration discussed above is usually referred to as soft-configuration, since it does not affect the runtime code of a node. There are, however, conditions in which one needs to replace a module (for example, replacing a low-pass filter with a bandpass filter) or even the entire set of modules, as in the case of a considerable fault in the deployed code or a total change in the user's requirement.

Typical tasks of a lower-level (re)configuration task is to make sure that a programme code is propagated successfully and the new set of modules are installed in a consistent manner. If some nodes have installed the code and others are still running old modules, there will be a significant conflict. For this reason, the lower-level configuration services should be able to monitor consistency of modules and successful code propagation and installation.

5.1.4 The kernel

The kernel interfaces the higher-level services with the wireless sensor network. Its basic purpose is to transform the higher-level syntax into lower-level syntax that can be processed and executed by the network elements. Moreover, it plays a vital role when an in-network decision is made based on global knowledge. For example, it stores global variables such as the number of cluster heads allowed to form a hierarchical topology.

5.2 Modelling, reasoning and application

The three upper layers are very closely related because the modelling and reasoning processes are application specific. Hence, the first important decision that was made by the three responsible students was to identify the sensors which would deliver meaningful data.[4] The student responsible for the modeling layer identified two important components that were useful to storing raw sensor data (a distributed database) and the higher-level features and their interpretation (a knowledge base). Furthermore, he defined the way to analyse the statistical significance of the data in the database and defined how to save interpretations in the knowledge base.

It was decided by the entire team and their supervisor to use fuzzy logic for classifying the stochastic and time domain features. So, the task of the student who was responsible for the reasoning layer was to define fuzzy sets and membership functions. Therefore, the task of this student was more of analysis than software design.

The student who was responsible for the application layer was responsible to design a graphical user interface and to interface the application with the reasoning layer. The graphical user interface should enable users to declaratively put requests to the sensing layer or the reasoning layer so that they can observe the change in the measurement data or simply obtain higher-level activities from the reasoning layer.

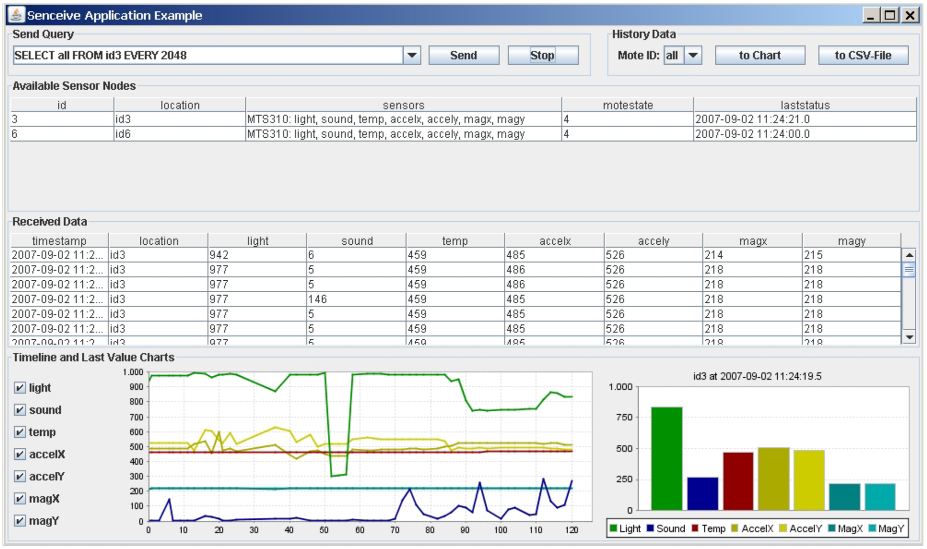
6 Implementation and evaluation

The Senceive team presented the conceptual architecture and the refined version thereof to the whole project groups and received constructive feedback. Afterwards, they proceeded with the implementation plan. The sensor network was set-up based on the TinyOS 2.x runtime environment. The network was made up of Crossbow MICAz nodes with MTS300 and MTS310 sensor boards. The sensing subsystem supported single-sensor access, stream sampling, resource arbitration and power management. Newly arriving nodes could autonomously register with the network, synchronise to global network time, receive commands to alter configuration or start sensing tasks and reliably deliver data to a sink.

The management subsystem provided external control of and dynamic binding to the network. Remote applications could access the network and collect data using the Java Remote Method Invocation (RMI). The query/subscription processing service made available a complete description of the services that were supported by the sensing subsystem, including the type of query and subscription requests that could be processed. Likewise, the higher-level configuration service provided network administrators with information pertaining to the number of available nodes and their spatial distribution as well as the configuration functionalities. Figure 4 shows the administrator's view of the Senceive system.

The wireless sensor network supported multi-hop communications; the minimum distance for a single-hop communication was determined by the service quality (tolerable end-to-end delay and packet loss) that was defined by the administrator. The received signal strength was used to set a threshold for multi-hop communication. For data gathering and command dissemination (higher-level configuration), the data collection and dissemination protocols introduced by sdlib (Chu et al., 2006) was adopted.

Figure 4 An overview of the Senceive configuration service administration interfaces (see online version for colours)



SELECT all FROM id3 EVERY 2048

Available Sensor Nodes

id	location	sensors	motestate	laststatus
3	id3	MTS310: light, sound, temp, accelx, accely, magx, magy	4	2007-09-02 11:24:21.0
6	id6	MTS310: light, sound, temp, accelx, accely, magx, magy	4	2007-09-02 11:24:00.0

Received Data

timestamp	location	light	sound	temp	accelx	accely	magx	magy
2007-09-02 11:2...	id3	942	6	459	485	526	214	215
2007-09-02 11:2...	id3	977	5	459	486	526	218	218
2007-09-02 11:2...	id3	977	5	459	486	526	218	218
2007-09-02 11:2...	id3	977	146	459	485	526	218	218
2007-09-02 11:2...	id3	977	5	459	485	526	218	218
2007-09-02 11:2...	id3	977	5	459	485	526	218	218
2007-09-02 11:2	id3	977	5	459	485	526	218	218

6.1 Data collection

The Senceive group performed also an experiment with regards to the impact of adopting a layered architecture of the timeliness (latency) of the recognised activity (context). A time diffusion Su and Akyildiz (2005) protocol was used to calculate the time needed to collect data from any node within the network to a remote base station that interfaced the network with a laptop computer. The experiment was conducted, thus, students distributed several MICAz nodes in different rooms, at the faculty of computer science in such a way that the node depth increased with every node. Whereas some nodes directly communicated with the base station, other nodes used intermediate nodes, based on the local decision regarding the signal strength. This ensured the establishment of a link with a reliable quality. The average collection time was about 10 ms for nodes within a one-hop range; 14 ms for nodes within two-hop range; 20 ms for nodes within three-hop range and 46 ms for nodes within four-hop range. The data collection time fluctuated; the worst being 24 ms for nodes within one-hop range; 25 ms for nodes within two-hop range; 45 ms for nodes within three-hop range and 64 ms for nodes within four-hop range. An additional test based on multiple long-run queries on the nodes resulted in the same collection time.

The results led to the expected conclusion that collection time is most significantly affected not due to the higher-level services but due to an increase in node depth. Only nodes within two-hop range showed unexpected behaviour, usually needing a shorter time to deliver message to the base station. A probable cause for this is the time synchronisation. As the implemented time synchronisation protocol does only provide synchrony with a resolution of around 20 ms, real-collection times could be higher. The above results can nevertheless be regarded as proof of reliability. All node messages reach the base station with a relatively short delay.

6.2 Command dissemination

Dynamic configuration of the network was supported by defining several simple commands that could directly be executed inside a node. These commands related to measurement thresholds, transmission power levels, low-power listening status, buffer size, acoustic gains, etc. By aggregating these commands, control of the behaviour as well as performance of the network can be enhanced. The basic challenge was to successfully (and timely) propagating these commands at runtime.

The test environment for the collection performance was also used to test the reliability of command dissemination. Unfortunately the dissemination protocol could not offer any meta information about routing status as the collection protocol did. Thus, it was hard to account lost packets and how many times they were retransmitted. As a result, it was not possible to determine whether the delay in command dissemination was due to retransmission of lost packets or due to the system complexity.

With regards to the impact of multi-hop communication on command dissemination, the experiment showed an expected behaviour, nevertheless. Nodes with a node depth of 1 always received commands with a negligible delay and responded promptly. With a higher node depth, however, the response time increased. During testing, nodes within one-hop always responded directly, while nodes within two and three hops away usually responded with a 0–2 s delay and nodes within four hops away usually needed 1–2 s to respond. In some cases, the response time was significantly higher, reaching up to 4 s. The positive result of the test was that no command was lost.

The results showed that the dissemination protocol reliably delivered commands to the nodes as long as time span between sending two commands was large enough. The protocol did not guarantee, however, a low-delivery time if multi-hop command delivery was necessary. As the number hops increased, the command delivery time reached up to the factor of ca. 100 times slower than the data collection time.

The network management subsystem was implemented in Java and so was the reasoning subsystem. mySQL was used to implement the knowledge base and the database.

6.3 Discussion

During the modelling process, frequency domain audio features (which were very useful for the recognition task) could not be extracted as expected. To recognise human speeches or speech-related activities, the surrounding acoustic should be sampled at a frequency of at least 8,000 Hz. This is because, the human speech lies in the frequency range of 0–4 kHz. This much sampling could not be supported both by the available memory size of individual nodes and by the TinyOS runtime environment. Therefore, the modelling process of acoustic data was limited to time domain and stochastic features.

Senceive was implemented and tested according to its initial description. The students successfully delivered the prototype and the documentation on time. This same system was later used in two third-party projects with minor modifications: in one of them, the simple fuzzy logic recognition was replaced by a more advanced scheme that was based on the hidden Markov models, and the wireless sensor network was complemented by wired, high-sensitive microphones to capture and process acoustic signals. In the other project, additional contexts for a train setting were introduced to determine if seats in a carriage were occupied or free; and if windows and light systems were properly functioning.

The Senceive system proved to be relatively simple to set up and to configure for both settings. Users with little experience in wireless sensor networks could be able to operate and monitor it. Perhaps, the simplicity and easiness in supporting multiple applications can be explained by the software complexity in the middle layers and in the management subsystem. Both the query-processing service and the configuration services perform a significant computation to transform user friendly, higher-level commands and requests into lower-level system commands that can be processed by the wireless sensor nodes. Apparently, implementation of the two services was labour intensive. Such complexity can only be justified if indeed multiple applications can use the sensing system. It is also worth to note that the system is not energy efficient.

Recording audio data with Crossbows MTS310/300 is problematic. Several test implementations using the Crossbow driver or directly sampling the ADC cannot achieve a sample frequency above 4 kHz. This results in low-quality audio for signal processing. Another problem is that the amount of data that can be processed locally is very small. Furthermore, the 512 kB flash memory does not allow an audio stream recording longer than 30 s at sampling frequency of 16 kHz. One of the test implementations uses a two-stage buffer recording strategy – one of the buffers uses RAM to temporarily store sensed data while the other buffer is used to store data in flash memory. The 1,000 Byte RAM buffer fills within 250 ms at 4 kHz while writing to the 1,000 Byte flash memory needs 72 ms.

Delivering an instantaneous value of the sound level of any of the sensor nodes does not make sense because of the time-variant nature of sound signals and the randomness of the measurement. Subsequently, the sound sensor (mic) is the only sensor that does not deliver raw data to the base station. Instead, the middleware supports a local preprocessing.

The actual implementation does a stream sampling with the maximum achievable sampling frequency, that is, 4 kHz. With the default buffer size of 800 Bytes and a 16 bit/value sample resolution, this results in a sampling period of 100 ms. The sensors need a warm up time of 1,200 ms before a steady-state value of the ambient measurement reaches the value of 500. The value delivered to the base station represents the average noise level 100 ms duration. The implemented algorithm pre-process the samples and delivers to the base station using Equation (1) and (2). The factor 2 in Equation (2) is used to as a scale factor, which is specific for Micaz nodes.

$$N_{\text{sum}} = \sum \left(N_{\text{steady\_state}} - \text{ADCValue}[i]\right)^2 \tag{1}$$

where N_{sum} is the summation of the deviation of the steady-state noise level from the ADC output (individual samples).

$$N = 2 \times \sqrt{\left(\frac{N_{\text{sum}}}{n}\right)} \tag{2}$$

where n is the number of samples.

Local pre-processing using for noise level estimation is plausible but precise measurements would need additional application side calibration.

7 Lessons learned

Project works prepare graduate students not only to the real world outside of their universities, but also to their immediate concern – the writing of a meaningful graduating thesis. Because they have to work in teams and keep deadlines, the challenges help them learn to efficiently organise themselves, manage time and set priorities.

Several lessons were learned during the development of the Senceive system. Firstly, the performance or achievement of students depends on the standards set for them by their supervisors and by the clarity of the objectives of their task. This does not mean, however, that students' educational background does not play a role. On the contrary, this withstanding, a project work being a first step towards developing complex systems, most of the students lack confidence and are vague about their contribution. Setting a high standard enables them to discover their potential. Secondly, defining milestones was the crucial step towards supporting students to manage their time properly and measure progress objectively. Having said this, project works require flexibility. Everything may not be executed as expected because individual students approach the same problem in different ways. Here, the supervisor needs to demonstrate a good judgment and accommodate individual differences. Finally, the main objective of a project work should be less of developing an excellent, compact and efficient system than identifying and understanding the various components of a complex system and building individual subsystems that function in concert. Finding the right balance is very challenging because of the associated time and the implication on keeping deadlines.

References

Benbasat, A. and Paradiso, A.J. (2007) 'A framework for the automated generation of power-efficient classifiers for embedded sensor nodes', *SenSys '07: Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, New York, NY, USA: ACM, pp.219–232.

Chao, X., Dargie, W. and Guan, L. (2008) 'Energy model for h2s monitoring wireless sensor network', *CSE '08: Proceedings of the 2008 11th IEEE International Conference on Computational Science and Engineering*, Washington, DC, USA: IEEE Computer Society, pp.402–409.

Chintalapudi, K., Fu, T., Paek, J., Kothari, N., Rangwala, S., Caffrey, J., Govindan, R., Johnson, E. and Masri, S. (2006) 'Monitoring civil structures with a wireless sensor network', *IEEE Internet Computing*, Vol. 10, No. 2, pp.26–34.

Chu, D., Lin, K., Linares, A., Nguyen, G. and Hellerstein, J.M. (2006) Sdlib: a sensor network data and communications library for rapid and robust application development', *IPSN '06: Proceedings of the Fifth International Conference on Information Processing in Sensor Networks*, New York, NY, USA: ACM, pp.432–440.

Dargie, W., Chao, X. and Denko, M. (2009) 'Modelling the energy cost of a fully operational wireless sensor network', *Springer Journal of Telecommunication Systems*.

Dargie, W. and Tersch, T. (2008) 'Recognition of complex settings by aggregating atomic scenes', *IEEE Intelligent Systems*, Vol. 23, No. 5, pp.58–65.

Hermann, C. and Dargie, W. (2008) 'Senceive: a middleware for a wireless sensor network', *AINA '08: Proceedings of the 22nd International Conference on Advanced Information Networking and Applications (aina 2008)*, Washington, DC, USA: IEEE Computer Society, pp.612–619.

Intanagonwiwat, C., Govindan, R., Estrin, D., Heidemann, J. and Silva, F. (2003) 'Directed diffusion for wireless sensor networking', *IEEE/ACM Transactions Networking*, Vol. 11, No. 1, pp.2–16.

Kim, Y., Schmid, T., Charbiwala, Z.M., Friedman, J. and Srivastava, M.B. (2008) 'Nawms: nonintrusive autonomous water monitoring system', *SenSys '08: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, New York, NY, USA: ACM, pp.309–322.

Koh, B.H. and Dyke, S.J. (2007) 'Structural health monitoring for flexible bridge structures using correlation and sensitivity of modal data', *Computers and Structures*, Vol. 85, No. 3–4, pp.117–130.

Lorincz, K., Chen, B., Waterman, J., Werner-Allen, G. and Welsh, M. (2008) 'Resource aware programming in the pixie os', *SenSys '08: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, New York, NY, USA: ACM, pp.211–224.

Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R. and Anderson, J. (2002) 'Wireless sensor networks for habitat monitoring', *ACM International Workshop on Wireless Sensor Networks and Applications (WSNA 2002)*, pp.88–97.

Su, W. and Akyildiz, I.F. 2005 'Time-diffusion synchronization protocol for wireless sensor networks', *IEEE/ACM Transactions Networking*, Vol. 13, No. 2, pp.384–397, Available at: http://dx.doi.org/10.1109/TNET.2004.842228

Werner-Allen, G., Lorincz, K., Welsh, M., Marcillo, O., Johnson, J., Ruiz, M. and Lees, J. (2006) 'Deploying a wireless sensor network on an active volcano', *IEEE Internet Computing*, Vol. 10, No. 2, pp.18–25.

Woo, A., Madden, S. and Govindan, R. (2004) 'Networking support for query processing in sensor networks', *Communication on ACM*, Vol. 47, No. 6, pp.47–52, Available at: http://doi.acm.org/10.1145/990680.990706.

Yick, J., Mukherjee, B. and Ghosal, D. (2008) 'Wireless sensor network survey', *Computer Network*, Vol. 52, No. 12.

Notes

[1]The acronym stands for sensing and perceiving. A comprehensive treatment of the technical aspect of the Senceive system is given in Hermann and Dargie (2008).

[2]Note the discrepancy between the schedule specified in Table 1 and the way things progressed in reality. The schedule specified that division of labour should occur during the second week of the project, before the conception of the system architecture. This could not be done, however, before students acquired a better understanding of the entire system and before they are confident of their own contribution. This was discussed with the supervisor and an agreement was reached to exercise some flexibility as long as the milestones were reached at the appropriate time.

[3]Note that there are additional sensors and other resources in a single node and associated requests that should be processed locally as well.

[4]These were temperature, light and acoustic sensors.